BACHELOROPPGAVE:

EpilepsyApp - development of smart phone app for young people with therapy-resistent epilepsy

FORFATTERE:
Adiljan Abdurihim
Andrius Januska

DATO:
23.05.2010

Sammendrag av Bacheloroppgaven

Tittel:	EpilepsyApp - utvikling av app til smarttelefoner for bruk av mennesker med terapiresistent epilepsi.	Nr:
		Dato: 23.05.2010
Deltakere:	Adiljan Abdurihim	
	Andrius Januska	
Veiledere:	Simon McCallum	
Oppdragsgiver:	Høgskolen i Gjøvik	
Kontaktperson:	Randi Stokke, randi.stokke@hig.no, phone: +4761135342	
Stikkord	Norway, Norsk	

Antall sider: 165	Antall vedlegg:	Tilgjengelighet: Åpen

Kort beskrivelse av bacheloroppgaven:
EpilepsyApp er en mobil applikasjon utviklet for mennesker med epilepsy. Denne applikasjonen er en løsning for daglig utføring av forskjellige registreringer. Bruker kan registrere, lagre og eksportere data som gjelder epilepsi blant dem er å sette påminnelser for medisiner som skal tas. I tillegg er det mulighet å aktivere alarm som sender SMS til valgte kontakter med GPS lokasjon av bruker ved anfall. Kort tips om epilepsy er telgjengelig.

Applikasjonen er utviklet ved bruk av PhoneGap framework som muliggjør å tilpasse applikasjonen for ulike platformer som Android, iPhone, BlackBerry osv. På grunn av PhoneGap sine begrensninger og tidsrestriksjoner ikke alt funksjonalitet er tilpasset for flere platformer. Det vil si de følgende funksjoner - medisin påminnelse, anfall monitoring og advarsel av kontakter, er implementert bare for Android platformen.

Applikasjonen er utviklet ved bruk av følgende teknologier - JavaScript, HTML5 og CSS. Android spesifiske deler (plug-ins) er utviklet ved bruk av Java programmeringsspråk. I tillegg til det brukte vi MySQL database for lagring av tips og PHP og Ajax for kommunikasjon mellom mobilenheten og ekstern server for å laste ned tips. Alle dataene som brukeren registrerer blir lagret i lokal WEB SQL Database og SD minnekort på mobilenheten.

Summary of Graduate Project

Title:	EpilepsyApp - development of smart phone app for young people with therapy-resistent epilepsy	Nr:
		Date: 23.05.2010
Participants:	Adiljan Abdurihim Andrius Januska	
Supervisor:	Simon McCallum	
Employer:	Høgskolen i Gjøvik	
Contact person:	Randi Stokke, randi.stokke@hig.no, phone: +4761135342	
Keywords		
Pages: 165	Appendixes:	Availability: Open

Short description of the main project:
EpilepsyApp is a mobile phone application for people who have epilepsy. This appliacion helps people with epilepsy to record, collect and export data related to epilepsy, provides medicine reminder notifications, monitors and detects several types of seizures, warns users contact people about a seizure by sending out SMS with GPS location data, displays epilepsy related tips to user.

The application is developed using PhoneGap platform that enables to use the application on different mobile platforms like Android phones, iPhone, BlackBerry, Windows Phone etc. However, not all of the features of the application are made cross-platform during this bachelor thesis, because of PhoneGap limitations and time restrictions. That is to say, medicine reminder notification, seizure monitoring and contact people warning is implemented only for Android platform.

The application is basically developed using web technologies - JavaScript, HTML5 and CSS. Android specific parts (plug-ins) are developed using Java programming language. We use MySQL database for storing tips on a remote server and PHP and Ajax to serve tips from the database to a mobile device. All the data saved by a user is stored on a local Web SQL Database and SD card of the device.

Preface

We would like to thank everybody who has contributed to the development and testing of EpilepsyApp !

Special thanks to Simon McCallum, Randi Stokke, Patrick Bours, Muhammed Derawi and Jayson for the help and support to go further.

Thanks to Johan for active feedback and translation of EpilepsyApp into Dutch.

Thanks to Dilshad, Mavlan, Rahile and Camilla for the translation of EpilepsyApp into other languages.

And of course we thank every and each who has not been mentioned above for the support during development of EpilepsyApp .

Contents

List of Figures

List of Tables

1 Introduction

1.1 Document Structure

The project report is divided into following 6 main chapters:

Introduction. In the first chapter we introduce reader with the background of this project, objectives that we are going to achieve with this project and what are restrictions we are set into. We shortly describe the academic background of team members and explain the development methodology that has been used for this project.

Requirement Specification. The second chapter contains description of functional and supplementary requirements and constraints for the application.

Application Design. The third chapter describes the architectural and design decisions and other challenges we met during the design phase of the project.

Implementation. The fourth chapter contains implementation descriptions and examples. Here we describe how we implemented specific architectural challenges by providing code snippets for interesting solutions we made, graphs for collected accelerometer data and screen-shots for graphical user interface illustration.

Testing. The fifth chapter describes the strategy and implementation for our application testing.

Conclusion. In chapter number six we discuss the results we achieved during this project and provide the conclusions for the results.

In addition to the main chapters we have 7 **Appendixes** that contain additional information for the project report.

1.2 Project Background

About 0,5% of the population in Norway have epilepsy. About five hundred children in Norway every year get epilepsy symptoms. According to the statistics, 70% of people who have epilepsy are getting better after using one or two different medicines. The remaining 30% have no possibility for the treatment and suffers from cognitive disorders which need daily observation, registration and notification. Lack of registration becomes a significant con in controlling attacks for people with epilepsy. All these factors lead to social isolation and doubt, because of cognitive disorder. Youth with epilepsy who are cared by their parents face significant problems when they move out. Major problem is that every individual with epilepsy has problem in managing their daily life, that includes all necessary registrations, planning and arranging [1].

The popularity of smart phones and their functionality has increased significantly the last few years. Affordable prices made smart phones common item in a daily life for

many people, especially youth. Smart phone became an inseparable accessory and tool for majority, helping to perform different kinds of tasks using mobile applications.

1.3 Target Audience

The main target group for the application that will be developed during this project are people from the whole world who have epilepsy and need a careful planning of their daily activities and regular use of medication.

Another target group for the application are epilepsy associations that would be interested in contributing or cooperating in some way, to provide an even better application.

The target audience for the report of this project are the examiner, people who are interested in system development, mobile applications development or in further development of EpilepsyApp.

1.4 Project Objectives

During this bachelor thesis we are going to achieve several objectives. We have divided the objectives into three groups: result, effect and learning objectives.

Result Objectives

The main goal with this project is to develop a fully functional and reliable mobile phone application, that would be used by people who have epilepsy and help them to manage their daily lives. Since there are many types of phones available on the market, it is very important to make this solution available for as many platforms as possible. This is a secondary goal of this project.

Effect Objectives

Often reasons for getting epileptic seizures are forgetting to take medicines, disordered way of life, unbalanced use of alcohol etc. Therefore it is very important for people who have epilepsy to get reminders to take medicines, and to be able to register various daily activities they perform, medicines they take, seizures they get and later to be able to analyze the collected data and see relations between them. Mobile phone application that will be developed during this project should help people to perform mentioned tasks in an easy way and thus to contribute to better control of their epilepsy.

Living with epilepsy introduces less security and self confidence for a person and much concerns for persons family members. Another objective for this project is to develop a solution that would provide the person and her family members with more confidence and less concerns when they are not together. Possibility to be able to immediately warn her family members or other close people when a seizure occurs or is likely to occur would give the person a better confidence. At the same time, being able to know persons location when a seizure occurs, would provide less concerns for both parts.

Today there are some types of instruments that provide with one or another function

that is needed for people with epilepsy. The application should be an alternative instrument for existing ones combining several functions in one device. Since smart phones has become so popular and many people have (or will have in the nearest future) some kind of smart phone, the application should be a cheaper and more comfortable alternative to existing ones.

This application will make a contribution for further research done by Randi Stokke, who analyzes how smart phones can increase the quality of life for young people with epilepsy.

Learning Objectives

By working with this application, the members of the project are seeking to learn the development of cross-platform mobile device applications, use various mobile devices' sensors, integrate applications into other systems and system development in a structured and methodological way.

1.5 Project Description

This project was suggested by Randi Stokke from Health department at Gjøvik University College in October of 2011 in relation with Mobile System Programming and Object Oriented System Development course at the school. Both of the bachelor thesis group members together with other two students contributed to the initial planning and development of the application prototype. However, the initial planning and development was done for Android based devices only. This project took a step forward when it was suggested as a bachelor thesis project later. The project has been chosen by current group members and the idea of cross-platform mobile phone application arose.

During this project we will develop a cross-platform mobile phone application that will help people with epilepsy to better control their epilepsy by recording various daily activities, seizures, medicine usage etc. in an easy and efficient way, provide reminders for taking medicines and possibility to warn person's relatives when a seizure occurs or is likely to occur.

The project scope is divided into two major parts: the features that must be done to provide a fully functional and useful application that meets requirements from the employer and the features that would extend the functionality of the application and make it even more useful, but that are not strict requirements from the employer. The scope is divided into these two groups of features:

1. Major features:
   - diary with possibility to register seizures, taken medicines, moods, menstruation cycle and additional notes;
   - exporting data in graphical presentation to a local PC;

- reminders for taking medicines;
- daily epilepsy related tips on the welcome screen.
- warning contact people;

2. Additional features:
   - sending stored data to an e-mail, preferably in PDF format;
   - seizure detection;
   - integration with Telenor's Shepherd platform;
   - medicines recognition;
   - simulation of P2P connection with external system.

The core functionality of the application will provide an easy to understand and comfortable way of registering seizures, medicines taken, moods and other relevant data.

Providing a graphical representation of collected data and possibility to export it to a file and save it on a local PC will help person to analyze the data in an easier way, and provide a possibility to present the data to health personnel when needed.

One of the most important features of the application is medicine reminder. It will let a person to remember to take medicines on time, as it is very important for epilepsy treatment.

Daily epilepsy tips is another useful function of the application. It will provide a person who has epilepsy with useful information and contribute to better control of his/her epilepsy.

Warning contact people of a person with epilepsy gives better self confidence for a person and less concerns for family members or other close people.

The additional features of the application were suggested for this project in cooperation with project supervisor Simon McCallum, project initiator Randi Stokke and the group members. Those features however are less prioritized and will only be developed in case we meet project deadlines, as they are not necessary for the core application to function.

Appraisal

There are, however, some features that will not be able to implement during this bachelor project due to time limit and restrictions.

The application that will be developed is intended to be a multi-platform application, to reach as many users as possible. However, due to time restrictions, two main platforms, Android and iOS, are prioritized, since they are most popular among users.

Communication with doctors (health systems) needs a high level of security. Due to Norwegian legislation, the process to get permission to implement communication with

doctors (health systems) would take too long time to meet the project deadline. Because of this, it was decided not to implement this feature in this bachelor project.

The major features are most important to develop for a fully functional application, so project group will concentrate on implementing them first. Consequently, some of the additional features may remain unimplemented. However, the project group predicts to implement at least two of the additional features.

1.6 Project Restrictions

Because the application is going to collect health related data, data storage and transfer is highly regulated by Norwegian legislation. The group will be restricted to store and transfer data according to Datatilsynnet's provisions.

Test users of the application have to be chosen randomly, without knowing if they have epilepsy or not. That will give a more predictive analysis of the data.

It also comes a restriction on the technology that can be used to develop applications for multiple platforms. We are restricted to use a technology that is suitable for creating multi-platform applications for mobile devices.

Due to limited resources, the application will only be tested on a few types of smart phones. Testing on other types of phones is dependent on what types of smart phones the testers of the application will use.

1.7 Academic Background

Our bachelor team has same experience and competence. Both of us has learned programming in C++, Java, JavaScript, PHP and other scripting and markup languages. Both of us had mobile programming for Android as an extra course. Having all what we have learned by now in Gjøvik University College it will be first big practical development so far. Although we have chosen Phonegap cross platform for coding that uses HTML, JavaScript and CSS, the native language for the Android development will be used. So it is a good practice of all languages.

Basics and all the extras that we have learned from University College will be a helpful kit in future development. The knowledge about software development methodology will help us quickly decide what kind of development method is to choose from. We believe that all concepts we have will be used in our bachelor thesis.

Although we see some future concerns in some fields that are unknown for us, we believe that we will get enough assistance from tutors in Gjøvik University College. The right use of data from sensors and ability of analyzing biometric graphs are major and important issues where we will need assistance. Adapting knowledge in these fields will be necessary.

1.8 Framework

1.8.1 Software Development Methodology

It was important for us, two, to choose the most appropriate software development methodology. As none of us had tried at least one of methodology in practice we had begun with discussion of every followed by critical assumption. The discussion and statement according to facts opened eyes and gave more knowledge about every software development. Weighing out all the advantages and disadvantages, finally we have decided on incremental methodology. We believe that incremental methodology like Scrum would be a great solution for the development of mobile application. Different levels and extra futures that could consume a lot of resources and time became a reasonable factor for choosing incremental and iterative software development methodology - Scrum. We admit that the popularity of the methodology became an extra overwhelming factor for our choice.

Working with the methodology setting test term has shown that the group was able to adapt it. The group realized that including of pair-programming in some of the futures that needed more time was desirable. That's why we have decided that XP methodology will be used in some part of the project development.

When time came to deliver the pre-project plan we have realized that a group of two people was too little to organize all the roles. It could be unfair giving one member decision priority if some issue arise for the voting. Therefore it was decided that each member will play the role of Scrum Master every sprint term.

The draft of product backlog according to the project description written by the Product Owner has been discussed. The suggestion for extra features from both parts has been given. The Product Backlog has been discussed with attendance of supervisor in order to have a realistic insight of development (see Table 1). After prioritizing and dividing features into different groups we started our development.

1.8.2 Implementation of Scrum

The implementation of software development methodology that we have chosen became little bit complicated due to the number of developers in the group. The traditional daily scrum meeting has been slightly changed as we implemented written daily log. Every member had to check Progress Log for the check of actual development.

At the beginning of every week we have a meeting with a supervisor where we discuss status report submitted Thursday and where we plan the week. All the shared tasks must be written in daily log using google document .xls that shows tasks to accomplish and accomplished. As we have been working together for about two and half year it was easy to follow the plan and write down daily logs. The Trello has been used for voting and following all the product backlogs development progress. Later on we added several

sections to Trello like bugs, meeting, deadlines etc for managing our work in an efficient way. These techniques were useful to use as our Supervisor was abroad with difference of 12 hours and most time we were on remote operation.

When we were up to the complicated part of code, we just sat together and tried to solve the problem. Here we have used XP, pair-programming. It was useful development methodology to solve such kind of problem when assistance needed or we stuck.

The regular meetings with the Product Owner at the end of every sprint term, two weeks, was a good stimulus for the progression. The demo will be presented and further decisions about new futures will be discussed at the meeting. We will try follow the traditional iteration of Scrum. Both parts, team and Product Owner were satisfied to follow selected software development methodology.

Of course the aim was to finish all features by the deadline. We wouldn't take chance and promise too much that's why features have been developed incrementally. First is to develop the main core of the application, secondary modification of the application and finally if we have enough time to develop at least two extra features. Such kind of separation of features to the group help us to minimize stress. At the time of development we realized that time to time we were behind the planned work process and later we have seen that development gone better. Frankly, we should admit that lack of experience lead to wrong time estimates.

1.8.3 Schedule

The schedule is an important part for the systematical work, that's why we spent a lot time for planning it and discussing what kind of extra tools should we use. The fundamental project follow was Gantt Chart diagram. This diagram shows the expected progress. The high level representations with some milestones is provided in a diagram in the pre-project report (see Appendix A). The scheduling plan was consisting of logging all data, analyzing and development. Time frame for improvements according to the feedback and suggestions has been set. As we couldn't gather any data from users who had seizure according to legacy it was too important to upload the beta version of the application as soon as it possible. The schedule based on iterations helped us a lot to determine the reality for developing features.

1.8.4 Project Organization

As the group was created and the Bachelor thesis has been chosen we started our organizational work.The assigned room from University for this project was shared by two groups. We have decided that the room will be used at least two days in a week, but welcomed for extra gatherings. The Monday and Thursday have been assigned as a office day where Mondays we were discussing feedback from status report submitted on Thursday together with our supervisor, Mr. Simon McCallum. All the plans for the sprint

term and weekly plan has been done the same day. Thursdays we were informing our supervisor about the progress sending him a status report. At the end of every sprint, on Thursday, we were presenting demo for the product owner.

We are agreed on working minimum 8 hours per day, four days in a week. We were aware, that having two people in the group, it was requiring more overtime. We set minimum in order to stabilize our work, over wise it didn't take a long time before we started with overtime. By the end of the sprints it was a usual factor when group members worked about six to seven days per week. It was much more excitement rather just accomplishment. It was very strict that we follow time log and work log that was including not just time consumed for the project, but they description of accomplishment The Trello was a useful tool to observe what each of the group member was working on. It was practical to post comments, bugs, suggestions and etc at Trello that was giving an automated notification for the members. This tool was very useful when we were working remotely.

The distribution of role as project manager was tricky as we were two people in the group. So, rollover of the role for the each Sprint term was decided. That gave us to experiment us as a project manager. The rules for making decision and voting was stated clearly in rules regulation written in both English and Norwegian languages.

In case if there are disagreements, we had to refer to the these rules, but we hadn't any argue in whole process of development. As a project contact we have assigned Adiljan Abdurihim and responsibility for the web page and blog was divided between members.

Mr. Simon McCallum was assigned as the supervisor for this project. He was actually in our wish list for the supervision. He leads the Game Technology Lab and other courses where among of them is Mobile System Programming. The background of supervisor was an advantage for us to ask him assistance as we had not a lot of experience about cross platform framework and other issues. Even though Mr. Simon McCallum was abroad for some period, we had enough assistance remotely through video conference. The product owner, Randi Stokke advised us a lot giving us information about Epilepsy, that we totally were imaging in a different way. She helped a lot to create different scenarios and on time feedback being available anytime. Having a good background both in technological part and health it was up to us to use these resources in right way. The meeting with each, supervisor and product owner, has been decided ahead with an ability to cancel if there is no necessity. That kind of approach helped us to progress in structural way. Although we had to cancel some meeting and rearranging meeting for the different day, we were satisfied.

2 Requirement Specification

The first step in the development of a new system is to clarify the system requirements. Since project initiator has only described the system very briefly, it is very improtant to find out all functional and operational requirements in order to develop the application as expected. We have divided requirement specificaiton into three parts: functional requirements, supplementary requirements and constraints.

2.1 Functional requirements

2.1.1 ProductBacklog

First of all, as we have chosen Scrum as a development methodology for this project, specifying a Product Backlog was needed. Since project initiator does not have any knowledge about system development and Scrum methodology, the Product Backlog was formed by joint efforts of Product Owner, project supervisor and project group members. The formed Product Backlog represents the high-level functional system requirements (see Table 1).

ID	Name	Importance	Initial estimate	Notes
1	Diary for registrations	100	30	Seizures, medicines taken, mood, comments and menstruations.
2	Medicine reminder	80	7	Do not stop until registration done.
3	Warning contact people	60	10	Include GPS location.
4	Exporting data to local PC	50	10	Graphics, text, backup data.
5	Daily epilepsy related tips	45	7	With possibility to update with new tips.
6	Seizure detection	40	20	Use accelerometer data.
7	Sending stored data to en email in PDF format	25	4	Clarify with Datatilsynnet about requirements.
8	Integration with Telenor's Shepherd platform	20	7	-
9	Medicine recognition	18	20	-

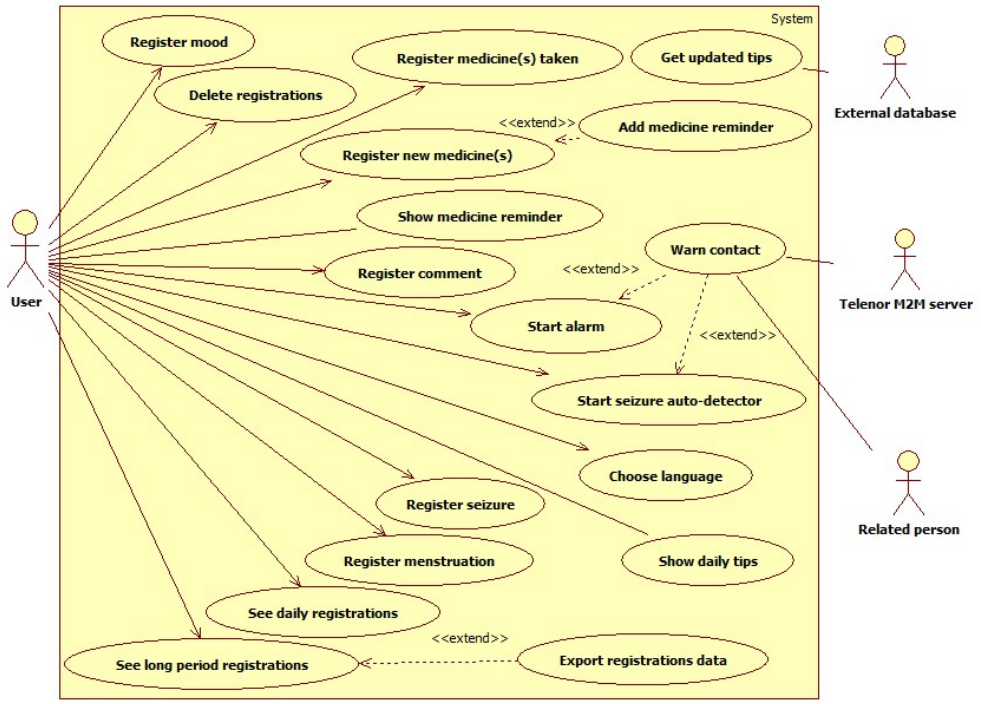
Table 1: Initial Product Backlog.

The initial product backlog was basically worked out from the product features that were supplied by project initiator Randi with some adjustments after discussing with project supervisor Simon and us. Later the importance points were assigned to each item in the Product Backlog and initial estimations done.

2.1.2 Use Case Diagram

Further, as the Product Backlog was formed, in order to better understand the interactions between users and the system, we used a Use Case methodology as suggested by R.V. Strumpf and L.C. Teague [2]. By using this methodology we defined system boundaries and identified all actors that will interact with the system. Use Cases, system boundaries, actors and their interactions are represented in a Use Case diagram (see Appendix B).

Figure 1: EpilepsyApp Use Case diagram.



Further follows the descriptions of the Use Cases represented in the Use Case diagram.

2.1.3 High-Level Use Cases

We used high-level Use Cases to present a sequence of internal actions by which the system responds to the actions of an actor, as suggested by R.V. Strumpf and L.C. Teague [2]. This type of Uses Cases let as describe the system actions when everything goes well. In this section we present Use Cases for more complex actor and system interacions. Full collection of Use Cases created during the process of requirement specification can be found in Apendix B.

ID:	2
Use Case:	Register medicine(s) taken
Actors:	User
Purpose:	To register if user has taken medicines or not.
Description:	User selects to register medicine(s) he has to take at a particular time. The list of medicines that user uses is displayed and user can choose one or more medicines and define if a particular medicine was taken or not. When user selects, he can save the registration into diary and medicine reminder will be cleared if it has been set. The registration is confirmed.

ID:	7
Use Case:	Register new medicine
Actors:	User
Purpose:	To register a new medicine that user will use constantly.
Description:	User requires to register a new medicine that he is going to use constantly. The system gives to alternatives: to select medicine from a predefined list or to enter own medicine name. User selects one of the two alternatives and is provided possiblity to enter a dose, time for usage and if reminder for the medicine usage should be activated or not. When user has filled out all required infromation he can save the new medicine. User gets confirmation about registered medicine.

ID:	9
Use Case:	Edit medicine details
Actors:	User
Purpose:	To edit the details for a particular medicine.
Description:	User requires to edit the details about a particular medicine. User first performs the steps from Use Case 8. User chooses to edit details about the selected medicine. User is provided with a possibility to change the dose, time and activate or deactivate the reminder. User makes changes, saves them and gets a confimation message.

ID:	13
Use Case:	Export registrations data
Actors:	User
Purpose:	To save the registered data from a mobile device to a local PC for a given period of time.
Description:	User requires to export the registrations data from his mobile device to his local PC for a particular period of time. User performs steps from Use Case 12. The user defines the period and selects to export the data. System saves registrations data onto users device both in graphical and text representation.

ID:	15
Use Case:	Start alarm
Actors:	User
Purpose:	To give user a possibility to inform his contact people about a possible seizure.
Description:	User activates the alarm. After a short period of time the system requires to confirm if the user wants to continue monitoring or to stop. If user chooses to stop the alarm, the alarm will be stopped, if not – it will continue to monitor and repeat the confirmation step in short intervals. If user does not respond to the confirmation message, the system will continue with User Case 17.

ID:	16
Use Case:	Start seizure auto-detector
Actors:	User
Purpose:	To start monitoring users movements in order to detect a seizure.
Description:	User requires the system to monitor for possible seizure. User selects to start seizure auto-detector. System starts collecting sensor infromation and performs seizure detection operations as long as user decides to stop the seizure monitoring or a seizure is detected. If a seizure is detected, system continues with Use Case 17.

ID:	18
Use Case:	Show medicine reminder
Actors:	User
Purpose:	To remind user about taking medicine at a predefined time.
Description:	System is triggered to show the reminder notification message at a particular time of a day predefined by user. If user responds to the reminder and registers medicine usage, the reminder is cleared. If user does not respond to the reminder, reminder repeats at 5 mins period.

As we can see from the listed Use Cases they give us a better understanding of what actions user can perform and how system is supposed to respond to users actions. However, such type of Use Cases is not enaugh to fully understand all functional system requirements. In further step of system requirement specification we use Expanded Use Cases.

2.1.4 Expanded Use Cases

In order to fully clarify the functional system requirements for the most sophisticated user and system interactions we have chosen to use Expanded Use Case methodology. This type of Use Cases extend the Use Cases defined earlier in the process and takes into account alternative flows of events and error situations. In this section we have selected to represent three of them, that are the most complicated Use Cases of all others. All Expanded Use Cases that we have created during this project can be found in Apendix B.

ID:	7
Use Case:	**Register new medicine**
Actors:	User
Purpose:	To register a new medicine that a user will use constantly.
Overview:	User requires to register a new medicine that he is going to use constantly. He is provided with a possibility to choose medicine from a list or to enter own medicine name. After, user enters a dose, defines time for taking medicine and chooses if a medicine reminder should be enabled or not. When all the details about the medicine are entered user saves it, and the list of all registered medicines is displayed.
Preconditions:	-
Postconditions:	A new medicine will be added to the medicine list.
Special requirements:	-

Flow of events	
ACTOR ACTION	**SYSTEM RESPONSE**
1. This use case begins when a user wants to register a new medicine.	2. Provides a possibility either to choose medicine name from a list or to enter a medicine name.
3. User selects to choose medicine from the list provided.	4. Shows the list of known medicines.
5. User select a medicine name he wishes to add to the medicine list.	6. Provides a possibility to enter medicine details.
7. User enters the dose, sets time and selects to enable reminder for this medicine and confirms details.	8. Displays registered details for the medicine.
8. User saves the medicine.	9. Saves medicine and displays all medicines registered.

Alternative flow of events
Line 3: User chooses to enter own medicine name. Provides possibility for the user to enter medicine name. If user enters medicine name, system proceeds in step 6. If user does not enter medicine name returns to step 2.
Line 7: User enters wrong dose. Indicate error and return to step 6.
Line 7: User wants to enter additional time for taking medicines. Proceed by returning to step 6.

ID:	13
Use Case:	**Export registrations data**
Actors:	User
Purpose:	To provide user with possibility to save registrations data to his PC.
Overview:	User requires to save all registrations data to his local PC. He is provided with a possiblity to select a period of time for which data should be exported. User selects a period and registered data is shown in both graphical and text representation. User then saves data into phones memory card that is easily accesible from his PC.
Preconditions:	It must be registered some data. Mobile phone must have a memory card.
Postconditions:	User gets saved a file with graphical and text representation of data into his mobile devices memory card.
Special requirements:	Graphical representation of data should be clear and of good quality.

Flow of events

ACTOR ACTION	SYSTEM RESPONSE
1. This use case begins when a user wants to export registered data to phones memory card.	
2. User selects a period for which data should be exported.	3. Generates and displays diagram and text for the registered data.
4. User chooses to save data to memory card.	5. Generates a data file and saves it on moblie devices memory card in the applications data directory. Indicates that an export operation is being performed.
	6. When file is generated and saved on the devices memory card, displays a message confirming successful operation with details about a name of the file generated and its location.
7. User reads and closes the message and leaves current activity.	

Alternative flow of events

Line 3: There is no any data registered. Display an informative message. Return to step 2.
Line 5: There is no memory card in users mobile device. Display informative message. Return to step 4.

ID:	15
Use Case:	**Start alarm**
Actors:	User
Purpose:	To give user a possibility to inform his contact people about a possible seizure.
Overview:	User activates the alarm and the system periodically requires to confirm if the user wants to continue monitoring or to stop. If user chooses to stop the alarm, the alarm will be stopped, if not – it will continue to monitor and repeat the confirmation step in short intervals. If user does not respond to the confirmation message, the system will warn users contact people.
Preconditions:	GPS has to be enable if a user wants to send warning message with his location details.
Postconditions:	Users contact people will be warned about users seizure.
Special requirements:	User must be able to start the alarm imediately.

Flow of events

ACTOR ACTION	SYSTEM RESPONSE
1. This use case begins when a user wants to start the alarm function.	
2. User initiates the alarm function.	3. System checks if user has enabled GPS sensors and has selected any contacts for sending a warning message to. If GPS sensors are enabled and at least one contact is selected proceeds in step 4.
	4. After a short period of time system displays a message asking to confirm if user wants to continue monitoring or not, and plays a sound.
5. User does not respond to the message for 30 seconds.	6. Gets users location data.
	7. Sends a warning message to users pre-selected contacts.
	8. Starts making audible alerts to attract passengers' attention and displays the first-help infromation on the screen of the mobile device.

Alternative flow of events

Line 3: GPS is not enabled and/or no contacts are selected. Shows an alert message that GPS should be enabled and/or at least one contact selected.
Line 5: User responds to the message and selects to continue monitoring process. Return to step 4.
Line 5: User responds to the message and selects to stop the monitoring process. The monitoring stops.
Line 6: GPS is not enabled. Proceed in step 7.
Line 7: User has not defined any contacts. Proceed in step 8.
Line 7: User has no money in his account to send a message. Proceed in step 8.
Line 7: Users mobile device is out of mobile signal coverage. Proceed in step 8.

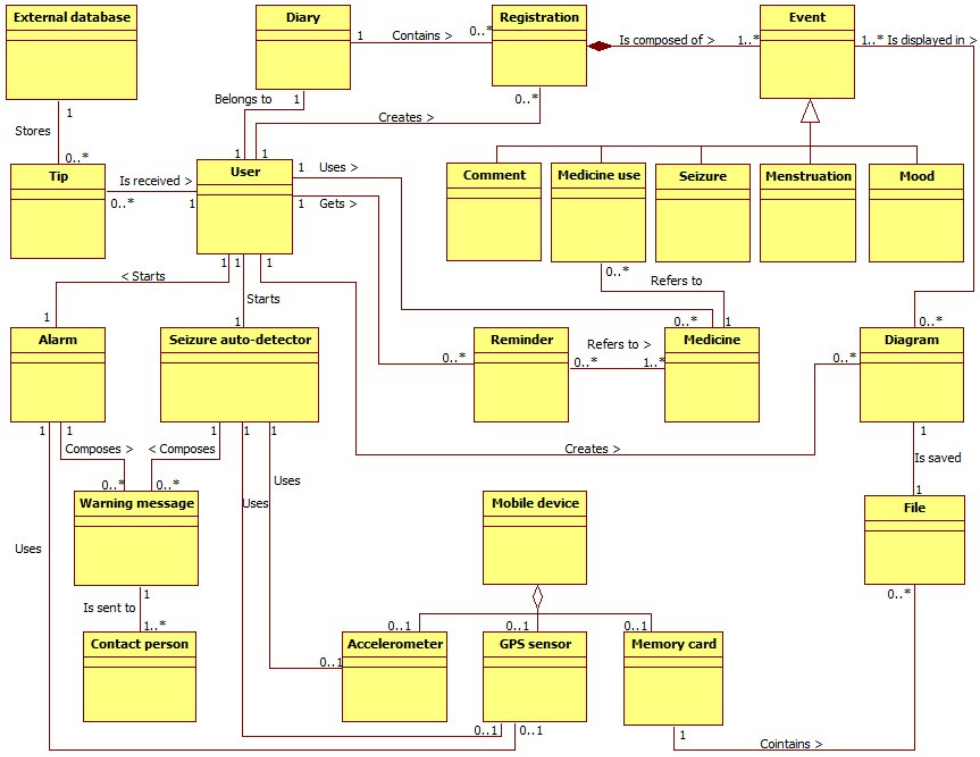
By using Expanded Use Cases we have now clearly defined how system should respond when user selects alternative interactions with a system and how system should react when error situations arise. This will help us in designing the application later in the development process.

When we have clarified functional system requirements we would like to see how the things belong together. We use Domain Model approach to find theese relations.

2.1.5 DomainModel

In this section we represent the Domain Model of the application. This helps us to model the concepts in our projects problem domain. It shows all concepts that are related to the context of the application and their relationships (see Figure 2).

Figure 2: EpilepsyApp Domain Model.



We can see from the domain model, that *User* is the "main" concept in our application, as it has most connections with other concepts. User is an initiator of many interactions or receives the results from the interactions. User can create diary, register new medicines, create registrations in the diary, get medicine reminders, create diagrams of registered events, start alarm and seizure auto-detector and receive tips from the databse.

Another important concept in the application is *Diary*. It cointains all the registrations created by user. *Registration* can be composed of one or more events, and one *Event* is either a *Seizure*, a *Comment*, a *Medicine use*, a *Menstruation* or a *Mood*.

Later all registered events can be used to draw a *Gaph* that is saved to a *File* on the mobile devices *Memory card*.

As user registers a new *Medicine* it can be assigned a *Reminder* that will be received by a user. Medicine also refers to the medicine use concept, when user registers it as taken or not taken.

Alarm and *Seizure auto-detector* are both initiated by a user and they both can result into a *Warning message* that will be received by a *Contact person* of the user. They also use an *Accelerometer* and a *GPS sensor* of a *Mobile device*.

Every *Tip* that is displayed to the user comes from the *External database*, before it is saved locally on a mobile device.

The Domain Model analysis helps us to see the whole picture of the application that we will develop. Such understanding will be important when we start to create an architecture of the application.

2.2 Supplementary requirements

In addition to the functional requirements the application has a set of supplementary requirements that we have to take into account in the development process. To specify supplementary requirements we used RUP's approach FURPS+ [3] and have included those sections that are relevant for our project. This section lists all additional requirements for the application.

2.2.1 Functionality

We have defined many functional requirements for the application using Use Case methodology. Theese requirements are more each Use Case specific. In this subsection we define system-wide functional requirements.

User tutorial

When user starts to use application for the first time, it is important for him to know how application works and what he can do with the application. A short and clear tutorial will provide with explanations of using each feature of the application. The tutorial has to be easy to find and represent the list of all features that are explained, so the user can easily choose the section of interest.

Data backup

The application will store data that is very important to user. That implies that having a backup copy of data is needed. User should be able to save all the registered data outside the application and export it to an other device when needed. However, data backup should be initiated by user, not automatically.

Security

The application will store some health related data. According to Norwegian legislation, health related data has to be properly secured. That applies specially for the data that will be transmitted outside the application. In order to clarify what security requirements applies to our application, we have contacted the Datatilsynnet for assessment. We got a list of aspects that we should take into consideration when developing application that handles medical information. However, according to the Datatilsynnet, since this application is meant to be only for private use, and will not communicate to any professional institutions, these aspects are more of advisory type than strict restrictions. The communication between us and the Datatilsynnet is documented in Apendix C.

Some of the aspects that we should take into consideration in this project:

- Application should support high security by default, but let user to decide the level of security.
- Data saved by the application should be encrypted.
- Application should use own PIN code to access it, or use the phone-lock to prevent undesired access.
- User should be able to delete the historical data saved by the application.
- Registered data should be deleted when the application is being uninstalled, or user should clearly know where these data are.
- It should be possible to delete data remotely if f. ex. the mobile device has been stolen.
- It is preferable for application to use its own calendar and reminder functions.
- Sending of e-mail messages should be encrypted or use a password to open a file.

2.2.2 Usability

For a mobile application to be successful and acceptable for users it must have a good usability. There are several things to consider when creating this application: accessibility, user interface aesthetics and consistency within the user interface, understandability and more.

First of all, the application must be easy to understand and learn to use it. It should not take long time for a person who knowns smart phones how to use EpilepsyApp. Use of icons together with text labels is preferable for application controls.

Easy navigation and clear layouts is another thing to have in mind when creating user interfaces for the application. User should always be able to navigate back to where he came from or to go to a start position of the application. It should be possible to do

registrations in as few clicks as possible. Alarm function should be accessible from the start point of the application. It is very important for a user to be able to use this function immediately when it is needed.

Layouts of the application elements should be clear and easy to understand, size of elements must be not too small and comply with the size of mobile devices screen.

Application should also respond to all user interactions with it. It has to respond with a visible feedback to each users click on the interactive elements of the application. User must get an informational message when something goes wrong. It should also give a feedback when registrations and operations that take long time are completed. Operations that take a long time should also provide a visual feedback about that something is going on and application is still operating.

2.2.3 Reliability

There are few aspects that are very important considering the reliability of EpilepsyApp. Since the application will store registrations related to medical and health information and will be used as a warning tool when an epileptic seizure occurs, the level of reliability of the application has to be very high.

First of all, the reliability of the data storage has to be at the highest level. People who will use this application will record their seizures and medicine usage history and other information related to their epilepsy. Later this data will be analyzed and used to improve the control of seizures. Any kind of data loosing would distort the information and could lead to making wrong decisions. Because of this, reliability of the stored data has to be very high. The application has to provide user with a possibility to save a backup copy of all registrations made as described in previous section and assure that data will not be deleted accidently. A possibility to recover from a device fail, lost phone, migration to another phone etc. is a key here.

In addition, correctness of the data registered is also very important for later analysis, so the process of saving data has to assure that correct data is being stored. The application has to minimize the risk for a user to save incorrect data and provide a possibility to delete incorrect registrations.

Another reliability issue is related to the alarm and seizure auto-detect functions. Because these functions will be used in critical situations, when a seizure occurs or is likely to occur, it is very important that these functions are available all the time. If the application fails to perform these functions it must inform user about it, so he can use alternative ways to warn his related people.

And finally, a medicine reminder has to be reliable as it performs an important function in helping user to remember to take medicines on time and register their usage. Fail of medicine reminder will not have such serious consequences as in previous cases, but it should be guarantied that user gets reminders for all medicines at a right time.

2.2.4 Performance

The highest requirement for the applications performance refers to the battery usage. The application should be able to perform its functions with minimal usage of battery power. This is specially applicable when application is in alarm and seizure-auto detect modes, as these functions will require extra battery power for sensors handling. The application should inform user if battery level is less than 15% when he tries to start seizure-autodetect function, as it could drain the battery fast and thus constraint functionality of the mobile device.

