NTNU – Trondheim
Norwegian University of
Science and Technology

Integration of Map Use and Context-Awareness with Public Transport Data

Margrete Schei Olsen

Master of Science in Engineering and ICT
Submission date: June 2013
Supervisor: Terje Midtbø, BAT

Norwegian University of Science and Technology
Department of Civil and Transport Engineering

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF CIVIL AND TRANSPORT ENGINEERING

Report Title: Integration of Map Use and Context-Awareness with Public Transport Data	Date: June 6, 2013 Number of pages (incl. appendices): 178			
	Master Thesis	X	Project Work	
Name: Margrete Schei Olsen				
Professor in charge/supervisor: Terje Midtbø				
Other external professional contacts/supervisors:				

Abstract:

This thesis examines the integration of maps to distribute public transport information. The topic is based on map usage for journey planning, route path visualization and real-time information retrieval. Additionally, the topic for map usage aboard public transport modes is examined, for distribution of journey information in real-time.

Information distribution within public transport systems has its basis in location-based services, which is an emerging field founded on the combination of information with spatial locations and context-awareness. The common way to distribute real-time information is by information screens at stops, in a textual format. In recent times, several mobile applications has been developed, distributing the same at-stop information.

The emerge of easy access to real-time information has proven to be of high interest of users. People are able to better time their arrival at stops, plan their journeys and they are generally more positive towards the use of public transportation.

In order to examine the topics in this thesis two prototypes was developed, covering the bus system in the city of Trondheim, Norway. The first prototype was developed for the mobile environment for multifunctional information distribution. The other prototype was developed as a web-service for use on public transport modes. In general, this thesis will show how a higher use of maps for information distribution can be implemented within the field of public transportation.

Keywords:

1. Location-based services
2. Transport GIS
3. Context-awareness
4. Real-time data

NTNU
Fakultet for ingeniørvitenskap og teknologi
Institutt for bygg, anlegg og transport

Master thesis

(TBA4925 - Geomatikk, masteroppgave)

Spring 2013

for

Margrete Schei Olsen

Integration of Map Use and Context-Awareness with Public Transport Data

BACKGROUND

The distribution of real-time information, especially through mobile platforms, aims for easy information access for the public and is a prominent topic in the today's society. These systems are for example well suited to promote public transport as transportation mode. The spatial component is an important factor in public transportation, and an issue that should be examined further is the possibility of higher integration of maps for navigation and information distribution in these systems.

TASK DESCRIPTION

The candidate is going to investigate systems for the distribution of real time information with a special focus on the spatial component. Based on the knowledge from existing works, in combination with own ideas, the candidate is going to develop a prototype which shows the possibilities of higher map integration for information distribution.

Aspects to be examined are:

- Technical solutions
- The handling and distribution of real-time data
- Accuracy
- Required information for this kind of service
- User-friendly map based interface

The task should also include a general investigation of the possibilities of higher map integration through web-based systems aboard public transport vehicles.

Start and submission deadlines

The work on the Master Thesis starts on January 14, 2013

The thesis report as described above shall be submitted digitally in DAIM at the latest at 3pm June 10, 2013

Professor in charge: Terje Midtbø

Trondheim, January 10, 2013. (revised: 3.6..2013)

Abstract

This thesis examines the integration of maps to distribute public transport information. The topic is based on map usage for journey planning, route path visualization and real-time information retrieval. Additionally, the topic for map usage aboard public transport modes is examined, for distribution of journey information in real-time.

Information distribution within public transport systems has its basis in location-based services, which is an emerging field founded on the combination of information with spatial locations and context-awareness. The common way to distribute real-time information is by information screens at stops, in a textual format. In recent times, several mobile applications has been developed, distributing the same at-stop information.

The emerge of easy access to real-time information has proven to be of high interest of users. People are able to better time their arrival at stops, plan their journeys and they are generally more positive towards the use of public transportation.

In order to examine the topics in this thesis two prototypes was developed, covering the bus system in the city of Trondheim, Norway. The first prototype was developed for the mobile environment for multifunctional information distribution. The other prototype was developed as a web-service for use on public transport modes. In general, this thesis will show how a higher use of maps for information distribution can be implemented within the field of public transportation.

Sammendrag

Denne masteroppgaven utforsker integrasjonen av kart for distribusjon av offentlig transportinformasjon. Temaet er basert på kartbruk for reiseplanlegging, rute visualisering og innhenting av sanntidsinformasjon. I tillegg vil kartbruk på offentlig transportmidler bli undersøkt, for distribusjon av reiseinformasjon i sanntid.

Informasjonsdistribusjon innenfor offentlig transport systemer har sin basis i lokasjonsbaserte tjenester, som er et fremtredende felt, basert på en kombinasjon av informasjon med romlige steder og kontekst-bevissthet. Den vanligste måten å distribuere sanntidsinformasjon er gjennom informasjonsskjermer på holdeplasser i et tekstlig format. I nyere tid har flere mobile applikasjoner blitt utviklet, som distribuerer samme informasjons som presentert på holdeplasser.

Økningen av lett tilgjengelig sanntidsinformasjon har vist seg å være av høy interesse for offentlig transport brukere. Folk kan lettere beregne ankomsttid til holdeplasser, planlegge reiser, og blir generelt mer positive til bruk av offentlig transport.

For å undersøke temaene i denne avhandlingen er to prototyper blitt utviklet, som dekker buss-systemet i Trondheim. Den ene er utviklet som en mobil applikasjon fungerende som en flerfunksjonell distribusjonskanal for transportdata. Den andre prototypen ble utviklet som en web-tjeneste for bruk ombord offentlige transportmidler. Generelt, vil denne avhandlingen vise hvordan økt kartbruk for informasjonsdistribusjon kan implementeres innenfor offentlig transport.

Preface

This master thesis is written at the Norwegian University of Science and Technology (NTNU), department of Civil and Transport Engineering, in Trondheim, Norway. It was conducted during spring of 2013, and was weighted 30 study points. The project is a result of a personal idea, which was further developed and defined with my supervisor.

I would like to thank my supervisor Terje Midtbø, for his contribution and assistance. Lastly I want to thank everyone that has contributed with ideas and system testing during the development phase of the work conducted related to this thesis.

June 6, 2013
Margrete Schei Olsen

Contents

Abstract **V**

Sammendrag **VII**

Preface **IX**

Contents **XV**

I Introduction 1

1 Introduction **3**

II Background and Related Work 7

2 Background **9**

2.1 Location-Based Services 9
2.2 Real-time Navigation 11
2.3 Usability Engineering 12
2.4 Limitations With Mobile Environments 13
2.5 Visualization 15
2.6 Task Solving With Mobile Map Services 17

3 Related work and systems **19**

3.1 In-flight entertainment . . . 20
3.2 Flightradar24 . . . 20
3.3 Tromskortet Ruteplanlegger . . . 21
3.4 Dit.no . . . 23
3.5 AtB Bussorakel . . . 24
3.6 Real-time Systems in Trondheim . . . 25

III Definition and Development of Prototypes 29

4 Proposed System Concepts 31
4.1 Overview of Concept . . . 32
4.2 Mobile application system . . . 33
4.2.1 Route Path Visualization . . . 33
4.2.2 Journey Planning . . . 34
4.2.3 Real-time information . . . 35
4.3 Mobile Application System Considerations . . . 36
4.3.1 Route Path Visualization . . . 36
4.3.2 Journey Planning . . . 37
4.3.3 Real-time Information . . . 39
4.4 Navigation System on Public Transport Modes . . . 40
4.5 Web-Based Real-time System Considerations . . . 41

5 Limitations and Considerations 43
5.1 Limitations of the Mobile Prototype . . . 44
5.2 Limitations of the Web-based Prototype . . . 45
5.3 Choices and Considerations . . . 46
5.3.1 Choice of Programming Language . . . 47
5.3.2 Choice of API . . . 48
5.3.3 Choice of Map Client . . . 49
5.3.4 Choice of Database . . . 51
5.3.5 Real-time System Service . . . 52

6 Developing Prototypes **53**
6.1 Requirements . . . 54
6.1.1 Mobile Application Prototype . . . 54
6.1.2 Web System Prototype . . . 56
6.2 General Design . . . 57
6.3 Android Class Structure . . . 59
6.4 Architecture . . . 60
6.4.1 Initialization architecture . . . 61
6.4.2 Location Manager . . . 62
6.4.3 Map Client Architecture . . . 63
6.4.4 Geocoding . . . 65
6.4.5 Real-time Service Architecture . . . 66
6.4.6 Google Direction Services Architecture . . . 69
6.5 Best Journey Travel Route Algorithm . . . 74
6.5.1 Validation of the Weighted Overlay Analysis . . . 77

7 Examination of the Mobile Application Prototype **79**
7.1 Initialization . . . 80
7.2 Container Layout Design . . . 81
7.3 Bus Route Path Visualization . . . 82
7.4 Journey Planner . . . 85
7.4.1 Validation of the Journey Planner . . . 87
7.5 Real-time by Bus Stop . . . 91

8 Examination of the Web-based System Prototype **97**
8.1 Test Scenario . . . 98
8.2 Visualization . . . 98

9 Related Problems **103**
9.1 Google Direction Services API . . . 103
9.1.1 Distance Values . . . 104
9.1.2 Routing . . . 105

9.2 Selection of Bus Route 108
9.3 Presentation of Bus Routes 109

IV Discussion and Further Work 111

10 Discussion 113
10.1 The Mobile Application Prototype 114
10.1.1 Route Path Visualization 115
10.1.2 Journey Planning 116
10.1.3 Real-time Information Distribution 118
10.1.4 Fulfilment of Requirements 118
10.2 The Web-Based Prototype 121
10.2.1 Fulfilment of Requirements 121

11 Further Work 123
11.1 Data Distributed from the Web Services 124
11.2 Route Path Visualization 126
11.3 Journey Planning 127
11.4 Visualization on Public Transport Modes 127
11.5 Usability Concerns 128
11.6 Scalability 129

V Conclusion 131

12 Concluding Remarks 133

Bibliography 140

Appendices A1

A Retrieving Departure Data for a Bus Stop A1

B Obtaining the Distance Between Two Locations **B1**

C Method for Selection of Possible Journey Routes **C1**

List of Tables

6.1 Percentage of importance given to the different factors included in the weighted overlay 76
6.2 Validation of the Weighted Overlay Analysis 77

7.1 Nearby bus stops of the departing location 88
7.2 Nearby bus stops of the destination location 88
7.3 Nearby bus stops of the departing location in the right direction . . 89
7.4 Nearby bus stops of the destination in the right direction 89
7.5 Journey data for the two best possible routes 90

List of Figures

3.1 Screen image showing the main page of Flightradar24's webpage . . 21
3.2 Resulting information window when using the journey planner *Tromskortet Ruteplanlegger* 22
3.3 The journey planner in Tromsø visualizing walking and driving (a), and the drawing of lines between waypoints along the route path (b). 22
3.4 Results from a departure-destination journey planner search using *Dit.no* (a), the result after one of the resulting journeys is chosen (b) and the result presented when map view has been chosen (c) . . 23
3.5 Illustration on how to ask a sentence to the AtB's bus oracle, and its response 24
3.6 List of nearby bus stops based on users location in *Bartebuss* (a) and the map view covering bus stops in the application (b) 26
3.7 The delayed map view when panning the map (a) and zooming out (b) in *Bartebuss* 26
3.8 The map view for bus stop selection in *Bussøyet* (a) and *Bybussen* (b) 28
3.9 Initial view of *Busskartet* when loaded in a web browser 28

5.1 An example showing the use of Mapnik for map rendering (OpenStreetMap Wiki, 2010) 50

6.1 The general view of the views related to the three functionalities in the prototype from a user perspective 57

6.2 High-level UML class diagram for all classes in the mobile application prototype 60
6.3 Flow chart for the initialization process 61
6.4 Flow chart for the location changed event 63
6.5 Google Maps Android API architecture 65
6.6 Flow chart of the process of obtaining distance values 70
6.7 Flow chart of the process of obtaining direction paths and parsing them to be drawn on the map view 71
6.8 Flow chart of the bus routes trip path process 73

7.1 The prototype application icon on the testing device (a) and the splash screen when the application is started (b) 80
7.2 Selected tab view for tab 1 (a), tab 2 (b) and tab 3 (c) 81
7.3 Normal state for the location button (a), and its pressed state (b) . 81
7.4 Interaction for the bus route path tab: initial view (a), touch gesture on location icon (b) and with touch gesture on bus stop icon (c) . . 82
7.5 View displaying the progress dialogue while conducting a departure search by selected bus stop (a) and the resulting view when the search is finished (b) 83
7.6 View displaying the progress dialogue when conducting a trip search for a selected bus route (a) and the resulting view when the search is finished (b) 84
7.7 The initial view of the journey planner tab (a) and the same view with departure and destination locations (b) 85
7.8 Informative progress dialogues after selecting journey planner map view in respective order of (a) then (b) 86
7.9 Interaction for the bus route path tab: initial view (a), touch gesture on the departure bus stop icon (b) and with touch gesture on the destination bus stop icon (c) 87
7.10 Initial view for the real-time tab (a), and the view after typing in a few letters (b) 91

7.11 View of the progress dialogue when pressing the nearby bus stops button (a) and the resulting list view displaying nearby bus stops (b) 92
7.12 The progress dialogue when selecting the map view button for bus stops (a), the resulting view (b) and the view after one touch gesture on a bus stop has been made (b) 93
7.13 The progress dialogue while retrieving real-time data (a), and the resulting real-time data for a bus stop search with bus stop in only one direction (b), and with stops in both directions (c) 94

8.1 Proposed zoom level for the web system prototype (a) and the view with a higher zoom level (b) 99
8.2 Visualization of the prototype with higher zoom levels than the proposed best suited zoom level 101

9.1 Displaying resulting nearby bus stops (a), and the same search conducted a minute later (a) for the same location 104
9.2 View showing the sudden end of a route path 106
9.3 Initial view of the city center of Trondheim (a), illustration of the normal route for bus route 5 in direction *Buenget* arriving in the city center (b), and the actual drawn route for the same bus route in the mobile application prototype. 107
9.4 Route direction problems the bus route 5 (a) and bus route 63 (b) 108
9.5 Displaying route selection from bus stop *Gløshaugen Syd* (a) and from *Prinsen Kino* (b) 109

List of Abbreviations

API	Application Programming Interface
JSON	JavaScript Object Notation
LatLng	Latitude/Longitude coordinate pair, in the WGS84 datum
LBS	Location-Based Services
POI	Point of Interest
SOAP	Simple Object Access Protocol
UML	Unified Modeling Language
XML	eXtensible Markup Language

Part I

Introduction

Chapter 1

Introduction

Mobile phones have revolutionized the communication and life style of the modern generation. People have mobile access at almost any time and anywhere, providing them a wide range of opportunities within the mobile network. Mobile networking is in high speed development, and mobile applications are more frequently used for search tasks and navigational use. These tasks are called location-based services (Virrantaus et al., 2001a). Location-based services has originally been based on spatial information gathering for e.g. nearby restaurants and shops, and has in the later years expanded to real-time systems for public transportation.

