



Norwegian University of
Science and Technology

Prototyping a location aware application for UBiT.

A map-based application, designed, implemented and evaluated.

Bjarne Sletten Olsen

Master of Science in Computer Science

Submission date: June 2009

Supervisor: Ingeborg Torvik Sølvsberg, IDI

Problem Description

In [1] several possible applications based on location-awareness are described.

Based upon these proposals, design a prototype and implement it using existing technologies and tools.

The prototype should demonstrate the possibility for taking advantage of existing services.

Evaluate the prototype in regard to the technology used as well as the usability for library users.

References:

1. Bjarne Sletten Olsen, "Lokasjonstjenester for Universitetsbiblioteket i Trondheim",
Prosjektoppgave NTNU, 2008, 69 p., <http://folk.ntnu.no/oleh/UBIT2010/Lokasjonstjenester.pdf>

Assignment given: 23. January 2009

Supervisor: Ingeborg Torvik Sølvberg, IDI

Abstract

Through the research performed in this thesis, it has been shown how location awareness and maps can be exploited to facilitate the use of library resources, such as information on documents and objects. A prototype has been developed to demonstrate the feasibility of integrating several different information sources for this use. The prototype created allows for users located within the city centre of Trondheim to search for documents and to locate the library departments holding them. The user is shown a map and given information on how to travel to the nearest bus stop, as well as bus schedules on how to get to the selected library department.

Several information sources for the prototype has been identified and evaluated. The prototype communicates with BIBSYS for document information retrieval, Google Maps for map generation, team-trafikk.no for bus schedules querying and Amazon.com and LibraryThing.com for book cover image downloading. To ensure data consistency some local data sources are also maintained, such as a list of all the UBiT (NTNU library) departments in Trondheim.

The prototype was implemented so that it would satisfy a set of requirements. These requirements were created by applying the technique of use cases. Each requirement has been discussed and prioritised based on requests from UBiT. The most important requirements have been incorporated into the design of the prototype. This focuses on modularity and it has been discussed how the external sources best can be integrated with the prototype. The prototype is implemented using a combination of programming languages. The differences between these languages have posed a challenge, and solutions to how these can be avoided are presented. The prototype has been tested according to an extensive test plan, and the results of these tests have been document and evaluated.

Each of the design decisions have been evaluated and discussed, and suggestions on how these could have been improved are given. Finally, suggestions on how the functionality of the prototype can be extended are presented.

The prototype created in this thesis allows for users, familiar or unfamiliar with the city and its transportation network, to locate a document and travel to the library holding it. It demonstrates how emerging technologies such as location awareness can contribute to increased use of library services.

Preface

This thesis marks the end of 5 years at NTNU. This final assignment has required a large variety of the skills I have acquired throughout my education, and has proved to be both challenging and interesting.

I would like to thank my supervisor professor Ingeborg T. Sølvsberg. Her experience and ability to see the larger picture has guided me, especially when I dug myself to deep in technical details.

I would also like to thank UBiT, represented by Ole Husby and Joost Hagle, for participating in defining the requirements for the application.

Trondheim, 17th June 2009
Bjarne Sletten Olsen

Table of Contents

Abstract	i
Preface	iii
Table of Contents	v
List of Figures	vii
List of tables	viii
Listings	viii
1 Introduction	1
1.1 Current situation	1
1.2 Assumptions and premises	2
1.3 Goals.....	2
1.4 Structure	2
2 Background	3
2.1 UBiT.....	3
2.2 Location aware applications today	4
2.3 Previous work.....	4
2.4 Prototype description.....	5
3 Requirements.....	7
3.1 Use cases	7
3.2 Functional requirements	11
3.3 Non-functional requirements.....	13
4 Available technologies	15
4.1 Operating systems	15
4.2 Programming Languages.....	16
4.3 Location technologies	17
4.4 External services	17
5 Design.....	19
5.1 Top-level design	19
5.2 Design considerations	20
5.3 Class Diagrams.....	22
5.4 Sequence Diagrams	24
6 Implementation.....	29
6.1 Mobile application.....	29
6.2 Server side application and adapters	29
7 Walkthrough.....	37
7.1 Logging the users location	37
7.2 Searching for documents	37
7.3 Displaying the map.....	38
8 Testing.....	41
8.1 Testing the mobile application	41
8.2 Testing the server side.....	42
8.3 Testing service communication.....	44
8.4 System Test	44
8.5 Test summary	46
9 Evaluation and discussion	47
9.1 Mobile application.....	47
9.2 Server-side application	48
9.3 Adapters and services.....	49

9.4	The tools and technologies	52
9.5	Usability and usefulness	53
9.6	Conclusion.....	54
10	Future work	57
10.1	Real time bus schedules	57
10.2	Map at the library	57
10.3	Advanced searching options.....	59
11	References	61
A.	Appendix A – Use case point estimation 1	63
B.	Appendix B – Use case point estimation 2	65
C.	Appendix C – Project schedule	67
D.	Appendix D - Test documentation	69
E.	Appendix E - Bus stop addresses	79
F.	Appendix F - Library information file	81
G.	Appendix G - ECDL 2009 accepted poster article.....	85
H.	Appendix H – Source code information.....	91

List of Figures

- Figure 1. Library departments in Trondheim 3
- Figure 2. The overall design with three modules. 19
- Figure 3. Class diagram of the mobile application in C#..... 22
- Figure 4. Class diagram - Server Application 23
- Figure 5. Sequence diagram - Start-up use case..... 24
- Figure 6. Sequence diagram - Locate document part 1 25
- Figure 7. Sequence diagram - Locate document part 2..... 26
- Figure 8. Search interface..... 29
- Figure 9. Partial BIBSYS SRU query result for keyword "Java" 32
- Figure 10. The response received on a bus query 33
- Figure 11. Wireless coverage and located bus stops 35
- Figure 12. Registering user coordinates 37
- Figure 13. Search interface..... 38
- Figure 14. Search results 38
- Figure 15. Map with user indicated..... 39
- Figure 16. Map with library indicated..... 39
- Figure 17. Test documentation overview 41
- Figure 18. TC01-01 - Mobile application 42
- Figure 19. TC02-06 - Correctly indicating bus schedules..... 43
- Figure 20. TC02-07 - Correctly indicating the nearest bus stop 43
- Figure 21. TC03-03 - Comparing document searches 44
- Figure 22. TC04-01 - GPS coordinates registered at the server..... 44
- Figure 23. TC04-01 - The user indicated in the map 45
- Figure 24. TC04-01 - The library indicated in the map 45
- Figure 25. Bing Enterprise Maps ASP.net controller..... 50
- Figure 26. Image service analysis results 52
- Figure 27. Real time bus schedules 57
- Figure 28. Local map of library 58
- Figure 29. Gantt diagram - Schedule for project..... 67

List of tables

Table 1 – Use case 1..... 8
Table 2 – Use case 2..... 8
Table 3 – Use case 3..... 9
Table 4 – Use case 4.....9
Table 5 – Use case 5.....10
Table 6 – Use case 6.....10
Table 7 – Functional requirements.....12
Table 8 – Non-functional requirements.....13
Table 9 – Test summary.....46

Listings

Listing 1: Inverse geocoding.....30
Listing 2: A MARC record in XML format.....30
Listing 3: Determining input line type.....33

1 Introduction

As the number of mobile devices sold reaches new heights every day, ubiquitous computing is becoming a natural part of the modern life. No longer does one have to sit behind a desk to perform tasks such as browsing, searching or editing of documents. As the mobile phones grow more powerful new applications emerge, providing functionalities that not long ago would have been unthinkable. Some of these applications take advantage of the location of the user. Such location aware applications are able to provide contextual information, reconfigure, trigger actions or select nearby objects based on this location.

The NTNU library (UBiT) offers a large variety of services and has a large collection of items distributed throughout various departments in Trondheim. Information on each of these objects resides in a central database. Gaining information on them can be quite time consuming for the user. If the information on these objects and services could become more available and visible to the user, the value of them could be highly increased. One way of accomplishing this could be through the use of location aware services.

In a project that handles this subject, several services that combine multiple library resources and exploits location awareness are presented[1]. Those of these services that could provide the most value to UBiT are combined into the design of one single application. This work takes advantage of GPS[2] and GeoPos[3] to pinpoint a user's location. It describes an application that enables the user to select from a variety of services. One of these services is the possibility to search for a document, where a document is defined as any physical or electronic object registered in the library database. If the document is found the user is presented with a map showing the respective library. A second possibility is to search for a library with documents on a given subject, also providing the user with a map with directions on how to get there. It is also possible to locate libraries with available computers, printers and rooms. The final functionality of the application described in this previous work is to display recent news from the present location of the user.

In the mentioned earlier project an overall design was presented along with recommendations on what positioning technologies and platforms to use[1]. Based on this work, a prototype implementing a subset of this functionality is to be developed. It will demonstrate the possibilities of location aware applications, as well as the integration of these services with other pre-existing ones.

1.1 Current situation

BIBSYS is a database containing information on objects located at 119 different Norwegian libraries. Each library has one or more departments. BIBSYS ASK provides a search interface for information on objects located at each of these departments.

Today a user in need of a document visits the web page of UBiT and enters one or several keywords. BIBSYS ASK is the search interface for the information on documents stored in libraries in Norway. When a search is performed, the user is presented with a listing of the results. By clicking on one of the results, the system shows an overview of the library departments holding the document. When the user select a department, a local shelf-map of the given library department is shown if it is available. The user has to determine the geographical position of the library manually, and no information on how to travel to it is generated. The user has no information on which department is closest to him or her.

1.2 Assumptions and premises

There are some elements of this thesis that are given by the preliminary work in the previous project, as well as by the fact that UBiT has contributed to define the functional requirements for the application [1].

It is assumed that a subset of the services described in the preliminary work will be implemented in a prototype. The information on objects in the library database will be retrieved using BIBSYS SRU, a service developed to access information in the BIBSYS database. It is assumed that the user is within the city limits of Trondheim when the application is used, and it is not required of the prototype to fetch information on objects located in other libraries than UBiT's departments.

1.3 Goals

The overall goals for this project are:

1. Demonstrate how location awareness and maps can be exploited to facilitate the use of library resources, such as information on documents and objects.
2. Show the feasibility of integrating several different information sources into the prototype.

1.4 Structure

Chapter 1 gives an introduction to the thesis, and describes the overall goals for the work. In chapter 2 UBiT is presented and a short presentation of existing location aware applications today is given, along with a description of the work that preceded this thesis. Then a description of the prototype and its functionality is given. Chapter 3 determines and ranks the functional- and non-functional requirements for the prototype, while chapter 4 describe some of the technologies that can be used to meet these requirements. The design of the prototype is described in chapter 5, before some important design considerations are discussed in chapter 6. A walkthrough of the functionality of the prototype is given in chapter 7. The result of testing the prototype against the requirements is presented in chapter 8, and the prototype is evaluated and discussed in chapter 9. Chapter 10 suggests some additions to the prototype that can be implemented at a later time.

2 Background

This chapter describes the background for the thesis. The NTNU library (UBiT) and BIBSYS are presented. An overview of the work done in a previous project is presented. Finally the desired functionality of the application is described, before some existing location aware applications are presented.

2.1 UBiT

As described earlier, the UBiT library consists of large number of departments. In addition there are other libraries that also have a number of departments spread across Trondheim. Figure 1 shows the departments of UBiT, HiST (Trondheim and Sør-Trøndelag University College) and Folkebiblioteket in the centre of Trondheim. Some departments are located further away from the city centre and are not shown. The map shows that it can be difficult to locate a specific document within a specific department. This thesis focuses on UBiT and its data collections. UBiT consists of 11 departments, and subscribes to a large number of journal databases and online databases.

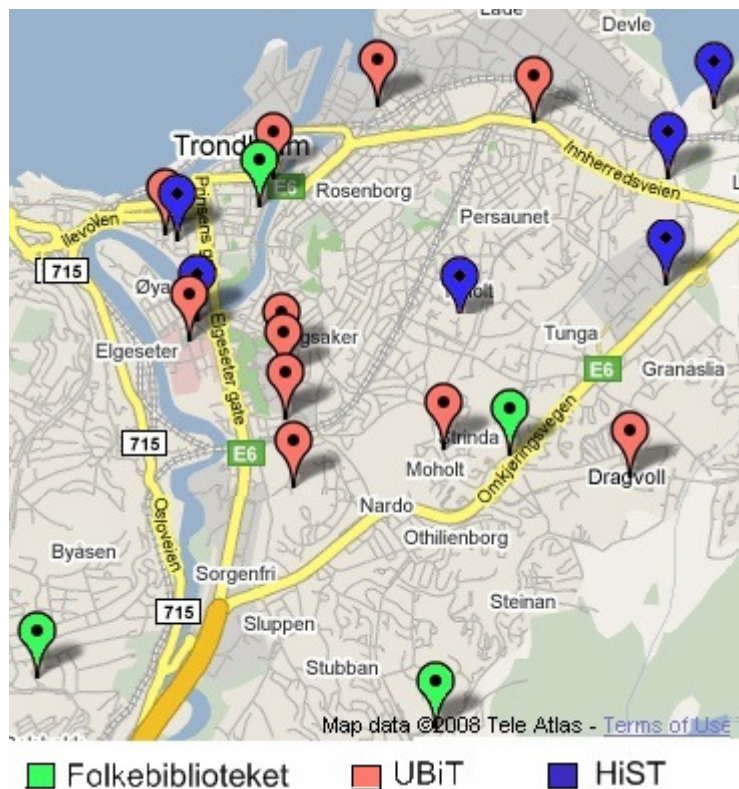


Figure 1. Library departments in Trondheim

2.1.1 BIBSYS

UBiT participates in BIBSYS, which is a collaboration between 119 libraries in Norway currently holding information on more than 14,800,000 items in its database[4].

BIBSYS offers BIBSYS ASK, a service that allows for search in printed literature and journals using simple or advanced search. When searching in BIBSYS ASK, the search is also relayed to MetaLiB which offers searches in digital book packages, FRIDA which contains information and documentation of scientific activities, and DIVA which offers scientific publications from universities across the Nordic countries. The information in the BIBSYS database is described in MARC format[5, 6]. MARC is an international metadata-format used worldwide. A MARC file consists of a series of lines, where each line contains some information on the document described. Each line begins with either a variable- or data control field that defines what information the line contains. As an example, a line beginning with the number 245 contains the title of the document. This structured data is well suited for parsing by an application that e.g. extracts information on documents.

In addition to the information available through BIBSYS, the library provides some special collections, including historical archives and picture collections.

2.2 Location aware applications today

There already exist a variety of location aware applications. This section gives a short presentation on two of them, demonstrating some of the possibilities that location awareness provides. Additional applications are described in the preliminary project [1].

WhosHere

WhosHere combines multiple localisation technologies to provide the user with information on other people, by performing searches on predefined attributes [7]. By searching for e.g. persons that could be interested in a carpool, users nearby with this attribute indicated are shown in a list, and you are given the opportunity to chat with them directly. The application allows you to see the position of users you previously have chatted with. It currently runs on iPhone and iPod Touch.

Sekai camera

Sekai camera is another iPhone application [8]. By holding the mobile phone in front of you and pointing the camera at any object, information registered by other users on the object in view is shown. It also allows for the user to add additional information on the objects. The application rely on GPS to determine the users position

2.3 Previous work

The prototype developed in this thesis, is one of several suggested applications described in a project produced in autumn 2008[1]. Among other suggestions was the presentation of historical photographs in a map, promoting library events, an interactive city guide and presentation of community generated content. This section gives a short description of these applications.

Presentation of historical photographs

More than 350 000 photographs are available through UBiT and their partners. By registering the position of the user, images from places nearby may be presented. Along with these

photographs, information on them can be shown, as well as information on where literature relevant to the photographs can be found.

Promoting library events

UBiT hosts several exhibitions. When a user is inside a library, information on upcoming events can be pushed to the user. This could in the form of a web page, or an mms containing an image of a poster delivered directly to the user's mobile device.

Interactive city guide

A large amount of tourists visit Trondheim every year, and a map on the users mobile device containing suggested routes or points of interest could make it easier for tourists to navigate the city. When the user is at a pre-indicated point of interest, historical photographs and information is displayed on the mobile device.

Community generated content

An open community solution allows for the users to append their own information to e.g. photographs in the UBiT image database. This can allow for large amounts of new information, but would also require some sort of quality control.

2.4 Prototype description

UBiT wants to increase the usage of its collections and to offer improved services, taking advantage of new technologies. The library has, as described in the beginning of this chapter, a large amount of information available. A challenge for the user can be to locate the exact document that is desired. By taking advantage of location aware services and mobile devices, the user can reach information on this document at the time and place preferred by him or her. A location aware map application can make it easier for users to navigate to the library, and might contribute in lowering the barrier for visiting it.

The aim of the application that is to be developed is to guide a user from his given position to a library holding a document desired by him. An example is when the user is in the city and needs a specific book right away. A search performed using the prototype running on the mobile device, informs the user that the desired book resides in several of the university library departments. The user is able to select the nearest department with the given document. The system generates a map, indicating the last registered position of the user as well as the position of the selected library department. The system also queries the local bus company on how to get there. A route to the nearest bus stop with connections to the department is then displayed on the map, along with the time of the next departure. A more elaborative description of the functionality is available in chapter 7 where a walkthrough of the prototype is given.

3 Requirements

This chapter determines the requirements that have to be fulfilled in order for the application to offer the services described. These requirements consist of two parts; functional and non-functional requirements. The functional requirements describe the features available to the user, and how the user has to interact with the software to take advantage of these. The non-functional requirements are a requirement that do not directly affect the user's experience of the software. As a tool for generating the functional requirements, use cases are applied.

3.1 Use cases

As a tool for determining the functional requirements for the prototype, textual use cases are applied. Textual use cases consist of several cases, or scenarios [9]. Each case describes a functionality that the user could take advantage of. A series of steps are defined, to describe what a user would have to do. Each of these steps consists of two parts; what the user does and how the system responds to this action. The textual use describes the full functionality of the map-application described in the preliminary work, summarised in section 2.3 [1]. The purpose of the use cases is to facilitate a discussion on the functional requirements and to make sure that the developer and the representatives from UBiT has the same basic understanding of the application's targeted functionality.

Each use case contains a title that describes the case. In each case one or several actors are defined, where actors are users, external systems that interact with the system or the system itself. Pre-conditions are requirements that have to be true before the use case can start. Each use case contains a series of steps that defines the main success scenario. When these steps are completed, the case is finished.

Variations are used to describe what happens if the main success scenario fails. Each variation is identified by the number of the corresponding success step and a letter.

The use cases described in the following pages are:

1. **Start up:** The user starts up the application and his or hers position is registered.
2. **Locate document:** The user issues a search with a keyword. A corresponding document is selected and the user navigates to the nearest library department, using the application.
3. **Subject based search:** The user selects a subject and navigates to the nearest library with documents on the given subject.
4. **Free capacity:** The user wishes to use a computer and is shown to the nearest library with free capacity
5. **Local news:** the user is shown news from the area nearby his or hers current position
6. **Where is.** The user desires to navigate to a given object inside a library. A shelf map is shown, indicating the position of the user and the desired object.

Case 1: Start up		
Actors: Private user, “Trådløse Trondheim”, GPS system		
Pre-conditions: The user is within the borders of “Trådløse Trondheim”		
Step	Private User	System
1	Start the application	Retrieves location information from localisation service, e.g. GPS Variation: 1a
2	The user enters username and password	The system retrieves the user’s preferences.
3		The coordinates, username and time of registration is passed to the server.

Table 1 – Use case 1

Table 1 describes what takes place when the user starts up the application. The user starts up the localisation application, and the position of the mobile device is updated on the server

Case 2: Locate document		
Actor: Private user, Bus schedule service, BIBSYS, Google Maps		
Pre-conditions: The user wishes to locate a document, and to travel to the nearest library department holding it. The coordinates of the user is registered.		
Step	User	System
1	The user selects ”Navigate”	
2	The use selects ”Document search”	
3	The user performs a keyword search.	The system retrieves information on the documents. Variations: 3a, 3b, 3c
4	The user selects a book that he/she wishes to reserve.	The system reserves the document, and retrieves information on the corresponding library.
5	The user selects bus as the preferred mean of transportation.	A map is presented with instructions on how to get to the nearest bus stop, along with information on bus schedules”. Variations: 6a
6	The user is at the final bus-stop.	A new map with navigation instructions to the library is presented.
Variations:		
3a – The document is not available		The search interface is presented again. The user is informed that the document is not available
3b – The document is registered in		The user is presented with information on when the

the database, but not physically available at the library		document will be available.
3c – The document is already reserved		The user is informed of when the document will be available, and is offered to register his/hers interest. If the document is available at other libraries, the user is informed of this
5a – No bus available		The user is informed that no bus is available. A walking route is presented.

Table 2 – Use case 2

In Table 2 the use case of detecting a book and viewing a map with navigation instructions is described. The mobile has the browser running, and the user has navigated to the main menu of the application. The user selects to perform a keyword search, and selects a document. A map is then presented, and the user takes advantage of this to navigate to the library holding the document.

Case 3: Subject-based search		
Actor: Private user, BIBSYS, Google maps,		
Pre-conditions: The user wishes to retrieve a map with directions to a library with documents on a given subject. The coordinates of the user is registered.		
Step	User	System
1	The use selects “navigate”	The Navigation-menu is presented.
2	The use selects to perform a subject-based search, and chooses a subject from a list.	Information on libraries with books on the chosen subject is retrieved. The closest one is indicated.
3	The user chooses one of the presented libraries	A map with directions is presented

Table 3 – Use case 3

Table 3 describes how the user, by selecting a subject, can get navigation instructions to a library department containing books on the given subject. A map with directions is presented as in use case 2.

Case 4: Free capacity		
Actor: Private User, BIBSYS, Google Maps, Bus schedule service		
Pre-conditions: The user wishes to retrieve a map with directions to a library with free capacity on e.g. computers. The coordinates of the user is registered.		
Step	User	System
1	Select “Navigate”	The navigation menu is presented
2	Select “Capacity”	Several capacity criteria’s are presented
3	Choose the capacity criteria	Capacity information for each library is retrieved. If

		preferred libraries are previously indicated, these are prioritized. Variation: 3a
4	The user navigates using the map presented.	
Variations		
3a – No library with available resources		The user is informed that no library is available

Table 4 – Use case 4

Table 4 describes how the user can get to a library with available computers and printers, and other equipment.

Case 5: Local news		
Actor: Private user, ATEKST, Google Maps		
Pre-conditions: The user is in Trondheim and wishes to get the latest news for his/hers nearby area. The coordinates of the user is registered.		
Step	User	System
1	Select “Local news”	The system gets the preferences previously indicated by the user, and displays the local news in a map.
2	The user clicks on one of the headlines	The complete article is presented

Table 5 – Use case 5

Table 5 describes how a user can get news for the local area, based on predetermined preferences and the user’s location.

Case 6: Where is ...		
Actor: Private user, Google Maps, BIBSYS		
Pre-conditions: The user is near a library, and wishes to get a map with directions to an object inside the library.		
Step	User	System
1	The user selects “where is”	The system checks if the user has any stored preferences.
3	The user selects which object to locate	The system retrieves information on the available objects in the library, along with the local map. The map and the directions is presented
4	The user follows the static map with directions to the object	

Table 6 – Use case 6

When the user is inside a library, the “Where is” part of the program is enabled. As described in Table 6, it enables the user to determine the location of a document, the nearest printer, computer, etc.

The 6 use cases described here are the foundation for the functional- and non-functional requirements that are presented in the next sections.

3.2 Functional requirements

Based on the use cases presented in section 3.1 a set of functional- and non-functional requirements is created. An informal ranking of the use cases shows that use case 1 (Table 1) and use case 2 (Table 2) are the most interesting ones. The implementation of use case 1 ensures the location awareness of the application, which is one of the premises for the application. The implementation of use case 2 will create an application that shows a great deal of functionality in one single service. It will also require the communication with other, already existing, services such as BIBSYS and a bus schedule service. Based on the complete list of use cases, the functional requirements in Table 7 are created, as well as the non-functional requirements in Table 8. Each of the requirements is assigned a priority based on the use-case observations. These priorities will determine the functionality implemented in the prototype.

Table 7 describes a complete list of the functional requirements for the application. Along with each requirement is an ID, as well as a priority. The priority can be High (H), Medium (M) or Low (L). A high priority indicates that the implementation of the requirement is highly relevant for fulfilling the goals given in section 1.3. A medium or low priority indicates that a requirement can be excluded from this project due to time- or cost restrictions. See Appendix A and Appendix B for a use case point estimation of the time necessary to implement the different set of requirements.

ID	Functional requirement	Priority
C01	Let the user perform a search for documents using free text search.	H
C02	Display a map containing both the position of the user and the position of a library with a given document.	H
C03	In C02 a route should be displayed on the map, so that the user can navigate to the library using it.	H
C04	In C02 navigation to and from bus-stops, and information on bus schedules should be possible.	H
C05	Let the user determine what libraries that are holding documents on a given subject.	M
C06	Display a map containing both the position of the user and the position of a library with documents on a given subject.	M
C07	When at a library, let the user see a library map with the position of a selected document indicated.	M
C08	Let the user determine what libraries have capacity on equipment such as printers, computers and rooms.	L
C09	Display a map containing both the position of the user and a library with capacity on printers, computers and rooms.	L
C10	When at a library, let the user see a library map, with a path to selectable equipment, such as printers, books etc.	L
C11	The application must support reservation of books.	L
C12	The user must be able to view local news, based on the user's position. Local can be defined as either part of town or on the city as total.	L
C13	The local news in C12 must be displayed in a map.	L

Table 7 - Functional Requirements

The functional requirements in Table 7 are listed according to priority. The ones that are indicated as high are outlined with a solid border. The first requirement, C01, demands that the applications enables user to locate documents using some sort of free text search method. As searching for documents is one of the fundamental functionalities of a library, and as retrieval of information from the library database is one of the premises for this application, this requirement gets a high priority.

C02 enables the user to view either him- or herself in a map, together with the position of a library in a map. By using a map and presenting the user with his or hers position, reaching the library can be simplified.

C03 and C04 give the user information on what bus to take to the library, and how to navigate to and from the nearest bus stop. This enables the integration of information from many

different resources, as required by the second goal in section 1.3, and can increase the value of the application from a user's point of view. Therefore C03 and C04 get a high priority.

The subject based search in requirement C05 enables users to locate a library based on the main subjects of the library department. This is practical in the case where a user has not decided upon a specific document, but wants to travel to a library with documents on a given subject. Again, a map is useful for presenting this sort of information, as defined in C06. These requirements have a medium priority because this subject-information is not available in the current library database. The C07 requirement, displaying a map on library-level with a selected document indicated, gets a medium priority as this is not one of the functionalities necessary to demonstrate the functionality of the prototype.

Determining the amount of available printers, computers and rooms is a nice-to-have function. It would increase the value of the application, but the implementation requires a large amount of work. It may prove valuable at a later time, but no means are currently available to measure these variables, so C08, C09 and C10 get a low priority.

Today the reservation of books through BIBSYS is not available. The implementation of C11 requires a change in the current structure of the reservation system of the libraries, and this requirement gets a low priority.

As the presentation of local news falls somewhat outside the traditional field of a library, requirements C12 and C13 get a low priority.

3.3 Non-functional requirements

Table 8 describes a complete list of the non-functional requirements for the application. As for the functional requirements, High (H), Medium (M) and Low (L) priorities are used to rank these. The requirements inside the heavy border are the ones with a high priority.

ID	Non-functional requirement	Priority
D01	The system must take advantage of an existing map-service	H
D02	The system must communicate with BIBSYS	H
D03	The system must retrieve bus-schedules from team-trafikk	H
D04	The application for the mobile device must be platform-independent	H
D05	The system must take advantage of the user's location.	H
D06	The location service must use GPS to locate the user	H
D07	The system must have a module-based design to facilitate additional functionality at a later stage	M
D08	Responsiveness	L
D09	Ease-of-use	L
D10	The navigational directions must be updated as the user moves, at an intersection level.	L
D11	The location service must use GeoPos to locate the user	L

Table 8 - Non-functional requirements

The use of map can simplify the experience for the end user. The prototype must be able to demonstrate communication with existing services, therefore D01, D02 and D03 gets high priorities. As it is desirable that as many users as possible are able to use the application with as little effort as possible, the platform-independence requirement D04 also gets a high

priority. D05, location awareness, is given as a premise for the project, and is given a high priority. D06 requires that GPS is used to determine the position of the user.

In requirement D07, module based design gets a medium priority. It is desirable that the application gets a module based design so that new functionality easily can be added to it at a later stage. However, as this is a prototype, the focus of the design does not have to be on this, so it gets a medium priority.

As the product resulting from this development is to be a prototype, aspects such as responsiveness and usability gets a low priority (D08, D09). A very frequent update of the user's current position and the navigational steps necessary to reach the destination is not needed to demonstrate the functionality (D10). There are examples of applications that are taking advantage of GeoPos, and the quality of this service is not satisfactory enough for it to be used as the primary localisation technique [1]. At a later stage, it can be added as a backup to be used if GPS is not available (D11).

Now the requirements that were formed based on the use cases described in the previous section are ready. Each of them has been elaborated on, and assigned a priority. Next is to decide how these requirements can be met.

4 Available technologies

This chapter describes the relevant technologies that can be used to fulfil the requirements determined in the previous chapter. It describes different operative systems for mobile phones, a selection of programming languages available for the server-side application, some location technologies available, as well as the different external services that could be beneficial to take advantage of in such an application. The descriptions in this chapter will function as a platform for design- and implementation-decisions that are to be presented in chapter 5.

4.1 Operating systems

There exist a large number of mobile devices, and each brand supports one or several operating systems. Each operating system put restrictions on which programming languages that can be used, and how the local resources can be accessed. Some platforms only supports a subset of the languages available, and some cross-platform languages are not able to fully exploit the possibilities of each platform.

In this section some of the most popular operating systems for mobile phones are described, as well as some of the most popular development languages.

Windows mobile

Windows mobile is an operating system for mobile phones that is designed to look and behave in somewhat the same manner as Windows for desktops. This simplifies the manufacturing of new applications for developers, as it is based on the WIN32 API. In addition many users will find the user interface easy to use because of its similarity to traditional Windows for desktops.

An analysis performed by Gartner in August 2008 shows that Windows Mobile was pre-installed on 12 % of all smart phones sold in the second quarter for 2008 [10]. This is a 0.5 per cent increase compared to the first quarter.

The 6.1 version of the Windows Mobile OS is currently the one most used. The 6.5 version was released during spring 2009. One of the most criticised aspects of Windows Mobile is the usability for touch screen cell phones, and the size of menus, icons and text. It is expected that this update will improve the touch-screen abilities of the operative system by addressing these issues.

Windows Mobile 7 is expected to reach market in 2010. This update can further increase the functionality of the OS, and better take advantage of the touch screen.

Symbian OS

In August 2008 the market share for Symbian was 57.1 per cent, making it by far the most sold operative system on smart phones [10]. However, this is a reduction of 8.5% from the first quarter of 2008. The leading manufacturer of Symbian phones is Nokia, with their software platform S60. The latest edition Symbian S60 v 6.5 is the first with support for touch screens. S60 gives access to device specific features such as the GPS through Flash Lite.

iPhone OS

The latest version of iPhone OS is 2.1.1. It is based on Apple's OS X, and is deployed on iPhone and iPhone 3G. Developers are highly encouraged, and somewhat forced, to develop applications that do not compete with applications already provided by Apple. Both a SDK and an API is available, and the number of applications developed by users is constantly growing. As of today applications can only be legally installed on the iPhone through Apples

App Store, and only applications that are approved by Apple can be submitted to this App Store. iPhone 3.0 is available for download 17th June 2009.

Android

Android is an operating system for mobile devices developed by Google[11]. It is based on a Linux kernel, and it is expected that user contributed software will be a large part of the operating system. This is accomplished by providing a SDK and API, facilitating Java development. Also C development is possible, although not officially supported by Google.

Only a few phones are currently sold with the Android OS pre-installed, such as the HTC Dream[12], but users have been able to perform aftermarket installation on other phones.

4.2 Programming Languages

As well as deciding upon which operative systems the applications will run on, it is necessary to decide what programming language to use. This section describes the most common programming languages used on mobile phones today.

Java

JME, previously known as J2ME, is a reduced version of Java Standard Edition (J2SE) designed to operate on devices with “limited memory, display and power capacity” [13]. It is partially platform independent, meaning that the devices the application is installed on has to support the profiles and configurations that the application was designed for [14]. Therefore an application design for a mobile phone can not be expected to run on, for instance, a java-supporting satellite tuner. A profile defines the API’s and interfaces available on that device, while a configuration defines how large a subset of J2SE is available on the device.

When an application is designed for a specific mobile device, the application may run on any device in the same category. JME is a subset of J2SE, and development on this platform is therefore very much like developing for J2SE. Java has traditionally been given a reputation for being slow, but whether this is still true is a subject widely discussed. Development of user interfaces for JME applications can be cumbersome and time consuming.

C#.NET

.Net Compact is a framework built for small devices, and facilitates communications through standardized protocols such as Simple Object Access Protocol (SOAP) and Web Service Description Language (WSDL) [15]. It runs on several Windows-based operative systems, such as Windows Mobile, by taking advantage of a Platform Adaptation Layer (PAL). Both C#.NET and VB.NET is supported.

.NET development is based on taking advantage of already existing standards. Communication with device specific technologies can be uncomplicated, and user interface development is simple drag and drop using Microsoft Visual Studio. Development on this platform can be less time consuming than the alternatives, but also less platform-independent.

C++

For software to run on the Symbian platform, Java and C++ are the most used alternatives. When it comes to appearance and user interface, different solutions exist based on the producer of the device.

4.3 Location technologies

There are several location technologies available for mobile devices. This section describes the outdoor localisation technologies that are most used; GPS, WiFi-based and GSM-based.

GPS

GPS is integrated into an increasing number of mobile devices[1]. The positioning service is quick when first connected, has high accuracy, and is free to use. The reliability is also high, as long as a free view to the sky is ensured. However, there is still a large number of mobile devices that does not have this technology installed. Also, in city centres, there is the possibility of ending up in signal shadows. There have been some reports on the possible lack of founding for the GPS system [16].

WiFi-based

When the user is indoors, or the view to the sky is blocked by buildings in narrow city streets, the connection to the GPS-satellites most likely will be lost. If this happens, positioning in wireless networks can be used. In the Metropolitan Area Network (MAN) "Trådløse Trondheim" the positioning service "GeoPos" is available [3]. This positioning service is less reliable, less accurate and slower than GPS. However it is available within some buildings, and does not require a GPS to be installed on the mobile device. Instead a WiFi card is required.

GSM-based

All mobile phones are able to access the GSM network if it runs at the same frequency as the phone was designed for. There exist several methods for pinpointing the position of the mobile device, and the most accurate ones has a resolution of approximately 65 meters [17].

4.4 External services

The prototype will retrieve information from many different services. This section provides a short description of the most relevant ones. Which of these will be used in the prototype is decided upon in the design considerations, section, 5.2.

Google Maps

Google offers an API that lets developers include maps in their web pages using JavaScript [18]. Through Events, Controls, Overlays and Services the map can be configured to serve many different aspects, one of which is displaying the route to and from different points of interest.

Bing maps for enterprise

Microsoft has for some time hosted the Live Maps service. This is now known as the Bing maps for enterprise (previously also known as Microsoft Virtual Earth). This service enables web users to search in online maps. The Bing maps exposes a series of services that allow developers to take advantage of location and local search features in their applications[19]. The Web services are available for three platforms; Silverlight, .Net and mobile applications. This service is currently in *Community Technical Preview*, meaning that one have to apply for access to the SDK [20]. The documentation does not appear to be complete.

Yahoo! maps

Yahoo maps is a map web service that provides Flash-, Ajax- and Map Image APIs[21]. This enables the generation of maps with or without of directional instructions, as well as

geocoding services. The design and functionality of Yahoo! Maps services are highly comparable to the Google Maps API.

BIBSYS

BIBSYS contain information about libraries and documents that can be used in the application. One way to access this information is through a web service [22] provided by BIBSYS, the BIBSYS SRU[23]. By passing a valid CQL query to this address an easily parsable XML result is returned.

Bus schedules service

On their web page Team Trafikk offers natural-text based search for bus schedules [24]. The result of this search is also in natural-text. To be able to communicate with this service one of two things can be done. Directly accessing the database containing the bus-schedules for Trondheim is one of them. The other is to perform a scraping¹ of the web page. When a scraping is performed the response to a query is read directly of the web page. The result is information on the bus-stops nearest to the user and the selected target, as well as information on departures between these bus-stops.

Amazon image service

Amazon has the book covers for a large part of the books they sell available as images on their servers. By passing ISBN² code of the document in an URL, the image is created if the document is present in their database. This service can be used to present the users with the cover of a book they search for, in addition to the textual information. If the user learns what the document looks like, this can simplify the task of locating the document when at the library.

LibraryThing image service

LibraryThing is an alternative to the Amazon image service[25]. It is an open community database that contains cover images of more than 39 million books, and that let their users contribute by uploading book covers. In January 2009 1 million covers were available by passing the ISBN of the document to their servers. In June 2009 this number has raised to more than 1.4 million cover images.

¹ A technique for extracting information from web pages.

² International Standard Book Number

5 Design

Based on the requirements and their priorities in chapter 3, this chapter presents the design of the prototype. By elaborating on some important design considerations the overall design, as well as class- and sequence diagrams, is constructed. The relevant technologies and services are discussed, and the ones to be used in the prototype are determined.

5.1 Top-level design

Figure 2 describes the design of the prototype. The red border indicates the modules that are to be implemented in the prototype.

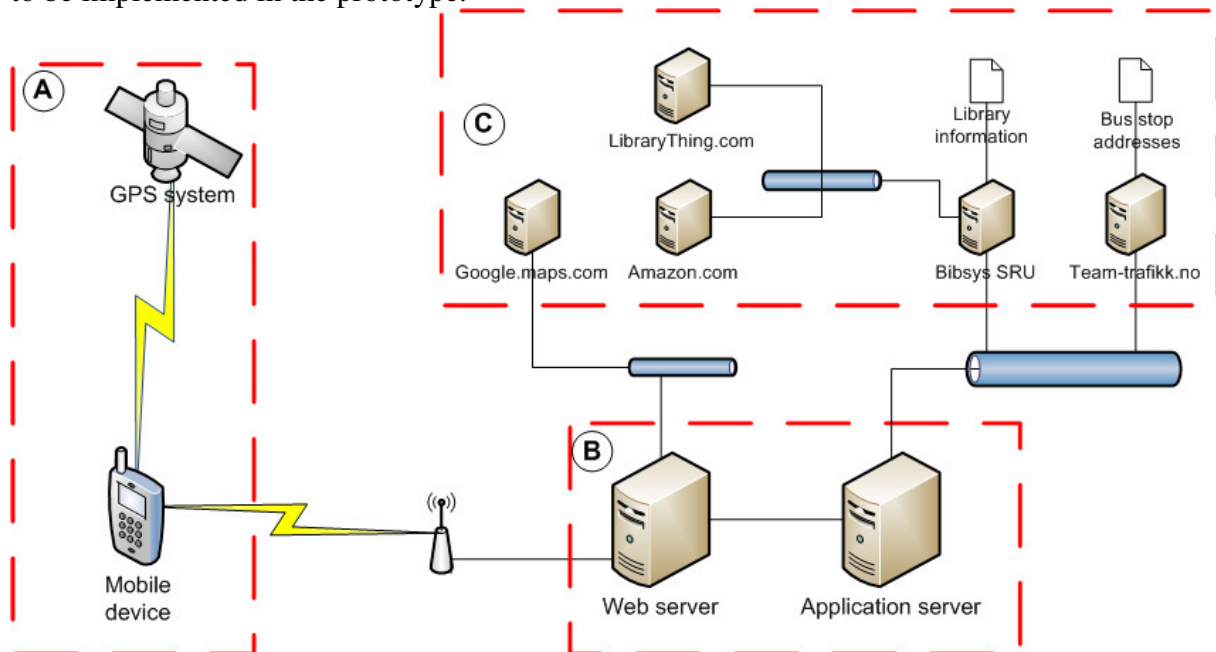


Figure 2. The overall design with three modules.
The mobile application (A), the server-side application (B) and adapters and services (C).
The dotted line indicates the prototype border.

The prototype consists of three modules as shown in Figure 2. Module A is the application running on the mobile device, handling communication with the localisation services. Module B is the server-side application, consisting of the Web server and the applications server. The application server responds to requests issued by the web server, and returns information retrieved from numerous local and external services. Module C consists of external services. In order for a service to be implemented, an adapter has to be created at the application server. This adapter handles all communication with the external service. If the source of information is to be replaced by another, all that has to be changed is the corresponding adapter. In addition the web server communicates directly with the Google Maps server. The next section discusses the design of the prototype.

5.2 Design considerations

In this section some important design aspects are discussed. It is necessary to determine whether the prototype should run solely on the mobile device, or if the computation is to be performed on a remote server. In the previous chapter a large variety of external service was presented, and it is determined which of these that are to be used. In Appendix A it is calculated that development of all the use cases would take approximately 19 weeks. This is not feasible in the span of this project, but by reducing the design, based on the priorities of the functional- and non functional requirements the development process is reduced to 9 weeks, as shown in Appendix B.

5.2.1 Distributed vs. local application

One of the aspects that should be considered is whether the application is to run in full on the local device, or fully on a remote server. Running the complete application locally will require more processor capabilities on the device. In addition it could reduce the level of portability. For these reasons it is desirable to deploy the application on a remote server. If this server is a web server, the user can navigate to a web page using a standard web-browser on the mobile device. However, to be able to access data from the localisation device on the unit, some form of local application is required.

5.2.2 Platform and programming language

Section 4.1 describes several different platforms, and in section 4.2 some programming languages are presented. In the non-functional requirements specified in section 3.3, requirement D04 states that UBiT has a high desire for a platform-independent application running on the mobile device. Both Opera and Firefox are developing browsers that are able to access the GPS device or any other localisation device. As a consequence of the technology emerging, it is reasonable to state that a mobile application will not be a part of a future application. Because .Net enables easy access to the GPS on the mobile device, and since the developer is using windows mobile with an internal GPS device, C#.Net will be used in the prototype. This has no consequence for the development or execution of the server, as this simply reads the users GPS coordinates from a file. It can appear as this decision directly interferes with the functional requirement of platform independence, but the mobile application is made for this prototype only and will not be part of the complete application.

5.2.3 Location service

The design of the application running on the server defines that the coordinates of the user is read from a text file on the server. This determines that the server side of the application is able to run independent of which localisation service that is used, as long as the coordinates are written to this file.

Localisation in the GSM network does not have a high enough resolution to support the functionality required by the prototype, and is therefore deemed unsuitable.

As several projects at NTNU have already attempted to take advantage of GeoPos in their work with map-based applications, it is demonstrated that this service pinpoint the user's position with a lower resolution and a higher delay than GPS [26]. The use of GeoPos as localisation service is therefore excluded from this project.

GPS signals are normally not available indoors, and it is therefore likely that it will not be possible to pin point the user's location when he or she is indoors. In the prototype this issue

can be solved by using the last registered position of the user before the building was entered. 38% of all smart phones sold in Europe have GPS installed[27]. As the use of GPS is one of the non-functional requirements it is desirable to show that the prototype functions with this location technology. The prototype will therefore focus on the use of GPS a location service.

5.2.4 Bus schedule service

The prototype must show that integration with external services is possible. One such service is Team-trafikk.no's *bussorakel*. The prototype will communicate with this service to extract bus schedules. An alternative is to access the bus schedule database directly. As this will not simplify the process of retrieving bus schedules, nor demonstrate the possibilities of exploiting existing services, this is not an option. Team-trafikk is the only bus company in the city of Trondheim, and their service *bussorakel* is selected as source for the local bus-schedules.

5.2.5 Map service

As map service for the application, Google Maps is selected. It provides a JavaScript interface enabling the generation of maps with directions displayed in them. When compared to Yahoo! Maps, it is obvious that these two have a lot in common. However, the developer has previous experience with the Google Maps API. As the timeframe for the thesis is limited, Google Maps is preferred in this case. Another alternative is Bing Enterprise Maps. That project is however in an early phase of development, and it would be quite time consuming to get an overview of the possibilities offered by this service. Google Maps is preferred over Bing Enterprise maps on the same basis as for Yahoo! Maps; the timeframe.

5.2.6 Book-cover data source

Both Amazon and LibraryThing provides free download of their book-covers. However, none of them are adequate for the amount of documents present in the BIBSYS library. The cover image will first be attempted downloaded from the servers at Amazon.com. If the image is not available here, the database at LibraryThing will be used.

5.2.7 Additional data sources

As the prototype will incorporate information from multiple services, it is necessary to perform some sort of data validation or conversion when data from one source is applied to another. This will in some cases require the use of additional data sources.

When the bus service is queried on a bus-schedule, the name of the bus stop nearest to the position of the user and the name of the bus stop nearest to the library are returned. However, in order to indicate these points in a Google map, the addresses of them must be retrieved. Therefore a local data file is created, where the addresses of the bus stops are registered. This registration is done manually by the developer, and in order to support additional bus stops this file will have to be extended. The server manages this file, which enables conversion from bus-stop names to bus-stop addresses.

When the user selects a document the corresponding library department is indicated in the map. When a book is queried for, the result contains an id that represents a given department. In order to plot the library department in the map, its address is retrieved. Again, the server

holds a manually generated file that converts from library id to address and coordinates. This file contains the address of each library department elaborative descriptions on how to get to the department, coordinates, phone numbers and so on. This information is also shown in the map.

5.2.8 Multiple users

Implementation of user login and settings will not contribute to the functionality in the demonstrating prototype, nor is it a requirement. Therefore the prototype will not handle multiple users and their preferences.

5.3 Class Diagrams

In section 5.1 the overall design of the system is established. Based on this design and decisions made in the previous section, the next step is to elaborate on the different components of the system.

5.3.1 Mobile application design

Figure 3 describes the design of the application that runs on the mobile device. The application is responsible for fetching the coordinates of the user, and to register them on the server. When the application starts up, the Form described in Figure 3 is loaded. It then registers the `gps_LocationChanged` and the `gps_DeviceStateChanged` methods as event handlers. When the location changes, the `gpsLocationChanged` method calls the `updateData` method that connects to the server and passes the parameters using an `HttpWebRequest`. On the server these parameters are written to a local file.

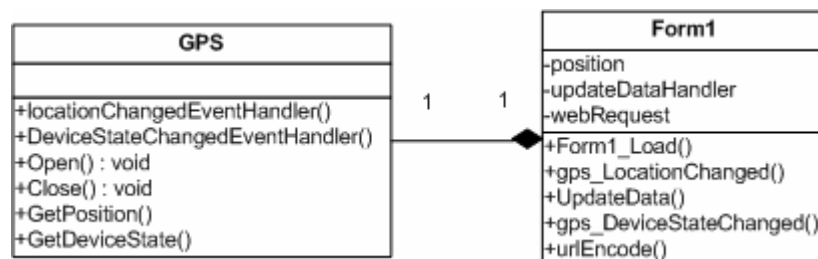


Figure 3. Class diagram of the mobile application in C#

5.3.2 Server side application design

A class diagram for the server side of the application is shown in Figure 4.

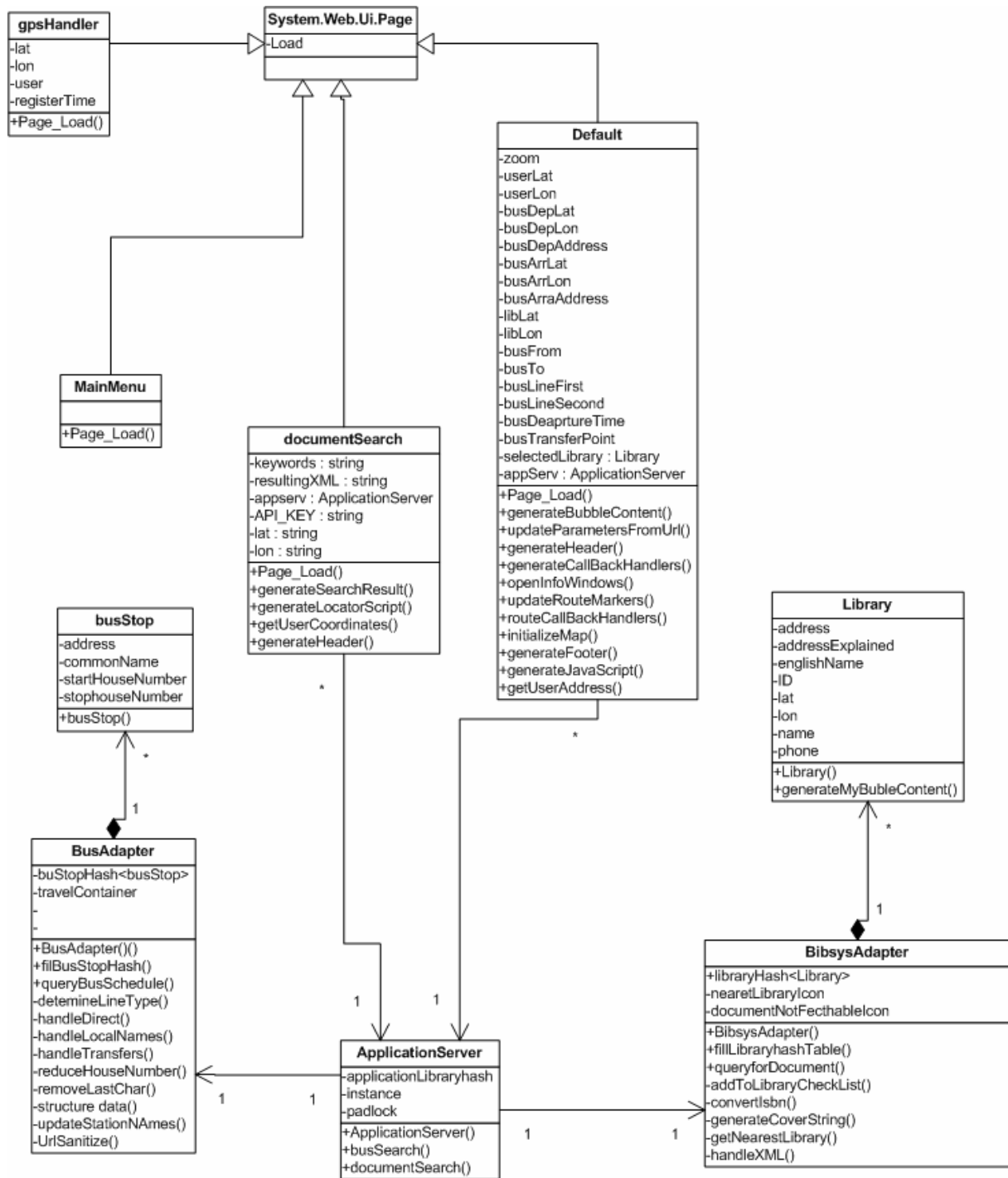


Figure 4. Class diagram - Server Application

In the bottom middle is the Application Server class. This is implemented in the singleton pattern, so it is only one instance of it at all times. The first time the instance of this class is requested, it generates the BibsysAdapter object and the BusAdapter object, and calls methods on them to fill up their respective hash map objects. Those methods read the required data in from local files. For each line in the library information file, an instance of the Library class is created and added to the hash map. The BibsysAdapter holds a collection of Libraries, and the BusAdapter holds a collection of bus stops.

At the top of the figure there are four classes that extend the Page class. These are the code behind files for the asp.net web pages. Each of these web pages implements the Page_Load

function that is registered as handler for the Load event in the Page object, so that when a page is requested, this method is automatically called.

The Default class is responsible for generating the map. It holds the variables necessary for this, and it has methods that each produces a part of the JavaScript necessary for this generation. The gpsHandler accept requests from the application running on the mobile device, and writes the user coordinates to a local file.

The next section describes the correlation and communication between the classes by the use of sequence diagrams

5.4 Sequence Diagrams

This section contains sequence diagrams representing use case 1 and 2 as described in chapter 3. The diagrams show the objects, and the messages that are exchanged between them, in each of the use cases. The purpose of these diagrams is to clarify how the system operates in the given cases, and in which sequence the interaction occurs.

The first sequence diagram, in Figure 5, describes the first use case where the GPS position of the mobile devices is determined. Whenever the GPS registers that the location of the user has changed, it notifies the positionChangedEventHandler in the Form1 class, which is the UpdateDataHandler. This method calls the updateData method on itself, which in turn creates an httpGet request, passing the new latitude and longitude of the device to the application server. On the server the new data is written to the local GPS file. Whenever the position of the user changes this sequence is repeated.

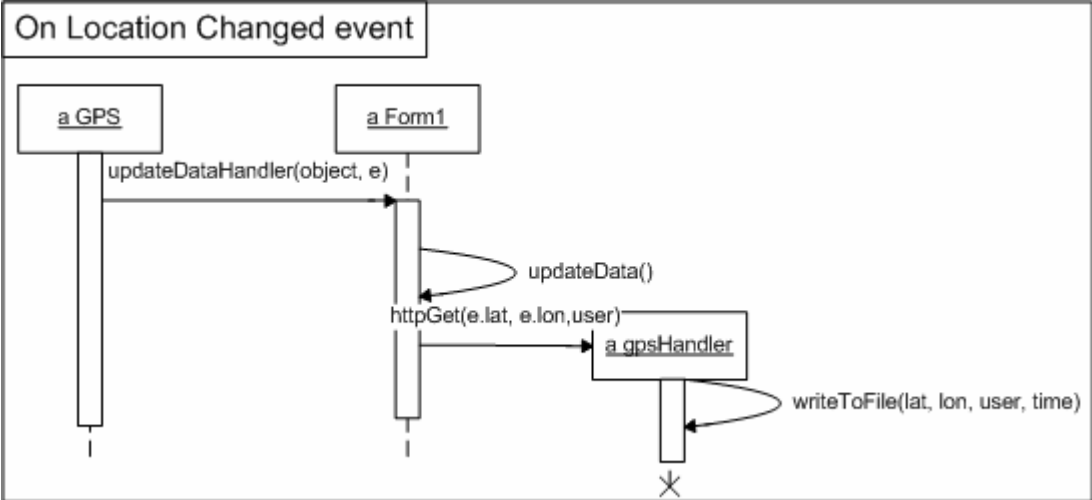


Figure 5. Sequence diagram - Start-up use case

The next sequence diagrams, Figure 6 and Figure 7, describe the process of retrieving a map with instruction on how to get to the library department holding a given book. Before these sequences can take place, the sequence in Figure 5 has to be performed at least once in order for the coordinates of the user to be registered at the server.

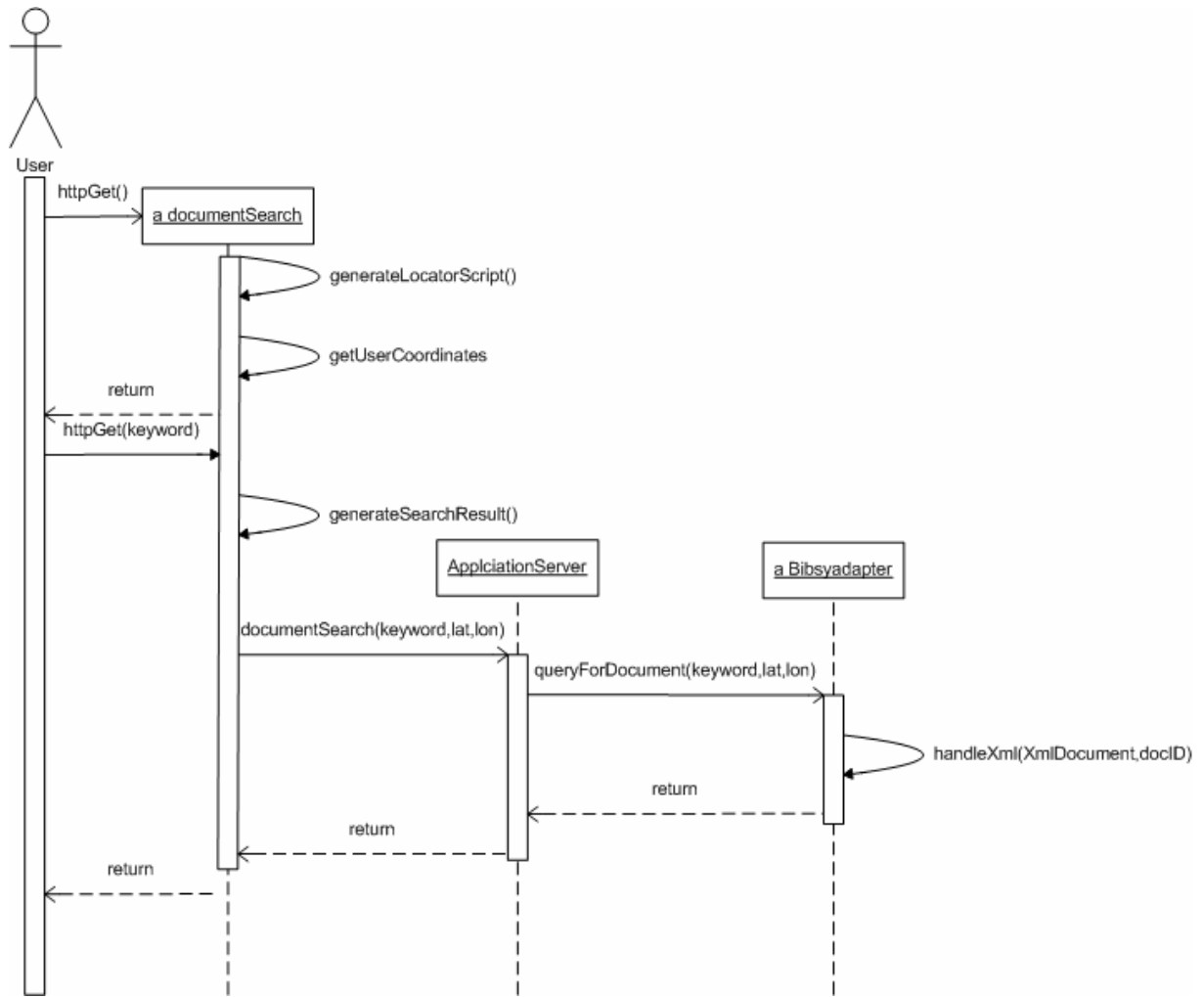


Figure 6. Sequence diagram - Locate document part 1

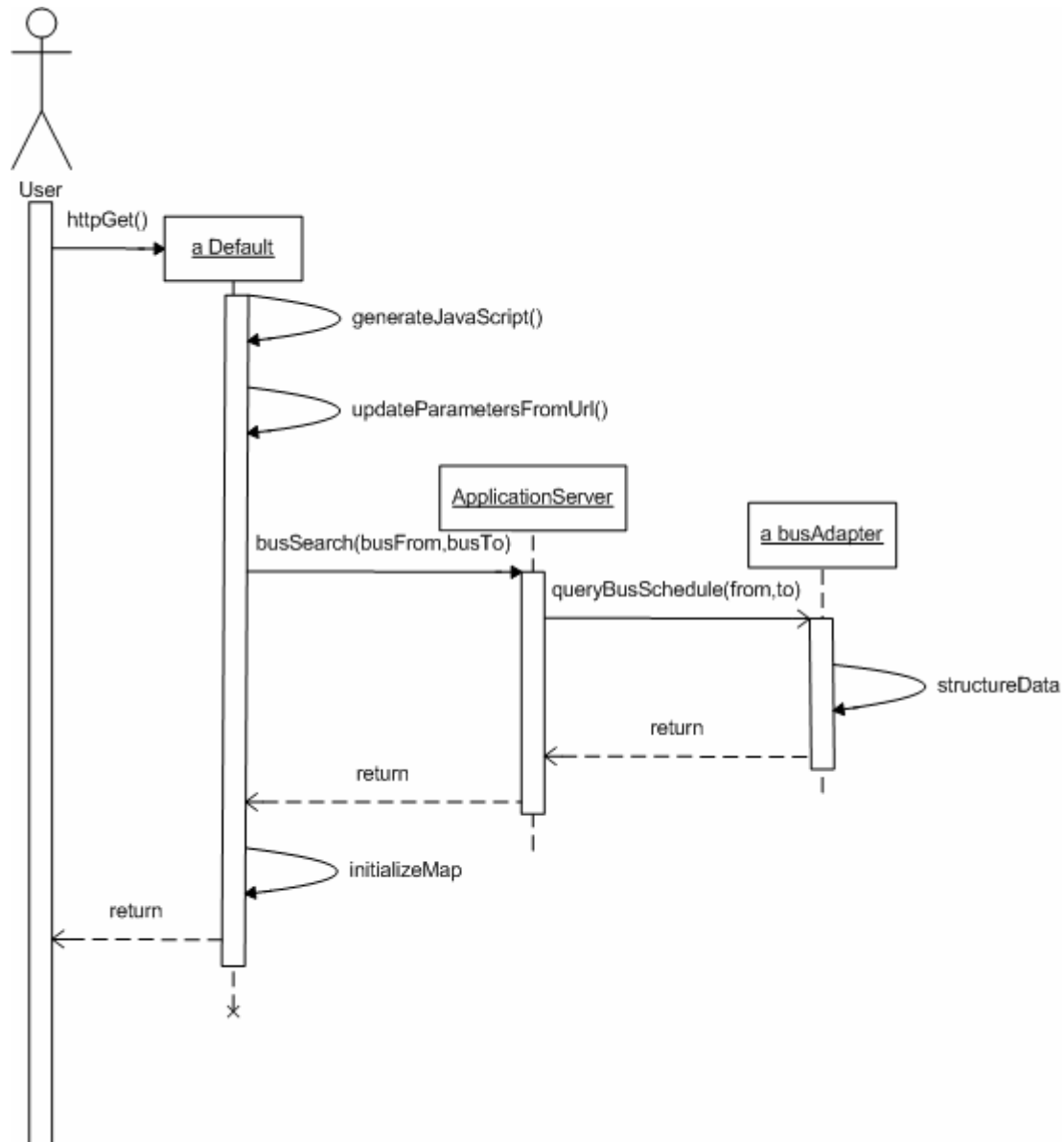


Figure 7. Sequence diagram - Locate document part 2

The first part, Figure 6, describes how a query is passed along to the BISYS service. When the user selects to perform a document search, the documentSearch class generates a JavaScript whose purpose is to retrieve the user coordinates. This is done by calling the getUserCoordinates method which reads the user coordinates from the local GPS file as it was generated in Figure 5. These coordinates are then translated into textual address by performing calls on the Google maps server. The calls on external servers are omitted from the figures for simplicity. When the control returns to the user, he is presented with a search field. A valid CQL query, e.g. a keyword, is entered and a new httpGet call is performed on the documentSearch class. Now the search result is generated. This is performed by calling the documentSearch method on the application server, passing along the keyword and the latitude and longitude of the user retrieved in the previous step. The application server queries the BibsysAdapter, passing along the parameters. This adapter issues a request to the external

BIBSYS SRU service, and parses the returned XML. It downloads cover images if they are available. The result is returned to the user.

Next, the sequence in Figure 7 takes place. By selecting one of the returned book results, an `httpGet` request is performed on the Default page. This object performs the `generateJavaScript` operation on itself. As a result, the parameters necessary for generating the map is retrieved from the URL and from the Session object. Then the `busSearch` call is performed on the application server, with the address of the user and the address of the selected library passed along. The application server is then responsible for passing the query along to the `BusAdapter`, which issues an `Http Request` on the remote bus schedule service. It structures the resulting data and returns the route information. Finally the Default web page prints the JavaScript for generating the complete map, and the control is returned to the user who is free to navigate the map as soon as the browser has performed the JavaScript calls required.

6 Implementation

This chapter describes how the prototype is implemented. The mobile application that logs the position of the user is presented. Then the implementation of the server side of the application, the web server, application server and adapters, is described

6.1 Mobile application

The mobile application is indicated as module A in the design shown in Figure 2. In the design section of this document it was elaborated on whether to use a distributed or local application. Due to the limited resources of mobile devices, and the use of JavaScript, it was determined that the application can not run completely on the local device. Due to current access restrictions on the local resources, the application can not run entirely remote.

The newest beta versions of desktop browsers such as Opera[28] and Firefox[29], support the Geolocation API specification provided by W3C[30]. This enables the browser to access the local device's localization equipment such as GPS through JavaScript. This is currently available for desktop browsers in beta versions only, and it is therefore expected to take some time before it can be used in mobile browsers as it is desired in this case.

The solution for the prototype is to develop a small local application that transmits the coordinates of the user to the server at given intervals.

This application is for ease of testing developed as a Windows Mobile application, using C#.Net.

The application listens to the local GPS device. If the position changes, it is transmitted to the server using an http-connection to an asp.net page at the server (gpsHandler.aspx). This asp.net page writes the passed parameters to a local file, gps.txt, logging the time of registration as well as the username of the user of the application.

6.2 Server side application and adapters

The server side of the application consist of the Web server that generates the web pages the user visits, the application server that ensures communication between the web server and the adapters, and the adapters that fetches information from the external services.

Web pages

The user is able to interact with the application using a web browser. The web pages are hosted by the web server in module B in Figure 2. Most mobile browsers on the market do not render JavaScript in accordance with the full standard. As a result of this the latest beta version of Opera (Opera Mobile 9.5 beta) that does render the Google Maps correctly, has been used during the development of the application[31]. It is reasonable to expect that mobile browsers in general will support the full JavaScript specification within short time.

When the user navigates to the start page of the application (mainmenu.aspx), the user is allowed to select "Search for documents". When this link is clicked, the user is redirected to search.aspx and the search interface is presented as shown in Figure 8. The server side application then prepares for queries against the BIBSYS SRU service.



The image shows a search interface within a rounded rectangular frame. On the left, the text "Search:" is followed by a text input field. To the right of the input field is a blue button with the text "Query" in white.

Figure 8. Search interface

At the same time the address of the user is determined by executing JavaScript code for communicating with the Google server. By using the coordinates provided by the application running on the user's mobile device, an inverse geocoding as shown in Listing 1 is performed. This example demonstrates one of the challenges with inconsistency between different data sources, as comma sign in decimal numbers has to be replaced by a period before they can be passed to Google for processing. The address is then stored and passed on to subsequent site-requests. This implies that if the user changes location, this page has to be loaded again for the address values stored in the application to be updated.

```

97.  var geoCoder = new GClientGeocoder();
98.      var latLng = new GLatLng(" +
99.          lat.Replace(',', '.', '.') + "," +
100.         lon.Replace(',', '.', '.') + @");
101.         address = geoCoder.getLocations(latLng,
102.             writeAddress);

```

Listing 1. Inverse Geocoding

Application server

The application server class handles communication between the web pages and the adapters providing the application with information, and is part of Module B in Figure 2. The application server class is responsible for instantiating and holding one instance of each of the adapters, and are able to perform operations on these. These operations generate information that is passed back to the web site that requested the information.

BibsysAdapter

The BibsysAdapter is located in module C in Figure 2. When the user submits a CQL query it is passed along to the BibsysAdapter. It communicates with the BIBSYS SRU, which is a service that accepts a query to be passed as parameters in an URL [23]. The keywords are then processed as a query against the BIBSYS database, and the result is displayed as an xml document on the web site. This xml document contains MARC information for each of the documents [5]. The BibsysAdapter parses the xml document using an XPathNavigator. The intention of using this kind of parser, as opposed to e.g. XmlReader, is that the XPathNavigator allows for non-sequential reading. This is desirable when information in one part of the document has to be read before it can be decided if another part is to be read later on. One example is that it is necessary to read the ISBN number of a document in order to fetch its cover image before the rest of the node is read. Also, the XPathNavigator is able to read the Xml document directly from a stream, as the one generated when fetching the result of the HTTP request. An example of a record in the xml document is shown in Listing 2. Here is only the information that is read by the BibsysAdapter presented, but as the XPathNavigator read through each of the nodes, all the information in the MARC record can be harvested.

```

1  <marcxml:record xmlns:marcxml="http://www.loc.gov/MARC21/slim">
2  marcxml:datafield tag="020" ind1=" " ind2=" ">
   <marcxml:subfield code="a">
       1-57851-558-0(ib.)
   </marcxml:subfield>
 </marcxml:datafield>
3  <marcxml:datafield tag="245" ind1="0" ind2=" ">
   <marcxml:subfield code="a">
       The digital enterprise
   </marcxml:subfield>

```

```

    <marcxml:subfield code="b">
        how to reshape your business for a connected world
    </marcxml:subfield>
    <marcxml:subfield code="c">
        edited with an introduction by Nicholas G. Carr
    </marcxml:subfield>
</marcxml:datafield>
4 <marcxml:datafield tag="852" ind1=" " ind2=" ">
    <marcxml:subfield code="a">
        TEK/IØT
    </marcxml:subfield>
    <marcxml:subfield code="z">
        (Ikke fjernlån)
    </marcxml:subfield>
</marcxml:datafield>
5 <marcxml:datafield tag="852" ind1=" " ind2=" ">
    <marcxml:subfield code="a">
        UBIS
    </marcxml:subfield>
</marcxml:datafield>
</marcxml:record>

```

Listing 2. A MARC record in XML format

Each entry in the xml document is parsed, and a selection of the data is stored, such as the library department holding a copy of the document (Node 4 and 5, Listing 2), the ISBN number (Node 2, Listing 2), the availability of the document (Node 4, Listing 2) and so on. The “Ikke fjernlån” value in Node 4, Listing 2, indicates that it is not allowed to remove the document from the library.

The first time the application runs the addresses of the library departments are read from a pre-generated local file and stored in a hash table. Later, if the library department exists in the hash table, the ID of the library is added as a hyperlink to the library name, as shown in Figure 9. “DORA” and “TEK/IDI” represents library departments of UBiT, and their names appear as hyperlinks. “NITH”, “UMN/INF”, “HIG” and “MF” are not library departments of UBiT, and are therefore not hyperlinked.

🔒 = Can not be taken away from the library ⚠️ = The library nearest to you holding the document




	Web database primer plus connect your database to the World Wide Web using HTML, CGI and Java Piroz Mohseni DORA ⚠️
	Essential Java style patterns for implementation Jeff Langr NITH TEK/DI ⚠️ 🔒 UMN/INF
	Java database programming Brian Jepson DORA ⚠️ HIG MF

Figure 9. Partial BIBSYS SRU query result for keyword "Java"

Some documents are not allowed to take out of the library. This is shown by a small padlock next to the library link, as shown in Figure 9. This indicates that the “Ikke fjernlån” property is present for this record, as shown in node 4, Listing 2. For each book, the library department that is nearest to the user is indicated by a yellow marker. This information is created by reading the latitude and the longitude of each of the coordinates for the user and each library that resides in Trondheim. The property of “nearness” is based on Pythagoras' theorem, and is derived by calculating the square distance between the points in each of the dimensions. Other measures could have been used, such as shortest travel time.

Further information on the availability of the document, if it is on shelf and so on, is not included because it is not available through the MARC code (Listing 2).

When the user has decided upon a document, he clicks the link of the desired library department and is redirected to the map.

BusAdapter

The BusAdapter is located in Module B in Figure 2. When the user has selected a library department, a query is generated that requests the next bus leaving from the position of the user to the address of the department. The addresses are generated by passing coordinates to the Google Maps Server. This requires some pre-processing before it can be used directly in the query. One example is the house numbers. As the GPS system has an accuracy of approximately 30 meters, Google Maps returns a series of house numbers, e.g. “Nordre gate 1-25”. This is not a deterministic street name that can be passed to team-trafikk, so the last part of the street number is removed, resulting in “Nordre gate 1” being passed to the bus-query. An alternative solution could be to take the average of the two numbers, placing the user in the middle of the street section

The result returned from the bus service is a natural text paragraph. Figure 10 shows what this result looks like when it is viewed on the bus company's webpage, with the numbers 1-4 added prior to each sentence.

Spørsmål :

next bus from høgskoleringen 5 to Maskinistgata 1

Spør >>

- 1 The station nearest to Maskinistgata 1 is Strandveien .
- 2 The station nearest to Høgskoleringen 5 is Gløshaugen Nord .
- 3 Bus 5 goes from Gløshaugen Nord at 10:10 am to Dronningens gate, holdeplass D2 at 10:18 am and bus 9 goes from Munkegata, holdeplass M5 at 10:23 am to Strandveien at 10:28 am .
- 4 The hours indicate the earliest passing times.

Figure 10. The response received on a bus query

The paragraph consists of four different types of textual lines relevant for this application. Line number 4 in Figure 10 is the line that indicates a successful query result. The entire paragraph is checked to ensure that this sentence is present. If it is, the rest of the paragraph is parsed as described below. If it is not present, a message is presented to the user explaining that there was an error when querying for bus schedules. Listing 3 shows how the remaining paragraph is parsed. The first two lines in Figure 10 are sentences that determine the name of the bus stop nearest to the user and the library. These lines are parsed using code line 105-107 in Listing 3. Sentences that describe busses with direct route to the given bus stops are not present in Figure 10, but these are parsed by code line 108-111, Listing 3. Finally sentences that describe bus routes with connections as line 3 in Figure 10, are handled by code line 112-115 in Listing 3. Each of the methods called in the different cases analyses the different sentences and determines the bus stop names, the bus numbers and the given departure times if required, respectively. If the structure of the sentences returned from the bus schedule service is changed in the future, this method and the methods called from it will have to be updated.

```
102. private string[] determineLineType(string line)
103. {
104.     string[] returnString = new string[6];
105.     if (line.Contains(" nearest to ")) {
106.         updateStationNames(line);
107.     }
108.     else if (line.Contains("and bus"))
109.     {
110.         returnString = handleTransfers(line);
111.     }
112.     else if (line.Contains("arrives at"))
113.     {
```

```

114.         returnString = handleDirect(line);
115.     }
116.     return returnString;
117. }

```

Listing 3. Determining input line type

If the read input line contains information on a transferring line, it only contains one time of arrival, as opposed to the direct line that contains a varying number of arrivals. This inconsistency is why the prototype displays only the next departure instead of a list of departures.

When a street name is returned from the bus query, an operation is performed on it to remove accents and apostrophes. This is done to ensure that the spelling is consistent with the one used in Google Maps when the directions are created. This can be done as the prototype is created to function within a limited area, and this manipulation will not create a street name that is equal to another already existing one.

As a result of the code running in Listing 3, a string array with the following six entries is created:

- The bus stop nearest to the user,
- The bus stop nearest to the library,
- The departure time of the next bus,
- The line number of the next departing bus,
- What bus line the user has to change to if a transfer is required,
- At which bus stop a potential transfer would take place

If no bus schedule is available, the user is returned to the main menu, where an error message is shown.

When the schedule information has been determined a Google map is created, generating routes from the user to his nearest bus stop and from the arrival bus stop to the library. The content of information bubbles that appear when an icon in the map is clicked is also pre-generated, and contains information on the bus stop addresses, bus departure times, transfer lines and so on.

When a query is submitted to the bus schedule service, the name of the bus stops nearest to the user is returned. However, these names are not deterministic addresses. As a solution to this, the bus stops returned have to be converted to street addresses. The addresses of a large part of the bus stops in the centre of Trondheim have been registered in a local file by the developer. As this is a prototype, some limitations can be put on the amount of bus stops required: As mobile internet is currently quite expensive, it is natural to believe that the application mostly will be used in areas with wireless network coverage. Figure 11 shows the bus stops in Trondheim indicated as blue dots, and the coverage area of the wireless network Trådløse Trondheim is indicated by the darkened area. The address of the bus stops that lay within or near this coverage area has been collected and is registered in the application, along with the ones surrounding the library departments. The list that converts from bus stop names to addresses is shown in Appendix E.

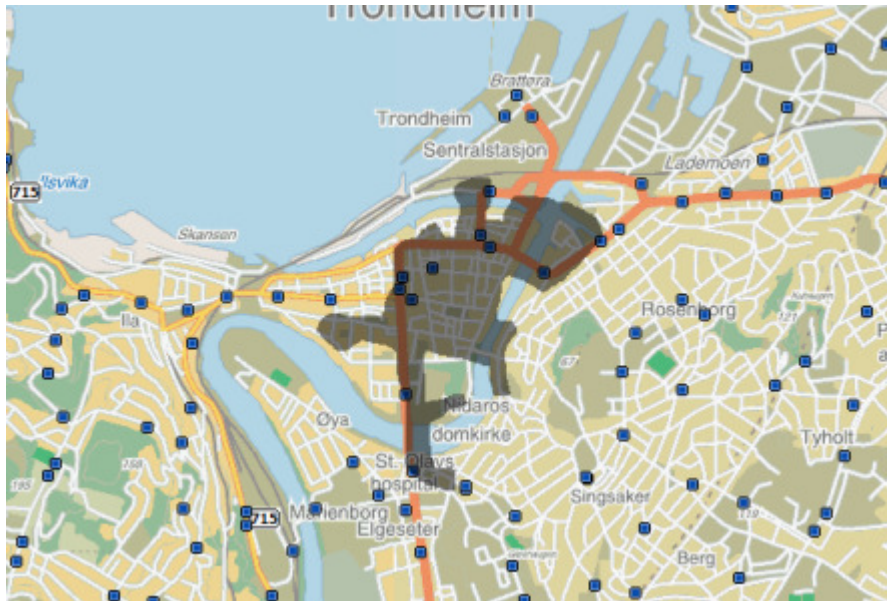


Figure 11. Wireless coverage and located bus stops

Image service

The image services are indicated in Module C in Figure 2. When the xml is returned from the BIBSYS SRU service, each node is parsed sequentially. Whenever a new record node is discovered, the ISBN number of the document is fetched. If this ISBN number is present, the prototype attempts to retrieve the cover image of the document from two different sources. Amazon provides the possibility of linking to an image by providing the ISBN number of the corresponding document as a parameter in an URL. By checking if the response from an http-request to this address is larger than a given threshold, it can be determined whether an image is available or not.

If an image is not available the same procedure is attempted against the LibraryThing service. The length of the response stream for this http request is also checked, and if it is large enough an html image tag is returned with this picture as a source. If the picture is not available here either, a “No image available” string is displayed where the image should be.

7 Walkthrough

This chapter demonstrates the functionality of the prototype. It shows what the user can do, from logging his or hers position to receiving the complete map with instructions on navigation and bus schedules.

7.1 Logging the users location

The first thing the user has to do is to log his or hers position. This is done using the GPS on the mobile device. It is assumed that the mobile application of the prototype is already downloaded, no installation is required.

Figure 12 shows the running mobile application. The user is informed that the device is connected to the remote server, and a counter informs the user every time the coordinates are updated. The latitude and Longitude are displayed, along with the number of satellites in view. In order to calculate the correct coordinates at least three satellites have to be in view. When this program runs the user's coordinates are transmitted to, and stored at, the remote server. The user can minimize this application and let it run in the background. If the user exits the application, the last registered position is stored at the server.



Figure 12. Registering user coordinates

7.2 Searching for documents

When the user navigates to the start page of the application using the mobile browser, the main menu is shown. In this prototype only the “Search for document” option is available, and when this is selected, the search interface in Figure 13 is shown.

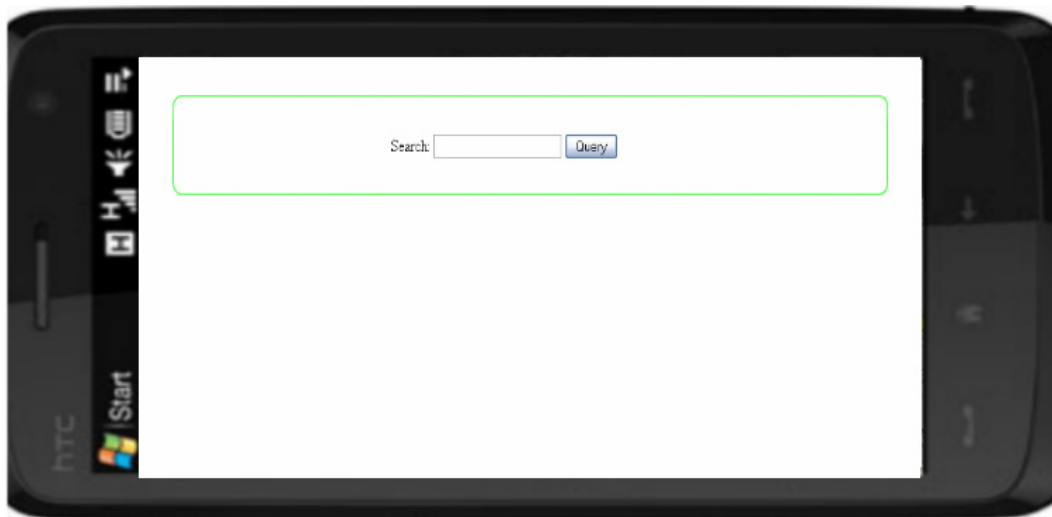


Figure 13. Search interface

By entering a keyword, or any CQL query, and pressing the “query” button, the user performs a search in the BIBSYS database. As an example, by entering the search word “Java”, the 10 first results are shown. Figure 14 shows a subset of these hits, as they appear on a mobile device.



Figure 14. Search results

The title and the authors of the document are displayed, along with the cover images of the corresponding books. All the library departments that holds a copy of the document is displayed, and if the department resides in Trondheim its name is a hyperlink. For each of the documents, it is indicated which library department is nearest to the user by a yellow triangle. If it is not possible to take the document out of the library, this is indicated by a padlock, as in the second result in Figure 14.

7.3 Displaying the map

When the user selects one of the libraries in Figure 14 a new page is displayed. If the users e.g. selected the “TEK/IDI” department, the page in Figure 15 is shown.

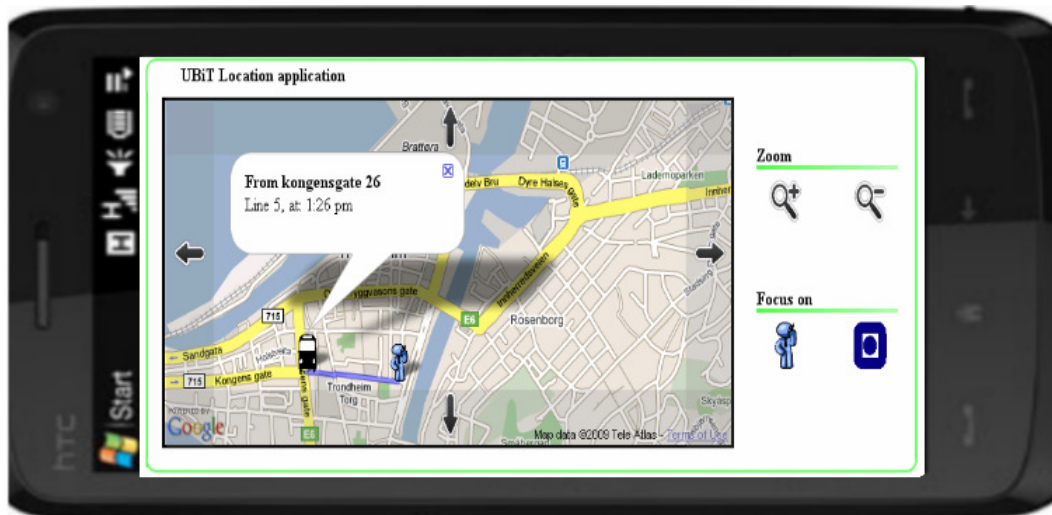


Figure 15. Map with user indicated

In Figure 15 the position of the user is indicated by a drawing of a person holding a mobile phone. There is also a route to the nearest bus stop (the bus icon) with connections to the selected library department. When the bus icon is clicked information on the next bus leaving for the library is shown. If the user would have to perform a transfer, information on this is displayed in the same bubble. By clicking on the library icon (NTNU logo) on the far right, the map shifts focus to the library department. The result of this operation is shown in Figure 16.



Figure 16. Map with library indicated

The arrival point of the bus is indicated by a bus icon. When clicking on this icon, the address of the bus stop is shown in the information bubble. A route to the library department (NTNU logo) from the bus stop is shown. If the NTNU logo in the map is clicked, the address and phone number of this library department is shown.

8 Testing

In Appendix D the test documentation is presented. The documentation contains test plans test designs, test cases for all the requirements described in section 3, as well as the test log. The purpose of the test documentation is to determine what has to be done in order to perform a complete test of the system, and to determine whether or not all of the requirements has been satisfied. The three modules of the system, the mobile application, the server application and the adapters are each tested individually, before they are tested together in a complete system test. The purpose of this chapter is to function as a test report, and to analyze and discuss the results of the tests. A complete overview of the test documentation is shown in Figure 17. The test plan, test designs, test cases and test log are located in appendix D, while this chapter is the test report.

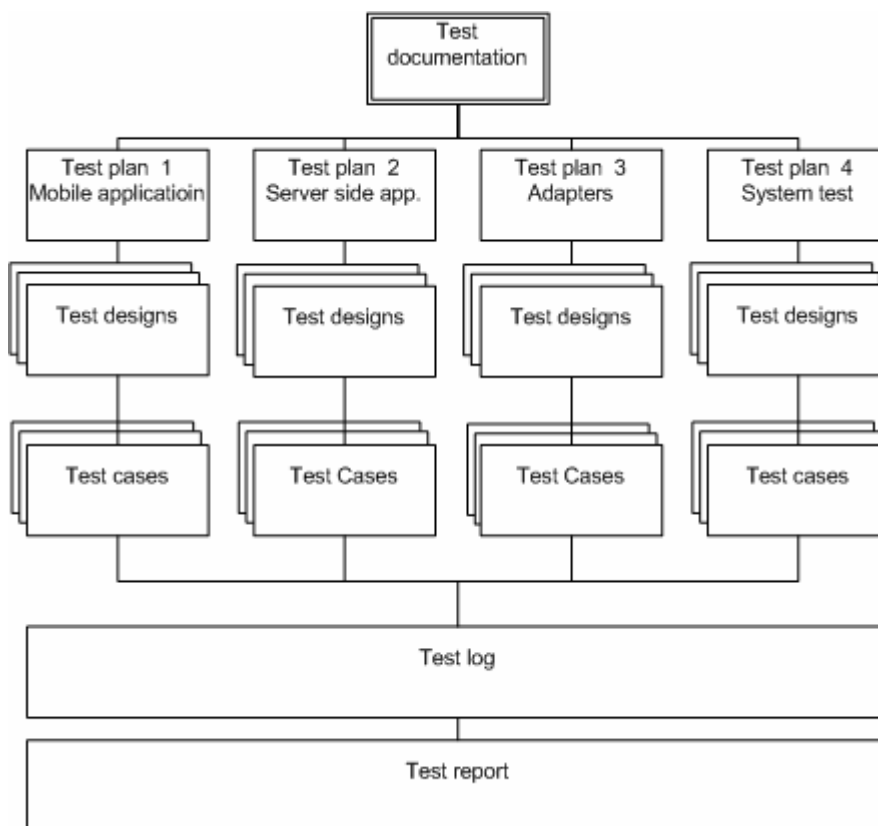


Figure 17. Test documentation overview

In order for the application to satisfy all of the requirements described in section 3, all four tests must pass. What is required in order to pass a test is described in each test plan, located in the appendix.

8.1 Testing the mobile application

The first plan ensures that the mobile device is able to communicate with the GPS system, so that the position of the user can be registered. The first test design handles the reading of the coordinates at the mobile device, while the other is concerned with the coordinates being

received at the web server. During this test, an external GPS unit was used to position the tester at the predetermined coordinates. The first three cases show that the mobile application correctly communicates with the local GPS device, by displaying the coordinates in real time to the user. Figure 18 shows the mobile application running on a mobile device, and displaying the coordinates as they were defined in the test case TC01-01, where the user is located on the street of “Bakke bru” in Trondheim.

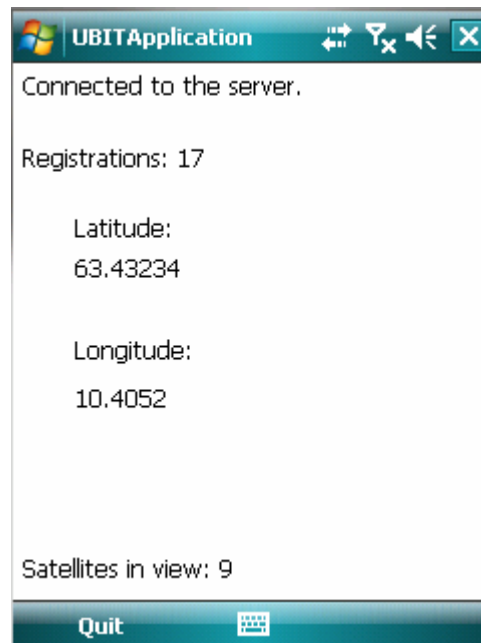


Figure 18. TC01-01 - Mobile application

In one of these cases, TC01-03, the observed latitude was off with two decimals. The observed longitude was 63.42745 while the external GPS reported 63.427496. This difference represents a distance that is lower than the resolution of GPS, and is therefore ignored.

The last case shows that the coordinates are correctly transmitted to, and registered at, the server.

The conclusion after the first tests is that the mobile application is successfully able to detect the position of the user, and to register that position at the server.

8.2 Testing the server side

This test plan ensures that the application server, the web server and the local file handling functions in accordance with the requirements.

The first tests in this section consist of registration of the information passed from the application server to the adapters. The first test confirmed that the keyword entered by the user is indeed the one used to query the BibsysAdapter, and the second confirmed that the bus service is queried for a route from the users address to the address of the library.

When testing TD02-02, “Passing addresses to bus adapter”, an error was detected. The Google map service translates user coordinates into readable addresses. The coordinates <63.43287, 10.391302> should translate into the address “Sandgata 1, 7012, Trondheim, Norway”. But this is a classified road and therefore its identification number, 715, is returned

instead of its name, “Sandgata”. As this street number is unknown to the bus schedule service, it is not able to serve requests based on this address. For this issue to be solved, the application should implement a translation from these identification numbers to street names. There are not many such roads within the city limits of Trondheim, and the problem is solved by generating a list with these translations and reading them into memory at start-up of the application.

Test design TD02-03, “Reading user coordinates from file”, confirms that the application is able to correctly read the coordinates of the user from the local GPS file.

In TD02-04, “Generating a static map”, a map is generated with simulated bus stops, departure times and bus lines. This map was correctly created, and it is determined that the application is able to provide a correct map if the background data is correct. Figure 19 shows the result of the test case TC02-06. The bus stop and the schedule are correctly indicated.

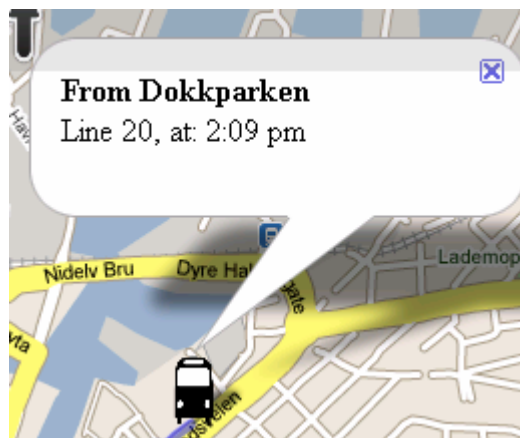


Figure 19. TC02-06 - Correctly indicating bus schedules

Finally, a document search is simulated in TD02-05, “Determining the nearest bus stop”. It is confirmed that the application is able to detect the bus stop nearest to the library holding the given document. Figure 20 shows one of the library departments, and the nearest bus stop correctly indicated.



Figure 20. TC02-07 - Correctly indicating the nearest bus stop

All the tests in this module have passed, except the one with the classified road issue, but suggestions on how this issue can be resolved have been given.

8.3 Testing service communication

When the application server requests some data from the external services, the adapters have to retrieve it.

The two first test designs confirm that the adapters connect to the correct external services for retrieval of data. Next a document search result for the prototype and the original library search engine is compared. The results show that the two searches return identical results. Figure 21 demonstrates two documents that match. The top part is from the prototype. The bottom is from the BIBSYS SRU.

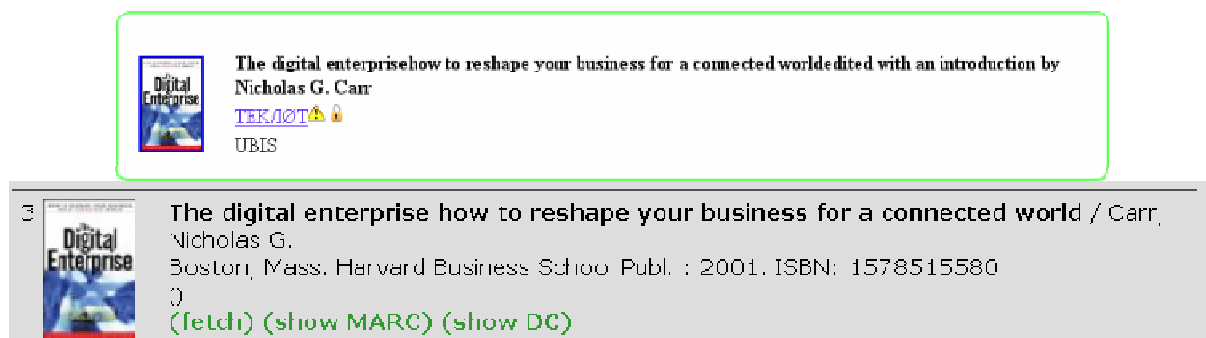


Figure 21. TC03-03 - Comparing document searches

Test design TD03-04, “Comparing bus schedule search”, shows that a bus route created in the prototype is identical with the response retrieved from the bus schedule service.

All tests have passed, and it is confirmed that the adapters accept requests from the application server, and that they are able to retrieve the data required of them.

8.4 System Test

The purpose of the system test is to ensure that all modules work together, and that the communication between them functions as designed. It also shows that the web server produces the user interface correctly.

The test was performed step by step as defined in the design specification. Figure 22 shows how the coordinates were correctly registered at the server when the test user was located in the position indicated by the test case.

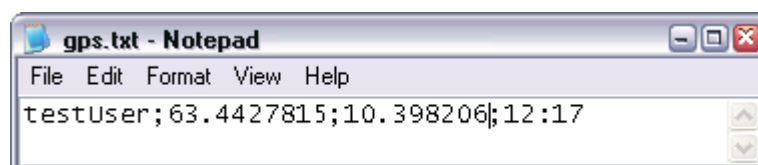


Figure 22. TC04-01 - GPS coordinates registered at the server

When the search for a document is performed, the document located at the library given in the test case is selected. The system correctly generates a map indicating the user's position (Figure 23). The nearest bus stop is also correctly indicated, along with information on the bus schedules.

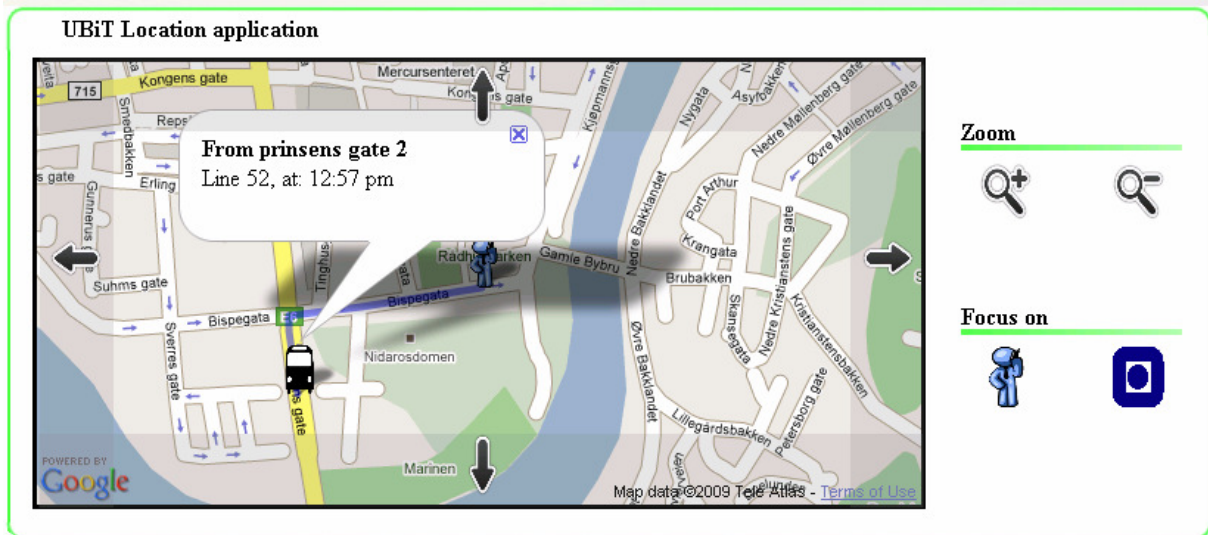


Figure 23. TC04-01 - The user indicated in the map

When focusing on the library holding the selected document, it is correctly indicated. The nearest bus stop is shown, along with information on the arrival address as shown in (Figure 24). In addition to this, address information on the library along with its phone number is shown.

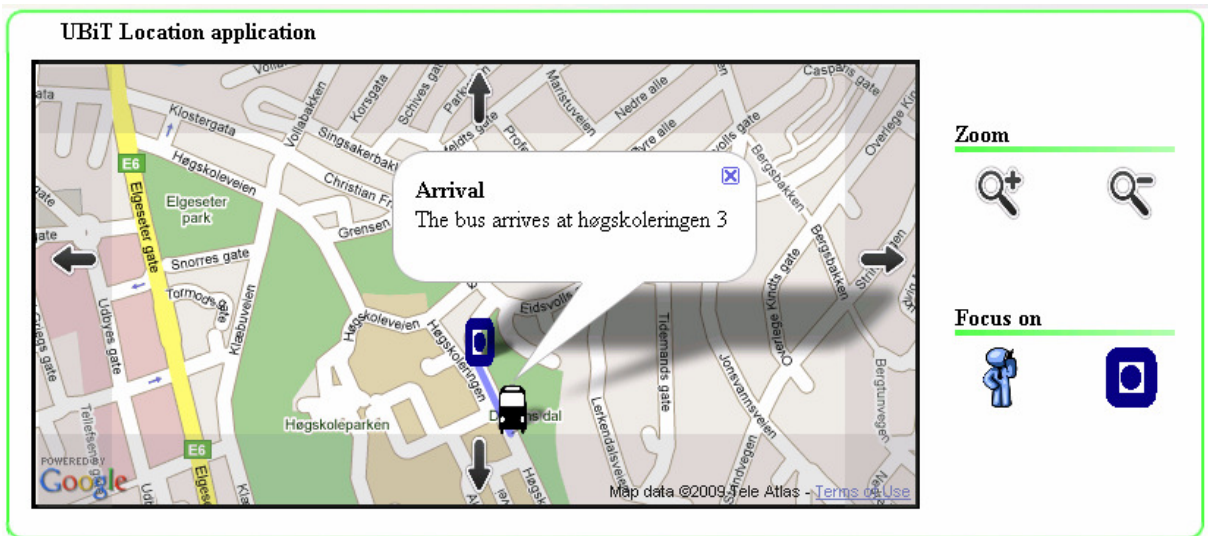


Figure 24. TC04-01 - The library indicated in the map

The map indicates the user, the bus stop nearest to the user, the bus stop nearest to the library, and the library correctly. In addition it displays the bus information for the next bus leaving for the library correctly. The prototype has passed the system test.

8.5 Test summary

Table 9 summarises the test results. All the tests were completed successfully, except the one with the classified road issue, TD02-02.

Test design	Test Case	Pass
TD01-01	TC01-01	Yes
TD01-01	TC01-01	Yes
TD01-01	TC01-03	Yes
TD01-02	TC01-04	Yes
TD02-01	TC02-01	Yes
TD02-02	TC02-02	No
TD02-02	TC02-03	Yes
TD02-03	TC02-04	Yes
TD02-03	TC02-05	Yes
TD02-04	TC02-06	Yes
TD02-05	TC02-07	Yes
TD03-01	TC03-01	Yes
TD03-02	TC03-02	Yes
TD03-03	TC03-03	Yes
TD03-04	TC03-04	Yes
TD04-01	TC04-01	Yes

Table 9 – Test summary

The test report shows that the user is able to perform a search for documents using free text search. A map is generated that contains both the position of the user as well as the position of the selected library department. A route is generated to the bus stop nearest the user. Bus schedules are correctly displayed, and a route is shown from the arrival bus stop to the library department.

All the functional requirements with a high priority are met.

The system takes advantage of the Google Maps service, and retrieves document information from BIBSYS. Bus schedules are correctly retrieved from team-trafikk. The application for the mobile device is currently not platform independent, but this is a design decision. The reason for this decision is given in section 5.2.2, and it is further discussed in section 9.1. The system successfully takes advantage of the user's location, and it retrieves this location using GPS.

The conclusion of the test phase is that the prototype satisfies all requirements with high priority given in section 3, except for the platform independence of the mobile device.

9 Evaluation and discussion

This purpose of this chapter is to evaluate the prototype created. The decisions made during the design and implementation phases are to be analyzed and evaluated. If the evaluation determines that some decisions have not been optimal, an alternative and improved solution will be discussed.

9.1 Mobile application

The mobile application communicates with the GPS device and transmits the GPS coordinates to the server. The need for a user first to download and install the mobile application, and then run two applications (both the mobile application and the web browser) in order to access the service is not optimal. Neither is the use of a Windows mobile application in accordance with the non-functional requirement for platform independence. An alternative solution that was considered was to integrate a web browser into the mobile application. This way the user only had to download, install and run a single application. Unfortunately there are some issues with this approach. In order to integrate a browser window in a .net application, we need to use a Windows ActiveX control. These controls have a much reduced functionality on the mobile platform, and enable only the use of Internet Explorer. For correct display of the Google Map generated by JavaScript, the JavaScript engine of the newest beta version of Opera 9.5 for mobile devices has to be used. As a result of this, the integration of an Internet Explorer window in the application using .net development is considered inadequate.

As Java was considered used as the development language for the mobile application, an integration of a browser window in such an application has to be considered. However, JME does not support this integration of browser windows. It does however support the running of other applications in the optional CHAPI package[32], but this would again give the two applications running at the same time. This would be preferable to the solution of the user to manually start two applications, but a third possibility is yet to be presented.

Opera and Firefox are two well known web browsers. Opera is currently available for both desktops and mobile devices, and is in fact the second most user mobile browser in the world[33]. Their newest beta version, Opera Mobile 9.5b, supports the retrieval of location information directly from the mobile devices by JavaScript, conforming to the Geolocation API specification [30]. The positioning can either be done by using e.g. the local GPS, or by wireless network access point triangulation. Mozilla has yet to release a Firefox mobile web browser, but it is reported that their first mobile browser will have the same localisation possibilities as Opera. As a consequence of this development, the prototype will in not long be able to run completely without the local application as long as the user has either Opera or Firefox installed on their mobile device. All the user has to do is to navigate to the web page, and it will automatically fetch the coordinates of the user. As a result of this development, the prototype contains a local application that can be removed as soon as the new browser versions are released. This prototype part is developed in C#.Net as this application is solely to be used during the test period of this prototype, and as a Windows mobile telephone is currently available to the developer.

In the future, the server application should be updated to support the Geolocation API specification as it is to be supported by Opera and Firebox mobile browser [30]. When this is implemented the user does not have to run the mobile application, but is able to grant the browser direct access to the local GPS device.

9.2 Server-side application

The server side application was designed as a web server accepting http requests from a web browser. The web server exploits services exposed by the application server to provide the user with the application interface. The Application server communicates with external service using adapters, so that if an information source changes structure or behaviour, only changes in the adapters would have to be made. The same applies if it is desirable to change the sources of information.

The web server

The web server is responsible for generating the JavaScript necessary to communicate with the Google Maps Server. This server handles the actual generation of the map, as well as the conversion from geographical points to addresses, and vice versa.

The generation of the Java Script is handled by a method being called upon the page load. This method requests parameters from either CGI or the Session object. These are combined with a static script. The complete Java Script is then written to the web page.

This generation of JavaScript makes the code for the web page rather illegible, and it would be hard to maintain. It also reduces the modularity of the code. If one part of the map generation is to be changed, the static pre-generated string has to be updated.

There exists a variety of alternatives to printing JavaScript strings in order to perform operations on the Google server. One of them is the Google Maps .Net control[34]. This is an open source project that attempts to wrap the functionality of Google Maps into asp.net. This way the developer will not have to write the JavaScript code, but instead call asp.net methods that handle the communication with the Google server. Unfortunately this project has some issues that require the wrapper to reproduce some of the functionality of the Google Server. Since the project site was latest updated in November 2006, this project appears to be abandoned. In order to use this as a solution in this project, it could be required to work around the issues. This will probably have had to be done without support of the original developers, and it would have to be considered whether this is in conflict with the terms of use for the Google Maps API[35].

Other .net wrappers are also available such as Subrigum [36], which is in Spanish, as well as one for the Geocoding alone, Desai [37]. For Java a complete wrapper is provided by Google [38].

As the JavaScript is printed as a string on the web page, debugging of this script is not available in the Visual Studio. The development therefore required the combination of a browser plug-in and an external validation tool. The use of a Java or .net wrapper can remove this issue, and can drastically shorten the development time for such an application. It can also contribute to the modularity, the readability, security and performance of the application. The modularity may be increased because one will be able to write in one language only, hence making it easier to move code around and to structure it in a modular way. This will also increase the readability. The security would be increased because it would be easier to write correct code when only one language is used, as well as the fact that debugging and writing assistance would be available in the developing environment. Performance may benefit from such a wrapper because there no longer is the issue of passing data from one programming language to another, as well as it can be easier to write the code efficiently.

In the prototype a web page (gpsHandler.aspx) is responsible for handling the registration of the user coordinates passed from the application running on the mobile device. As the coordinates are passed to the server they are written to a text file along with the name of the user that submit the coordinates as well as the time of registration. Using the external

file ensures that the localisation technology easily can be replaced at a later time, as long as the coordinates are written in the in the given form to the text file.

The application server

The application server is responsible for communication between the web server and the adapters. It is implemented using the Singleton pattern, meaning that only one instance of it exists at all times. This ensures that the data stored in local files only has to be read to memory one time, at server start-up. The application server functions as an abstraction layer between the web server and the adapters. It is very useful during development and testing, because it allows for easy interruption and manipulation of data.

It may be considered whether the reading of local data sources, such as the library information file, can be moved to the application server, or into a separate class. By doing this the adapters could become better structured and easier to modify, supporting the principles of modularisation.

9.3 Adapters and services

The adapters were created as stand alone classes, allowing the application server to perform operations on them. Communications with the Google Maps server is not done through the use of adapters.

Google Maps

The communication with the Google Maps server is not done through the use of an adapter because it uses JavaScript to generate the map. The simplest way of running this client-side script is by letting the user visit a page containing the script. Therefore the web server itself handles communication with the Google Maps server.

Google Maps was selected as the map service because the developer had previous experience with it, and because it offers relatively advanced possibilities. The issues with the use of JavaScript as map generator have been presented earlier in this section. Some of the alternatives to Google Maps presented in section 4.4 will be discussed in the following section.

The use of Bing Enterprise Maps as map service was rejected due to the state of the .net framework at the time this prototype was designed. It is developed an asp.net controller that allows developers to drag and drop map controls during the design of a web page that is to contain such maps. This controller allows the developer to write the methods that are called when events occur at the server, so that there is no need to wait for the call-back from the server in order to modify the functionality of the controller. The controller enables Ajax, so no full post back is required when the map is updated, reducing the network traffic. Figure 25 shows an example created for comparison with the prototype implemented using the map service from Google Maps. The code written for generating this example appears to be better structured, faster, more modular and easier to read than the one used to produce the corresponding example using Google Maps. However, the documentation for this controller is unstructured and is hard to understand. The development of such a controller may therefore require a longer start-up time, but the final result may benefit from it. This controller also provides the Bird's eye, located in the top right corner of Figure 25. This gives a diagonally 3-D view of the map that can simplify navigation for the user. The ASP.net controller also allows for the integration of additional services such as MSN and Silverlight streaming of multimedia directly on the web page. This map service should be tested in a new version of this project.

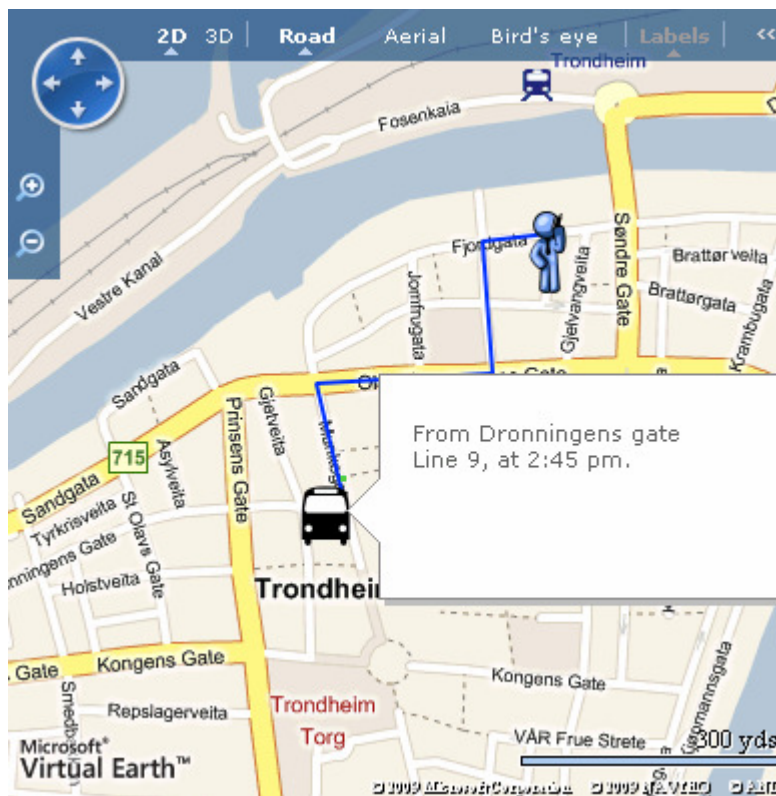


Figure 25. Bing Enterprise Maps ASP.net controller

Yahoo! Maps was earlier proposed as an alternative to Google Maps, but as this service suffers from the same drawbacks and as it has no larger selection of API's available this is not considered an option.

An issue worth mentioning about the Google Maps service is that some of the addresses are wrong. An example is "Høgskoleringen 1", which should indicate the location of the main building of NTNU, but which is indicated in a nearby street.

In further development it can be considered to replace the Google Maps with Bing Enterprise Maps. The Bing Enterprise Maps offers direct .Net API's, while Google Maps offers wrappers. Because the rest of the server application is written in .NET, this modification appears natural.

BibsysAdapter

The BibsysAdapter uses an XPathNavigator³ to traverse the results retrieved when querying the BIBSYS SRU. An alternative could be to use an XmlReader⁴, but as this is an event-based, forward-only XML pull parser, XPathNavigator was chosen as it was believed that it could become necessary to read backwards or ahead in the xml file. As a result, the fetching of ISBN numbers of documents occurs before the reading of the libraries holding it. In the xml document, however, they occur in the opposite order. With some modification in the design of this adapter, the XmlReader can be used instead. This may produce code that is easier to read, easier to debug and easier to manage. The biggest advantage with the XmlReader is that it only needs to hold a small part of the xml document in memory at all times, making it very efficient when it comes to memory use.

³ <http://msdn.microsoft.com/en-us/library/system.xml.xpath.xpathnavigator.aspx>

⁴ <http://msdn.microsoft.com/en-us/library/system.xml.xmlreader.aspx>

The file containing information on names, addresses and so on for each library department, was created as a part of this project. All information, except the coordinates, was retrieved from a webpage describing the different departments of NTNU; hence all departments of NTNU can be used as navigational targets in this prototype if it is desired. Some of the addresses are modified to some degree, so that each department is represented correctly in the map. This has no consequences for the user of the application. The opening hours are currently not included in this external file. The prototype supports the reading of those, and they can be appended to the file at a later time. This library information file is appended in Appendix F.

If the BibsysAdapter is updated the xml parsing should be updated according to the comments made in this section.

BusAdapter

The BusAdapter communicates with the bus schedule service. As team-trafikk is the only bus-company in the city of Trondheim, there exists no alternative information source except the direct database access approach. The use of natural text in paragraphs produces some extra work concerning data validation and consistency. Street names have to be stripped of special characters so that they can be placed in the map, and the addresses of bus stops have to be read from an external file. This work will however not be removed if the bus schedules are manually read from a database. The direct database access solution will in addition add the problem of locating the bus stop nearest to the user and the library, resulting in re-implementing much of the logic already present in *bussorakel*. In addition, such a solution will require the manual entry of all bus schedules into a database, and these schedules will not be up to date at all times.

There are additional issues with the solution of parsing real text paragraphs from the bus schedule service. The identification of strings, as described in section 6.2, depends on the words used in the sentence. If the sentences returned from the bus schedule service changes, this analysis must be rewritten. A solution to some of these issues could be to cooperate with the developers of *bussorakel*, and tag their returned paragraphs to indicate bus stop names, departure lines, transfer lines and arrival times.

The generated list of bus stops and their addresses are shown in Appendix E.

In the prototype the text in the information bubbles is created in the web server. In future development, the BusAdapter should handle the creation of the information bubbles generated when a user clicks on a bus icon the application. This would increase modularity, and make it easier to modify the content of these informational bubbles.

Image services

The Amazon server containing book cover images is used by the BIBSYS SRU service to display the front pages of documents returned by a query. Amazon's amounts of images are not enough to match all the documents returned when searching in the BIBSYS database. In an analysis performed on the search for 1500 search results, distributed over 15 keywords with 100 results for each, the percentage of images returned for each keyword were registered. The figures from this analysis are presented in Figure 26. It must be commented that it's not possible to get the cover images for all documents, as there are information on CD's and audio books in the results.

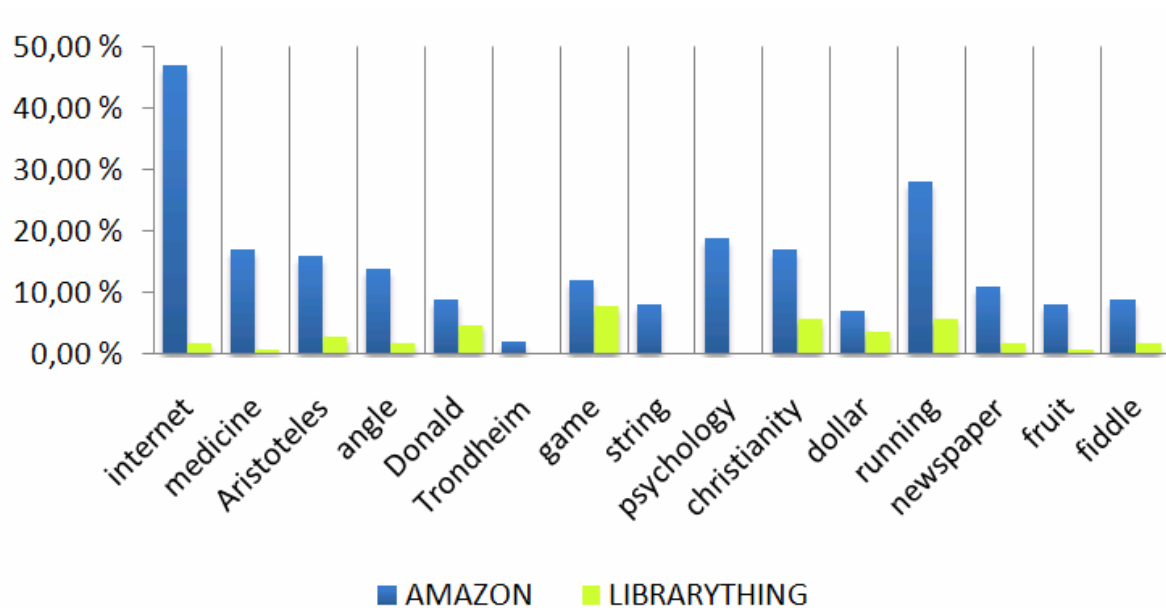


Figure 26. Image service analysis results

The numbers in Figure 26 show that Amazon returns a higher amount of images in each case than does the LibraryThing service. In fact, the number of images returned is 0 for 5 of the keywords for LibraryThing. In addition, this analysis does not consider images that are returned from both sources, so the benefit of implementing this service would not outweigh the additional waiting time for the user. Currently LibraryThing contain cover images for more than 1.4 million books. This is a little more than a tenth of the objects registered in the BIBSYS database. This number however is currently rising, as described in section 4.4.

If the ISBN number is present for a document, this approach requires in total three http connections to be established per image in a worst case scenario. Whether the image is available or not, it requires one connection to check if it is available from Amazon. If it is, another connection is required when the html image tag is read by the browser. If it is not available from Amazon, a new http connection is established to check the LibraryThing.com database. If it is present here, this also requires the download of the image when the browser reads the html image tag.

As a conclusion, Amazon proves the better of the two image sources presented. There are, however, many book cover image sources on the Internet, and as Amazon doesn't work satisfactory others should be considered

9.4 The tools and technologies

Both the server- and client side of the application was implemented using C# and the .NET framework. The decision was made due to a variety of reasons. C# was developed with network communication in mind, and it simplifies the process of communication between the mobile application and the server. On the client side, the .NET framework provides easy access to the GPS interface. Time was also an issue. C# is the language the developer has worked with the most during the last year, and has most experience in. There are no properties of the language that limits the development. The usual development environment for C# is Visual Studio, and the 2008 edition was used during this development. It provides the developer with advanced intellisense⁵ and debugging possibilities. Visual Studio has

⁵ Auto completion of code fragments.

integrated mobile emulators that simplify the development of the mobile application. When the Mobile 6 SDK is installed, a fake GPS application is available on the emulators which solves the problem of sitting indoors and develop on the GPS interface.

As an alternative to C#, Java was presented in section 4.2. The two languages are structurally similar, and neither gives functional restrictions on the development of this prototype.

The application generates JavaScript code that is interpreted by the user's browser. There are, as discussed earlier, some issues with this approach. In addition to the issues with generation of long static strings, are the ones with JavaScript as a language. Different browser has different JavaScript engines. As a result of this the prototype may end up looking different from browser to browser. An example of this is seen when the map generated is viewed in another browser than Opera Mobil 9.5 beta. The routes to and from the bus stops are not shown.

The current approach of generating JavaScript strings should, as discussed earlier, be replaced with a single-language approach.

9.5 Usability and usefulness

This section evaluates and discusses the usability and usefulness of the application as well as the graphical user interface and its functionality.

Mobile devices

Mobile devices have traditionally had small screens. This has changed during the last years, and large screen touch sensitive mobile phones have become more common. The prototype presents the user with a large map, currently at 600x300px. The user is able to navigate in this map using different approaches. On each edge of the map there are partly transparent buttons that when clicked pans the map in that direction. To the right of the map there are icons that indicate the user and the selected library. By clicking on these the map focus on the corresponding area.

One issue with the user interface is, as mentioned in section 9.1, that the user has to use a specific browser. Also, on the mobile devices, click-and-drag is not possible. Therefore the navigation buttons had to be created by the developer, and placed on top of and at the side of the map.

The map has currently a static size. If the possibility for multiple users is implemented, it can be simple to let the user select the preferred map size, as well as the layout of the controls. This way the entire user design can be modified to the user's needs. The same applies for the listings of the search results, which now are presented in a rather wide column. One way to implement this solution can be to generate a set of cascading style sheets (CSS), and let the user select from these. A pre-generation of a style sheet for each of the different device categories will not take to long time, and can be a nice way to increase the usability of the application. Another possibility is to define the size of elements as relative to the size of the screen, and create the border graphics in smaller units that repeat themselves accordingly.

The development of the prototype has not focused on the user interface to a great extent, as this was not one of highly prioritized requirements. In further development the design should be evaluated. As an example, it is possible to rearrange the buttons on the right hand side, to increase the size of the map. The development does not necessary has to focus on too small mobile screens, as devices with the smallest screens most lightly do not have GPS installed.

Speed and reliability

The application retrieves information from a large variety of sources. It is evident that the reliability of the application is no better than the worst of the external sources. The service that has been out of function the most during the development phase is the bus schedule service. As of now the application informs the user that bus schedules is not available if a connection to the bus schedule service can not be established. A way to reduce this problem in the future can be to run a copy of the bus schedule service on a server that is under UBiT control. This requires that the copy is updated according to the original bus schedule database when the schedules changes due to season, or when there is a change on one or more lines.

In section 3 the property of speed was not chosen as one of the high prioritised requirements. As a consequence, the application does spend some time on the retrieval of information. The fetching of book cover images takes some time because two different sources has to be checked before the image is actually fetched the third time. It does not implement any form of cache, which may have improved the performance. The parsing of returned data from the library database could have been somewhat different. At the time the xml documents are parsed with focus on memory use. By reading the entire document into memory, specific nodes may have been accessed more easily.

9.6 Conclusion

In section 1.3 two specific goals for this thesis were presented. The first was to demonstrate how location awareness and maps can be exploited to facilitate the use of library resources such as information on documents and objects. The application is able to detect the correct position of the user. By searching for keywords the user is able to find information on documents in the library database. If the documents are present in at least one of the departments of UBiT, the user is correctly informed of the department closest to him or her. A map is generated with information on bus stops and bus schedules that assist the user in navigating to the correct department. This will simplify the process of determining the physical location of a document and navigating to the library department holding a copy of it. The second goal was to show the feasibility of integrating several different information sources into the prototype. The prototype developed in this project communicates with a large amount of external services. It queries the BIBSYS database on information on objects. It reads the corresponding library locations from a local file. Bus schedules are fetched from the local bus company, and the addresses of the bus stops are read from a local file. Finally, maps are generated by communicating with the Google Maps server. The communications with the external services have been done in such a way that it is relatively uncomplicated to replace one information source with another. In conclusion, the prototype shows that it is feasible to integrate with several sources of information. Both the goals that were set in the beginning of the project have been met.

During development of the prototype, some valuable experiences have been made. If the development process was to start over again, there are some changes that could contribute to a better application and user experience. These experiences are summarised in the rest of this section.

The mobile application that is currently determining, and submitting to the server, the location of the user should be superfluous in future version. Instead the application should take advantage of the Geolocation API specification from W3C, enabling the browser to read the users position directly from the GPS installed on the mobile device. This simplifies the process for the user, as the only thing that has to be done in order to use the application, is to visit the web page.

The web server should abandon Google Maps and go for Bing Enterprise Maps as map service provider, to ensure that the entire application is written in the same language as much as possible. Alternatively one of the NET wrappers available could be used to implement the current communication with Google. This will increase the modularity, and could increase security, reliability and performance of the application.

The Application server should stay as it is. It helps the developer focus on the modularity, and simplifies development.

There are more speed efficient ways to parse the results received from the BIBSYS service than the methods currently used. By implementing a push parser, e.g. the XmlReader as described in section 9.3, the process of reading and structuring the read data can be vastly improved.

It should be attempted to collaborate with the developers of *bussorakel*, in an attempt to tag the textual results from the service. This would contribute to a more stable service, not relying on single words in the text that might change over time. Also the BusAdapter class should be responsible for generating information bubble content, ensuring modularity.

New book cover image sources should be considered. Not enough images are available at Amazon.com and LibraryThing, and the display of these images could greatly contribute to the experience for the user.

The graphical user interface should be designed so that it simplifies navigation for users with smaller mobile screens.

The prototype developed provides all the functionality that was required of it. By performing the modifications suggested in this section an even better application, with equal functionality, can be achieved.

10 Future work

While the previous chapter concluded on how the existing functionality could have been implemented better, this chapter focuses on how the functionality of the prototype can be extended to increase the value to the user. The suggestions presented here are in addition to the unimplemented functional requirements presented in section 3.2. New external services that can be integrated with the application are presented, and it is suggested how they can be implemented in the application. These suggestions are services that can be integrated into the application as it is today, and does not require the development of a new application.

10.1 Real time bus schedules

During the spring of 2009, the city of Trondheim was assigned money for providing the public transport travellers with real time information on bus schedules[39]. When this system was established in Oslo an API was provided and it was encouraged to develop applications that took advantage of this. There is no reason to believe that this will not also become available in Trondheim. This way the bus schedules displayed in the application can be updated to show real-time schedule information.

Shown in Figure 27 is what this may look like. To implement it, the BusAdapter would have to be updated so that it fetches bus schedules from the new source. Also the method that generates the information shown in the information bubbles should be updated.

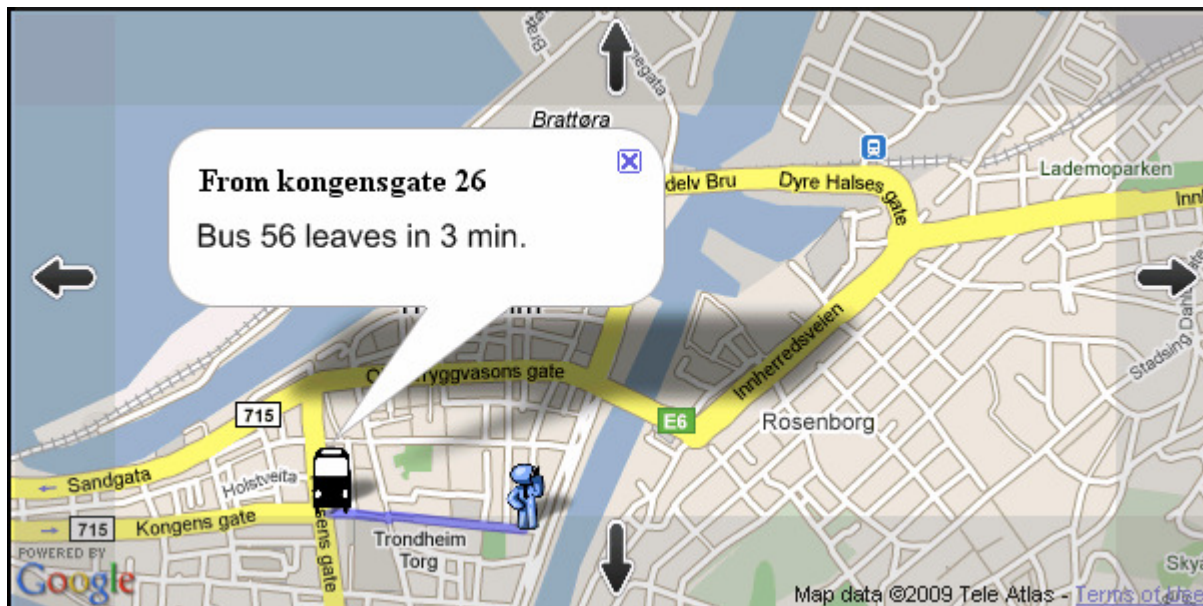


Figure 27. Real time bus schedules

10.2 Map at the library

When the user arrives at the library, it could be valuable to present the user with a map of the library. In this map the location of the document that was searched for is indicated. For some library departments, and some documents, this service already exists through BIBSYS.

The map can easily be created by passing the shelf number of document to an application running on the BIBSYS server. The base URL that the call has to be made to is:

i)

[“http://hyllekart.bibsys.no/app51/map/wicket/PageLocations?instance_id=no_ubi&encoding=utf-8”](http://hyllekart.bibsys.no/app51/map/wicket/PageLocations?instance_id=no_ubi&encoding=utf-8)

The parameter passed is simply the library department ID and the shelf the document resides on. An example is a document that resides in a shelf in the Dragvoll library department:

ii)

“&loc=DRAGVOLL&shelf=SIR%2F2009%3A3”.

By simply inserting a hyperlink composed of i) and ii), the user will by clicking on the link be showed the correct map. In order to generate the URL the correct parameters have to be fetched. These parameters are present in the xml document returned when performing a document query and it is therefore only a question of reading, saving and passing them to the web page displaying the map.

Figure 28 shows what the map delivered by BIBSYS looks like on their website, with the selected document indicated by a red marker.

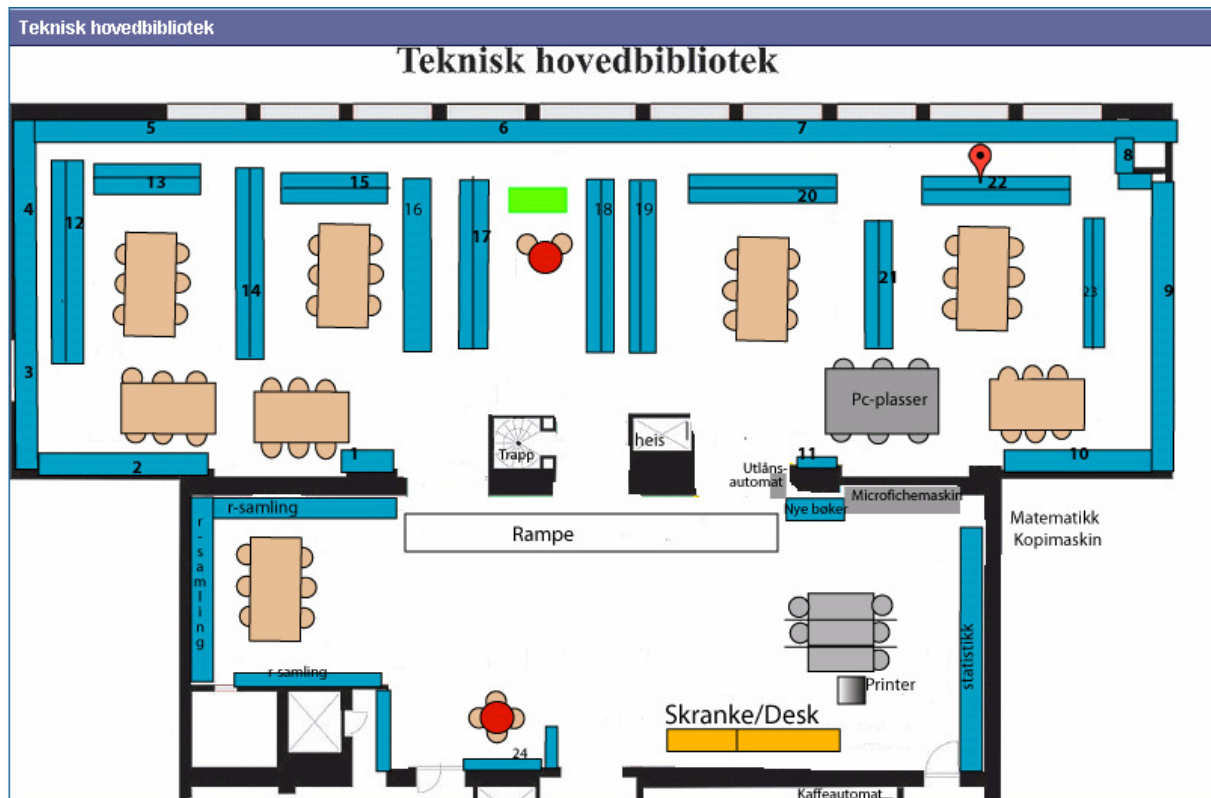


Figure 28. Local map of library

The library department map, as described in this section, was considered implemented in the prototype, but it became clear that very few documents can be placed in these maps. Only some of the library departments have such maps, and a subset of these actually maintains them. A lot of documents reside in storage, for which maps are not available.

If the amount of maps that is maintained is increased, this could prove a valuable extension to the application. Other elements than the book searched for could be indicated in such a map. It could include an overview of printers, toilets, resting areas, vending machines

and so on. This is already somewhat implemented in the current map solution from BIBSYS, but the quality could be improved, and the functionality should be adapted to mobile phones. Currently there is a menu (not indicated in Figure 28) on the left hand side of this web page that let the user selects items such as printers. The indicators of these objects occur however no different from indicators of documents in the map, and it can therefore be easy to mix them up, especially on a small screen.

10.3 Advanced searching options

Currently the search field in the prototype only accepts key words, as this was one of the requirements during the development of the application. The original BIBSYS, however, does offer a wide variety of search options. These include limiting the search to library departments which resides in Trondheim

The current search interface in the prototype does accept, as the original BIBSYS SRU, any valid CQL query. This implies that the users are able to pass queries like:

- i) `dc.title="origin of the species"`

The response then only contains information on documents that has that exact string in the title. By implementing an interface that automatically adds the required prefixes, as “dc.title” in i), the full functionality of an advanced search interface can be achieved. This allows the user to put restrictions on libraries to be included in the search results, limit the result to documents produced by certain authors and so on. The implementation of these search options could prove very valuable, as it at the time can be difficult to navigate the large amount of search results generated if one is unfamiliar with the CQL standard.

11 References

1. B.S. Olsen, "Lokasjonstjenester for Universitetsbiblioteket i Trondheim," Project, IDI, NTNU, 2008.
2. J.G. McNeff, "The Global Positioning System," 2002; <http://ieeexplore.ieee.org/iel5/22/21335/00989949.pdf?tp=&arnumber=989949&isnumber=21335>.
3. GeoPos, "Geopos - Item," 2008; <http://geopos.item.ntnu.no/>.
4. BIBSYS, "About BIBSYS Website," 2009; <http://www.bibsys.no/norsk/english.php>.
5. I.o.c. Network Development and MARC standrads office, "MARC 21 Format for bibliographic data," 2006; <http://www.loc.gov/marc/bibliographic/bdsummary.html>.
6. BIBSYS, "BIBSYS-MARC," 2009; http://www.bibsys.no/files/out/handbok_html/marc/marctoc.htm#TopOfPage.
7. myRete, "WhosHere Overview," 2008; <http://myrete.com/WhosHere.html>.
8. Gizmodo, "Sekai camera," 2009; http://www.gizmodo.com.au/2008/09/sekai_camera_turns_on_worlds_balloon_help-2.html.
9. M. Fowler, *UML distilled*, Addison-Wesley, 2004.
10. Gartner, "Gartner says worldwide Smartphone Sales Grew 16 Per Cent in Second Quarter of 2008," 2009; <http://www.gartner.com/it/page.jsp?id=754112>.
11. Google, "Android Developers," 2009; <http://developer.android.com/guide/index.html>.
12. HTC, "HTC Dream - Overview," 2009; <http://www.htc.com/www/product/dream/overview.html>.
13. Sun, "Java ME Technology," 2009; <http://java.sun.com/javame/technology/index.jsp>.
14. S. Helal, "Pervasive Java," *Pervasive Computing, IEEE*, vol. 1, no. 1, 2002, pp. 82-85.
15. C. Neable, "The .NET Compact Framework," *Pervasive Computing, IEEE*, vol. 1, no. 4, 2002, pp. 84-87.
16. Bobbie Johnson, "GPS system 'close to breakdown'," 2009; <http://www.guardian.co.uk/technology/2009/may/19/gps-close-to-breakdown>.
17. I. Jami, M. Ali, and R.F. Ormondroyd, "Comparison of methods of locating and tracking cellular mobiles," *Novel Methods of Location and Tracking of Cellular Mobiles and Their System Applications (Ref. No. 1999/046), IEE Colloquium on*, pp. 1/1-1/6.
18. Google, "Google Maps API," 2009; <http://code.google.com/apis/maps/documentation/>.
19. Microsoft, "Virtual Earth Articles," 2009; <http://msdn.microsoft.com/en-us/library/bb545001.aspx>.
20. Microsoft, "Live Framework SDK and Tools," 2009; <http://dev.live.com/liveframework/sdk/>.
21. Yahoo, "Yahoo! Maps Web Services," 2009; <http://developer.yahoo.com/maps/>.
22. W3C, "Web Services Architecture," 2004; <http://www.w3.org/TR/ws-arch/>.
23. BIBSYS, "BIBSYS sru web service," 2009; <http://sru.bibsys.no/>.
24. Team-trafikk, "Team Trafikk," 2009; <http://team-trafikk.no/>.
25. L. thing, "LibraryThing," 2009; <http://www.librarything.com/>.
26. J. Krogstie, "Prototype spring 08 - Wireless Trondheim living lab," 2009; <http://research.idi.ntnu.no/trimaks/services/demonstrators/tradlos-byvandring/submitted.pdf/view>.
27. Canalys, "Canalys research release 2008/082," *Book Canalys research release 2008/082*, Series Canalys research release 2008/082, ed., Editor ed.^eds., 2008, pp.

28. Opera, "Find me! Geolocation-enabled Opera build," 2009; <http://labs.opera.com/news/2009/03/26/>.
29. Mozilla, "Mozilla Firefox 3.5 Beta 4 Release notes," <http://www.mozilla.com/en-US/firefox/3.5b4/releasenotes/>.
30. W3C, "Geolocation API Specification," *Book Geolocation API Specification*, Series Geolocation API Specification, ed., Editor ed.^eds., 2008, pp.
31. Opera, "Opera Mobile," 2009; <http://www.opera.com/mobile/>.
32. Sun, "CHAPI," 2009; <http://java.sun.com/products/chapi/>.
33. StatCounter, "Top 9 mobile browsers from 1 jul 08 to 18 may 09," 2009; http://gs.statcounter.com/#mobile_browser-ww-daily-20080701-20090518-bar.
34. "Google Maps .Net Control," 2006; <http://gmapsdotnetcontrol.blogspot.com/>.
35. Google, "Google Maps/Google Earth APIs Terms of Service," 2009; <http://code.google.com/apis/maps/terms.html>.
36. Subrigum, "Google Maps para ASP.NET," 2009; <http://googlemaps.subgurim.net/>.
37. S.Y. Desai, "A .NET API for the Google Maps Geocoder," 2008; <http://www.codeproject.com/KB/custom-controls/GMapGeocoder.aspx>.
38. Google, "gwt-google-apis," 2009; <http://code.google.com/p/gwt-google-apis/>.
39. Aftenposten.no, "Nå får du vite om bussen er forsinket," 2009; <http://www.adressa.no/nyheter/trondheim/article1323322.ece>.
40. R.K. Clemmons, "Project Estimation With Use Case Points," *CrossTalk : the journal of defense software engineering*, vol. 2, 2006, pp. 18.

A. Appendix A – Use case point estimation 1

Use case point estimation for the complete design

These tables were created based on a method presented in [40]. This estimation calculates the cost of implementing all use cases listed in chapter 3.

Unadjusted Actor Weighting Table

Actor Type	Description	Weighting Factor	Number	Result
Simple	The actor represents another system with a defined application programming interface	1	2	2
Average	The actor represents another system interacting through a protocol, like Transmission Control Protocol/Internet Protocol	2	3	6
Complex	The actor is a person interacting via a graphical user interface	3	1	3
Unadjusted Actor Weight Total (UAW)				14

Unadjusted Use Case Weighting Table

Use Case Type	Description	Weighting Factor	Number	Result
Simple	Simple user interface. Touches only a single database entity. Its success scenario has three steps or less.	5	2	10
Average	More interface design. Touches two or more database entities. Between four and seven steps.	10	3	30
Complex	Complex user interface or processing. Touches three or more database entities. More than seven steps.	15	1	15
Unadjusted Use Case Weight Total (UUCW)				55

$$\text{Unadjusted Use Case Points (UUCP)} = \text{UAW} + \text{UUCW} = 14 + 55 = 69$$

Technical Complexity Factors

Factor Number	Description	Weight	Assigned Value (0 – 5)	Weighted Value	Notes
T1	Distributed system	2	4	8	
T2	Response time or throughput performance objectives	1	1	1	
T3	End-user online efficiency	1	1	1	
T4	Complex internal processing	1	0	0	

T5	Reusability of code	1	1	1	
T6	Easy to install	0.5	4	2	
T7	Ease of use	0.5	4	2	
T8	Portability	2	5	10	
T9	Ease of change	1	3	3	
T10	Concurrency	1	1	1	
T11	Special security objectives included	1	0	0	
T12	Direct access for third parties	1	0	0	
T13	Special User training required	1	0	0	
Technical Factor Value (TFactor)				29	

$$\text{Technical Complexity Factor (TCF)} = 0.6 + (0.01 * \text{TFactor}) = 0,6 + (0.01*29) = 0.89$$

Environmental Factors

Factor Number	Description	Weight	Assigned Value (0 – 5)	Weighted Value	Notes
E1	Familiarity with system development process being used	1.5	4	6	
E2	Application experience	0.5	3	1,5	
E3	Object-oriented experience	1	3	3	
E4	Lead analyst capability	0.5	2,5	1,25	
E5	Motivation	1	5	5	
E6	Requirements stability	2	4	8	
E7	Part time staff	-1	0	0	
E8	Difficulty of programming language	-1	1	-1	
Environmental Factor Value (EFactor)				21,25	

$$\text{Environmental Factor (EF)} = 1.4 + (-0.03 * \text{EFactor}) = 1.4 + (-0.03 * 21,25) = 0,7625$$

$$\text{Adjusted Use Case Points (UCP)} = \text{UUCP} * \text{TCF} * \text{ECF} = 69 * 0,89 * 0,7625 = 46,8$$

Effort in Person Hours = UCP * PHM(hours per man per use case point)
PHM is initially set to 18 in this case

Effort = 18*46,8 = 842 person hours in total
45 hours pr week = 19 weeks

B. Appendix B – Use case point estimation 2

In chapter 3 a set of use cases are presented. The following calculations estimate the cost of implementing the use cases that prove the most valuable to the prototype. The use cases selected are based on the priorities of the requirements described in the same chapter.

Unadjusted Actor Weighting Table

Actor Type	Description	Weighting Factor	Number	Result
Simple	The actor represents another system with a defined application programming interface	1	3	3
Average	The actor represents another system interacting through a protocol, like Transmission Control Protocol/Internet Protocol	2	2	4
Complex	The actor is a person interacting via a graphical user interface	3	2	6
Unadjusted Actor Weight Total (UAW)				13

Unadjusted Use Case Weighting Table

Use Case Type	Description	Weighting Factor	Number	Result
Simple	Simple user interface. Touches only a single database entity. Its success scenario has three steps or less.	5	1	5
Average	More interface design. Touches two or more database entities. Between four and seven steps.	10	0	0
Complex	Complex user interface or processing. Touches three or more database entities. More than seven steps.	15	1	15
Unadjusted Use Case Weight Total (UUCW)				20

Unadjusted Use Case Points (UUCP) = UAW + UUCW = 33

Technical Complexity Factors

Factor Number	Description	Weight	Assigned Value (0 – 5)	Weighted Value	Notes
T1	Distributed system	2	4	8	
T2	Response time or throughput performance objectives	1	1	1	
T3	End-user online efficiency	1	1	1	
T4	Complex internal	1	0	0	

	processing				
T5	Reusability of code	1	1	1	
T6	Easy to install	0.5	2	1	
T7	Ease of use	0.5	2	1	
T8	Portability	2	5	10	
T9	Ease of change	1	3	3	
T10	Concurrency	1	1	1	
T11	Special security objectives included	1	0	0	
T12	Direct access for third parties	1	0	0	
T13	Special User training required	1	0	0	
Technical Factor Value (TFactor)				27	

$$\text{Technical Complexity Factor (TCF)} = 0.6 + (0.01 * \text{TFactor}) = 0.6 + (0.01*27)=0.87$$

Environmental Factors

Factor Number	Description	Weight	Assigned Value (0 – 5)	Weighted Value	Notes
E1	Familiarity with system development process being used	1.5	4	6	
E2	Application experience	0.5	3	1,5	
E3	Object-oriented experience	1	3	3	
E4	Lead analyst capability	0.5	2,5	1,25	
E5	Motivation	1	5	5	
E6	Requirements stability	2	4	8	
E7	Part time staff	-1	0	0	
E8	Difficulty of programming language	-1	1	-1	
Environmental Factor Value (EFactor)				21.25	

$$\text{Environmental Factor (EF)} = 1.4 + (-0.03 * \text{EFactor}) = 1.4 + (-0.03 * 21.25)=0,7625$$

$$\text{Adjusted Use Case Points (UCP)} = \text{UUCP} * \text{TCF} * \text{ECF} = 33*0,87*0,7625 = 22$$

Effort in Person Hours = UCP * PHM(hours per man per use case point)

PHM is initially set to 18 in this case

Effort = 18 * 22= 396 person hours in total

45 hours pr week = 9 weeks

C. Appendix C – Project schedule

The Gantt diagram in Figure 29 is a plan for the development of the prototype and its evaluation. The plan has, on the whole, been followed.

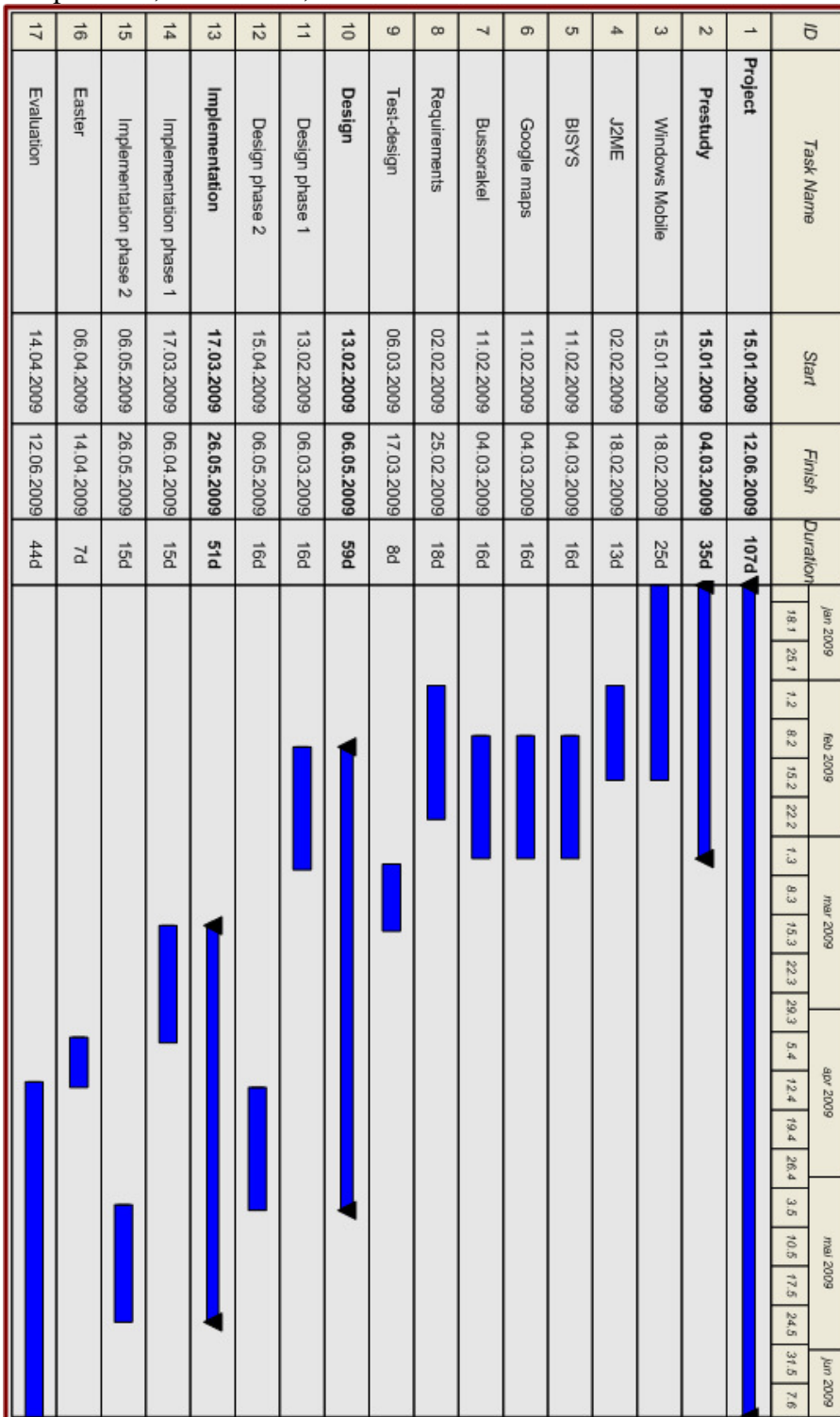


Figure 29. Gantt diagram - Schedule for project

D. Appendix D - Test documentation

Introduction

This document contains the plans for the testing of the prototype. It describes the goals with the testing, the background for the testing, the granularity of the tests, as well as what result is to be produced from these tests.

Goals

The goals of these test plans are to

1. Describe what has to be done in order to perform a complete test of the system.
2. Determine whether or not the application satisfies the requirements defined in chapter 3.

Background

The system consists of three parts. One part handles the interaction with the external services, such as *Bussorakel*. Another controls the localisation of the user. The third part, the web server integrates these operations, and presents the user with an interface, as well as handling the generation of the map. As a result of this, the tests will be divided into four parts. One for each of the modules, and one system test to ensure the integration of these modules.

Granularity

This test is to be a black box test of each of the modules. All functional and non-functional requirements with high priority, as described in section 0, are to be tested.

Test deliverables

As a result of this testing, two documents are to be produced:

- System Test Log, a document containing the log for each of the test cases
- System test Summary Report, a document that summarizes and analyses the result for each of the test cases. Chapter 8 functions as this report.

Figure 17, in chapter 8, gives an overview of the test documentation.

Test plan 1 – Mobile application

Goals

The goal of this test is to ensure that the mobile device is able to communicate with the GPS system to retrieve the location of the user, and that this information is relayed to the web server. This test covers module 1 of the system.

Requirements

The following requirements are tested in this test, as they are defined in section 0;
D04, D06

Test design

ID(TD01-XX)	Steps	Refs
TD01-01	Title: Generating user coordinates 1. Go to the coordinates defined in TC01-01 through 03 2. Start up the mobile application 3. Register the coordinates for each of the given points	D04, D05, D06
TD01-02	Title: Registering user coordinates 1. Start the mobile application 2. Go to the coordinates defined in TC01-04 3. Check the coordinates registered at the server, and observe that they are equal with the ones in the test case.	D05, D06

Test plan 2 – Server side

Goals

The goal of this test is to determine that the application server, web server, and database communicate correctly. The application must retrieve the correct information from the services and the database, as requested by the web server. Many of the tests are performed by running the application in debug modus, and by intercepting the various calls made between the modules of the application. This test covers module 2 of the system.

Requirements

The following requirements are tested in this test, as they are defined in section 0;

C01, C02, C03, C04

D01, D02, D03, D05

Test design

ID(TD02-XX)	Steps	Refs
TD02-01	<p>Title: Passing keyword to BibsysAdapter</p> <ol style="list-style-type: none"> 1. Simulate a keyword search by performing an httpGet call on the web server, using a normal computer, with the keyword in case TC02-01 2. Register the keyword as a search request is performed on the BibsysAdapter. 	C01
TD02-02	<p>Title: Passing addresses to BusAdapter</p> <ol style="list-style-type: none"> 1. Using the user position and the document given in cases TC02-02 and TC02-03, issue a http Get to the web server using a regular computer 2. Register the two addresses as a bus search is performed on the BusAdapter. 	C04
TD02-03	<p>Title: Reading user coordinates from file</p> <ol style="list-style-type: none"> 1. Using the document and the user position given in case TC02-04 and TC02-05, issue an http Get request from a regular computer. 2. Log the coordinates of the user as read by the application server 	C02, C04, D05
TD02-04	<p>Title: Generating a static map</p> <ol style="list-style-type: none"> 1. Perform a document search. Intercept the server when it attempts to perform a bus query and manually return the bus stops, departure line and departure times given in TC02-06 2. Log whether the map generated corresponds to the bus stops, departure line and departure time given in the test case 	C01, C02, C03, C04,
TD02-05	<p>Title: Determining the nearest bus stop</p> <ol style="list-style-type: none"> 1. Perform a document search. Intercept the server when it attempts to perform a document query, and return the list of documents given in case TC02-07 2. Log whether the arrival bus stop is the one nearest to the library holding the document 	C01, C02, C03, C04

Test plan 3 – Service communication

Goals

The goal of this test is to ensure that the external services are queried for the correct information, and that they are able to provide the information requested. This test covers module 3 of the system. The usual approach for these tests would be to perform operations on the adapters alone by running the application in debug modus.

Requirements

The following requirements are tested in this test, as they are defined in section 0;

C01, C02

D01, D03, D05

Test design

ID(TD01-XX)	Steps	Refs
TD03-01	Title: Determining document information source 1. Perform a query on the BibsysAdapter with the keyword given in TC03-01 2. Register the address that the adapters relays the query to	C01, D02
TD03-02	Title: Determining bus schedule source 1. Perform a query on the BusAdapter with the addresses given in TC03-02 2. Register the address that the adapters relays the query to	C03, C04, D03
TD03-03	Title: Comparing document information search 1. Perform a query on the BibsysAdapter with the keyword given in TC03-03 2. Register the 10 first results retrieved from the BIBSYS service 3. Perform a query for the same keyword on the BIBSYS SRU webpage 4. Compare the two results	C01, C02, D02
TD03-04	Title: Comparing bus schedule search. 1. Perform a query on the BusAdapter with the two addresses given in TC04-04 2. Register the departure bus stop, the bus line(s), transfer time if applicable, and the arrival bus stop. 3. Perform the same search on team-trafkk.no's bussorakel 4. compare the two results	C04, D03

Test plan 4 – System test

Goals

This test will ensure that the system works as a whole. The GPS system must be able to retrieve the position of the user, and request the web server for a search interface and a map based on these coordinates. The web server handles communication with the client, as well as relaying request of information to the application server, which retrieves this information from the database and external service. The web server must be able to accept this data, and to generate a map containing this information and present this to the user. This test covers module 4 of the system.

Requirements

The following requirements are tested in this test, as they are defined in section 0;

All functional- and non-functional requirements with high priority

Test design

ID(TD01-XX)	Steps	Refs
TD04-01	<p>Title: System test</p> <ol style="list-style-type: none"> 1. Go to the position specified in TC04-01. 2. Perform a search for the keyword specified in the case 3. Select the document occurring at the library defined in the case 4. Record the map returned 5. Perform a search for the same bus at team-trafikk.no, log the result 6. Perform a search for the same book at sru.bibsys.no, log the result 7. Compare the result in step 4 with the ones in 5 and 6. 	All

Test Case specifications

This section defines the test cases that are applied in the previous described test designs. Along with each case is an “expected result”. This is the result that is required in order for the test to pass. Each case carries a reference to the test design that uses it.

Case ID	Data	Expected result	Refs
TC01-01	<63.432813, 10.40519> Bakke Bru	<63.432...,10.405...>, with three decimals accuracy	TD01-01
TC01-02	<63.433178,10.391715> End of Prinsens gate	<63.433...,10.392...>, with three decimals accuracy	TD01-01
TC01-03	<63.427496,10.396693> Nidarosdomen	<63.427...,10.397...>, with three decimals accuracy	TD01-01
TC01-04	<63.432813, 10.40519> Bakke Bru	<i>Username;</i> 63.432813;10.40519; <i>registration Time</i>	TD01-02

Case ID	Data	Expected result	Refs
TC02-01	“internet”	“internet”	TD02-01
TC02-02	User-position: <63.43287,10.391302> Sandgata 1 Document: ISBN: 9173730130 (search for the author)	Address 1: Sandgata 1, 7012 Trondheim, Norway Address 2: Høgskoleringen 1, Trondheim, Norway	TD02-02
TC02-03	User-position: <63.427496,10.396693>, Bispegata Document: ISBN: 0130962775 (search for the author)	Address 1: Bispegata, Trondheim, Norway Address 2: Sem Sælands vei 7, Trondheim, Norway	TD02-02
TC02-04	User-position: <63.433178, 10.391715>. End of Prinsens Gate Document Cognitive neuroscience a reader (ISBN: 0631216596)	The GPS coordinates read from the file must be 63.433178, 10.391715	TD02-03
TC02-05	User-position: <63.427496,10.396693>, Nidarosdomen Document Building Oracle web sites by James J. Hobuss (ISBN: 013079841)	The GPS coordinates read from the file must be 63.427496,10.396693	TD02-03
TC02-06	Departure bus stop: Dokkparken Departure Line 20 Departure Time 2:09 pm Arrival bus stop: Studentersamfundet	The map must correctly indicate the two bus stops. The departure bus stop must indicate line 20 as the departing line, and 2:09pm as the departure time.	TD02-04

TC02-07	Document, isbn: 0130962775 Library: TEK/IDI	Bus stop: Høgskoleringen 6	TD02-05
---------	--	--------------------------------------	---------

Case ID	Data	Expected result	Refs
TC03-01	Keyword: “Internet”	http://sru.bibsys.no/ ...	TD03-01
TC03-02	From: Gyldenløvesgate 25 To: Kongens gate 66	http://team.trafikk.no/ ...	TD03-02
TC03-03	Keyword “Data”	The first 10 results must be identical	TD03-03
TC03-04	From: Fjordgata 40 To: Maskinistgata 1	The two bus departures must be identical	TD03-04

Case ID	Data	Expected result	Refs
TC04-01	Location “Nidarosdomen” Keyword “Internet” Library “Teknisk hovedbibliotek”	The map returned must indicate the user’s position and the location of the nearest buss top. The time of the next departure must be shown. The position of the arrival bus stop as well as the selected library must be correctly shown. The queries on the original sources must have the same result as indicated in the map.	TD04-01

Test log

For each of the test design and test cases, the result is logged in this section. For each test case it must be evaluated whether the test has passed and not (indicated by yes or no in the Pass-column). In order for a test to pass the result achieved must correspond to the expected result for each of the test cases (as defined in the case specification). If a test is not passed, a log is created with a description on what went wrong. When the occurring error is corrected, that specific test is performed gain, along with any tests that could be affected by the changes. When all the tests are passed, all requirements with high priority in section 3 are met.

Test-ID	Case	Registered result	Pass
TD01-01	TC01-01	The mobile application reports the position to be: 63.432813 and 10.40519	Yes
TD01-01	TC01-01	The mobile application reports the position to be: 63.433178 and 10.391715	Yes
TD01-01	TC01-03	The mobile application reports the position to be: 63.427496 and 10.396693	Yes
TD01-02	TC01-04	The log file on the server contains the following entry: testUser: 63.432813;10.40519;11:44	Yes

Test-ID	Case	Registered result	Pass
TD02-01	TC02-01	“internet”	Yes
TD02-02	TC02-02	Address 1: 715, 7012 Trondheim, Norway Address 2: Høgskoleringen 1, Trondheim, Norway	No
TD02-02	TC02-03	Address 1: Bispegata 2C, 7013 Trondheim, Norway Address 2: Sem Sælands vei 7, Trondheim, Norway	Yes
TD02-03	TC02-04	The read coordinates: 63.433178, 10.391715	Yes
TD02-03	TC02-05	The read coordinates: 63.427496,10.396693	Yes
TD02-04	TC02-06	Departure bus stop: Dokkparken Departure Line 20 Departure Time 2:09 pm Arrival bus stop: Studentersamfundet	Yes
TD02-05	TC02-07	The bus stop was the one nearest to the library. A screen shot was taken.	Yes

Test-ID	Case	Registered result	Pass
TD03-01	TC03-01	http://sru.bibsys.no /services/sru? operation=searchRetrieve&version= 1.1&query=internet&startRecord=1&	yes

		maximumRecords=100&recordSchema=info:srw/schema/1/marcxml-v1.1&stylesheet=/util/MARCXML.xsl	
TD03-02	TC03-02	http://team-trafikk.no/team_bussorakel.asp	Yes
TD03-03	TC03-03	<p>Titles returned from BIBSYS SRU</p> <p>Surfactants UK... Distributed database... Beach and nearshore... World-mining data 97... Determination of relative... Hjemmeside for Nosten... Age misreporting... Structures and abstractions... Abortion a tabulation... Dependable network...</p> <p>Titles returned from prototype:</p> <p>Surfactants UK ... Distributed databases... Beach and nearshore... World-mining data 97L... Determination of relative... Hjemmeside for Nostenet... Age misreporting and... Structures and abstractionsan... Abortion a tabulation... Dependable network...</p>	Yes
TD03-04	TC03-04	<p>Bus schedule registered in the application: From Olav tryggvasongate 8. Line 11 at 4:12pm</p> <p>Bus schedule received at team-trafikk.no: The station nearest to Maskinistgata 1 is Strandveien. The station nearest to Fjordgata 40 is Nova kinosenter. Bus 11 passes by Nova kinosenter at 4:12 pm and arrives at Strandveien , 3 minutes later .</p> <p>As Nova kinosenter is located in Olav Tryggvasongate 8, these routes are equivalent.</p>	Yes

Test-ID	Case	Registered result	Pass
TD04-01	TC04-01	<p>A screen shot was logged</p> <p>The bus route was identical</p> <p>The library locations was identical</p> <p>The routes was identical</p>	Yes

E. Appendix E - Bus stop addresses

The following list shows a subset of the bus stops in Trondheim, along with their respective addresses. The addresses are on a form that allows them to be directly displayed in the map.

Bus stop name	Bus stop address
Buran	Mellomveien 1
Dokkparken	Dokkparken 1
Rosenborg skole	Stadsing dahls gate 6
Sigurd Bergs allé	Sigurd bergs alle 18
Festningsgt	Festningsgata
Jonsvannsveien	Eidsvollsgate 23
Høgskoleringen	Eidsvolls gate 2A
Vollabakken	Christian frederiks gt 8
Studentersamfundet 2	Klostergata 28
Studentersamfundet 1	Elgeseter gate 1
Einar Tambarskjelves gate	Elgeseter gate 26
St Olavs hospital hovedporten	Olav kyrresgate
Prinsen kinosenter	Prinsens gate 2
Torvet	Kongensgate 26
Torget	Kongensgate 26
Sentrumsterminalen	Munkegata
Nordre gate	Nordre gate 29
Bakkegate	Innherredsveien 2
Hospitalskirka	Kongens gate 66
Kalvskinnnet	Kongensgate 108
Skansen	Ilevollen 6
Gløshaugen nord	Høgskoleringen 3
Gløshaugen syd	Høgskoleringen 6
Trondheim sentralstasjon holdeplass 10	Trondheim S
Trondheim sentralstasjon holdeplass 11	Trondheim S
Trondheim sentralstasjon holdeplass 13	Trondheim S
Trondheim sentralstasjon holdeplass 14	Trondheim S
Trondheim sentralstasjon	Trondheim S
Strandveien	Innherredsveien 54
Dragvoll	Loholt alle 85
Nova kinosenter	Olav Tryggvasons gate 8

F. Appendix F - Library information file

The following list shows the information registered on each of the libraries that are a department of UBiT. Each record contains the Norwegian name of the library, its English name, the phone number, its address, an explaining note on the address, as well as the latitude and the longitude. The list currently contains information on all of the institute- and section libraries at NTNU and some are not directly used in this prototype. They are included to show that any building can be used as a target for the navigational tool.

ID	Title	Eng.Title	Phone	Address	Address 2	Lat	Lon
ARK/ ArkA /F	Byggekunst, historie, teknologi	Dept. of Architectural Design, History and Technology	95090	Alfred Getz`vei 3	Sentralbygg I, 7 etasje	63.41 787	10.05 324
ARK/ ArkB/ C/D	Byggekunst , prosjektering og forvaltn	Dept. of Architectural Design and management	95080	Alfred Getz`vei 3	Sentralbygg I, 6etasje	63.41 787	10.05 324
ARK/ ArkE	Byggekunst.F orm og farge	Dept. of Architectural Design, Form and Colour Studies	95010	Alfred Getz`vei 3	Sentralbygg I, 2. etasje	63.41 787	10.05 324
ARK/ ArkH	Byforming og regionplanleg ging	Dept. of Urban Design and Planning	95030	Alfred Getz`vei 3	Sentralbygg I, 5 etasje	63.41 787	10.05 324
ARK/ DesA	Produktdesig n		90100	Kolbjørn Hejesvei 2B		63.41 8161	10.40 437
TEK/ Elek A	Elkraftteknik k - Høyspenning steknologi		94210	O. S. Bragstads plass 2 F	Elektroblok k F, 4. etg., Vestfløy	63.41 8785	10.40 0298
TEK/ ElekB	Elkraftteknik k- Energiformo ring og el. anl		94241	O. S. Bragstads plass 2 F	Elektroblok k E, 4. etg., Vestfløy	63.41 8785	10.40 0298
TEK/ ElekC	Elkraftteknik k - Kraftsysteme r		94215	O. S. Bragstads plass 2 F	Elektroblok k F, 4. etg., Vestfløy	63.41 8785	10.40 029
TEK/ Elek D	Teleteknikk	Telematikk/De p. of Telematics	94324	O. S. Bragstads plass 2	Elektroblok k A, 2. etg	63.41 8785	10.40 0298
TEK/ Elek G	Teknisk kybernetikk	Dep of Engineering Cybernetics	94383	o.s. bragstads plass 2	Elektroblok k D, rom 336/Henv.	63.41 8785	10.40 0298

APPENDIX

					Eva Amdahl, 1.etg		
TEK/ ElekJ	Elektronikk og telekommunikasjon		94408	O. S. Bragstads plass 2	Elektroblok A, 4.etg.rom 46	63.41 8785	10.40 0298
TEK/ IDI	Datateknikk og informasjonsvitenskap	Dep. of Computer and Information Science	93440	Sem Sælandsvei 7		63.41 6505	10.40 4524
MAT EM	Matematikk;	Dep. of Mathematical Sciences		Alfred Getz`vei 1	Sentralbygg 2	63.41 7473	10.40 482
TEK/ GEO BIB	Inst.bibl. for Geologi og Bergtekn		94810	Sem.Sælandsvei 1	Bergbygget 2.etg	63.41 6381	10.40 3937
TEK/ Petrol	Petroleumstekn. og anvendt Geofysikk		94925	S. P. Andersens vei 15A	Petroleumst ekn.Senter	63.41 1053	10.40 7925
ARK/ Bygg A/D/ F/M	Konstruksjonsteknikk	Dept. of Structural Engineering	94520	Richard Birkelandsvei 1A	MTI v/Perleporten 3.etg. mot Realf.bygget v/grete Lind	63.41 6349	10.40 8149
ARK/ Bygg B/Ba/ E/G/J /P	Bygg, anlegg og transport	Dept. of Civil and Transport Engineering	94650/94592	Høgskoleringen 7		63.41 3981	10.40 8823
ARK/ Bygg C	Vann og miljøtekn	Dept. of Hydraulic and Environmental Engineering /biblioteket Valgrinda	93876	S.P.Andersens veg 5	2.etg	63.40 948	10.40 7127
TEK/ Varme A/T EK/M ek	Energi-og prosessst.Strømningssteknikk		93566			0.0	0.0
TEK/ Varme B/C/ E	Energi- og prosesssteknikk		93860	Kolbjørn Hejes vei 1	Varmetekniske laboratorier, 4. etg	63.41 8308	10.40 3535
TEK/ Varme D	Energi-og prosessstekn - Termisk		92700	Kolbjørn Hejes vei 2	Varmetekniske laboratorier,	63.41 8542	10.40 467

APPENDIX

	energi				4. etg		
TEK/ Varme F	Energi- og prosesseteknikk. Strømnings teknikk	Dept. of Energy and Process Engineering	93860	Kolbjørn Hejes vei 2	Varmetekni ske laboratorier, 2.etg	63.41 8542	10.40 467
Verk A/C	Produktutvikl ing og materialer		93768	Richard Birkelandsvei 2B	2. etasje i Perleporten	63.41 6273	10.40 8755
Verk B	Produksjons- og kvalitetstekni kk		93800	S.P.Andersens veg 5	All henvendelse til Biblioteket Valgrinda	63.40 948	10.40 7127
Marin A	Marin prosjektering		95572		Marinteknis k Senter (MTS)	0.0	0.0
Marin B	Marint maskineri		95511		Marinteknis k Senter (MTS)	0.0	0.0
Marin C	Marine konstruksjon er		95535		Marinteknis k Senter (MTS)	0.0	0.0
Marin D	Marin hydrodynami kk		95530		Marinteknis k Senter (MTS)	0.0	0.0
Marin E	Maskineri/St yrkelab					0.0	0.0
Marin F	Norges Fiskerihøgsk ole					0.0	0.0
Marin G	Skips- og havlaboratori et					0.0	0.0
MAR INBI B	Bibliotek for marinteknikk		95729	Otto Nielsensvei 10	Marinteknis k Senter (MTS)	63.42 2169	10.43 5564
REA L/Fys ikA	Kondenserte mediers fysikk		93416	Sem Sælandsvei 7	2. etasje	63.41 6505	10.40 4524
REA L/Fys ikB/C	Biofysikk og medisinsk teknologi		93474	Sem Sælandsvei 9	4. etasje	63.41 6172	10.40 3462
REA L/Fys ikD	Teoretisk fysikk		93646	O. S. Bragstads plass 3	2. etasje	63.41 8851	10.40 0706
REA L/Fys ikE/H	Anvendt fysikk og fagdidaktikk		93451	Sem Sælandsvei 9	3. etasje	63.41 6172	10.40 3462
REA	Komplekse		93586	Sem Sælandsvei	1. etasje	63.41	10.40

APPENDIX

L/Fys ikF	materialer			9		6172	3462
REA LKje miA/ G/K	Materialteknologi		51200/9 4057	Sem Sælandsvei 12	Kjemiblokk II	63.41 6836	10.40 238
REA LKje miB	Organisk kjemi		50870	Sem Sælandsvei 8	Kjemiblokk III	63.41 6377	10.40 3922
REA LKje miC/ D/E	Kjemisk Prosessteknologi		94030	Sem Sælandsvei	Kjemiblokk V	63.41 6884	10.40 633
Kjemi F	Fysikalsk kjemi		50870	Sem Sælandsvei 14	Kjemiblokk I	63.41 7029	10.40 2183
Kjemi G	Teknisk elektrokjemi		94057	Sem Sælandsvei 6	Kjemiblokk IV	63.41 6631	10.40 5126
Kjemi H/M	Bioteknologi		93320	Sem Sælandsvei 8	Kjemiblokk III/IV	63.41 6631	10.40 5126
TEK/ IØT	Ind. økonomi og tekn.ledelse		93511	Alfred Getz`vei 1	Sentralbygg 1 , 9 etg.	63.41 7473	10.40 482
10.40 482	Bibl. for Ark/Bygg/De sign		95092	Alfred Getz`vei 3	SB 2, 2.etg	63.41 787	10.40 5324
DOR A	DORA		96028	Maskinistgata 1		63.43 8737	10.42 1695
DRA GVO LL	Biblioteket Dragvoll		96735	Dragvoll		63.41 0371	10.46 5109
GUN NER US	Gunnerusbiblioteket		92205	Erling Skakkes gate 47	inng. Kalvskinns g t	63.42 9315	10.38 5629
KUN ST	Bibl. ved Kunstakademiet		97919	Innherredsv.7	6.etg	63.46 1844	10.91 3434
MED ISIN	MedisinskBibliotek		868495	St.Olavs Hospital	Parkbygget 5.etg	63.42 0047	10.38 8913
MUSI KK	Bibl. ved Musikkonservatoriet		97318	Kjøpmannsgata 42	Olavskv.3.et g	63.43 3121	10.40 3803
REA L	Realfagbiblioteket		95127	Høgskoleringen 5	Realfagbyg get	63.41 4799	10.40 5866
TEK	Teknisk Hovedbibl		95115/9 5100	Høgskoleringen 1	Hovedbygg et	63.41 9277	10.40 4919
VAL GRIN DA	Biblioteket Valgrinda		93876	S.P.Andersens veg 5	2.etg.	63.40 948	10.40 7127

G.Appendix G - ECDL 2009 accepted poster article

The following article that gives a presentation of this master thesis has been accepted as a poster in the ECDL conference of 2009. It will be published in the proceedings for this conference, as a Springer LNCS (Lecture Notes in Computer Science) publication.

Gaining access to decentralised library resources using location-aware services

Bjarne Sletten Olsen and Ingeborg T. Sølvsberg

Norwegian University of Science and Technology
Department of Computer and Information Science
Sem Sælands vei 7-9
NO-7491 Trondheim
NORWAY
bjarnes1@stud.ntnu.no
<http://www.idi.ntnu.no>

Abstract. *The paper describes a prototype that enables library users to use their mobile devices to find the physical location of specific services or objects in a branch of a distributed library. It guides the users to this location using external map services, location-awareness and navigational tools. The architecture of the system is briefly described together with the integrated services.*

Key words: Location-aware applications, mobile applications, library resources

1 Introduction

As the number of mobile phones sold reaches new heights every day, ubiquitous computing is becoming a natural part of the modern life, and new applications emerge, providing functionalities that not long ago would have been unthinkable. Examples of this are WhosHere that enables users to broadcast their profile and to detect friends or people with common interests nearby[1], and Sekai Camera that enables users to tag real-life objects with comments using the camera on the mobile phone along with GPS[2]. Many of these functionalities take advantage of the location of the user. Such location-aware applications are able to provide contextual information, reconfigure, trigger actions or select nearby objects based on this location. In a geographical area there normally are several libraries or library branches, each holding documents on different subjects. Usually the library catalogues are available using a web-based search. The system described in this paper enables library users to use their mobile devices to find the physical location of specific services or objects in a branch of the library, and to guide the users to this location using maps and navigational tools. In this paper we describe a prototype that demonstrates the possibilities at the library of the Norwegian University of Science and Technology (NTNU). An overview of the functionality is presented along with the design of the prototype as well as plans

2 Bjarne Sletten Olsen and Ingeborg T. Spilvberg

for further development. The goal of this project is to demonstrate the possibilities offered by location-aware applications, as well as to show how already existing services can be integrated with this system.

2 The Prototype

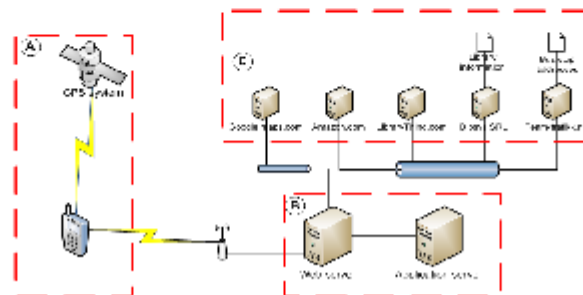


Fig. 1. Overall design with three modules. The Mobile application (A), the server-side application (B) and adapters and services (C).

The system consists of three modules as shown in Figure 1. Module A is the application running on the mobile device, handling communication with the GPS system. Module B is the part of the application running on servers and consists of the web server, the application server and the local database. Module C consists of adapters that handles communication with the external services, ensuring an identical interface for the application server, independent of what service the information is to be retrieved from. The server-side application communicates with the following services: BIBSYS grants access to the Norwegian library catalogue, containing more than 4.3 million titles, including the holdings of NTNU Library's 11 branches[3]. Google maps[4] provide a service that enables the construction of JavaScript-based maps. These constructors accept parameter's representing information that can be included in the map, such as location of Points of Interest (POI). The system is able to retrieve bus schedules from the local bus company Team Trafikk[5]. They provide a natural text search engine that responds to queries on bus schedules. The queries have to be on the form "When is the next bus for point A leaving from point B?". The response is in natural text, and has to be parsed. Amazon.com and Librarything.com are used as sources for book cover images.



Fig. 2. Running prototype with the user, the departure bus stop, the arrival bus stop and the library indicated.

2.1 A walk-through example

A library user is in the city and needs a specific book right away. A search in BIBSYS, using the prototype running on his or hers mobile device, informs the user that the desired book resides in several of the university libraries branches. Information on the status of the document (checked out, available and so on) is part of the BIBSYS system, and access to information is not provided as of now. The user selects the nearest library with the given document. The system generates a map, indicating the last registered position of the user as well as the position of the selected branch library. When the user wants to go to the branch library by bus, the system queries the local bus company on how to get there. A route to the nearest bus stop with connections to the library is then displayed on the map, along with the time of the next departure. The example is shown in Figure 2. When she or he arrives at the library, a map of the library is made available and guides the user to the shelf in which the document resides.

3 The system and technologies

The mobile application was written using C#¹. As newer browsers are already experimenting with the direct access of user coordinates through JavaScript, a stand alone application will not be necessary in the future, and the present mobile-side application is used only for testing. As platform independence is not required for the server-side application, this is written using C#. One of the goals with the development of the prototype is to demonstrate that integration with other existing services is feasible. An adapter is created for each of the

¹ <http://msdn.microsoft.com/en-us/vcsharp/aa336789.aspx>

4 Bjarne Sletten Olsen and Ingeborg T. Sjølvberg

services the prototype needs to communicate with. In order to add new services a new adapter must be created for each service, and the map generation must be updated to handle the new information available.

4 Future Work

As the system described in this paper is a prototype, there are some services that are not yet included, although planned for. One of these is the addition of a news-service, presenting the user with the latest news for the nearby area in map. This can be displayed either on city-, district- or street-level depending on the quality of the location-information associated with the news. As not all mobile devices support the use of GPS, some WiFi based location services have been considered. In the city of Trondheim, where NTNU resides, a localisation service named GeoPos[7] is available. This enables localisation services on all devices with a WiFi card installed. This would add the possibility of indoor-localisation to the application, facilitating the creation of additional services.

References

1. WhosHere, <http://myrete.com/WhosHere.html>
2. Gizmodo, http://www.gizmodo.com.au/2008/09/sekai_camera_turns_on_worlds_balloon_help-2.html
3. BISBSYS Ask, <http://www.bibsys.no/wps/wcm/connect/BIBSYS+Eng/>
4. Google Maps API, <http://code.google.com/apis/maps/>
5. Team Trafikk home page, <http://team-trafikk.no/>
6. JSR 179: Location API for J2ME, <http://www.jcp.org/en/jsr/detail?id=179>
7. GeoPos, <http://www.geopos.no/geopos/>

H. Appendix H – Source code information

Attached to the thesis is a zip file that contains the source code for the prototype developed. The appended source code consists of two parts. One folder contains the source code for the mobile application. One contains the source code for the server side application.

Mobile application (masterGPSWindowsMobile)

In the Debug subfolder an executable file is placed. By transferring this file to a Windows Mobile device, and running it, the application starts submitting the coordinates to a server address hard coded in the application. The address of this server must be updated in the source code to the one running the server side application.

Server side application (LocationHttpServer)

The application can be run directly from Visual studio, or published to an IIS server. The folder consist of one Project folder and one WebSites folder. The source code contains an API-key that has to be changed whenever the server runs on another domain than the one this key was registered for (in Default.aspx.cs). The file paths in the code must be changed to reflect the change of environment.

Begin using the application by visiting the mainMenu.aspx web page.