Other performance issues, like response to user interactions with the application should be at the similar level as it is normal for other good quality mobile applications. Two slow application would distract users from using it. So if the application is doing some harder work that is taking longer than 3 seconds it should immediately inform user about it, and let user to know that he has to wait a little.

2.2.5 Supportability

The application must have a user manual in it that would help users to find out how to use different functions correctly, as described in previous sections of this chapter.

The application should support multiple languages. Project initiator Randi had an initial requirement to make the application in Norwegian, because a similar application that was on the market did not have support for Norwegian. As we saw a potential for this application to be useful for a wider range of user audience we set a requirement for the application to support English and Norwegian languages at the initial point, with a possibility to increase a number of languages supported. As an additional requirement to this, the application should be able to detect user devices language automatically and apply that language to the application, if it supports that language.

The application should be maintainable in the future. That involves updating the collection of daily tips, medicine lists and other information that is relevant to be updated in the application.

2.2.6 Legal requirements

Because the application will handle medical and health-related information, it has to take into account laws and restrictions related to handling of such information as determined by Norwegian legislation. This issue is closely related to the security issues discussed in the section 2.2.1.

2.2.7 Licensing

2.2.8 Partial releases

At the end of every Sprint we should provide a functioning prototype of the application as required by Scrum development methodology. Such kind of iterative development and releases will help to control that the application is being developed according to

specifications and contribute to regular testing.

By the way, we have decided to release a beta version of EpilepsyApp to the Android Market after five two-weeks Sprints, on 26th of March 2012. The reason for this is to access a wider range of testers and test the application on as many different Android platforms as possible.

2.3 Constraints

2.3.1 Design constraints

EpilepsyApp is meant to be a cross-platform application. At the first step it should support Android and iOS based mobile devices as they comprise the largest share of smart phones market. This requirement restricts the technology to be chosen in the architecture planning and implementation phases.

2.3.2 Physical constraints

There are also constraints for what types of mobile devices the application should be developed. As we are targeted to develop the application for Android and iOS mobile devices, here comes the constraint to target devices that have accelerometer, GPS sensor and possibility to send SMS. However, this constraint only applies for some functions of the application - alarm and seizure auto-detect. The diary and all other functions related to it should be available on the devices that do not have mentioned features.

3 Application Design

We have used a number of different architectural views to depict different aspects of the system. In this section we represent the most architecturally important issues.

3.1 Architectural Goals and Constraints

3.1.1 Cross-platform Problem

We have a requirement to develop a cross-platform mobile application. This implies restrictions in choosing a technology for developing the application and impacts the architecture of the application. As we have a requirement to concentrate first on Android and iOS platforms we should find a solution that would apply for both platforms. Here we have too choices:

- Create two applications - one for each platform.
- Find and apply the technology that solves the cross-platform problem.

First approach does not really solve the cross-platform problem, because the product would be two different applications. There are both advantages and disadvantages to follow this strategy.

The advantages would be that the application would run on a native platform code, what means that no extra load to the application would be added. Application development technology for a particular platform is well documented and tested, it has best practices how to solve common application development problems.

However, there are pretty many disadvantages to choose this alternative. First of them - time costs. Because two applications need to be developed it would take longer time to learn two different development technologies for developing for each platform. It would also mean writing double as much code than as for one application. Second, extending to a third platform would mean creating a third application and so on. Third, the maintainability and supportability would be difficult and and time consuming, because of the same reasons.

The advantages and disadvantages with using a cross-platform development technology would be opposites as compared to using the first alternative. Solving the cross-platform problem would men that an additional layer has to be added to the application to unify the development interfaces for different platforms. This would add an additional load to the application and could possibly slow down the performance. The cross-platform mobile development approach is relatively new, what means that the technology is possibly not as well documented as in the first alternative. On the other hand,

the idea of “Create once and implement anywhere” looks much more reasonable, as it would mean less time costs for learning technology and development, better supportability and maintenance for the application and easier extension to a larger number of platforms.

Considering all the advantages and disadvantages discussed above and knowing that we have very limited time resources, we have decided to use the second approach as it seems a more reasonable solution for the projects problem.

3.1.2 Choosing Development Framework

Because a framework for cross-platform mobile application development could affect the overall architecture of the application we have decided to put this discussion before we start to create the application design and implement it. After searching the Internet for cross-platform mobile development frameworks we have found several of interest for us.

PhoneGap development framework is free mobile application development framework, based on web standards as HTML, CSS and JavaScript, that we are familiar with from before. It has a well documented JavaScript API, that allows to use native platforms features as accessing file system, use accelerometer and GPS sensors, camera directly from JavaScript code. PhoneGap has good support for our primary target platforms Android and iOS, also supports many other platforms - Windows Phone, BlackBerry, Symbian, webOS, Bada and more[4].

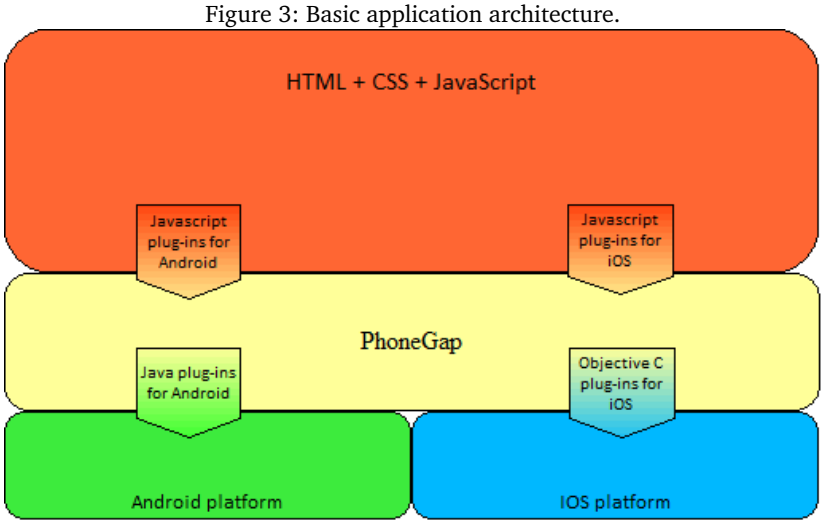
Considering all the features of PhoneGap we have decided that it fulfills all the requirements that we need to develop EpilepsyApp so we have chosen to use it as a development framework for our application.

3.2 Basic Architecture

As we have chosen PhoneGap as a framework for developing our application, we got some factors to take into account when considering the architecture for the application. First, the most part of the application will be developed using HTML, CSS and JavaScript. The native application code will only be used as a wrapper for launching the web browser with different HTML pages and running services. PhoneGap is a middle layer between the native level and web level of the application, that allows the application to run in the web browser and communicate with the native level. The basic architecture of EpilepsyApp is represented in Figure 3

In addition to PhoneGap’s built-in plug-ins for accessing native platforms features, we have possibility to write custom plug-ins for the platform features that are not implemented by PhoneGap (for example to run background services).

Figure 3: Basic application architecture.



3.3 Application Structure

This section represents the applications structure - application decomposition into layers and structural components (see Figure 4). It corresponds to the Implementation view of RUP methodology.

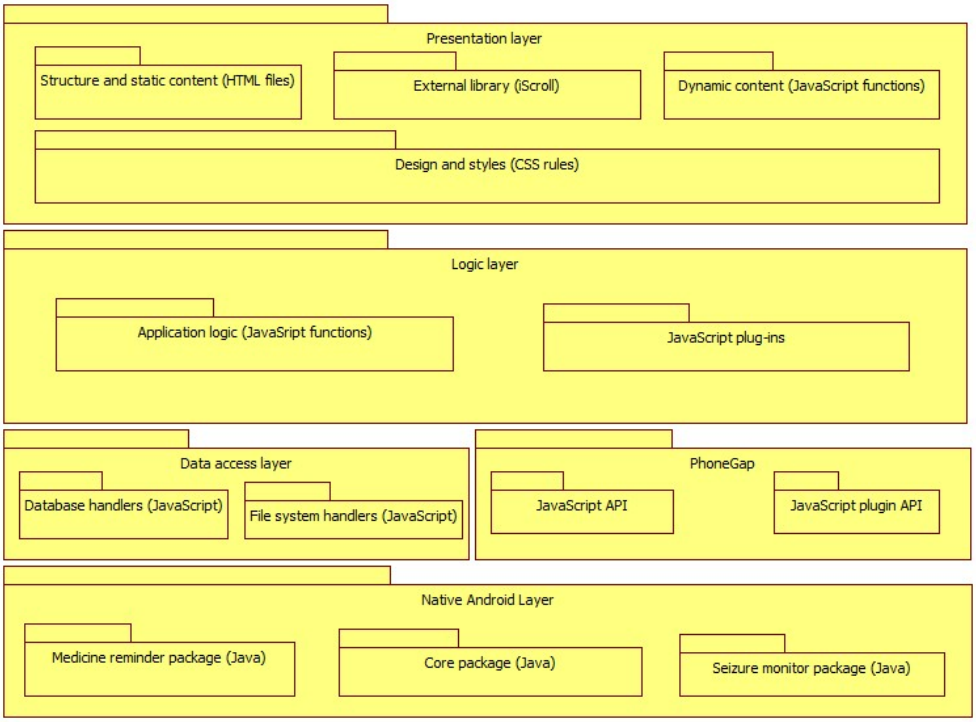
The application is divided into five layers: presentation, logic, data access, native Android and PhoneGap layer. This architecture is basically corresponding to classical 3-layers architecture - presentation, logic and data access. However, in addition to those three traditional layers we have some functionality that is implemented in the native Android layer, that is basically running services needed to start the application, control medicine reminder notification and seizure monitoring. And finally, we have a PhoneGap layer, that connects together logic, data access and native Android layers.

Presentation layer - includes all HTML files that creates structures for applications user interfaces. HTML represents the statical content. In addition to HTML we use JavaScript for dynamically setting layouts, setting event listeners on user interface elements, handling user inputs and dynamically setting styles. The third part of the presentation layer is CSS rules, that define layouts of user interfaces and sets styles. Finally, we have external JavaScript libraries that are responsible for particular functionality. iScroll is used for scrolling the content in the view and Flotr2 library is used for drawing graphs.

Logic layer - is responsible for interacting between the presentation and data access layers. This layer contains JavaScript functions that perform application logic: create calendar, perform event registrations, handle data transfer between data access and presentation layers, handles user preferences etc. It also contains JavaScript plug-ins to access the native functionality of Android.

Data access layer - performs functionality related with data access from local database.

Figure 4: Application structure.



It includes JavaScript functions that are responsible to save data to database and retrieve data from it. This layer communicates with logic and PhoneGap layers.

Native Android layer - is basically responsible for starting the application and doing background work, that JavaScript can not perform from the upper layers. This layer includes three packages: Core, Medicine reminder and Seizure monitor packages.

Core package is responsible for starting and initiating the application.

Medicine reminder package takes the logic for medicine reminder notification.

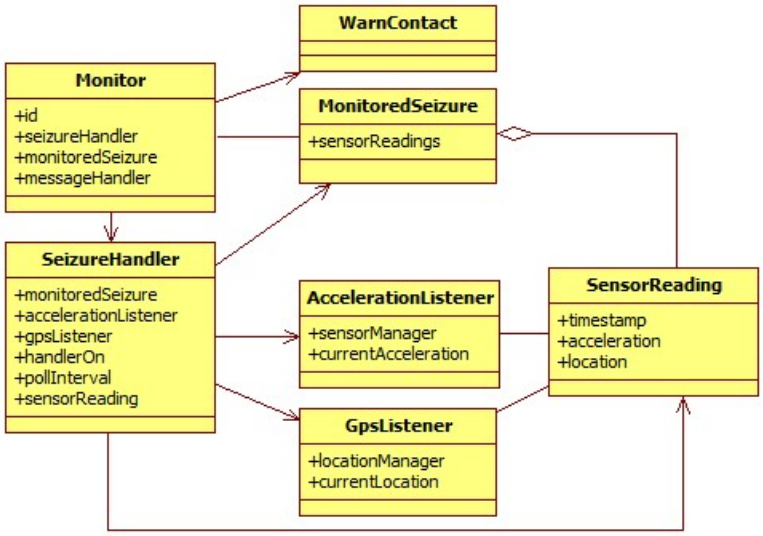
Seizure monitor package includes classes that are responsible for monitoring mobile devices sensors (accelerometer and GPS) and performing warning activities. Classes in this package are initiated through the JavaScript plug-ins from the logic layer of the application and communicates with this layer through plug-ins (see Figure 5).

Splitting the application into different layers help us to structure the application code better and provide better code modification possibilities in a particular layer in the future, without affecting other layers.

3.4 Seizure Monitor Package

In this section we represent the most important package in the application - *Seizure monitor package*. As mentioned above, classes in this package are responsible for seizure monitoring activities. Classes of this package and their relations are provided in Figure 5.

Figure 5: Seizure monitor package.



Seizure monitor package contains seven classes: Monitor, SeizureHandler, AccelerationListener, GpsListener, SensorReading, MonitoredSeizure and WarnContact.

Monitor class is used to start, stop and control the monitoring activity, either seizure monitoring or seizure auto-detection. Monitor is created when user starts monitoring activity through the user interface in the representation level. The JavaScript plug-in in the application logic level activates the plug-in on the Android native layer, through the PhoneGap layer. The Java plug-in on the native Android layer starts the service that creates the object of Monitor class.

We have implemented a *Singleton* software design pattern for the *Monitor* class, that ensures that only one instance of *Monitor* object will be created[5]. This is important to ensure, that user starts only one monitor and if it is already started, it will use the same instance of *Monitor* object.

SeizureHandler is a class, that is actually responsible actual monitoring process. It initiates sensor listeners and data structure for saving sensor data and handles data transfers.

GpsListener and *AccelerometerListener* classes are responsible for getting real-time data accordingly from location and acceleration sensors. We have implemented *android.location.LocationManager*[1] for *GpsListener* for helping to handle location data, and *android.hardware.SensorManager*[2] for *AccelerometerListener* for helping to handle sensor data. Objects of theese classes only stores the current sensor values.

SensorReading class is responsible for "packing" the sensor data. When *SeizureHandler* requests sensors data in a fixed time intervals, object of the *SensorReading* class is

[2]http://developer.android.com/reference/android/location/LocationManager.html/
[2]http://developer.android.com/reference/android/hardware/SensorManager.html/

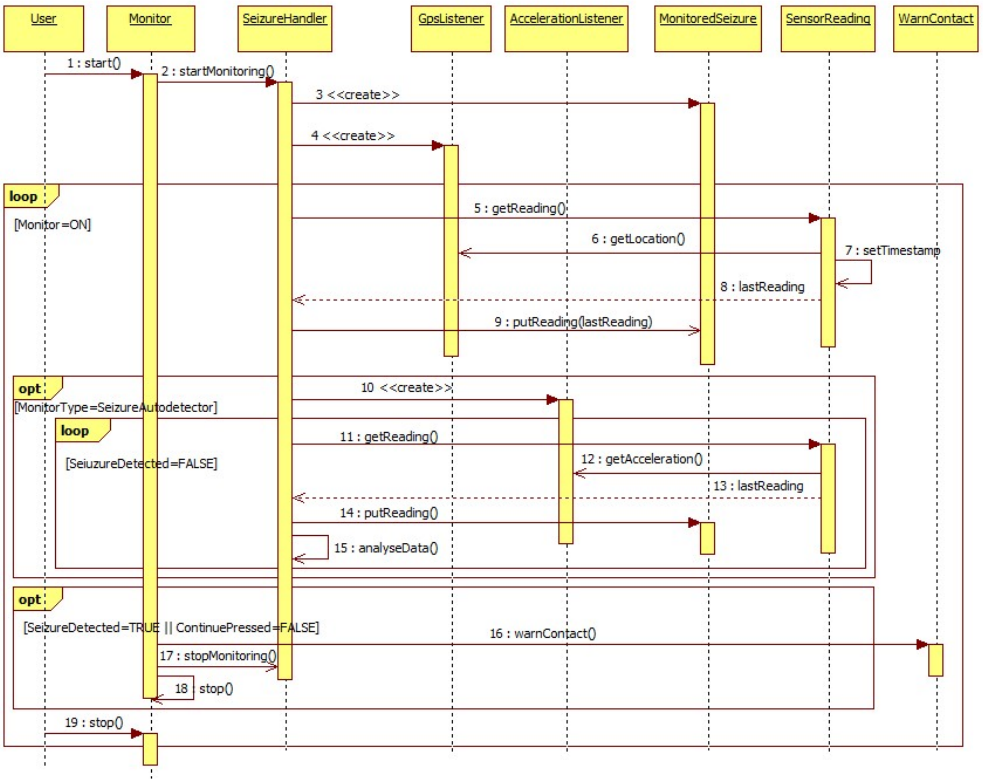
responsible for receiving current sensors data, adding timestamps for data registrations and returning them to the requester.

MonitoredSeizure class is responsible for storing all the sensor readings received from *SensorReading* into one data structure. This data structure is accessible for *SeizureHandler* class object which uses this data for analysis.

Finally, *WarnContact* class is responsible for sending SMS messages for users selected contacts, when seizure is detected or when user do not respond to seizure monitors request. When object of this class is initiated it searches the local database for stored phone numbers and by using *android.telephony.SmsManager*[3]sends messages to all registered contacts.

Figure 6 shows the interactions between objects of the classes from the *Seizure monitor package* in a sequence diagram.

Figure 6: Seizure monitor - sequence diagram.



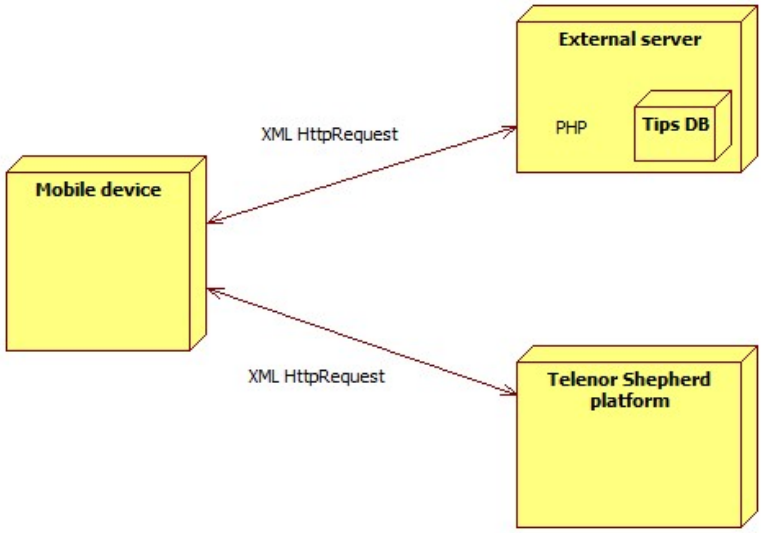
3.5 External Connections

Though the application is mainly running on a single mobile device, it has two connection with external systems - external tips database and Telenor Shepherd platform (see Figure 7).

[3]http://developer.android.com/reference/android/telephony/SmsManager.html/

Given that the application has to be able to update epilepsy related tips when needed, it needs an external storage for saving tips. We have chosen to store tips in a MySQL database, that contains only one table with all tips. Storing tips in a database is a simple and efficient way for retrieving them and maintaining update operations. To request new tips from external tips database we use XML HttpRequests on the application side, because it is the way of retrieving data that PhoneGap supports. On the server side we implement a simple PHP script that queries tips database and returns data to the requester. We have chosen to use MySQL and PHP on the server side, because these are popular and free alternatives and most servers support these technologies. This will assure better migration possibilities in the future.

Figure 7: Application external connections.



Another external connection from the application can be established with Telenor's Shepherd M2M platform using Shepherd API by sending XML HttpRequests.

3.6 Data Storage

We have considered several possible alternatives for storing data that the application will operate with. First, we defined what kind of data the application needs to store. According to the system requirements, we need to store various registrations data (seizures, medicine use etc.), user preferences (language, gender), strings for different languages (English and Norwegian at the initial point), tips and medicine lists. We considered the following alternatives:

- Web SQL database,
- native Android SQLite database,
- Web storage,
- files on the file system.

Considering the data storage we had to take into account requirements for cross-platform compatibility, data security, reliability and performance and the type of data being saved. Here we discuss the storage alternatives for various registrations, tips and medicine lists, as they comprise the most part of the data registered.

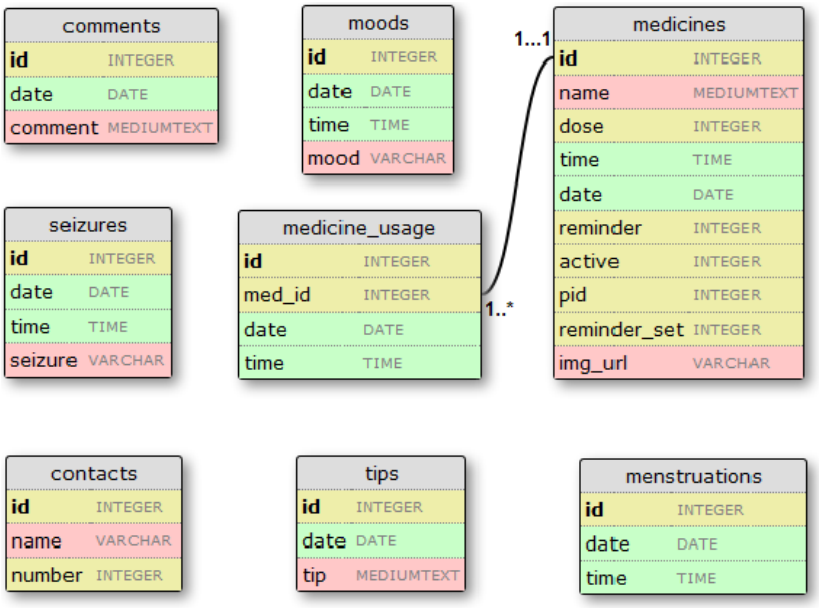
Given that the amount of data to be saved for different registrations, tips and medicine lists can be relatively high and have relatively complex structure, Web storage and writing data on a file would not be good alternatives. Web storage is not suitable for storing complex data structures, as it's used to save key and value pairs and it would only put extra complexity for the application and could possibly be a reliability issue. Saving data into a file would cause performance problems when amounts of data get high over time. It would also not meet the data security and reliability requirements, as files can be easily accidentally deleted by a user or disrupted in other way.

Two other alternatives - Web SQL database and native SQLite database are both suitable for storing this type and amount of data we are targeting. Both of them have good reliability and security characteristics. However, using Web SQL would guaranty cross-platform compatibility as it is supported by both Android and iOS (and other platforms) and can be directly reached by using PhoneGap API. The drawback in comparison with native SQLite database, that Web SQL database has worse performance capability. Using Android native SQLite database, on the other hand, would not meet the cross-platform compatibility requirement. To use the native SQLite database we would need to write plug-ins for each platform we are targeting, because it is the only way to access native platforms database from the PhoneGap application. Transferring data between native and web application levels each time we need to get registrations data could also cause

additional load for the applications performance.

Considering the facts above and after testing the Web SQL database's capabilities we have decided to choose this alternative for data storage in our application. The model for the local database is provided in Figure 8.

Figure 8: Application local database model.



The database does not require a complicated structure. It is simply made of eight tables, six of which are independent. Only *medicine* and *medicine_usage* tables are related to each other. Table *medicines* holds all the information related to each medicine that user registers into his medicine list. *medicine_usage* refers to *medicine* by *med_id* field, and holds all daily registrations of each medicine. The medicine will be registered in the table *medicine_usage* only if user registers it as "taken" to preserve storage space in the database.

3.7 Design Challenges

3.7.1 Communication Between Web and Native Levels of the Application

As our application needs to communicate between the web application level and native application level it is important to find an efficient way for information interchange between these levels. We have considered several types of information sharing between these levels:

- Using database,
- Using shared file(s),
- Using plug-ins to send information.

As both web application and native application levels have access to the Web SQL database, we have implemented it to share information between these levels. This type of information interchange is used for medicine reminder. Because medicine reminder is not controlled by web level application in any way, using plug-ins to share information is not appropriate. Using shared file could be a choice, but writing to file on the web application level has unsatisfactory performance. Thus an optimal way of sharing information between different levels is database.

When a new medicine is registered or an old medicine is set a reminder, it will be registered into the local database table *medicines* from the web application level. Native level checks for reminders after changes in *medicines* table are done and schedules all reminders. When a reminder time comes, the native level application presents a notification for a user and sets reminder to active in the table *medicines* in the database. When the user registers a medicine use from the web level application, the reminder will be set as inactive in the database and this will prevent the native level application from repeating the reminder. In such a way the communication is held between two different levels of application in this case.

Another type of communication between two different levels of application is implemented by sending information via plug-ins where it is appropriate. This type of communication is possible when the native level of application is initiated from the web level of application via plug-ins. In this case, a piece of information that needs to be sent from web level to native level, is sent as a parameter in a JavaScript plug-in and received by Java plug-in on the native level. This type of information interchange is most efficient, because no need to query database is needed. However, it is only appropriate in situations where web level of application initiates communication with native level via plug-ins.

3.7.2 Internationalization

As in the requirement specification we have a requirement to implement at least two languages, some efficient way of internationalization is required. We have again several choices to choose from:

- Use different text file for each language on the web level of application,
- Implement native level internationalization capabilities,
- Use JavaScript objects for each language.

As we have noticed before, that reading a file on the web level of application is of poor performance, we have refused this alternative. Because retrieving strings is included in almost every step that user takes when he interacts with the application, this would no-

ticeably decrease the overall performance of the application and create a bad impression about the application for the user.

Using the native levels internationalization capabilities would ensure better performance than the first alternative, but this would require to implement plug-ins for each different platform we are developing for. It would require significantly more coding and make the application less portable.

We came up with our own solution for internationalization. We use a JavaScript object for each language we implement, that contains all strings that are used in the application. Using this approach satisfies performance and portability requirements for this relatively small application, as well as minimum coding is required. A disadvantage of this alternative is if we implement a large number of languages, as it would cause a need to handle large files. However, we see this approach as the most appropriate for our application and choose to implement it.

4 Implementation

In this chapter we will represent how we realized the application by introducing the development environment and tools we used in the development process, explaining coding techniques we used to solve particular problems and showing some code examples and screen-shots of the application.

4.1 Tools

In this section we describe what type of tools we have use during the project.

4.1.1 Development Environment

Selecting right development environment is important for efficient work-flow. As we have chosen to use PhoneGap as a framework for developing EpilepsyApp we have decided to use PhoneGap's recommended software development environment[4]. We found out that despite the fact that by using PhoneGap we can create cross-platform applications, we need different tools for being able to compile applications for different platforms. For example, to develop application for iOS we would need an Intel-based computer with MAC OS X (at least version 10.6) and Xcode for compiling code. In addition to this we would need iOS developer certification and membership in Apple Developer Portal[1].

We decided to go another way - first to develop EpilepsyApp for Android platform and then make adjustments to adapt application for iOS based phones. This was an easier way to start and we had plans to publish a test version of the application early in development process. Developing for Android platform provides with possibility to publish applications in Android Market (now called Google Play) with no extra concerns as compared with Apple Store. That was what we needed.

As the decision was made, the recommended development environment for Android platform was Eclipse[2]with integrated Android SDK[3].

An advantage of Eclipse is that by using additional plug-ins you get support for editing code in different programming and scripting languages. It covered all the needs for editing and compiling code in our project as it has support for all languages we intended to use - Java, JavaScript, PHP, HTML and CSS.

Another advantage of Eclipse is that it has an integrated support for SVN version control system. Gjøvik University College has provided us with SVN access on schools servers that we planned to use for storing and versioning project files. We saw this as an

[1]Apple Developer Portal. http://www.developer.apple.com
[3]Eclipse. http://www.eclipse.org/downloads/.
[3]Android SDK. http://developer.android.com/sdk/index.html/.

important advantage, since we could use one tool to solve several problems and improve the speed of development.

Both of group members had an experience in using Eclipse from previous development projects and it was also recommended by project supervisor Simon McCallum. Summing all things up the decision was easy to make without need to consider other development environments.

4.1.2 Other Tools

In addition to the main development environment we used several other tools to perform other tasks in the project.

We used *WWW SQL Designer*[4]to draw and keep updated local database model. It is an Open Source and easy to use tool for drawing database models that runs on a web browser.

Another Open Source tool that we used for drawing UML diagrams is *Star UML*[5]. This is a good quality UML platform that supports drawing many types of UML diagrams and export them into an image file.

One more tool that we used for editing icons and images is *GIMP 2*[6]. GIMP 2 is a very popular Open Source image manipulation program with a wide range of capabilities for image editing.

Trello[7]is a web based collaboration tool that helps to manage the work flow of the project. It provides a web "board" with customizable lists of work flow events, like "To-do", "Doing" or "Done" lists. This tool was especially useful for our Scrum development methodology, because we were always able to access the "development board" even if we were not sitting in the development room.

For drawing graphs for accelerometer data representation and analysis we used an Open Source tool *R-project*[8]. This is an advanced tool for statistical computing and graphing that was recommended by Patrick Bours.

Writing the pre-project plan we used *GanttProject*[9]for creating Gantt Chart for the project. It is an Open Source tool used for project scheduling and management.

And finally, *TeXworks*[10]with LaTeX engine was used for writing and generating the final report of the project. Using LaTeX was highly recommended by project supervisor Simon as it obviously speeds up report writing. In addition to these tools we used a LaTeX template[11]provided by Gjøvik University College to meet formatting requirements for the report.

[4]WWW SQL Designer. http://code.google.com/p/wwwsqldesigner/.
[5]Star UML. http://staruml.sourceforge.net/en/.
[6]GIMP2. http://www.gimp.org/.
[7]Trello. https://trello.com/.
[8]R Porject. http://www.r-project.org/.
[9]Gantt Project. http://www.ganttproject.biz/.
[11]TeXworks. http://www.tug.org/texworks/.
[11]Gjøvik University College bachelor thesis LaTeX template. http://gtl.hig.no/index.php/Resources.

4.2 User Interface and Navigation

When implementing the user interface and navigation for EpilepsyApp we have taken into account requirements specified in the Requirement Specification Document (see Appendix B). The following subsections describe how those requirements were fulfilled in the implementation phase.

4.2.1 Navigation

The requirement states that the application should have an easy navigation and that user should always be able to navigate back and to the start position of the application.

We have considered several alternatives for how the navigation in the application should be implemented to fulfill requirements and to be uniform for all platforms. Because f. ex. Samsung phones and iPhones use different navigation approaches it was clear that using native device's implementation would be manufacturer-dependent and less predictable. We have decided to create our own implementation for navigating in the application.

Another thing to consider was *how* to implement navigation. We have considered placing the navigation bar either at the top or at the bottom of the screen. We wanted the navigation bar always be visible on the screen no matter how large the content on the screen was. The problem was that web browsers (that our application runs on) on mobile devices do not support "fixed" elements on a screen (something that would be quite easy to do in a usual web page). Leaving the navigation bar hidden somewhere at the bottom of the content was considered as not satisfying solution.

We tried to implement couple of JavaScript frameworks that solve the problem described. However, neither *JQueryMobile* nor *xui* gave us satisfactory results. Fixed navigation bar with JQueryMobile was somewhat unstable and overall application performance became slower. The performance decrease applies also to *xui* framework.

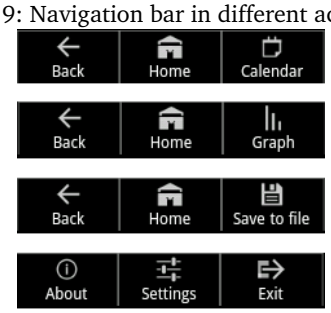
After searching the Internet we have finally found a solution - *iScroll*[12]. This script allowed us to fix the navigation bar at the bottom of the screen and scroll the rest of the content.

This navigation bar is implemented in all application's activities and contains in most cases three buttons: *back*, *home* and one extra button that is dependent on the context, f. ex. *calendar* in Today activity, *graph* in Calendar activity or *exit* in the main view (see Figure 9).

Navigation bar is implemented in all activities and have the same structure that is defined by using HTML and styled by CSS. Because in each activity navigation buttons are different and has a different function to perform, they are defined dynamically using JavaScript. Here is the code snippet that implements a dynamical navigation bar button

[12]Cubiq.org iScroll. http://cubiq.org/iscroll-4/.

Figure 9: Navigation bar in different activities.



creation:

```
function addNavButton(label, id, url) {
  var buttonList = document.getElementById('footbar').getElementsByTagName('
      UL')[0];
  var li = document.createElement('LI');
  var a = document.createElement('A');
  a.href = url;
  var div = document.createElement('DIV');
  div.className = "navIcon";
  div.id = id;
  var labelNode = document.createTextNode(label);
  a.appendChild(div);
  a.appendChild(labelNode);
  li.appendChild(a);
  a.addEventListener('touchstart', onButtonTouch, false);
  a.addEventListener('touchend', onButtonTouchEnd, false);
  buttonList.appendChild(li);
  myScroll.refresh();
}
```

Function addNavButton(label, id, url) takes three parameters: label that will be displayed on a button, identifier for a button that is used for setting an icon for a button and URL of the HTML file that the button navigates to, or a JavaScript function that will be initiated when a button is touched. For example to create a *Save* button on a navigation bar, we just need to call a function

```
addNavButton('Save', 'save', 'javascript:saveEvents();');
```

on activity load. Full listing for navigation bar implementation is provided in Appendix D.

The dynamic creation of navigation bar buttons (and other elements) let us to reuse the same code in different places and ensures uniformity of elements.

4.2.2 User Interface

The main requirements that we took into consideration when creating user interface was simplicity, understandability and easiness to use. Using list approach, icons and big enough elements helped us to achieve this goal.

In Add Event, Settings, Today, Medicines and Add Medicine activities the list approach has been implemented. This approach keeps the structure of the user interface clear and understandable. The information that is not necessary to be shown is hidden inside the list and displayed when the header of the list item is touched. At the same time, when a child list is being displayed, the whole screen scrolls up or down to focus on the

selected list. Showing and hiding lists is implemented by dynamically setting CSS styles for elements.

The lists are formed dynamically using JavaScript, so if it would be any changes it would be easy to adapt them at any time. Invariable data like seizure types or moods are saved in an array, while medicines are retrieved from the local database to form list dynamically. The same functions that form lists in different activities are implemented just with different values provided.

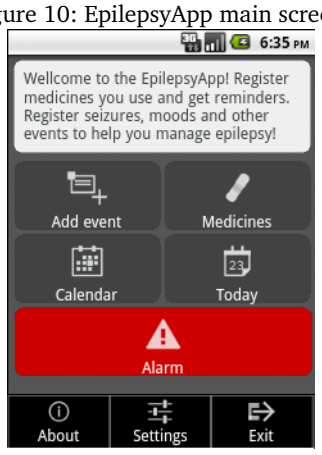
We have also implemented many icons with labels to make it clear what elements on the screen mean. We have used icons provided by Android Design[13]that are free for use.

An important issue for the user interface was to make it to respond to user touches immediately. As applications on mobile devices with touch-screens act differently than other web applications, a different approach has been used to provide a fast responsive interaction feeling. For each element that performs some action we have implemented two types of JavaScript *event listeners*: *click* listeners performs the action (function) that is assigned for the element, while *touch* listeners change the CSS style values for the elements, that gives a fast response when user touches elements on a screen. We need these two types of listeners, because *touch* events occurs immediately, while *click* events have a delay of about 400 ms. Using touch listeners make the application respond faster and perception is everything [6].

4.3 Diary

One of the most important features for this application is diary for epilepsy related matters. We have implemented diary functionality in four different activities: *Add Event*, *Medicines*, *Calendar* and *Today*. All of the diary activities are accessible from the main screen of the application (see Figure 10). This section describes how these activities have been implemented.

Figure 10: EpilepsyApp main screen.



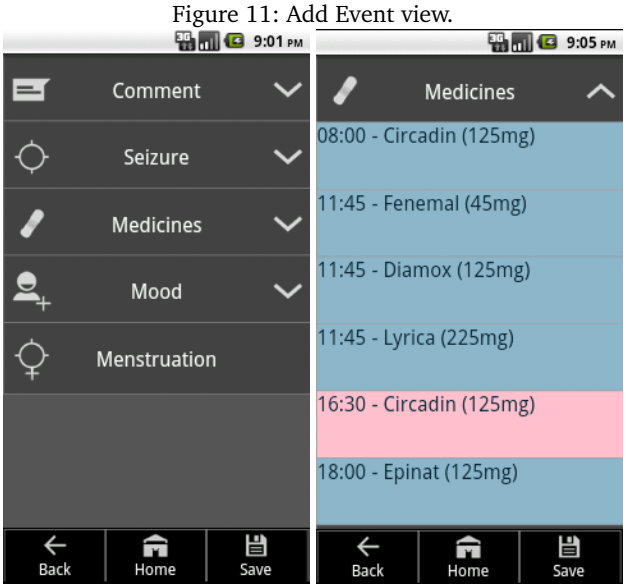
[13]Android Designer. http://developer.android.com/design/downloads/index.html

4.3.1 Add Event View

The Add Event activity is for making registrations in the diary. It provides with possibility to register seizures, moods, comments, medicines taken (and not taken) and menstruations (if relevant) (see Figure 11).

Because this view will be used very often we have tried to make the user interface as simple and clear as possible. For this reason we have decided to show every registration item as a list item, which contains a hidden list of items where it is relevant. Clicking on the parent item results the list of child items to appear and the screen automatically scrolls to focus on the list selected (see Figure 11).

Figure 11: Add Event view.



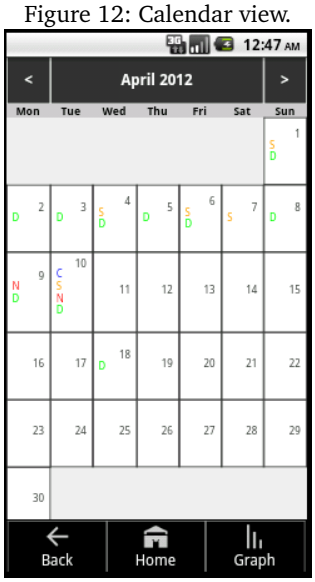
Registering each event in the Add Event activity does not take more than three clicks for a user, what we consider as an acceptable number of clicks to make a registration. By the way, user can select several events to save at the same time and thus to perform needed registrations quickly.

4.3.2 Calendar View

Calendar activity provides an overview over monthly registrations. The overview in the Calendar view is quite general, not specifying seizure types, moods or medicines taken. It simply displays that particular types of event has been registered by showing a letter in a day cell. The label describing each letter is provided at the bottom of the calendar 12.

Calendar view gives a general overview of registrations made by the user and is an entry point to the registrations of a particular day, because clicking on a particular cell in the calendar will open the Today view for selected day with all registrations made for that day. It is specially useful when a user needs to register events in the past if he would have forgotten to, or if he wants to delete a particular event in the past.

Figure 12: Calendar view.



The Calendar activity is also en entry point to the graph view of monthly registrations. Clicking on the graph button in the navigation bar will take user to the graph view for selected month. Navigation to other months is easy and lets to have an overview over registrations for months in the past. User can come back to a current month by simply clicking on the month name.

4.3.3 Today View

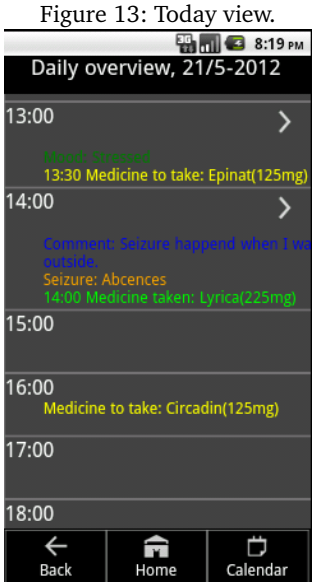
Today activity provides all the registrations made for a particular day and has reminders for medicines that have to be taken that day. The Today view is presented as a list with hourly blocks for entries. Clicking on each hour-item will direct user to the Add Event view where the registration of events can be made. Saving the data will get user back to the Today activity and the registration will be saved for hour that has been selected. One hour-block can have several registrations.

Today view has two entry points - one from the main screen and one from the Calendar activity. If Today activity is entered from the main screen it will display registrations and provide a possibility to make registrations for a current day, otherwise it will show registrations for the selected day in Calendar view and provide with possibility to make registrations for that day.

Currently the Today activity is provided with possibility to delete registrations made by clicking on the “>” icon. Thus editing a registered event would mean deleting it first and then registering a new one instead.

The Today view also shows the medicines that has to be taken if they are not taken yet. When a particular medicine will be registered as taken it will be shown in the Today view accordingly (see Figure 13).

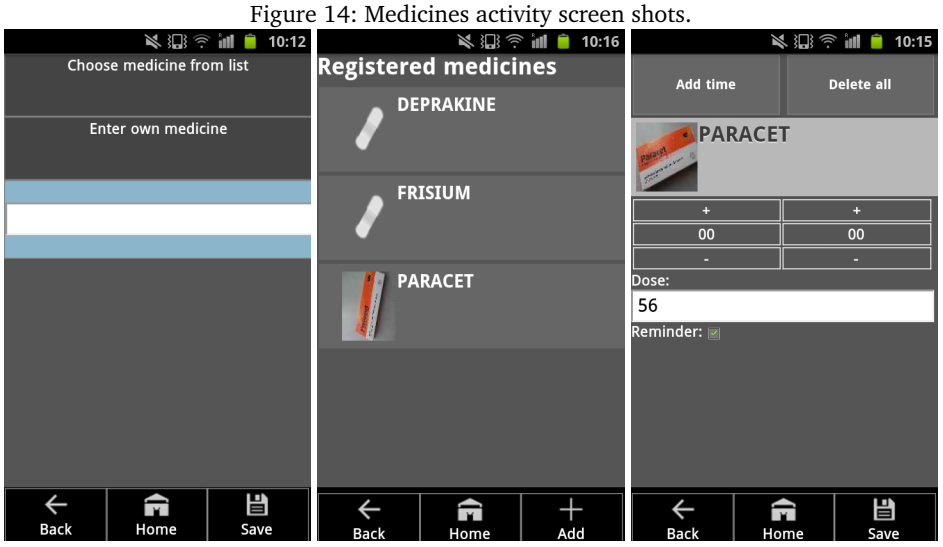
Figure 13: Today view.



4.3.4 Medicines View

Medicines part of the application is divided into three. They are *Add New Medicine*, *List of Medicines* and *Editing of Medicine* (see Figure 14). As the *Medicines* button pressed on main page the list of all registered medicine will be displayed line by line. Every medicine will include default image on the left side of its title. If photo of the medicine is taken then default image will be replaced by this photo. At this page user can navigate to *Add Medicine* page by clicking on *Add* button located in navigation bar at the bottom (see section 4.2.1).

Figure 14: Medicines activity screen shots.

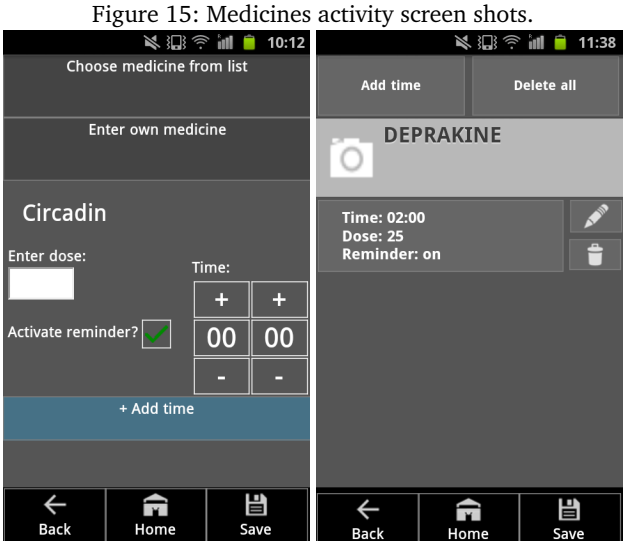


Add new medicine view is appeared with two choices. First is to display all the available medicines as a drop down list or as input field for manual fill out of name for the

medicine. As user chooses name from drop down list or adds medicine name it will follow with a box of extra data request. This kind of implementation will collect all additional data like the dose, time for reminder in incremental way leading user step by step (see Figure 15).

The detailed information about medicine and its edition is available when user selects one of the medicine from the list. The page will display detailed information about the medicine with allowance of editing and deleting it (see Figure 15). Taking a photo of medicine or browsing photo from the device are available. We have implemented both *touch on* the list for added time or *on icon* click for editing medicine dose, time and disabling/enabling reminder (see Figure 15). As the medicines are very important for the user to register and follow after that highly affects his/her daily life it was included all possible function for easiness and logical constraints. We tried to use right affordances in order to lead user to the right way.

Figure 15: Medicines activity screen shots.



4.4 Data Storage

As we have defined in the design phase of the application we will use a local Web SQL database for complex data structures and a Web storage (window.localStorage) to store simple types of persistent data.

4.4.1 Web SQL Database

As described in section 3.6, we have implemented Web SQL database as a main data storage for our application. The data that is stored in the database is various registrations as comments, moods, seizures, medicine usage and menstruation, also contacts, tips and registered medicines as presented in the database model (see Figure 8).

Database will be created in the application the first time user starts the application,

because it is necessary for making all kinds of registrations. The JavaScript function will check if the database exist every time the application starts to ensure integrity.

We have implemented a *database handler*, which takes responsibility for creating database and its structure and making transactions between logic application layer and the database. As we have layered our application, database handler belongs to the data access layer. When a new activity is loaded it will create a new database handler object if it is needed for that activity. Database handler implements the *Database* object from PhoneGap library and it uses PhoneGap API to query the database:

```
// Creates new database handler object
function openDb() {
  var db = window.openDatabase("database", "1.0", "EpilepsyAppDatabase",
      1000000);
  return db;
}
```

All queries to Web SQL database in PhoneGap applications are performed via transactions as in the example bellow:

```
// starts transaction by calling saveEventsToDatabase() function
db.transaction(saveEventsToDatabase, errorDb, resetScreen);

// Performs INSERT query against database
tx.executeSql('INSERT INTO seizures (date, time, seizure) VALUES (date("'+
    sqlDate+'"), time("'+sqlTime+'"), ?)', [seizure]);
```

The first line of code creates a transaction and calls function *saveEventsToDatabase* to perform a query. The second line of code is taken from function *saveEventsToDatabase* and performs *insert* a query against the database that inserts a new seizure registration into the database. *[seizure]* is a parameter that represents a seizure type in this case and will replace a "?" in the query with its value.

4.4.2 Web Local Storage

PhoneGap has a support for storing data in the Web local storage that we tried to take advantage of, where it was appropriate. This powerful functionality let us to solve some of the problems like storing personal data, keeping track of chosen application language, assisting in navigation implementation and storing some other persistent values for the future use. This storage is faster than Web SQL database and it is easier to use it, but it is not well suited for storing complex data.

What is the main power of Web local storage, that the data stored in it is globally accessible for all web pages (for all activities in case of our application) in the web application. The idea of Web local storage is kind of persistent memory cache where you can store a value assigned to a key and it will stay unchanged even if the application will be closed and memory of the device cleared. This type of storage can store up to 10MB data[7]. It uses a simple syntax to put values into a storage and retrieve values from it:

```
// Save value into Web local storage
window.localStorage.setItem("language", selectedLanguage.value);
```

```
// Retrieve value from Web local storage
window.localStorage.getItem("language");
```

The code snippet above presents how we have implemented the Web local storage to keep track of users preferred application language.

We have also used this type of storage for keeping track of user navigation. For example, because Today view has two possible entry points into it, we keep track from which activity the user comes to it. Both Calendar and Main activity sets an entry into the Web local storage if the user is navigating from this activity. Then in Today activity it checks the storage and defines for which date the Today view should be prepared.

Assigning values in a Web local storage has been very handy when we received a bug from Android 4.0 users. The report showed that all users who updated to Android 4.0 failed to load Edit Medicines view. That kind of problem was indicating that the solution would be drastic. Fortunately we have used Web local storage that solved the issue of drastic changes in the code.

Implementation of this feature in our application let us avoid unnecessary storage of data in the database what increases performance of the application, and to avoid changes in the code to solve named bug above.

4.5 Medicine Reminder Notification

Medicine reminder function in the application provides user with a message in the notification bar and a sound, that helps user to take medicines on time. Medicine reminder notification is mainly implemented on the native level of the application. As currently we have developed the application for Android platform, medicine reminder is developed using Java programming language. Interaction between the web level of the application and native level of the application happens when user registers a new medicine and sets reminder for it, and when user registers medicine usage, the medicine reminder is automatically cleared.

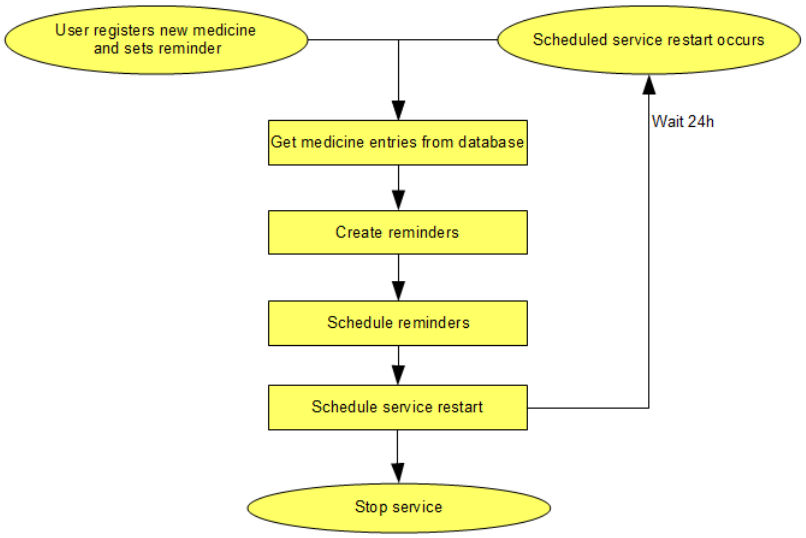
4.5.1 Medicine Reminder Scheduling

The process of how medicine reminders are scheduled is displayed in Figure 16.

The scheduling process is performed in the native application level and is based on Android *Service*[14]class. It starts either when user registers a new medicine or when a scheduled service starts. In the first case user initiates the *EpilepsyAppService* by saving a new medicine into a medicine list. The initiation of the service is performed via custom JavaScript PhoneGap plug-in that we have implemented:

```
backgroundservice.prototype.restartEpilepsyAppService = function(){
  PhoneGap.exec(null, null, "BackgroundService", "restartEpilepsyAppService
      ", []);
};
PhoneGap.addConstructor(function() {
  PhoneGap.addPlugin('BackgroundService', new backgroundservice());
});
```

Figure 16: Medicine reminders scheduling process.



The function prototype defines that *restartEpilepsyAppService* function should be initiated on the native level, when this function is called. It uses *PhoneGap.exec()* function that actually initiates the plug-in on the native level of the application. Call to this function results that the Java plug-in on the native level of the application is initiated. The part of the plug-in that initiates *EpilepsyAppService* looks as follows: lstinputlistingsourceCode/javaPluginEpilepsyAppService.java *BackgroundService* class is our implementation of Java PhoneGap *Plugin* class. Function *execute* is a method in *BackgroundService* class that is inherited from *Plugin* class. It initiates the service that schedules medicine reminders. This service is implemented to run on its own thread, so it doesn't affect the performance of current application work flow, as the scheduling work is done in the background.

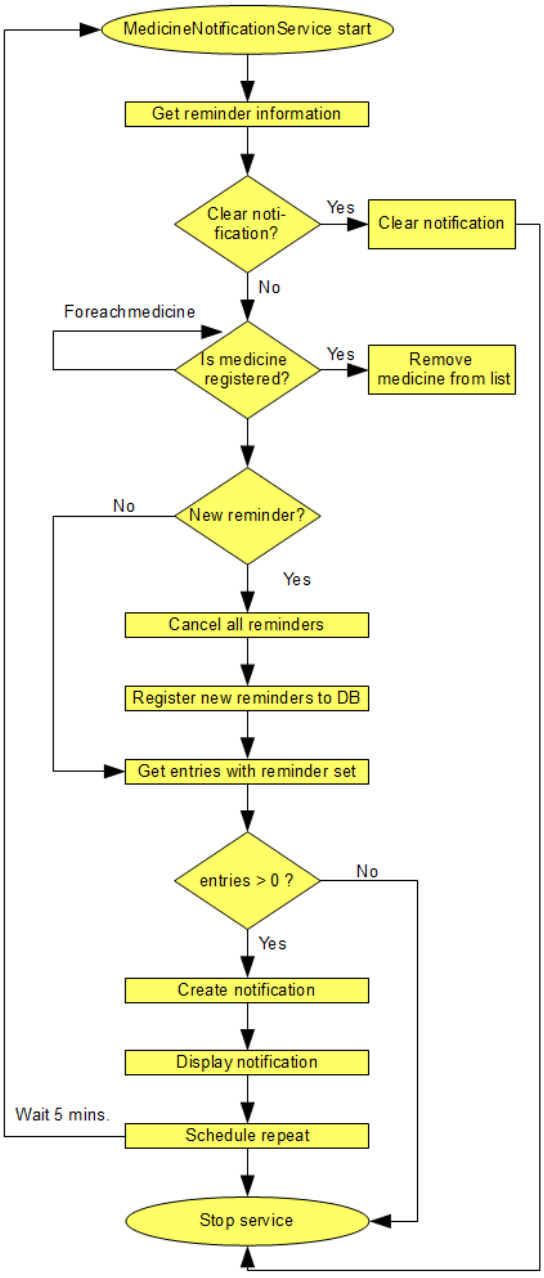
Reminder scheduling is performed by *EpilepsyAppService* class. This service searches the Web SQL database for the medicines which have reminder set. If there are more than one medicine that have a reminder set for the same time stamp, it composes one reminder for several medicines, to avoid showing several reminder notifications in notification bar at the same time. The service uses *AlarmManager*[15]object that implements Android systems *Alarm Service*, that is responsible to initiate the notification at the correct time. Full code listing for *EpilepsyAppService* class is provided in Appendix D.

4.5.2 Medicine Reminder Notification Implementation

To better understand how the medicine reminder notification mechanism is implemented in EpilepsyApp we provide a diagram that shows the processes involved (see Figure 17).

[14]Android services. http://developer.android.com/guide/topics/fundamentals/services.html
[15]Android AlarmManager class. http://developer.android.com/reference/android/app/AlarmManager.html

Figure 17: Medicine reminder notification implementation.



Medicine reminder notification is implemented in *MedcineNotificationService* class that is extended from Androids *Service* class. This service is initiated by the Android system service *Alarm Service*. *MedicineNotificationService* is responsible for creating, displaying and repeating notification in the notification bar until medicine usage is registered by a user.

First of all, when the service starts it creates a new *NotificationManager*[16]that is responsible for showing the notification message in the notification bar and play notification sound. Further, the service retrieves data that is associated with this service call. The service receives four types of data that are needed for the execution: *alarmId* that represents ID of the AlarmManager that scheduled this call, *type* which represents if it's a new notification or a repeated notification, *idArrayList* that contains IDs of medicines that should be included in the notification message and *clearNotification* which can contain values *true* and *false* and defines if the service should clean notification in the notification bar or not.

Before issuing a new notification message the service first checks all medicine IDs it received, to find out if this medicine is still not registered. This is important to check, because it could happen that user took medicine before the reminder was issued, so there is no need to display notification for this medicine. All medicines that are already registered either as taken or not will be removed from the list.

Further the service checks if it is a new reminder or an update to previously issued reminder. If it is a new reminder it has to cancel all currently scheduled reminder updates to avoid collision, and register into the database that the reminders are set for currently received medicine IDs. When this is done, the service checks the database once again to retrieve all the medicine entries that have a reminder set. This is needed because it could happen that some update reminders were canceled in previous steps. If there are any medicine entries with reminder set the service will proceed execution, otherwise it will stop.

When all needed information is prepared, the service starts to compose the notification message for all medicine entries that have a reminder set and creates the *Notification* object[17]. Because default notification bar view does not allow to display multiple lines for a notification, we had to implement a custom notification layout[18]. With our own defined notification layout we are able to show each medicine on a different line, what makes the notification message clearer. When notification is ready, the *NotificationManager* object that was created previously issues the notification, what results that the notification is displayed in the notification bar and notification sound is played.

When the medicine reminder notification is issued, the *MedicineNotificationService*

[16]Android NotificationManager class. http://developer.android.com/reference/android/app/NotificationManager.html
[18]Notification class. http://developer.android.com/reference/android/app/Notification.html
[18]Status bar notifications. http://developer.android.com/guide/topics/ui/notifiers/notifications.html

schedules the reminder repeat after 5 minutes. Reminder repeat scheduling process is similar to the initial reminder scheduling as described in previous section. The scheduling function uses the *AlarmManager* and sets alarm type to “update_reminder”, what results that this service will be called after 5 minutes with updated data.

Full code listing for medicine reminder notification implementation is provided in Appendix D.

4.6 Data Export

We have implemented a possibility to export user registrations data in two different ways - in a graphical representation and as a data table. In this section we describe the implementation of both data export types.

4.6.1 Data Graphical Representation

One of the functional requirements for our application was a possibility to display registered data graphically, in order to use it for analysis. This required to find a method for how to represent different types of data in one graph and find right tools for drawing graphs on a mobile device.

The following data was needed to represent on a graph: seizures, medicine usage, moods and menstruations. To clearly separate different types of data we have decided to use different types of data representation in one graph.

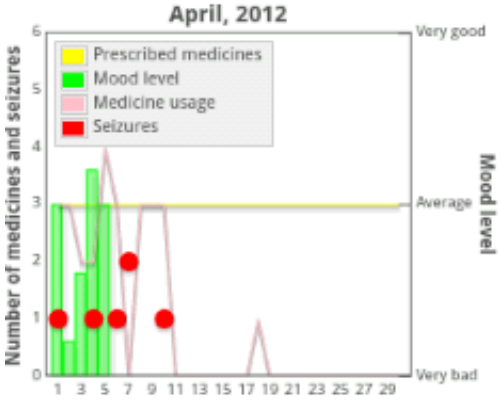
Medicines are something that a person who has epilepsy uses regularly, most usually daily. For this reason, to represent the flow of daily medicine usage we have chosen to use a curve, that represents a number of medicines taken each day. In addition to this we also added an extra curve that shows how many medicines a person was supposed to take (how many medicines he has registered into his list of medicines)(see Figure 18).

Seizures occurs irregularly dependent on a person. Sometimes it may happen one ore more times a day, while otherwise it maybe happens once in a month. Taking this into account we have chosen to represent seizures as dots on a graph. A dot y coordinate represents how many seizures occurred on a particular day (see Figure 18).

Mood was something more difficult to represent. Counting mood registrations or representing each mood on a graph would not make sense or make the graph unreadable. We have decided to use a percentage scale to represent an average mood level for a day. This is not an ideal solution, because the mood can change extremely during a day, but it at least gives some information on how a person was feeling. Using this technique we assigned each mood item a value ranging from 10% to 100% (see Appendix D). Evaluating a mood level for a particular day we simply calculate an average of all registered moods for that day (see Figure 18).

When we had a clear view of how the data should be represented we started looking for a tool that can draw graphs on mobile devices browsers. We have considered

Figure 18: Data graphical representation on a mobile device.



Google Chart tools, RGraph, Hightcharts JS and AwesomeChartJS that are known tools for drawing graphs on web-pages using JavaScript, HTML and CSS. However, none of the mentioned tools did not fulfill all the requirements we needed. Google Chart tools has a limitation that it can only be used when application has access to the Internet. RGraph is a nice tool, but it has not enough customization possibilities that we needed. Highcharts JS has a wide range of chart types and suggests a high level of chart customization. However, it requires using JQuery framework and do not support drawing graphs on Android 2.x and lower versions. AwesomeChartJS only supports drawing simple charts that do not met our requirements.

After a long searching we found a tool *Flotr2*[19]that met all the requirements for our graph drawing. It does not have much documentation, but provides several examples that are enough to learn and start using it. In addition it provides a possibility to export generated charts into a .png file. This feature was especially useful later in the project when we worked with data export.

We have implemented data visualization for one month period, because drawing graph for a longer period seemed to be unreadable on a mobile phones screen. Shorter than one month periods were considered as too short for analysis purposes as it is hard to see any trends.

4.6.2 Data Table

As we have seen the necessity of data export in table format the decision has been taken to include this feature in the application. This kind of data representation beside the graphics is important for better analysis of all registered data by a user.

The implementation of the functionality was little bit tricky. The tables that we were using in database unfortunately has been created with same row names. That made difficult to retrieve data from the database as it gave similar outcomes. Such kind of

[19]Flotr2. http://humblesoftware.com/flotr2/

situation made it difficult to get mood, seizure and other registrations from the database according to the date period set by the user. We had two ways to solve this problem:

- Changing the row names in tables in database.
- Copying all necessary data into a new table.

The first alternative has been dropped as the application has been already uploaded to Android Market (Google Play) Changing the name of the rows of tables could cause extra possible errors for the downloaded applications. It was more about editing all the methods operating on database. That's why it was second alternative to choose for the best solution. After copying all data to the new table we used SQL syntax to get all the data in the range of a given date. Afterwards all collected data has been written to the file with the use of PhoneGap function. The suggestion for append or writing every time new file for presented to project owner. Project owner has chosen append type for the file as it was better for the user to see all historical data export. Statistical table data export will be available as a separate date defined export of data or monthly data export included in graph. All data is exported to an HTML file.

4.7 Data Backup and Restore

Implementing registration data backup feature was one the most important steps to ensure data integrity. This feature provides user with possibility to backup all the registrations he made in the past and to use this backup later, in case he wants to move application to another phone or if data is lost in some way. The backup function stores a data file on users mobile device's memory card and can be copied to users PC from there. We describe below how we implemented backup and restore functionality in the application.

4.7.1 Backup

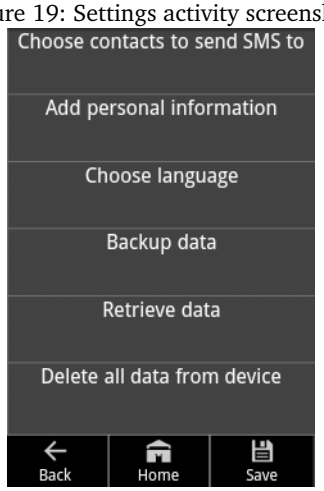
User can backup all the registrations from Settings activity in the application by choosing *Backup data* option (see Figure 19).

An interesting thing about using PhoneGap API is that it uses a very different approach to calling functions than we are used to. It uses a *callback* approach. It means that if you need to call several functions in a script which results are dependent on each other, you have to call the first function and define in its parameters what is next function to call when this function finish execution. You can actually define two functions - one for successfully finished execution and another for failed execution as in this example:

```
fileEntry.createWriter(successFunction, errorFunction);
```

Using this approach makes coding very procedural and the code itself is more compli-

Figure 19: Settings activity screenshot.



cated to understand. You have to find out an exact sequence of steps that have to be made to reach a result.

The backup routine starts with getting access to user devices memory card:

```
window.resolveLocalFileSystemURI("file:///sdcard/", gotRootDir,
    errorFunction);
```

If access has been obtained the next function that will be called is *gotRootDir(rootDir)*, with the link to a root directory as a parameter. If not, the *errorFunction* will be called, that just gives a warning message to a user. The code for *gotRootDir* function looks as follows:

```
function gotRootDir(rootDir) {
    rootDir.getDirectory(homeDir, {create: true, exclusive: true},
        createFolder, fail);
}
```

When this function starts executing it calls another function *getDirectory* that tries to get access to home directory for EpilepsyApp . If the directory does not exist, it will be created. If access is obtained, it will try to get access to *Backups* folder in the same way as in previous example. When the access to the *Backup* folder is obtained application tries to create a file in that directory with auto-generated filename:

```
function createFile(backupDir) {
      var date = new Date();
      var year = date.getFullYear();
      var month = date.getMonth()+1;
      var day = date.getDate();
      filename = "backup_"+day+"_"+month+"_"+year+".txt";
      backupDir.getFile(filename, {create: true, exclusive: false},
          gotFileEntry, fail);
}
```

Filename is generated using current date value. To give access to write data to the file we have to call another function *gotFileEntry* that prepares file for writing:

```
function gotFileEntry(fileEntry) {
    fileEntry.createWriter(gotFileWriter, fail);
}
```

Finally, when the file is ready for writing data into it, we start retrieving data from the database:

```
function gotFileWriter(writer) {
  var db = openDb();  // create DatabaseHandler object.
    wr = writer;
    db.transaction(getAllEntries, fail);
}
```

This function first creates a *DatabaseHandler* object, that handles all database transactions to the local Web SQL database. The connection with the database is made when *DatabaseHandler* calls its *transaction* method. The data retrieving from the database starts when *transaction* method issues a callback to *getAllEntries* function. This function simply queries database's different tables to retrieve data from them. All data is then saved into an array, because direct writing to a file is not supported by PhoneGap. We do not list the code for database querying here, because database handling is described in another section.

Finally, when file is ready for writing and all data entries are retrieved, the data can be written to the file. This is performed by simply calling a PhoneGap's function

```
writer.write(data);
```

where *writer* is the file writer we created before, and *data* is the data array that we used to save entries retrieved from the database.

4.7.2 Restore

User can start data restore from *Settings* activity by selecting *Retrieve data* option (see Figure 19). The data restore routine starts similarly as in the data backup - first application tries to get access to the memory card of the mobile device, then get the home directory for EpilepsyApp and *Backup* directory that resides in the home directory. Further application tries to get a list of all backup files that user have saved:

```
var directoryReader = dirEntry.createReader();
directoryReader.readEntries(readerSuccess, error);
```

First a *DirectoryReader* object is created for the current directory, then it's used to read the content of the directory. When the content of the directory is obtained the application forms a visual representation of the list of files contained in the directory (see Appendix D for the code). Each entry of the list that represents files in the directory has an event listener attached, that responds to *click* events:

```
listEntry.setAttribute("onclick","startRetrieve(this)");
```

When user touches on the list item the application starts the routine for getting access to the file that was chosen by the user. The routine is exactly the same as described in Backup subsection 4.7.1, until a file entry is obtained. When the file entry is obtained, the file is prepared for reading and the reading process starts:

```
function readAsText(file) {
   var reader = new FileReader();
```

```
        reader.onloadend = function(evt) {
        var string = evt.target.result;
            data = string.split(" ");
        confirm();
        };
        reader.readAsText(file);
    }
```

FileReader object reads in data from the file and saves it as a string. Later this string is splited into different strings and saved into *data* array. At this point user is warned about that all current data stored in database will be deleted and replaced with data from the backup file. If user confirms to proceed, then actual data restore process starts.

```
function onConfirm(button) {
    var db = openDb();
    if(button==1){
        db.transaction(deleteDatabase, errorDb);
        db.transaction(createTables, errorDb);
        db.transaction(restoreDatabase, errorDb);
    }
    else
        return false;
}
```

The process of restoring data into database starts with deleting current database, then a new database and its tables are created and finally data is retrieved from the array and recorded into the database. Full code for restoring data to database is provided in Appendix D.

4.8 Communication with external systems

EpilepsyApp can communicate with two external systems - external tips database and Telenor Shepherd platform[20]. In this section we describe how we have implemented communication with these systems.

4.8.1 Tips Database

EpilepsyApp provides with a possibility to update the application with new daily tips.

Tips are currently stored in an external MySQL database on a server provided for this project by Gjøvik University College . The database contains the only table *Tips*, that keeps all tips provided by project initiator Randi Stokke. It is a simple table which has three fields: *id*, *date* and *tip*.

On our web server we have placed a PHP script that reads the database and returns tips to the application. Because only new tips has to be sent to user, the script receives a date of last tip saved in a local database from the application. Tips are retrieved from the database according to the date provided and sent back to the application in JSON[21]format. We use JSON format because it minimizes the data amount sent through the data network and it is easy to parse retrieved data with JavaScript in the application. Code for PHP script that retrieves data from the database is provided in Appendix D.

[20]Telenor Objects. http://www.telenorobjects.com/
[21]JavaScript Object Notation. http://www.json.org/

The application requests data from the external tips database by sending XMLHttpRequest objects using Ajax approach. This approach is the only one that is supported by PhoneGap to retrieve external data. Using Ajax we can retrieve data in the background without interfering with user interface of the application[8]. Function *updateTips* shows how we have implemented Ajax approach for retrieving tips:

Ajax uses the approach of checking HTTP *ready state* that come with responses from server in HTTP headers. Each time the ready state changes, function *loadTipsFromRemoteDatabase* will be called as we see from the function above. Each time this callback function is called, it checks if ready state of HTTP request is equal to 4, what means that the the request is complete and the response is ready, and status equals to 200, what means that the request was successful[22]. The data received has to be parsed, because it was sent in JSON format. In JavaScript it is done in a simple way:

```
var data = eval ('('+this.responseText+')');
```

Function *eval* parses the response from the server received as a string and saves data into an array of JavaScript objects *data[]*. Further, the data from the array is saved into the local database.

4.8.2 Telenor Objects M2M Communication

As it was decided before, the implementation of M2M (machine to machine) communication could be a useful solution for those who want easily and simply get various data from different devices. This possibility can collect all necessary data including all registrations made in EpilepsyApp for the analysis. Such kind of solution can resolve problems of waiting time to a doctor and even let share collected data with a doctor for future analysis. Of course extra devices and provided service should be purchased from Telenor if user decides.

As we have begun contact with the Telenor about the idea to test implementation of Telenor Objects M2M communication, we have discovered that the basics in EpilepsyApp should be developed before integration. Additionally we have been provided with documentation by Telenor that would give an idea of integration with Telenor Objects. The documentation has been read and M2M Shepherd platform has been tested with some objects provided by trial profile. It was interesting to test the power of API that lets different devices to communicate with each other.

M2M communication operates with simple HTTP request POST and GET methods. The flexibility of getting posted data from Shepherd as XML or JSON format is available that resolves adapting retrieved data in our application.

When the tutorials have been accomplished and some small tests on Shepherd and objects has been done in web browser we have started integration of Telenor Objects with our application. The POST function with the GPS location and temperature as a test value

[22]http://www.w3schools.com/ajax/ajax_xmlhttprequest_onreadystatechange.asp

has been successfully posted on Shepherd using mobile device. No errors arose at the test process. As we have started integration of GET part, where the stored data from other devices in Shepherd would be downloaded is failed. After four days with several tests the report about the issue has been sent to Telenor Object team. Even several different ways of getting data did not bring us to the satisfying result. As we could not get any response or error on PhoneGap we begun with some tests. We tried to use different web browsers on computer to see how it acts. The team has detected several different responses to the same GET operation (see Appendix F). The result of tests and the report has been sent to the Telenor.

The reply for the sent report from Telenor Object team had a clear message that the Shepherd had not encountered any such kind of problem providing us with several alternatives among them are links that provide codes written in other programming languages F. Our team decided to make extra investigation of existence of same issues. After some surveys we have found that the issue with GET operation was probably in PhoneGap that was failing even though it is not showing any error code[23]. It seems that PhoneGap is still in the progress that fails to operate on authenticated area with GET HTTPRequest. Due to time limitations dedicated for the Telenor Objects and error that have not been resolved the integration of M2M communication through Shepherd was canceled. We still wish and hope to provide this function in future versions if the issue will be resolved by the PhoneGap. As a result we have only one way, POST function for the Telenor Objects.

4.9 Internationalization Implementation

We have analyzed and described the internationalization problem in the previous chapter section 3.7.2. In this section we describe more specifically how use of different languages is implemented in EpilepsyApp .

We have chosen to store all strings for each language that we use in the application in one JavaScript file. Strings for each language are stored in a separate *associative array*[24]. Each string in an associative array is identified by a *key*, that corresponds to a *value*. Using associative arrays lets us to identify strings in each array easily, by simply providing a key. Every associative array has same keys which correspond to values, that are specific for each language, as in the example below:

```
// English strings
var strings_en = {
  "btn-delete":"Delete",
  "btn-edit":"Edit"
}
// Norwegian strings
var strings_no = {
  "btn-delete":"Slett",
  "btn-edit":"Rediger"
```

[23]http://groups.google.com/group/phonegap/browse_thread/thread/7f5bf338ef069a69?pli=1

```
}
```

To retrieve strings from the associative arrays we have implemented two slightly different functions - *getString* and *getStringHtml*. They differs only in how the data is returned. The first function returns a simple string retrieved from the array, while the second returns a HTML element. The first function is used to retrieve strings in a JavaScript code, while the second function we use in HTML markup. We implemented the second function, because it lets to reduce the amount of code to be written each time we need a string in HTML markup.

Both functions first check which language user has chosen and then use *switch* statement to choose a right associative array to retrieve a string from. Users preferred language is retrieved from the *window.localStorage* item *language*. Function uses a key to identify string in the associative array and returns it:

```
function getString(key) {
  var resultString;
  switch(window.localStorage.getItem("language")) {
  case "fr":
    return resultString = strings_no[key]; break;
  case "nb":
    return resultString = strings_no[key]; break;
}
```

Function *getStringHtml* instead of returning a string creates a HTML element by directly writing the string at the point where it is inserted:

```
document.write(resultString);
```

Using internationalization implementation makes code writing and reading a little more complicated, as you have to write functions and to read keys instead of strings themselves. Each time when we need to insert a string in the JavaScript code, we write *getString(key)*, and if we need the same string in the HTML markup we have to write *<script>getStringHtml(key)<script>*. However, implementing internationalization provides significant benefits for the application.

4.10 Seizure Monitor

One of the important features of this application is Seizure Monitor. Seizure Monitor actually is ground functionality for two sub linking features like Seizure Detection listening the accelerometer data behavior and Alarm initiation.

4.10.1 Alarm Function

Here we are going to describe Alarm functionality. Implementation of the Alarm was developed in *PhoneGap* and *Native Android* code. We tried to produce most part of the functionality in *PhoneGap* as it important in future development on different platforms. Basically start of Alarm invokes *The Service* class written in Java that is initiated through JavaScript plug-in. Service start will show indication of its start by setting an icon at the

[24] JavaScript associative array. http://www.pageresource.com/jscript/jarray2.htm/

notification bar. Afterwards the delay time will be set before asking user if he/she wants to continue monitoring. If user wishes to continue then it will postpone send of SMS to chosen contacts. It will repeat as long as you cancel or ignore the alert message. If the alert message is ignored the service part will initiate beep sound with vibration until someone recognize the person and it is shut down. The ignorance of alert message can indicate the occurrence of seizure. In a such situation the SMS with GPS location will be sent to the chosen contacts. The loaded page from service will provided first help transitional image instruction for guidance of possible nearby persona. The Alarm function has been changed slightly according to the given scenario to satisfy product owners suggestion. This alternative according to the previous version has faded unnecessary annoying sound right after Alarm initiation.

Warning Contacts

Warning contacts by sending SMS with the GPS location will be available if user have chosen contacts from the settings. User have two alternatives. They are choosing contacts from the list of available contacts stored in your mobile device or input a mobile number manually. Here we have set limitation for the number of contacts with ability to remove from list if necessary. The list of chosen numbers must not exceed more than five contacts. All defined numbers will be saved to the database in order to secure sending of SMS to the right person/people. Every time when alarm is initiated and have not been canceled it will look after mobile numbers in database for sending alarm message.

Template for message: *SENDER HAD A SEIZURE | LOCATION:* `http://maps.google.com.my/maps?f=d&source=s_d&saddr=Latitude,Longitude`

4.10.2 Automatic Seizure Detection

Automatic seizure detection was not one of the major features for the EpilepsyApp , but an additional one that would give the application a high value. The idea was to use mobile devices accelerometer sensor for detecting seizures automatically. However, due to time limitations we have not implemented this feature in the current application version, but we have started to prepare bases and provide opportunities for further development.

Because we had no idea of how epileptic seizures could be detected by a mobile devices accelerometer, we have contacted Mr. Patrick Bours, an associate Professor in Information Security in Gjøvik University College , who could give us some suggestions on the issue we were working with. The first thing we had to do was to collect accelerometer data for various normal daily activities like walking, running, going up or down stairs, jumping etc., and epileptic seizure specific movements like falling or jerking.

To collect accelerometer data we have implemented *AccelerationListener* class from the *Seizure monitor package* described in section 3.4. In addition to this, we have implemented a help class *AccelerometerMonitorService*, which helps to retrieve data from the

AccelerationListener during data collection process and write accelerometer readings to file. Source code for these classes is provided in Appendix D.

An important thing in collecting accelerometer data for analysis is to get accelerometer readings in high frequency. Mr. Patrick Bours suggested to take accelerometer readings at least 100 times per second (each 10 milliseconds) for data to be useful for this type of analysis we needed. We were not sure about if our mobile devices were able to collect data in such a high frequency, but the tests showed that the devices managed to handle this load.

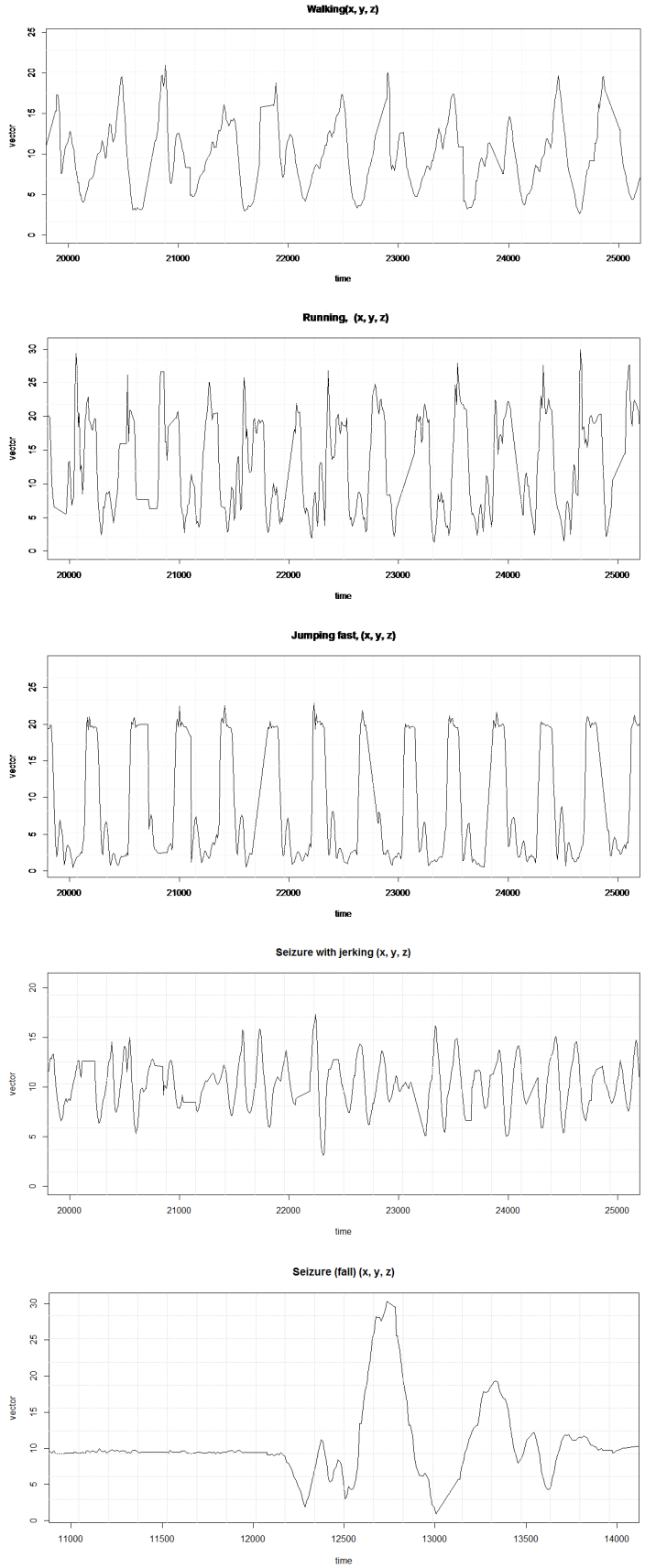
We have chosen to write accelerometer readings data to a CSV (comma separated values) file format, as this file format was supported by R Project tool that we were going to use for drawing graphs for accelerometer data, and it is generally well supported and widely used file format. Collecting data in CSV file format provides us with more flexible data analysis possibilities if it would be needed.

We have collected accelerometer data for around 20 different normal daily activities and epilepsy seizure specific falling and jerking. Epilepsy specific movements were simulated by team members, trying to imitate seizures as accurately as possible. Because of this, seizure data may not be completely accurate.

To analyze differences between movements of normal daily activities and seizure specific movements, we have drawn graphs for each type of activity to see possible repeating patterns. Graphs for some of the activities are presented in Figure 20. Graphs for all activities that were observed are provided in Appendix E.

Further implementation of seizure auto-detection would follow the analysis of collected data and trying to find repeating patterns in the graphs and describing the differences between natural activities movements end seizure specific movements. Then the patterns found should be described with mathematical algorithms and programmed into the application. The logic for seizure auto-detection could be implemented as described in the sequence diagram in Figure 6 in section 3.4. The data analysis process would include the background service trying to compare collected accelerometer data to the predefined patterns. If the pattern of the data collected by the monitor would match the seizure specific pattern, the monitor would issue an alarm and warning messages for users contacts would be sent.

Figure 20: Accelerometer data for various activities.



5 Testing

In this chapter we discuss the testing strategy and its implementation in our project.

5.1 Testing Strategy

As we have described in the pre-project plan, the testing of the application have lasted during the whole project period (see Appendix A). To test EpilepsyApp we have implemented *White box* and *Black box* testing techniques. Using a combination of these techniques provides us with a comprehensive application testing.

White box testing is a method of testing software that tests internal structures or workings of an application[9]. This type of testing have been implemented in our application in order to test if the application is performing actions in a correct way and giving right outputs.

Using black box testing, no knowledge of internal logic or code structure is required. The types of testing under this strategy are totally based/focused on the testing for requirements and functionality of the work product/software application[10]. We have implemented two types of black box testing methods - *User accepatance testing* and *Beta testing*.

We have used User acceptance testing in order to find out if the application works as expected an meets users expectations. We have involved the project initiator Randi, our supervisor Simon and other people who were interested in testing the application.

Using a Beta testing we have expectations to test our application with as many different devices and different platforms as possible. To implement the Beta testing, at the end of the fifth Sprint when the application had its main functionality implemented, we have released EpilepsyApp beta version to the Google Play marketplace.

5.1.1 White-box Testing

White box testing was performed by both group members during the whole development process. To test the application we used the debug mode on our development environment both with real devices and device emulators. We have logged outputs from different functions and tried to compare them to the expected results. This way of testing was very helpful and it helped us to catch many small and more serious logical bugs.

During the development process we have used three main mobile devices for testing: HTC Wildfire with Android version 2.2.1, Samsung Galaxy Mini with Android version 2.2.2 and Samsung Galaxy S2 with Android version 2.3.5. These devices are different in performance capabilities, screen sizes and have different platform versions. In addition

to this we have tested the application on different virtual Android and iPhone emulators.

Testing of the application on the iPhone emulator was interrupted later in the development process, because the development machine that we used for testing our application has updated its development environment (XCode) to be able to compile applications for iOS 5. After that, trying to compile the application for lower versions of iOS (that our application is compatible with) resulted in requiring to update the whole operating system of the development machine. Since it was not possible to upgrade to a higher version of operating system because of financial and time restrictions, the testing of the application on iPhone emulator was suspended.

At the end of each Sprint before Sprint Demo meeting we have performed functionality tests, to see how the implemented functionality corresponds with the functional requirement specifications. Having the results for these tests we were able to present for the project initiator Randi a correct view of current functionality.

Late in the development process, we had a possibility to test EpilepsyApp on a tablet, mainly for testing how our application looks on large screens. The test gave satisfying result. Application looked well on a large screen and could perform other functionality as expected.

5.1.2 Black-box Testing

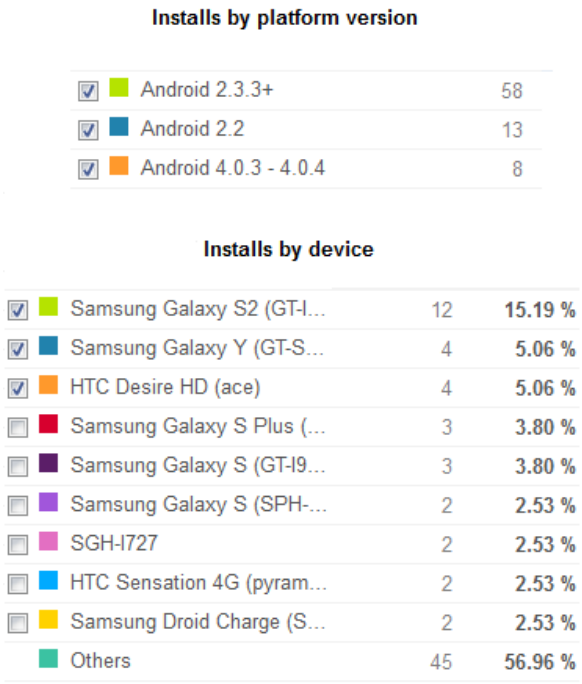
Black box testing was performed by both project initiator Randi, supervisor Simon and other people who helped us to test the application. The purpose of this test was mainly testing the functionality and user acceptance. This type of testing was particularly active and effective, because we were in a close contact with these testers all the time. We got much feedback that was helpful in fixing bugs and improving user acceptance.

One of the major bugs that was reported by this type of testers was detected by a user who used the application on a mobile phone with Android version 4. When we tested the application on our test devices we could not recreate this type of error. It turned to be an error that was only present for devices with Android version 4. After careful analysis we have discovered that the problem was, that devices with Android 4 were incompatible with the way how we had implemented data sharing between different activities. The problem was fixed by implementing Web local storage as described in section 4.4.2.

Another way of black box testing that we have used was performed via Google Play service by releasing a *beta* version of the application to the market. This type of testing was totally random, because random people have installed application on various devices. We have noticed some advantages and disadvantages with this type of testing.

The main advantage with this testing was that we could test the application on a wide range of mobile devices. It gave as at least a clear view that the application is able to be installed and run on many different Android platform versions and many different phone types as presented in Figure 21.

Figure 21: EpilepsyApp installs on different Android platform versions and mobile devices via Google Play.



Another advantage of this type of testing was that people were able to send automatically generated error reports via Google Play. This type of reporting was especially useful, because we could concretely see where bugs in the application were. The reports were sent for errors that crashed the application - serious bugs that had to be fixed immediately to keep application stable and users satisfied.

One of the bugs that we have fixed with the help of report sent by a beta tester via Google Play was related to incorrect handling of medicine dose. We have implemented the medicine dose to be a decimal number, but when the user entered a floating point number and tried to save a new medicine, the whole application crashed and could not start again. After analyzing this error report (see Figure 22) we immediately found out that the problem was, that a function tried to parse a float number as an integer. This bug was fixed by implementing a float type number for the medicine dose instead of integer.

We also got several responses from users of EpilepsyApp who installed the application from Google Play about the functionality and user acceptance of it. People sent responses by e-mail address that was provided in Google Play and the website of the project, wrote comments in the Google Play or just wrote a response message together with auto-generated error report.

However, the number of responses from Google Play users was smaller than we expected, what was the main disadvantage with this testing method. By using this testing

Figure 22: Error report from Google Play. Bug for incorrect medicine dose handling.

v1.6 May 21, 2012 1:45:07 PM 1 reports 1 reports/week

```
java.lang.RuntimeException: Unable to create service no.hig.stud.bachelor.epilepsyapp.EpilepsyAppService:
java.lang.NumberFormatException: unable to parse '56.2' as integer
at android.app.ActivityThread.handleCreateService(ActivityThread.java:1959)
at android.app.ActivityThread.access$2500(ActivityThread.java:117)
at android.app.ActivityThread$H.handleMessage(ActivityThread.java:989)
at android.os.Handler.dispatchMessage(Handler.java:99)
at android.os.Looper.loop(Looper.java:130)
at android.app.ActivityThread.main(ActivityThread.java:3691)
at java.lang.reflect.Method.invokeNative(Native Method)
at java.lang.reflect.Method.invoke(Method.java:507)
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:907)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:665)
at dalvik.system.NativeStart.main(Native Method)
Caused by: java.lang.NumberFormatException: unable to parse '56.2' as integer
at java.lang.Integer.parse(Integer.java:383)
at java.lang.Integer.parseInt(Integer.java:372)
at java.lang.Integer.parseInt(Integer.java:332)
at no.hig.stud.bachelor.epilepsyapp.EpilepsyAppService.setReminders(EpilepsyAppService.java:144)
at no.hig.stud.bachelor.epilepsyapp.EpilepsyAppService.onCreate(EpilepsyAppService.java:39)
at android.app.ActivityThread.handleCreateService(ActivityThread.java:1949)
... 10 more
```

method we were not sure how many people will give us some feedback. In the description of our application in Google Play we have informed users that it is a test version of the application and that all types of feedback are welcome. As we see now, it was probably not enough, so we should have implemented some reminder in the application itself to ask people to send some feedback to as.

6 Conclusion

6.1 Discussion and Results

6.1.1 Results

The *Requirement Specification* section have shaped the incremental work that made possible to see some of those functionalities implemented in reality. The result shows that most of the features defined in the initial product backlog has been developed. The main functionality that was presented by the Product Owner are developed. The diary that includes all necessary registrations like seizure, mood, menstruation, medicines and its edition are developed. The data export both graphical and statistical are available. These data are saved on the device and only if user wishes self can be shared to other devices. Unfortunately we have not implemented direct sending of data to persons doctor due to the legislation restrictions.

Reminder for the taking medicine became one of the big step forward as it was one of the most desired function not only by product owner but users. Here we should admit even though some functionalities were great to implement, but there were still law to follow before all, so direct text messaging with your doctor or professional has been not included to the kit. This function had an extra drawback as the service should be open 24/7 or at least a professional staff that could answer received questions.

As a last requirement from product owner was to display a short tips. That would help users to learn more about the epilepsy through short, but informative text that would be located on main page. To implement this function our team has build up an external server to place all the tips that could be altered if necessary including of newer ones. Unfortunately the Norwegian letters fail to be loaded in right way although all possible solutions are tested including store of language in SQL. The extra features have been accepted for the development at the meeting with supervisor and product owner.

The Telenor Objects M2M communication was a useful implementation that gave us the idea of application integration with external system using its APIs. Unfortunately, although pushing data to server was successful the retrieving data from server failed because of PhoneGap incompatibility with cross-domain scripting.

Accomplishment of Alarm function and sending SMS with GPS location became an extra useful feature for the application giving more seriousness to the development. A simple Alarm function has been incremented before it get an extra expansion. The Alarm includes slide show for first help instruction and implemented an extra functionality - Seizure detection. This powerful and very advanced function has been started at the last

sprint and therefore we could not develop it due to time limit. But we are glad and happy that with the supervision of Patrick Bours and suggestions from Muhammed Derawi we could at least build fundamental solution for the future development. So, seizure detection collects all accelerometer data. Some samples for normal physical activities with different scenarios and seizure have been collected. The EpilepsyApp team wanted to fulfill this feature very much, but unfortunately we found that biometric data analysis and the collection of seizure samples are time consuming development. This factor was main cause that the development has been terminated.

Here should we admit that some of the suggestions from *Datatilsynet* has been taken into consideration and implemented in our application. User has full control over the data and full deletion of entire data including database file systems and files are available. We have not implemented any authorization, so user is responsible self for securing data. We have implemented own defined calendar as Datatilsynet suggest this part more secure for sensitive information. We didn't implement encryption of data as it won't be sent via our application. The folder location for all data saved on mobile is clearly alerted if user initiates any export functions.

As a conclusion for the result we can add the positive response from product owner. It was important when delivered product had satisfied very much expectations of product owner.

6.1.2 Situation

The download of the Application in the Market is increasing. It is very amazing that the beta version of application attracts so many users. It is about 82 active installs and 205 total installs for 22.05.2012 that is a good number for users with that specific symptom. The internationalization of the application into 8 different languages became a factor for more installs from outside of Norway. The feedback from users are positive and some of them are sent to email. That shows care for the project and that users await future updates.

6.2 Future Development

Even if most part of the application is developed there were still some extra features that could be included. We think that the application has a big potential in future development. The scope can be extended even to the Master Degree. Actually one of our member, Adiljan Abdurihim, possibly going to continue the development of the application as Master Degree Thesis.

The extra features like TelenorObjects 4.8.2 and Seizure Detection 4.10.2 could be developed to provide extra powerful functions for users.

The possibility for development extra features like triggers requested by some users was open.

Alarm function could be implemented with an extra functionality where user could activate time period before sending warning SMS. That would be useful if a user goes for a walk assuring that if seizure occurs there a message will be sent to the chosen contacts with the GPS location. Actually, one of our members had a conversation with one person about the possible application features and he was very interested even he hadn't none with the epilepsy syndrome. He would use application for his father who would always forget the way back home. It was nice thing knowing that the application could be used not only by people who have epilepsy.

6.3 Evaluation

6.3.1 Brief

The possible problems for the project development haven't been detected. We expected that the check of the application could hang over for it's liability from *Datatilsynet*. Otherwise we had the normal stream of product development.

6.3.2 Work Organization

We followed group rule and leader was changing every Sprint term. It was a normal routine for us to discuss solution, problem or concerning issues when we were going for break. But most important thing is that sometimes we were working remotely and it was tools like Trello and GoogleDocs where we were leaving messages. We have learned a lot of systematical job performing work. Every day it was kind more habit filling out Progress and Time Log and checking for messages in Trello. It became a motor for progressive development. Assigning fast days for weekly meeting with supervisor and every end of sprint followed by demo presentation with status report delivery. This timetable let us focus on the job performing all tasks at the certain time.

6.3.3 Work Distribution

The work distribution of responsibilities didn't differ from that is told before1.8.4. So, every member was working independently on their feature, but coming to the more advanced implementation the group shared work between.

6.3.4 Subjective Reflection

Adiljan Abdurihim

The whole process was very interesting and challenging. I have learned not only to work in a systematical way and learn things individually but at the same time I have tested the long period group work. Some small fails let me look differently to things done and of course its solution. Although application can be a simple in look, but I believe it does a lot and could do more if only we had more time. Learning new framework and remembering all the taught was nice experience. Of course there are something I would do differently, but I do not regret as they added extra knowledge for the future. As a last thing I would

underline that every sleepless night, hard work and stress over the progress worthed it, as I have seen that people needed this. As a negative part I would say that we hadn't possibility to test our application once more on iOS due to some complications5.1.1. And of course the GUI i would change to something new.

Andrius Januska

Working with this bachelor project was both very interesting and useful for me. At the beginning of the project we have set learning objectives for the project - to learn development of cross-platform applications and to learn system development in a structured way. I think we have reached these objectives and learned even more. We have got acquainted with the technology for cross-platform mobile development, successfully implemented Scrum development methodology for organizing the whole development process and learned to work as a team.

I think we have also reached our project result objectives at a high level, as we have developed a good and useful application. This statement is based on EpilepsyApp users responses that we have received so far and on project initiators feedback. Observing that the number of application users from different parts of the world is increasing every day (we have õ0 active installs per today) makes me to believe that the application is useful to those who need it.

The whole project period we have worked actively in accordance with the group rules that we defined at the beginning of the project. We have attended meetings with project initiator and project supervisor as we have planned, written status reports periodically and kept our project website and blog up to date. It was only minor deviations from the schedule that we have established at the beginning, that was mainly caused by sickness of both team members.

The difficult part of the project for me was a wide range of project scope. Despite the fact that we had prioritized product features at the beginning of the project, it was hard to concentrate only on those features, because other functionality (like seizure detection) seemed to be of a high value for the application, and this could possibly have influenced the quality of more prioritized features.

Another challenge for the project that could be mentioned is making estimates. The things that we thought will not take a long time usually took longer time than we expected. Fortunately, the overall estimation for the project was close to what we have achieved.

6.4 Conclusion

Writing this section and being aware of fulfillment of three years in Gjøvik University College we look back to the whole process. Although, we were not certain what would be the result of application written in PhoneGap, that is a pretty new framework it turned

up with a satisfying result.

All the learned theoretical knowledge has been implemented in our practical work. It was interesting to see how actually the software development methodology works and if all the expectations for incremental development are as we thought.

It was challenging to develop EpilepsyApp. The first steps were slow and difficult as we had to remember many of taught and read a lot of tutorials before starting coding. Beside coding there were some aspects like GUI implementation, we have chosen own GUI design in CSS as tests showed that it was quicker in response. The complicated part of the project were to sew all different functions in one, to make them communicate with each other in a right way. Additionally to this we had to secure the right representation of all the data registered as the analysis would affect in any minor changes.

The process of development has enhanced our ability of developing application for the mobile devices. This Bachelor Thesis has united all of our abilities in implementation, where we had an opportunity to use most of our knowledge in practice, opening newer horizons.

Bibliography

[1] Stokke, R. 2011. Smarttelefon som verktøy til økt mestring og opplevd kontroll over hverdagen for. *Høgskolen i Gjøvik*.

[2] Robert V. Stumpf, L. C. T. 2005. *Object-Oriented System Analysis and Design with UML*. Pearson Prentice Hall.

[3] Eeles P., Strategic Services, I. S. G. What, no supplementary specification? http://www.ibm.com/developerworks/rational/library/3975.html. [Online; accessed 04.03.2012].

[4] Phonegap.com. How phonegap works. [Online; accessed 14.01.2012]. http://phonegap.com/about.

[5] Sourcemaking.com. Singleton design pattern. [Online; accessed 22.02.2012]. http://sourcemaking.com/design_patterns/singleton.

[6] Pfeiffer, D. March, 2011. Developing better phonegap apps. http://floatlearning.com/2011/03/developing-better-phonegap-apps/. Accessed online: 14.01.2012.

[7] MSDN, M. Introduction to web storage. [Online; accessed 14.02.2012]. http://msdn.microsoft.com/en-us/library/cc197062

[8] Wikipedia.org. Ajax. [Online; accessed 01.02.2012]. http://en.wikipedia.org/wiki/Ajax_(programming).

[9] Wikipedia.org. White-box testing. [Online; accessed 14.05.2012]. http://en.wikipedia.org/wiki/White-box_testing.

[10] Parekh, N. Black box testing. [Online; accessed: 12.05.2012]. http://www.buzzle.com/editorials/4-10-2005-68349.asp.

A Pre-project Plan

HØGSKOLEN I GJØVIK

POJECT PLAN
FOR
BACHELOR THESIS

EpilepsyApp

Students:

Adiljan Abdurihim
Andrius Januska

GJØVIK, 2012

TABLE OF CONTENTS

1. GOALS AND RESTRICTIONS

1.1. Background

About 0,5% of people in Norway have epilepsy. About five hundred children in Norway every year get epilepsy symptoms. According to the statistics, 70% of people who have epilepsy are getting better after using one or two different medicines. The remaining 30% have no possibility for the treatment and suffers from cognitive disorders which need daily observation, registration and notification. Lack of registration becomes a significant con in controlling attacks for people with epilepsy. All these factors lead to social isolation and doubt, because of cognitive disorder. Youth with epilepsy who are cared by their parents face significant problems when they move out. Major problem is that every individual with epilepsy has problem in managing their daily life, that includes all necessary registrations, planning and arranging (Stokke R., 2011).

The popularity of smart phones and their functionality has increased significantly the last few years. Affordable prices made smart phones common item in a daily life for many people, especially youth. Smart phone became an inseparable accessory and tool for majority, helping to perform different kinds of tasks using mobile applications.

Taking into account these two aspects - difficulties of people who have epilepsy to manage their daily life and advance of technology, it was suggested by Randi Stokke from Health department at Gjøvik University College, to develop an application, that would help youth with epilepsy to manage their daily life. This application will make a contribution for further research by Randi Stokke, that analyzes how smart phones can increase the quality of life for young people with epilepsy.

1.2. Project objectives

Result objectives

The main goal with this project is to develop and test a multi-platform application that would help young people who have epilepsy to manage their daily life by registering daily activites and information associated with them.

Effect objectives

Often reasons of getting epileptic seizures are forgetting to take medicines, disordered way of life, unbalanced use of alcohol etc. Mobile phone application that will be developed during this project will help people who have epilepsy to better plan their daily activities, take medicines on time and thus will contribute to minimizing the risk for possible seizures.

The logged seizure, medicament usage and other data will help users to see historical data and present it to a doctor for further analysis. According to this data a treatment can be adjusted or improved. In this way, the application will be a useful tool for improving personal care.

Alarm function of the application will be a contributing part for more security and self confidence in social life participation by people with epilepsy. Knowing that they can activate alarm and that all notifications will be reminded, users will minimize own psychological press remembering, frightening to be alone, etc.

The application will be an alternative for existing tools with similar functionality and will minimize costs of use by including several necessary functions united and providing easiness of use . It will also provide a possibility for extensions for interested master grade students taking into account all optional features.

Learning objectives

Working with this project will help the project members, among other things, to learn to develop applications for different mobile platforms, use various mobile phone sensors, analyze data, integrate applications into other systems and export application data.

1.3. Restrictions

Data storage, testing and connecting to external systems will face particular restrictions from Datatilsynnet. The group will be restricted to store and transfer data according to legal regulations. Test users of application have to be chosen randomly, without knowing if they have epilepsy or not. That will give a more predictive analysis of various data.

It also comes a restriction on the technology that can be used to develop applications for multiple platforms. We are restricted on using PhoneGap since this is the only effective technology for creating multi-platform applications for mobile phones.

Some of the features that will be implemented in the application require phone sensors like GPS, accelerometer, camera. Thus the application will not be fully functional on phones that do not have all required features.

Due to limited resources, the application will only be tested on a few types of smart-phones, both Android and iOS. Testing on other types of phones is dependent on what types of smart-phones users that will test the application have.

2. SCOPE

2.1. Project description

The scope of the project is to make an application that could help youth with epilepsy to improve daily life by registering everyday data concerned with epilepsy and provide a better communication between a patient and medical staff.

The project scope is divided into two major parts: the features that must be done to provide a fully functional and useful application that meets requirements from the employer and the features that would extend the functionality and make application even more useful, but are not strict requirements from the employer. The scope is divided into these two groups of features:

1) Major features:
   - diary with possibility to register seizures, medicines taken, moods, menstruation cycle and additional notes;
   - exporting data in graphical presentation to a local PC;
   - reminders for taking medicines;
   - daily epilepsy related tips on the welcome screen.
   - warning contact people;

2) Additional features:
   - sending stored data to an e-mail, preferably in PDF format;
   - seizure detection;
   - integration with Telenor's Shepherd platform;
   - medicines recognition;
   - simulation of P2P connection with external system.

Core functionality of the application will present an easy and understandable solution that will include tips of the day in transitional representation that will be updated from external database. User will be able to navigate easily to the required elements such as registering current condition in diary, and will be able to give an opportunity to register and view data of older dates in a calendar. Registration of medicines from the existing list and manual registration will be able to perform. The ability to define days and time for taking medicines, as well as a dose, will be included. Here we should point out that all registered medicines will be reminded as notification with a snoozing beep sound. The application will also have an alarm function that will activate all the necessary sensors on mobile device, according to user-defined user settings. Alarm activation will enable the application to send SMS warnings, including geo-location, to the chosen contact people. All data that a user registers will be stored in a local database for future

reviews and analysis. The graph representation of the data will be provided with possibility to export it to personal computer.

Conversion of stored data to PDF format file and sending it through email will be as an additional feature and will be developed if a project team meets the time limits.

At this step the alarm function will be upgraded with extra features that will allow a mobile device to detect physical changes like fall and shake by using biometric analysis.

Use of M2M (machine to machine) communication, using Telenor Objectswill be tested for integration of extra devices if user is interested and requires them.

Medicines recognition will be an additional feature that will help users to detect the right type of medication to take, according to the registered medicines timeplan.

The last feature that is estimated as an optional, the application will be able to set peer to peer connection with a computer that will simulate an external network for sharing stored data. This part won't be tested with real health network but can be an relevant feature in future development, if necessary. This part goes more on understanding and learning the communication through secure lines.

2.2 Appraisal

There are, however, some features that will not be able to implement during this bachelor project due to time and restrictions.

The application that will be developed is intended to be a multi-platform application, to reach as many users as possible. However, due to time restrictions, two main platforms, Android and iOS, are prioritized, since they are most popular among users.

Communication with doctors (health systems) needs a high level of security. Due to Norwegian legislation, the process to get permission to implement communication with doctors (health systems) would take too long time to meet the project deadline. Because of this, it was decided not to implement this feature in this project.

The major features are most important to develop for a fully functional application, so project group will concentrate on implementing them first. Consequently, some of the additional features may remain unimplemented. However, the project group predicts to implement at least two of the additional features.

3. PROJECT ORGANISATION

3.1. Responsibilities and roles

Adiljan Abdurihim is responsible for e-mail contacts with various people involved in the project. He is responsible for general process log registration and summarizing them at the end of each Sprint. Web blog design and its update including most of the process is responsibility of Adiljan Abdurihim.

Andrius Januska will be responsible for project website development, updating it with relevant information and maintenance, SVN repository establishment, backing-up and archiving documents and the appearance of the project final report.

The group leader role is shared by both group members as described in "Group rules" (see appendix A). The member playing a role as a group leader for a Sprint is responsible for task management and the procedures to be followed during that Sprint, including writing meeting protocols and status reports. The group leader has also a decision right for a particular Sprint as described in apendix A.

All other roles and responsibilities will be shared by both group members and there will always be collaboration at highest level in all activities of the project.

3.2. Routines and rules in the group

The group members, supervisor and employer came with defined routines that can be slightly changed and cancelled if necessary. Following routines are been defined:

- Weekly work time is about 30 hours, Monday - Thursday from 08:00 - 16:00.
- Daily sprint meeting will be held every morning.
- Every member has to register time log daily at the end of day and a schedule for the next day will be discussed.
- On Mondays - meeting with a supervisor: feedback, discussion, planning a week.
- Every second Thursday (every end of a sprint) we hold meetings with stakeholder: demo the application, getting feedback, planning next sprint.
- On Thursdays it will be submitted status reports to the supervisor.

The group rules and routines are defined in the document "Group rules" (see appendix A).

3.3. Tools

- **PhoneGap:** a HTML5 platform that will allow us to develop application for the different platforms. We will, however, prioritize developing for Android and iOS platforms. Aplication will be developed using PhoneGap version 1.3.0.
- **Trello:** will be used as Scrum management tool for work flow presentation using post-it cards. We have chosen it in order to see clearly all needed and planned tasks. It is simple and has a lot of functionality that can be useful for our project management.
- **LaTeX:** is a document preparation system for high-quality typesetting. We think that it will be a useful tool for efficient production of documentation.
- **Gnuplot:** for the graphical representation we will use Gnuplot integrated with LaTeX.
- **SVN:** will be used to keep version control during bachelor thesis development. We use defined space from GUC for SVN.
- **Wordpress:** is used for both web site and blog.
- **GoogleDocs:** is used for writing and sharing notes.

Some other tools can be used additionally to the list above. All necessary decisions for additional tools will be done at the process of development according to "Group rules" (see appendix A).

4. PLANNING, MONITORING AND REPORTING

4.1. Software development methodology

It is very important to choose a right software development process for a project to be successful. Considering an appropriate software development method we have to take into account the features that are specific for our project.

First of all, the project mainly is oriented according division of work into two parts: we have features that must be developed to have a fully functional application, and we have features that are less prioritized and will be developed in case the project do not exceed the time limits. That is to say, the project has not completely restricted scope.

The project is assigned relatively a short period of time as for two developers and the project team has a little experience in working with such type of projects. In addition, we will work with a new technology that is not well known for us. All this add up to an increased uncertainty in estimating and make the project more risky, and is important to consider an appropriate software development methodology.

For a bachelor thesis it is important to provide a considerable amount of documentation for the whole development process. Because of this, we need to choose a development process that takes into account a well structured execution of work flow and documentation. It is important that both the working application and documentation has to be delivered at the end of bachelor thesis accomplishment.

The application touches unknown topic that specifies on epilepsy, that is why we need a continuous assistance from a third party, Randi Stokke, on this field. It is also very important that steps and functions of the application are tested early in development and repeats continuously during entire development process. Scheduled meetings with the employer on development progress and feedback both from Randi Stokke and us is awaited.

As we are going to develop application in PhoneGap, that is very new for us and it is on its way of progress, we have considered that according to project specifics described above the traditional software methodology is not appropriate for this project even though it has its own pros. Simply, we have to be able to rework fails. That is why agile methodology will be a great solution working on whole process allowing rework rather concentrating fully on documentation that is typical traditional, waterfall, methodology.

We have considered several popular agile development methodologies - XP, IBM Rational Unified Process (RUP) and Scrum are among them.

XP has many attractive features for our project: it has simple rules, is based on teamwork, delivers poduct iteratively. However, XP concentrates much on coding and testing, but according to B.Hudges and M.Cotterel this type of development could lead to very unstructured code, since the code is developed simply to meet existing requirements and not possible future extensions to the application as future requirements tend to be uncertain (Hudges , Cotterell, 2002). Another weakness of XP is lack of documentation. “We do not document excuses for failure because we plan to succeed” claims Don Wells (www.extremeprogramming.org). These weaknesses make XP not appropriate for our project, where both structured development with possibility for future extensions and proper documentation should be provided.

RUP has also been considered as a possible development methodology, because it is a well structured and modern software development process that supports iterative development, is proactive to changing requirements and provides flexibility. However, RUP is not to be used in all kinds of projects, it fits better for large scale projects. We have considered that RUP needs too much time to learn how to use it and this methodology uses much documentation. These factors would cost a lot of time for our project and make it too complex. By the way, the team members need to be expert in their field to develop a software under this methodology (http://www.my-project-management-expert.com). We consider this as significant drawbacks for our bachelor project, so we decided not to use RUP.

We have chosen to use Scrum as our project development framework. This decision was made because, in our mind, this methodology with some adjustments best fits our project specifications. With Scrum we can take advantage of iterative development and deliver new functionality at the end of each sprint. In this way we can test our application early in development process, and continue testing during whole development process each time new features are developed. The feedback we get after testing can then be used to improve the existing or developing new features at the following sprints. In addition to this, with Scrum we can keep a good structure and provide a proper documentation for the development process.

In the following section we describe how Scrum will be implemented in our project.

4.2. Scrum implementation

Scrum has a precise division of roles into Scrum Master, Product Owner and Team. In our project because of lack of personal project leader is acting as Scrum Master. Product Owner are Randi Stokke and Simon McCallum and of course group members, we, are Team who is responsible for the implementation of work.

Scrum suggests sprint that lasts between one week or one month period. As our project is small and awaits continuous collaboration with employer we have decided that two weeks sprint is proper for efficient work.

As we have chosen Scrum for our software development methodology we are going to plan it according the methodology suggestions. We will hold daily 15 minutes meeting, daily scrum, with a group members to plan the work for the day that will be checked and noted at the end of day. Checklist with Scrum questions will be used to see the development progress:

- What have you done since yesterday?
- What are you planning to do today?
- Any impediments/stumbling blocks?

Every Monday we will be having a weekly meeting with supervisor planning work for the week and discussing necessary issues around project.

Every sprint term will start with “sprint planning” meeting with supervisor and group members who will plan new sprint discussing status report submitted at the end of each sprint term including feedback from employer and taking in account “sprint retro-perspective” meeting summary. This kind of work will help us to see our mistakes and solve them leading to the progress, improving ourselves.

Beside Scrum development methodology

After logical and objective discussion we came to the solution that some elements of XP can be used in combination with Scrum. XP is a good solution for pair programming where each of the group will be updated to the changes in code and will collaborate to produce efficient code. We should admit that sometimes some of code will be delivered individually, here we see that pair programming, XP, won’t be followed. The pair programming will be used for advanced functions where codes are more complex and time estimates exceed sprint term for individual performance/accomplishment.

5 ORGANIZATION OF QUALITY ASSURANCE

5.1. Documentation, standard applications and source code

Documentation for source-code will be mainly done as in-line comments. Comments will provide most possible information for others who are interested in continuing this project. Documentation will be provided in English as we would like to share it with entire world, where English is most commonly used language.

5.2. Configuration management

As it important to document our work and code have to be developed by two different people the SVN will be used for version control of our code that can be shared easily and saved safely. The location for storing data provided by Gjøvik University College will be used. The written documentation will be written in LaTeX because of its efficiency. For the work performance and report "Trelol" will be used combining it with GoogleDocs for both time-log and other written documentation. See 3.1.

5.3. Risk assessment

"Epilepsy App" project, like all other software development projects, meets various risks. It is smart to consider and evaluate possible risks that can affect the process of the application development. The table shows probability of occurrence, effects to the process, estimated risk, acceptability and mitigation.

Table 1. Risk assessment

Description	Probability	Effects	Acceptability	Mitigation
Bad time estimates	6	slow overall progress of the project	tolerable	Better planning of sprint
Sickness of team member	3	slow overall project progress	tolerable	Extra work is awaited.
Lack of competence	4	slow sprint accomplishment	tolerable	Tutorials and books
Low number of testers	7	little feedback on application	tolerable	More promotion
Loss of source code	2	slow down the project	tolerable	Make back-ups

Mitigations

Bad time estimates can cause a huge problem in the progress of the project. In order to make good time estimates it will be deeply discussed with supervisor every time we plan the next sprint period taking in account experience from our fails.

As it was stated in “Group Rule” (see Appendix A) every member responsible to fulfill 30 hours of work. So sickness of team member means that he have to fulfill those 30 hours that is out the set period time.

PhaneGap is a new technology used to create an application for different platforms. It is not a course we have here in Gøvik University College that is why the lack of competence is awaited. The solution can be a self study and supervision of Simon McCallum who will help us in understanding and possible solutions. Writing plug-ins for some of functions will be tricky as it had to be written for different platforms and here we will try to use all possible staff in University College for basic understanding or discussion.

Every software needs testing for solving bugs encountered. As application is primarily for people with epilepsy and we can not test on people with epilepsy according to regulations, we have to find volunteers that can be difficult as they are not primary target. This kind of problem will be solved by more promotion.

Loss of source code can be a possibility even though the GUC has a good server. Having idea of any possibilities like machine damage, thief and etc the source code will be taken for back-up, weekly using external hard disk or Dropbox.

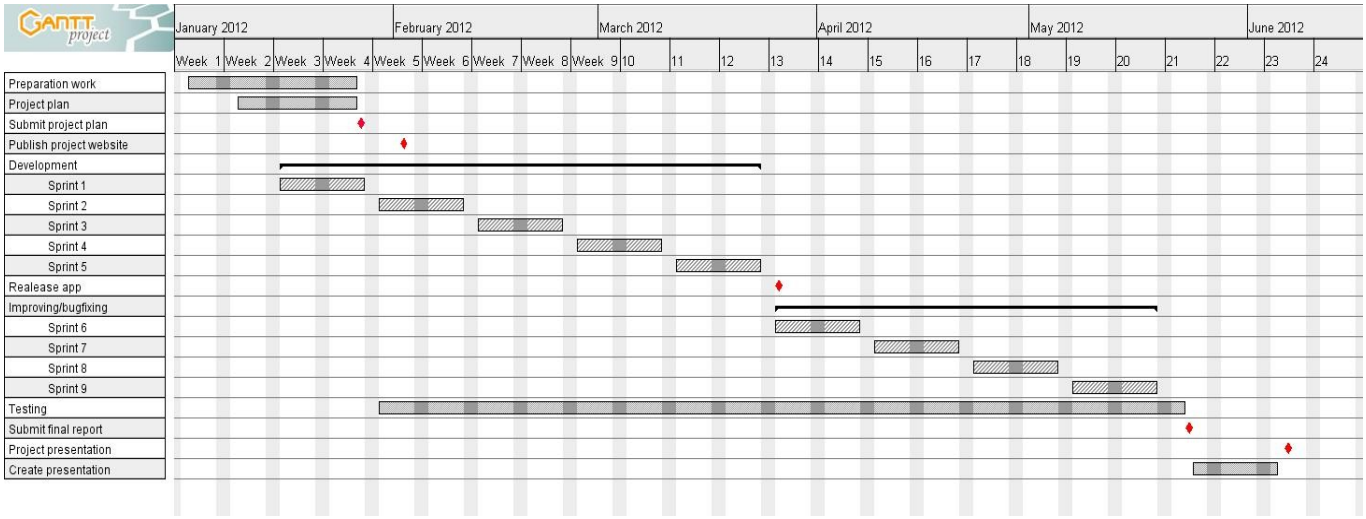
6. PLAN FOR IMPLEMENTATION

As we have chosen Scrum as development methodology for our project, we will work in sprints of two weeks. Before we actually start with development process we will do some preparation work, like writing project plan, establishing project website, SVN repository, blog, installing needed software and doing other preparation tasks. Since the project plan has to be submitted 27 January, we plan to do all the preparation work until then.

From the week 3 we start our first sprint, that will start the development process. We are planning nine sprints of two weeks each for our project. In the first five sprints we will mainly concentrate on developing new features, that will be decided on every sprint planning meeting. At the end of fifth sprint we are planning to make a beta release of the application, that will have most of the functionality we have planned to add. The next four sprints we are planning to concentrate more on fixing bugs and improving the quality of the application. To be able to do this we need some feedback about the application from test users. The testing process will cover whole development process, starting from the second sprint. At the first beginning the application will be tested by the development team and the employer, later, when the application gets more functionality we will start testing it with users. The complete version of the application will be tested after the fifth sprint.

The Gantt chart in the picture bellow represents the overall project implementation plan (see Figure 1):

Figure 1. Gantt Chart for the EpilepsyApp project



As we see from the picture above, there are also marked some milestones with red diamonds. They represent some important deadlines and events in the project.

RESOURCES

1. Stokke R. *Smarttelefon som verktøy til øktmestring og opplevd kontroll over hverdagen for unge voksne med epilepsi*, HiG, 2011.

2. Huges B., Cotterell M. *Software project management.* McGraw-Hill international, 2002.

3. http://www.extremeprogramming.org/values.html - *see. Courage*

4. http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-rup-software-development.html

APPENDIX A

Group Rules for the Bachelor's thesis “EpilepsyApp”

§1 This project has a group leader. Every member will be playing a role of leader in his sprint term see §2.
§2 When there are disagreements in the project, both members present their own substantive views. If the disagreement continues, the stakeholder/employer and the supervisor will be contacted for consultation. If there are still disagreements about the project, it will be given decision right to the leader according to the rolling round role. Rolling round role is where one of member will have will be acting as a group leader and decision right, that will last in period of two weeks. By the beginning of the next Sprint term group leader role will be given to second member. The first sprint term begins on 16.01.2012 at 00:00, policy-orientation given to Andrius Januska. Both team members **must** seek to reach a joint decision, decision right, extra vote, applied only in **exceptional** cases.
§3 Each group member is expected to work on the project about 30 hours per week. Normal working hours for this project is between 8.00 to 16.00, from Monday to Thursday, Friday - project-free day. Each team member is responsible to write a time log. If a member does not perform the agreed work regularly, it should contact the higher powers for possible resolution.
§4 The member who is late for the time as agreed in §3, have to work extra to complete the total 8 hours per day regulation.
§5 In case of illness or the need for permission it must be cautioned early as possible, preferably 24 hours before the permission starts.
§6 If it incurred costs related to the Bachelor's thesis, these should be distributed equally between both group members.
§7 Both group members can sign documents on behalf of the group if it is not violating or leads to negative consequences for the project.

I have read and accepted all the paragraphs in this document.

Adiljan Abdurihim

Andrius Januska

09.01.2012, Gjøvik

B System Requirements Specification Document

B.1 Introduction

B.1.1 Purpose

The purpose of this document is to clarify the specifications and requirements for application that needs to be developed.

B.1.2 Scope

The scope of this document covers functional and supplementary requirements for the application.

B.2 Problem statement

The problem that this application tries to solve is as follows:

- inconvenient registration of daily events to paper-based diary,
- lacking a clear overview of registered data over longer period,
- peoples forgetfulness to take medicines on time,
- peoples who have epilepsy feeling of insecurity when they are alone,
- lack of epilepsy related information.

This problem affects young people with refractory epilepsy in age of18-26 and results in insufficient control of their epilepsy.

A successful solution to this problem would be a mobile application that would help users to maintain a regular lifestyle with enough sleep, controlled alcohol usage, relevant physical activity, healthy food, stress control, regular usage of medicines and avoiding other seizure triggers, and provide an efficient way to register and review epilepsy related data and provide a sense of security when a person is alone.

B.3 Alternative and competing products

There are several products in the worldwide market that have some of the features required, but none of them fully satisfy user needs.

My Epilepsy Diary is an epilepsy diary application designed for Android based devices. It is a very popular application used by many people. However, it is not an ideal application, because it is quite complicated to use and is not very reliable (according to user feedbacks in Google Play). It is not suitable for Norwegian market. First of all, the appli-

cation does not have a Norwegian language. Secondly, it possibly does not comply with Norwegian legislation: the application transfers medical and health information through the network (possibly without implementing strict security requirements) and stores data on a server.

Fallofon is a device that is similar to a phone and is adapted to people who have epilepsy. The function of this device is to detect seizures and alarm closest contacts of a person. The problem is that it is big in size and heavy to carry, it is expensive to use (several subscriptions are needed) and has just the function of recognizing seizures and alarming the contact people.

Potential mobile application by Epilepsiforbundet. Epilepsiforbundet are working on a project of creating an epilepsy application. This application at its first stage will include only the diary functionality and is kind of a “simple” application.

EpDetect is a popular application for automatic seizure detection and alerting persons contacts. However, it is the only functionality of the application.

B.4 Functionality

The functional requirements for the application are described by using a product backlog, high and low levels use cases and use case diagram.

B.4.1 Product Backlog

The product backlog is prepared in collaboration among the product initiator, project supervisor and development team members. It includes main functionality of the application, importance for each product feature and initial estimates done by developers.

ID	Name	Importance	Initial estimate	Notes
1	Diary for registrations	100	30	Seizures, medicines taken, mood, comments and menstruations.
2	Medicine reminder	80	7	Do not stop until registration done.
3	Warning contact people	60	10	Include GPS location.
4	Exporting data to local PC	50	10	Graphics, text, backup data.
5	Daily epilepsy related tips	45	7	With possibility to update with new tips.
6	Seizure detection	40	20	Use accelerometer data.
7	Sending stored data to en email in PDF format	25	4	Clarify with Datatilsynnet about requirements.
8	Integration with Telenor’s Shepherd platform	20	7	-
9	Medicine recognition	18	20	-

Table 2: EpilepsyApp - initial product backlog.

B.4.2 Use Case Diagram

Use case diagram presents all actors that have some interactions with the system and defines what actions users of the system can perform.

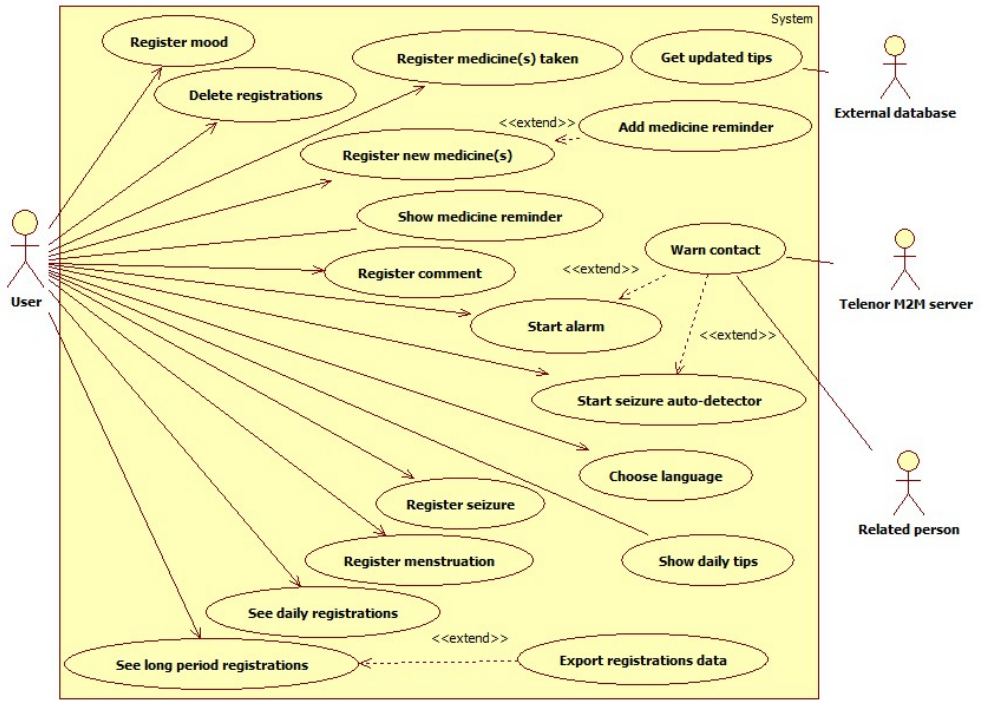


Figure 23: EpilepsyApp - use case diagram.

B.4.3 Use Case Table

Use case table presents all use cases defined for this project for clear review.

No.	Use Case name
1	Register mood
2	Register medicine(s) taken
3	Register seizure
4	Register comment
5	Register menstruation
6	Delete registrations
7	Register new medicine
8	See medicine details
9	Edit medicine details
10	Add medicine reminder
11	See daily registrations
12	See long periond registrations
13	Export registrations data
14	Choose language
15	Start alarm
16	Start seizure autodetector
17	Warn contact
18	Show medicine reminder
19	Show daily tips
20	Get updated tips
21	Collect sensor data

Table 3: EpilepsyApp - use cases table.

B.4.4 High Level Use Cases

High level use cases present users actions against the application and describes how the system has to respond to these actions.

ID	1
Use Case:	Register mood
Actors:	User
Purpose:	To register mood of the user.
Description:	The user wants to register how he feels or how he felt before/ after a seizure. He chooses to register a mood, gets a list of predefined moods and chooses one. The mood is saved into diary. The registration is confirmed.

ID:	2
Use Case:	Register medicine(s) taken
Actors:	User
Purpose:	To register if user has taken medicines or not.
Description:	User selects to register medicine(s) he has to take at a particular time. The list of medicines that user uses is displayed and user can choose one or more medicines and define if a particular medicine was taken or not. When user selects, he can save the registration into diary and medicine reminder will be cleared if it has been set. The registration is confirmed.

ID:	3
Use Case:	Register seizure
Actors:	User
Purpose:	To register a seizure that has just occured.
Description:	User selects to register a seizure that he just had. The list of predefined seizure types is displayed and user can select one. When user selects, he can save the registration into diary. The registration is confirmed.

ID:	4
Use Case:	Register comment
Actors:	User
Purpose:	To register a comment related with epilepsy.
Description:	User selects to register a comment related to epilepsy. The input field for writing a comment is displayed and user can write. User can clear or save the comment at any time. When user is finished writing the commnet, he can save the registration into diary. The registration is confirmed.

ID:	5
Use Case:	Register menstruation
Actors:	User
Purpose:	To register a menstruation if/when it is relevant.
Description:	User selects to register a menstruation. He can select and clear menstruation at any time before he saves the registration. When user has selected, he can save the registration into diary. The registration is confirmed.

ID:	6
Use Case:	Delete registrations
Actors:	User
Purpose:	To delete a particular registration.
Description:	User requires to delete a particular type of registration on a particular day. When the system shows all registrations on a particular day, user can select one and delete it. The registration will be deleted from the diary and user gets confirmation.

ID:	7
Use Case:	Register new medicine
Actors:	User
Purpose:	To register a new medicine that user will use constantly.
Description:	User requires to register a new medicine that he is going to use constantly. The system gives to alternatives: to select medicine from a predefined list or to enter own medicine name. User selects one of the two alternatives and is provided possiblity to enter a dose, time for usage and if reminder for the medicine usage should be activated or not. When user has filled out all required infromation he can save the new medicine. User gets confirmation about registered medicine.

ID:	8
Use Case:	See medicine details
Actors:	User
Purpose:	To show the details of the selected medicine for a user.
Description:	User requires to see the details about a particular medicine. When the system displays the list of medicines, the user selects one and gets all registered times, doses and reminders for a selected medicine. User can choose to exit or to edit the medicine details.

ID:	9
Use Case:	Edit medicine details
Actors:	User
Purpose:	To edit the details for a particular medicine.
Description:	User requires to edit the details about a particular medicine. User first performs the steps from Use Case 8. User chooses to edit details about the selected medicine. User is provided with a possibility to change the dose, time and activate or deactivate the reminder. User makes changes, saves them and gets a confimation message.

ID:	10
Use Case:	Add medicine reminder
Actors:	User
Purpose:	To add a reminder for the selected medicine.
Description:	User requires to add a reminder for a particular medicine he uses constantly. System displays a list of registered medicines and when the user selects one, he will see the information about that medicine. User chooses to set a reminder for a particualar usage time(s). User saves the selection and get confirmation about it.

ID:	11
Use Case:	See daily registrations
Actors:	User
Purpose:	To display the oveview of the registrations of a particular day.
Description:	User requires to see all registrations of a particular day. System provides with a possibility to select a particular day. User chooses one day and gets all the registrations of that day. The user can exit or choose to edit or delete the registrations.

ID:	12
Use Case:	See long period registrations
Actors:	User
Purpose:	To display the overview over registrations of a particular period of time.
Description:	User requires to see the overview of the registrations of a particular month. System displays high-level details about what registrations have been made for each day during the selected period of time. User can toggle between the graphical and non-graphical representation of registrations.

ID:	13
Use Case:	Export registrations data
Actors:	User
Purpose:	To save the registered data from a mobile device to a local PC for a given period of time.
Description:	User requires to export the registrations data from his mobile device to his local PC for a particular period of time. User performs steps from Use Case 12. The user defines the period and selects to export the data. System saves registrations data onto users device both in graphical and text representation.

ID:	14
Use Case:	Choose language
Actors:	User
Purpose:	To give user a possibility to choose between different languages for the application.
Description:	User requires to change the application language to a different one. When the user selects to change the language system displays a list of available languages for the application. User selects required language and the applications language changes accordingly.

ID:	15
Use Case:	Start alarm
Actors:	User
Purpose:	To give user a possibility to inform his contact people about a possible seizure.
Description:	User activates the alarm. After a short period of time the system requires to confirm if the user wants to continue monitoring or to stop. If user chooses to stop the alarm, the alarm will be stopped, if not – it will continue to monitor and repeat the confirmation step in short intervals. If user does not respond to the confirmation message, the system will continue with User Case 17.

ID:	16
Use Case:	Start seizure auto-detector
Actors:	User
Purpose:	To start monitoring users movements in order to detect a seizure.
Description:	User requires the system to monitor for possible seizure. User selects to start seizure auto-detector. System starts collecting sensor infromation and performs seizure detection operations as long as user decides to stop the seizure monitoring or a seizure is detected. If a seizure is detected, system continues with Use Case 17.

ID:	17
Use Case:	Warn contact
Actors:	Users contact(s)
Purpose:	To warn user relatives or other contact people about a seizure.
Description:	System is triggered about users seizure. It composes a warning message including information about users location and sends warning messages to users predefined contacts.

ID:	18
Use Case:	Show medicine reminder
Actors:	User
Purpose:	To remind user about taking medicine at a predefined time.
Description:	System is triggered to show the reminder notification message at a particular time of a day predefined by user. If user responds to the reminder and registers medicine usage, the reminder is cleared. If user does not respond to the reminder, reminder repeats at 5 mins period.

ID:	19
Use Case:	Show daily tips
Actors:	User
Purpose:	To show daily tips related to epilepsy for a user.
Description:	When user starts the application, the system shows a tip of the day. If user requires to close the tip, ir will be closed imediately, if not - the system closes the tip after some period of time.

ID:	20
Use Case:	Get updated tips
Actors:	User
Purpose:	To get new tips from the external system.
Description:	User requires to update current tips collection. System provides with possibility to update tips and user selects to update. If users device is not connected to the network, system warns the user about it. Otherwise, system downloads new tips from external system and informs user about it.

B.4.5 Low Level Use Cases

Low level use cases go into deeper details than high level use cases. It describes alternative flows of events and system reactions to fails. This document contains use cases only for the most complicated use cases defined before.

ID:	7
Use Case:	**Register new medicine**
Actors:	User
Purpose:	To register a new medicine that a user will use constantly.
Overview:	User requires to register a new medicine that he is going to use constantly. He is provided with a possibility to choose medicine from a list or to enter own medicine name. After, user enters a dose, defines time for taking medicine and chooses if a medicine reminder should be enabled or not. When all the details about the medicine are entered user saves it, and the list of all registered medicines is displayed.
Preconditions:	-
Postconditions:	A new medicine will be added to the medicine list.
Special requirements:	-

Flow of events

ACTOR ACTION	SYSTEM RESPONSE
1. This use case begins when a user wants to register a new medicine.	2. Provides a possibility either to choose medicine name from a list or to enter a medicine name.
3. User selects to choose medicine from the list provided.	4. Shows the list of known medicines.
5. User select a medicine name he wishes to add to the medicine list.	6. Provides a possibility to enter medicine details.
7. User enters the dose, sets time and selects to enable reminder for this medicine and confirms details.	8. Displays registered details for the medicine.
8. User saves the medicine.	9. Saves medicine and displays all medicines registered.

Alternative flow of events

Line 3: User chooses to enter own medicine name. Provides possibility for the user to enter medicine name. If user enters medicine name, system proceeds in step 6. If user does not enter medicine name returns to step 2.

Line 7: User enters wrong dose. Indicate error and return to step 6.

Line 7: User wants to enter additional time for taking medicines. Proceed by returning to step 6.

ID:	13
Use Case:	**Export registrations data**
Actors:	User
Purpose:	To provide user with possibility to save registrations data to his PC.
Overview:	User requires to save all registrations data to his local PC. He is provided with a possiblity to select a period of time for which data should be exported. User selects a period and registered data is shown in both graphical and text representation. User then saves data into phones memory card that is easily accesible from his PC.
Preconditions:	It must be registered some data. Mobile phone must have a memory card.
Postconditions:	User gets saved a file with graphical and text representation of data into his mobile devices memory card.
Special requirements:	Graphical representation of data should be clear and of good quality.

Flow of events

ACTOR ACTION	SYSTEM RESPONSE
1. This use case begins when a user wants to export registered data to phones memory card.	
2. User selects a period for which data should be exported.	3. Generates and displays diagram and text for the registered data.
4. User chooses to save data to memory card.	5. Generates a data file and saves it on moblie devices memory card in the applications data directory. Indicates that an export operation is being performed.
	6. When file is generated and saved on the devices memory card, displays a message confirming successful operation with details about a name of the file generated and its location.
7. User reads and closes the message and leaves current activity.	

Alternative flow of events

Line 3: There is no any data registered. Display an informative message. Return to step 2.
Line 5: There is no memory card in users mobile device. Display informative message. Return to step 4.

ID:	15
Use Case:	**Start alarm**
Actors:	User
Purpose:	To give user a possibility to inform his contact people about a possible seizure.
Overview:	User activates the alarm and the system periodically requires to confirm if the user wants to continue monitoring or to stop. If user chooses to stop the alarm, the alarm will be stopped, if not – it will continue to monitor and repeat the confirmation step in short intervals. If user does not respond to the confirmation message, the system will warn users contact people.
Preconditions:	GPS has to be enable if a user wants to send warning message with his location details.
Postconditions:	Users contact people will be warned about users seizure.
Special requirements:	User must be able to start the alarm imediately.

Flow of events

ACTOR ACTION	SYSTEM RESPONSE
1. This use case begins when a user wants to start the alarm function.	
2. User initiates the alarm function.	3. System checks if user has enabled GPS sensors and has selected any contacts for sending a warning message to. If GPS sensors are enabled and at least one contact is selected proceeds in step 4.
	4. After a short period of time system displays a message asking to confirm if user wants to continue monitoring or not, and plays a sound.
5. User does not respond to the message for 30 seconds.	6. Gets users location data.
	7. Sends a warning message to users pre-selected contacts.
	8. Starts making audible alerts to attract passengers' attention and displays the first-help infromation on the screen of the mobile device.

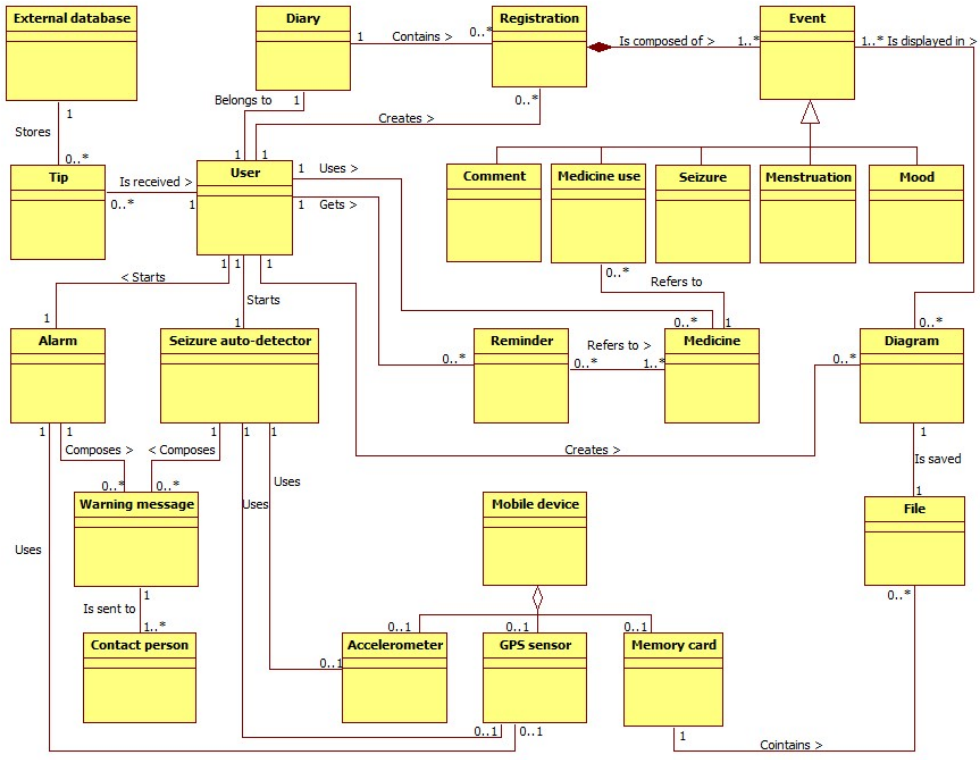
Alternative flow of events

Line 3: GPS is not enabled and/or no contacts are selected. Shows an alert message that GPS should be enabled and/or at least one contact selected.
Line 5: User responds to the message and selects to continue monitoring process. Return to step 4.
Line 5: User responds to the message and selects to stop the monitoring process. The monitoring stops.
Line 6: GPS is not enabled. Proceed in step 7.
Line 7: User has not defined any contacts. Proceed in step 8.
Line 7: User has no money in his account to send a message. Proceed in step 8.
Line 7: Users mobile device is out of mobile signal coverage. Proceed in step 8.

B.4.6 Domain model

The domain model for the application provides a clear view of how different concepts in the application are related to each other and the context of the application.

Figure 24: EpilepsyApp - domain model.



B.5 Supplementary Requirements

This section describes other requirements that are not covered by functional requirement specification.

B.5.1 User tutorial

The application has to have a user tutorial which describes how the application works. The tutorial has to be easy to find and represent the list of all features that are explained, so the user can easily choose the section of interest.

B.5.2 Data backup

The application has to provide the data backup possibility. The user has to be able to take a backup copy of all the registrations he made and save it separately from the application, preferably on another device.

B.5.3 Security

The application will store some health related data. According to Norwegian legislation, health related data has to be properly secured. That applies specially for the data that will be transmitted outside the application. The security requirements are defined taking into the consideration Datatilsynnet's advices and suggestions.

The list of aspects to take into account:

- Application should support high security by default, but let user to decide the level of security.
- Data saved by the application should be encrypted.
- Application should use own PIN code to access it, or use the phone-lock to prevent undesired access.
- User should be able to delete the historical data saved by the application.
- Registered data should be deleted when the application is being uninstalled, or user should clearly know where these data are.
- It should be possible to delete data remotely if f. ex. the mobile device has been stolen.
- It is preferable for application to use its own calendar and reminder functions.
- Sending of e-mail messages should be encrypted or use a password to open a file.

B.5.4 Usability

For a mobile application to be successful and acceptable for users it must have a good usability. There are several things to consider when creating this application: accessibility, user interface aesthetics and consistency within the user interface, understandability and more.

First of all, the application must be easy to understand and learn to use it. It should not take long time for a person who knowns smart phones how to use EpilepsyApp. Use of icons together with text labels is preferable for application controls.

Easy navigation and clear layouts is another thing to have in mind when creating user interfaces for the application. User should always be able to navigate back to where he came from or to go to a start position of the application. It should be possible to do registrations in as few clicks as possible. Alarm function should be accessible from the start point of the application. It is very important for a user to be able to use this function immediately when it is needed.

Layouts of the application elements should be clear and easy to understand, size of elements must be not too small and comply with the size of mobile devices screen.

Application should also respond to all user interactions with it. It has to respond with a visible feedback to each users click on the interactive elements of the application. User must get an informational message when something goes wrong. It should also give a feedback when registrations and operations that take long time are completed. Operations that take a long time should also provide a visual feedback about that something is going on and application is still operating.

B.5.5 Reliability

There are few aspects that are very important considering the reliability of EpilepsyApp. Since the application will store registrations related to medical and health information and will be used as a warning tool when an epileptic seizure occurs, the level of reliability of the application has to be very high.

First of all, the reliability of the data storage has to be at the highest level. People who will use this application will record their seizures and medicine usage history and other information related to their epilepsy. Later this data will be analyzed and used to improve the control of seizures. Any kind of data loosing would distort the information and could lead to making wrong decisions. Because of this, reliability of the stored data has to be very high. The application has to provide user with a possibility to save a backup copy of all registrations made as described in previous section and assure that data will not be deleted accidently. A possibility to recover from a device fail, lost phone, migration to another phone etc. is a key here.

In addition, correctness of the data registered is also very important for later analysis, so the process of saving data has to assure that correct data is being stored. The application has to minimize the risk for a user to save incorrect data and provide a possibility to delete incorrect registrations.

Another reliability issue is related to the alarm and seizure auto-detect functions. Because these functions will be used in critical situations, when a seizure occurs or is likely to occur, it is very important that these functions are available all the time. If the application fails to perform these functions it must inform user about it, so he can use alternative ways to warn his related people.

And finally, a medicine reminder has to be reliable as it performs an important function in helping user to remember to take medicines on time and register their usage. Fail of medicine reminder will not have such serious consequences as in previous cases, but it should be guarantied that user gets reminders for all medicines at a right time.

B.5.6 Performance

The highest requirement for the applications performance refers to the battery usage. The application should be able to perform its functions with minimal usage of battery power. This is specially applicable when application is in alarm and seizure-auto detect modes,

as these functions will require extra battery power for sensors handling. The application should inform user if battery level is less than 15% when he tries to start seizure auto-detect function, as it could drain the battery fast and thus constraint functionality of the mobile device.

Other performance issues, like response to user interactions with the application should be at the similar level as it is normal for other good quality mobile applications. Two slow application would distract users from using it. So if the application is doing some harder work that is taking longer than 3 seconds it should immediately inform user about it, and let user to know that he has to wait a little.

B.5.7 Supportability

The application must have a user manual in it that would help users to find out how to use different functions correctly, as described in previous sections of this chapter.

The application should support multiple languages. Project initiator Randi had an initial requirement to make the application in Norwegian, because a similar application that was on the market did not have support for Norwegian. As we saw a potential for this application to be useful for a wider range of user audience we set a requirement for the application to support English and Norwegian languages at the initial point, with a possibility to increase a number of languages supported. As an additional requirement to this, the application should be able to detect user devices language automatically and apply that language to the application, if it supports that language.

The application should be maintainable in the future. That involves updating the collection of daily tips, medicine lists and other information that is relevant to be updated in the application.

B.5.8 Legal requirements

Because the application will handle medical and health-related information, it has to take into account laws and restrictions related to handling of such information as determined by Norwegian legislation. To comply with these requirements the application should follow advices received from Datatilsynnet.

B.5.9 Partial releases

At the end of every Sprint we should provide a functioning prototype of the application as required by Scrum development methodology. Such kind of iterative development and releases will help to control that the application is being developed according to specifications and contribute to regular testing.

The beta version of the application should be released to the Google Play store after Sprint number 5.

B.6 Constraints

B.6.1 Design constraints

EpilepsyApp is meant to be a cross-platform application. At the first step it should support Android and iOS based mobile devices as they comprise the largest share of smart phones market.

B.6.2 Physical constraints

Devices that will use the application must have accelerometer, GPS sensors and a possibility to send SMS messages for full functionality. Devices that do not have these features will not be able to use seizure alarm and seizure auto-detect functions, but should be able to use other features of the application.

C Documentation for communication with the Datatilsynnet

Letter to Datatilsynnet, 07.02.2012:

English version (Norwegian version is followed below)

Dear Catharina Nes,

We are two third year bachelor students from Gjøvik University College who are developing an mobile application "Epilepsy App" for our bachelor thesis, spring 2012. The project was proposed by Health and Nursing department at Gjøvik University College. While designing the project we have encountered some issues that may need to be considered by Datatilsynet.

The application is targeted to people with epilepsy who are over 18. The application aims to help people with epilepsy manage their every day lives, by registering daily events for self review. All data will be stored in a local database on a mobile device and so will only be used for personal review. Users will be informed before installing the application about what kind of data will be stored on a mobile device and how the data will be processed. Here we would underline that no global storage of data will be used. All the tests of the application will be performed by random people. We have divided the application into different levels and would ask your assistance to clarify if it is OK to develop these features:

Level one:
The user will be able to register their data in a calendar. This includes different moods, seizure types, activities, menstruation cycles, etc. All data will be recorded to the local database on a mobile device for self review.

Level two:
In addition to level one, the user will be able to register a schedule for the medicines they use. A reminder function will inform the user when to take registered medicines. All data will be stored locally on a mobile device for self review.

Level three.
Two first levels are included with an extra feature that will let user to send their data via email to themselves.

We would be grateful if you could assist and ensure we follow the right process. Any useful information and advice will be appreciated.

We would like to know what permissions do we need to implement each of the three levels above. Is there some permission that we can get from "Datatilsynet" for these features. We are also looking to include any email correspondence as an appendix to our thesis, so we are also hoping that you give us permission to include your response in our thesis.

Sinecerly,
Adiljan Abdurihim and Andrius Januska

Norsk versjon

Hei Catharina Nes

Vi er to studenter fra Høyskolen i Gjøvik, som har begynt å utføre sin bachelor oppgave i våren 2012. Vi skal utvikle en applikasjon "EpilepsyApp" for mobiltelefoner. Oppgaven er gitt av seksjon for helse, teknologi og samfunn ved HIG. Gjennom utvikling av prosjektet vi har kommet til noen punkter som vi søker råd og tilbakemelding fra Datatilsynet.

Applikasjonen som skal utvikles i henhold med oppdragsgiveren er tiltenkt mennesker fra og med 18 år. Applikasjonen er ment å hjelpe mennesker som har epilepsi til å planlegge sin hverdag gjennom å kunne registrere nødvendig informasjon. All data som registreres brukes kun for eget oversikt, og lagres lokalt på en mobilenhet. Brukere vil bli varslet før installering av applikasjonen ok hva slags data skal lagres på mobile enheter og hvor dataene blir behandlet. Vi vil understrekke at ingen global data lagring skal brukes. Applikasjonen blir testet av tilfeldige brukere. Vi har delt applikasjonen i tre nivåer og trenger deres råd om hva vi kan utvikle slik at det ikke blir i strid med norske lover.

Nivå èn:
Bruker skal kunne registrere data til kalender på egen mobiltelefon. Dataene som skal registreres er tilstand, mensturasjonssyklus, anfallstype. All data skal lagres lokalt på en mobil enhet for egen oversikt.

Nivå to:
I tillegg til nivå èn, man skal kunne registrere en plan for medisinbruk. Det skal også være en funksjonalitet for påminnelse for medisinering etter registert tidsplan. All data skal lagres lokalt på en mobil enhet for egen oversikt.

Nivå tre:
De to første nivåene skal inkluderes med ekstra funksjonalitet som skal gi mulighet for en bruker å sende registrerte data om seg selv, til seg selv, via e-post.

Vi håper at Dere kan hjelpe oss med å gå en riktig veg. All nyttig informasjon og råd settes en sotr pris på.

Vi trenger Deres tillatelse for å inkludere denne e-post samtalen i vår bachelor prosjektets dokumentasjon.

Mvh,
Adiljan Abdurihim and Andrius Januska
HiG - Høgskolen i Gjøvik

http://hovedprosjekter.hig.no/v2012/imt/in/epilepsiplanlegger/?page_id=20

Letter from Datatilsynnet, 09.02.2012:

Dear Adiljan and Andrius,

Thank you for your email about your research project. It looks very interesting. I will discuss the issues you raise with my colleagues and get back to you as soon as possible. (Would an answer in Norwegian be as good as an answer in English, or do you prefer the latter?)

Best regards,

Catharina

Catharina Nes

Seniorrådgiver

tlf: 22 39 69 06

cane@datatilsynet.no

Letter from Datatilsynnet, 08.03.2012:

From: Catharina Nes <catharina.nes@datatilsynet.no>
Date: 2012/3/8
Subject: VS: Angående APP | About APP
To: "hagadax@gmail.com" <hagadax@gmail.com>
Cc: Helge Veum <helge.veum@datatilsynet.no>

Hei,

Jeg har konferert med min kollega Helge Veum og han har gitt noen kommentarer til løsningen deres, se under.

Vi legger for øvrig med to uttalelser vi har kommet med i forbindelse med henvendelser som gjelder tilsvarende teknologi som den dere utvikler:

· Sak 11/01039 omhandler bruk av hjemmemålinger og ansvar i den sammenhengen. Selv om løsning ikke er lik den dere utvikler er saken trolig interessant for dere.

· Sak 11/00802 gjelder bruk av videotelefoni. Heller ikke helt den samme løsningen som deres, men punktene om forventinger til publikumstjenester er relevant. I begge sakene beskrives hva som forventes der hvor regelverket gjelder.

Vi håper dette var klargjørende. Ta kontakt hvis dere trenger ytterligere informasjon. Lykke til videre med oppgaven!

Vennlig hilsen,

Catharina

Fra: Helge Veum
Sendt: 5. mars 2012 13:20
Til: Catharina Nes

Kopi: Atle Årnes
Emne: SV: Angående APP | About APP

For denne løsningen gjelder regelverket trolig ikke dersom dette er noe som den private selv velger å anskaffe å ta i bruk. Kommentarene må derfor ses som best practice. Uansett bør leverandører legge seg mot dette, og arbeide mot en løsning som er så god at den ville være akseptabel også om den var innenfor regelverket.

Noen punkt inn i mot deres løsning – som da er anbefalinger…

- Dette er trolig privat bruk – og utenfor regelverket (i hvert fall så lenge tas i bruk av den private selv, på eget initiativ, og det ikke kommuniseres mot profesjonelle aktører)

- Appen bør støtte god sikkerhet. Deretter bør det være opp til brukeren selv velge hva han (ikke) tar i bruk (La "sikkerhet" være på som default)

- Krypter innholdet! Ingen grunn til at dette ikke skal gjøres…

- Tilgang til appen? Egen PIN-kode / passord – eller baserer den seg på telefonens lås? Det siste alternativet er svakt, og bør være basert på at brukeren selv velger den svake løsningen.

- Historikk i appen – når slettes det? Har brukeren mulighet for å styre dette.

- Avinstallasjon – sørg for at data også slettes, eller at bruekr er bevist på hvor opplysningen er.

- Hvor trygg er data i appen når telefonen stjeles? Fjernsletting? Binding mot telefonnummer / SIM for autentisering (mer relevant når løsningen er knyttet mot en tjeneste)?

- Bruk av kalender og alarmer - brukes telefonens default-kalender, eller er det en egen løsning i appen? Det siste er klart å foretrekke mht til sikkerhet. Dersom noe dyttes ut i alminnelig kalender, bør det være minimalisert, og evt. linkes inn mot den sikrede appen.

- E-post – Bør krypteres eller unngås! ZIP eller PDF med egetvalgt PWD er den nærmeste løsningen.

Letter to Datatilsynet, 20.04.2012:

Hei igjen,

Først og fremst tusen takk for veiledning og informasjonen som hjalp oss å passe på de noe viktige aspekter.

Vi ville be dere om tillatelse for å bruke e-mail samtale, mellom dere og oss, for å inkludere dem i vår "Final Report" i Bachelor Thesis.

MVH, Adiljan Abdurihim og Andrius Januska

Letter from Datatilsynet, 23.04.2012:

From: Catharina Nes <catharina.nes@datatilsynet.no>
Date: 2012/4/23
Subject: SV: Angående APP | About APP
To: Adiljan Uighur <hagadax@gmail.com>

Hei,

Det går bra å inkludere vår epost i oppgaven deres. Vi ser frem til å lese den.

Vennlig hilsen,

Catharina

D Implementation Examples

D.1 Navigation bar implementation

```
// HTML markup for navigation bar
<div id="footbar">
  <ul>
  </ul>
</div> <!-- /Footbar -->

// CSS styling rules for navigation bar
#footbar {
  position: absolute;
  left: 0;
  bottom: 0;
  width: 100%;
  height: 56px;
  background-color: #000000;
}

#footbar ul {
  display: inline;
    list-style: none outside none;
    height: 100%;
}

#footbar ul li {
  float: left;
  width: 33%;
  display: block;
  height: 100%;
}

#footbar ul li a {
  display: block;
  height: 100%;
  width: 100%;
  margin: 1px;
  color: #FFFFFF;
    text-align: center;
    text-decoration: none;
    border: 1px solid #888888;
}

#footbar ul li a:active, #footbar ul li a.active {
  background-color: #888888;
  color: #FFFFFF;
}

.navIcon {
  height: 32px;
  width: 32px;
  margin-left: auto;
  margin-right: auto;
  background-repeat: no-repeat;
}

#back {
  background-image: url('../images/icons/mdpi/back.png');
}
```

```
#home {
  background-image: url('../images/icons/mdpi/home.png');
}

#save {
  background-image: url('../images/icons/mdpi/save.png');
}

// JavaScrip functions for navigation bar implementation

// Dynamically add navigation buttons to the navigation bar
addNavButton(getString('btn-back'), 'back', 'javascript:goBack();');
addNavButton(getString('btn-home'), 'home', 'index.html');
addNavButton(getString('btn-save'), 'save', 'javascript:saveEvents();');

/*
 * Function addNavButton(label, id, url) is used to put navigation buttons
     to a
 * navigation bar. Buttons will be appended from left to right.
 * label - label of the button
 * id - id of the button
 * url - the url to the page linked or javascript function performed
 */
function addNavButton(label, id, url) {
  var buttonList = document.getElementById('footbar').getElementsByTagName('
      UL')[0];
  var li = document.createElement('LI');
  var a = document.createElement('A');
  a.href = url;
  var div = document.createElement('DIV');
  div.className = "navIcon";
  div.id = id;
  var labelNode = document.createTextNode(label);
  a.appendChild(div);
  a.appendChild(labelNode);
  li.appendChild(a);
  // Buttons use touchstart and touchend event listeners for fast response
  a.addEventListener('touchstart', onButtonTouch, false);
  a.addEventListener('touchend', onButtonTouchEnd, false);
  buttonList.appendChild(li);
  myScroll.refresh();
}

//When a button on a screen is touched changes class to "active"
function onButtonTouch(e) {
  e.currentTarget.className = "active";
}

//When a button is released empties the class.
function onButtonTouchEnd(e) {
  e.currentTarget.className = "";
}
```

D.2 Lists implementation

```
// EXAMPLE SHOWS HOW LIST IS IMPLEMENTED FOR SEIZURE TYPE LIST
// IN ADD EVENT ACTIVITY

// HTML markup for 'seizure list' implementation

// List header
<li>
  <a id="regSeizure" class="longButton" href="javascript:togglePopUp('
      regSeizure', 'seizureTypeList');">
    <div id="seizureIcon" class="icon"></div>
    <p><script>getStringHtml('btn-seizure');</script></p>
  </a>
</li>

// Hidden child list with seizure types
<div id="seizureTypeList" style="display:none" class="popUp">
</div>

// CSS styling rules for lists

#regComment, #regSeizure, #regMedicine, #regMood {
  background-image: url('../images/icons/hdpi/down.png');
  background-repeat: no-repeat;
  background-position: right;
}

.longButton {
    width: 100%;
    height: 100%;
    display: block;
    color: #FFFFFF;
    border: 1px solid #808080;
  text-align: center;
  text-decoration: none;
  padding-top: 2px;
  padding-bottom: 2px;
  background-color: #444444;
}

.longButton.active, .longButton:active {
  background-color: #888888;
  color: #FFFFFF;
  font-weight: bold;
}

.selected {
  background-image: url('../images/icons/mdpi/selected.png');
  background-repeat: no-repeat;
  background-position: right;
}

.cleared {
  background-image: url('../images/icons/mdpi/notselected.png');
  background-repeat: no-repeat;
  background-position: right;
}

.icon {
  height: 48px;
  width: 48px;
  background-repeat: no-repeat;
  position: relative;
  top: 50%;
  margin-top: -24px;
  margin-left: 5px;
  float-left;
```

```
}

.longButton p {
    margin: 0px;
    float: left;
    margin-left: auto;
    margin-right: auto;
    width: 100%;
    display: block;
}

#seizureIcon {
  background-image: url("../images/icons/hdpi/seizure.png");
}

.iconUpDown{
  background-image: url("../images/icons/hdpi/up.png");
  position: right;
}

.popUp {
  display: none;

}

.popUp ul {
    width: 100%;
    display: block;
    list-style-type: none;
    padding: 0;
    margin: 0;
}

.medicine, .mood, .seizure, .comment, .language {
  color: #092B3B;
    border: 1px solid #A0A0A0;
  text-align: left;
  padding-top: 2px;
  padding-bottom: 2px;
  background-color: #8AB5CA;
}

// JavaScript functions for lists implementation

/*
* Function that controls creation of seizure type list
* Seizure types are stored in seizureTypeArray[]
* The same function can be used for creating other lists
* simply replacing data array provided.
*/
function addSeizureTypeList() {
  var seizureTypeList = document.getElementById('seizureTypeList');
  var ul = document.createElement('UL');
  for(var i=0; i<seizureTypeArray.length; i++) {
    var a = document.createElement('A');
    a.className = "longButton seizure";
    a.textContent = seizureTypeArray[i];
    a.value = i+1;
    a.id= "seizure_type_"+a.value;
    a.addEventListener('click', addToScreen, false);
    var li = document.createElement('LI');
    li.appendChild(a);
    ul.appendChild(li);
  }
  seizureTypeList.appendChild(ul);
}
```

```
// Function that unhides a child list
function showPopUp(id) {
  document.getElementById(id).style.display="block";
  myScroll.refresh();
}
// Function that hides a child list
function hidePopUp(id) {
  document.getElementById(id).style.display="none";
  myScroll.refresh();
}

// Function that controls list displaying and hidding
// dependent on its current state
function togglePopUp(headerId, listId) {
  var list = document.getElementById(listId);
  var but = document.getElementById(headerId);
  if(list.style.display=="none") {
    but.style.backgroundImage="url('../images/icons/hdpi/up.png')";
    showPopUp(listId);
    myScroll.scrollToElement(document.getElementById(headerId), 500);
    myScroll.refresh();
  }
  else {
    but.style.backgroundImage="url('../images/icons/hdpi/down.png')";
    hidePopUp(listId);
  }
}
```

D.3 Medicine reminder implementation

EpilepsyAppService.java

```
package no.hig.stud.bachelor.epilepsyapp;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Iterator;

import android.app.AlarmManager;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Binder;
import android.os.Bundle;
import android.os.IBinder;
import android.os.Parcel;
import android.os.RemoteException;
import android.util.Log;
import android.widget.RemoteViews;
import android.widget.Toast;

/**
* MedicineNotificationService class is responsible for composing, displaying
* and repeating medicine notification.
*/
public class MedicineNotificationService extends Service {
  private static final int COL_ID = 0;
  private static final int COL_NAME = 1;
  private static final int COL_DOSE = 2;
  private static final int COL_TIME = 3;
  private static final int COL_DATE = 4;
  private static final int COL_REMINDER = 5;
  private static final int COL_ACTIVE = 6;
```

```
private static final int COL_PID = 7;
private static final int COL_REMINDER_SET = 8;

private static final int REMINDER_TIMEOUT = 10;   // Reminder repeat after
     x minutes
private final static int NEW_REMINDER = 0;    // First time reminder
private final static int UPDATE_REMINDER = 1; // Reminder repeat
private final static CharSequence STICKER_TEXT = "EpilepsyApp medicine
    reminder";
private final static CharSequence NOTIFICATION_TITLE = "Time to take
    medicines!";
private final static String TAG = "EpilepsyAppService";

private NotificationManager notificationManager;
private String EOL = System.getProperty("line.separator");

  /*
   * Called when the service is created. Initialazies the
       NotificationMananger.
   */
    @Override
    public void onCreate() {
      notificationManager = (NotificationManager)getSystemService(Context.
          NOTIFICATION_SERVICE);
    }

    /*
     * Controls the flow of the MedicineNotificationService. Called when
         the
     * servis is started.
     */
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
      // Get data sent to this service
      Bundle receivedData = intent.getExtras();
      // If this service was called to clear notification, then all
      // notifications are cleared and service stopped.
      if(receivedData.getBoolean("clearNotification")) {
        notificationManager.cancelAll();
      }
      else {
        // Saving received data.
        ArrayList<Integer> ids = (ArrayList<Integer>) intent.
            getIntegerArrayListExtra("idArrayList");
        int alarmId = receivedData.getInt("alarmId");
        int reminderType = receivedData.getInt("type");
        // Check and remove ids for medicines that are already taken.
        Iterator<Integer> iterator = ids.iterator();
        while(iterator.hasNext()) {
          Integer id = (Integer) iterator.next();
          if(isMedicineTaken(id) | isReminderCleared(id)) {
            iterator.remove();
          }
        }
        // Register reminders into database for the ids received.
        if(reminderType == NEW_REMINDER && ids.size() > 0) {
          cancelSetRepeatReminders(alarmId+10000);
          setRemindersIntoDatabase(ids);
        }
        // Get all entries that have a reminder set.
        ArrayList<Medicine> entries = getEntriesWithReminderSet();
        // Create and show notification if there are more than 0 entries.

      if(entries.size() > 0) {
          // Create notification message.
          String notificationMessage = createNotificationMessage(entries);
```

```
      // Create a PendingIntent that will launch activity from the
          notification.
      Intent newIntent = new Intent(MedicineNotificationService.this,
          EpilepsyAppActivity.class);
      newIntent.putExtra("sender", "notificationIntent");
      PendingIntent notificationIntent = PendingIntent.getActivity(
          this, 0,
                      newIntent, PendingIntent.FLAG_CANCEL_CURRENT);
      // Show notification.
      showNotification(notificationIntent, notificationMessage,
          entries.size());
      // Schedule reminder repeat after 10 minutes.
      scheduleReminderRepeat(ids, alarmId);
    }
  }
  // Clear the data that was sent to this service.
  receivedData.clear();
  // Stop this service.
    MedicineNotificationService.this.stopSelf(startId);
    return START_REDELIVER_INTENT;
}

/*
 * Returns all medicine entries that have currently reminder set.
 */
private ArrayList<Medicine> getEntriesWithReminderSet() {
  ArrayList<Medicine> dbEntries = new ArrayList<Medicine>();
  Cursor cursor = null;
SQLiteDatabase myDatabase = null;
// Tries to open internal database
try {
  myDatabase = SQLiteDatabase.openDatabase("/data/data/no.hig.stud.
      bachelor.epilepsyapp/app_database/file__0/0000000000000001.db",
      null, Context.MODE_PRIVATE);
  if(myDatabase != null) {
    Log.d(TAG, "Database found: " + myDatabase.toString());
    String sqlQuery = "SELECT * FROM medicines WHERE reminder_set =
        1";
    // Tries to perform a query and get medicines with reminder set.
    try {
      cursor = myDatabase.rawQuery(sqlQuery, null);
      // If there are any reminders set, puts Medicine objects
      // into an ArrayList.
      if(cursor.getCount() > 0){
        cursor.moveToFirst();
        while(!cursor.isAfterLast()) {
          Medicine current = new Medicine();
          current.setId(cursor.getInt(COL_ID));
          current.setName(cursor.getString(COL_NAME));
          current.setDose(cursor.getFloat(COL_DOSE));
          current.setTime(cursor.getString(COL_TIME));
          current.setDate(cursor.getString(COL_DATE));
          current.setReminder(cursor.getString(COL_REMINDER));
          current.setActive(cursor.getInt(COL_ACTIVE));
          current.setPid(cursor.getInt(COL_PID));
          current.setReminderSet(cursor.getInt(COL_REMINDER_SET));
          dbEntries.add(current);
          cursor.moveToNext();
        }
      }
      cursor.close();
     }
     catch (Exception e) {
       e.printStackTrace();
     }
  }
  myDatabase.close();
```

```
    }
    catch (Exception e) {
      e.printStackTrace();
    }
    SQLiteDatabase.releaseMemory();
    return dbEntries;
  }

    /*
     * Creates and returns a message that will be displayed in medicine
         reminder notification.
     */
    private String createNotificationMessage(ArrayList<Medicine> entries)
        {
      String message = "";
      Iterator<Medicine> iterator = entries.iterator();
      while(iterator.hasNext()) {
        Medicine medicine = (Medicine) iterator.next();
        message += medicine.getName() + ", " + medicine.getDose() + "mg" +
             EOL;
      }
      return message;
    }

    /*
     * Schedules repeating of reminder.
     */
    private void scheduleReminderRepeat(ArrayList<Integer> ids, Integer
        alarmId) {
      int newAlarmId;   // New ID for repeated reminder.
      // Defines the restart time after REMINDER_TIMEOUT minutes. Reminder
        // will be started at 1 sec. and 0 millis. of the minute.
        Calendar restartTime = Calendar.getInstance();
      restartTime.add(Calendar.MINUTE, REMINDER_TIMEOUT);
        // We add some delay for repeat reminders so that new reminders
            can take place
        // first and cancel repeat reminders if needed.
        restartTime.set(Calendar.SECOND, 1);
      restartTime.set(Calendar.MILLISECOND, 0);
      // All update reminders have format 2hhmm
      newAlarmId = 20000 + restartTime.get(Calendar.HOUR_OF_DAY)*100 +
          restartTime.get(Calendar.MINUTE);
      // Creates an intent and sends ids of medicines that has to be
          checked
      // to the intent data. Sends reminder type UPDATE.
      Intent newIntent = new Intent(MedicineNotificationService.this,
          MedicineNotificationService.class);
      newIntent.putIntegerArrayListExtra("idArrayList", ids);
      newIntent.putExtra("alarmId", alarmId);
        newIntent.putExtra("type", UPDATE_REMINDER);
      // Creates an AlarmManager object, clears all PendingIntent's with
      // the same information, if there are any, and schedules a new
          reminder.
        AlarmManager am = (AlarmManager)getSystemService(ALARM_SERVICE);
        // We create a new intent with received old reminder ID just for
            canceling it.
        PendingIntent cancelIntent = PendingIntent.getService(this,
            alarmId,
        newIntent, PendingIntent.FLAG_CANCEL_CURRENT);
        am.cancel(cancelIntent);
        // The PendingIntent to launch AddEvent activity if user selects
            this notification.
        // Uses the same intent ID that was used to call this service.
        PendingIntent reminderRepeat = PendingIntent.getService(this,
            newAlarmId,
                        newIntent, PendingIntent.FLAG_CANCEL_CURRENT);
```

```
    am.set(AlarmManager.RTC_WAKEUP, restartTime.getTimeInMillis(),
        reminderRepeat);
}

/*
 * Shows the medicine reminder notification in the notification bar.
 */
private void showNotification(PendingIntent contentIntent, String
    message, int count) {
  // Defines notification icon, sticker text and sets it to current
      time.
  Notification notification = new Notification(R.drawable.notification
      ,
        STICKER_TEXT, System.currentTimeMillis());
  // Implements custom view for the notification.
  RemoteViews contentView = new RemoteViews(this.getPackageName(), R.
      layout.custom_notification_layout);
  contentView.setImageViewResource(R.id.image, R.drawable.notification
      );
  contentView.setTextViewText(R.id.title, NOTIFICATION_TITLE);
  contentView.setTextViewText(R.id.text, message);
  // Defines parameters for notification
  notification.number = count;  // Number that will be displayed in
      the notification.
  notification.contentView = contentView;
  notification.contentIntent = contentIntent;
  notification.defaults |= Notification.DEFAULT_SOUND;
  notification.defaults |= Notification.DEFAULT_VIBRATE;

  // Shows notification.
  notificationManager.notify(R.drawable.notification, notification);
}

/*
 * Registers reminders for the list of medicine ids sent.
 */
private void setRemindersIntoDatabase(ArrayList<Integer> ids){
  // Tries to open a database.
  SQLiteDatabase myDatabase= SQLiteDatabase.openDatabase(
      "/data/data/no.hig.stud.bachelor.epilepsyapp/app_database/
          file__0/0000000000000001.db",
      null, Context.MODE_PRIVATE);
  // If database exists, sets reminders for the entries with sent ids.
  if(myDatabase != null) {
    for(int i=0; i<ids.size(); i++) {
      Log.d(TAG, "Registering reminder for id: " + i);
      Object[] args = { ids.get(i) };
      String sqlQuery = "UPDATE medicines SET reminder_set=1 WHERE id
          =?";
      try {
        myDatabase.execSQL(sqlQuery, args);
       }
       catch (Exception e) {
         e.printStackTrace();
       }
    }
  }
  myDatabase.close();
  SQLiteDatabase.releaseMemory();
}

/*
 * Returns true if medicine entry is already registered in database as
      taken,
 * otherwise returns false.
 */
private boolean isMedicineTaken(int id) {
```

```
    // Conversion from int to String
    String[] args = {String.valueOf(id)};
  Cursor cursor = null;
  boolean taken = false;
  SQLiteDatabase myDatabase = null;
  // Tries to open database.
  try {
    myDatabase= SQLiteDatabase.openDatabase("/data/data/no.hig.stud.
        bachelor.epilepsyapp/app_database/file__0/0000000000000001.db",
        null, Context.MODE_PRIVATE);
    // If database exits checks if medicine is registered in
    // the medicine_usage table in the database with current day.
    if(myDatabase != null) {
      String sqlQuery = "SELECT * FROM medicine_usage WHERE (med_id=?
          AND date=date('NOW'))";
      try {
        cursor = myDatabase.rawQuery(sqlQuery, args);
        Log.d(TAG, "IsMedicineTaken(): ");
        Log.d(TAG, "Total entries from database: "+cursor.getCount());
        // Sets taken to true if there are found entries in the database
            .
        if(cursor.getCount() > 0) {
          taken = true;
        }
        cursor.close();
       }
       catch (Exception e) {
         e.printStackTrace();
       }
      myDatabase.close();
    }
  }
  catch (Exception e) {
    e.printStackTrace();
  }
  SQLiteDatabase.releaseMemory();
  return taken;
}

  /*
   * Returns true if medicine entry is already registered as not-taken,
   * that is, reminder for it has been cleared, otherwise returns false.
   */
  private boolean isReminderCleared(int id) {
    // Conversion from int to String
    String[] args = {String.valueOf(id)};
  Cursor cursor = null;
  boolean cleared = false;
  SQLiteDatabase myDatabase = null;
  // Tries to open database.
  try {
    myDatabase= SQLiteDatabase.openDatabase("/data/data/no.hig.stud.
        bachelor.epilepsyapp/app_database/file__0/0000000000000001.db",
        null, Context.MODE_PRIVATE);
    // If database exits checks if reminder for given medicine id
    // has been cleared.
    if(myDatabase != null) {
      String sqlQuery = "SELECT * FROM medicines WHERE (id=? AND
          reminder_set=2)";
      try {
        cursor = myDatabase.rawQuery(sqlQuery, args);
        // Sets cleared to true if there are found entries in the
            database.
        if(cursor.getCount() > 0) {
          cleared = true;
        }
        cursor.close();
```

```
            }
            catch (Exception e) {
              e.printStackTrace();
            }
           myDatabase.close();
         }
       }
       catch (Exception e) {
         e.printStackTrace();
       }
       SQLiteDatabase.releaseMemory();
       return cleared;
     }

       /*
        * Cancels all reminders of the type UPDATE_REMINDER in the interval
        * of the next hour.
        */
       private void cancelSetRepeatReminders(int alarmId) {
         int cancelUntilId;    // Last reminder ID to be canceled.
         // Checks if repeat reminder is set before 23:00. If so, sets the
             last
         // reminder to be canceled ID to ID corresponding reminder after 1
             hour.
         if((alarmId - 20000) < 2300) {
           cancelUntilId = alarmId + 100;
         }
         // If repeat reminder is after 23:00, removes all reminders set
             until
         // 23:59.
         else {
           cancelUntilId = 22359;
         }
         AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
         // Perform reminder canceling for given IDs.
         for(int i = alarmId; i <= cancelUntilId; i++){
           Intent newIntent = new Intent(MedicineNotificationService.this,
               MedicineNotificationService.class);
           PendingIntent canceledIntent = PendingIntent.getService(this, i,
               newIntent, PendingIntent.FLAG_CANCEL_CURRENT);
           am.cancel(canceledIntent);
         }
         // If last reminder to cancel is at 23:59, restarts this function to
         // remove repeat reminders between 00:00 and 01:00.
         if(cancelUntilId == 22359) { cancelSetRepeatReminders(20000); }
       }
}
```

Reminder.java

```
/**
* Class Reminder is used for storing information about one reminder.
* One Reminder object can contain infromation about several medicines.
*/
public class Reminder {
  private String time;
  private ArrayList<Integer> ids = new ArrayList<Integer>();
  private ArrayList<String> messages = new ArrayList<String>();

  Reminder(String t, int id, String msg) {
    time = t;
    ids.add(id);
    messages.add(msg);
  }

  private void setTime(String t) {
    this.time = t;
  }
```

```
    private void addId(int id) {
      ids.add(id);
    }

    private void addMessage(String msg) {
      messages.add(msg);
    }

    private String getTime() {
      return this.time;
    }

    private ArrayList<Integer> getIds() {
      return this.ids;
    }

    private ArrayList<String> getMessages() {
      return this.messages;
    }
  }
```

Medicine.java

```
package no.hig.stud.bachelor.epilepsyapp;

/**
 *  Class that stores data about one medicine entry.
 */
public class Medicine {
  private Integer id;
  private String name;
  private Float dose;
  private String time;
  private String date;
  private String reminder;
  private Integer active;
  private Integer pid;
  private Integer reminder_set;

  public Medicine() {
    this.id = null;
    this.name = null;
    this.dose = null;
    this.time = null;
    this.date = null;
    this.reminder = null;
    this.active = null;
    this.pid = null;
    this.reminder_set = null;
  }

  public Medicine(Integer id, String name, Float dose, String time, String
      date) {
    this.id = id;
    this.name = name;
    this.dose = dose;
    this.time = time;
    this.date = date;
    this.reminder = "on";
    this.active = 1;
    this.pid = 0;
    this.reminder_set = 0;
  }

  public Integer getId() {
    return id;
  }
```

```
public void setId(Integer id) {
  this.id = id;
}

public String getName() {
  return name;
}

public void setName(String name) {
  this.name = name;
}

public Float getDose() {
  return dose;
}

public void setDose(Float dose) {
  this.dose = dose;
}

public String getTime() {
  return time;
}

public void setTime(String time) {
  this.time = time;
}

public String getDate() {
  return date;
}

public void setDate(String date) {
  this.date = date;
}

public String getReminder() {
  return reminder;
}

public void setReminder(String reminder) {
  this.reminder = reminder;
}

public Integer getActive() {
  return active;
}

public void setActive(Integer active) {
  this.active = active;
}

public Integer getPid() {
  return pid;
}

public void setPid(Integer pid) {
  this.pid = pid;
}

public Integer getReminderSet() {
  return reminder_set;
}

public void setReminderSet(Integer reminder_set) {
  this.reminder_set = reminder_set;
```

```
    }
}
```

MedicineNotificationService.java

```
package no.hig.stud.bachelor.epilepsyapp;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Iterator;

import android.app.AlarmManager;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Binder;
import android.os.Bundle;
import android.os.IBinder;
import android.os.Parcel;
import android.os.RemoteException;
import android.util.Log;
import android.widget.RemoteViews;
import android.widget.Toast;

/**
* MedicineNotificationService class is responsible for composing, displaying
* and repeating medicine notification.
*/
public class MedicineNotificationService extends Service {
  private static final int COL_ID = 0;
  private static final int COL_NAME = 1;
  private static final int COL_DOSE = 2;
  private static final int COL_TIME = 3;
  private static final int COL_DATE = 4;
  private static final int COL_REMINDER = 5;
  private static final int COL_ACTIVE = 6;
  private static final int COL_PID = 7;
  private static final int COL_REMINDER_SET = 8;

  private static final int REMINDER_TIMEOUT = 10;   // Reminder repeat after
       x minutes
  private final static int NEW_REMINDER = 0;    // First time reminder
  private final static int UPDATE_REMINDER = 1; // Reminder repeat
  private final static CharSequence STICKER_TEXT = "EpilepsyApp medicine
      reminder";
  private final static CharSequence NOTIFICATION_TITLE = "Time to take
      medicines!";
  private final static String TAG = "EpilepsyAppService";

  private NotificationManager notificationManager;
  private String EOL = System.getProperty("line.separator");

    /*
     * Called when the service is created. Initialazies the
         NotificationMananger.
     */
      @Override
      public void onCreate() {
        notificationManager = (NotificationManager)getSystemService(Context.
            NOTIFICATION_SERVICE);
      }

      /*
```

```
 * Controls the flow of the MedicineNotificationService. Called when
     the
 * servis is started.
 */
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
  // Get data sent to this service
  Bundle receivedData = intent.getExtras();
  // If this service was called to clear notification, then all
  // notifications are cleared and service stopped.
  if(receivedData.getBoolean("clearNotification")) {
    notificationManager.cancelAll();
  }
  else {
    // Saving received data.
    ArrayList<Integer> ids = (ArrayList<Integer>) intent.
        getIntegerArrayListExtra("idArrayList");
    int alarmId = receivedData.getInt("alarmId");
    int reminderType = receivedData.getInt("type");
    // Check and remove ids for medicines that are already taken.
    Iterator<Integer> iterator = ids.iterator();
    while(iterator.hasNext()) {
      Integer id = (Integer) iterator.next();
      if(isMedicineTaken(id) | isReminderCleared(id)) {
        iterator.remove();
      }
    }
    // Register reminders into database for the ids received.
    if(reminderType == NEW_REMINDER && ids.size() > 0) {
      cancelSetRepeatReminders(alarmId+10000);
      setRemindersIntoDatabase(ids);
    }
    // Get all entries that have a reminder set.
    ArrayList<Medicine> entries = getEntriesWithReminderSet();
    // Create and show notification if there are more than 0 entries.

  if(entries.size() > 0) {
      // Create notification message.
      String notificationMessage = createNotificationMessage(entries);
      // Create a PendingIntent that will launch activity from the
          notification.
      Intent newIntent = new Intent(MedicineNotificationService.this,
          EpilepsyAppActivity.class);
      newIntent.putExtra("sender", "notificationIntent");
      PendingIntent notificationIntent = PendingIntent.getActivity(
          this, 0,
                        newIntent, PendingIntent.FLAG_CANCEL_CURRENT);
      // Show notification.
      showNotification(notificationIntent, notificationMessage,
          entries.size());
      // Schedule reminder repeat after 10 minutes.
      scheduleReminderRepeat(ids, alarmId);
    }
  }
  // Clear the data that was sent to this service.
  receivedData.clear();
  // Stop this service.
    MedicineNotificationService.this.stopSelf(startId);
    return START_REDELIVER_INTENT;
}

/*
 * Returns all medicine entries that have currently reminder set.
 */
private ArrayList<Medicine> getEntriesWithReminderSet() {
  ArrayList<Medicine> dbEntries = new ArrayList<Medicine>();
  Cursor cursor = null;
```

```
  SQLiteDatabase myDatabase = null;
  // Tries to open internal database
  try {
    myDatabase = SQLiteDatabase.openDatabase("/data/data/no.hig.stud.
        bachelor.epilepsyapp/app_database/file__0/0000000000000001.db",
        null, Context.MODE_PRIVATE);
    if(myDatabase != null) {
      Log.d(TAG, "Database found: " + myDatabase.toString());
      String sqlQuery = "SELECT * FROM medicines WHERE reminder_set =
          1";
      // Tries to perform a query and get medicines with reminder set.
      try {
        cursor = myDatabase.rawQuery(sqlQuery, null);
        // If there are any reminders set, puts Medicine objects
        // into an ArrayList.
        if(cursor.getCount() > 0){
          cursor.moveToFirst();
          while(!cursor.isAfterLast()) {
            Medicine current = new Medicine();
            current.setId(cursor.getInt(COL_ID));
            current.setName(cursor.getString(COL_NAME));
            current.setDose(cursor.getFloat(COL_DOSE));
            current.setTime(cursor.getString(COL_TIME));
            current.setDate(cursor.getString(COL_DATE));
            current.setReminder(cursor.getString(COL_REMINDER));
            current.setActive(cursor.getInt(COL_ACTIVE));
            current.setPid(cursor.getInt(COL_PID));
            current.setReminderSet(cursor.getInt(COL_REMINDER_SET));
            dbEntries.add(current);
            cursor.moveToNext();
          }
        }
        cursor.close();
       }
       catch (Exception e) {
         e.printStackTrace();
       }
    }
    myDatabase.close();
  }
  catch (Exception e) {
    e.printStackTrace();
  }
  SQLiteDatabase.releaseMemory();
  return dbEntries;
}

  /*
   * Creates and returns a message that will be displayed in medicine
       reminder notification.
   */
  private String createNotificationMessage(ArrayList<Medicine> entries)
      {
    String message = "";
    Iterator<Medicine> iterator = entries.iterator();
    while(iterator.hasNext()) {
      Medicine medicine = (Medicine) iterator.next();
      message += medicine.getName() + ", " + medicine.getDose() + "mg" +
           EOL;
    }
    return message;
  }

  /*
   * Schedules repeating of reminder.
   */
```

```
private void scheduleReminderRepeat(ArrayList<Integer> ids, Integer
    alarmId) {
  int newAlarmId;   // New ID for repeated reminder.
  // Defines the restart time after REMINDER_TIMEOUT minutes. Reminder
    // will be started at 1 sec. and 0 millis. of the minute.
    Calendar restartTime = Calendar.getInstance();
  restartTime.add(Calendar.MINUTE, REMINDER_TIMEOUT);
    // We add some delay for repeat reminders so that new reminders
        can take place
    // first and cancel repeat reminders if needed.
    restartTime.set(Calendar.SECOND, 1);
  restartTime.set(Calendar.MILLISECOND, 0);
  // All update reminders have format 2hhmm
  newAlarmId = 20000 + restartTime.get(Calendar.HOUR_OF_DAY)*100 +
      restartTime.get(Calendar.MINUTE);
  // Creates an intent and sends ids of medicines that has to be
      checked
  // to the intent data. Sends reminder type UPDATE.
  Intent newIntent = new Intent(MedicineNotificationService.this,
      MedicineNotificationService.class);
  newIntent.putIntegerArrayListExtra("idArrayList", ids);
  newIntent.putExtra("alarmId", alarmId);
    newIntent.putExtra("type", UPDATE_REMINDER);
  // Creates an AlarmManager object, clears all PendingIntent's with
  // the same information, if there are any, and schedules a new
      reminder.
    AlarmManager am = (AlarmManager)getSystemService(ALARM_SERVICE);
    // We create a new intent with received old reminder ID just for
        canceling it.
    PendingIntent cancelIntent = PendingIntent.getService(this,
        alarmId,
    newIntent, PendingIntent.FLAG_CANCEL_CURRENT);
    am.cancel(cancelIntent);
    // The PendingIntent to launch AddEvent activity if user selects
        this notification.
    // Uses the same intent ID that was used to call this service.
    PendingIntent reminderRepeat = PendingIntent.getService(this,
        newAlarmId,
                    newIntent, PendingIntent.FLAG_CANCEL_CURRENT);
    am.set(AlarmManager.RTC_WAKEUP, restartTime.getTimeInMillis(),
        reminderRepeat);
}

/*
 * Shows the medicine reminder notification in the notification bar.
 */
private void showNotification(PendingIntent contentIntent, String
    message, int count) {
  // Defines notification icon, sticker text and sets it to current
      time.
  Notification notification = new Notification(R.drawable.notification
      ,
        STICKER_TEXT, System.currentTimeMillis());
  // Implements custom view for the notification.
  RemoteViews contentView = new RemoteViews(this.getPackageName(), R.
      layout.custom_notification_layout);
  contentView.setImageViewResource(R.id.image, R.drawable.notification
      );
  contentView.setTextViewText(R.id.title, NOTIFICATION_TITLE);
  contentView.setTextViewText(R.id.text, message);
  // Defines parameters for notification
  notification.number = count;  // Number that will be displayed in
      the notification.
  notification.contentView = contentView;
  notification.contentIntent = contentIntent;
  notification.defaults |= Notification.DEFAULT_SOUND;
  notification.defaults |= Notification.DEFAULT_VIBRATE;
```

```
  // Shows notification.
  notificationManager.notify(R.drawable.notification, notification);
}

/*
 * Registers reminders for the list of medicine ids sent.
 */
private void setRemindersIntoDatabase(ArrayList<Integer> ids){
  // Tries to open a database.
  SQLiteDatabase myDatabase= SQLiteDatabase.openDatabase(
      "/data/data/no.hig.stud.bachelor.epilepsyapp/app_database/
          file__0/0000000000000001.db",
      null, Context.MODE_PRIVATE);
  // If database exists, sets reminders for the entries with sent ids.
  if(myDatabase != null) {
    for(int i=0; i<ids.size(); i++) {
      Log.d(TAG, "Registering reminder for id: " + i);
      Object[] args = { ids.get(i) };
      String sqlQuery = "UPDATE medicines SET reminder_set=1 WHERE id
          =?";
      try {
        myDatabase.execSQL(sqlQuery, args);
       }
       catch (Exception e) {
         e.printStackTrace();
       }
    }
  }
  myDatabase.close();
  SQLiteDatabase.releaseMemory();
}

/*
 * Returns true if medicine entry is already registered in database as
      taken,
 * otherwise returns false.
 */
private boolean isMedicineTaken(int id) {
  // Conversion from int to String
  String[] args = {String.valueOf(id)};
Cursor cursor = null;
boolean taken = false;
SQLiteDatabase myDatabase = null;
// Tries to open database.
try {
  myDatabase= SQLiteDatabase.openDatabase("/data/data/no.hig.stud.
      bachelor.epilepsyapp/app_database/file__0/0000000000000001.db",
      null, Context.MODE_PRIVATE);
  // If database exits checks if medicine is registered in
  // the medicine_usage table in the database with current day.
  if(myDatabase != null) {
    String sqlQuery = "SELECT * FROM medicine_usage WHERE (med_id=?
        AND date=date('NOW'))";
    try {
      cursor = myDatabase.rawQuery(sqlQuery, args);
      Log.d(TAG, "IsMedicineTaken(): ");
      Log.d(TAG, "Total entries from database: "+cursor.getCount());
      // Sets taken to true if there are found entries in the database
          .
      if(cursor.getCount() > 0) {
        taken = true;
      }
      cursor.close();
     }
     catch (Exception e) {
       e.printStackTrace();
```

```
        }
        myDatabase.close();
      }
    }
    catch (Exception e) {
      e.printStackTrace();
    }
    SQLiteDatabase.releaseMemory();
    return taken;
  }

    /*
     * Returns true if medicine entry is already registered as not-taken,
     * that is, reminder for it has been cleared, otherwise returns false.
     */
    private boolean isReminderCleared(int id) {
      // Conversion from int to String
      String[] args = {String.valueOf(id)};
    Cursor cursor = null;
    boolean cleared = false;
    SQLiteDatabase myDatabase = null;
    // Tries to open database.
    try {
      myDatabase= SQLiteDatabase.openDatabase("/data/data/no.hig.stud.
          bachelor.epilepsyapp/app_database/file__0/0000000000000001.db",
          null, Context.MODE_PRIVATE);
      // If database exits checks if reminder for given medicine id
      // has been cleared.
      if(myDatabase != null) {
        String sqlQuery = "SELECT * FROM medicines WHERE (id=? AND
            reminder_set=2)";
        try {
          cursor = myDatabase.rawQuery(sqlQuery, args);
          // Sets cleared to true if there are found entries in the
              database.
          if(cursor.getCount() > 0) {
            cleared = true;
          }
          cursor.close();
        }
        catch (Exception e) {
          e.printStackTrace();
        }
        myDatabase.close();
      }
    }
    catch (Exception e) {
      e.printStackTrace();
    }
    SQLiteDatabase.releaseMemory();
    return cleared;
  }

    /*
     * Cancels all reminders of the type UPDATE_REMINDER in the interval
     * of the next hour.
     */
    private void cancelSetRepeatReminders(int alarmId) {
      int cancelUntilId;    // Last reminder ID to be canceled.
      // Checks if repeat reminder is set before 23:00. If so, sets the
          last
      // reminder to be canceled ID to ID corresponding reminder after 1
          hour.
      if((alarmId - 20000) < 2300) {
        cancelUntilId = alarmId + 100;
      }
```

```
          // If repeat reminder is after 23:00, removes all reminders set
              until
          // 23:59.
          else {
            cancelUntilId = 22359;
          }
          AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
          // Perform reminder canceling for given IDs.
          for(int i = alarmId; i <= cancelUntilId; i++){
            Intent newIntent = new Intent(MedicineNotificationService.this,
                MedicineNotificationService.class);
            PendingIntent canceledIntent = PendingIntent.getService(this, i,
                newIntent, PendingIntent.FLAG_CANCEL_CURRENT);
            am.cancel(canceledIntent);
          }
          // If last reminder to cancel is at 23:59, restarts this function to
          // remove repeat reminders between 00:00 and 01:00.
          if(cancelUntilId == 22359) { cancelSetRepeatReminders(20000); }
        }
}
```

D.4 Data graphical export implementation

```
/**
 * FUNCTIONS IN DIAGRAM.JS ARE USED TO FORM A GRAPH REPRESENTATION FOR
     REGISTERED
 * DATA. GRAPH DRAWING IS IMPLEMENTED WITH HELP OF FLOTR2 LIBRARY.
 * http://humblesoftware.com/flotr2/
 */

var html;
var homeDir = "EpilepsyApp";
var filename;
var max = 3;
var activeMed;

/**
 * Function that will be started when the activity loads up. Calls other
     functions
 * that initiates the layout and retrieves required data.
 */
function onGraphLoad() {
  addScroller();
  addNavButton(getString('btn-back'), 'back', 'javascript:goBack();');
  addNavButton(getString('btn-home'), 'home', 'index.html');
  addNavButton(getString('btn-save-to-file'), 'save', 'javascript: saveGraph
      ();');
  var db = openDb();
  db.transaction(getActiveMedicinesFromDb, errorDb, successDb);
  db.transaction(getMoodsFromDb, errorDb, successDb);
  db.transaction(getEntriesFromDb, errorDb, drawGraph);
}

/**
 * Retrieve all moods from the database for required month
 */
function getMoodsFromDb(tx) {
  var today = new Date(window.localStorage.getItem("currentDate"));
  var year = today.getFullYear();
  var month = today.getMonth()+1;
  var firstDay = year+"-"+(month>9?"":"0")+month+"-01";
  var lastDay = year+"-"+(month>9?"":"0")+month+"-"+getNumberOfDays(year,
      month-1);

  var createMoodLevelArray = function(tx, result) {
    var days = [];
    for(var i=0;i<result.rows.length;i++) {
      var day= (result.rows.item(i).date).substr(-2, 2);
```

```
      var moodLevel = evaluateMood(result.rows.item(i).mood);
      if(days[day-1] == undefined) {
        days[day-1] = 1;
        dbData[5][day-1] = moodLevel;
      }
      else {
        days[day-1]++;
        dbData[5][day-1] = (dbData[5][day-1]+moodLevel)/days[day-1];
      }
    }
  }
  tx.executeSql('SELECT * FROM moods WHERE (date >= ? AND date <= ?)', [
      firstDay, lastDay], createMoodLevelArray, errorDb);
}

/**
 * Function evaluates mood level.
 */
function evaluateMood(mood) {
  var moodLevel;
  switch(mood) {
    case "Angry":
      moodLevel = 30;
      break;
    case "Bored":
      moodLevel = 60;
      break;
    case "Freaked":
      moodLevel = 30;
      break;
    case "Frustrated":
      moodLevel = 10;
      break;
    case "Happy":
      moodLevel = 90;
      break;
    case "Hyper":
      moodLevel = 100;
      break;
    case "Lonely":
      moodLevel = 50;
      break;
    case "Said":
      moodLevel = 30;
      break;
    case "Stressed":
      moodLevel = 20;
      break;
    case "Tired":
      moodLevel = 50;
      break;
    default: moodLevel=0;
  }
  return moodLevel;
}

/**
 * Retrieve all active medicine from the database.
 */
function getActiveMedicinesFromDb(tx) {
  var setActiveMedicines = function(tx, result) {
    activeMed = result.rows.length;
  };
  tx.executeSql('SELECT * FROM medicines WHERE active=1', [],
      setActiveMedicines, errorDb);
}
```

```
/**
 * Function that draws a graph.
 */
function drawGraph() {
  var container = document.getElementById("container");
  var seizureArray = [];
  var medicineArray = [];
  var moodArray = [];
  var menstruationArray = [];
  var activeMedicinesArray = [];
  var xTicks = [];
  var date = new Date(window.localStorage.getItem('currentDate'));
  var year = date.getFullYear();
  var month = date.getMonth();
  var numberOfDays=getNumberOfDays(year, month);
  for(var i=1; i<= numberOfDays; i++) {
    if(((i+1) % 2) == 0) {
      xTicks.push(i);
    }
  }
  for(var i=0; i<numberOfDays;i++) {
    if(dbData[1][i] == 0) {
      var seizure = null;
    }
    else {
      var seizure = dbData[1][i];
    }

    if(seizure > max) {
      max = seizure;
    }
    seizureArray.push([i+1, seizure]);

    if(dbData[4][i] > max) {
      max = dbData[4][i];
    }
    medicineArray.push([i+1, dbData[4][i]]);

    moodArray.push([i+1, dbData[5][i]]);
    menstruationArray.push([i+1, dbData[3][i]]);
    activeMedicinesArray.push([i+1, activeMed]);
  }
  var seizures = {
            data: seizureArray,
            points: {
              show: true,
              radius: 5,
              fill: true,
              fillColor: 'red'
            },
            label: getString('lbl-seizures')
    }
  var medicines = {
      data: medicineArray,
      lines: {
        lineWidth: 1,
        show: true
      },
      label: getString('lbl-medicine-usage')
  }
  var moods = {
      data: moodArray,
      bars: {
        show: true,
        lineWidth: 1,
        barWidth: 1
      },
```

```
        label: getString('lbl-mood-level'),
        yaxis: 2
    }
    var activeMedicines = {
        data: activeMedicinesArray,
        lines: {
          lineWidth: 1,
          show: true
        },
        label: getString('lbl-prescribed-medicines')
    }
     var graph = Flotr.draw(container, [activeMedicines, moods , medicines,
          seizures], {
       colors: ['yellow', 'lime', 'pink', 'red'],
       title: monthNames[month]+", "+year,
       parseFloat: false,
           xaxis: {
             ticks: xTicks
           },
       yaxis: {
         max: max+2,
             minorTicks: null,
             tickDecimals: 0,
             title: getString('lbl-number-of-medicines-and-seizures'),
           },
           y2axis: {
            min: 0,
            max: 100,
            title: getString('lbl-mood-level'),
            ticks: [
                      [0, getString('lbl-very-bad')], [50, getString('lbl-
                          average')], [100, getString('lbl-very-good')]
              ]
           },
           grid: {
             verticalLines: false,
             horizontalLines: false
           },
           HtmlText: false
       });
}

/**
 * Function used to save the graph into a HTML file.
 */
function saveGraph() {
  document.getElementById('loading').style.visibility = "visible";
  var container = document.getElementById("container");
  setTimeout(function() {
    html = '<html><body>'+container.innerHTML+'<img style="margin-top:-5px;
        margin-left: -5px" src="'+container.firstChild.toDataURL()+'"></img
        ></body></html>';
  }, 10);
  getHomeDirectory();
}

/**
 * Get root directory of a device.
 */
function getHomeDirectory() {
  var gotRootDir = function(rootDir) {
    rootDir.getDirectory(homeDir, {create: true, exclusive: false}, gotFS,
        fail);
  }
  window.resolveLocalFileSystemURI("file:///sdcard/", gotRootDir, fail);
}
```

```
/**
 * Create a file.
 */
function gotFS(homeDir) {
  var date = new Date(window.localStorage.getItem("currentDate"));
  var year = date.getFullYear();
  var month = monthNames[date.getMonth()];
  filename = "diagram_"+month+"_"+year+".html";
  homeDir.getFile(filename, {create: true, exclusive: false}, gotFileEntry,
      fail);
}

/**
 * Create a file writer.
 */
function gotFileEntry(fileEntry) {
    fileEntry.createWriter(gotFileWriter, fail);
}

/**
 * Write data to file.
 */
function gotFileWriter(writer) {
    writer.onwriteend = function(evt) {
      document.getElementById('loading').style.visibility = 'hidden';
      alert(getString('alrt-graph-saved') + ' /sdcard/EpilepsyApp/' +
          filename);
    }
    writer.write(html);
}
```

D.5 Backup and restore implementation

backup.js

```
/**
 * BACKUP.JS IMPLEMETS FUNCTIONALITY TO CREATING A BACKUP COPY OF ALL
     REGISTRATIONS
 * AND STORING ON DEVICES SDCARD.
 */
  var homeDir = "EpilepsyApp";
  var subhomeDir="BackUps";
  var data=new Array();
  var filename;
  var wr;
  var j=0;
/**
 * Function runs on the activity load and gets a root directory on a memory
     card.
 */
    function onDeviceReady() {
      var gotRootDir = function(rootDir) {
        rootDir.getDirectory(homeDir, {create: true, exclusive: true},
            createFolder, fail);
      }
      window.resolveLocalFileSystemURI("file:///sdcard/", gotRootDir, fail);
     }
/**
 * Create a backup directory in EpilepsyApp home directory.
 */
    function createFolder(){
      var gotRootDir = function(rootDir) {
        rootDir.getDirectory(subhomeDir, {create: true, exclusive: true},
            gotFS, fail);
      }
      window.resolveLocalFileSystemURI("file:///sdcard/EpilepsyApp/",
          gotRootDir, fail);
    }
```

```
/**
 * Create a file for writing backup data.
 */
    function gotFS(subhomeDir) {
      var date = new Date();
      var year = date.getFullYear();
      var month = date.getMonth()+1;
      var day = date.getDate();
      filename = "backup_"+day+"_"+month+"_"+year+".txt";
      subhomeDir.getFile(filename, {create: true, exclusive: false},
          gotFileEntry, fail);
    }

/**
 * Create a file writer.
 */
    function gotFileEntry(fileEntry) {
        fileEntry.createWriter(gotFileWriter, fail);
    }

/**
 * Retrieve data from database.
 */
    function gotFileWriter(writer) {
      var db = openDb();
      wr = writer;
      db.transaction(getAllEntries, fail);
    }

/**
 * Write data to file.
 */
    function copy(writer, tex){
        writer.onwriteend = function(evt) {
          alert(getString("backup-taken")+"\/"+filename);
          window.location="settings.html";
        }
        writer.write(tex);
    };

/**
 * Get all entries from the database.
 */
    function getAllEntries(tx) {
      tx.executeSql('SELECT * FROM medicines', [], showAllMedicines, errorDb
          );
      tx.executeSql('SELECT * FROM seizures', [], showAllSeizures, errorDb);
      tx.executeSql('SELECT * FROM moods', [], showAllMoods, errorDb);
      tx.executeSql('SELECT * FROM medicine_usage', [],
          showAllMedicinesUsage, errorDb);
      tx.executeSql('SELECT * FROM menstruations', [], showAllMenstruations,
           errorDb);
      tx.executeSql('SELECT * FROM comments', [], showAllComments, errorDb);
    }

/**
 * Save medicine entries into an array.
 */
    function showAllMedicines(tx, result) {
        for(var i=0; i<result.rows.length;i++) {
          data[j]=" dora: "+result.rows.item(i).id +" "+ result.rows.item(i)
              .name+" :end "+ result.rows.item(i).dose +" " + result.rows.
              item(i).time + " " + result.rows.item(i).reminder+" "+result.
              rows.item(i).active+" "+result.rows.item(i).pid+" "+result.
              rows.item(i).reminder_set+" ";
          j++;
```

```
        }
    }

/**
 * Save seizure entries into an array.
 */
    function showAllSeizures(tx, result) {
        for(var i=0; i<result.rows.length;i++) {
          data[j]=" tutq: "+result.rows.item(i).id + " " + result.rows.item(
              i).date + " " + result.rows.item(i).time + " " + result.rows.
              item(i).seizure+" :end ";
          j++;
        }
    }

/**
 * Save mood entries into an array.
 */
    function showAllMoods(tx, result) {
        for(var i=0; i<result.rows.length;i++) {
          data[j]=" keyp: "+ result.rows.item(i).id + " " + result.rows.item
              (i).date + " " + result.rows.item(i).time+ " " + result.rows.
              item(i).mood+" :end ";
          j++;
        }
    }

/**
 * Save medicine usage entries into an array.
 */
    function showAllMedicinesUsage(tx, result) {
        for(var i=0; i<result.rows.length;i++) {
          data[j]=" ishl: "+result.rows.item(i).id +" "+ result.rows.item(i)
              .med_id+" "+ result.rows.item(i).date +" " + result.rows.item(
              i).time + " ";
          j++;
        }
    }

/**
 * Save menstruation entries into an array.
 */
    function showAllMenstruations(tx, result) {
        for(var i=0; i<result.rows.length;i++) {
          data[j]=" aghr: "+ result.rows.item(i).id + " " + result.rows.item
              (i).date+" ";
          j++;
        }
    }

/**
 * Save comment entries into an array.
 */
    function showAllComments(tx, result) {
        for(var i=0; i<result.rows.length;i++) {
          data[j]=" hewr: "+result.rows.item(i).id +" "+ result.rows.item(i)
              .date +" " + result.rows.item(i).time + " "+ result.rows.item(
              i).comment+" :end ";
          j++;
        }
    copy(wr, data);
    }
```

restore.js

```
/**
 * REBACKUP.JS IMPLEMETS FUNCTIONALITY TO RESTORE ALL REGISTRATIONS TO
     DATABASE
```

```
 * FROM THE BACKUP FILE STORED ON DEVICES SDCARD.
 */
  var data = new Array();
  var dir = "EpilepsyApp/Backups/";
  var filename;
  var fs;
  var files;

/**
 * Function runs on the activity load and gets a root directory on a memory
     card.
 */
    function onDeviceReady() {
      addScroller();
      addNavButton(getString('btn-back'), 'back', 'settings.html');
      addNavButton(getString('btn-home'), 'home', 'index.html');
      addNavButton(getString('btn-delete'), 'recycle', "javascript:
          deleteBackUpsAll();");
        window.requestFileSystem(LocalFileSystem.PERSISTENT, 0,
            onFileSystemSuccess, fail);
    }

/**
 * Get the backup directory.
 */
    function onFileSystemSuccess(fileSystem) {
      fs = fileSystem;
        fileSystem.root.getDirectory("EpilepsyApp/Backups", {create: false,
            exclusive: false}, getDirSuccess, fail);
    }

/**
 * Read entries from the directory.
 */
    function getDirSuccess(dirEntry) {
        // Get a directory reader
        var directoryReader = dirEntry.createReader();
        // Get a list of all the entries in the directory
        directoryReader.readEntries(readerSuccess,fail);
    }

    function getThis(el){
      var eid = el.getAttribute("id", 1);
      filename=eid;
      startRetrieve();
    }

/**
 * Display direcotry content.
 */
    function readerSuccess(entries) {
      files = entries;
        var i;
        var ul = document.getElementById("backup");
        var li;
        var a;
        for (i=0; i<entries.length; i++) {
          li = document.createElement("LI");
          li.innerHTML = entries[i].name;
          b = document.createElement("button");
          b.setAttribute("onclick","doDeleteFile(this);");
          b.id = entries[i].name;
          b.style.float="right";
          b.style.width="10%";
          b.style.height="20%";
          b.style.backgroundImage="url('../images/icons/mdpi/recycle.png')";
          a = document.createElement("A");
```

```
            a.id = entries[i].name;
            a.setAttribute("class","link");
            a.style.width = "70%";
            a.setAttribute("onclick","getThis(this);");
            a.appendChild(li);
            ul.appendChild(a);
            ul.appendChild(b);
        }
    }

    function startRetrieve() {
        window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, gotFS, fail)
            ;
    }

    function gotFS(fileSystem) {
        fileSystem.root.getFile("EpilepsyApp/BackUps/"+filename, null,
            gotFileEntry, fail);
    }

    function gotFileEntry(fileEntry) {
        fileEntry.file(gotFile, fail);
    }

    function gotFile(file){
        readAsText(file);
    }

    function readDataUrl(file) {
        var reader = new FileReader();
        reader.onloadend = function(evt) {
        };
        reader.readAsDataURL(file);
    }

/**
* Read file and save entries into an array data[]
*/
    function readAsText(file) {
        var reader = new FileReader();
        reader.onloadend = function(evt) {
            var string=evt.target.result;
            data = string.split(" ");
            whatToDo();
        };
        reader.readAsText(file);
    }

    function whatToDo() {
      navigator.notification.confirm(
                'The entire DB will be replaced, do you want to retrieve
                    data from backup file?',
                 onConfirm,              // callback to invoke with index of
                      button pressed
                'Warning',               // title
                'Retrieve,Cancel'        // buttonLabels
        );
    }

    function onConfirm(button) {
      var db = openDb();
        if(button==1){
          db.transaction(deleteDatabase, errorDb);
          db.transaction(createTables, errorDb);
          db.transaction(retrieveDB, errorDb);
        }
        else
```

```
        return false;
  }

/**
* Write data back to database.
*/
  function retrieveDB(tx){
    for(var i=0; i<=data.length; i++){
      if(data[i]=="dora:"){
        var j=i+2;
        var pill=data[j++];
        while(data[j]!=":end"){
          pill=pill+" "+data[j++];
        }
        tx.executeSql('INSERT INTO medicines (name, dose, time, date,
            reminder, active, pid, reminder_set) VALUES(?,?,?,?,?,?,?,?)',
      [pill, data[j+1], data[j+2], new Date(), data[j+3], data[j+4], data[
          j+5], 0]);
      }
      else if(data[i]=="tutq:"){
        var j=i+4;
        var seizure=data[j++];
        while(data[j]!=":end"){
          seizure=seizure+" "+data[j++];
        }
        tx.executeSql('INSERT INTO seizures (date, time, seizure) VALUES
            (?,?,?)',
        [data[i+2], data[i+3], seizure]);
      }
      else if(data[i]=="keyp:"){
        var j=i+4;
        var mood=data[j++];
        while(data[j]!=":end"){
          mood=mood+" "+data[j++];
        }
        tx.executeSql('INSERT INTO moods (date, time, mood) VALUES(?,?,?)
            ',
        [data[i+2], data[i+3], mood]);
      }
      else if(data[i]=="ishl:"){
        tx.executeSql('INSERT INTO medicine_usage (med_id, date, time)
            VALUES(?,?,?)',
        [data[i+2], data[i+3], data[i+4]]);
      }
      else if(data[i]=="aghr:"){
        tx.executeSql('INSERT INTO menstruations (date) VALUES(?)',
        [data[i+2]]);
      }
      else if(data[i]=="hewr:"){
        var j=i+4;
        var comment=data[j++];
        while(data[j]!=":end"){
          comment=comment+" "+data[j++];
        }
        tx.executeSql('INSERT INTO comments (date, time, comment) VALUES
            (?,?,?)',
        [data[i+2], data[i+3], comment]);
      }
    }
  alert(getString("alrt-retrieved")+filename);
  }
```

D.6 Seizure monitor implementation

Monitor.java

```
package no.hig.stud.bachelor.epilepsyapp;
import android.app.Notification;
import android.app.PendingIntent;
import android.os.IBinder;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.hardware.SensorManager;
import android.location.LocationManager;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.widget.Toast;
import no.hig.stud.bachelor.epilepsyapp.R;
import no.hig.stud.bachelor.epilepsyapp.WarnContact;

/**
* Monitor class is used to start, stop and control the monitoring
* activity, either seizure monitoring or seizure auto-detection.
*/
public class Monitor extends Service {
    public static boolean RUNWHAT = false;
    public static boolean RUN_ALERT_TIMER = false;
  //if user press continue on reminder, the alerttimer should stop
    public static boolean CONTINUE_PRESSED = false;
    public static int MONITOR_INTERVAL;
    public static int ALERT_LIMIT = 10;
    public static boolean REMINDER_STARTED = false;
    public static int REMINDER_CONTEXT = 1;
    public static int ALERT_CONTEXT = 2;
    public static String CONTACT_PERSON = "";
    public static String CONTACT_NUMBER = "";
    public static int REMINDERINTERVAL_MINUTE = 0;
    public static int REMINDERINTERVAL_SECOND = 15;
    public static int RESPONDINTERVAL_MINUTE = 2;
    public static int RESPONDINTERVAL_SECOND = 10;
    public static boolean MONITOR_RUNNING;

  final static int myID = 1234;
    public static SeizureHandler seizureHandler;
    static MonitoredSeizure ms;
    static Monitor monitor;
    Handler messageHandler;
    Context ctx;
    public static Intent reminderActivity, warnContactActivity;

    @Override
    public void onCreate() {
        super.onCreate();
        // Sets for singleton
        monitor = this;
        MONITOR_RUNNING = true;
        RUN_ALERT_TIMER = true;
        warnContactActivity = new Intent(Monitor.this, WarnContact.class);
        warnContactActivity.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        //setting foreground notification
        Notification notification = new Notification(R.drawable.monitor,"
            Epilepsi",
        System.currentTimeMillis());
        Intent notificationIntent = new Intent(this, Monitor.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
            notificationIntent, 0);
        notification.setLatestEventInfo(this, "The Monitor",
```

```
        "Monitor is activated data collecting in porcess ...", pendingIntent
            );
        startForeground(myID, notification);
         // starts datarecording
        seizureHandler = new SeizureHandler((SensorManager)getSystemService(
            SENSOR_SERVICE),
                (LocationManager)getSystemService(Context.LOCATION_SERVICE),
                     getApplicationContext());
        seizureHandler.startMonitoring();
    / starts the reminder
        reminderTimer();
    }

    /*
     * singleton
     */
    public static Monitor getInstanceOf(){
       return monitor;
    }

    /*
     * should stop all recording
     */
    public static void stopMonitor(){
    MONITOR_RUNNING = false;
        RUN_ALERT_TIMER = false;
        seizureHandler.stopReading();
        getInstanceOf().stopSelf();
    }

    public static void reminderTimer(){
       Thread t = new Thread(new Runnable() {
             int i = 0;
             int interval = (Monitor.REMINDERINTERVAL_MINUTE*60)+Monitor.
                 REMINDERINTERVAL_SECOND;
                public void run() {
                    while(i<interval && MONITOR_RUNNING ){
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException ex) {}
                     i++;
                    }
                    reminderHandler.sendEmptyMessage(1);
                }
            });
            t.start();
    }


static Handler reminderHandler = new Handler(){
      @Override
      public void handleMessage(Message msg) {
          if(msg.what == 1 && MONITOR_RUNNING){
            Toast.makeText(monitor, "Reminder", Toast.LENGTH_LONG).show();
              CONTINUE_PRESSED=false;
              alertTimer();
          }
      }
    };

    /*
     * checks for user response
     */
    public static void alertTimer(){
         Thread t = new Thread(new Runnable() {
            int timeLimit = 0;
```

```
            int interval = Monitor.RESPONDINTERVAL_MINUTE*2 + Monitor.
                RESPONDINTERVAL_SECOND;
            public void run() {
                while(MONITOR_RUNNING == true && timeLimit< interval &&
                    CONTINUE_PRESSED == false){
         try {
                        Thread.sleep(1000);
                    } catch (InterruptedException ex) {}
                    timeLimit++;
                }

                if(CONTINUE_PRESSED==true && MONITOR_RUNNING == true){
                  reminderTimer();
                }
                else if(timeLimit == interval && CONTINUE_PRESSED == false){
                    alertHandler.sendEmptyMessage(1);
                }
            }
         });
         t.start();
    }

    /*
     * handles the alertTimer thread
     */
    static Handler alertHandler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
            /*
             * should store monitor as if wer done
             */

            if(MONITOR_RUNNING == true && CONTINUE_PRESSED == false){
                Monitor.getInstanceOf().startActivity(warnContactActivity);
              Toast.makeText(monitor, "Warning", Toast.LENGTH_LONG).show();
             CONTINUE_PRESSED = false;
            }
        }
    };

    @Override
    public void onDestroy() {
      stopMonitor();
      String test = seizureHandler.getLocation().toString();
      String time = seizureHandler.lastReading.getTimeStamp().toString();
    }
}
```

SeizureHandler.java

```
package no.hig.stud.bachelor.epilepsyapp;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.hardware.SensorManager;
import android.location.LocationManager;
import android.net.Uri;
import android.util.Log;
import java.util.logging.Level;
import java.util.logging.Logger;
import no.hig.stud.bachelor.epilepsyapp.AccelerationListener;
import no.hig.stud.bachelor.epilepsyapp.GpsListener;
import no.hig.stud.bachelor.epilepsyapp.MonitoredSeizure;
import no.hig.stud.bachelor.epilepsyapp.SensorReading;

/**
* SeizureHandler is a class, that is actually responsible actual monitoring
    process.
```

```
* It initiates sensor listeners and data structure for saving sensor data
    and handles
* data transfers.
*/
public class SeizureHandler {
    MonitoredSeizure seizure;
    AccelerationListener al;
    GpsListener gl;
    boolean runit = false;
    Thread t;
    private static int POLL_INTERVAL = 500;
    public static SensorReading lastReading; //beeing requested from
        WarnContact
    double b;
    Uri c;

    public SeizureHandler(SensorManager sm, LocationManager lm, Context ctx)
        {
        al = new AccelerationListener(sm);//AccelerationListener(sm);
        gl = new GpsListener(ctx, lm);
        seizure = new MonitoredSeizure();
        lastReading = new SensorReading();
    }

    /*
     * starts the monitoring on a seperate thread
     */

    public void startMonitoring(){
        al.startAccelerationListening();
        gl.startGPSlistener();

    /*
         * run aslong as user dont abort
         *
         */
        runit=true;
            t = new Thread(new Runnable() {
                public void run() {
                    while(runit){
                        /*
                         * sleeps to set pollrate
                         */
                        try {
                            Thread.sleep(POLL_INTERVAL);
                        } catch (InterruptedException ex) {
                            Logger.getLogger(SeizureHandler.class.getName())
                                .log(Level.SEVERE, null, ex);
                        }
                        setReading();
                    }
                }
            });
            t.start();
    }

    /*
     * Collects a single reading and stores it in the total seizure
     */
    public void setReading(){
      lastReading = new SensorReading(gl.getLocation());

        // The Warning.
        // The sms notification with GPS location of a user not responding
            wil be sent to a contact person
        SQLiteDatabase myDataBase;
        String time = lastReading.getTimeStamp().toString();
```

```
            String gpslocation = getLocation().toString();
            seizure.putReading(lastReading);
        }

        private boolean checkDataBase() {
         SQLiteDatabase checkDB = null;
         try {
              checkDB = SQLiteDatabase.openDatabase("/data/data/no.hig.stud.
                  bachelor.epilepsyapp/app_database/file__0/0000000000000001.db
                  ",
                                  null, Context.MODE_PRIVATE);
              checkDB.close();
         } catch (SQLiteException e) {}
         return checkDB != null ? true : false;
        }

        /*
         * call to stop
         */
        public void stopReading(){
            al.stopListening();
            gl.stop();
            SQLiteDatabase.releaseMemory();
            runit = false;
        }

        /*
         * should sumarize after completing recording
         */
        public MonitoredSeizure moveToDB(){
         return seizure;
        }

        public Uri getLocation(){
            return lastReading.getLocation();
        }
    }
```

MonitoredSeizure.java

```
    /*
     * one instance contains one monitored session
     */
    package no.hig.stud.bachelor.epilepsyapp;
    import java.util.Vector;
    import android.util.Log;

    /**
    * MonitoredSeizure class is responsible for storing all the sensor readings
    * received from SensorReading into one data structure.
    */
    public class MonitoredSeizure {
        Vector<SensorReading> timeLine;
        public MonitoredSeizure(){
        timeLine = new Vector<SensorReading>();
        }

        public void putReading(SensorReading reading){
            timeLine.add(reading);
        }

        public Vector<SensorReading> getTimeLine() {
            return timeLine;
        }

        public SensorReading getLastReading(){
            return timeLine.lastElement();
        }
```

```
}
```

AccelerationListener.java

```
package no.hig.stud.bachelor.epilepsyapp;
import android.app.Service;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.IBinder;

/**
* AccelerationListener class is responsible for reading accelerometer data
*/
public class AccelerationListener extends Service implements
    SensorEventListener {
  private SensorManager sensorManager;
  private final float[] currAccel;
  boolean running = true;
  boolean accelLock = true;
  private static final String TAG = "AccelerationListenerService";

    /*
     * Constructor. Sets SensorManager.
     */
    public AccelerationListener(SensorManager sm){
      currAccel = new float[3];
        sensorManager = sm;
    }

    /*
     * Starts accelerometer listening.
     */
    public void startAccelerationListening() {
      sensorManager.registerListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                SensorManager.SENSOR_DELAY_FASTEST);//
    }
    /*
     * Unregisters the listener
     */
    public final void stopListening() {
         sensorManager.unregisterListener(this);
     }

    /*
     * Sets current accelerometer-data
     */
    public synchronized void onSensorChanged(final SensorEvent event) {
      final float[] val = new float[]{event.values[0], event.values[1],
          event.values[2]};
      if (event.sensor.getType() != Sensor.TYPE_ACCELEROMETER) {
          return;
        }
      updateAccel(val[0], val[1], val[2]);
    }

    /*
     * Never in use but has to be implemented.
     */
    public void onAccuracyChanged(Sensor arg0, int arg1) {
    }

    /*
     * Returns accelerometer-data when polled
     */
```

```
    public final float[] returnAcceleration(){
      return currAccel;
    }
    /*
     * Updates values in the data structure.
     */
    public void updateAccel(float x,float y,float z){
      currAccel[0] = x;
      currAccel[1] = y;
      currAccel[2] = z;
    }
}
```

GPSListener.java

```
package no.hig.stud.bachelor.epilepsyapp;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;
import java.util.List;

/**
* GpsListener class is responsible for retrieving GPS location
*/
public class GpsListener implements LocationListener{
    private Context ctx;
    private Uri curLocation;
    LocationManager locationManager;

  /*
  * Constructor.
  */
    public GpsListener(Context c,LocationManager lm){
        ctx = c;
        locationManager = lm;
        setLocation(null);


    }

  /*
  * Updates each time the location has changed
  */
    public void onLocationChanged(Location loc){
       setLocation(loc);
    }

    public void onProviderDisabled(String provider){
        Toast.makeText(ctx,"Gps Disabled",Toast.LENGTH_SHORT ).show();
    }

    public void onProviderEnabled(String provider){
      Toast.makeText(ctx,"Gps Enabled",Toast.LENGTH_SHORT).show();
    }

  /*
  * Sets the location
  */
    public void setLocation(Location loc) {
        List<String> providers = locationManager.getProviders(true);
        Location l = loc;
        /* Loop over the array backwards, and if you get an accurate
            location, then break out the loop*/
```

```
            if (l == null){
                for (int i=providers.size()-1; i>=0; i--) {
                        l = locationManager.getLastKnownLocation(providers.get(i
                            ));
                        if (l != null) break;
                }
            }
            curLocation = Uri.parse("http://maps.google.com.my/maps?f=d&source=
                s_d&saddr="+l.getLatitude()+","+l.getLongitude());
        }

      /*
      * Return current location
      */
        public Uri getLocation(){
          if (curLocation==null){
            curLocation=Uri.parse("the location is undefined, please try to
                contact sender");
            return curLocation;
          }
            return curLocation;
        }

      /*
      * Stop GPS listener.
      */
        public void stop(){
            locationManager.removeUpdates(this);
        }

      /*
      * Start GPS listener.
      */
        public void startGPSlistener(){
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
                 0, 0,this);
        }
    }
```

SensorReading.java

```
    package no.hig.stud.bachelor.epilepsyapp;
    import android.hardware.SensorEvent;
    import android.net.Uri;
    import android.os.Bundle;
    import android.util.Log;
    import java.security.Timestamp;
    import java.util.Calendar;

    /*
    * SensorReading class is responsible for ''packing'' sensors data.
    */
    public class SensorReading {
        java.sql.Timestamp timestamp;
        Bundle acceleration;
        Uri location;

        public SensorReading(){
            acceleration = null;
            location = null;
            timestamp = null;
        }

        public SensorReading(Bundle accel){
            acceleration = accel;
            location = null;
            timestamp = new java.sql.Timestamp(Calendar.getInstance().getTime().
                getTime());
```

```
    }

    public SensorReading(Bundle accel, Uri loca){
        acceleration = accel;
        location = loca;
        timestamp = new java.sql.Timestamp(Calendar.getInstance().getTime().
            getTime());
    }

    public SensorReading(Uri loca){
        location = loca;
        timestamp = new java.sql.Timestamp(Calendar.getInstance().getTime().
            getTime());
    }

    public Bundle getAcceleration() {
        return acceleration;
    }

    public Uri getLocation() {
        return location;
    }

    public java.sql.Timestamp getTimeStamp() {
        return timestamp;
    }
}
```

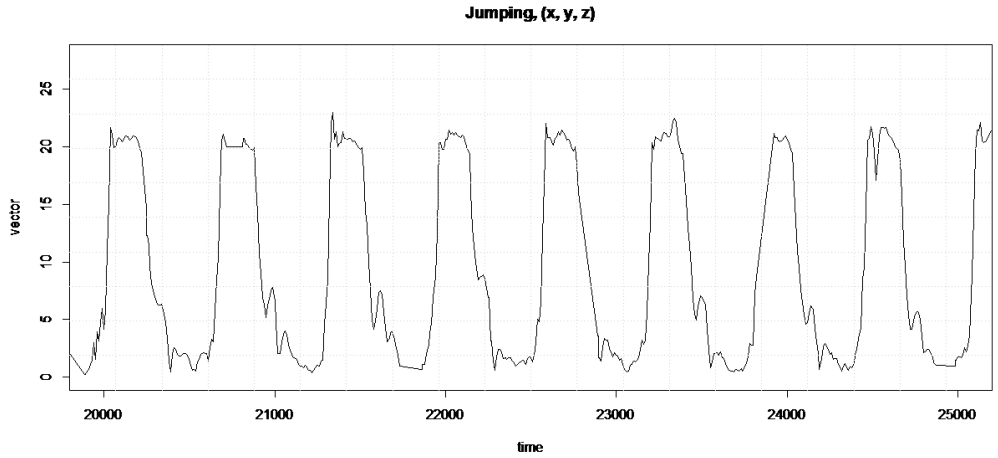
WarnContact.java

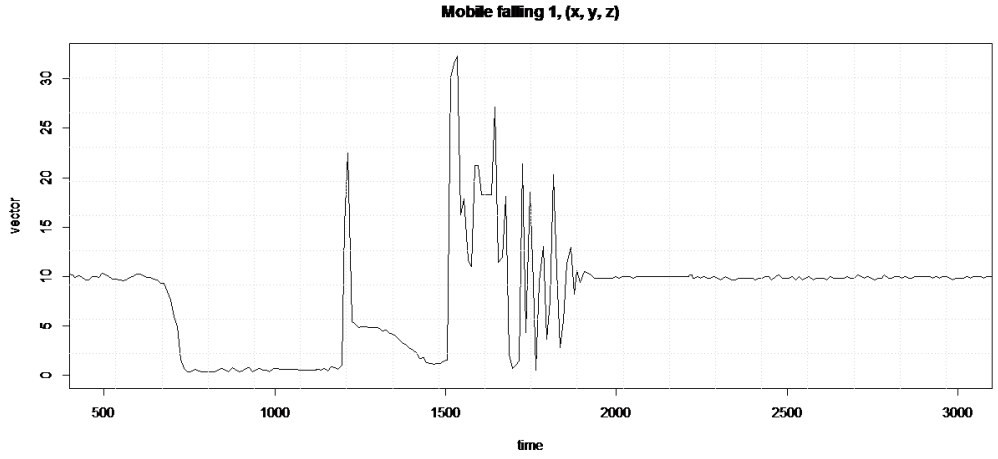
```
package no.hig.stud.bachelor.epilepsyapp;
import java.util.Calendar;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
import no.hig.stud.bachelor.epilepsyapp.R;
import no.hig.stud.bachelor.epilepsyapp.Monitor;

/*
* WarnContact class is responsible for sending SMS to contacts.
*/
public class WarnContact extends Activity {
    public static int CONTACT_ALERT_LIMIT = 30;
    static PendingIntent pi;
    static WarnContact warnContact;
    private static Cursor curSor;
    public static Intent warnContactActivity;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        warnContact = this;
        alert();
```

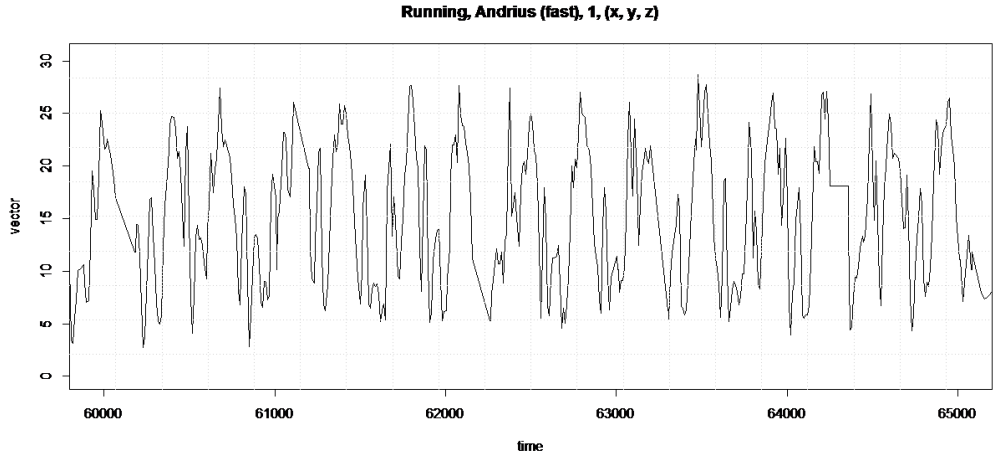
```
      Intent newIntent = new Intent(WarnContact.this, EpilepsyAppActivity.
          class);
    newIntent.putExtra("sender", "alarm");
    WarnContact.getInstanceOf().startActivity(newIntent);
      pi = PendingIntent.getActivity(this, 0, new Intent(this, WarnContact
          .class), 0);
      alertContactTimer();
  }

  protected static WarnContact getInstanceOf() {
  return warnContact;
}

private static class OKListener implements OnClickListener {
      public void onClick(View arg0) {
          Monitor.stopMonitor();
          warnContact.finish();
      }
  }

  public static void alertContactTimer(){
       Thread t = new Thread(new Runnable() {
          int timeLimit =0;
          public void run() {
      while(Monitor.MONITOR_RUNNING == true && timeLimit<
          CONTACT_ALERT_LIMIT){
        try {
                      Thread.sleep(1000);
                  } catch (InterruptedException ex) {}
                      timeLimit++;
              }
      if(timeLimit == CONTACT_ALERT_LIMIT && Monitor.MONITOR_RUNNING ==
          true){
                  alertContactHandler.sendEmptyMessage(1);
              }
          }
       });
       t.start();
  }

  /*
   * handles the alertTimer thread
   */
  static Handler alertContactHandler = new Handler(){
      @Override
      public void handleMessage(Message msg) {
          // should store monitor as if wer done
          alert();
          sendSMS();
      }
  };

  public static void alert(){
      Log.d("Monitor","ALERT HAS BEEN INITIATED");
  }

  public static void sendSMS(){
     SQLiteDatabase myDataBase;
     myDataBase =  SQLiteDatabase.openDatabase("/data/data/no.hig.stud.
         bachelor.epilepsyapp/app_database/file__0/0000000000000001.db",
         null, Context.MODE_PRIVATE);
     String select = "Select * from numbers";
   try {
     curSor = myDataBase.rawQuery(select, null);
     curSor.moveToFirst();
     if(curSor.getCount()<=0){
```

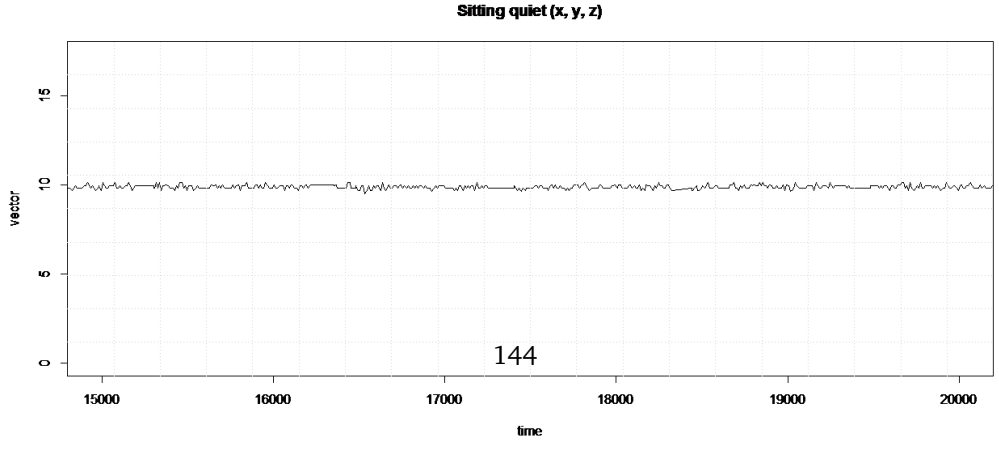
```
          Toast.makeText(warnContact, "None number is selected for sending
              warning sms", Toast.LENGTH_LONG).show();
         }
         for (int i = 0; i < curSor.getCount(); i++) {
          Monitor.CONTACT_NUMBER = curSor.getString(1);
          SmsManager sms = SmsManager.getDefault();
            StringBuilder sb = new StringBuilder();
            sb.append("USER HAD A SEIZURE | LOCATION: ");
            sb.append(Monitor.getInstanceOf().seizureHandler.lastReading.
                getLocation().toString());
            String s = sb.toString();
          if(Monitor.CONTACT_NUMBER != ""){
                sms.sendTextMessage(Monitor.CONTACT_NUMBER, null,s, null, null
                    );
               }
               else {
                   Toast.makeText(warnContact, "Warning has been not sent via
                       sms, please choose contacts in settings ", Toast.
                       LENGTH_LONG).show();
                }
           curSor.moveToNext();
         }
          curSor.close();
       }
        catch (Exception e) {
               e.printStackTrace();
       }
        finally{
               myDataBase.close();
               SQLiteDatabase.releaseMemory();
       }
      }
}
```

E Accelerometer readings graphs for various activities

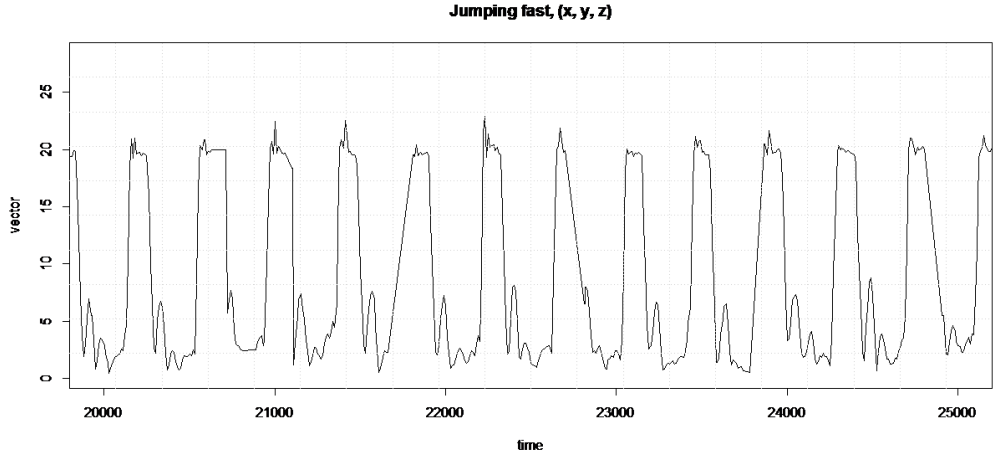

Jumping, (x, y, z)
vector
time
0
5
10
15
20
25
20000
21000
22000
23000
24000
25000

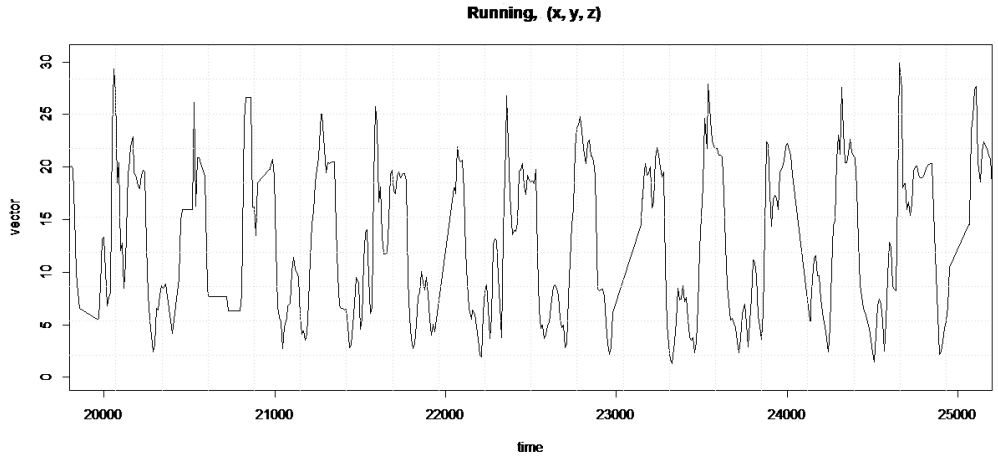

Mobile falling 1, (x, y, z)
vector
time
0
5
10
15
20
25
30
500
1000
1500
2000
2500
3000

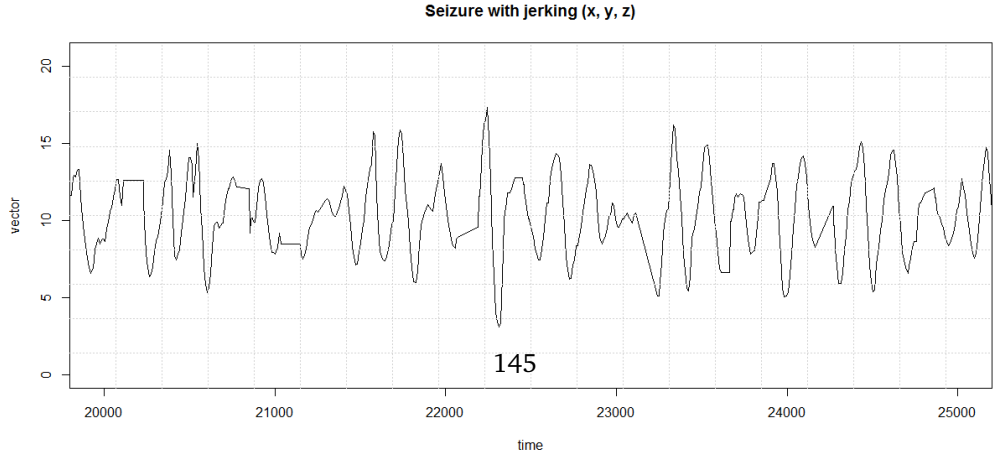

Running, Andrius (fast), 1, (x, y, z)
vector
time
0
5
10
15
20
25
30
60000
61000
62000
63000
64000
65000

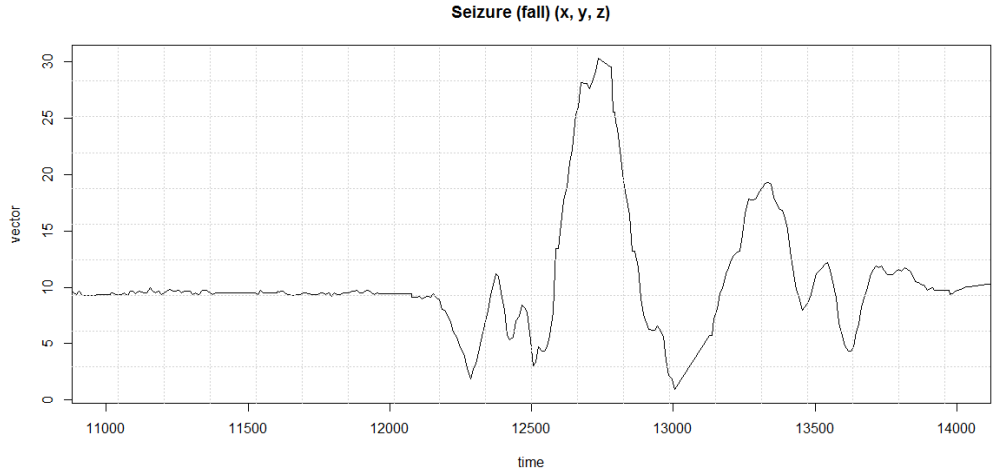

Sitting quiet (x, y, z)
vector
time
0
5
10
15
15000
16000
17000
18000
19000
20000

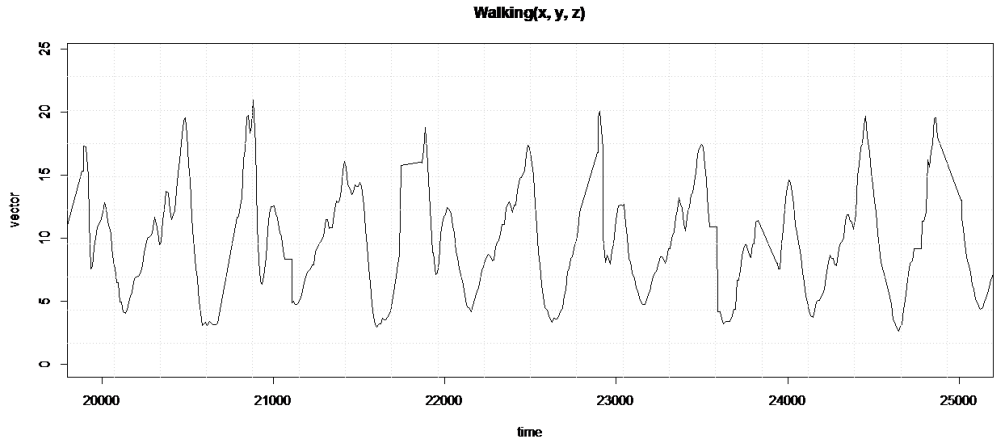

Driving on a good road, (x, y, z)
vector
0
5
10
15
20
25
50000
52000
54000
56000
58000
60000
time

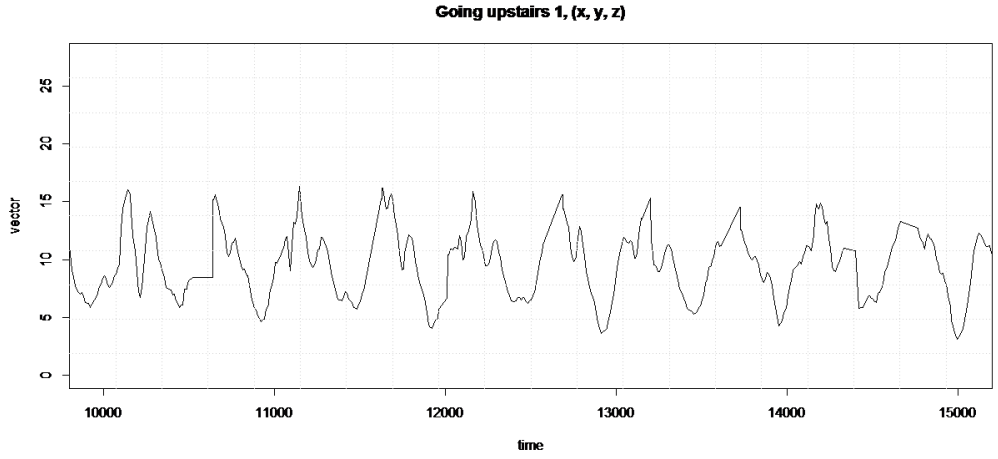

Jumping fast, (x, y, z)
vector
0
5
10
15
20
25
20000
21000
22000
23000
24000
25000
time

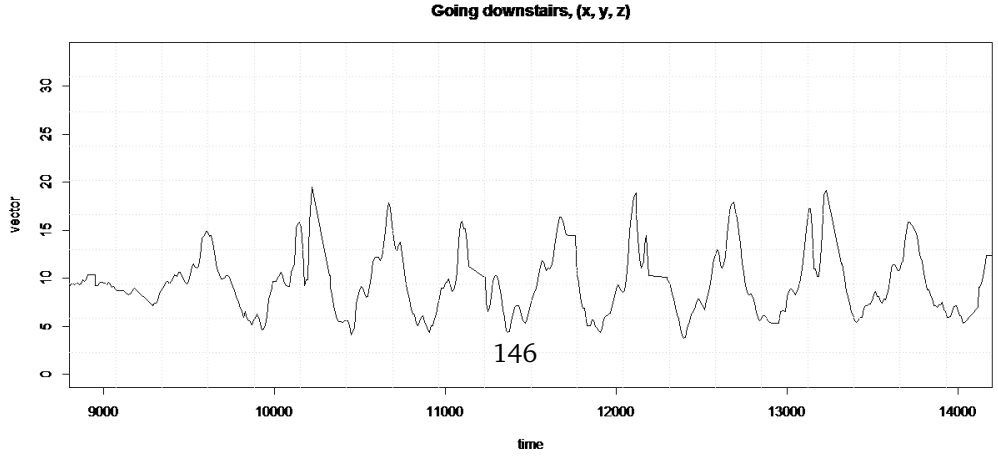

Running, (x, y, z)
vector
0
5
10
15
20
25
30
20000
21000
22000
23000
24000
25000
time

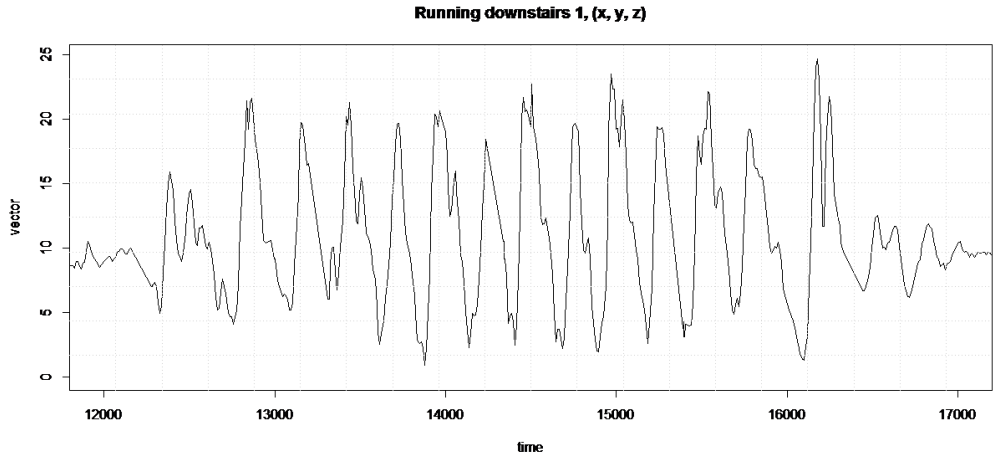

Seizure with jerking (x, y, z)
vector
0
5
10
15
20
20000
21000
22000
23000
24000
25000
time

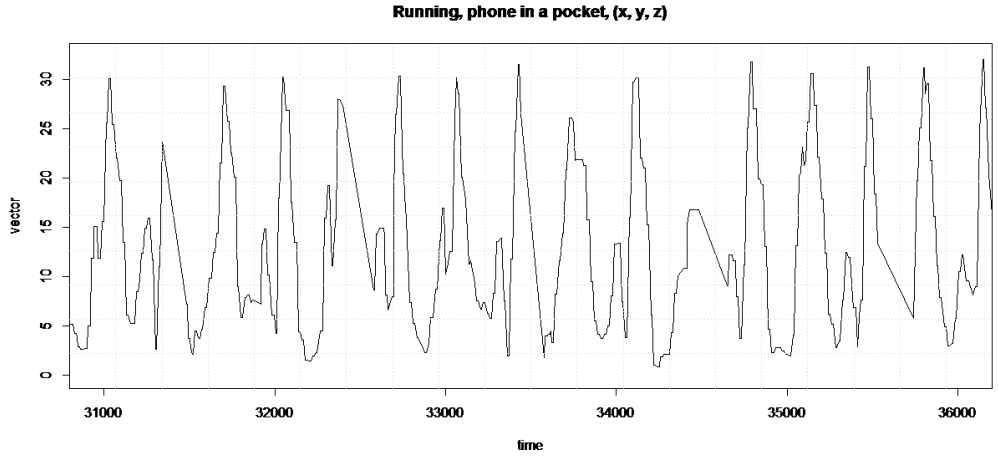

Seizure (fall) (x, y, z)
vector
0
5
10
15
20
25
30
11000
11500
12000
12500
13000
13500
14000
time

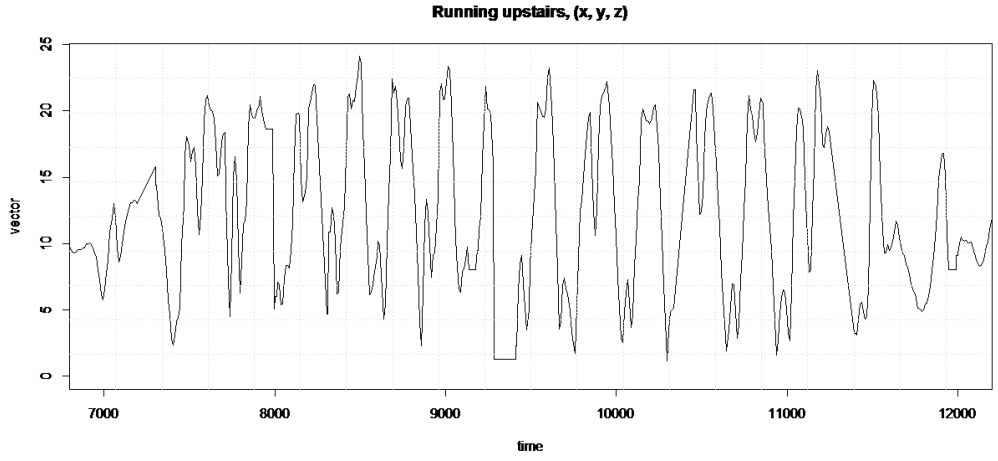

Walking(x, y, z)
vector
0
5
10
15
20
25
20000
21000
22000
23000
24000
25000
time

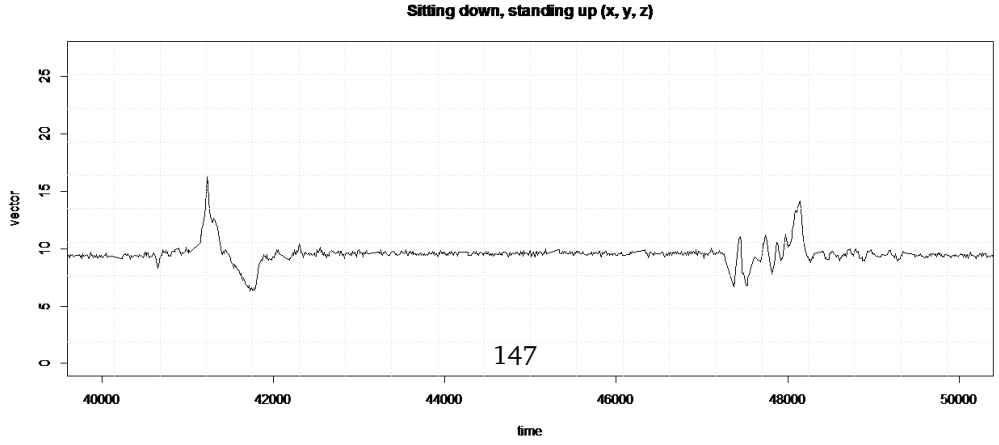

Going upstairs 1, (x, y, z)
vector
0
5
10
15
20
25
10000
11000
12000
13000
14000
15000
time

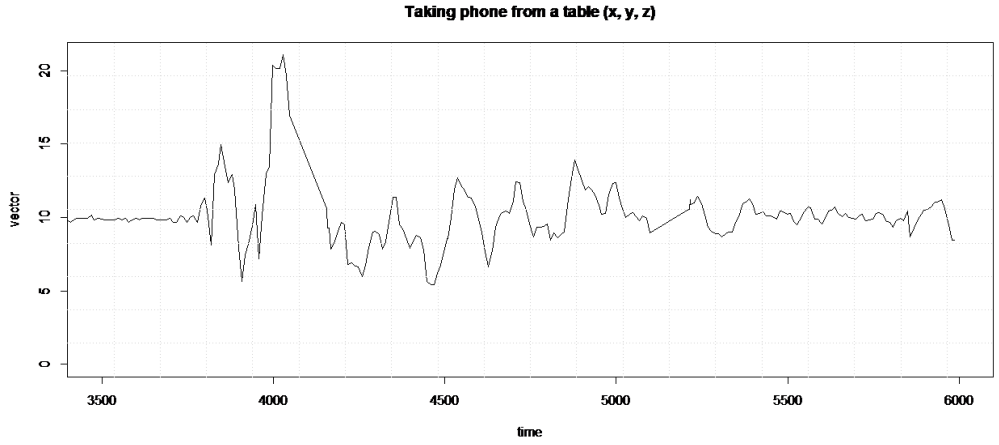

Going downstairs, (x, y, z)
vector
0
5
10
15
20
25
30
9000
10000
11000
12000
13000
14000
time

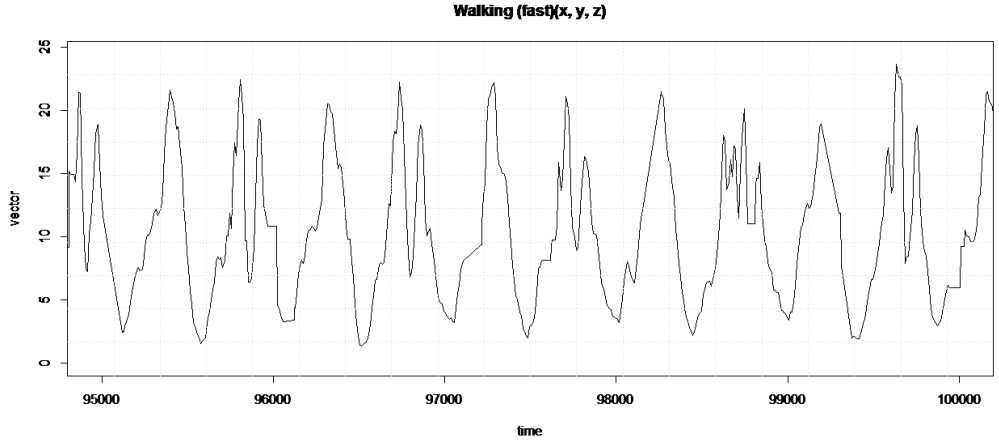

Running downstairs 1, (x, y, z)
vector
time
0
5
10
15
20
25
12000
13000
14000
15000
16000
17000

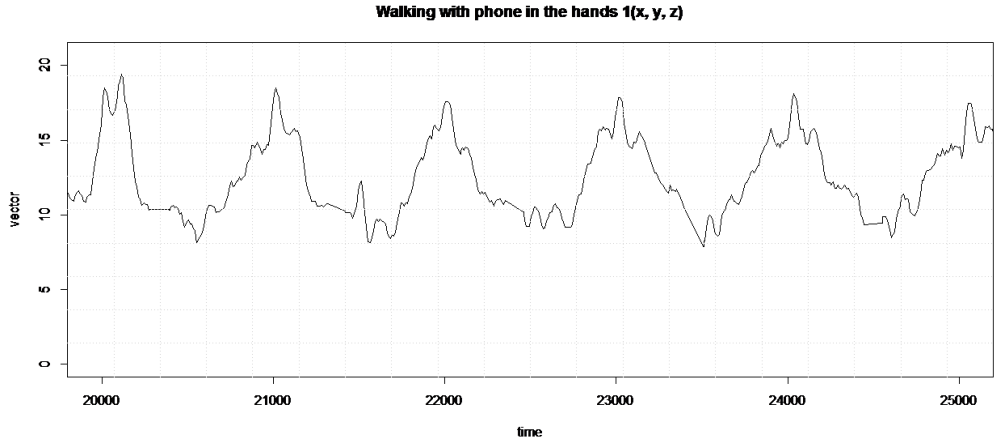

Running, phone in a pocket, (x, y, z)
vector
time
0
5
10
15
20
25
30
31000
32000
33000
34000
35000
36000

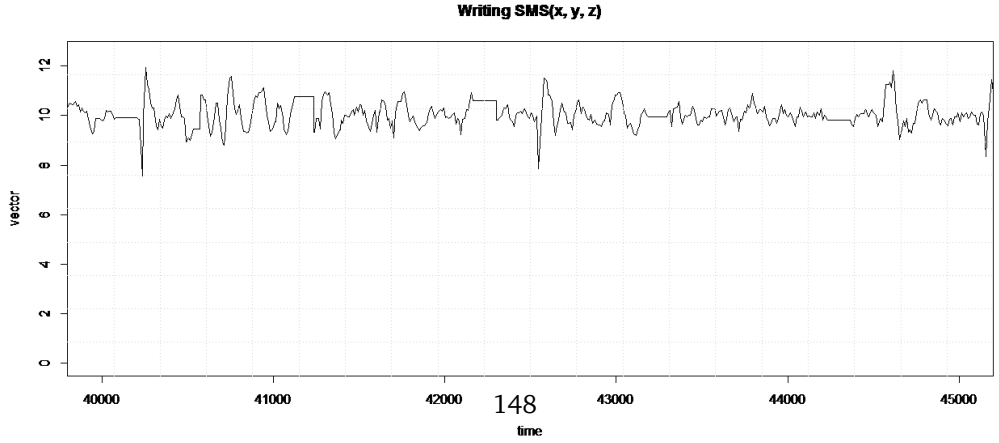

Running upstairs, (x, y, z)
vector
time
0
5
10
15
20
25
7000
8000
9000
10000
11000
12000


Sitting down, standing up (x, y, z)
vector
time
0
5
10
15
20
25
40000
42000
44000
46000
48000
50000


Taking phone from a table (x, y, z)
vector
0
5
10
15
20
3500
4000
4500
5000
5500
6000
time


Walking (fast)(x, y, z)
vector
0
5
10
15
20
25
95000
96000
97000
98000
99000
100000
time


Walking with phone in the hands 1(x, y, z)
vector
0
5
10
15
20
20000
21000
22000
23000
24000
25000
time


Writing SMS(x, y, z)
vector
0
2
4
6
8
10
12
40000
41000
42000
43000
44000
45000
time

F Bug Report for Telenor Objects

BUG REPORT
18.04.2012 12:56

"GET" XMLHttpRequest

Brief:
The "EpilepsyApp" team has been working to integrate the ability of m2m connection provided by TelenorObjects in the period of about four days. Unfortunately all the tries to GET data from Shepherd failed. According to the several tests we have discovered some issues that can be a potential problem in getting access of data using mobile devices.

Disadvantages:
The failure in operation of shepherd for mobile devices.
That can be a problem to integration of different machines/devices involving mobile phones

We operate on:
PhoneGap - Framework for cross-platforms version 1.5
Coding: HTML5 and JavaScript, some native part is coded by Java (Android)

Username for shepherd access: higadiabd

Mobile devices on test:
Samsung Galaxy S2, Samsung Galaxy Mini, HTC Wildfire A3333

Issue/Bug:
The use of post in order to post values for temperature and GPS location succeeded. We haven't encountered any difficulties.

When we tried to get posted data from shepherd using api:
https://api.m2m.to/v2/observation/dev:12E0BFCA-80B6-74EA-7EB2-EF94157595CF/

using all possible ways listed below:
1.
https://username:password@api.m2m.to/v2/observation/dev:12E0BFCA-80B6-74EA-7EB2-EF94157595CF/

Testing direct access providing username and password on browser we get following results:

Here we get an extra window for verification of username and password in Mozilla Firefox 11.0
Same in IE version 9.0
Nothing happens using Opera/9.80 (Windows NT 6.1; U; en) Presto/2.10.229 Version/11.62
and only Google Chrome 18.0.1025.162 goes directly to the link without any extra verification.

We should note that canceling authentication window in Mozilla Firefox alerts an additional window for authentication whereas in IE just once.

2. Using javascript in PhoneGap like below

```
ajax=new XMLHttpRequest();
…. setting headers..... /user-agent/
ajax.open("GET", url, false, username, password ); | ajax.open("GET", url,
```

```
false, [username, password] );
…..
```

we get nothing neither to display or receive data.
The “readyState” is 4 then “status” is “0”.

P.S. The jquery method has been used too.

At the period of all the tests we have been using api devices, objects following all changes both in dashboard and through https connection in browser.

Unfortunately all the links from TelenorObjects team, whom we sincerely thank for all help and assistance, didn’t help.

Assumptions:
In a conclusion having all the results listed above we came to that mobile application can’t access the data due to the extra authentication window that doesn’t let the access to the data. Therefore we do not receive either 404 or 200 for the status.

The resolvement of such bug, if it is the reason we do not get access to the shepherd, would be a perfect possibility to test our “Epilepsy App” application integrated with the Telenor Objects providing users the simplicity of getting necessary data directly to the Mobile device.

Sincerely, Adiljan Abdurihim and Andrius Januska

G Meeting protocols

Subject: Scoping the project

Date: 07 December, 2011
Place: HIG, A032
Duration: 120 minutes

Participants: Adiljan Abdurihim, Andrius Januska, Simon McCallum

Meeting #0 - Wednesday

After announcement of bachelor project groups and their supervisor, it was made first contact with supervisor by e-mail. From very first email conversation group had been provided by some important information. Email is dated from 01.12.2011.

> **"** 1, the report will be in Latex. I prefer students to learn Latex and use it for a large document. As programmers it should also appeal to your sense accuracy
>
> 2, I will be meeting with you remotely from Jan 22 to Apr 3. I expect weekly meetings by Skype or hangout. with a progress report of at least half a page 12 hours before the meeting
>
> 3, I require the use of version control. Be it GIT or SVN or whatever. **"**

The very first meeting with Simon McCallum to discuss about work flow and realistic framework for the bachelor project next semester.

Questions:

1. What kind of equipment we need for our project ?
2. Why to get a "group room"?
3. What is the realistic scope for the project ?
4. When you say submission of report 12 hours before, do you mean GMT +1 ?
5. What are challenges with using PhoneGap?
6. SCRUM?

1. iPhone, Android phone, LG Phone 7, Symbian.
2. HOS-HIG are stakeholders, equipments are expensive so it is better to store them in school.
3. Scope will be defined by the end of the January. It was adviced that scope should be defined in order to learn new things, use experience gathered through the courses and of cos be interested in it, but balance is to be kept. Important is to define the scope that will help project members to see the achievements by prioritizing most essential and leaving rest as an extra features.
4. Skype or hangout, G+, meetings will be held with Simon at 9 a.m. Norwegian time (Monday, Wednesday or Thursday) because of New Zealand time that differs +12 hours.
5. Using PhoneGap that built on use of javascript, HTML5, CSS3, multi touch, native will be limited. It will be more web browse app. Layouts and multi-touch will be difficult to perform. Members have to make more survey.
6. Trello will be used for the scrum work flow (this we have to be agreed on later).

Extra
Some extra issues have been discussed during the meeting and advices have been given.

7. How to get data from phone and how to store? - Legal issues have to be considered
8. If a person agrees than we can collect data, otherwise not allow to install or use offline version. Such kind of resolution can be kept in mind for further discussion.
9. Choose either Schools SVN or Googlecode f.ex.
10. Tools selection: PhoneGap, Trello?
11. User manual for the app.
12. Write one or two communications public blog. Promote the project. What are you doing? Show the development progress.

Tools for the use during the bachelor:
PhoneGap: cross platform that will allow us to develop application for the different platforms where Android and iOS are prioritized.
Trello: can be used as scrum work flow post it cards for remote use.
Gnuplot: For the graphical representation.
LaTeX**:** is a document preparation system for high-quality typesetting.
Doxygen: it can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in LaTeX) from a set of documented source files.
http://www.gtl.hig.no/index.php/Main_Page : some useful information.

As a good practice decisions have to be explained according to the architecture and technology. Feedback from stakeholder, Randi Stokke, is important. Be sure inform users of all kind personal data that will be stored.

At the time, Simon McCallum is away Jayson Mackie can be available for extra supervision.

Next meeting: 4th, January 2012, 9:00. 11 and 18th, of January 2012 meetings will be held in HiG.
24 of January (21:00, Norwegian time Tuesday, thee meeting will be held remotely).

Referent Adiljan Abdurihim

Meeting Protocol #1

Subject: Scoping the project

Date: 04 January, 2012
Place: HIG, E127
Duration: 9:00 - 11:00 (2 hours)

Participants: Adiljan Abdurihim, Andrius Januska, Randi Stokke, Simon McCallum

In the first official bachelor meeting with employer Randi Stokke and supervisor Simon McCallum we basically discussed the scope of the project, the features that should and could be included in the application.

We represented the list of features that we would prioritize on this project:

1. Diary (with moods, seizures, medicine use, menstruation cycles registration)
2. New medicines registration (download from the Internet, register self)
3. Alarming for medicament use
4. Reaching contacts under critical situations, providing GPS location.
5. Recording, storing and transferring seizure data (to local machine)
6. Presenting data in graphs.
7. Integrate app with Telenor Objects
8. Communicating to doctors (medical systems)
9. Medicines recognition.

The following has been discussed in deep:

1. Universal design - make it possible to control the app with voice.
2. Use the international Epilepsy icon, candle with blue circle around (for better recognition)
3. The application could be able show on the screen of the mobile information about what to do when one gets a seizure. Procedures will be described in steps by using images. The phone then should beep to attract someones attention around.
4. It should be possible to update information on the application, for ex. to update medicine list, tips of the day from external database.
5. The application could recognize when one has a seizure (Fallofon function).
6. The app should have possibility to start seizure alarm from the lock-screen (this should be optional).
7. The application could be integrated with Telenor Objects.
8. Tips of the day - show it automatically for a user when app starts, does not have to be overcrowded. Use external database to get tips from.
9. When selecting seizure types it should also display the explaining text.
10. The app should save the history of medicine usage.

The supervisor emphasized that it is very important to make the app maintainable, also when the project will be finished. It is important that somebody, not having programming skills could be able to update information for the app, to keep the app alive,

Further we should find what information we can use from the Epilepsiforbundet magazine, find out more about permissions to save medical data locally and transferring data, what protocol is used for transferring medical data in Norway. We should also contact Muhammed Derawi to talk about accelerometer usage in identifying seizure falls (he is working on Phd and has done relevant researches).

The supervisor adviced EditPadPro tool for searching and replacing regular expressions. Python is also a tool that we can get use for in this project.

We also asked the employer to have a look at the prototype app that was developed in Mobile Programming course, to evaluate the functionality, user-friendliness and to give us some feedback on this.

Next meeting was scheduled at 9:00 on 11th January, 2012 in E127, HiG.

Meeting Protocol #2

Subject: More about scoping, feedback on test app

Date: 11 January, 2012
Place: HIG, E127
Duration: 9:00 - 11:00 (2 hours)

Participants: Adiljan Abdurihim, Andrius Januska, Randi Stokke, Simon McCallum

Two main things we have discussed in this meeting was scope of the project and feedback from Randi about the test app.

Considering the scope and deciding what features we have to include in the app we have decided that we need to clearly separate the features that must be done to develop a fully functioning application and what can be done, to extend the functionality of the application. The features set of fully functioning application would than be the main target, that would scope the project. Additional features would be developed in case we meet the time limits. The most important in the development process is to work systematically, document the working process and always justify decisions we make.

Datatilsynet should be contacted to find out about regulations for saving medicines data locally on mobile devices, sending e-mails to yourself with medical information. This information is needed to go further with medicines part of the application. Two things to be take in consideration while writing e-mail:

1. Be clear and divide each functionality as one question.
2. Ask for permission to publish email conversation.

Further, some feedback on the test app was provided by Randi. Some of the things she commented:

- Medicines are usually taken every day, strictly at the same time of the day, one or two times a day. There is no need to specify the days of the week, when medicines should be taken. Medicines use should be set to everyday by default, and it should be possible to set several times on the same day for the same medicine.
- The list of medicines is about 20-30, ten of them are most usual.
- Make it possible to write extra notes when registering seizure. Text field have to be added.
- Icons should be more understandable, use text in addition to icons.
- Give some period before starting alarm and sending SMS.
- Reminder to take medicines should be well noticeable and do not stop if not stopped manually.

Finally, we have agreed with Randi and Simon on working and meeting plan. We have decided to have weekly Skype meetings with Simon on Mondays 9 a.m. when he is in New Zealand. Status reports will be provided after each Sprint, on Thursday. We have decided to have Scrum planning meetings and Scrum demo meetings with Randi every second Thursday. In meetings with Randi we will demonstrate what has been done during the last Sprint and decide what will be done on the next Sprint. Simon can take a part in a meeting remotely if it is needed.

Next meeting with Simon is scheduled 18.01.2012, 9:00 Next meeting with Randi - 26.01.2012, 14:00.

Referent Andrius Januska

Meeting Protocol #3

Subject: Biometrics, process of data analysis.

Date: 11 January, 2012
Place: HIG, A
Duration: 14:30 - 15:00 (30 minutes)

Participants: Adiljan Abdurihim, Andrius Januska, Patrick Bours, Muhammed Derawi.

The main purpose of the meeting is to understand the process of biometric analysis. We had an introduction to biometrics where we get an information about how the process of analyzing data collected from different sensors occur. It was pointed out that in order to analyze the difference of different types of changes it should be provided data collection with different state that will help indicate the accurate way of physical change detection. Additional thing in analysis is understanding the physical change, ie changes of body in x, y ,z-axis.

Some sample codes, instruction and supervision in analysis of biometric data will be provided by Muhammed Derawi and Patrick Bours.

If the wished future will actual for the development it will be very actual for the cooperation. Both Patrick Bours and Muhammed Derawi are interested in this process of analysis and collecting data from different sensors.

It was given a clear signal from group members that the meeting is been held in order to understand the world of biometrics and its process, but this feature will be an optional due to time limits.

Referent: Adiljan Abdurihim

Meeting Protocol #4

Subject: Project planning

Date: 18 January, 2012
Place: HIG, A018B
Duration: 10:00 - 10:30 (30 minutes)

Participants: Adiljan Abdurihim, Andrius Januska, Simon McCallum

This time we had a short meeting with project supervisor Simon. The things we discussed during the meeting were mainly related to the the project planning:

- Configuration management. We should describe what tools for version control we will use, and what kind of documents will be versioned.
- Tools for HTML code documentation. We need some tool to create documentation from HTML5 comments. Some kind of website mapping tools may be used to show the relationship between different pages. We may also need some script that could read HTML code and extract comments.
- Project licensing. We have found out that the product that we will create ownership is kind of divided between the employer and our team. We should agree with the employer what type of license we will implement.
- Blogging. We need to write some blogs. One of them should be more scientific, while the one we have is good for introducing our project to potential users and showing what we are doing.
- Test users. It is important to find some accidental users who could test the application we develop.

From next week (week 4) we will start to have weekly Skype meetings on Mondays 10.00 with Simon.

Referent Andrius Januska

Meeting Protocol #5

Subject: More about saving and transferring sensitive data

Date: 23 January, 2012
Place: Remote Skype conversation
Duration: 10:00 - 10:15 (15 minutes)

Participants: Adiljan Abdurihim, Andrius Januska, Simon McCallum

This time we had a short remote Skype meeting with project supervisor Simon. The main thing that we discussed in this meeting was about saving and transferirng sensitive health related data on mobile devices. We have specified what different types of sensitive information we need to save. According to this we have tree levels of application:

1) Diary without medicines
2) Diary including medicines registration
3) Diary with possibility to send registered data to self via e-mail.

According to these different levels, we will contact the Datatilsynnet and ask for what kind of information we need to get permissions to save data locally, if any.

Referent Andrius Januska

Meeting Protocol #6

Subject: Sprint Demo meeting 1

Date: 26 January, 2012
Place: HIG, E216
Duration: 14:00 - 14:30 (30 minutes)

Participants: Adiljan Abdurihim, Andrius Januska, Randi Stokke

It was a first sprint demo meeting of the project. We demonstrated to the employer the progress of the development process. Since we had not done so much yet, it was mainly demonstrating of new ideas for the app's GUI. Randi gave us some feedback about it, so we can stay on a right way.

We have also discussed some logic for medicine reminder and registration. Randi gave us some real life examples and we decided that we do not have to apply any restrictions on when a user should be able to register medicine use.

At the end of the meeting we discussed about what features we should include for the next sprint. We are ready for a new 2 weeks sprint!

Meeting Protocol #7

Subject: Weekly Meeting wit a supervisor

Date: 30 January, 2012
Place: HIG, A018B
Duration: 10:00 - 10:15 (15 minutes)

Participants: Adiljan Abdurihim, Andrius Januska, Simon McCallum

Todays meeting was mainly focused on Status Report and Second Sprint Term planning. The team has discussed some issues of design that made us to change taken decision because of lack documentation on framework (jquerymobile 1.0.min) and problems of getting functions work properly with a phonegap version 1.3.0. The time consume became more when it is awaited and tolerable.

Documentation on html, that is used by PhoneGap is little bit hard and we are waiting for some clear answer of performing documentation. Supervisor taking some contacts with university college staff to find more about it. The team will be informed as soon as it possible. The code and demo version of the app will be sent to supervisor at the end of this week for review. The code will be used for better understanding and finding a better solution for the code documentation.

The team according to the advice of supervisor will document all necessary changes and decisions with a detailed and reasonable explanation.

Written by Adiljan Abdurihim

Meeting Protocol #8

Subject: Sprint Midterm Meeting

Date: 07 February, 2012
Place: Remotely, Skype
Duration: 10:00 - 10:19 (19 minutes)

Participants: Adiljan Abdurihim, Andrius Januska, Simon McCallum

On this meeting we have discussed about next issues that were most relevant.

We have been informed by our supervisro that “Datatilsynet” is toward new regulation that makes some of our functions to be reconsidered. “Datatilsynet” will not allow norwegian institution to use GoogleApps.(read: http://www.datatilsynet.no/Nyheter/2012/Nei-til-bruk-av-Google-apps/).
So, it has been advised to gather more information in order to be able to use GoogleMaps or at least an open source alternative Open street map. (see: http://www.openstreetmap.org/). Here it is important to find information about the processing data send by an user. How much and how long is stored this information and what the data is used for.

It is still not clear about the idea of a good code structure that’s why it is advised to send some bunch of code to Jayson, Runo and supervisor in order to get a feedback and possible solution.

Written code can be sent to supervisor in order to test it on iPhone. But the SVN access would be a better solution for continuous check and easier update. The IT of school will be contacted in order to create an account for Simon McCallum.

Email to “Datatilsynet”, Catharina Nes will be sent today to get feedback and build up a communication with the right person. (see the attachment “email Datatilsynet”).

Documentation for HTML is kinda tricky and maybe it is best to use inline comments.Otherwise the possible documentation creator for javascript files is found and will be tested.

Written by Adiljan Abdurihim

Meeting Protocol #9

Subject: Weekly meeting with supervisor

Date: 13 February, 2012
Place: HIG, A018B
Duration: 10:00 - 10:30 (30 minutes)

Participants: Adiljan Abdurihim, Andrius Januska, Simon McCallum

During this remote Skype meeting with our supervisor we discussed several project related issues.

Before the meeting Simon tried to test the app on the iPhone virtual device, but it seemed to have some issues to run Phone Gap on XCode. Finally, during the meeting we succeeded and we could test the app on the iPhone virtual device. The problem was, that it was chosen a wrong version of virtual device. The main targets that we wanted to test - scrolling, fixed footbar, database and GUI elements - seemed to function us expected. There was just a slight difference on some GUI elements as compared to Android phones.

We have discussed how to name in English our “oppdragsgiver” Randi Stokke in the documentation, because the word “employer” that we used before is not exactly the same as “oppdragsgiver” in Norwegian. Possible alternatives would be “Subject matter expert”, “Project definer”, “Project initiator”, “Project coordinator” or “End user representative”, depending on how we consider her and our relation to the project.

We got a tip from Simon that Latex would make the documentation job easier, and it’s probably worth to start using it now. Simon also adviced us to contact Tom Røise to discuss project progress in relation to the requirements.