Efficient information-based systems can help users navigate through the routes of public transport systems faster, make more convenient connections, and thus make the system more appealing. This factor may in turn generate new users. A easily available application system for smartphones providing public transport users navigational task solutions and journey information, may lead to preference of public transport. This reaction may hence reduce the private car use for transportation, and thus contribute in supporting the related issues. By promoting public transportation, related environmental issues to private car use may thus be reduced (Grotenhuis et al., 2007; Hare, 1997; Harrington and McConnell, 2003;

Kane et al., 2008).

The main idea of this thesis is to promote the integration of maps for information distribution for public transport data. The use of maps is increasing, but there is still a lot of work to be done regarding a higher use of maps for information distribution within the field of public transportation. Further, the idea is that the solution will be a contribution regarding the problem of making public transportation more appealing to the people, by better information distribution of journey planning possibilities, real-time information and route directions for all parts of a trip.

This thesis will try to investigate this problem by two approaches. The first approach is to examine the existing literature and systems in order to survey the principles, technology and approaches used. The second approach is to develop a prototype for the mobile environment and a web-based prototype to be used on public transport modes. The motivation for development of prototypes is to investigate how the findings may work in practice, and to experience the possibilities and challenges related to such prototypes. Both prototypes will be based on the bus network in the city of Trondheim, Norway.

The mobile prototype is based on the idea of a multifunctional system supporting all tasks public transport users may request for real-time, route information and journey planning. The web based prototype is believed to shed light on improved information distribution when travelling on public transport mode. The prototypes developed are based on the bus system distributed by *AtB* in the city of Trondheim, Norway. Both prototypes will implement maps and context-awareness for information distribution to the users.

This thesis is divided into five parts. Part I is this introduction, followed by part II which examines the background literature and related work. Part III defines the definitions and development of the two prototypes before part IV and V presents the discussion and suggested further work before the final conclusion are presented.

Part II consists of two chapters, chapter 2 to 3. Chapter 2 gives an introduction to the related literature for use of location-based services and development within the mobile environment. Existing work focusing on transport information and location-based services are presented in chapter 3.

Part III consists of chapter 4 to 9. Chapter 4 introduces the proposed system concepts for the two prototypes developed in this thesis. The related considerations and limitations to be made are presented in chapter 5. Chapter 6 presents the development process related to requirements, design of the prototypes and their architecture, while chapter 7 and 8 presents a walkthrough examination of both prototypes. Related problems found during the development phase and examination of the prototypes are presented in chapter 9.

Part IV consists of chapter 10 to 12. The discussion presented in chapter 10 discusses the final result from the development of the prototypes before the pointers to further work is presented in chapter 11. The thesis is concluded with the final concluding remarks in chapter 12.

Part II

Background and Related Work

Chapter 2

Background

Location-based services are complex systems, and especially when integrated with realtime information data. Several topics are worth examining for the decision-making during the design process of the prototypes. This section will present the major fields which the task related to this thesis cover.

2.1 Location-Based Services

From the emerging capability of mobile GPSs and the mobile network infrastructure to position the terminals on the earth, there have emerged new types of spatio-temporal real-time services, called Location-Based Services (LBS) (Virrantaus et al., 2001a). The first of these services emerged in the mid-1990s, when mobile devices gained the opportunity to integrate global positioning systems, mobile data connections and distribute geographic information (Raper, 2009). Location-based services have similar tendencies within commercial potential as with Geographic Information Systems, which main characteristics is to link spatial data and the vast amount of non-spatial attribute data (Virrantaus et al., 2001a).

Location-based services is the very foundation of applications using spatial positioning. The early solutions based on LBS were car-navigation systems, using GPS receivers to present directions on a map (e.g. the Bosch TravelPilot in 1995). After the year of 2000, the widespread implementation of the general packet radio system (GPRS) introduced the possibility of linking mobile devices to the Internet wirelessly (Raper, 2009), adding real time information to the LBS.

The location-based services are based on the technology related to determine the position of a terminal with a mobile network. This is done with increasing precision on the earth, providing more defined and precise task problem opportunities. Since this emerge, a large amount of effort has been put into the development of mobile services (Von Hunolstein and Zipf, 2003).

The mobile phones of today provides accessibility to internet services and applications on demand. The location services within the mobile environment uses one or more positioning methods to compute the location estimate which is delivered to the device in coordinates of the WGS84 datum. The rapid growing usage of such mobile data terminals generates a huge market and business opportunities for related services (Virrantaus et al., 2001a).

Mobile tour guides have become one of the most popular context-aware application systems, providing information about surrounding environment by filtrating relevant information on location. It is also important to note that GIS is also knowledge on data analysis and visualization. Within GIscience (Reichenbacher, 2001b) mobile cartography is a relatively new field now available in GIS-based applications.

2.2 Real-time Navigation

Realtime information is becoming increasingly more common in modern public transport systems, mostly presented as dynamic at-stop information. Several projects have gained evidence that this kind of information is appreciated by public transport users (e.g. Cassidy and White, 1995; Forsyth and Silcock, 1985; German Federal Ministry of Education and Research, 2002; GoTic, 2002; Grotenhuis et al., 2007; Infopolis2, 1998; Lehtonen and Kulmala, 2002; Nijkamp et al., 1996; Sekara and Karlsson, 1997).

Grotenhuis et al. (2007) argues how travel information is one of the factors promoting public transportation. Various studies have indicated that such information is important and can make a substantial contribution to the overall public transport quality (e.g. Balcombe et al., 2004; Egeler, 2001; Kenyon and Lyons, 2003; Stradling et al., 2001). The motivation behind most of the studies focusing on improving public transportation is the related environmental issues, which may be reduced with a promotion of public transport.

To accomplish such promotion of public transportation, the public transport mode must be clean, reliable, easy to find and use, it must run frequent services, and the trip time should ideally be quicker than by personal vehicle (Kane et al., 2008). Kane et al. (2008) also points out that more importantly, public transport users must be able to plan a particular journey in the time available, and that it is characteristic that even regular users may not know many routes of the system for planning their trips.

In the study conducted by Grotenhuis et al. (2007), they made a survey on eleven different information types that are believed to be desired by public transport users. The five most desired pre-trip information requested found by the study group were: maps with all interchanges, integrated connections in timetables and on signs, quickest route by multimodal journey planner, alternative routes by multimodal journey planner and total travel time. More studies shows similar

positive attitudes towards systems supporting multi modal travel (e.g. Egeler, 2001; Kenyon and Lyons, 2003). Kane et al. (2008) states that most importantly, users must be able to plan a journey ahead.

Dziekan and Kottenhoff (2007) arranges the positive outcomes of dynamic at-stop real-time information into: reduction in wait time (Watkins et al., 2011), positive psychological effects, increased willingness-to-pay, adjusted travel behaviour, mode choice, higher customer satisfaction and better image.

2.3 Usability Engineering

Usability engineering applies to the techniques and concepts assessing the ease of use of a product or a system based on systematic evaluations, inquire methods and system inspection (Good et al., 1986; Nielsen and Hackos, 1993). It is also the key factor in human-machine-interaction, as it is the aspect which usually refers to the quality of the user interface. The International Organization for Standardization (ISO) (ISO, 1992/2001), defines usability as: "a software is usable when it allows the user to execute his task effectively, efficiently and with satisfaction in the specified context of use".

From this definition we derive the three usability attributes: effectiveness, efficiency and satisfaction. All of which will be employed in this study to assess the usability of an interactive web map client. Effectiveness refers to whether or not the users are able to achieve their goals using the system, and efficiency refers to how fast the users are able to accomplish a task. Finally, satisfaction refers to how the users feel about the system and the use of it (Abran et al., 2003; Çöltekin et al., 2009).

Location-based services and realtime navigation systems uses location as a main feature and these are a starting point towards enhanced context-aware services

(Von Hunolstein and Zipf, 2003). There are however other parameters which must also be accounted for in order to design user friendly map systems (Looije et al., 2007; Zipf et al., 2002).

2.4 Limitations With Mobile Environments

The mobile environment sets restrictions and limitations which should be addressed when developing and designing mobile services (Virrantaus et al., 2001b). The mobile environment is subjected to usability issues that can be divided into technical, environmental and social challenges (Gorlenko and Merrick, 2003). These usability problems are mainly related to the technical issues such as display size and bandwidth as well as user centred factors as demographics and preferences (Chittaro, 2006).

Environmental issues cover temperature, light conditions, noise, mobility of the user, distraction and cognitive and physiological constraints of the user, to mention some presented by Looije et al. (2007). One may not know the environment present to the user when using the mobile device or e.g. if the user is walking at the same time.

In terms of social challenges, developers may face usability issues related to privacy, acceptance, adoption issues, comfort and personalization (Looije et al., 2007). Some users may be more accepting towards a system if privacy can be guaranteed, and personalization may enhance comfort and acceptance.

Visualization applications and standards do not scale well to mobile devices, and some limitations do not seem to disappear in the near future due to the size of mobile phones. However, we can see an increasing popularity among tablets which may solve some of the size-related limitations within the mobile environment. Compared to a desktop environment, Chittaro (2006) presents the following

complications for the mobile environment:

- The physical environment varies extremely. A mobile device may be used in lighting conditions that range from light to total darkness affecting the perception of colours and graphics.

- Different or new applications and services are in demand by mobile users, as well as many traditional applications would fail on mobile devices. Most people will e.g. appreciate working on a stationary computer for heavy workload activities, while visual purposes would be better suited on a mobile device for easy accessibility anywhere.

- Mobile contexts make it difficult for users to focus their attention mainly on the device, unlike in an office environment. This is due to the significant large number of external events that must be considered and responded to. For example, one may carry out parallel activities such as walking.

- Serious safety issues may be involved when using mobile devices concurrently with other activities. For example, when using a mobile device when driving a vehicle, walking along a path, users may be or limiting the capability to follow a safe path or mind surrounding dangers.

Heidmann et al. (2003) argues how the limited size issue of mobile screens limits the display of detailed information. For map use Heidmann et al. (2003) elaborate that only a few map objects can be visualized on a particular view, how the usage of labels are restricted in terms of describing semantics of map object, and that structural information such as routes and paths can only be displayed partly or roughly simplified.

2.5 Visualization

In a tourist example, a basic task may be locating the position of a specific sight and to correlate it with his or her actual position. Von Hunolstein and Zipf (2003) explains that to support this cognitive transition, landmarks nearby should also be visualized in addition to the tourist's position and the position of the given sight.

The issues related to limited cognitive resources and safety within the mobile environment presents as an additional motivator to employ effective mobile visualizations which can easily be interpreted by "at-a-glance" behavior (Chittaro, 2006).

Context-awareness gives considerably added value to visualization on mobile devices (Chittaro, 2006). These may include the geographic location of the user and the user's interests by e.g. highlighting personalized points of interest.

Chittaro (2006) presents six check-list points to be considered when developing and designing mobile visualizations:

- Mapping: How is the information visually encoded?
- Selection: Among the data available, what is relevant to the considered task?
- Presentation: How is the visualization laid out on the available screen space?
- Interactivity: What tools are provided to explore and rearrange the visualization?
- Human Factors: Are human perception and cognitive capabilities being taken into account?
- Evaluation: Has the effectiveness of the visualization been tested on users?

Chittaro (2006) argues how all POIs should not be visualized in search related tasks, but a range should be masked out and visualized based on the user's preferences, as they are then not equally relevant to the user's goal. Chittaro (2006) also states the importance of a middle-way constraint evaluation, where too loose constraints will result in an overplotted map while too strict constraints will result in an empty map. Further the visualized POIs may have an additional visualization effect such as an action bar, where this effect visualizes how well the constraints are met according to the given preferences. E.g. if all constraints are met by a POI the action bar is completely green.

The presentation problem in the mobile environment is a serious problem, which has led researchers to propose new approaches, specifically designed for mobile devices. There are particularly two emerging approaches according to Chittaro (2006):

- *Visual references to parts of interest that are outside the view area.* This approach views the detailed view augmented with interactive visual references to the context. Users are hence made aware of parts that are relevant to the current task although the context view is not shown.

- *Intuitive ways of switching among parts of the visualization.* This approach is based on the traditional idea of navigating by scrolling and zooming among different parts in a visualization, but reduces cognitive complexity of the activity.

Level of detail

Looije et al. (2007) argues how it is crucial to show a map with the right level of detail. One can not show a great amount of information on a small screen, and the challenge is to get as much information visible without cluttering it. Looije et al. (2007) describes three important points which are based on the current state

of the user in terms of visualization:

- Nested level of detail: higher level of detail is nested in a lower one (Meng, 2005). For example, a user standing outside a airport wants to go to the city center, and receives an overview of the region and a detailed description of the connection of transport from the airport to the city center.

- Level of detail dependent on location: the preferred level of detail is dependent on the amount of information present in the map (Bozkurt et al., 2005). This can be explained as a map covering a city requires more detail than a rural area.

- Single window details on demand: Due to the screen size limitation, information such a legend should not be shown at all times. Preferably, legends are not demanded if the symbols are self-explanatory (Sarjakoski and Nivala, 2005). However, the users may require a legend at a certain time, and information should hence be available through e.g. a information window.

Overall, use of the Gestalt principles (Wertheimer, 1923) should be kept in mind when developing application systems. These are described as proximity, similarity, closure, simplicity, continuity, good shape, and that common fate can enhance map legibility. As well as other methods such as selective filtering; everything must not be shown, and representation; centering and variable scale.

2.6 Task Solving With Mobile Map Services

Von Hunolstein and Zipf (2003) emphasizes that for location-based services in a tourist perspective; tourists will only use such a service system, if it provides sufficient support for various tasks to reach their goals more easily than without a electronic device. It is also important to notice how tasks and needs in a mobile environment differs from a stationary one, since the scope for potential tasks is

broader, and situation-based tasks may change more rapidly. Focus should hence be on practical and typical tasks.

Tourists especially, are often characterized by neither non-structured nor specific, and may benefit of a changing environment. This nature must be supported by mobile services (Von Hunolstein and Zipf, 2003). Most systems today are developed to support actual on location tasks, while Von Hunolstein and Zipf (2003) argues that mobile systems need enhanced functionality to support whole trip cycles of three major phases: before trip, on site and after trip.

The most obvious tasks related to location-based services are navigational and orientation tasks. These also serve as tools for exploring and planning purposes including searching for points of interests (Von Hunolstein and Zipf, 2003). Reichenbacher (2001a) distinguishes between four groups of user tasks within the mobile environment: locators, proximity, navigation and events. These are considered high-level tasks which can be redefined and further subdivided into several subgroups. Zooming and panning are examples of low level tasks, which are considered used within each high-level task to further adjust the visualization needs by map based interaction.

A typical solution to the navigational and way-finding problem is shortest path routing. This algorithm can be optimal in many cases, but as Von Hunolstein and Zipf (2003) describes, tours should not only show the fastest way from one point to another, but should also offer a structured overview of possible activities.

Chapter 3

Related work and systems

Real-time is a frequently used word these days, and it is mainly associated with real-time information related to public transportation. The real-time systems today are usually represented as information screens at public transport stops, distributing departure information. Similar screens are also found aboard public transport modes, normally presenting next stop destination. These systems may also distribute distance to the next stop measured in time.

In the later years there has also been an emerge in real-time applications, either as web-based systems or as applications within the mobile environment. These systems usually provide the same real-time information as the information screens at public transport stops, and some provides also additional information, such as tube maps for the whole public transport network.

This chapter will present earlier work and systems, related to the work covered in this thesis. The presented systems are based on the most common application systems we know of as of today, systems present in Trondheim, which is the selected environment in focus in this thesis and systems similar to the prototypes developed in this thesis, which are introduced the next chapters.

3.1 In-flight entertainment

Airlines were early on with providing entertainment systems for their passengers. This was primarily for general entertainment through music, videos and similar, which further developed into an interactive flight map (Alamdari, 1999; Frisco et al., 2003). Passengers are enabled interactive flight information through a map portion of the flight path. The map is centered on the airplane icon, representing the location of the airplane. The map usually consist of one or more identifiers (Frisco et al., 2003). The identifiers provided on the map system may be icons, text or other unique identifying symbols, which can be associated with POIs, map locations, or other particular locations or facilities. Passengers may also be presented with a selection of POIs which on request can display related information about selected identifier (Frisco et al., 2003).

3.2 Flightradar24

Flightradar24 provides live air traffic from around the world. This service is offered through a website and as a mobile application for both the Android and iOS platform. The system is based on automatic dependent surveillance-broadcast (ADS-B) technology (Flightradar24 AB, 2013). Flightradar24 AB (2013) describes this technology as :

1. Aircraft gets its location from a GPS navigation source (satellite)

2. ADS-B unit on aircraft transmits signal about the location and destination of the aircraft, as well as other related information such as e.g. travel route, speed, departing airport and aircraft model.

3. ADS-B signal is picked up by a receiver connected to Flightradar24

4. Receiver feeds data to Flightradar24

This system enables users to see flights in real time, and gain information about flights such as: departure and destination airport, speed, type of aircraft etc. Airports are marked on the map as POIs. Google Maps is used as a base map for the visualization.

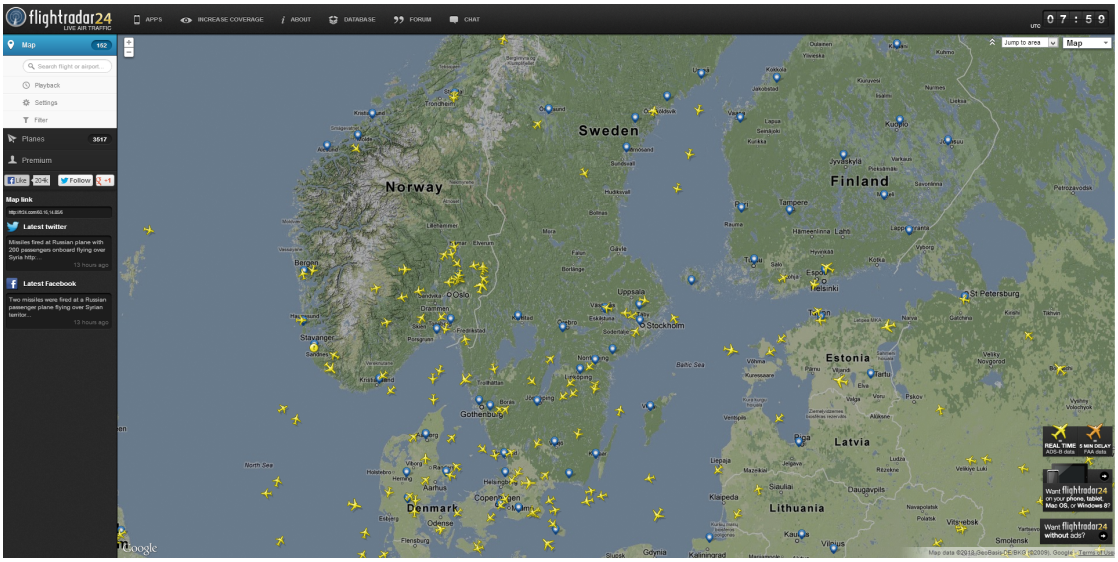

Figure 3.1: Screen image showing the main page of Flightradar24's webpage

3.3 Tromskortet Ruteplanlegger

In the city of Tromsø, the operating bus company has provided a journey planning web-page for bus riders. The journey planner allow users to type in departure and destination either as an address, place or a stop, and retrieve a full journey ahead with walking distances, bus travel time and any route transfers if present.

The journey planner in Tromsø visualize the complete journey with both walking and bus directions as seen in figure 3.2, but the route paths are only based on arbitrary lines drawn on the map, and not actual travel path directions for walking and driving, as seen in figure 3.3.a.

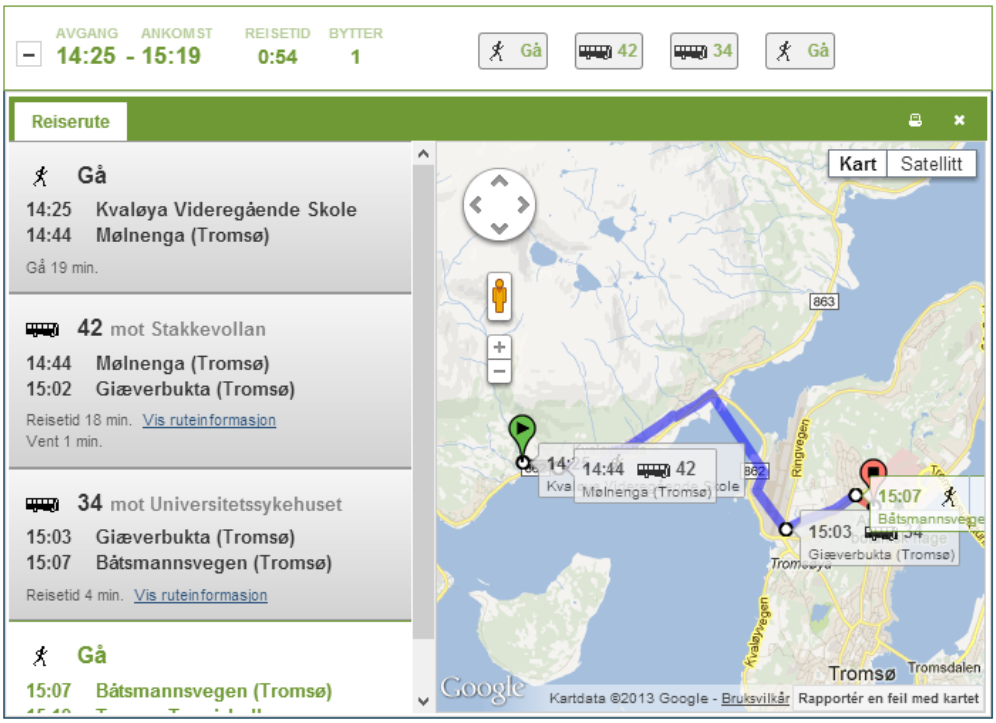

Figure 3.2: Resulting information window when using the journey planner *Tromskortet Ruteplanlegger*

Figure **3.3.a**

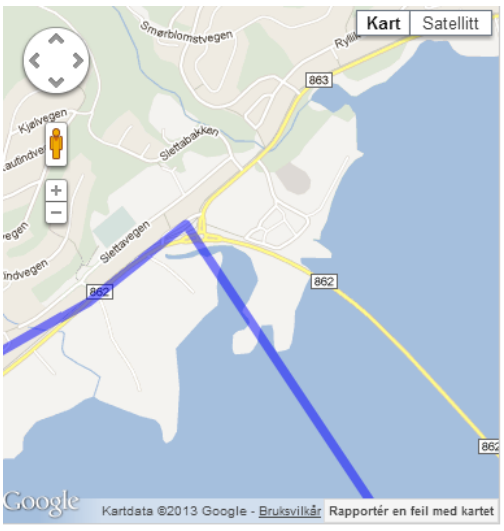

Figure **3.3.b**

Figure 3.3: The journey planner in Tromsø visualizing walking and driving (a), and the drawing of lines between waypoints along the route path (b).

3.4 Dit.no

Dit.no is a web-based route planner and a real-time information system. They are including both public transport and private transport information, supporting the highly requested need for multimodal transport information for planning purposes, as described in the previous chapter. The project was started based on the intention that travellers had to use several websites to gather journey information which is now collected into one by *dit.no*.

The application system allows users to search for a journey and review the results textually or as a map view. The map presented is similar to *Tromskortet Ruteplanlegger*, presenting both walking and transport paths. *Dit.no* also presents the transport path as a simple line with some waypoints for direction correction, based on major stops along the route. The walking paths are improved compared to *Tromkortet* as they follows the road network.

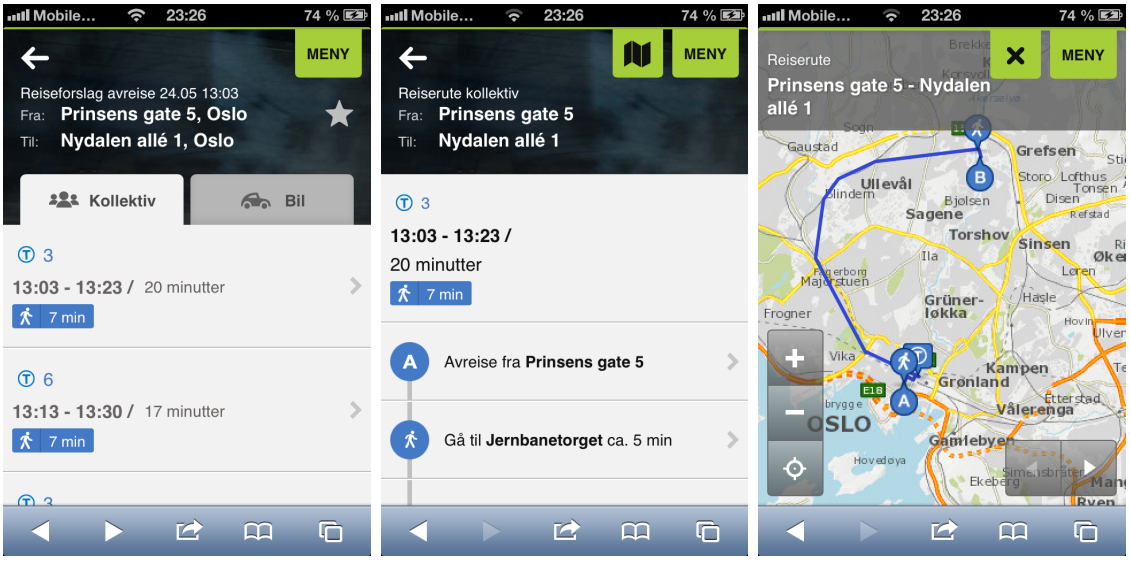

Figure **3.4.a** Figure **3.4.b** Figure **3.4.c**

Figure 3.4: Results from a departure-destination journey planner search using *Dit.no* (a), the result after one of the resulting journeys is chosen (b) and the result presented when map view has been chosen (c)

3.5 AtB Bussorakel

The official journey planning system distributed by *AtB*, the operating bus service company in the city of Trondheim, is a so called bus oracle. Users may ask the oracle for journey information, in form of a sentence. The asking sentence is build up of a departure from a given location or bus stop, to their desired destination. Additional variables can be used, such as departure or destination time or a given date for the search. The result is given on a textual form including departing bus stop, route number, departure time, destination bus stop and scheduled time of arrival at destination bus stop. If the journey planned was based on locations and not bus stops, the result is based on the closest bus stop to the location input.

Figure 3.5: Illustration on how to ask a sentence to the AtB's bus oracle, and its response

3.6 Real-time Systems in Trondheim

Focusing on the city of Trondheim, there are many real-time application systems to be found. There are two real-time systems that stands out for their work, as they have created their own APIs based on *AtB's* open data web service. These are *BusBuddy* (Norangshol, 2011) and *Bartebuss* (Andersen). *Busbuddy* is also a distributor of the real-time data collected by *AtB*, through their API. They are also considered the most popular in terms of API usage. The application related to *BusBuddy* is however not well developed as opposed to *Bartebuss* which may be the most popular real-time system by usage.

Bartebuss is a mobile application for bus riders in the city of Trondheim. The application provides users with several choices for real-time information finding. Users may search for nearby bus stops where results are given in a list view providing name of bus stop, direction (as in to or from the city centre), distance from current position of the user, and an approximate accuracy of the distance measurements. When choosing one of the listed bus stops, users are directed to a view presenting all bus routes available from the given stop and their departures for the next hours. Users may also choose to change the view into a prioritized list displaying the next departing bus stops, or choose to only display departure information for their stored favourite bus routes.

Overall *Bartebuss* is a well-developed mobile application for its purpose, but there is room for improvements especially in terms of map implementation. The developer has chosen Google Maps for map view, but the map is loaded as a web-based map instead of a MapView class or a MapFragment, which is customized for the mobile environment. This makes map rendering slower, and the application uses more time rendering the map on initialization and during panning and zooming. The screen view when panning and zooming the map in the application is shown in figure 3.7.a and 3.7.b. Additionally, the icons representing bus stops and user location, could also been customized for more easily recognition.

Figure **3.6.a**

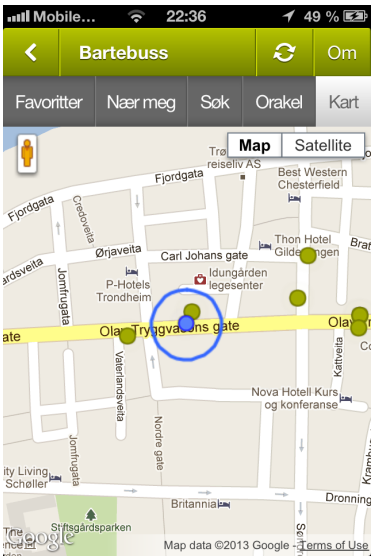

Figure **3.6.b**

Figure 3.6: List of nearby bus stops based on users location in *Bartebuss* (a) and the map view covering bus stops in the application (b)

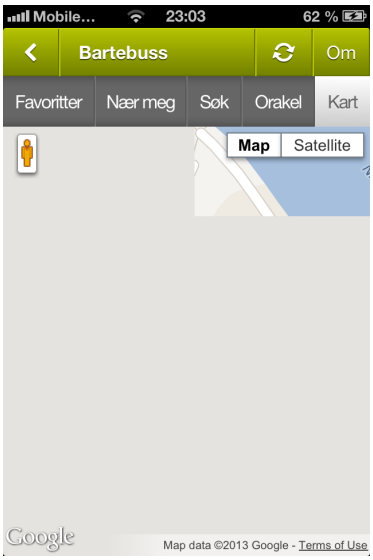

Figure **3.7.a**

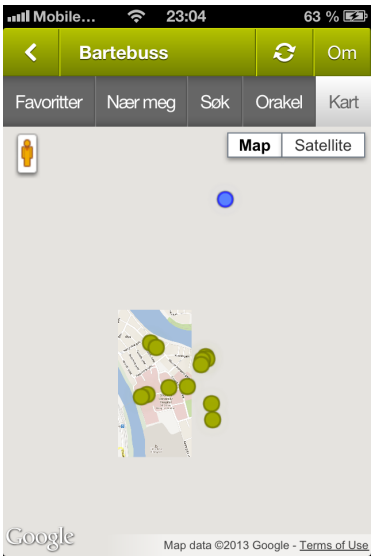

Figure **3.7.b**

Figure 3.7: The delayed map view when panning the map (a) and zooming out (b) in *Bartebuss*

Another shortage in terms of map view in *Bartebuss* is related to journey planning. The application embeds *AtB's* bus oracle and the result is given in a textual format only, as delivered by *AtB*. This could have been improved, as conducted by e.g. the bus company in Tromsø, and *Dit.no*.

Bussøyet is another mobile application similar to *Bartebuss*. Users may choose a bus stop among a list covering the city of Trondheim, choose a nearby bus stop based on the users location or choose a bus stop from the map, as seen in figure 3.8.a. When a bus stop has been chosen, users are presented a view of the next buses departing the bus stop, their route number, their destination direction and minutes left until they will depart the bus stop.

Bybussen also provides users with the option of locating a bus stop through either a search by bus stop names, or through selection from a map view, as seen in figure 3.8.b. The application additionally provides functionality to ask the bus oracle for journey planning purposes. Results from using the bus oracle are presented textually as conducted by *AtB* and *Bartebuss*.

Busskartet.no is a web-application which simulates the position of all public transport modes in the city of Trondheim. The simulation is based on scheduled departure data. The different transport modes are visualized with different symbols to separate them from each other. To obtain information about transport mode and route number represented by the POIs, one must direct the mouse pointer on one of the POI icons. The initial view of *Busskartet* is shown in figure 3.9.

Figure **3.8.a**

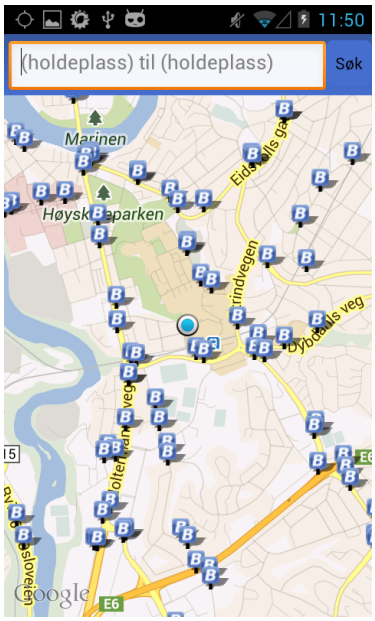

Figure **3.8.b**

Figure 3.8: The map view for bus stop selection in *Bussøyet* (a) and *Bybussen* (b)

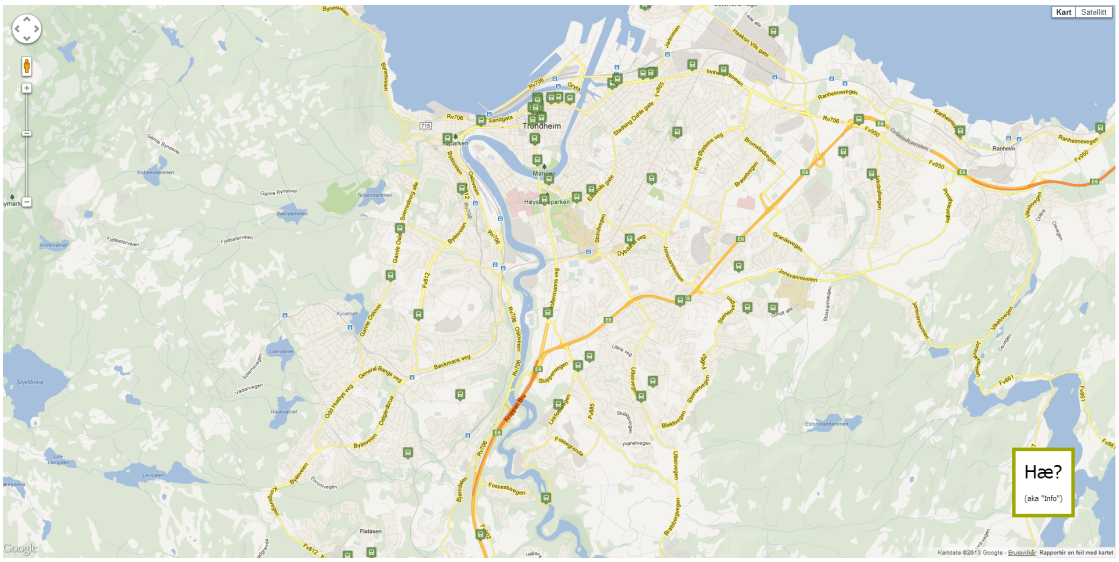

Figure 3.9: Initial view of *Busskartet* when loaded in a web browser

Part III

Definition and Development of Prototypes

Chapter 4

Proposed System Concepts

In order to investigate the possibilities of new methods to distribute and visualize public transport data with a high level of map implementation for information distribution, two prototypes were developed. The first prototype is focusing on the mobile environment and easy-access information distribution, embedding planning functionalities. The other prototype is focusing on the information distribution to public transport users on public transport modes. Both prototypes are based on the bus network by *AtB* in the city of Trondheim.

The concepts of the prototypes are based on the literature from the related fields and the similar systems that exists. The focus will be on integrating map visualization with public transport data for improved navigational and planning purposes.

This chapter describes the motivation for the prototypes presented in this thesis and gives a general overview of the features the two prototypes will implement. The rationale behind development and design choices will be elaborated, based on the background literature presented in part II. The overview presented in this chapter forms the basis for the development design process in chapter 5 and 7.

4.1 Overview of Concept

The main purpose with the prototypes is to examine the possibilities of higher map integration in combination with public transport data, as well as to integrate several information functionalities into one system, to ease the dissemination of information for public transport users. With a mobile application prototype, users are supposed to gain all the real-time and journey planning information they may need, in one single application. The web-based prototype is designed for use aboard public transport modes, and it is supposed to distribute more information, that the system implemented today. The use of map in such a system is believed to help public transport riders navigate to their surroundings when riding with public transport modes more easily.

The use of map integration for navigational purposes is based on the concept of location-based services and the positive outcomes of context-awareness. This features the idea that people navigate better to their surroundings with a map, than by e.g. textual descriptions. The systems are also non-dependent on each other, but may be used in relation to one another. A scenario may be a user planning a trip ahead with the mobile application prototype. When following through with the proposed journey presented by the prototype, the user may more easily keep track of the travel route with the web-based prototype aboard the bus. The idea with the web-based system is also that it can distribute information for all users, non-dependent if they obtain other systems for similar tasks, making users able to navigate themselves to the surrounding areas during the bus ride.

Lastly, the motivation behind the prototypes is influenced by the positive attitude they cause towards use of public transportation. This is indicated, by several studies within the field of transportation, to help towards the private car use problem, which is a major reason for the pollution of the environment. All in all, the prototypes are intended as information distributing systems to public transport users which may have a positive effect on a much greater cause in the long run.

4.2 Mobile application system

There are many systems providing real-time information, and some also combines a map with bus stops for users to choose real time information from a selected bus stop. This presents users with an overview of actual stops. However, there are several improvement that can be done in terms of map integration and information distribution. The first being the use of an embedded map client, which improves map handling and the map tile rendering process. In terms of visualization, many of the systems today lack a more personalized view, and are not using custom icons, which will ease the interpretation process of the POIs presented on the map view.

The use of maps should also be expanded to the journey planning functionality. Most systems enabling planning functionality presents the result in a textual format only. Lastly, no system have been found with the feature of providing users with visualizations the actual direction path of public transport modes in combination with real-time or scheduled data.

One of the main usability issues for public transport users is convenience. A public transport system should be convenient, reliable and easy to use. The users also want to gain real-time information from the use of one system rather than using several. Therefore, the proposed mobile application system is intended as a multifunctional system which may solve several of the problems public transport users may have.

4.2.1 Route Path Visualization

One of the functionalities of the prototype is intended be a visual map representation for departing bus routes and their related direction path along their travel route. The view should hold POIs for the spatial location of the user and all bus

stops in the public transport network. Bus stops should be selectable, to obtain departing route information. This information should thus be usable to obtain direction paths, in form of route paths presented as drawn lines on the map view.

This approach requires real-time information through GPS positions of all bus stops in the network, as well as real-time departure and each bus route's direction path. alternatively, for a less accurate visualization, simulation can be made based on real-time and scheduled data as conducted by *Busskartet*. The functionality further needs public transport information in terms of real-time and scheduled data as well as trip information and information about bus stops location, name and id as well as users location to present POIs on the map view.

4.2.2 Journey Planning

The traditional way of planning a trip from A to B is to describe the draw a path between the two locations or to obtain turn by turn directions between two locations. Another common way is to give the directions in a textual format. The journey planning functionality of the prototype should focus on the use of map for information distribution related to journey planning, and integrate the search algorithm with real-time or scheduled public transport data.

The users may plan a whole journey from their current position or a given address to a final destination which will provide a combination of routing and public transport timing, based on shortest path evaluation measured in total travel time. Shortest path is a commonly used algorithm for journey planning functionalities.

A case scenario may be a user who has several bus stops nearby, where the closest bus stop is rarely visited by public transport modes, while a more distant one is a high-trafficked stop for public transport modes. It is then more suitable to walk a bit further and catch a bus from the more distant bus stop, than to walk to the nearest stop and wait for the bus, if the total travel time is longer if the nearest

bus stop is chosen as departing location.

The journey algorithm proposed in this system will combine the time spent walking to the more distant bus stop with wait time and travel time from that bus stop in the decision making of desired route. A maximum walking distant will be set to avoid inconvenient travel routes. A option for subjective opinions on distance and time may support the usability of the application. The optimization algorithm must also consider if the user can make it to the bus stop in time, else, another route or a later departure must be evaluated.

4.2.3 Real-time information

The real-time information functionality is based on retrieving real-time data in any way desired. As the main focus in this thesis is to use map visualization, the functionality should offer a map view with all bus stops in the public transport network including the users current position for navigational purposes, to which bus stops are near by. The user may then choose a bus stop to retrieve its related real-time information.

The users should also be able to quickly search through the bus stop database to retrieve a bus stop on name recognition, or search for close bus stops based on the users spatial location. Both of these options should present users with the same view presenting real time information for selected bus stop.

The real-time data can be retrieved from a web-service and be parsed to a readable formate before distributed to users through the application interface. The use of such web-services should be implemented to only retrieve data when needed to avoid overloading the web service.

4.3 Mobile Application System Considerations

The development of an application system requires several considerations. One of the main things that is essential regarding the prototype is that it is accurate and easy to use. If this is not the reality, the application will not be used.

As the application is based on real-time data, it is essential that the data distributed is accurate. Related work concerning real-time distribution shows that people are positively affected by this information, in terms of higher usage of public transport services as well as a reduction in wait time at stops. People are timing their arrival at stop better, due to the real-time and scheduled information distribution. An inaccurate information distribution by the prototype will immediately negatively reduce users attitude towards the prototype.

The other main consideration is that the possible tasks presented in the prototype must not be too complicated to use. It is a general discovered phenomena within computer science, that usability is a major field which must be addressed. Users must be able to understand the functionality of each task within a short amount of time, and preferable without much description or guidance.

Lastly the tasks performed by users should not be remarkably time consuming. The processes in the prototype should hence be limited to a minimum by optimization. Another factor that can help the usability while processes are conducted is by always informing users of working processes by implementing progress dialogues.

4.3.1 Route Path Visualization

The main issue concerned with the route path visualization functionality is regarding the usability potential. Users must understand the use of this functionality and how to conduct a route path visualization task. This functionality is a new idea, hence users are not familiar with this type of information distribution. It

is essential that users understand the workflow on how the functionality is to be used.

To help overcome the related usability issues, there is need for the creation of a good way for users to conduct the task, without much description. Each step of the workflow must be intuitive and encourage users to further interaction until the task is fully completed.

The route path visualization tab is intended to be the initially selected tab in the prototype, it is perhaps even more important that is has a high level of usability. The initial impression users get when using the prototype may be affecting the attitude towards the rest of the prototype functionality.

4.3.2 Journey Planning

The journey planning functionality is a complex task requiring several considerations. As for the route path visualization functionality, the use of the journey planner must be intuitive. Another major consideration for this task is the computing time of the best suited route algorithm. The journey algorithm must hence be optimized. If this is not addressed it may reduce the users attitude towards to prototype.

Users must easily understand the workflow for typing in departure and destination addresses to get started. They must also be notified if one of the input values in either the departure og destination field is not valid, and refused to move to the next state, to avoid frustration of non-existing results.

After conducting the input part of the task, users must again intuitively understand that they must choose a resulting view to review the calculated route path. Results from journey planning functionalities are sometimes presented with a map view supplemented with textual descriptions as well, as conducted by *Tromskortet*

Ruteplanlegger. This solution is eliminated in the prototype in terms of being represented as a combination view, and the textual result view is filtrated in its own view, which can be selected after departure and destination location have been chosen. This is based on visualization concepts about not overplotting the map view with information.

Users should always be presented with a possible route as long as the buses are operating. In cases where this is not present users should be notified that there are no route possible and the possible reasons for this. These reasons may be based on the users location. If the users is e.g. in another city, is it given that there will be no bus stop in walking distance available to the user. Also, at night time, there are most likely no operating bus services.

A visualized route must be legible by the users in terms of conducting the actual trip. