

Alin Mihai Napirca
Thomas Øvsttun Ditman

Medicine Reminder System

Bachelor's thesis in Bachelor's thesis in Bachelor in Programming

Supervisor: Peter Nussbaum

May 2023

Alin Mihai Napirca
Thomas Øvsttun Ditman

Medicine Reminder System

Bachelor's thesis in Bachelor's thesis in Bachelor in Programming
Supervisor: Peter Nussbaum
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science



Medicine reminder application

Authors

Alin Mihai Napirca
Thomas Øvsttun Ditman

CC-BY 2023/05/21

Summary of Bachelor Thesis

Medicine is an important part of modern society, providing everything from the prevention of life-threatening diseases to vitamin supplements. Shortcut came to us wanting to create a solution for the common mistake of forgetting to take the medication at the correct time. This would be solved through a mobile application for Android using Ultra-wideband beacons. The project was developed using Scrum methodology, with a focus on modern practices. While the task was only for a proof of concept, usability was a focus through the use of Material Design for the user interface. The project was written using several coding languages for different devices, from Kotlin for Android, to Swift for iOS. The project put focus on including users in the design process through the use of several rounds of user testing in various stages of development. One of our main focuses was on the use of Ultra-wideband and the technology tied to them. The project was successful in the completion of the proof of concept application.

Sammendrag av Bacheloroppgave

Medisin er en viktig del av det moderne samfunnet. Medisin gjør alt fra forebygging av livstruende sykdommer, til vitamin tilskudd. Shortcut kom til oss med et ønske om å skape en løsning til et vanlig problem, nemlig å glemme å ta medisiner til riktig tider. Denne løsningen ble skapt som en Android programvare med bruk av UWB beacons. Prosjektet tok til bruk SCRUM metodikk, med et fokus på moderne tankegang. Selv om oppgaven bare var en proof of concept applikasjon, så ble det puttet et fokus på brukbarhet gjennom innføring av Material Design for brukergrensesnittet. Prosjektet var skrevet i flere forskjellige kodespråk, fra Kotlin for Android, til Swift for iOS. Prosjektet puttet et fokus på inkludering av brukere i design prosessen, med flere runder av brukertester i forskjellige faser av utviklingen. Et av hovedfokusene var å ta til bruk de modern UWB beacons og teknologien knyttet til dem. Prosjektet klarte å produsere et akseptabelt sluttprodukt gjennom en proof of concept applikasjon.

Preface

We would like to thank our supervisor Peter Nussbaum for his incredible supervision and availability. We also wish to thank Shortcut for not only giving us the privilege of working on their project, but also for letting us have access to their office as well as extra meetings and lectures. Furthermore we would to thank our contact person at Shortcut, Ram Kumar, and especially the lead Android developer Jason Toms for his incredible help for the coding in the project.

In the end we would like to thank each other on the group for a job well done and teamwork committed.

Table of Contents

Summary of Bachelor Thesis	iii
Sammendrag av Bacheloroppgave	v
Preface	vii
Table of Contents	ix
Figures	xiii
Tables	xv
Code Listings	xvii
Glossary	xix
1 Introduction	1
1.1 Background	1
1.2 Project description	2
1.3 The importance of medication	2
1.4 Limitations	2
1.4.1 Product limitations	3
1.4.2 Marketing limitations	3
1.4.3 Time limitations	3
1.5 Users	3
1.6 Group members	3
1.6.1 Academic background	4
1.6.2 Motivation	4
1.7 Roles	5
1.7.1 Student Roles	5
1.7.2 Shortcut	5
1.7.3 Supervisor	5
1.8 Thesis structure	7
2 Development Process	9
2.1 Development method	9
2.2 Choice of method	9
2.3 Implementation	10
2.3.1 Meeting with host & supervisor	10
2.3.2 Internal meetings	10
2.3.3 Issue board	11
2.4 Sprint overviews	11
2.4.1 Short overview of each sprint	12

3	Requirements	15
3.1	Use cases	15
3.1.1	Actors	15
3.1.2	User stories	17
3.1.3	Advanced use case	18
3.2	Backlog	19
3.3	Domain model	19
3.4	Operational requirements	19
3.5	Security Requirements	20
4	Technologies	23
4.1	Beacon technology	23
4.1.1	Estimote	23
4.1.2	iBeacon	24
4.1.3	Eddystone	24
4.2	Android	24
4.3	iOS	25
4.4	Room	26
5	Design and Implementation	27
5.1	Structure	27
5.2	Kotlin	31
5.2.1	Data	31
5.2.2	Composables	31
5.2.3	Dependency Injection	32
5.2.4	Room functions	34
5.3	Swift	38
5.3.1	SwiftUI	38
5.3.2	Beacon Code	39
6	User Interface	43
6.1	Material Design	43
6.2	Navigation	44
6.3	Upcoming	45
6.4	Medication add wizard	45
7	User testing	47
7.1	First iteration	47
7.1.1	User tests: Round 1	47
7.2	Final product	48
7.2.1	User tests: Round 2	49
8	Code Quality	51
8.1	Code review	51
8.1.1	Code reviews from professional	51
8.1.2	Example of results from code review	52
8.2	CI/CD	54
8.2.1	Detekt	55
8.2.2	Unit testing	56

9 Discussion	59
9.1 Development Process	59
9.1.1 Scrum	59
9.1.2 GitLab	60
9.1.3 Atlassian	60
9.2 Technical Design	61
9.2.1 User Experience Design	61
9.2.2 Jetpack Compose	62
9.2.3 Dependency Injection	62
9.2.4 Beacons	62
9.2.5 Android API 23	64
9.2.6 Problem-solving through Pair Programming	65
9.2.7 Teamwork	65
9.2.8 Time allocation	67
10 Conclusion	69
10.1 The good	69
10.1.1 The results	69
10.1.2 Database	70
10.1.3 User experience	70
10.1.4 Koin	70
10.1.5 iOS solution	70
10.1.6 Teamwork	71
10.2 The bad	71
10.2.1 Beacon troubles	71
10.2.2 Design of the application	72
10.2.3 Lacking desired functionality	72
10.2.4 Notifications	72
10.2.5 Testing coverage	72
10.3 The future	73
10.3.1 Optical character reading	73
10.3.2 Future app integration	73
10.3.3 iOS version	73
10.4 Final conclusion	74
Bibliography	75
A Additional Material	77
A.1 Initial Project plan	79
A.2 NTNU Project Agreement	93
A.3 Project Proposal	101
A.4 Reference Group Contract	105
A.5 All wireframes	107
A.6 User Tests Round 1	109
A.7 User Tests Round 2	123
A.8 student timeallocation	131

A.9 Meeting Minutes, all meeting minutes can be viewed in the project repository 140

Figures

1.1	The different people/organizations part of our bachelor thesis. The students are illustrated in red, the NTNU supervisor is illustrated in blue and the contact persons from Shortcut are illustrated in green.	6
2.1	A picture illustrating the project backlog from the Jira hub which the students abode by.	11
3.1	Use case diagram for user interaction in the application	16
3.2	Example of backlog for sprint six.	20
3.3	Domain model for the project.	21
4.1	Room architecture.	26
5.1	System components and connections.	29
5.2	System components and connections.	30
5.3	Gitlab repository structure.	30
5.4	LazyList of the log of medicine taken, filled with dummy data.	33
6.1	Before and after examples of buttons from the migration	44
6.2	The "Upcoming" page with the navigation bar highlighted.	44
6.3	MaterialTheme DatePicker in action	45
7.1	A portion of the wireframe used during the first round of user tests. To see the rest of the wireframe please refer to Appendix A.5	48

Tables

1.1	A table showcasing the different subjects that the students deemed relevant for the bachelor project to succeed.	4
3.1	User Story 1.	17
3.2	User Story 2.	17
3.3	User Story 3.	17
3.4	Use case (advanced).	18
4.1	Comparison between UWB and BLE Beacons, from [6].	24
9.1	A table to showcase the areas and how much time was spent within those areas by the students.	67

Code Listings

1

class without Room annotations.312

example from History.kt.323

Dependency Injection initialization.344

Dependency Injection Definition.355

Data class with Room annotations.366

converter definition.377

Entity Data class.378

With Medication.38 9

SwiftUI struct for the application, logic removed.4010

for the initialization of the Beacon in Swift.41 11

solution for storing variables.5212

solution for storing variables.5313

code for Upcoming.kt.5414

for code for Upcoming.kt.5515

of the pipeline file that compiles the Kotlin code for later testing.5616

of several warnings from Detekt.5617

of the pipeline file that handles linting test.5718

of the test for the groups time converting function.57 19

of the pipeline file that handles unit testing test.58

Glossary

Android Operating system for Android devices. iii, 2, 3, 24–27, 31, 43, 45, 51, 53, 60, 62–64, 67, 69–71, 73, 74

API Application Programming Interface. 3

beacon Signal emitters used for fine-tuning position. iii, 1–3, 15, 17, 18, 20, 23, 24, 26, 28, 39, 60, 62–64, 67, 69–71

CI/CD Continuous Integration, Continuous Deployment. 54, 55

Continuous Integration and Continuous Deployment A method to frequently deliver solutions to customers by introducing automatic updates into the stages of app development. 12, 54

Dependency Injection (DI) Design pattern in which an object or function receives other objects or functions that it depends on. 32, 70

Estimote A polish company that produces beacon technology such as UWB and BLE. 12, 23

Flutter An open-source UI software development kit. 74

Gitlab Open source code sharing site, used for keeping track of the history of the project. 11, 54, 56, 60

iOS Operating system for Apple devices, most commonly used in iPhones and iPads. iii, 13, 19, 24–27, 38, 51, 63, 64, 70, 73, 74

Jetbrains Czech company which makes tools for software developers and project managers. 74

Jetpack Compose Modern Kotlin toolkit for building native UIs for Android. 4, 31, 38, 62, 66

Jira Issue tracking solution by Atlassian. xiii, 11

- Koin** Dependency Injection method for Android applications. 32, 34, 70
- Kotlin** Statically-typed programming language mainly used in the development of Android based solutions. iii, 4, 12, 25, 31, 32, 39, 43, 62, 67, 70
- Kotlin Multiplatform Mobile** Technology simplifies the development of cross-platform projects. 74
- Material Design** Google recommended framework/guidelines for usability and design. iii, 43
- NTNU** Norges Teknisk-Naturvitenskapeige Universitet. xiii, 4, 6, 7, 10, 56, 60, 63, 66, 67
- OCR** Optical Character Recognition. 72, 73
- optical character recognition** Conversion of an image of text into a readable text format to avoid the need to manually transcribe. 72
- Room** Local database solution for Android. 26, 27, 31, 34, 37, 70
- Scanbot** A data capture Software. 73
- scrum** Development methodology commonly used in Agile programming. iii, 5, 9–11, 59, 60
- SDK** Software Development Kit. 12, 24, 26, 60, 62–64, 69, 71, 73
- Shortcut** Company hosting the bachelor. iii, xiii, 1, 2, 4–6, 9, 10, 13, 45, 51–53, 60–67, 69, 73
- SQLite** Library implementing a small, fast, self-contained, high-reliability, full-featured, SQL database engine. 70
- Swift** Programming language mainly used in the development of iOS based solutions. iii, xvii, 25, 26, 38, 41, 70, 71
- SwiftUI** Modern Swift toolkit for building UIs for iOS devices. 25, 38, 64, 67, 70
- test-driven development** Development process that focuses on creating tests before developing the actual code.. 72
- UI** User Interface. 31, 32, 43, 62
- Ultra-wideband** Low frequency emitter technology.. iii, 1, 2, 23, 69
- user-centered design** Iterative design process where the user and their needs are the focus of the design process. 72

UWB Ultra-wideband. xv, 15, 17, 23, 24, 43, 69, 71

UWBManagers Class for getting UWB capabilities and interacting with nearby UWB devices. 74

wizard A step by step process of installing or inserting information. 18, 45

Chapter 1

Introduction

This chapter will introduce the reader to the group and their capabilities. The chapter will also go into the project itself, explaining the objective, scope, boundaries, and more.

1.1 Background

Medicine is something most people will have to take at regular intervals at some point in their lifetime. This entails taking a specific dosage at a specific time of day, sometimes multiple times a day. One of the main problems with taking your medicine is remembering to take it or remembering if you have already taken it.

To help with this there are already applications and other systems to help mitigate this problem. These typically send you a notification to your personal smartphone at the appropriate times of the day. This isn't the cure-all that was hoped for. This is because getting a notification to take your medicine does not mean you will take it or are even able to take it.

For this project, Shortcut proposed the group to develop a proof of concept for an Android application and system to help with this problem. This will be done through the use of an Android application as well as Ultra-wideband beacons, -hereby referred to as UWB Beacons- and other smaller smart devices such as smartwatches.

The main difference or selling point of our project is the ability to give reminders to you based on your location. The existing products can only remind you to take your medicine without considering if you are in a situation capable of taking your medications. Our project will take location into consideration to make sure you are able to take your medication when reminded to.

1.2 Project description

For this project, Shortcut asked us to create a proof-of-concept medicine reminder system for Android devices that makes use of additional contexts, such as wearables (i.e smartwatches) and location by making use of Ultra-wideband beacons (UWB Beacons) to increase the likelihood of a successful medication intake at a given time. In order for the system to succeed it has to guide the user in making the medication intake into a habit, and as per the literature of BJ Fogg *Tiny Habits* [1], the successful creation of a habit requires three components which the application has to fulfill. These are:

- Availability: At the time of medication, is the medicine available?
- Prompt: At the time of medication, is the user reminded to take their medicine?
- Motivation: At the time of medication, is the user incentivized to take their medication?

Additionally, if the scope permitted, the group was also asked to document the CE marking evaluation for the app in order to correctly market it as a medical app if it qualifies as such, and to document the process by taking into account the pitfalls of CE marking for a developer of a medical application and mitigations that can be built in the application to convey these. The main criteria for a successful project was the creation of the proof of concept application.

1.3 The importance of medication

Though the application developed shall only serve as a proof-of-concept for a medication reminder system, it is important to emphasize the consequences of missing a medicine intake for a person that relies on their medicine to be healthy. For some that may be life-threatening, such as for diabetics and asthmatics, while others may suffer from less threatening symptoms, everyone is subject to forgetting their medicine intake, be it they are in a meeting, or far from home and their medicine or they have run out of medicine without them realizing. With the application developed the students intend to focus on creating a habit out of taking their medicine, and by using a digital system to organize one's data, use of Ultra-wideband beacons to contextualize the user's position in regards to their medication, and multiple prompts throughout the day to remind them of their medication. Though motivation may be difficult to achieve as it varies from person to person the students emphasize cheerful notifications and prompts to help ease the user into taking their medication.

1.4 Limitations

There are three limitations to establish for the project.

1.4.1 Product limitations

Due to the use of beacon technology, the minimum Android API is API 18. Additionally, the use of user permissions and other features such as a date picker and a time picker through the `kotlinx.datetime` library requires a minimum Android API of 24. This API level is high enough to cover all the libraries needed for the application to operate, and also low enough to be able to be run on almost all Android devices currently in circulation.

With that being said, the scope of the bachelor thesis will be an application limited to only Android devices. It was initially planned that the scope could be expanded to include an iPhone application. This ended up happening to a limited scope due to difficulties that defined in section 9.2.4.

1.4.2 Marketing limitations

Due to the application possibly falling under the definition of a medical device, requirements set by *CE Marking* [2] will have to be followed in order to correctly label and market the application as such.

1.4.3 Time limitations

Finally, due to the nature of the project, it is important to follow the development plan and respect the milestone deadlines to keep the project on track. The project, both the application and report have to be finished and delivered by the 22nd of May.

1.5 Users

The target group for this application is elderly people, especially those who rely on taking their medicine regularly, however, it can also be used by a younger audience that relies on regularly taking their medication. On a broader scope, anyone who wishes to have a more digitized and structured system for their medication intake is viewed as the target group. This required the application to be easy and intuitive to use due the potential user group including non-technologically inclined individuals.

1.6 Group members

The student group consists of two students, namely:

- Alin Mihai Napirca
- Thomas Ditman

1.6.1 Academic background

Both students have attended the BSc in Programming at NTNU in Gjøvik and have experienced a variety of programming languages, programming paradigms, technologies, and solutions throughout their tenure at NTNU, which served as a great foundation for this project and its focus on the use of the Kotlin and the Kotlin library Jetpack Compose. For this project, the following subjects were especially relevant for the project to succeed:

Table of Subjects:

PROG2007 - Mobile Programming	Overall knowledge about mobile programming with an emphasis on Kotlin and Android devices
IDATG2204 - Data modeling and database systems	Knowledge about databases and data modeling with a focus on SQL queries and database APIs
PROG2052 - Integration Project	Project management through Scrum and personal software development goals
PROG2006 - Advanced Programming	Introduction to various programming paradigms and overall programming knowledge
IDG1362 - Design Thinking	Knowledge about user-centered design and user experience
PROG1004 - Software Development	Introduction to agile methodologies and basic principles in software architecture and Design

Table 1.1: A table showcasing the different subjects that the students deemed relevant for the bachelor project to succeed.

1.6.2 Motivation

One of the students in the group, Thomas, had a good relationship with Shortcut from a previous summer stint with the firm and asked personally for a project that could suffice as a BSc project for the group. Shortcut accepted and created this ambitious project for the students which sparked a lot of excitement to undertake a task in the medical field because of the impact it can have. The medical field is extremely sensitive and gives the sense that whatever contribution is made will help in making every day easier for the people that would end up using the group's solution.

1.7 Roles

Though the group consisted of only two members it was important to assign roles and responsibilities to all participants in this project.

1.7.1 Student Roles

Thomas Ditman will act as the group leader. He will be in charge of making sure the project is progressing and will be the primary contact person with Shortcut. He will keep track of meetings, schedules, and time management.

Alin Mihai Napirca will be the group's Scrum Master. His responsibilities will be to lead the Scrum meetings; Sprint Planning, Daily Scrum, Sprint Retrospective, and Sprint Review. Additionally, he will be maintaining the issue backlog and making sure the group follows the agile framework of application development.

Both group members will also be developers. This means that as a group they are responsible for taking on and completing issues in the product backlog for each sprint.

A visual representation of this can be viewed in Figure 1.1.

1.7.2 Shortcut

Shortcut AS is the employer and the contact persons are Ram Kumar and Jason Toms. There were Scrum meetings held through Slack with them as well as a weekly meeting to update them on progress, problems, and struggles regarding the project.

1.7.3 Supervisor

For this bachelor's thesis, the group's supervisor will be Peter Nussbaum. There have been established weekly meetings on Wednesdays in the beginning with the possibility to shift to a bi-weekly meeting once the project is in gear.

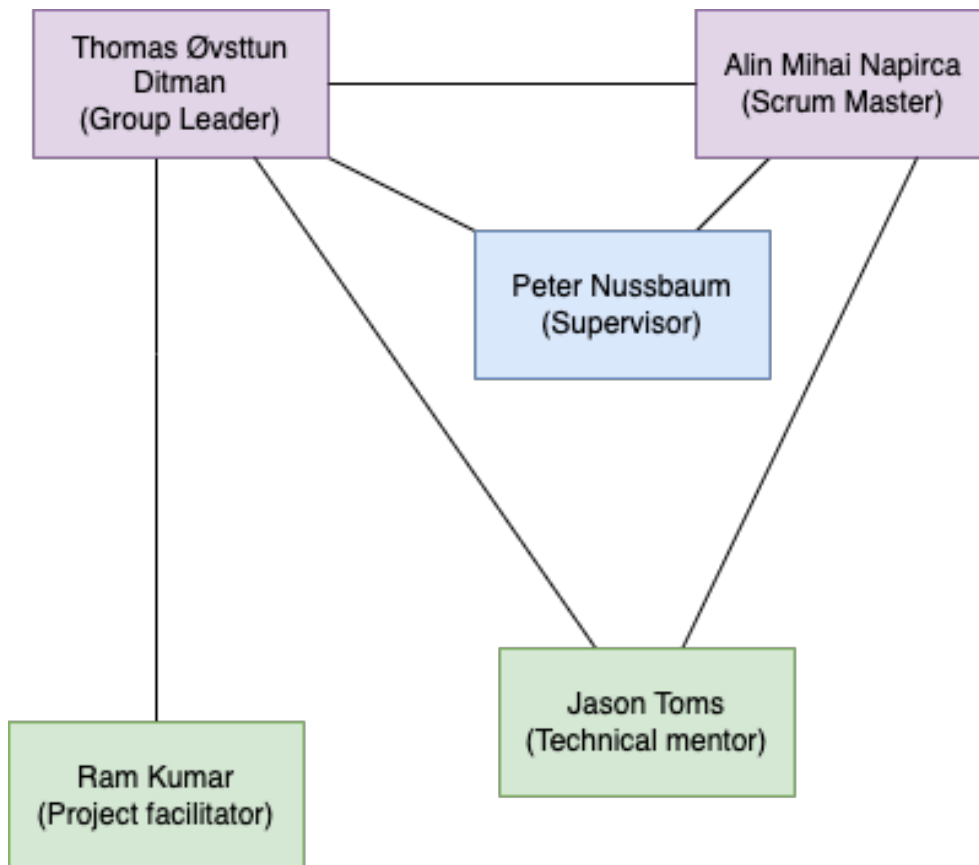


Figure 1.1: The different people/organizations part of our bachelor thesis. The students are illustrated in red, the NTNU supervisor is illustrated in blue and the contact persons from Shortcut are illustrated in green.

1.8 Thesis structure

The structure of the thesis makes use of NTNUs template for student theses at a BSc and MSc level [3], however, due to internal discussions between the students and the supervisor it has been slightly altered to better suit the reading experience of the thesis. The thesis contains ten chapters, each containing a multitude of sections and subsections. Throughout the thesis, the students make use of multiple figures, tables, and code listings to visualize and solidify certain arguments and points, where all of these, and additionally a glossary, can be found in their respective lists as entries above the introductory chapter. The sequential order of each chapter is as follows:

- **Introduction:** The introductory chapter where the task, the employee, and the student group are presented.
- **Development process:** The chapter where the group's planning process and a road map of the project period are presented.
- **Requirements:** This chapter covers the various actors interacting with the application, and the specifications of the application regarding operation and security.
- **Technologies:** In this chapter, the technologies used by the students are presented together with reflections about why those were chosen, and what other solutions there were.
- **Design and Implementation:** Chapter where the students present the technical structure of the project and focus on the decisions taken in regard to coding the application.
- **User Interface:** The chapter contains important choices and thoughts regarding the user interface.
- **User testing:** This chapter focuses on user tests and their effect on the final product.
- **Code Quality:** This chapter takes a closer look at source code written by the students, its function, and its importance to the application.
- **Discussion:** This chapter contains the student's reflections about various choices that had to be made across the project period.
- **Conclusion:** This closing chapter contains the student's thoughts on the final product; what went well and what could be improved, the project process, and future work. Additionally, it covers the student's experience of the BSc project.

Chapter 2

Development Process

There are many different choices that a project can take in terms of how time will be spent, what activities will be done, and how communication will be used. This chapter will go into specifics of what the group primarily chose as the development method.

2.1 Development method

A development method is a type of framework that is used to chart the activities that were used during the project. There are many different types of development methods that are used depending on the characteristics of a project. Something all methods have in common is their purpose, to make a system for progress. You can find a project plan for this project for how the group planned for the project to go in appendix A.1. This chapter will describe what method the group used and the group chose that method.

2.2 Choice of method

The specifications and requirements given by Shortcut for the project were quite open-ended and loose. This led to the group being able to set most of the requirements for themselves. The requirements the group set for themselves included a flexible schedule for milestones, with heavy communication between the group and the client at Shortcut. Due to these factors, the team chose the Scrum framework as our development method.

The use of sprints with a well-defined backlog made the entire process flexible. This is due to having been able to shift around issues based on the priorities depending on how critical they were in a given sprint. The group established consistent meetings with the supervisor and Shortcut to keep the client included and invested in the development process. This led to a good flow of ideas between the client and the group.

2.3 Implementation

Throughout the project, the group used two-week long sprints. This was due to the high amount of communication desired between the supervisor and the client. This meant that the group could get frequent feedback and comments regarding decisions. This is contrast to typical sprints in Scrum that are one week long. Two weeks was chosen due to experience from a previous project by the students were one week sprints were used. This was because the group felt more comfortable updating the supervisor less frequently.

2.3.1 Meeting with host & supervisor

Throughout the project meetings were held regularly between the team members, the supervisor, and a contact from Shortcut. These meetings were held at the same time every other week, barring a few exceptions late in the project.

Sprint review meetings

It was established early on that there would be a sprint review meeting during during the weekly meetings directly following the end of a sprint. Present at these meetings would be both of the members of the team, as well as a representative from Shortcut, and the supervisor from NTNU. The main focus of these meetings was to communicate our progress during the previous sprint with exactly what tasks had been completed. The other purpose was to establish what the next sprints' tasks had been established to be.

2.3.2 Internal meetings

Primarily there were two types of internal meetings used in the project. One of these was a sprint planning meeting, and the other was more general work meetups.

Sprint planning meetings

Sprint planning meetings were held at the start of each sprint. The main purpose of these meetings was to establish the tasks for the upcoming sprint, and who was to be assigned to said tasks. Each sprint had an established backlog to pull tasks from. These tasks were generally broad in scope, with fewer larger tasks rather than smaller more specific tasks. Each member of the group was allowed to simply pick the tasks they wanted to work on.

Work meeting ups

Several times a week meetings were held between the two team members set aside simply to work together. These meetings were primarily to make sure help was

accessible between the two members, as well as being able to bounce ideas off each other. In retrospect, this also heavily helped the morale of the team and led to greater team cohesion. These meetings were organized in an ad hoc manner, mostly a day or two ahead of time.

2.3.3 Issue board

To be in line with the Scrum development method, the group made use of an issue board to track what tasks were being worked on that sprint. There were a couple of different options for an issue board solution that were considered. Using the Gitlab issue tracker¹ was an initial proposition. This was decided against due to the team wanting to use a tool more in line with what would be used in the real world. Another option that was considered was Kitemaker.com², as it was a solution that the team already had some experience with from a previous project. Kitemaker was decided against however due to the team desiring to use a more commonly used tool. To meet the criteria of a modern commonly used tool the team settled on using Jira³ as an issue board, as well as a time tracker.

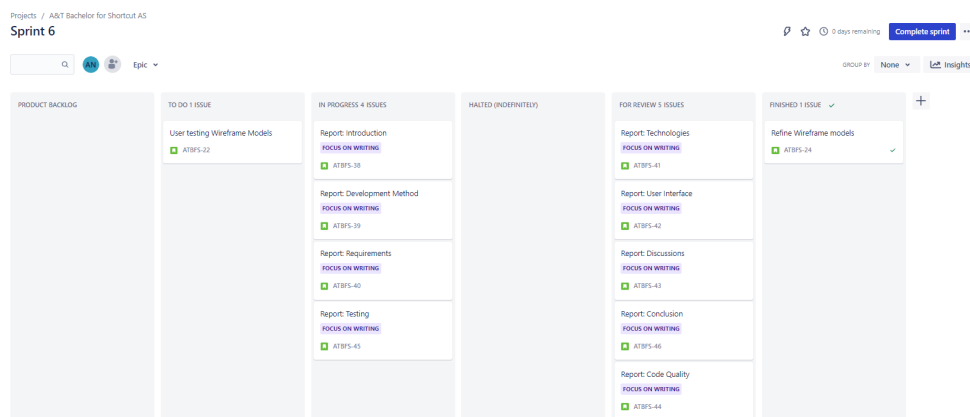


Figure 2.1: A picture illustrating the project backlog from the Jira hub which the students abode by.

2.4 Sprint overviews

Since the team decided to use the Scrum development method, there was heavy use of sprints during the project. Already in the project planning phase a loose idea of what tasks were to be done during each sprint. These sprints were subject to change throughout the project as tasks were either under or overestimated.

¹Gitlab has some functionality regarding issue trackers and setting them up as issue boards.

²An modern solution for issue tracking and collaboration.

³A site and software by Atlassian commonly used to organize work and meetings.

2.4.1 Short overview of each sprint

This section will be a short overview of what was actually worked on during each sprint.

Sprint 1: Project planning 11. Jan - 1. Feb

This was the only sprint to deviate from the normal two-week length. This was due to wanting to start the second sprint on the first of February. This sprint was spent finalizing the requirements and project description, as well as deciding the tools that were to be utilized in the project. The plan developed can be seen in appendix A.1.

Sprint 2: Design and setup 3. Feb - 16. Feb

This sprint was primarily spent designing wireframe models and setting up the coding environment for the rest of the project. At this point, the team was ahead of schedule, which led to time being spent setting up the Continuous Integration and Continuous Deployment⁴ pipeline for the project.

Sprint 3: Development 17. Feb - 3. Mar

The first steps of the coding process were undertaken during this sprint. During this sprint, the emphasis was primarily on getting the user interface of the application working, without any of the logic in the background being needed. This means essentially setting up the skeleton of the code, with empty view models⁵ being created for each page.

This was also the first sprint where user tests were conducted, a process that would last over the next several sprints.

Sprint 4: Logic 4. Mar - 17. Mar

This was when the functionality of the application started being developed, with more of the back end being developed. This was when the database of the application and persistence started becoming usable.

Sprint 5: Problems 18. Mar - 2. Apr

This was the sprint where the largest amount of research was done into the Estimote beacon technology SDK. This led to the discovery of a major issue with the SDK. From a meeting during this sprint, it was decided what would be done to solve this problem. This problem is gone more into detail in section 9.2.4. Outside of the discovery of the problem, this sprint was mostly spent working on

⁴Continuous Integration, Continuous Deployment

⁵Classes in Kotlin that are used for storing logic and data related to a given page.

the functionality of the application, with a focus on the notification system and dependency injection.

The beginning part of the report was also at this point being written, with the simple sections covered such as Technologies and Requirements as they were already covered in previous documents required for this project.

Sprint 5.5: Easter 3. Apr - 12. Mar

Initially, a plan for an Easter break was set during this time frame. This plan had to be adjusted due to the addition of the solution to the problem discovered in the previous sprint. Easter break instead had to be used to research and produce a beacon solution for IOS. It was labeled sprint 5.5 due to not being an initially planned sprint.

Sprint 6: 13. Apr - 30. Apr

This sprint was spent finishing the application. Primarily done by making all the pieces coded up to this point integrate together in a clean way. The biggest challenge during this sprint was the sheer amount of merge conflicts that had to be fixed between the team members.

This sprint was also used to present the resulting application to both the supervisor and a lead Android developer at Shortcut.

Sprint 7: 1. May - 21. May

The final sprint of the project was solely dedicated to writing the bachelor report. The final draft was sent to the supervisor for approval, and finally fully delivered.

Chapter 3

Requirements

This chapter will go more into detail as to what the requirements were for the project. This will be done through the use of use cases, domain models, and more in-depth explanations of requirements.

3.1 Use cases

Use cases are, "a written description of how users will perform tasks on your website"[4]. This section explains how use cases were used for this project, as well as the factors important to use cases. The use case diagram constructed for this project is found in figure 3.1

3.1.1 Actors

USERS

A user will interact with the app and register their medication and beacons. The user uses his smart devices in conjunction with UWB beacons in order to receive push notifications as reminders in order to get the user to take their medication.

SMART DEVICES

Smart devices are objects such as smartwatches, phones, and tablets that will send the user notifications when it is time and place for them to take their medication. This will be activated when the user is close enough to their medication and when a beacon has picked up on the user's location to remind the user that it is time to take their medicine.

UWB BEACONS

Beacon are "alarm clocks" that will remind the user that it is time to take their medication when the user is nearby their medication, but also when some time

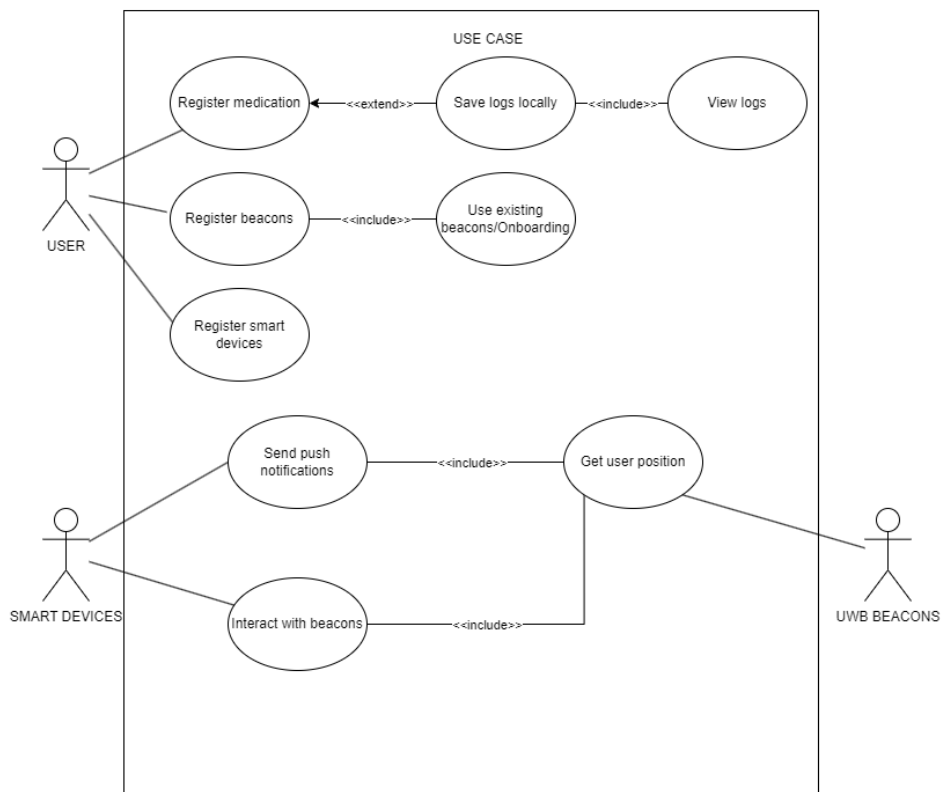


Figure 3.1: Use case diagram for user interaction in the application

has passed since the user has been in the vicinity of their medication. Beacon are gone more into detail in section 4.1.

3.1.2 User stories

The user stories are issues taken directly from our product backlog and will be used in order to describe how the team sees the functionality and actions that will be performed by a user. This however expects the user to be using one or more smart devices and also that the user already has a setup of UWB beacons.

Table 3.1: User Story 1.

User Story: A user wants to add medicine to his medicine reminder app	
Actor:	User
Goal:	Add a new medicine with frequency and dosage
Explanation:	The user will move to the main menu where they will find a few buttons that they can click on. One of them will be labeled "Add new medicine" whereupon clicking the user will be prompted with a new menu. In this menu, the user must fill some mandatory fields in order to register this medicine. These fields are "How often do you need to take your medicine?", "How much medicine do you need to take at each dosage?" and "Over which period are you taking your medicine?". Upon filling all these fields the user can register their medicine in the medicine reminder app.

Table 3.2: User Story 2.

User Story: A user wants to add new beacons or onboard existing beacons	
Actor:	User
Goal:	Add a beacon to improve localization of user
Explanation:	The user will move to the main menu where they will find a few buttons that he/she can click on. One of them will be labeled "Add beacon" whereupon clicking the user will be prompted with a new menu.

Table 3.3: User Story 3.

User Story: A user wants to check their medicine history	
Actor:	User
Goal:	Allow the user to check the period of their medication
Explanation:	The user will move to the main menu where they will find a few buttons that they can click on. One of them will be labeled "Add beacon" whereupon clicking the user will be prompted with a new menu. In this menu, the user can view the history of their medication period through a daily view where each day saves the time slot and what medicine was taken at that given moment.

3.1.3 Advanced use case

Table 3.4: Use case (advanced).

User Story: A user wants to check their medicine history	
Actor:	User
Device:	Mobile phone
Primary Actor:	User
Interest:	For the user to add data to the system so they can start using the application
Explanation:	The user will move to the main menu where they will find a few buttons that they can click on. One of them will be labeled "Add beacon" whereupon clicking the user will be prompted with a new menu. In this menu, the user can view the history of their medication period through a daily view where each day saves the time slot and what medicine was taken at that given moment.
Type:	Essential
Pre condition:	User has the medicine and prescription in front of them so they can accurately input the data in the application
Post condition:	The user has added chosen medicine info in the application
Program flow:	<ol style="list-style-type: none"> 1. The user presses the Manage Medication button 2. New page appears and the user presses the Add new medication button 3. A field wizard appears which the user has to fill to complete, but can quit at any time 4. The user fills in all fields and completes the wizard 5. User is taken back to the Manage Medication page where they can see the newly added medication on their page 6. The user has the ability to view medicine info, add new medication or change page to view additional application functionality

Additional functionality:	<p>The user presses on medicine to view info</p> <ol style="list-style-type: none"> 1. The user presses on the medicine on their screen 2. They can view data such as dosage, frequency, remaining medicine 3. Close the window by pressing on the Return button <p>The user presses on the button labeled Upcoming</p> <ol style="list-style-type: none"> 1. User presses the Upcoming button and is taken to the home page 2. The user can view the time till the next medicine dosage and status on whether or not they have taken their medicine for morning, midday, and evening 3. User can press the history button to view the whole report log of their usage of the app or maneuver to another page using the buttons at the bottom of the page <p>The user presses on the Settings button</p> <ol style="list-style-type: none"> 1. User is prompted with several settings regarding the app 2. Can turn on/off snooze and vibrations, change font size, change alarm sound, and customize hours for morning, midday, and evening
----------------------------------	--

3.2 Backlog

The backlog was an itemized list of all the tasks that needed to be fulfilled throughout the entirety of the project. During sprint planning meetings, items were moved from the backlog into the sprint issue board. During the project, new items for the backlog were discovered that needed to be added. An example of this happened when an iOS portion of the project was added before Easter. An example of the backlog for sprint six is shown in figure 3.2

3.3 Domain model

Shown in figure 3.3 is the domain model defined early on in the project. The domain model explains how each of the elements of the project should have related to-, and interacted with each other.

3.4 Operational requirements

Due to the nature of the project as a proof of concept, the group has had to make some important decisions concerning the operational requirements. In conjunction with Shortcut the team has figured out some points:

- The application will be a local application to the device, this means that the group will only need to accommodate 1 (one) user at a time.

Issue ID	Issue Title	Progress	Status
ATBFS-24	Refine Wireframe models	100%	FINISHED
ATBFS-41	Report: Technologies	50%	FOCUS ON WRITING, FOR REVIEW
ATBFS-42	Report: User Interface	50%	FOCUS ON WRITING, FOR REVIEW
ATBFS-43	Report: Discussions	50%	FOCUS ON WRITING, FOR REVIEW
ATBFS-46	Report: Conclusion	50%	FOCUS ON WRITING, FOR REVIEW
ATBFS-44	Report: Code Quality	50%	FOCUS ON WRITING, FOR REVIEW
ATBFS-22	User testing Wireframe Models	0%	GJØREMÅL
ATBFS-38	Report: Introduction	50%	FOCUS ON WRITING, UNDER ARBEID
ATBFS-39	Report: Development Method	50%	FOCUS ON WRITING, UNDER ARBEID
ATBFS-40	Report: Requirements	50%	FOCUS ON WRITING, UNDER ARBEID
ATBFS-45	Report: Testing	50%	FOCUS ON WRITING, UNDER ARBEID

Figure 3.2: Example of backlog for sprint six.

- Due to the project's nature as a medical aid, it is important that the app doesn't miss a reminder, due to this the team desired an up time of at least 99.5%.

3.5 Security Requirements

Security is an aspect that will be incredibly important for this project. This is primarily due to the nature of the project as a medical application. Personal medical data is host to a very large amount of laws and regulations. To avoid a lot of the issues that come with storing medical data the project will not make use of any cloud databases. This means that the only data being stored is data that is unavailable to anyone but the user of the application.

Due to the project being a proof of concept, the security aspect will take a bit of a side role.

- Data will be stored locally on the user's device, this mitigates the need to store data in a server which requires heavy security needs.
- There is no worry about privacy issues tied to a beacon, this is because a beacon cannot store any information tied to a user, the only thing a beacon can do is broadcast a signal [5].

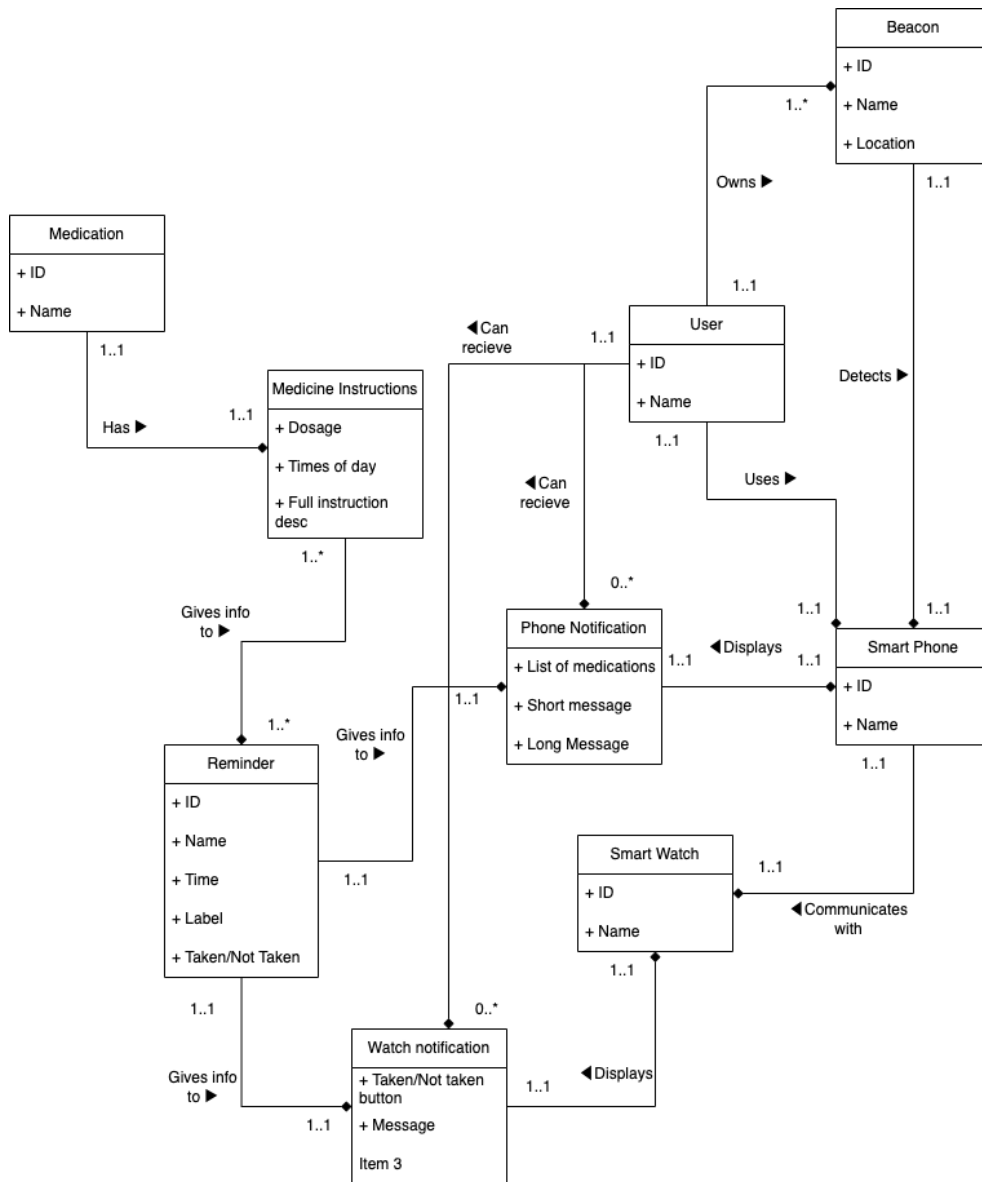


Figure 3.3: Domain model for the project.

Chapter 4

Technologies

This chapter will go over the various technologies used in the project, as well as similar options that could have been chosen.

4.1 Beacon technology

Beacon technology using UWB technology is a relatively new concept in the programming world. In the most basic sense, beacons is a Bluetooth device that functions on the Ultra-wideband protocol (UWB) [6]. This protocol uses short-range radio waves to communicate. UWB devices are incredibly good at communicating over short distances to share information such as distance between the devices. This is done by calculating the time it takes for the data to travel between the devices. Other major benefits of the UWB beacons is low power consumption, high accuracy, and low cost of production. In table 4.1 there is a comparison between UWB and Bluetooth beacons (BLE).

There are many different providers of beacons available to the public. Some of the most popular are:

- Estimote¹
- Ibeacon²
- Eddystone³

The difference between these options is the availability of API's and the kind of data being output from the beacons.

4.1.1 Estimote

Estimote is a company from Poland that specializes in UWB and Beacon technology. Estimote had previously worked with Bluetooth low energy (BLE) and wearable technologies. In recent years they have worked on producing beacons

¹<https://estimote.com>

²<https://developer.apple.com/ibeacon/>

³<https://www.mokoblue.com/all-about-eddystone-beacon/>

#	UWB	Bluetooth(BLE beacons)
Battery	Low consumption	Low consumption
Range	up to 200 meters (656 feet)	up to 70 meters (230 feet)
Accuracy	10 centimeters (3.9 inches)	up to a meter
Cost	Low	Low
Best For	Proximity Marketing, Customer Analytics, Indoor Navigation, Smart Homes, Factory Automation, Asset-Tracking, Logistics	Proximity Marketing, Customer Analytics, Loyalty, Indoor Location

Table 4.1: Comparison between UWB and BLE Beacons, from [6].

along with an SDK to use them. Until now this SDK has only been available for iOS, but an Android version is in production. This SDK has only been available to customers who have bought a development kit for the beacons.

4.1.2 iBeacon

iBeacon is Apple’s own incarnation of the beacon technology. iBeacon is primarily a software solution for beacons. This means that there aren’t specific beacons produced by Apple that can be configured and used by other developers for their own purpose. This enables other beacons to communicate with IOS devices. Apple does sell beacons, however these beacons are used only for specific purposes and are not programmable [7].

4.1.3 Eddystone

Eddystone is the Google implementation of beacon technology. It aims to be open and multiplatform in its purpose. This means that you can use Eddystone both on IOS and Android devices [8].

4.2 Android

Android is the name of the operating system that Android devices run on. These devices are primarily phones such as Google Pixel, Samsung Galaxy S22, or phones from lesser-known brands such as the One Plus 11g [9]. The Android OS also supports many other devices such as televisions, watches, and many more. This makes Android one of the most versatile operating systems on the market.

There are currently two main operating systems for phones in the current market. This is Android with about 70% of the market share and IOS with about 30% of the market share worldwide[10]. When comparing this to Norway this statistic actually flips to IOS being the most popular operating system for phones [11].

Android currently runs using Java as the main compiled programming language. This is due to Java already being intended as a language that is able to compile on any device using a JVM. However, Java is no longer the main intended programming language for Android . Kotlin was created by JetBrains in 2016 as a superset of the Java language. Kotlin was designed to "hit the same notes as the classic programming language (Java), while addressing all the flaws and limitations that make it unsuitable for the future"[12]. Kotlin is 100% inter-operable with Java, this means that you can write Java code inside a Kotlin file, and still have it compiled.

For the Kotlin code the team used a primary framework for the code called Jetpack Compose. Jetpack compose is a coding framework by JetBrains meant to modernize and simplify writing the UI for Android applications. This is done by shifting the way the code for the UI is written. In the earlier framework Views were used to write the UI. Views were written using XML code. With Jetpack Compose, the UI is instead written using entirely Kotlin code. Examples of Compose code can be found in section 5.2. The reasoning for using jetpack compose can be found in section 9.2.2.

Since the main focus of this group's project was making a proof of concept application for a modern technology the group decided that the more modern language would be optimal for this project. Due to this, as well as familiarity with the language, the team chose Kotlin as the main programming language for this project.

There is more on how Kotlin is used for the project in section 5.2.

4.3 iOS

IOS is the operating system that runs on Apple devices such as iPhones, iPads, AppleTVs, and Apple Watches. IOS is programmed using Apple's own programming language Swift. Swift was developed as a new solution to the previous language used for iOS, Objective-C. While Objective-C can still be used for IOS development and is even inter-operable with Swift, Swift is seen as a much more popular option for IOS [13].

One major change in the language somewhat recently is the popularisation of the SwiftUI programming framework. SwiftUI is to Swift, what Jetpack compose is to Kotlin. SwiftUI lets the developer program the UI of their application through the use of Swift code and structures, rather than using the previous UIKit⁴. Currently more and more companies and developers are switching over to SwiftUI,

⁴UIKit is the previous toolkit used for programming in Kotlin .

due to faster implementation, easier understandability, and less lines of code.

Initially when planning the project the group planned to not use Swift or program for IOS at all. It was seen as a stretch goal if it was discovered that there was an abundance of time available. The amount of time needed was much larger for Swift as neither member of the group had any large amount of experience with the language. Later on in the project, it was discovered due to some issues with the SDK used for the beacons that the team would use Swift for a part of the project as well. The reasoning for this decision is described in section 9.2.4. How IOS was used in the project is covered in section 5.3.

4.4 Room

Room is a persistence library provided by Jetpack as part of Android . It operates through the use of an abstraction layer for an SQLite database. Room works using three main components: The database class, database entities, and data access objects (DAO), as presented in figure 4.1. The database class is the main entry point for the connection to the persistent data. The database entities are representatives of the tables in the database. Finally, the database access objects provide the functions that let us manipulate the data in the database.

The actual use and implementation of Room in the project is covered in section 5.2.4.

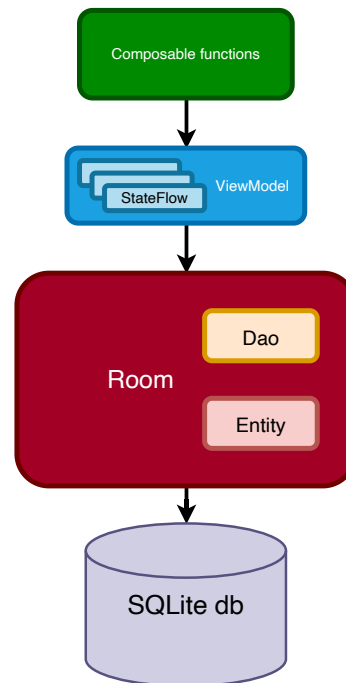


Figure 4.1: Room architecture.

Chapter 5

Design and Implementation

In this chapter, the design of the product will be explained and shown, with a focus on the decisions taken in terms of the actual code for the application.

5.1 Structure

For the project, it was decided on using a fairly standard solution for our system. One of the main features of this system is the intentional avoidance of any storing of data on any sort of non-local database. One of the large risks of the project was the fact that the application would classify as a medical application. This would bring obligations from the storage of medical information if the application is classified for CE marking. After going through the flowchart as seen in Figure 5.1, the group came to the conclusion that the application would be classified as a medical application. Due to the nature of personal medical data, the group decided to only use local storage mediums on the devices to avoid the security requirements of storing medical data on cloud servers such as Firestore¹. This was done by asking the user for permission upon launching the application the first time and storing all data locally on the user's device to avoid a security breach of information. This risk was initially discovered in the risk assessment in the project plan, found in Appendix A.1.

In Figure 5.2 one may see how each of the main components of the system communicated to each other. A notable part is how the communication between the Room functions and the database was through generated SQL commands. Also, note how the Android system and IOS system were completely separate from each other, this is due to how the project was separated as described in section 9.2.4.

The Gitlab repository is represented through Figure 5.3. Everything was contained in one repository, with folders separating the project into broad categories, with code being placed in the Android or IOS folders, and images and notes being placed in the other folders.

¹A google cloud service for an online NoSQL database

The in-depth CE Marking process is as follows:

1. Is the software a computer program?

Yes, a mobile application classifies as a computer program according to the definition on Britannica [14].

2. Is the software incorporated in a medical device?

No, the software is a standalone application for a mobile device.

3. Is the software performing an action on data different from storage, archival, lossless compression, communication or simple search?

Yes, the application is also constantly looking for the user's position through the use of Beacons and uses this context to give notifications and prompt the user to take their medication. Every time a notification is accepted by the user, i.e. when the alarm rings and the user takes their medicine, the data is updated.

4. Is the action for the benefit of individual patients?

Yes, the application is intended to give the patient a digital approach to a medicine reminder system and can be used as a substitute or together with a pill box, which is an analog medicine reminder system.

5. Is the app intended to be used for the purpose of diagnosis or treatment?

Yes, the application is intended to be used for the purpose of treatment as it reminds the user to take their medicine based on time and location contexts.

6. Is the app an accessory of a medical device?

Due to the fact that the answer to the previous step in the flowchart is yes, this step is avoided.

7. The software is regulated as a medical device.

This means the application has to be correctly CE marked.

8. Does the app contain a measurement function?

No, the application is only storing data about the medication, not about the patient's condition or medical examinations.

9*. **Action B:**

The app is considered to be a medical device with a low risk. For CE marking, the manufacturer has to follow the procedure of Annex VII MDD (self-certification)². This means that to market the application the students will have to fill out a form and provide the relevant information to the Medical Device Regulation (MDR) office and provide the relevant information so that a CE-mark can be assigned.

²The medical device regulation for the region of Europe and EU space can be found at: <https://www.medical-device-regulation.eu/2019/07/26/annex-vii-part-1/>



Figure 5.1: System components and connections.

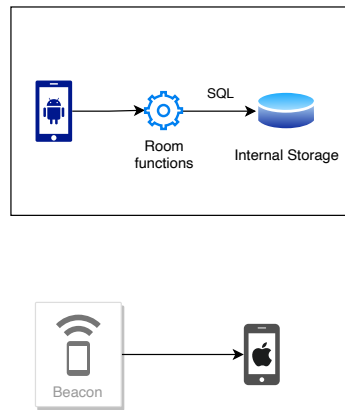


Figure 5.2: System components and connections.

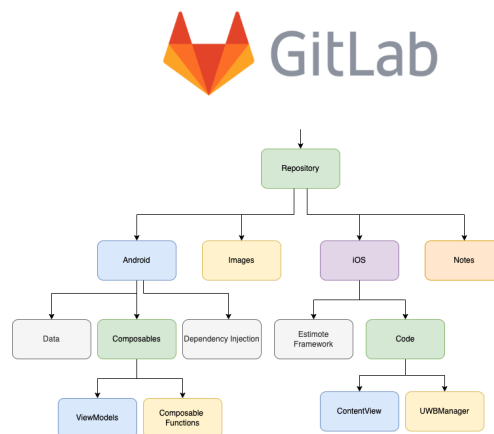


Figure 5.3: Gitlab repository structure.

5.2 Kotlin

The entirety of the Android application was programmed using the Kotlin programming language³. The Jetpack Compose library primarily defined our patterns in terms of programming. Compose is a programming framework for Kotlin that changes the way you write code. Initially, Kotlin wrote UI using a concept known as "Views", which is primarily written in XML code. Compose changes this by shifting all the code into functions written using Kotlin. This makes coding the UI much simpler and faster to do. Further reasoning as to why compose was chosen for the project is located in section 9.2.2.

Broadly speaking the Kotlin code is separated into four main components:

- Data
- Composables
- Dependency Injection
- Room functions

5.2.1 Data

Data is the broad category for which the group put any code that had to do with data, ranging from defining data types to anything relevant to the database. For instance, the Medication data class is defined in a sub-folder within the data package. An example of the medication data class is seen in code listing 1.

```
/**
 * Medication data class.
 */
data class Medication(
    val medicationId: Int,
    val name: String,
    val timeOfDay: String,
    val dosage: String,
    val description: String,
    val treatmentPeriod: LocalDate
)
```

Listing 1: Medication class without Room annotations.

5.2.2 Composables

Composable functions are the main draw of Jetpack Compose. The UI of the Android application was entirely built through their use. There were three main types

³Kotlin is a programming language which is a superset of Java, see section 4.2.

of composable function used for the project, Standard functions, MaterialTheme3 functions, and custom composables. Standard composables are functions that are supplied upon importing the library, these include functions like "Column" and "AlertDialog". MaterialTheme3 functions are composables that are designed by the Material Design team to fit in with their design framework. These included functions such as "Card" and any color use. The latter are the custom composables created by the group. These include "LocationPage" and "MedicationItem".

There were some more interesting functions used in our custom composables. These are for instance the "LazyColumn". The LazyColumn creates a Column where the contents are able to change and be defined dynamically. An example of this is shown in code listing 2.

```
@Composable
fun HistoryPage(viewModel: HistoryViewModel = koinViewModel()) {

    //Lists all items currently
    LazyColumn(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        items(viewModel.log) { item ->
            IndividualHistoryitem(item = item)
        }
    }
}
```

Listing 2: LazyColumn example from History.kt.

In the example above the dynamic list to be shown was the list of medications that had already been taken by the user in the past. The code above in conjunction with the IndividualHistoryItem composable produced the UI shown in figure 5.4.

5.2.3 Dependency Injection

Dependency Injection (DI) is defined as "a programming technique that makes a class independent of its dependencies. It achieves that by decoupling the usage of an object from its creation" [15]. For this project, DI entailed the creation of single instances of viewModels and Database access objects (DAO). This is due to issues being created from several instances of DAOs, or re-initializing DAOs.

For this project, Koin⁴ was used as the library for Dependency injection. For a detailed explanation as to why Koin was chosen please see section 9.2.3. The main difference is that Koin dependency injection is done at a project-wide level.

⁴A dependency injection library for Kotlin.

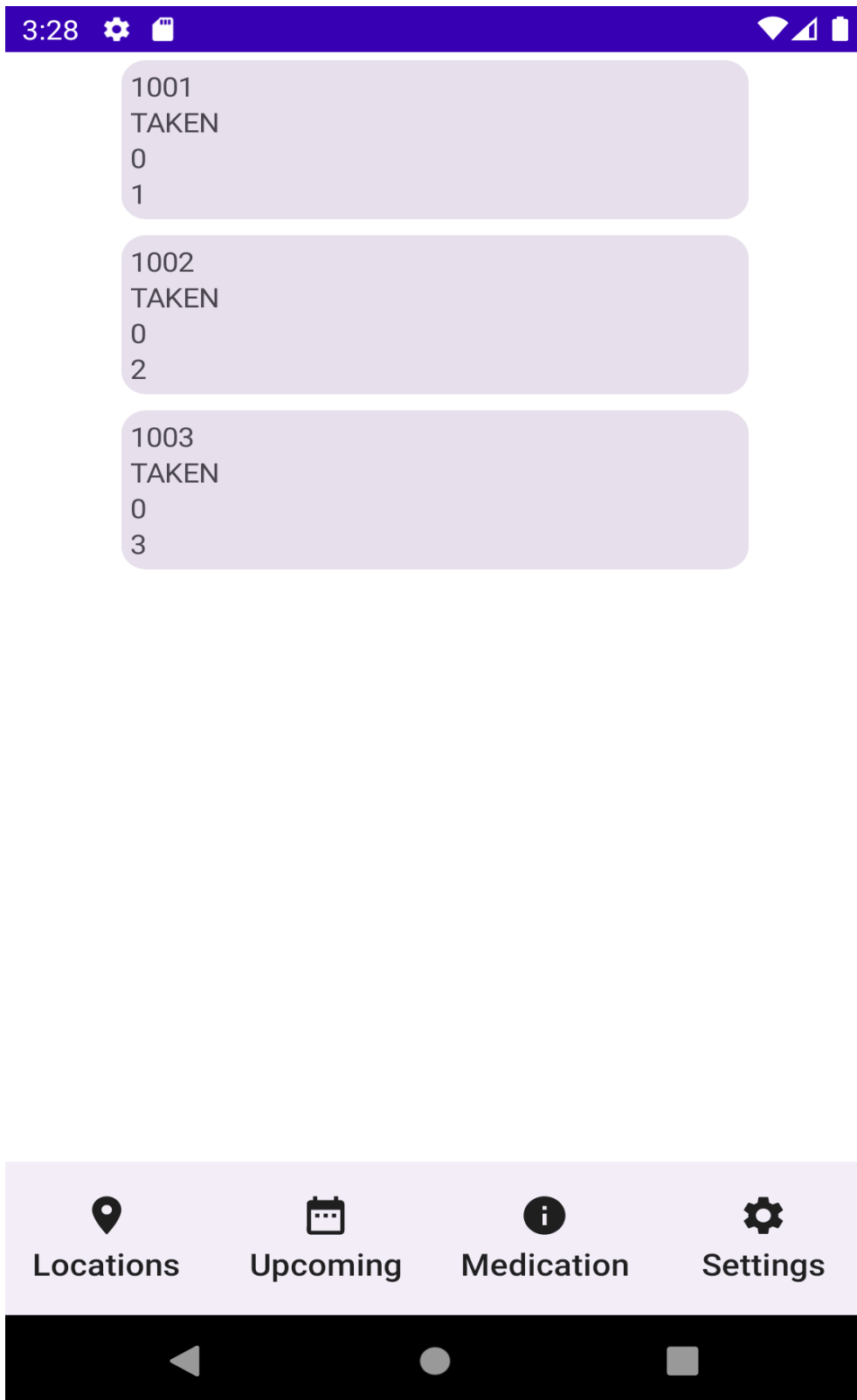


Figure 5.4: LazyList of the log of medicine taken, filled with dummy data.

```
class MainApplication : Application() {  
  
    override fun onCreate() {  
        super.onCreate()  
  
        startKoin {  
            androidContext(this@MainApplication)  
            androidLogger()  
            modules(appModule)  
        }  
    }  
}
```

Listing 3: Koin Dependency Injection initialization.

In code listing 3 it is shown how Koin wraps around the entire application for its injection.

In code listing 4 it is shown how the group defined a single instance of the database, as well as single instances of the medication viewModel. Typically there are difficulties getting parameters to a viewModel, having to use viewModel factories. This was entirely avoided using the Koin viewModel function where the function can simply define that the viewModel was to receive the parameters from Koin itself.

5.2.4 Room functions

Database and DAO

To achieve of persistence⁵ for the project was through the use of the Room library. This was done through the use of a single injected database instance generated from Koin dependency injection. This was done through the use of a "single" function, which creates a single instance of a section of code that is accessible anywhere in the project. The initialization is shown in code listing 4 This instance of the database was then used to initialize the various database access objects (DAO), which were then in the same way initialized as "single" instances in Koin.

Database entities

The database entities were set up using annotated data classes. Annotated data classes are data classes that generate code on compile time based on annotations

⁵By persistence, it is meant that certain data will stay persistent after closing the application and reopening it.


```
val appModule = module {  
  
    //creates a singleton database, whenever we access the database  
    // it will use this piece of code  
    single {  
        Room.databaseBuilder(  
            androidContext(),  
            AppDatabase::class.java,  
            "application-database"  
        ).build()  
    }  
  
    //Gets alarmManager instance  
    single {  
        androidContext().getSystemService(Context.ALARM_SERVICE)  
            as AlarmManager  
    }  
  
    //whenever we access the locationDao, we want to use this  
    single { get<AppDatabase>().locationDao() }  
  
    viewModel {  
        parameters -> DetailedLocationViewModel(  
            get(),  
            get(),  
            get(),  
            locationId = parameters.get()  
        )  
    }  
}
```

Listing 4: Koin Dependency Injection Definition.

added through the use of the "@" symbol. in code listing 5 it is shown that the entire data class was annotated with the "Entity" annotation and that the individual data values are annotated with "PrimaryKey" and "ColumnInfo" values. These were used to let the compiler know how the data was to be configured to an SQLite format.

Data converters

Throughout the project, the group wanted to use certain non-standard data types. LocalDate is a non-standard data class used to operate on date and time values

```
/**
 * Medication data class.
 *
 * Annotations are for defining how the data
 * class interacts with the database
 */
@Entity
data class Medication(
    //Primary key
    @PrimaryKey(autoGenerate = true)
    val medicationId: Int,

    //Name of medication
    @ColumnInfo(name = "name")
    val name: String,

    //Time of day for when medication is to be taken
    @ColumnInfo(name = "time_of_day")
    val timeOfDay: String,

    //How much medication to take in each dose
    @ColumnInfo(name = "dosage")
    val dosage: String,

    @ColumnInfo(name = "desc")
    val description: String,

    @ColumnInfo(name = "treatmentPeriod")
    val treatmentPeriod: LocalDate
)
```

Listing 5: Medication Data class with Room annotations.

without requiring the use of a higher API level. This had to be used due to Java's atrocious standard time API [16]. A problem occurs when it was directly attempted to annotate this data type into an Entity class. To solve this the team used a data converter as shown in code listing 6. Here the team explicitly defined methods to convert the non-standard data type into a data type that can be inserted into an SQLite database. In this case, the data type was a string value. There is a converter for each direction of operation, one for inserting values into the database, and one for getting values from the database.

```

class Converters {

    @RequiresApi(Build.VERSION_CODES.O)
    @TypeConverter
    fun fromTimeStampToDate(value: String?): LocalDate? {
        return value?.let { LocalDate.parse(it) }
    }

    @TypeConverter
    fun fromDateToTimestamp(date: LocalDate?): String? {
        return date?.toString()
    }
}

```

Listing 6: LocalDate converter definition.

Relations

The project also encountered some issues when the data types that were used included lists or arrays of items. Lists and arrays are not supported directly by Room since they would not fit neatly into an SQLite database. The solution to this was through the use of cross-referencing several tables to each other. For instance, a location would have a list of medications that were available at the said location as seen in code listing 7. In order for Room database to be able to store these medications properly the group had to implement a new entity data class that explicitly contains a reference to the location and the medication stored at said location. This was done according to Room's own recommendation on many-to-many relations [17] and can be seen in code listing 8.

```

/**
 * Data class for location data
 */
@Entity
data class Location(
    @PrimaryKey(autoGenerate = true)
    val locationId: Int,
    @ColumnInfo(name = "area_name")    val area_name: String?,
    @ColumnInfo(name = "beacon_name")  val beacon_name: String?,
)

```

Listing 7: Location Entity Data class.

```

/**
 * Relation between locations and medications
 */
data class LocationWithMedication(
    @Embedded val location: Location,
    @Relation(
        parentColumn = "locationId",
        entityColumn = "medicationId",
        associateBy = Junction(LocationMedicationCrossRef::class)
    )
    val medications: List<Medication>
)

```

Listing 8: Location With Medication.

5.3 Swift

Swift⁶ is the main programming language for IOS. Initially, the group was not intending to create anything with Swift, however, due to complications defined in section 9.2.4, the group decided on programming part of the project using Swift. This was a challenge due to the group's inexperience with Swift as a programming language.

5.3.1 SwiftUI

SwiftUI is functionally the equivalent of Jetpack Compose for Swift. Instead of using the old paradigm of storyboards to write the UI of the application, a different data structure is used in a code format. In SwiftUI this is done via "structs" which contain the functions, rather than Compose which has the functions loose, though the general concept and structure are quite similar.

In code listing 9 it is shown how the UI was implemented using SwiftUI. The UI itself was defined using the "body" variable, which includes a "VStack" with the elements for the UI.

Of note the "UWBManagerExample" variable is annotated with "ObservedObject", this annotation lets the UI know whenever the variable has updated so that the UI can update with the new element. For this project this means that whenever a new distance is from the UWB Beacon is observed the UI is updated with the new distance.

⁶Swift is a programming language developed by Apple for IOS programming, for more info see section 4.3.

5.3.2 Beacon Code

To construct the Beacon code for the project, the group took heavy use of the supplied example code made available from Estimote in the SDK. The code was adapted to fit the need of our project, with the inclusion of the "ObservableObject" extension for the class. This was required for the use of the "ObservableObject" annotation seen in code listing 9. Another element in the Beacon code was the addition of a "Published" annotation to the distance variable. This made the variable visible to the main application. in Kotlin terms this would make it a public variable. The code for the Beacon can be seen in code listing 10.

```

//Main view of the application
struct ContentView: View {
    //ObservedObject means that the UI will update
    //when this is changed
    @ObservedObject var uwb = UWBManagerExample()

    //This part displays the actual UI
    var body: some View {

        //VStack same as column essentially
        VStack {
            Spacer()

            //Button for requesting notifications
            Button("Request Notification Access") {
                //Logic code here
            }
            Spacer()

            //Sends a notification five seconds after button
            //is pressed, though you have to exit
            //the app for it to show
            Button("Push notification") {
                //Logic code here
            }

            Spacer()

            //shows current distance to beacon
            Text("\(uwb.currentDistance)")
                .padding()
            if (uwb.currentDistance < 1.0) {
                Text("You are within distance")
                    .padding()
            }
            Spacer()
        }
    }
}

```

Listing 9: Main SwiftUI struct for the application, logic removed.

```
//Object we use to access Estimote shenanigans
class UWBManagerExample : ObservableObject {
    private var uwbManager: EstimoteUWBManager?

    //Published so that we can access it in the View earlier
    @Published var currentDistance: Float = 0.0

    init() {
        setupUWB()
    }

    private func setupUWB() {
        uwbManager = EstimoteUWBManager(
            positioningObserver: self,
            discoveryObserver: self,
            beaconRangingObserver: self
        )
        uwbManager?.startScanning()
    }
}

// REQUIRED PROTOCOL
extension UWBManagerExample: UWBPositioningObserver {
    //Runs when position update happens
    func didUpdatePosition(for device: UWBDevice) {
        currentDistance = device.distance
    }
}

// OPTIONAL PROTOCOL FOR BEACON BLE RANGING
extension UWBManagerExample: BeaconRangingObserver {
    func didRange(for beacon: BLEDevice) {
        print("beacon did range: \(beacon)")
    }
}
}
```

Listing 10: Code for the initialization of the Beacon in Swift.

Chapter 6

User Interface

In the proposal document for this project as outlined in appendix A.3, the application was to be a proof of concept. For the project this meant that the user interface was not of great importance as an end goal. This is due to the focus being on the functionality of the application and utilization of the UWB technology.

Even though the focus was not on the appearance of the application, the group still wanted to establish a consistent user interface.

6.1 Material Design

Material Design is Google's open source design system [18]. Material Design advocates for a consistent design with user friendly interfaces, and works for a good user experience. Since both Android and Material design were designed by Google, Android makes heavy use of Material Design in even the baseline of programming Kotlin. This is through the use of "themes" and more specifically "MaterialTheme".

MaterialTheme is a method in which a developer can standardise the color and general design of their application. This is done through unified design files and classes that contain the variables defining color. These classes are then wrapped around the entire application, so that the colors can be called on when needed. This makes changing the color scheme of the application much easier, since you can just change these constant colors, instead of manually changing all of them one by one.

Initially in the project, the team used MaterialTheme iteration two for our UI. This worked quite well, with all the buttons acquiring their uniform appearances. Partway through our project however, the group was invited to attend a conference for Android developers. During this conference there was a talk on "Material Design 3" on how to migrate to it from existing Material Design 2 code-bases. After the conference the group agreed that migrating the code-base would be a good experience, and show off more modern design decisions. The difference is shown in figure 6.1.

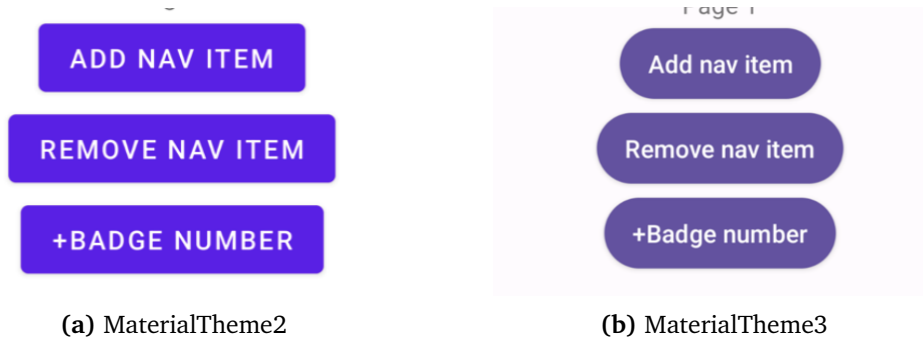


Figure 6.1: Before and after examples of buttons from the migration

6.2 Navigation

One of the most important elements of an application is how the user navigates around it. The navigation for the project needed to be available at all times, so that the user could easily get from page to page.

To this end the team implemented a navigation bar using the material library. This library contained both the desired functionality, as well as a pleasant design for the bar. The bar consists of four elements, each of which navigates to a new page. The elements consist of an image and some text, each of which are meant to be sufficient to know where the button will take the user.

The navigation bar element is one which the team worked very well. The code behind the upcoming bar is quite nice and clean, considering some of the challenges that had to be overcome. Certain navigation options needed to contain code that would lead to other pages. These were problems that were solved by passing functions as variables through the navigation functions.

An element which could have been better would have been to add cus-



Figure 6.2: The "Upcoming" page with the navigation bar highlighted.

tom images to the navigation bar. Currently these images just use MaterialTheme's standard images. If this proof of concept were to be developed in the future these icons would be custom to better fit the page they represent.

6.3 Upcoming

The upcoming page, shown in figure 6.2, had a few main elements to it. These are the medication status-, and upcoming medication cards. The status card contains the information and status of the current days medication. The color changes depending on if a medication for a given time of day has been missed, taken, or is soon supposed to be taken. The upcoming medication shows when the next medication for the day is to be taken. In the example figure, the next set of medication was to be taken nineteen minutes from when the figure took place.

6.4 Medication add wizard

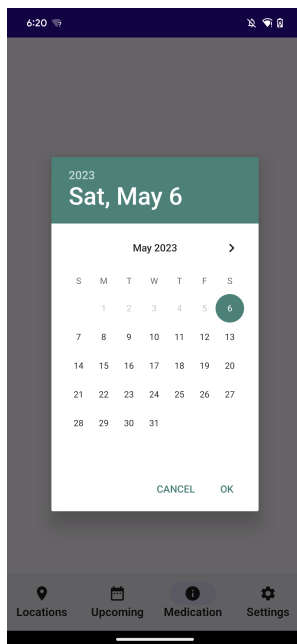


Figure 6.3: MaterialTheme DatePicker in action

An element of the application that the group was quite pleased with the result of was the "Add medication wizard"¹. This wizard went through many iterations and complications before the team was able to settle on the final result. Initially, the wizard functioned using a function called an "AlertDialog". This turned out to be a terrible solution due to the window moving about when highlighting an option inside of the dialog.

After a meeting in person with a lead Android developer at Shortcut, the wizard was brought up as a terrible implementation with a terrible user experience. It was recommended instead to implement a new navigation option, where a navigation manager, similar to the one used for the navigation bar, would be implemented. This solution had some hardships along the way but turned out to be a much smoother and better experience to use. The new solution even gave the ability to deny certain actions to the user that would have caused issues for the application.

¹A wizard refers a window or set of windows to take the user step by step through a process.

One of the steps in the wizard included selecting a date for the end date of the prescription of the medication. The implementation of this was initially quite terrible, with having to type in a number using a keyboard for day, month, and year. Later the team discovered that Material Theme had a composable function for just this use, with a "DatePicker" function. This turned out to be a good solution for this step of the wizard. The date picker can be seen in figure 6.3.

Chapter 7

User testing

During the following chapter, the focus will be on the students' approach to user tests; describing the methodology used during the user tests, and how the results were used to assess the functionality and intuitiveness of the design and then improve the application according to the feedback received.

7.1 First iteration

The group heavily emphasized getting several test users during the development, as it would lay the foundation upon which the application was built. With this in mind, the first course of action for the group was to develop a set of wireframes to present to the test users. This was done by each student individually where they both created 'their' version of how they envisioned the application to present itself and then the group came together and merged the two solutions to get the best of both ideas. With this product, the first round of user tests ensued and this laid the foundation for the development of the application.

7.1.1 User tests: Round 1

During the first round of testing the main goal was to get as many users as possible to test the interface of the application. As the medicine application is intended for older audiences, preferably some that use or have used some sort of medication regularly, it was important to consider them as the target audience as to ensure that the product the students developed would meet their needs and preferences. Nevertheless, it was hard to find elderly people that have a history of using the medication regularly beyond friends and family, and thus the initial round of testing consisted of users of various ages and backgrounds, from students to retired.

Since the solution used for user testing was just a wireframe there was no real functionality tied to the different buttons, thus the main focus was not to test how the application will operate, but rather get feedback with regards to visibility, intuitiveness, and choice of colors & icons for different tasks. A sample image of which wireframes were used in the first round of user tests can be seen in Figure

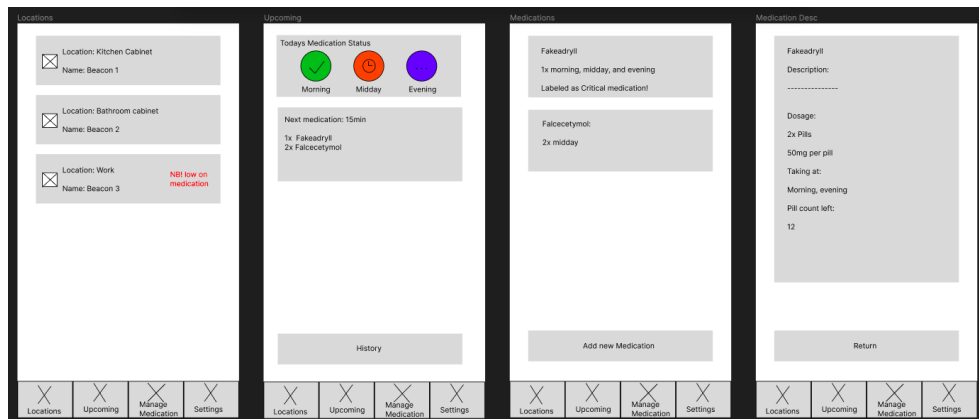


Figure 7.1: A portion of the wireframe used during the first round of user tests. To see the rest of the wireframe please refer to Appendix A.5

7.1. During the user tests the students received feedback regarding design and the test users' opinions on notifications and localization using beacons. The user test results can be viewed in Appendix A.6.

It quickly became obvious that the younger the test users the less they had to offer in terms of constructive feedback because of their inexperience in having to take regular medication, however, as the sample of test users was limited every single test user was an important asset for the students to get an overview over what was intuitive and well designed, and what needed to be improved or reworked.

By the end of the first round of user tests the students had a group of roughly 20 participants which made for a considerate sample size to garner feedback from and improve the wireframe design. Those who were keen on seeing the project progress were offered a contract by the student group where they agreed to be called for a second round of user tests sometime near the end of the project period, one such contract can be seen in Appendices as A.4 Reference Group Contract.

7.2 Final product

During the end of the bachelor project period, a second round of user tests was conducted, this time with a finished product heavily influenced by feedback received during the first round of user tests, but also by feedback from the supervisor Peter and employees from Shortcut namely Jason and Ram. The second round of user tests was conducted in early May and was used to get the test users' perspective on how they would use the application, its intuitiveness, and its effectiveness by showcasing how the reminder system works.

7.2.1 User tests: Round 2

For the second round of user tests the focus was on how the feedback received from the test users was translated by the students and applied to the application solution. This time, the questions were more aimed towards the test user using the application and how intuitive it was to conduct certain tasks such as adding a new medication to the reminder system and how to change the alarm for different times of day sections; Morning, Midday, and Evening. With the use of the Reference Group Contract mentioned previously, the test users were the same as for the first round of user tests. To showcase the progress made from the wireframe design to the application solution the students presented the test users with their answers from round one of user tests to refresh their memory and then the students urged the test users to conduct a few actions within the application to see how intuitive the design was, and then afterward discuss the progress made from a wireframe to an application.

The feedback received through the second round of user tests was positive and the test users were pleased with the progress made, however, it is to be noted that not all test users from round one could participate in the second round of user tests. A full display of the second round of user tests can be viewed in Appendix A.7.

Chapter 8

Code Quality

For this project the group wanted to have an early focus on the quality of the code. This was done primarily due to previous projects at university not having a focus on this part of the development process.

Due to the late integration of the IOS portion of the project, that portion was not able to get the same code quality treatment as the Android segment.

8.1 Code review

One of the simplest and most effective measures that was implemented for code quality assurance was the requirements of merging code. Before any member of the group could merge code into the main branch, the code would need to be looked over, approved, and most importantly understood by both student in the group. This ensured that every member of the team was on the same page regarding the code, every step of the way.

This worked quite well throughout the project, with the main purpose being achieved of keeping the understanding of the code united. The group did encounter issues with this process however, stemming from differences in sizes of merge requests. Some members of the team had smaller and more frequent merge requests, while others had larger merge requests. This ended up leading to more merge conflicts than originally anticipated, which meant more time spent fixing merge conflicts. This could have been fixed or mitigated through communication regarding the size of commits.

8.1.1 Code reviews from professional

Starting with week eight, the teams contact with Shortcut offered to start taking looks at our code. This had several purposes. One was to make sure that the group was following best practice code conventions in the Android application. Another was to come with suggestions on how to simplify code with solutions that the group was unaware of. The final was for Shortcut to keep up with the progress being made on the project.

8.1.2 Example of results from code review

The best example of improving best practices in the code for the project occurred when the View models were looked at. Initially all the variables for a View model were stored as standard variables, visible to all. This can be seen in code listing 11. The contact from Shortcut mentioned that this is generally bad practice, and pointed to the solution of using "View States" to store the data for a view model. The result the group ended up with is seen in code listing 12.

```
class MedicationViewModel(  
    private val dao: MedicationDao  
) : ViewModel() {  
    // Initialization of values:  
    // wizardInput = Wizard pop-up data  
    var wizardInput = mutableStateOf(WizardInput())  
  
    // showDialog = boolean to show alert,  
    // turns true on button press  
    var showDialog = mutableStateOf(false)  
  
    // wizardStep = Wizard step -> step 1 is the  
    // name of the medicine  
    var wizardStep = mutableStateOf(1)  
  
    // isEmpty = empty check variable -> true means field is empty  
    var isEmpty = mutableStateOf(true)  
}  
  
data class WizardInput (  
    var name: String = "",  
    var dosage: String = "",  
    var description: String = "",  
    var timeOfDay: String = "",  
    var treatmentPeriod: String = ""  
)
```

Listing 11: Initial solution for storing variables.

```
class MedicationViewModel(  
    private val dao: MedicationDao  
) : ViewModel() {  
  
    private val _viewState: MutableStateFlow<MedicationViewState> =  
        MutableStateFlow(MedicationViewState())  
  
    val viewState = _viewState.asStateFlow()  
  
    /** Setter functions **/  
}  
  
data class MedicationViewState(  
    val name: String = "",  
    val dosage: String = "",  
    val description: String = "",  
    val timeOfDay: String = "",  
    val treatmentPeriod: LocalDate = "2023-06-01".toLocalDate(),  
    val showDialog: Boolean = false,  
    val wizardStep: Int = 1,  
    val isEmpty: Boolean = true,  
    var medicationList: List<Medication> = listOf<Medication>()  
)
```

Listing 12: Ultimate solution for storing variables.

An instance where the code was improved through a the use of a concept the group was unaware of occurs in code listing 13. In this example the history button was supposed to be aligned to the bottom of the screen. The team went through many solutions until a lead Android developer at Shortcut mentioned that a simple solution would be through the use of a "Spacer" composable with a "weight" of 1. The solution is shown in code listing 14.

```
@Composable
fun UpcomingPage() {

    //Column containing items for the page
    Column(
        modifier = Modifier
            .padding(all = 20.dp)
            .background(MaterialTheme.colors.background)
    ) {
        //Card containing the medication
        // status of the taken medications
        MedicationStatus()
        UpcomingMedicationCard()
        HistoryButton() //TODO: Make align to bottom of screen
    }
}
```

Listing 13: Initial code for Upcoming.kt.

This process was invaluable for the quality of the project, and was one of the most effective methods of learning for the project. It showed the incredible value that comes from having a senior developer available for insights from a more experienced individual.

8.2 CI/CD

CI/CD or more generally known as "Continuous Integration and Continuous Deployment" is a term for autonomous checks of the code before the code goes into deployment [19]. For this project the use of CI/CD was something the team wanted to use early and use often, to learn more through emulating an actual work environment. To this end, the group took use of Gitlabs own built in CI/CD pipelines.

Gitlab pipelines is a code infrastructure tool that allows for pieces of code to run when certain criteria are met. For this project this meant that Detekt would run every time code was merged into the main branch. In code listing 15 you can see how this code is set up in the gitlab-ci.yml file for the Kotlin code. The code triggers a "runner"¹ to build a version of the project, and then run certain tools to check if the code passes certain requirements. The use of these infrastructure tools is something the group utilised from the IIKG3005 Infrastructure as Code course.

¹This uses some computing power from one of the developers' PC to run the piece of code

```

@Composable
fun UpcomingPage(
    viewModel: UpcomingViewModel = koinViewModel(),
    toHistoryNav: () -> Unit
) {
    val viewState = viewModel.viewState.collectAsState()
    //Column containing items for the page
    Column(
        modifier = Modifier
            .padding(all = 20.dp)
            .background(MaterialTheme.colorScheme.background)
            .fillMaxSize()
    ) {
        //Card containing the medication status of the taken medications
        MedicationStatus()
        UpcomingMedicationCard()
        Spacer(modifier = Modifier.weight(1f))
        HistoryButton(toHistoryNav = { toHistoryNav() })
    }
}

```

Listing 14: Solution for code for Upcoming.kt.

8.2.1 Detekt

Detekt is a static code analysis tool for Kotlin [20]. It is a program referred to as a "linter", which is a term used for tools that have been used to find "smelly"² code. This kind of code generally also doesn't follow good coding conventions. Using a linter helped eliminate these potential risks. To emulate a CI/CD pipeline the group used Detekt. Code listing 16 gives an example a warning from detekt. Code listing 17 shows the lint yml code that ran everytime the branch was merged into the main.

In the end this was a foolproof solution for the project. There were many warnings that occurred throughout the project that the team disagreed with. Warnings that simply did not matter would still show up in the error log. Minor warnings do not make the test fail however, which still made this linting process a benefit in the end, since it helped find some fairly large errors in the code.

Detekt also caused some headaches due to the merging process. Due to the nature of the runner used for the project being a local runner on a team members computer, the detekt checks could not happen unless said computer was open and unlocked. This was a major issue that should have been fixed through the use of

²"Smelly" code is a term used for code that has a high potential for bugs or exploits.

```

image: java:8-jdk

before_script:
  - export GRADLE_USER_HOME=`pwd`/.gradle
  - export ANDROID_HOME=/Users/thomasditman/Library/Android/sdk

cache:
  key: ${CI_PROJECT_ID}
  paths:
    - .gradle/
    - .gradle/wrapper
    - .gradle/caches

stages:
  # List of stages for jobs,
  # and their order of execution
  - build
  - test
  - deploy

```

Listing 15: Part of the pipeline file that compiles the Kotlin code for later testing.

```

w: file:medicinereminder/composables/upcoming/Upcoming.kt:39:5
Parameter 'toHistoryNav' is never used

```

Listing 16: One of several warnings from Detekt.

a cloud runner using NTNU's instance of openstack. A cloud runner was decided against due to the limited importance of linting for the project.

8.2.2 Unit testing

Unit testing is an invaluable part of the testing process. Through the use of Unit testing the developer makes sure that the code is doing what it is supposed to be doing. This is done by logic tests to make sure that the actual result you get is the same as the result you are getting. In listing 18 the code is showing how the team tested the function converting a string value to a time value. The code to run the unit tests in the Gitlab pipeline is shown in code listing 19.

Ultimately the unit testing portion of the project was a part of the project that went heavily under-utilized. Unit testing was only written for a limited set of logic functions and database queries. The team should have invested more time and effort into the unit tests, as well as implementing more types of tests, such as UI tests that would run actions on the UI. Testing was a part of the project that

```
lint-test-job: # This job also runs in the test stage.
  stage: test # It can run at the same time as unit-test-job
            # (in parallel).
  only:
    - merge_requests
  script:
    - echo "Linting code... This will take about 10 seconds."
    - cd code/
    #- detekt --config config/detekt/detekt.yml
    - ./gradlew check
    - ls
    - echo "No lint issues found."
  artifacts:
    paths:
      - /code/app/builds/reports/lint-results-debug.html
    expire_in: 1 week
```

Listing 17: Part of the pipeline file that handles linting test.

```
@Test
fun testHourStringToIntConversion() {
    val expected = 12

    val testString = "12:00"

    val actual = getTimePairFromString(testString)

    assertEquals(expected, actual.first)
}
```

Listing 18: Part of the test for the groups time converting function.

was strived for early on, but never achieved success due to lack of time invested.

```
unit-test-job: # This job runs in the test stage.
  stage: test  # It only starts when the job in the build
               # stage completes successfully.
  only:
    - merge_requests
  script:
    - echo "Running unit tests... This will take about 60 seconds."
    - cd code/
    - ./gradlew :app:testDebugUnitTest
```

Listing 19: Part of the pipeline file that handles unit testing test.

Chapter 9

Discussion

This section of the report is reserved for discussions regarding the development process and environment, design and technology choices, issues and how they were resolved, and how this impacted the final solution.

9.1 Development Process

During the planning phase, it was clear that a good foundation had to be laid for the rest of the project to flourish. Initially, multiple tools and software were tested to find what would be best suited for the group, but there were also a few obvious choices that were predetermined which will be discussed later in this chapter.

The project started with a heavy focus on documenting the Project Plan and Specification of Requirements documents, before starting to develop the app by scoping out wireframe models to conduct user tests. As mentioned in section 7 the user tests played a crucial role in the group developing a user-oriented application to satisfy the vast majority. The group tried to stick closely to the Gantt chart, however, the pace of development fluctuated throughout the project period resulting in changes to the product backlog and thus to the development cycle.

Toward the end of the project period, the focus switched from development to project writing. During the application development cycle, the group did small additions to the final project report to alleviate the workload toward the end. This laid a strong foundation to begin writing and made the last month of the project period easier.

9.1.1 Scrum

The chosen framework to structure the development period was Scrum. Scrum provided a great flow throughout the duration of the project work as it had clear checkpoints for each sprint and served as a great tool to maintain good progress throughout the project period.

As an agile framework, Scrum gives the opportunity to backtrack and rethink certain aspects, but also advance forward and work with tasks outside the current

scope which proved very useful as the progress fluctuated over the course of the project due to various issues such as a missing beacon SDK for Android which will be discussed later in section 9.2.4. The weekly Sprint Review and Sprint Retrospective meetings were a great channel to receive feedback and engage in sparings with both the supervisor and Jason or Ram from Shortcut to improve both documentation and the codebase. Throughout the whole project, the target was a weekly meeting which proved to be adequate. Nevertheless, not every single meeting was as productive and this is due to issues addressed in earlier meetings being already solved or sometimes updates were given to both the supervisor and the contacts at Shortcut through e-mails or through the Slack communication channel. All in all, Scrum proved to be a great tool and was useful throughout the project.

9.1.2 GitLab

Gitlab was an easy choice for the group as it has been the solely used tool for storing codebases throughout the student's tenure at NTNU. Even though Gitlab offers several features for planning, issue- & time tracking and security management it was mainly used as a collaborative version control tool for the project's files such as figures, documentation, wireframes and source code. As the students were familiar with Gitlab it provided no frustrations and was overall a great tool to use.

9.1.3 Atlassian

Initially, the only software used from Atlassian was to collaborate and structure project work. Firstly, the group only used Jira, however, by testing multiple software and tools it branched to also use Confluence and Tempo, more on that in the next paragraphs. Atlassian proved to be a nice platform because it served as a multi-tool for the logistics part of the project where both the group, the supervisor and Shortcut could have easy access to the current backlog, see meeting reports, and time spent on different tasks which improved communication between all parties.

Jira

Jira served as the main hub for all Scrum-related work. Here the group laid plans for each sprint, created issues and set up work to be done for the product backlog. As mentioned in section 2.3.3, the group had previously used another tool called Kitemaker¹ to fulfill these goals, however Jira proved more complete as it is part of the Atlassian suite. The advantage that Jira provides is that it linked up meeting minutes and time tracking together to specific issues all in on a single platform.

¹Kitemaker is a different tool used to keep track of issue boards.

Confluence

As mentioned in the earlier paragraph Atlassian offers a tool to document and write meeting minutes which is named Confluence. Confluence allowed for open collaboration between the students, the supervisor, and the employees at Shortcut where everyone had all notes related to meetings within the Jira workspace which also provided direct reference to the relevant topics. There were other solutions tested at the beginning of the project such as Google Docs and Evernote which worked fine, however, Confluence allowed for multiple solutions to co-exist within one space by using Atlassian, and worked as the best way to organize and store project notes and gave easy access to these documents to all participants.

Tempo

The last tool used from the Atlassian suite was Tempo to track time. In previous projects, time tracking was done manually with the help of spreadsheets which was not an effective nor organized method, thus the group spent a good amount of time early on to find a good solution. Tempo proved as a great solution because it was easy to implement into the existing workspace on Jira where time could be directly linked to an issue from the product backlog. This integration made Atlassian ideal for the solution the group was aiming for and was overall a great experience.

9.2 Technical Design

The technical design was a critical aspect that had to be addressed before and during the development of the application. This meant there was a heavy emphasis on how the application should be developed, the development environment, libraries, and tools used during the development period, but also how problems and issues appeared and how they were addressed.

9.2.1 User Experience Design

There was a heavy focus on receiving feedback from user tests to adjust the application to suit the users' needs. The user tests were conducted from the very beginning of the project period to get a good grasp of how the users envision their ideal medicine reminder system. Thus, the CI/CD pipeline was established to be able to continuously push an updated version of the application to receive feedback and upgrade it, which was the loop by which the development of the application iterated. Additional tests were conducted at the end of the application development period to compare how the first version and the final version compared and look at differences. The results of choosing to follow UX design paradigm are explained in section 10.2.5.

9.2.2 Jetpack Compose

This project came with a strong recommendation to use Kotlin for the development of the application, and more specifically the up-and-coming toolkit for Android user interface (UI) development Jetpack Compose. The chosen platform for the development of the medical reminder application is Android and Android devices thus making the preferred programming language to use Kotlin. However, since Kotlin is still a relatively new programming language it led to a debate about developing the application with vanilla Kotlin or Jetpack Compose which had to be assessed to best fit our project.

Vanilla Kotlin offers familiarity and stability for the students as it was covered during the PROG2007 - Mobile Programming course and would allow for a seamless start to the development of the application. Vanilla Kotlin also makes use of XML Sheets which to this day [2] is the recommended way to create and mold UIs in Android development. On the other hand, Jetpack Compose offers efficiency and flexibility to its developers [21] and is regarded by many to be the future of UI development in the Android ecosystem. With the introduction of Jetpack Compose the use of XML sheets will likely diminish and the overall use of Jetpack Compose in the Android development community will likely increase.

All in all the choice was to use Jetpack Compose and there is little negative to say about it. The development process was seamless and there were no frustrations rooted from Jetpack Compose itself as there were lots of resources and documentation which were easy to navigate and convert to our use.

Shortcut is also an avid supporter of Jetpack Compose which made it easier to ask questions regarding development problems. Additionally, this gave meaningful constructive feedback and guidance concerning this new coding paradigm.

9.2.3 Dependency Injection

As a dependency injection solution, the chosen service was Koin. Koin offered a lightweight framework that was easy to learn and use as the syntax is in Kotlin, and did not add overhead as it does not live as a layer of another library [22]. Additionally, Shortcut advised using Koin as it covers all the necessary areas for our project and could offer assistance if issues were to occur. The other option was to use Hilt, however, Hilt is a way more complex library as it provides flexibility and customizability. Nevertheless, the out-of-box usability of Koin allowed more time to be spent on the development of the application itself rather than on learning Hilt.

9.2.4 Beacons

During the final phases of the application development beacons were the last major part left to be implemented. Earlier in the development phase beacons caused some trouble where the SDK for Android would crash, and whenever the SDK would work you could not interact with the beacons unless they were not owned

by you basically rendering the SDK useless as one has to take ownership of a beacon to be able to configure it. Some quick research showed that the beacons provided by Shortcut were UWB beacons which were only supported by a handful of devices, of which neither member of the team had [23]. At the time because it was still early in the development phase the issue was left at that and requested Shortcut and the university (NTNU) to provide a compatible phone to use during the project period.

Coming back to this issue later during the project, Shortcut provided a Google Pixel 7 Pro to use during the project period, however, this did not resolve the issues regarding the SDK for the beacons. A multitude of SDKs was tested on the new device, from both Estimote and other providers, and sample code from the Android developer website [24] using the UWB Jetpack Library. All of these gave the same results as last time, the SDK would either crash or the beacons could not be configured because the SDK would not run unless the beacons were disowned. By looking into the matter more, on the Estimote forums an article regarding this very issue was addressed by a member of the Estimote team where it was stated that only the IOS SDK was functional at the time and that an Android SDK would come "soon" [25]. At that time this response was 30 days old and the safe assumption was that "soon" meant sometime this year. This came as a huge blow to the project as beacons were an integral part of reminding a user to take their medicine, and some measures had to be taken to alleviate this issue. To find the best possible solution this issue was brought up and thoroughly discussed together with Jason from Shortcut and Peter from NTNU. Together as a group came up with the following solutions:

1. Drop beacon support

The least likely and also least satisfactory was the idea of dropping the usage of beacons as a whole. This would lead to a huge portion of the project being dropped and thus leaving a huge hole in the application and its intended use, however, it would leave more time to properly polish the other areas of the application. Though this idea was brought up it was quickly dropped for other better ideas.

2. GPS localization

Another idea was to use GPS coordinates in order to localize the user and push notifications based on them arriving at positions labeled on the map. This solution would satisfy the idea of pushing notifications to the user whenever they arrive at a location near their medication such as at home or at work however, using GPS coordinates would have reduced precision in pinpointing which would not fulfill the task of the objective of this project which states "Add a location context to the medicine reminder that addresses availability in addition to prompt". Without accurate position localization, the app could not give a corresponding prompt for the medication because there would be no accurate availability data of the user.

3. Create IOS solution

The best idea and the idea that ended up being implemented was to create a beacon-only solution using Swift and SwiftUI for IOS. This would satisfy the project description by making use of the beacons to get the correct location context and would be able to communicate with the user's device as intended and described in the project description. The only downside with this idea was that this would not communicate with the rest of the application that was developed for Android. That means that the medication list and medication context would be missing from the IOS solution as there was no time to migrate the project from Android to IOS. Nevertheless, this would allow the Android application to be fully functional as a standalone in addition to us being able to showcase how to implement beacons in the existing application whenever Estimote releases their Android SDK.

4. Estimote Cloud Service

The last idea, which was also least explored due to it being more extreme was to make use of the Estimote Cloud Service to configure the beacons. Some forum threads hinted at the Cloud Service, run by Estimote, being more reliable for Android however, this idea had a lot of uncertainty and risk revolving because the Cloud Service that Estimote provides is locked behind a huge paywall and it is not certain whether or not it would work if paid for.

With all these solutions listed the students came to the conclusion that an IOS solution would be the best fit. Due to time constraints, the whole application will not be ported over to an IOS solution in Swift, however, with this solution the group can showcase the use of beacons and send push notifications to both phones and smaller smart devices such as smartwatches and satisfy the project description of integrating beacons as a reminder tool for the users.

9.2.5 Android API 23

When deciding the API level for the application's deployment the group had to take several factors into consideration.

Firstly, the minimum version for certain features is to be functional. This was contentious as there were several libraries and tools used during development that the group was able to find good solutions for together with Shortcut which removed the need for a higher API-level Android device. Secondly, was the target audience of the application. Since the application is primarily aimed at older audiences, the group had to adapt the application to be run on older Android devices as elderly people are not expected always to have a newer device.

Lastly, the development environment was also contentious. The reason is that older API levels would not offer the same features and security capabilities as newer API levels.

This lead the group to make a choice between API 22 - Lollipop 5.1 which is the version that would cover the vast majority of devices, at 99.3%, and API

23 - Marshmallow 6.0 which covers fewer devices at 98.2% [26] but includes the benefit of runtime permissions [27] which improves the security overall as the application must request permission from the user at runtime rather than at installation to access certain features such as storage and alarms which allows the user to choose whether to grant the application control or not.

With this in mind, the group chose to use API level 23 for this application because the added benefit of an additional layer of security outweighs the fact that it covers fewer devices.

9.2.6 Problem-solving through Pair Programming

Throughout the project period, the students worked mostly individually on their tasks and came together once or twice a week to resolve merge conflicts and review code together. Nevertheless, there were periods when issues arose and the students saw value in resolving them through pair programming.

Since both students had their own branches to work with their respective tasks there were times when code was able to compile on one's machine, but not on the others. This was systematically resolved through pair programming where both students could improve productivity and problem-solving by sharing knowledge between the driver; the code-writer, and the navigator; the reviewer.

One such instance was early during the development of the medication wizard page, as this page was mostly developed by just one student much of the design such as object placement, space distribution, and font sizes made the application look and behave unusually on different devices. This was accentuated whenever the students were presenting their progress during the weekly sprint review meetings. As issues arose the group became more aware and chose to work together through pair programming to resolve them together as a unit to increase the effectiveness and speed of the solutions.

Another instance was during the development of the settings page of the application, where the settings are saved locally on the user's phone for them to be persistent. The application would load said settings and save them each time the user boots up the application. Since this was by one of the students in the group, when this piece of code was merged into the main branch the code would no longer compile for the other student. This happened because the other student did not have any cached settings data on their device and thus the application fetched the settings values as null which caused the application to crash since this did not show up as a clear error message the issue was solved through pair programming where both students tried to reverse-engineer the issue by using breakpoints and looking at various values at different times at the application's runtime.

9.2.7 Teamwork

Since the end of the COVID-19 restrictions the group; both students, the supervisor Peter, and the employees at Shortcut agreed on a more direct approach for

the duration of the bachelor thesis. With this established early, the group had a healthy collaboration and a great communication channel which benefited greatly in creating a solid product.

Communication

As mentioned in the earlier section the group as a whole established a great communication channel during the early phases of the project.

The group established weekly sprint meetings in order to review and present their work and discuss the remainder of the work going forward. These were conducted in person on the NTNU campus while Shortcut joined online. These were conducted on Wednesdays at 13:30 however, due to the flexibility of the group both the time of day and day could be moved whenever the students had other school activities at that time or whenever the supervisor or the people at Shortcut could not attend due to meetings, exams or traveling. This proved to be a healthy relationship throughout the project period between all parts and helped the project flourish as a whole.

Additionally, the students worked together through a combination of both digital and in-person work. Most work meetings took place at the university's campus, but they would occur online once in a while due to students not being in Gjøvik at the time. Communication was mostly done through Discord and in person during meetings where the students could share ideas, resolve problems, and discuss topics about development.

Lastly, because the Shortcut office is in Oslo, they could not attend the weekly meetings physically and thus had to join online via Google Meetings. Due to this, the students took the liberty to take the trip down to Oslo a few times. Here the students were able to meet in person with Jason and Ram and receive help and guidance, but also a sparring partner regarding development. The students also had the opportunity to experience how the workdays are at Shortcut and also join in on extracurricular activities past the work hours such as GDG - Google Developer Groups where the students attended a meeting regarding Material3 Design in Jetpack Compose which served as a great initiative to migrate the application from using Material2 to using Material3.

Collaborative work

Throughout the duration of the bachelor project, the student group established a rigid work schedule during the work week. The aim was to be available for project-related work between 10:00 and 16:00 from Monday to Friday in order to mimic an environment that is similar to a normal work week, however, that did not exclude the possibility to be available before or after these hours even on the weekends. As mentioned earlier, since the COVID-19 restrictions were lifted for this semester the students opted for a more direct approach for this bachelor process, thus the preferred method of cooperative work and meetings was to be

in person, often on NTNUs campus. This, in tandem with the weekly sprint retrospective meeting and sprint review meeting, allowed the group to have a great overview of the workload at different stages of each sprint through a healthy communication channel between the students, the supervisor, and the employees at Shortcut. A few meeting minutes can be viewed in Appendix ?? to showcase the structure used for meetings throughout the project period.

9.2.8 Time allocation

Time allocation for the duration of the project period was done by utilizing labels for different activities from the project backlog. Though most of the time for this bachelor project was spent on development, a large part of the project in beacons was unable to be implemented for Android and in Kotlin and some time had to be spent acclimating to Swift and SwiftUI programming language which created a detriment in time distribution. For this bachelor project, the period of work was from the 9th of January 2023 til the 21st of May 2023 making the scope of the project to be exactly 20 weeks of work. The target workload was 30 hours per week, with a week of Easter holiday making the target workload per student 570 hours per student. Both students were well between the parameter given and this was well reflected in the product produced during the bachelor period with both students being satisfied with the time and effort invested in the project. The table below represents the time allocation of the students.

Table 9.1:

Meetings with supervisor	25 hours
Meetings with Shortcut	40 hours
Development work	531.5 hours
Thesis work	402.75 hours
Refactoring	85 hours
Swift & Beacons work	45 hours

Table 9.1: A table to showcase the areas and how much time was spent within those areas by the students.

A full preview of the workload distribution of each student can be found in Appendix A.8

Chapter 10

Conclusion

The conclusion of this report will focus on what was ultimately produced. This will be done by talking about what went well, and what the group was proud of. Then go over what went poorly, what could have been improved upon, or simply solutions the team ended up not being able to implement. Then there will be mentions of what the team sees as the future for the project, be it future developments, integrations, or exactly what the next step would be. In the end, there will a short section on the final thoughts from the group.

10.1 The good

This section will explain what the group felt went well throughout the projects. There will also be some examples of solutions that the group was especially happy with.

10.1.1 The results

Shortcut desired a proof of concept application, that could be used to make sure that the concept of the application was able to be implemented. The main purpose of the proof of concept would be to test the proprietary hardware's capability, in this case, the Ultra-wideband (UWB) beacons. This is something the team succeeded in, with some caveats. All the functionality laid out in the project description, seen in Appendix A.3, has been achieved with the project. The only difference is that the application had to be split into two pieces. Each of these pieces contains their respective parts of the requirements laid out in the project description or combined together. One of the lead Android developers at Shortcut, Jason Toms, even said "I am very satisfied with the functionality of the app. All of the major features planned were implemented, with understandable temporary solutions put in place due to third-party SDK issues out of your control."

10.1.2 Database

One area the team was very happy with the end result was the database solution of the application. This is because the group was able to make use of a new method of storing data locally. Previously the team had only stored data and persistence over an online database. It also meant that the group was able to make use of their knowledge of SQLite databases from a previous course to implement and understand the functions that Room ¹ makes use of. What the group was especially happy with were the converter functions and the relations between the various stored data values.

10.1.3 User experience

The team thought that the end result of the user experience (UX) for the application turned out quite well. The team was especially proud of the user experience of the wizard, and the evolution of the wizard throughout the project. The wizard was initially a nightmare to use, with the keyboard and the text boxes would shift about the screen at each and every step. The result was a much smoother and better UX.

10.1.4 Koin

The implementation of dependency injection in the application through the use of Koin ² was something the team was pleased with the result of. This is primarily due to neither member of the team having worked with Koin in the past, as well as dependency injection initially seeming like a fairly abstract problem. The solution in the code turned out simple, elegant, and served all the purposes that were required of it.

10.1.5 iOS solution

As outlined in section 9.2.4, the team had many issues getting the beacon library for Kotlin to work. This led to the team eventually coming to a solution the team was happy with. This was implementing a part of the project for iOS using Swift. The reason the team was pleased with this solution as the group created something out of nothing and did not just give up. The group discovered a fundamental problem with the project, and instead of just throwing in the towel, the group persevered with the more difficult of the solutions available.

One of the reasons the team was as pleased with being able to use the iOS beacon solution was due to the difficulty involved with implementing it. Neither student in the group had much any knowledge or experience with programming in Swift. Therefore it was even better when the end product not only was able to be finished in Swift, but was also able to make use of SwiftUI, a more modern

¹Room is the persistence library that was used in the project, see 4.4.

²Dependency Injection (DI) library for Android

framework for Swift programming. The main reason the group was as happy with the solution is due to the short period of time from the problem being discovered, to the discussion, to start working on a solution and finally presenting a solution.

10.1.6 Teamwork

The team feels that throughout the project the teamwork has been quite excellent. There were quite a few meetings where the group would meet up to work with each other, explaining issues, or figuring out solutions for problems. There was never a breakdown in communication between the members.

An element that greatly helped team cohesion and morale was doing extracurricular activities together, such as traveling to and from Oslo to go to the Shortcut offices.

10.2 The bad

This section will go over the shortcoming that the team felt resulted from the project. The section will also go over why these shortcomings happened. The chapter will also mention difficulties encountered during the entire project.

10.2.1 Beacon troubles

The topic has been reiterated several times during the rapport, but towards the latter end of the project, the group hit a large problem with the project regarding the SDK for the UWB beacons. The solution is gone over in detail in section 9.2.4. While the problem was solved, and the group was happy with the eventual solution, there are still elements of the solution which the team was unhappy with.

The main issue that the team was unhappy with was the simple fact that the team was unable to implement the beacon element of the Android application. The team was disappointed that the Android project essentially ended with a big hole in functionality. It did however lead to a lot of learning and knowledge. This is that when a new technology, library, or SDK is taken into use, it needs to be made sure early on that the element actually functions. This is especially important if the element is the core of the application being created. The group should have made a proof of concept first with the beacons to make sure they worked, then move on to the actual application development. This is something the team learned is an inherent risk that comes with relying on third-party elements. So while the group is not pleased with the problem, the group is happy with the knowledge and experience that came from solving it.

10.2.2 Design of the application

While established early on as a non-important element of the application, the group is still not pleased with the appearance of the application. Even though the group used knowledge gained from the IDG1362 course "Introduction to user-centered design" to improve the user experience of the application, the knowledge for making the application look better and prettier was not put into proper use. Thus, while elements such as usability were put into focus, the focus of the group was not to make the application properly good-looking.

For this to have been deemed a more successful part of the project, a heavier focus on design would have to have been used. This could possibly have been done through the use of resources from Shortcut. This would have been done by contacting the design team at Shortcut.

10.2.3 Lacking desired functionality

Early on in the project, one of the functionalities that were brought up was for optical character recognition (OCR) to be implemented in the application. This would be for being able to automatically read the labels on medication so that the user would not have to manually input values themselves. This did not end up being implemented into the project due to the complexity it would bring. This complexity would come not only from the implementation itself of coding an OCR, but also from interpreting the text that would come from it.

10.2.4 Notifications

One area where the team was unhappy with the result was the notification system of the application. The functionality of the notifications was less than desired. This meant that the original intent was for the application to be a yes/no option, rather than just a notification. Compounding on this the notifications in testing were routinely later than the intended timing, often not going off until forty-five seconds later than desired. This is something that was not looked closer at due to time constraints.

10.2.5 Testing coverage

Early on in the project one aspect the group wanted to accomplish well was the use of test-driven development. This is the idea of making tests and making the code for the tests to work. This idea was not followed through with for more than the initial stage of the project, with testing of the code in general falling by the wayside. This is for the team an incredible oversight, which should have been rectified much earlier.

10.3 The future

The end of this bachelor project does not necessarily mean the end of this project. The project ended in a spot that could lead to the potential for future development. The potential futures for this project will be explained below.

10.3.1 Optical character reading

As mentioned earlier in the previous section, the OCR was an element of the application that the team wanted to implement. If the application was to be worked on further, this would be one of the main goals to implement. This could be done in several different ways. There are paid SDKs that exist such as Scanbot³ that could fit this solution. However using knowledge from the previously mentioned PROG2051 course, the team could also attempt to develop their own OCR, though this option is unlikely due to the complexity it would require. There was also discovered that both Google [28] and Apple [29] had developed their own OCR SDKs through the use of machine learning, however, this was discovered too late in the project to be able to implement. This would be the most likely avenue for implementing OCR for the application in the future.

10.3.2 Future app integration

This team was not the only team working with Shortcut for their bachelor project. There were also teams from other universities there. During an excursion to the Shortcut offices, it was brought up that this project's application could fit into an already existing medical application. This could be one of the major ways this project would continue its future. If this option is taken, then the next step for this application would be the refinement of the functionality. This would then likely be followed by a design overhaul to align the design of the application with the existing solution. The final step would then be to integrate the applications together, making the future of this application exciting future as it would make it ready to be released and used in the real world, which is one of the reasons the team is studying for this bachelor.

10.3.3 iOS version

Considering the project ended up with some functionality already implemented in iOS, as well as the initial project description making space for an iOS version of the application, a future iOS version of the application is a very probable option. This version would mimic the Android application as closely as possible in design, so as to be able to reuse decisions made during the course of this project. An alternative solution to this would be to instead of creating an independent iOS solution, create a cross-platform solution. This would be done through the use

³<https://scanbot.io/products/barcode-software/barcode-sdk/>

of frameworks like Flutter⁴ or Kotlin Multiplatform Mobile⁵. The more likely of these two options would be independent solutions per platform due to requiring access to components like UWBManagers which may not be accessible to non-native applications.

10.4 Final conclusion

The team feels that the objectives that were made have been achieved, and the team is happy with the end result. The team is happy about the nature of the finished project. The project is something that could bring true value to society by being able to help people's health and the well-being of the users. As mentioned earlier, Shortcut has already come forward with possibilities for the future of the project through integration into existing products. We would like to thank Shortcut one final time for letting us work with them on this project and for all the help they provided.

⁴A cross-platform framework for iOS and Android developed by Google

⁵A cross-platform framework by JetBrains for Android and iOS

Bibliography

- [1] B. Fogg, *Tiny Habits* (Edition unavailable). Harvest, 2019, ISBN: 9780358003991.
- [2] Nictiz. 'Ce marking.' (), [Online]. Available: https://www.nweurope.eu/media/3201/ce-marking_infographic-medical-apps.pdf. (accessed: 01.25.2023).
- [3] C. of Practice for Computer Science Education at the Norwegian University of Science and Technology. 'Thesis-ntnu.' (), [Online]. Available: <https://github.com/COPCSE-NTNU/thesis-NTNU>. (accessed: 13.04.23).
- [4] 'Use cases.' (), [Online]. Available: <https://www.usability.gov/how-to-and-tools/methods/use-cases.html>. (accessed: 12.05.23).
- [5] J. Pleško. 'A guide to beacon technology in 2021.' (), [Online]. Available: <https://infinum.com/blog/bluetooth-beacons/>. (accessed: 12.05.23).
- [6] Bleesk. 'What is uwb?' (), [Online]. Available: <https://bleesk.com/uwb.html>. (accessed: 26.04.2023).
- [7] A. Inc. 'Air tag lose your knack for losing things.' (), [Online]. Available: <https://www.apple.com/airtag/>. (accessed: 26.04.2023).
- [8] M. Li. 'What is eddystone?' (), [Online]. Available: <https://www.mokoblue.com/all-about-eddystone-beacon/>. (accessed: 26.04.2023).
- [9] S. Segan. 'The best android phones for 2023.' (), [Online]. Available: <https://uk.pcmag.com/smartphones/41662/the-best-android-phones>. (accessed: 01.05.2023).
- [10] 'Phone os marketshare worldwide.' (), [Online]. Available: <https://gs.statcounter.com/os-market-share/mobile/worldwide>. (accessed: 01.05.2023).
- [11] 'Phone os marketshare in norway.' (), [Online]. Available: <https://gs.statcounter.com/os-market-share/mobile/norway/>. (accessed: 01.05.2023).
- [12] 'What exactly is kotlin?' (), [Online]. Available: <https://codeop.tech/what-exactly-is-kotlin/>. (accessed: 02.05.2023).
- [13] 'The good and the bad of swift programming language.' (), [Online]. Available: <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-swift-programming-language/>. (accessed: 03.05.2023).
- [14] 'Computer program.' (), [Online]. Available: <https://www.britannica.com/technology/computer-program>. (accessed: 19.05.2023).

- [15] Thorben. 'Design patterns explained – dependency injection with code examples.' (), [Online]. Available: <https://stackify.com/dependency-injection/>. (accessed: 03.05.2023).
- [16] (), [Online]. Available: <https://medium.com/decisionbrain/dates-time-in-modern-java-4ed9d5848a3e>. (accessed: 12.05.23).
- [17] F. Muntenescu. 'Database relations with room.' (), [Online]. Available: <https://medium.com/androiddevelopers/database-relations-with-room-544ab95e4542>. (accessed: 04.05.2023).
- [18] 'Material design.' (), [Online]. Available: <https://m3.material.io>. (accessed: 07.05.2023).
- [19] 'What is ci/cd?' (), [Online]. Available: <https://www.redhat.com/en/topics/devops/what-is-ci-cd>. (accessed: 06.05.2023).
- [20] (), [Online]. Available: <https://github.com/detekt/detekt>. (accessed: 06.05.2023).
- [21] AndroidDev. 'Why adopt compose.' (), [Online]. Available: <https://developer.android.com/jetpack/compose/why-adopt>. (accessed: 05.03.2023).
- [22] KotzillaDev. 'Koin - kotlin dependency injection.' (), [Online]. Available: <https://www.kotzilla.io/koin/>. (accessed: 06.03.2023).
- [23] 'List of uwb-enabled mobile devices.' (), [Online]. Available: https://en.wikipedia.org/wiki/List_of_UWB-enabled_mobile_devices. (accessed: 08.02.2023).
- [24] AndroidDev. 'Ultra-wideband (uwb) communication.' (), [Online]. Available: <https://developer.android.com/guide/topics/connectivity/uwb>. (accessed: 20.03.2023).
- [25] J. -Leader-. 'Estimote app errors and cannot find any beacons.' (), [Online]. Available: <https://forums.estimote.com/t/estimote-app-errors-and-cannot-find-any-beacons/11929/3>. (accessed: 20.03.2023).
- [26] 'Android api levels.' (), [Online]. Available: <https://apilevels.com/>. (accessed: 12.03.2023).
- [27] 'Request runtime permissions.' (), [Online]. Available: <https://developer.android.com/training/permissions/requesting>. (accessed: 14.03.2023).
- [28] 'Recognize text in images with ml kit on android.' (), [Online]. Available: <https://developers.google.com/ml-kit/vision/text-recognition/v2/android>. (accessed: 18.05.2023).
- [29] (), [Online]. Available: https://developer.apple.com/documentation/vision/recognizing_text_in_images. (accessed: 18.05.2023).

Appendix A

Additional Material

A.1 Initial Project plan

Project Plan Medicine Reminder System

Alin Mihai Napirca
Thomas Øvstun Ditman

January 2023

Contents

1	Goals and framework	3
1.1	Background	3
1.2	Project Goals	3
1.2.1	Result Goals	4
1.2.2	Effect Goals	4
1.2.3	Learning Goals	4
1.3	Constraintments	4
2	Scope	5
2.1	Subject area	5
2.2	Scoping	5
2.3	Description	5
3	Organization	6
3.1	Roles	6
3.1.1	Shortcut	6
3.1.2	Supervisor	6
3.1.3	Group Roles	6
3.2	Routines and rules	6
4	Planning	7
4.1	Parts of the project	7
4.2	Plan for Usage of Model	7
4.3	Plan for status meetings	8
5	Quality Assurance	8
5.1	Plan for quality standard	8
5.2	Documentation, standards, configurations, tools, etc	9
5.3	Plan for inspection and testing	9
5.4	Risk analysis on project level	10
6	Plan for procedure	11
6.1	Gantt diagram	11
6.2	Milestones	12

1 Goals and framework

1.1 Background

Shortcut is a software development company that specializes in creating Android and IOS applications. Shortcut is based in Scandinavia, with offices in Copenhagen in Denmark, Stockholm in Sweden, and Bergen and Oslo in Norway. Shortcut¹ is one of the leading app developers in Scandinavia, having worked on such applications as:

- Æ: Food and Grocery app for REMA 1000
- Ruter: Bus and Transportation
- BankID: Finance and Security
- Vipps: Financial and ease of payment
- SBanken: Finance and Banking

Medicine is something most people will have to take at regular intervals at some point in their lifetime. This entails taking a specific dosage at a specific time of day, sometimes multiple times a day. One of the main problems with taking your medicine is remembering to take it or remembering if you have already taken it.

To help with this there are already applications and other systems to help mitigate this problem. These typically send you a notification to your personal smartphone at the appropriate times of the day. This isn't the cure-all that we are hoping for. This is because getting a notification to take your medicine does not mean you will take it, or are even able to take it.

For this project Shortcut proposes us to develop a proof of concept for an Android application and system to help with this problem. This will be done through the use of an Android application as well as Ultra Wide Band beacons, -hereby referred to as UWB Beacons- and other smaller smart devices such as smartwatches.

The main difference or "selling point" of our project is the ability to give reminders to you based on your location. The existing products are only capable of reminding you to take your medicine without taking it into consideration if you are in a situation capable of taking your medications. Our project will take location into consideration to make sure you are able to take your medication when reminded to.

1.2 Project Goals

We have chosen to split the project's goals into a few categories. We have the result goals, which is the project's goal defined by Shortcut. Effect goals are what we wish to achieve through the use of the project application. Finally, we have learning goals. These entail what we wish to learn throughout the completion of the project

¹You can read more about Shortcut and apps they have developed following this link: <https://shortcut.io/norway/>

1.2.1 Result Goals

The main goal as established by the Shortcut is to create a proof of concept medicine reminder system. The system will be created through the use of contextual programming for prompts. This will be done using an Android application, that can communicate with a smartwatch, in conjunction with the use of UWB Beacons for close distances.

1.2.2 Effect Goals

One of the main problems with remembering to take medicine regularly is the formation of habits. As per the literature of habit formation[2] the three ingredients for a successful habit are:

- Motivation: How motivated is the user to take the medication?
- Availability: Is the medication available when the user is to take the medication?
- Prompt: When/how is the user reminded to take the medication?
- Reduction: Reduce the likelihood to take too much medicine or not enough medicine.

Our focus will be on the "Availability" and the "Prompt" aspects.

1.2.3 Learning Goals

Through the completion of this project one of the most important aspects for us as a group is to learn through doing. Because of this our main focus on learning will be:

- Use of SCRUM methodology[7] to organize work
- Usage of technologies tied to an Internet of Things line of thinking
- Working with real people who have real requirements for our end product
- Proper use of the CI/CD pipeline for quality control

1.3 Constraintments

There are a few limitations we have to establish for the project. Due to the use of Beacon technology, we have to use at least Android API 18. With that being said, the scope of the bachelor thesis will be an application limited to only Android devices, however, if time permits we will extend and create an application for iOS devices as well.

Due to the application possibly falling under the definition of a medical device, we might have to follow some requirements set by "CE Marking"[6]

Due to the nature of the project, we will need to finish the project by the 22nd of May. This deadline will include both the application itself and the report for the project

2 Scope

2.1 Subject area

Medication is an important aspect of modern life. Throughout the lifespan of an individual, they are likely to be prescribed medication that is to be taken at various points throughout the day. These medications can be for life-threatening conditions, with missing medication being dangerous.

An example of this could be anti-coagulants², which is a type of heart medicine which decreases the clotting ability of blood. This is an important medicine to help prevent strokes in individuals. Another large aspect is taking too much medication, which could be as dangerous or even more dangerous than forgetting to take your medication.

The application we are developing will therefore be a way of tracking and reminding you to take your medication. The focus will not be on prescribing or recommending any kind of medication, only reminding the user of medications they will already be taking.

2.2 Scoping

The two biggest mobile platforms for applications in Norway are Android and IOS. For this project, we will scope only to include Android applications. This is due to prior knowledge of the Kotlin language. If we find ourselves with a large amount of time after having worked on the Android application we want to keep open the possibility of scoping out to include an IOS alternative as well.

2.3 Description

Our project is for the creation of a proof of concept application for Shortcut. This will be a mobile application. There will also be functionality for wearable technologies such as smartwatches. The application should have the following functionality:

- User should be able to set reminders for taking medication.
- Make use of location context in the form of UWB beacons to make sure the user is in the correct location for taking the medicine.
- Display a visible indication of when the medicine is to be taken.
- Display medicine that has been taken.
- Smartwatch integration for notifications.

The application is inclined toward anyone who takes medicine regularly, however, we identified two groups as our targets. Firstly, the users who just want a structured system whenever they have to take their medicine. This group of people does not struggle with their memory particularly and just wants a great system to remind them to take their medicine and to register whenever they have similar to a pillbox (when the box for the respective day is empty the patient can be sure they already have taken their medicine). The other group are people who often cannot recall taking their medicine, like elderly people and people who have problems with memory or concentration such as patients diagnosed with ADHD [4].

²Anti coagulants are a type of heart medication, typically referred to as blood thinners: <https://www.nhs.uk/conditions/anticoagulants/>

3 Organization

3.1 Roles

3.1.1 Shortcut

Shortcut AS is our employer and our contact persons are Ram Kumar and Jason Toms. We will be holding Daily Scrum through Slack with them and also have a weekly meeting to update them on our progress, problems, and struggles if any.

3.1.2 Supervisor

For this bachelor's thesis, our supervisor will be Peter Nussbaum. We have established to meet each week on Wednesdays in the beginning and shift to a bi-weekly meeting once we have kicked the project in gear.

3.1.3 Group Roles

Thomas Ditman will act as the group leader. He will be in charge of making sure the project is progressing and will be the main contact person with Shortcut. He will keep track of meetings, schedule, and time management.

Alin Mihai Napirca will be the group's Scrum Master. His responsibilities will be to lead the Scrum meetings; Sprint Planning, Daily Scrum, Sprint Retrospective, and Sprint Review. Additionally, he will be maintaining the issue backlog and making sure the group follows the agile framework of application development.

Both group members will also be developers. This means that as a group they are responsible for taking on and completing issues in the product backlog for each sprint.

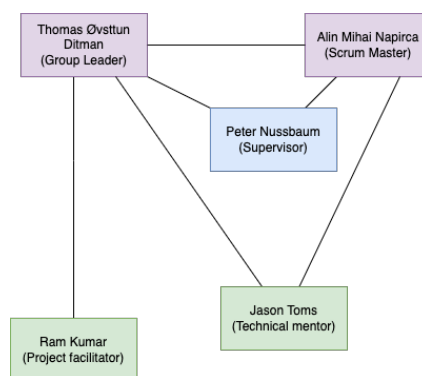


Figure 1: The different people/organizations part of our bachelor thesis. The students are illustrated in red, the NTNU supervisor is illustrated in blue and the contact persons from Shortcut are illustrated in green.

3.2 Routines and rules

- Members are expected to attend a bi-weekly meeting with the supervisor and Shortcut on Wednesdays at 13.30

- Members are expected to attend Daily Scrum meetings with Shortcut via. Slack on a daily/weekly basis.
- Members are expected to work roughly 30 hours each week during the duration of the bachelor thesis.
- Members are expected to notify in case they cannot attend a meeting or fulfill the required hours worked.
- Members are expected to be available between 10.00 - 16.00 from Monday to Friday.
- All formal meetings should be documented and summarized.
- If members constantly are unable to comply with the rules, the remainder of the group will contact the supervisor to determine the course of action. If all else fails, the course supervisor should be notified in advance to find solutions.

4 Planning

4.1 Parts of the project

Since the assignment we have been given by Shortcut is very open on how to perform the actual project, we have decided to go with SCRUM methodology for organizing our project. One of the main reasons we chose SCRUM[7] as our methodology is familiarity. Several of our previous projects were driven using SCRUM, and we thought it was quite effective.

In tandem with SCRUM, we will be using JIRA[3] to organize our sprints and tasks within them. This will give us a good overview of what needs to be done and who is doing what. There will also be a daily scrum, consisting of updating the customer of what will be accomplished on that day, with stand-ups being organized if it is felt necessary.

We have chosen to keep our sprints to two-week periods of time, with sprints starting on a Wednesday, and ending two weeks later on a Tuesday. This will give good flexibility to communicate with the customer for potential changes to the project while being a long enough stretch of time to properly work into a task.

We are planning to use sprint planning meetings and sprint reviews to confirm what tasks need to be done, and what has been accomplished in the previous sprint respectively. A two-week period also fits well into our timespan for meetings with Shortcut. We will set up joint meetings with Shortcut and the Supervisor on Wednesday every week.

4.2 Plan for Usage of Model

During the sprint planning meeting, tasks will be distributed between the group members, but they can also be assigned during sprint stand-up sessions with Shortcut or our supervisor. Each group member has the responsibility to manage their assigned tasks and make sure that each task is completed within the

designated time frame.

When a group member begins working on a task, they will first change the task's status from "TO DO" to "IN PROGRESS" (Figure 4.2). If they encounter any obstacles that prevent them from progressing, they can change their status to "HALTED", tasks in this state are frozen indefinitely till the requirements to continue the task are met. Before a task can be marked as complete, it must go through a review phase which will be done manually by both group members to come to a conclusion together.

Sprint reviews will take place during each meeting with the client, which allows for timely feedback from product owners. A sprint retrospective meeting will take place during the daily stand-up meeting on Wednesdays at the start of a new sprint. During this meeting, team members will reflect on questions such as: "What went well?", "What could be improved?", and "What were this sprint's struggles?"

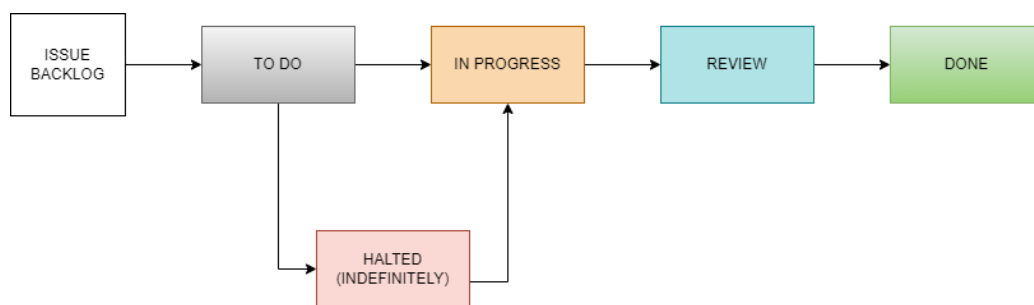


Figure 1: The workflow the students will follow in the scrum agile workflow.

4.3 Plan for status meetings

As mentioned in section 3.2 **Routines and Rules** we will conduct weekly a meeting with our supervisor from NTNU. During this meeting, we will update him on our progress during the previous week as well as ask for feedback and questions for the part of the project we are working on. The supervisor will primarily help with the documentation and planning side of the project.

We have also set up to have a recurring meeting with a contact from Shortcut. This will primarily be to communicate how we are faring in our progress for the project. This will also be an opportunity to gather help in the programming aspects of our project since we will be able to directly ask experts in Android development for help if we are stuck on an issue.

5 Quality Assurance

5.1 Plan for quality standard

Our main method for quality assurance will be through the use of both manual and automatic tools. The manual tools will be through the use of extensive peer review of tasks that have been accomplished. Any code that is to be pushed to the main branch of the project will have to be checked by the other member

of the team.

Additionally, we will use automatic tools through the use of the CI/CD³ pipeline available in GitLab. Through this pipeline we will run various tools, such as Detekt, to check our code for potential issues and "smelly" code.⁴ We will also use automatic testing on our code when attempting to merge into the main branch of the project. This will primarily be done through the JUnit testing for Android.

5.2 Documentation, standards, configurations, tools, etc

Table 1: Technology overview

Name	Type	Used for
Openleaf	Tool for editing LaTeX documents	Project report
Jira	Digital scrum and issue board	Project organization
GitLab	Code repository and version control	Code control
Android Studio	IDE for Android development	Code production
Draw.io	Diagram Production	Design and visualization
Discord	Communication between team members	Communication
Google Meet	Communication with Shortcut	Communication
Slack	Communication of short questions for Shortcut	Communications
Estimote	UWB Beacon technology	Beacon

In regards to the beacon technology specifically, there are a large number of potential options to choose from. We will be using the Estimote[1] beacon technology in this project. Estimote is one of the beacon technologies with the longest battery life in the world of beacons, with a life of 21.4 months[5]. Though the main reason we are using the Estimote technology is that we were provided a developer kit by Shortcut.

5.3 Plan for inspection and testing

For this project, we want to put a good focus on testing our solutions as we produce them. To further this goal we will early on establish a CI/CD pipeline for our git repository. The goal of this pipeline is to make sure that the code we have written will accomplish the intended goal. This means following a sort of "test-driven development" to a degree. Another aspect of the pipeline will also be a "lint" check, to make sure that the code not only works but is avoiding any common pitfalls and "smelly" code.

Outside of this automatic testing, we will also follow a thorough peer review process, where code that is written will have to be checked by the other member of the team before it can be pushed to the main branch. This will make sure that both members of the team are aware of how the code works and why it is written as it is.

³Continuous integration, continuous delivery, see: <https://en.wikipedia.org/wiki/CI/CD>

⁴"Smelly" code is code that has an increased risk of creating bugs or does not follow general code conventions. see: <https://refactoring.guru/refactoring/smells>

5.4 Risk analysis on project level

In the following section, we provide a list of possible risks that would be damaging to our bachelor thesis. We categorize every risk with a number and give the risk a frequency and consequence in regard to our project and plot it into table 2. The number is not a threat level, but an ID to categorize each threat. The spectrum of color goes from bright green where risks are non-considerable in threat and/or happen rarely to bright red where risks are considered catastrophic to our project and/or happen regularly/are guaranteed to happen.

Table 2: Initial risk matrix

Frequency/ Consequence	Rare	Unlikely	Probable	Likely	Certain
Catastrophic		1			
Critical	2	3, 5	4, 6		
Major					
Minor					
Non-considerable					

In table 3, we describe what risk each number identifies. We also keep the color code to indicate how dangerous each risk is.

Table 3: Risks analyzed

Risk	Description	Frequency/Consequence
1	Loss of group member due to various reasons (Sickness, family emergency, etc.)	Unlikely/Catastrophic
2	Not meeting deadline(-s)	Rare/Critical
3	Loss of data in form of documentation and/or source code	Unlikely/Critical
4	Insufficient tests/lackluster tests	Probable/Critical
5	Applications does not satisfy the customers expectations	Unlikely/Critical
6	Application does not qualify for the CE marking qualifications	Probable/Critical

This last table, table 4, showcases our plan to mitigate the different risks in order to reduce their likelihood to occur, but also how damaging they will be to our project if they happen.

Table 4: Risk mitigation plan

Risk	Severity	Risk Mitigation
1	Medium	Due to this being a two-person project this could be quite damaging and have grave consequences. If this is to happen, the project will carry on, however probably be evaluated and scaled down to a narrower scope than initially proposed with Shortcut. Everything will be well documented with tests and comments so that the work can easily be continued by other colleagues.
2	Low	Having clear goals for each sprint and understanding the workflow of a Scrum framework where tasks can be re-evaluated and revisited at a later time. The project should have clear milestones so that if any deadlines are missed there is still something to show.
3	Low	Having a GitLab repository for version control with frequent commits where we can go back and revisit the old versions in case something goes wrong. Additionally, we will be using Overleaf which has a co-write feature so both of us have access to all documents at any time. In case everything else fails, also make frequent pulls from the Git repository to also store files locally.
4	Medium	In case of scarce user tests we might have to result to other people's test data and academic papers in order to grasp the needs and frustrations of possible users, from both patients who take/have taken medicine regularly and users who have not. This might lead to outdated data and possible flaws and is something we will heavily prioritize in order to succeed in our CI/CD workflow.
5	Low	Having regular meetings with Shortcut which includes Daily Scrum and also weekly Sprint Retrospective meetings to show our progress and get feedback on our work. Additionally, use slack as a tool for guidance in case a meeting on short notice is not possible
6	Medium	Establish a good communication channel and work closely with Shortcuts HQ in Denmark which can provide guidance for possible pitfalls and mitigation we need to address, and also do in-depth research on the subject on our own.

6 Plan for procedure

6.1 Gantt diagram

This is the Gantt diagram (Figure 2) that illustrates our plan for this bachelor thesis. The whole period is split into seven sprints, where Sprint 1 is the longest as it is the initial sprint to kick-start the project whereas the other six are more even in length to simulate the scrum workflow.

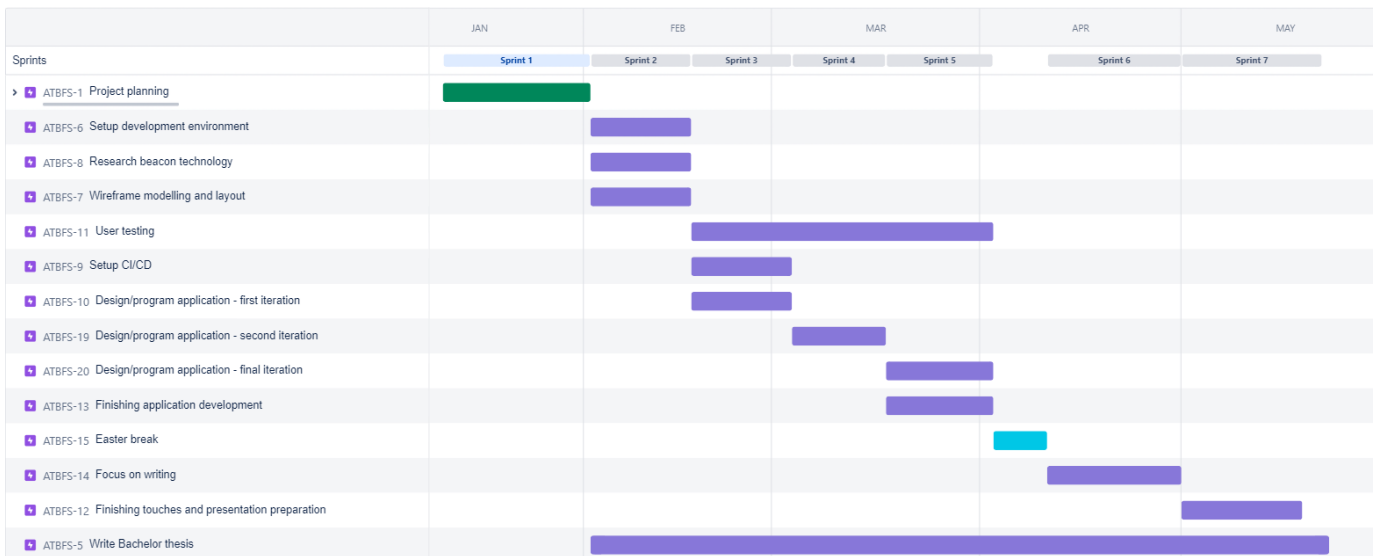


Figure 2: Gantt diagram showcasing the different sprints and the goals for each sprint.

6.2 Milestones

1. 01.11.23 "Kick-start" Bachelor thesis
2. 01.02.23 Finish sprint 1 - Project planning phase
3. 02.16.23 Finish sprint 2
4. 02.17.23 Start sprint - Begin emphasis on user tests
5. 03.03.23 Finish sprint 3
6. 03.17.23 Finish sprint 4
7. 04.02.23 Finish sprint 5 - Most coding should be done by now
8. 04.03.23 - 04.10.23 - Easter break
9. 04.11.23 - Focus on writing bachelor thesis
10. 04.30.23 - Finish sprint 6
11. 05.18.23 - Finish sprint 7
12. 05.22.23 - Deliver bachelor thesis
13. 05.23.23 -> Focus on preparation for presentation

References

- [1] *Estimote Beacon*. URL: <https://estimote.com/uwb-beacons>. (accessed: 01.25.2023).
- [2] BJ Fogg. *Tiny Habits*. Edition unavailable. Harvest, 2019. ISBN: 9780358003991.
- [3] *Jira*. URL: <https://www.atlassian.com/software/jira>. (accessed: 01.25.2023).
- [4] NHS. *Attention deficit hyperactivity disorder (ADHD)*. URL: <https://www.nhs.uk/conditions/attention-deficit-hyperactivity-disorder-adhd/>. (accessed: 01.25.2023).
- [5] Nick. *Compare Beacon*. URL: <https://www.aislelabs.com/reports/beacon-guide/>. (accessed: 01.25.2023).
- [6] Nictiz. *CE Marking*. URL: https://www.nweurope.eu/media/3201/ce-marking_infographic-medical-apps.pdf. (accessed: 01.25.2023).
- [7] Ken Schwaber. *What is Scrum?* URL: <https://www.scrum.org/resources/what-is-scrum>. (accessed: 01.25.2023).

A.2 NTNU Project Agreement

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett

Er den rett som den som skaper et åndsverk har til å fremstille eksemplarer av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater

Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater

Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn

Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring

Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt:
Veileder ved NTNU: PETER NÆSSBAUM. e-post og tlf.
Ekstern virksomhet: SHORTCUT AS Ekstern virksomhet sin kontaktperson, e-post og tlf.: Jason TOMS jason.toms@shortcut.no
Student: THOMAS Ø.D. Fødselsdato: 05.10.99
Ev. flere studenter ¹ Alin Mihai Popica 20.01.2023

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	<input type="checkbox"/>
Bacheloroppgave	<input checked="" type="checkbox"/>
Prosjektoppgave	<input type="checkbox"/>
Annen oppgave	<input type="checkbox"/>

Startdato:
Sluttdato:

Oppgavens arbeidstittel er:

¹ Dersom flere studenter skriver oppgave i fellesskap, kan alle føres opp her. Rettigheter ligger da i fellesskap mellom studentene. Dersom ekstern virksomhet i stedet ønsker at det skal inngås egen avtale med hver enkelt student, gjøres dette.

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven². Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten

² Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

<input checked="" type="checkbox"/>	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven
-------------------------------------	--

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

<input type="checkbox"/>	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt
--------------------------	---

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

<input type="checkbox"/>	Oppgaven skal være offentlig
--------------------------	------------------------------

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Opgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss	Sett dato
<input type="checkbox"/>	ett år
<input type="checkbox"/>	to år
<input type="checkbox"/>	tre år

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder: Dato:
Veileder ved NTNU: Dato:
Ekstern virksomhet: ØYVIND STØLEN, COO Dato: 30.01.23
Student: THOMAS ØD. Dato: Jan. 20. 2023
Ev. flere studenter Aren Mihai Nepricea 20.01.2023


Fastsatt av prorektor for utdanning 10.12.2020

STANDARDMAL ved avtale om konfidensialitet mellom student og ekstern virksomhet i forbindelse med studentens utførelse av oppgave (master-, bachelor- eller annen oppgave) i samarbeid med ekstern virksomhet, jf. punkt 9 i standardavtale om utføring av oppgave i samarbeid med ekstern virksomhet.

Student ved NTNU: THOMAS ØD
Fødselsdato: 05.10.99
Hvis flere studenter: Alen Mihai Napruca # 08.00
Ekstern virksomhet: SHORTCUT AS

1. Studenten skal utføre oppgave i samarbeid med ekstern virksomhet som ledd i sitt studium ved NTNU.
2. Studenten forplikter seg til å bevare taushet om det han/hun får vite om tekniske innretninger og fremgangsmåter samt drifts- og forretningsforhold som det vil være av konkurransemessig betydning å hemmeligholde for den eksterne virksomheten. Det er den eksterne sitt ansvar å sørge for å synliggjøre og tydeliggjøre hvilken informasjon dette omfatter.
3. Studenten er forpliktet til å bevare taushet om dette i 5 år regnet fra sluttdato.
4. Kravet om konfidensialitet gjelder ikke informasjon som:
 - a) var allment tilgjengelig da den ble mottatt
 - b) ble mottatt lovlig fra tredjeperson uten avtale om taushetsplikt
 - c) ble utviklet av studenten uavhengig av mottatt informasjon
 - d) partene er forpliktet til å gi opplysninger om i samsvar med lov eller forskrift eller etter pålegg fra offentlig myndighet.

Signaturer

Student: THOMAS ØD
Dato: 01.12.23
Hvis flere studenter: Alen Mihai Napruca
Ekstern virksomhet: ØRVIND STØLEN, COO
Dato: 30.12.23 

A.3 Project Proposal

Prosjekt: Medisin påminnelses system / Medicine Reminder System - Proof of Concept.

Arbeidsgiver : Shortcut AS, REBEL, Universitetsgata 2, 0164 Oslo

Kontakt : Ram Kumar, +47 911 90 661, ram.kumar@shortcut.no

Motivation: A context based smart reminder system for medicine intake.

A large section of the population take regular medication or supplements through the day. It is common to forget to take the medication at the correct time. Forgetting to take the medication can adversely affect the person - which sometimes can be deadly for patients with life-threatening conditions and are reliant on the medication. It can also have substantial negative consequences if the patient takes an incorrect dosage.

Traditional methods have relied on reminders on watches, phones and other *reminder* methods to prompt the user to take medication based on time of day and/or medication interval (every x hours). Traditional methods also *tracks* the medication dosage based on manual input. A combination of online pharmacy + end user device (pill bottle, push notification on app and so on) which needs extensive buy-in and is not available to all most users. Most of these are geographically related.

The traditional systems use *prompts* to remind the user and rely on the *motivation* of the user to accomplish the medicine intake. This might always not be the most effective strategy. The medical reminders on most of the solutions triggers a *prompt* (alarm, push notification) that does not consider the context the user is in - whether the user *can* actually take the medication (*availability*). Most of the existing solutions does not adjust the their *prompt* s based on the user context (when the medication is actually *available*).

Objective

With the proliferation of wearables among the target groups (Smart watches, fitness trackers, ...) in addition to a connected Mobile phone, the eco-system is ripe for a smarter reminder system that takes into account other context than just time of day/interval. We can add additional contexts such as **location** to the reminder system to increase the likelihood of a successful dosage intake.

As per [literature on habit formation](#), a successful habit needs 3 ingredients to be happen:

- *motivation* : How motivated the user is to take the medication?
- *availability*: Is the medication available when the user is to take the medication?
- *prompt*: When/how is the user reminded to take the medication?

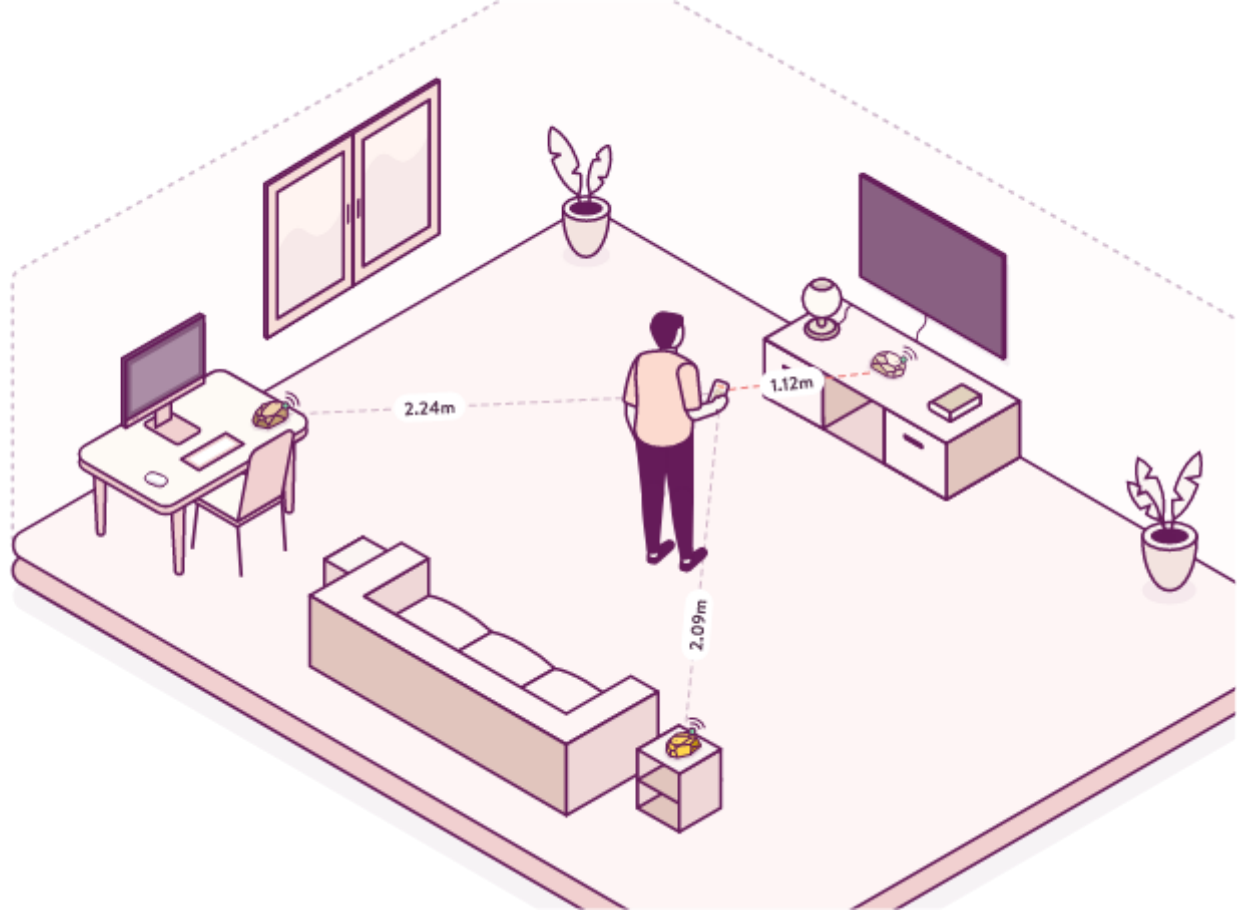
For a successful behaviour to accomplish, focus should be on *availability* and contextual *prompt*; not solely on the *motivation* of the user.

Tasks

- Create a proof-of-concept medicine reminder **Android App** that prompts the user to take an appropriate medication dose at a given time.
- Add a **location** context to the medicine reminder that addresses *availability* in addition to *prompt*
- If scope permits, document the **CE marking** evaluation for the app - what are the pitfalls that we, as a developer of a medical app, need to be aware of? are there mitigations we can build into the app to address these?

Technology

- With location and distance (UWB, Bluetooth) sensor technologies being increasingly available on the Mobile devices, location awareness can be built into apps to figure out the distance to medicine location (cabinets at home, shelf/work table at offices, ...) equipped with a beacon.



- Add the distance (*availability*) as a factor to the logic of when the user should be *prompted* to take the medication.
- Track the time/instance (and if scope allow, dosage) when the user takes the medication and show visible indication of when the medication should be taken next.
- Easy visibility of medication status accessible to the user.
- Add an integration for smart-watch (via notifications or customised component) for interacting with the prompt / see medication status.

References

- [Ultra-wideband \(UWB\) communication support on Android Devices](#)
- [Estimote UWB beacons](#)
- [Estimote SDK for Android](#)

Notes

- Shortcut AS will make available the Estimote Beacons required for development and testing during the project.
- Shortcut can also provide android test device(s) needed for testing the app under development.

MEDICAL APPS, IS CERTIFICATION REQUIRED? CHECK WHETHER YOUR APP REQUIRES CE MARKING

Consumers and healthcare providers increasingly use medical apps. Some medical apps are considered as medical devices. CE marking ensures that medical devices comply with the European requirements. This infographic shows whether a medical app needs CE marking. After going through the flow chart you will also know which risk categories the app comes under and what form of certification is required.



Nictiz has drawn up this flow chart on the basis of the general MEDDEV documentation for medical devices. The aim of this chart is to explain the assessment specifically for medical apps. This chart is not intended to serve as a full legal assessment and does not reflect the classification rules in full detail. Please refer to the decision trees in classification guidance document MEDDEV 2.4/1 documentation and Annex IX of the MDD for this purpose. The chart furthermore does not address classification of in vitro diagnostic standalone software, see Annex II IVD directive and MEDDEV 2.1/6, p. 13 in this respect.

A.4 Reference Group Contract

Reference Group Contract

Medicine Reminder System

As part of the BSc the group would like to invite you to be a test user of the medicine reminder app where you will play an important part in the development process. The feedback and insights you provide will help improve the app and ensure that it meets the needs of its users. In order to ensure that we are all on the same page, we have created this reference group contract. Please read it carefully and let us know if you have any questions or concerns.

The purpose of this contract is to establish a reliable bond between the developers and the test users during the duration of the development of the medicine reminder application and bachelor thesis; January-May 2023

Expectations:

- Be contacted again at a later time and asked if you are able to participate in user testing the application and give feedback and insights on what you think.
- During the user tests answer to the best of your ability questions regarding the application, but also subjects such as notifications and alarms.
- Report to us any inconsistencies or issues you see or encounter during the user tests.

Rules and Guidelines:

- No personal data or information will be stored however, the user test agrees that their opinions and feedback will be represented in the application and the test results visible in the bachelor thesis will be public in NTNUs online library.
- Test users should provide honest and constructive feedback, and not share information with anyone outside of the testing group.
- Test users should notify if they wish to withdraw from the user testing reference group.

Your willingness to participate in our test group is appreciated, and the value of the feedback and insights you bring will help improve the medicine reminder app. By signing this contract, you agree to the expectations and responsibilities outlined above. If you have any questions or concerns, please don't hesitate to reach out to us. Thank you for your participation!

User Representative

Date

Erling Hoffmann

01.03.23

Contact info:

Alin Mihai Napirca - mihaian@stud.ntnu.no

Thomas Øvstun Ditman - thomaodi@stud.ntnu.no

A.5 All wireframes

Location Detailed

Blank

Medicine remaining:
Falscoetylmoi 250 Tablets (50mg)
Falscoetylmoi 250 Tablets (50mg) **Running low!**

Back

Locations Upcoming Manage Medication Settings

Locations

Location: Kizwah Cabinet
Name: Nephew 1

Location: Darfuran cabinet
Name: Nephew 2

Location: Blank
Name: Nephew 3 **NO live on medication**

Locations Upcoming Manage Medication Settings

Upcoming

Today's Medication Status

Morning Midday Evening

Next medication: 2x Falscoetylmoi 250 Falscoetylmoi

History

Locations Upcoming Manage Medication Settings

Medications

Falscoetylmoi
Scanning, midday, and evening
Labeled as Critical medication

Factory/lot:
In factory

Add new medication

Locations Upcoming Manage Medication Settings

Medication Desc

Falscoetylmoi
Description:
.....

Dosage:
2x Pills
Strong per pill
Taking at:
Morning, evening
Pill count left:
12

Return

Locations Upcoming Manage Medication Settings

Settings

Screen enabled

Text size **Large**

Morning alarm: 00:00 Midday: 12:00 Evening: 18:00

Alarm sound: Default

Vibration

Return

Locations Upcoming Manage Medication Settings

Wizard 0

Notice
You will need to have your medication in front of you to input the new description.

Cancel Next

Wizard 2

Enter name of description of medication (optional)

Back Next

Wizard 1

Enter name of medication

Back Next

Wizard 3

Enter times of medication

Morning Midday Evening Custom

Back Next

Wizard 4

Enter dosage

Back Next

History

HISTORY

January

29-01-2023 Monday

04:12 Falscoetylmoi taken

12:20 Falscoetylmoi taken

20:17 Falscoetylmoi taken

30-01-2023 Tuesday

04:00 Falscoetylmoi taken

12:15 Falscoetylmoi taken

20:18 Falscoetylmoi taken

31-01-2023 Wednesday

04:03 Falscoetylmoi taken

12:20 Falscoetylmoi taken

20:02 Falscoetylmoi taken

February

Notification, not near

Some time to take medication, but you are not near your medication

Notification, near

Time to take medication

Medication Info Page

Medication Name:

Medication description:

.....

.....

.....

.....

A.6 User Tests Round 1

User tests round 1:

Question:

Have you used/will use any medication regularly over a longer period of time?

Have you ever used a medicine reminder system such as pill box or a phone app?

So, this is the main screen: Show "Upcoming page"

What do you see? -> Explain color scheme on "Morning, Midday, Evening"

How would you go about to add a new medicament to the front/start page?

Can you view your medication description?

Can you go back and check your medication history?

What can you see on the history page?

Do you know what beacons are?

Can you go and check the beacons page, what do you see?

What are your thoughts on notifications?

- Do you like more/less notifications?

- What about frequency?

Show notifications:

Do you read the whole notification?

What are your thoughts on snoozing notifications?

After seeing this, would you consider using our app in case you ever need to take medication regularly?

USER 1:

Yes

No

Some color scheme, maybe indicates whether or not medicine has been taken at said time of day? Don't know what blue color is. Also, "my medication" is on the page.

Go to the bottom left button "Add new medicine". At new page click "Add medicine" and go through wizard -> description maybe useless since you as the patient know what/why you take your medicine?

Click on the info tab on the "Add new medicine" window.

(Had to explain no back button due to it being a Samsung/Android app). User proceeded to "step back" to the main page (Upcoming) and from there click the "History" button.

Seems like a monthly based history page where I can view the last three days of taken medication.

No, haven't heard of them before.

My two beacons, one stationed in the kitchen and one in the bedroom.

I use them as indicators if I am to open the app sending those notifications.

Less and consistent is better.

More frequent as it approaches time to take medication.

No, I know what each application does so depending on what app it is I open it instantly, or if it is not important, I open the app at a later time.

I see the benefit if for example you would be stuck in a meeting with ten minutes to go before you have to take your medication, then you could snooze the alarm and have it send a new notification after the meeting is over, however I can see it could be dangerous to keep snoozing and omitting to take your medicine.

If I only had to take a single medication a would use a pill box if I really needed to, however I would consider it if I was to take more than one medicine at a time.

USER 2:

Yes

No

Sees medication and medication info. Color scheme is a bit confusing. Sees different times of day.

Bottom right button goes to "Add new medication", here another button for "Add medicine" and proceed through wizard. Likes the "description tab as it may be good to write a description whenever you for example have to take two medications at the same time and one or both depend on a diet or has to be taken in a specific order.

Got a bit lost at the beginning, tried to go back to "Upcoming page" first but found the info button in the end.

Remembers the button from the upcoming page so moved back there and clicked on the "History" button.

Monthly view with days. Medication name & time.

No

Two boxes, indication my two beacons I think. Seems like they share the same name, one stationed in the kitchen and one in the bedroom.

Good, I read my notifications and rely on them usually before i open an app.

I like more notifications, also enjoy when they make stories / use humour. (I see you havent been near your medication in a while! What happend, are you lost? :P)

Would like to see more frequent notifications as it approaches time to take medication.

Yes

Sees it as a good addition, "the good outweighs the bad".

I regularly take asthmatic medicine and would love to use such an app as a pill box would not be very suited for my type of medication.

USER 3:

No

No

Seems like a front page, it shows some medicine and a timer, also a color code for morning, midday and evening, probably to mark off if medicine has been taken at that time?

Looked a while at the buttons, then proceeded to click on "Add new medicine". When prompted with a new page, looked a bit around and then clicked on "Add medicine", then went through with the wizard

Quickly noticed the info button and clicked on it to view medicine description.

Remember a button that could do such functionality on "Upcoming" page, so the user went back to find the "History" button.

Monthly view with everyday + time of day the user has logged they have taken their medicine. Reminds them of a bank transaction history, which is pleasant.

No

Two boxes, my beacons with their name and location.

Ok, don't like when they are "spammy"

Less, i do not like getting spammed that I am not on an app.

Frequency should vary on the situation -> long time till medication little to no notifications. closer to medication -> more notifications

Sometimes, depends on the app

Would be interesting but may develop into a habit.

Seems like an interesting idea, however it would mean I have to set up a beacon system also. -> vet ikke?

USER 4:

No

No

Some color scheme on top, probably depicting whether or not I have taken medicine for that time of day, and some medicine with dosage and time left till medication.

Found the button labeled "Add new medicine" quickly, and when prompted with new page the user immediately clicked on the "Add medicine" button and then went through the wizard to add new medication? (Description not very useful?)

On the page the user quickly found the "info" button. (What is icon used for? ->Antar de er info iconet?)

The user maneuvered back to the front page (Upcoming) and found the "History" button.

Month of January with last three days, what medication was taken at what time and the month of February at the button not selected.

No

My two beacons, their name and location.

Like to be notified about news, but not "deals"

I enjoy useful notifications; in this instance I would take more to appeal to the "intended" target group (older people).

Personally, would take more to be on the safe side,

Well that depends on the app.

Maybe useful in some scenarios but could be dangerous.

I would consider it, I haven't used a pill box either so I wouldn't really know until I try.

USER 5:

Yes

Yes

Three times of day, one for each meal?

From the "Add new medication", went to "Add medicine" and proceeded through wizard.

Found the (i) icon quickly to find medication description.

Went back to the Upcoming page to find the "History" button, remembered it from the initial start of application.

- Monthly view with days. Medication name & time.

No

Seems like my "beacons" with location. Named also, not by me or?

Notifications are good, I often forget if I don't get reminded.

More, but within reason.

Would like to see more frequent notifications as it approaches time to take medication.

Yes

Should be optional, good thinking!

Yes, I did not use a medical pill box, but this looks like something that would be beneficial for me!

USER 6:

Yes

No

Confusing color scheme, looks like the time of day for medication intake?

Quickly maneuvered to the button at the bottom of the screen, clicked "Add new medication", then on the new page clicked on the "Add medicine" button then went through the wizard seamlessly. Asked about a possible calendar solution.

Clicked on the newly appeared medication to open the additional information context.

Remembers the button from the upcoming page so moved back there and clicked on the "History" button.

Remarks that it looks similar to a banking history page -> Medication name & time.

No

My two beacons, one stationed in the kitchen and one in the bedroom and their name

Dislikes notifications, disables notifications for many applications because they are bothersome.

Dislikes a large number of notifications, however within the project context would be open for some creative notifications than may lay emphasis on humor?

Important medicine should have more frequent notifications.

No

Snooze may be a good workaround whenever the user can not take their medicine.

Yeah, looks interesting, never used something similar.

USER 7:

No

No

Color scheme is odd, maybe indicates whether or not medicine has been taken after this meal? Don't know what blue color is.

Bottom right button goes to "Add new medication", here another button for "Add medicine" and proceed through wizard. Likes the "description tab as it may be good to write a description whenever you for example have to take two medications at the same time and one or both depend on a diet or has to be taken in a specific order.

Found the (i) icon quickly to find medication description.

Wanted to move backwards to the Upcoming page because the user remembers the button there but had some trouble remembering the name of the page.

Monthly view with days. Medication name & time.

No

A pair of beacons? They are named and assigned to a location.

Good, I often dictate if I open the app or not based on notification.

I like more notifications.

For important medication there should be more notifications, it does depend on the urgency, i.e how close to medication intake it is.

Yes

Sees it as a good addition, the user should be allowed to choose what suits them best.

Looks like a superb application, would love to see this fully done. Would use!

USER 8:

No

No

Odd choice of colors, however it looks like there is the status of the medication intake for different time of day.

Bottom right button goes to "Add new medication", then clicked on the "Add medicine" button and proceeded through wizard.

Got a bit lost at the beginning, tried to go back to "Upcoming page" to click for info there first but found the info button in the end.

Went back to the "Upcoming page" and clicked the "History" button there, helped that the user got a bit lost during the previous question.

Some banking overview likes the familiarity.

No

Two boxes, indicating my beacons. They are assigned a name and a location.

I rely on my notifications to open an app, not much of a phone person.

I like more notifications; I tend to forget more now with age.

Would like to see more frequent notifications as it approaches time to take medication.

Yes

Snoozing should be something the user themselves decide upon, I would personally have it on.

Would love to test this out, I don't think a pill box would be of much help to me.

USER 9:

Yes

Yes

Sees medication and medication info. Color scheme is a bit confusing. Sees different times of day.

From the "Add new medication", went to "Add medicine" and proceeded through wizard. Liked the option for a description. Maybe for date, add a calendar?

Found the (i) icon quickly to find medication description.

Remembers the button from the upcoming page so moved back there and clicked on the "History" button.

Remarks that it looks similar to a banking history page, its good to have something familiar, makes it more understandable -> Medication name & time.

No

My beacons with name and location

Don't like notifications that are pesky, should only send one out if there is something important happening.

The fewer the better, I do appreciate "good/informational" notifications.

For this application, a larger number of notifications than I would prefer seems reasonable because of the importance that the notification reaches the user.

No, I tend to know what the notification is about and open the app accordingly. I.e., Messenger notifications are because someone messaged me.

Should be a choice, the user decides.

Looks interesting, not very well-versed with beacons, are they necessary for the app to function?

USER 10:

Yes

Yes

Sees medication and medication info. Color scheme is a bit confusing. Sees different times of day.

Looked a while at the buttons, then proceeded to click on "Add new medicine". When prompted with a new page, looked a bit around and then clicked on "Add medicine", then went through with the wizard. Loves the addition of the description field as because of experience with medication that involved a rigid diet that could have been noted in the application.

Found the (i) icon quickly to find medication description.

Remembered the button from the Upcoming page so moved back there and clicked on the "History" button.

Monthly view with days within each month, contains the medication name & time. Maybe not very important if used with a pill box?

No

My two beacons, their name beacon 1 and beacon 2 and their location one stationed in the kitchen and one in the bedroom.

Notifications are good for elder people like me who tend to forget more.

I like more notifications, also enjoy when they use humor and colors!

Would like to see more frequent notifications as it approaches time to take medication.

Yes

Sees it as a good addition, there are more positives than negatives. Better to have the option remain to the user to avoid a tragedy.

Looks wonderful! Would love to use something like this, I have never seen anything like it before.

USER 11:

Yes

No

Sees medication and medication info. Color scheme is a bit confusing. Sees different times of day.

From the "Add new medication", went to "Add medicine" and proceeded through wizard. Liked the option for a description. Add one of those "neat" calendar, makes date picking easier.

Found the (i) icon quickly to find medication description.

Went back to the "Upcoming page" and clicked the "History" button there.

Medication name & time. Have a nice arrangement with days where medicine is taken for each month.

No

My two beacons, one stationed in the kitchen and one in the bedroom.

Yes, however I dislike when they are too pushy, send me the important stuff, not all news.

If they are important I don't mind, however I would not like for apps to remind me of deals I do not care about.

Frequency should vary on the situation -> long time till medication little to no notifications. closer to medication -> more notifications

Depends on the application.

Should be something the user decides, however I would be for a snooze button. Makes managing time a bit easier knowing I can reliably be notified again of my medication.

Looks like something I would at least try if it would help me, however I don't have the necessary experience with long treatments to know how it would benefit me.

USER 12:

Yes

Yes

Peculiar blue color? Looks like there is the status of the medication intake for different time of day.

Bottom right button goes to "Add new medication", then clicked on the "Add medicine" button and proceeded through wizard.

Got a bit lost at first but found the info button in the end.

The user maneuvered back to the front page (Upcoming) and found the "History" button.

Monthly view with days. Looks like a banking history page, nice for a familiar look.

No

My two beacons, their name beacon 1 and beacon 2 and their location one stationed in the kitchen and one in the bedroom.

Good, I tend to read them before opening my applications.

I enjoy useful notifications, for this instance I would love to have more notifications, especially when my medicine can be crucial to my survival.

Would like to see more frequent notifications as it approaches time to take medication.

Yes

Should be enabled, because of how crucial it can be for someone to miss their medication, especially because they cancelled an alarm at an unfortunate time.

I have used pill boxes before and this looks like a neat application to use in conjunction with them, because of the reminding notifications and not only relying on me seeing the pill box!

A.7 User Tests Round 2

User tests round 1:

Question:

Start by showing the wireframe used in round one of testing and then show the application and explain the process of going through feedback to improve the wireframe into the application.

On the Upcoming page, what do you see?

Add a medication, explain how you would do that.

View the additional information about the medication you just added beyond what is shown on the Medication page.

(Optional add more medications to view the scrolling functionality)

Go to Upcoming what changed?

Go to settings and change hour for a time of day, then return to Upcoming page, what changed?

Go to locations, add a new beacon.

Add a medication to the beacon you just created.

With the progress made from a wireframe to the actual application are you content with the product, and would you use it if you had to take a treatment over a longer period of time?

USER 1:

Seems similar to the wireframe just presented.

I see the times of day and a countdown towards something.

The user seamlessly went to the “Medication page” and started going through the wizard. Likes the addition of the calendar (datePicker). The user remarks it was “familiar” because of the last user test.

The user quickly found the (i) icon and viewed the additional information. Liked the addition of the “Remaining treatment period field”.

Skipped.

Now the countdown seems more reasonable, however still, why the blue color?

The user went to the “Settings page” and changed the Morning time of day clock from 12:00 to 13:54. The countdown is now only 15 minutes, cool!

The user went to the last button on the navigation bar and arrived at the “Location page”, on here added click on the button and added a new beacon at “Home” (self-chosen location).

Was a bit confusing at first, but the user clicked on the new beacon, went to add a medication and halted again. The by accident clicked the medication and it disappeared unsure if they did something wrong. Going back to the “Location page” the medication was there. Remarkd there was not intuitive that the medication just disappears from the list.

Looks like you guys did a great job! A bit confusing however, this is something that I can get used to. Would definitely try it, however I do not know if I’ll be aware that such a solution exists.

USER 2:

Looks good, I enjoy a bit of color rather than the bland white one.

A countdown on the middle, the times of day on top, all blue.

The user remembers well the first round of user tests and maneuvered quickly into the wizard. Remarks it is nice, and that data remains in the fields going back and forth between steps. Also enjoys the calendar (datePicker) and likes it more than an input field.

The user quickly found the (i) icon and viewed the additional information. Liked the addition of the "Remaining treatment period field"

Added two more medications, but since it took a long time decided to skip.

Countdown changed on the screen.

The user went to the "Settings page" and changed the Morning time of day clock from 12:00 to 14:20. Going back to the Upcoming page the countdown changed!

Maneuvered to the "Location page", on here added click on the button and added a new beacon at "Office" (self-chosen location).

Clicked on the beacon, and then from the list appeared clicked on all added medications. Going back to the "Location page" the medications are now under "Office" beacon.

Very happy with how it turned out, would love to use something if I would have to undergo a longer treatment.

USER 3:

Remembers well the previous user tests, likes the familiar look.

The Upcoming page seems similar, also there is a countdown now, but there is no medicine?

The user seamlessly went to the "Medication page" and started going through the wizard. Likes the addition of the calendar (datePicker). The user remarks it was "familiar" because of the last user test.

The user quickly found the (i) icon and viewed the additional information. Liked the addition of the "Remaining treatment period field". The previous round of user tests helped!

Skipped.

Now the countdown seems more reasonable, however still, why the blue color?

The user went to the "Settings page" and changed the Morning time of day clock from 12:00 to 14:44.

The user went to the last button on the navigation bar and arrived at the "Location page", on here added click on the button and added a new beacon at "Home" (self-chosen location).

Was a bit confusing at first, but the user clicked on the new beacon, went to add a medication and halted again. The by accident clicked the medication and it disappeared unsure if they did something wrong. Going back to the "Location page" the medication was there. Remarked there was not intuitive that the medication just disappears from the list.

Looks like you guys did a great job! A bit confusing however, this is something that I can get used to. Would definitely try it, however I do not know if I'll be aware that such a solution exists.

USER 4:

Enjoyed having a refresher does not exactly remember what the tests were about the last time.

Same color scheme on time of day, maybe add another icon to the blue color? Maybe a stopwatch.

The user went to the "Medication page" from the navigation bar, remarked it is nice that the buttons down below are labeled. The clicked on the "Add medicine" button and went through the wizard. The fact that the user is advised to have their medication with them is an important factor, well-thought out!

Clicked on the (i) found the additional information, likes that the remainder of the treatment period is shown.

Skipped.

Countdown still odd? Dislikes the blue color still.

The user went to the "Settings page" and changed the Midday time of day clock from 12:00 to 15:01. (The user picked only the Midday time of day and the student remarked the other two options, Morning and Evening, will not affect the countdown and advised the user to only change the Midday option.)

Clicked on the last button on the navigation bar as remarked by the test user and arrived at the "Location page", on here added click on the button and added a new beacon at "Me" (self-chosen location).

Rather confusing, there is no indicator that something happened. Managed to click on the beacon and add the medication, but would like to see some message that the action has been performed rather than the medication just disappearing.

Wonderful solution! Seems like you guys took to heart the feedback received, and it shows. Would definitely try!

USER 5:

Familiar look, enjoy the fact that there is some color now rather than the white one.

I see the times of day and a countdown towards something.

The user seamlessly went to the "Medication page" and started the wizard. Enjoys the calendar (datePicker).

The user quickly found the (i) icon and viewed the additional information. Liked the addition of the "Remaining treatment period field."

Added 12 medications to test the scrolling, neat however I doubt I'll ever have this many medications to take, hopefully never!

Cool that the countdown is affected by the time of day, still rather odd number, no?

The user went to the "Settings page" and changed the Morning time of day clock from 12:00 to 15:35.

Maneuvered to the "Location page", on here clicked the button and added a new beacon at "At home" (self-chosen location).

Was a bit confusing at first, but the user clicked on the new beacon, went to add a medication and halted again. The by accident clicked the medication and it disappeared unsure if they did something wrong. Going back to the "Location page" the medication was there. Remarkd there was not intuitive that the medication just disappears from the list.

Very nice solution! I would consider it; I haven't used a pill box either so I wouldn't really know until I try. Maybe I can use the two together?

USER 6:

Familiar look, its good to get a refresher.

Sees a table with a countdown. Sees different times of day at the top.

The user went to the "Medication page" and started going through the wizard.

Clicked on the (i) found the additional information, likes that the remainder of the treatment period is shown.

Skipped.

Countdown is changed and it looks reasonable now!

The user went to the "Settings page" and changed the Evening time of day clock from 12:00 to 15:59. (The user picked only the Evening time of day and the student remarked the other two options, Morning and Middy, will not affect the countdown and advised the user to only change the Evening option.)

Maneuvered to the "Location page", on here added click on the button and added a new beacon at "Home" (self-chosen location).

Clicked on the new beacon and was prompted with a new page. Here clicked on the medication showing and then clicked on the button to finish the action. The medication is now under the "Home" beacon.

Looks interesting and it shows that you guys put a lot of effort into it, however I'm not very well-versed with beacons, are they necessary for the app to function? If not, this looks like a neat solution!

A.8 student timeallocation

User / Worklog	Logged
Alin Mihai Napirca.....	581.92
09/Jan/23 - Research & project plan writing	1
09/Jan/23 - First internal meeting discussing technologies and setting up meetings	2
11/Jan/23 - First meeting with supervisor	0.5
12/Jan/23 - Jira research	1
12/Jan/23 - Reasearching beacons & estimate a bit	1.25
12/Jan/23 - Project plan work	3
13/Jan/23 - Working with documentation	6
14/Jan/23 - Quick online meeting	0.25
14/Jan/23 - Working with documentation	4
14/Jan/23 - Working with documentation	1.5
14/Jan/23 - Project plan work	3
16/Jan/23 - Risk analysis & latex research	7
17/Jan/23 - Project plan writing: Gantt chart, Planning & Roles	6
18/Jan/23 - Meeting with supervisor	0.5
18/Jan/23 - Internal meeting, discussion whats missing on project plan	0.33
18/Jan/23 - Working on issue ATBFS-1	2
18/Jan/23 - Working on issue ATBFS-1	1.5
19/Jan/23 - Finishing up roles & rules for project plan	2
19/Jan/23 - Short meeting to prepare for workday at Shortcut	0.33
19/Jan/23 - Finishing touches	2
19/Jan/23 - Meeting with Shortcut	0.75
20/Jan/23 - Workday at Shortcut	8
21/Jan/23 - Project plan work	2.5
23/Jan/23 - Research for app	1.25
23/Jan/23 - Use case work	3
24/Jan/23 - Meeting with Shortcut	1
24/Jan/23 - Internal meeting	0.5
24/Jan/23 - Wireframe & Use case work	4.5
25/Jan/23 - Meeting with supervisor - feedback project plan	0.5
25/Jan/23 - Improve project plan with given feedback	4
26/Jan/23 - Project work & research	6
27/Jan/23 - Jetpack Compose research	3
27/Jan/23 - Finishing up on improving project plan	2.5

User / Worklog	Logged
28/Jan/23 - Specification of Requirements document work	2
28/Jan/23 - Jetpack Compose research	1
29/Jan/23 - Use case stories & roles	3
31/Jan/23 - Internal meeting	2
31/Jan/23 - Finishing touches to get ready for presentation	2.5
31/Jan/23 - Wireframe	3
01/Feb/23 - Meeting with supervisor & Shortcut	0.58
01/Feb/23 - Wireframe updates	2
02/Feb/23 - Work on improvements from yesterdays meeting	4
02/Feb/23 - Work on improvements from yesterdays meeting	2.5
03/Feb/23 - Finishing document	3
04/Feb/23 - Drafting questions for user tests	2.5
04/Feb/23 - Tested mom & dad + two friends on our wireframe and some questions around notifications and design.	4
04/Feb/23 - Internal meeting	0.42
06/Feb/23 - Improvements to Specification of Requirements document	1.5
06/Feb/23 - Advanced use case work	3
07/Feb/23 - Advanced use case work	2
07/Feb/23 - Explore tools / research	3.5
08/Feb/23 - Jetpack compose research	2
08/Feb/23 - Jetpack Compose research from book	3
09/Feb/23 - Meeting with supervisor & Shortcut	0.5
09/Feb/23 - Meeting to discuss CI/CD pipeline, user tests, and reference group document	1.5
09/Feb/23 - Reference group document research	1
09/Feb/23 - Jetpack Compose research	2
10/Feb/23 - Setting up meetings/user tests for next week	2
10/Feb/23 - Reference group contract research	1.5
10/Feb/23 - Draft Reference group document	2.25
10/Feb/23 - Testing CI/CD pipeline	1
11/Feb/23 - Programming	2
11/Feb/23 - Programming	1.5
12/Feb/23 - Creating reference group contract	2
13/Feb/23 - Document work	2.5
13/Feb/23 - Done with document	1
13/Feb/23 - Meeting work	0.75

User / Worklog	Logged
13/Feb/23 - Internal meeting	0.5
14/Feb/23 - Some research	1
14/Feb/23 - Checkout pipeline setup	1.5
14/Feb/23 - Checking documents	1
15/Feb/23 - Meeting with supervisor & Shortcut	0.5
15/Feb/23 - Looking at discussed technologies	2
16/Feb/23 - Jetpack Compose column research	1.5
16/Feb/23 - Programming	3
17/Feb/23 - Research	1
17/Feb/23 - Programming	2.5
18/Feb/23 - Programming	3
18/Feb/23 - Jetpack compose research	3
19/Feb/23 - Programming	2.5
20/Feb/23 - Programming	2
20/Feb/23 - Programming	1
20/Feb/23 - Look at merge requests	1.5
21/Feb/23 - Look at merge requests	1
21/Feb/23 - Programming	1
22/Feb/23 - Meeting with supervisor & Shortcut	0.5
22/Feb/23 - Programming	2
22/Feb/23 - Merge conflicts/Prep for meeting	1
23/Feb/23 - Programming	2.5
25/Feb/23 - Programming	2.5
25/Feb/23 - Getting signatures	0.5
25/Feb/23 - Programming	2
25/Feb/23 - Fixing meeting reports	0.75
26/Feb/23 - Programming	3
26/Feb/23 - Fixed some meeting details	2
27/Feb/23 - Merge requests	1
27/Feb/23 - Programming	4
28/Feb/23 - Pushing wizard to git	1
28/Feb/23 - Programming	5
01/Mar/23 - Meeting with supervisor & Shortcut	0.5
02/Mar/23 - Programming	2.5
03/Mar/23 - Programming	5

User / Worklog	Logged
04/Mar/23 - Programming	2
04/Mar/23 - Programming	1
04/Mar/23 - Programming calendar	3
06/Mar/23 - Programming	5
07/Mar/23 - Working on medication page & UI	3
07/Mar/23 - Programming w/ Thomas	3
08/Mar/23 - Meeting with supervisor & Shortcut	0.5
08/Mar/23 - Programming	2
08/Mar/23 - Jetpack Compose Research	1
09/Mar/23 - Working on issue ATBFS-5	1
09/Mar/23 - Working on issue ATBFS-5	4
10/Mar/23 - Working on medication page & wizard	3
10/Mar/23 - Working on medication page & UI	2
11/Mar/23 - Working on medication page & wizard	6
12/Mar/23 - Programming - Calendar Wizard	4
12/Mar/23 - Programming - Calendar Wizard	3
13/Mar/23 - Programming - Calendar Wizard	5
16/Mar/23 - Workday at Shortcut - Meetings - Code review/walkthrough - Application development - GDC Android event	11
17/Mar/23 - Code refactoring	3
18/Mar/23 - Code refactoring	3
18/Mar/23 - Code refactoring	2
18/Mar/23 - Code refactoring	2.5
20/Mar/23 - Wizard work	2
20/Mar/23 - Programming & Research	2
21/Mar/23 - Programming -> Beacons troubleshooting	2
21/Mar/23 - Working on issue ATBFS-29	2.5
22/Mar/23 - Meeting with supervisor & Shortcut	0.5
22/Mar/23 - Wrote on report	2
22/Mar/23 - Report work	2
22/Mar/23 - Wrote on report	2
23/Mar/23 - Medication wizard work	3
23/Mar/23 - Refactoring design	2
23/Mar/23 - Wrote more on report	2
23/Mar/23 - Medication wizard work	1.5

User / Worklog	Logged
24/Mar/23 - Finalizing discussions for first draft	6
25/Mar/23 - Adding fields in report & fixing meeting minutes	2
26/Mar/23 - Conducting user tests	3
27/Mar/23 - Meeting from 14:00 - 15:00 then user tests	2
27/Mar/23 - Conducting user tests	1.5
28/Mar/23 - Worked on finalizing medication wizard	4
29/Mar/23 - External information page for medication	4
29/Mar/23 - External information page for medication	2
30/Mar/23 - Meeting with supervisor & Shortcut	0.75
30/Mar/23 - External information page for medication	4
31/Mar/23 - External information page for medication	3.5
01/Apr/23 - Fixing some comments	1.5
01/Apr/23 - Getting the Reference Group Contract signed and conducting additional tests	3.25
02/Apr/23 - Getting the Reference Group Contract signed	2
03/Apr/23 - Working on report while on Easter holiday	3
04/Apr/23 - Working on report while on Easter holiday	2
04/Apr/23 - Working on report while on Easter holiday	1
05/Apr/23 - Meeting with supervisor & Shortcut + meeting minutes refactoring	1.5
06/Apr/23 - Working on report while on Easter holiday	2
06/Apr/23 - Working on report while on Easter holiday	2
07/Apr/23 - Working on report while on Easter holiday	2
08/Apr/23 - Working on report while on Easter holiday	4
08/Apr/23 - Working on report while on Easter holiday	2
09/Apr/23 - Setting up a preview of a datePicker	1
09/Apr/23 - Researching datePickers and timePickers in Jetpack Compose	3
10/Apr/23 - Calendar work	4
11/Apr/23 - Calendar work	1
11/Apr/23 - Calendar work	4.5
12/Apr/23 - Prepping code for pair programming	4
13/Apr/23 - Medication wizard refactoring with Thomas	3
13/Apr/23 - Finishing work on my own	2
14/Apr/23 - Further work with Medication page improvement	4
15/Apr/23 - Medication wizard refactoring with Thomas	2
15/Apr/23 - Finishing a datePicker implementation using M3	1.5
15/Apr/23 - Further work with Medication page improvement	4

User / Worklog	Logged
16/Apr/23 - Further work & refactoring with Medication page	3
17/Apr/23 - Further work w/ Thomas	4
17/Apr/23 - Working on report topics	4
17/Apr/23 - Programming + Researching some kotlinox.datetime library functionality w/ Thomas	2.5
18/Apr/23 - Start introduction	3
18/Apr/23 - Further work & refactoring with Medication page	2
19/Apr/23 - Meeting with supervisor & Shortcut	0.5
19/Apr/23 - datePicker work	1
19/Apr/23 - Refactoring work	2
20/Apr/23 - Working on improving Medication page code	6
21/Apr/23 - Meeting with supervisor & Shortcut	0.5
21/Apr/23 - Pair programming w/ Thomas -> code refactoring	3
22/Apr/23 - Pair programming w/ Thomas -> code refactoring M3 updates for code	4
23/Apr/23 - Working on finalizing code & fixing merge requests	6
24/Apr/23 - Finalizing report with topics on M3 update and GCD meetup at Shortcut	3
25/Apr/23 - Finishing medication extra info page	2.5
25/Apr/23 - Finishing medication extra info page	3
26/Apr/23 - Finalizing codebase for merge	2
26/Apr/23 - Finishing work & Doing merge requests	2
26/Apr/23 - Meeting with supervisor & Shortcut: - Prepped a demo for presentation	1.25
27/Apr/23 - Finishing work & Doing merge requests	4
27/Apr/23 - Finishing medication extra info page	3
28/Apr/23 - Meeting with Shortcut, Peter could not attend	0.5
28/Apr/23 - Glossary	1.5
28/Apr/23 - Merge requests	1
28/Apr/23 - Report work	3
29/Apr/23 - Adding references for existing work in the thesis.bib file	2
29/Apr/23 - Report work	1
30/Apr/23 - Fixing some inconsistencies	2
01/May/23 - Report work	3
02/May/23 - Meeting to discuss topics of the report	0.75
02/May/23 - More report work	4
03/May/23 - Meeting with supervisor & Shortcut	0.5
03/May/23 - Reviewing feedback from Peter (supervisor) regarding report	3
03/May/23 - Reviewing feedback from Peter (supervisor) regarding report	1

User / Worklog	Logged
04/May/23 - Report work, fixing pictures and tables	3.5
04/May/23 - Adding topics to Introduction section	2
04/May/23 - Reviewing feedback from Peter (supervisor) regarding report	1
05/May/23 - Working on topics together w/ Thomas	4.5
06/May/23 - Working on topics together w/ Thomas	2.5
06/May/23 - Finishing report introduction	4
07/May/23 - Working on topics discussed at school with Thomas	3
07/May/23 - Working on topics together w/ Thomas	1.5
08/May/23 - Working on report	3
08/May/23 - Working on report feedback from supervisor with Thomas	3.5
09/May/23 - Working on report	7
10/May/23 - Working on report with Thomas	3
10/May/23 - Fixing consistency in report	1
11/May/23 - Finalizing user tests topic, need round two + pictures/appendix	2.5
11/May/23 - Fixing topics regarding user tests	4
12/May/23 - Internal meeting w/ Thomas	0.5
13/May/23 - Round two of user tests	3.5
14/May/23 - Finishing introduction topics with pictures and references	5
14/May/23 - Working on user test topic in report	3
15/May/23 - Meeting with supervisor & Shortcut	1
15/May/23 - Report work	3
16/May/23 - Report work	2.5
16/May/23 - Report work	3
17/May/23 - Report work w/ Thomas	4
18/May/23 - Report work, fixing typos and refactoring some paragraphs	5
19/May/23 - Finishing report work, need to add pictures & tables + references for both	4
20/May/23 - Finished report, need to add some glossary references and appendix for user tests	5.5
21/May/23 - Finishing touches: - Fixing all pictures, tables and appendices - Glossary - Last grammar check - Adding all extra documents like meeting minutes and hours spent	7

Sak	Registrert tid
ATBFS-4 - Specification of Requirements.....	6.25
ATBFS-8 - Research beacon technology.....	18
ATBFS-9 - Setup CI/CD.....	10
ATBFS-16 - Project Plan.....	40.25
ATBFS-18 - Specification of Requirements.....	9
ATBFS-21 - Wireframe models.....	18
ATBFS-22 - User testing Wireframe Models.....	25
ATBFS-23 - Research into Estimote SDK.....	30
ATBFS-24 - Refine Wireframe models.....	3
ATBFS-25 - Setup CI/CD Detekt.....	6
ATBFS-29 - Code base UI.....	148.25
ATBFS-35 - Setup Project report Doc.....	4
ATBFS-36 - Research potential competitors.....	4.5
ATBFS-37 - Code database connectivity.....	43.5
ATBFS-39 - Report: Development Method.....	14
ATBFS-40 - Report: Requirements.....	9.5
ATBFS-41 - Report: Technologies.....	13.25
ATBFS-42 - Report: User Interface.....	12
ATBFS-44 - Report: Code Quality.....	13.5
ATBFS-46 - Report: Conclusion.....	26.25
ATBFS-48 - Swift learning.....	8.5
ATBFS-49 - Swift programming.....	15
ATBFS-52 - Debug.....	46.75
ATBFS-53 - Refine report.....	8.5

A.9 Meeting Minutes, all meeting minutes can be viewed in the project repository

09.01.2023 Internal meeting

Date

9th of January 2023

Participants

- @Alin Mihai Napirca
- @Thomas Øvstun Ditman

Goals

- Catch up after Christmas break
- Think about the scope of the project and possible technologies we can use
- Set up meeting with supervisor

Discussion topics

Subject	Item	Notes
Scope		<ul style="list-style-type: none">• Talked about possible solutions to motivation/availability/promt for patients• Introduced Estimate Beacons to use
Technologies	<ul style="list-style-type: none">• IMetric• Jira• GitLab (Issue board)• Overleaf• Jetpack Compose	<ul style="list-style-type: none">• Iterated through a few technologies we think we could use, so far we have decided on GitLab for version control, Overleaf for documents and Jetpack Compose as the development tool.
Meetings		<ul style="list-style-type: none">• Setup a meeting with supervisor for 01.11.2023

Action items

- Setup a meeting with supervisor for 01.11.2023

Summary

A short meeting where we discussed the given bachelor project description and some of the technologies we will use such as Jetpack Compose application development, Overleaf for writing documents, and GitLab for project structuring and management

Duration

1 hour and 30 minutes

16.03.2023 Meeting with supervisor & Shortcut

Date

16th of March 2023

Participants

- @Alin Mihai Napirca
- @Thomas Øvstun Ditman

NTNU:

- @Peter Nussbaum - supervisor

Shortcut AS:

- @Jason Toms - Shortcut

Goals

- Updates regarding code & report
- Code walkthrough with Jason

Additional questions

- **Koin usage** and where to use Koin?
- AlertDialog variable for functions
- Android versions
- Date variable for time & calendar
- Settings tab implementation

Discussion topics

Subject	Notes
Updates regarding code & report	<ul style="list-style-type: none">• Quick update on this sprints' progress, things are going fine so far but Alin encountered some difficulties and is a bit behind schedule• Report writing has is on stand-by due to this week having a bit more work than otherwise
Code walkthrough with Jason	<ul style="list-style-type: none">• Code seems to be on the right trajectory, a few things to consider/re-evaluate<ul style="list-style-type: none">◦ Private set of variables in view models<ul style="list-style-type: none">▪ Functions in the view model to change data, that way view model has ownership of variables and the pages display the data.◦ AlertDialog code is messy, take a step back and consider using a new page for each step of the wizard.

Questions from students

- Kotlin usage
 - Used where need so looks good for now
- Android version

✔ Action items

- ✔ Send a report draft to Peter by Wednesday (22.03.2023) next week.
- ✔ Refactor AlertDialog to different screens
- ✔ Refactor view models and view models data to be private and have functions that change values

👉 Summary

Sprint 4 review meeting. The code is coming along well, with slight hiccups that will leak into sprint 5 but should not take more than 1-2 days to catch up. Report writing is halted, but will resume this sprint and the students plan on submitting a report draft by next week's meeting. Additionally, the students together with Jason did a code walkthrough where they asked some questions regarding development and guidelines to follow going forward and got some feedback on what is good/bad and how to improve.

🕒 Duration

1 hour 15 minutes

15.05.2023 Meeting with supervisor & Shortcut

Date

15th of May 2023

Participants

- @Alin Mihai Napirca
- @Thomas Øvstun Ditman

NTNU:

- @Peter Nussbaum - supervisor

Shortcut AS:

- @Jason Toms - Shortcut

Goals

- The final meeting, discuss the report

Discussion topics

Subject	Notes
Report	<ul style="list-style-type: none">• Conclusion<ul style="list-style-type: none">◦ Mention a higher level, see making an approach for health and the betterment of the wellbeing of people◦ Zooming out the perspective of the application◦ Mention the "toolbox" we have built over the project in a reflection◦ Summary: A compression of all the work we did, one sentence from each chapter "Aim is ..." Summary is sort of a short form version of the entire document

Action items

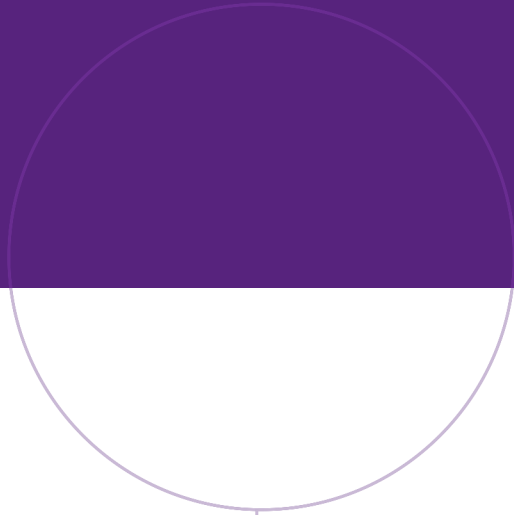
- Update report with given feedback.

Summary

The last meeting together with the supervisor and Shortcut was spent going through the report and receiving some final feedback.

Duration

30 minutes



 **NTNU**

Norwegian University of
Science and Technology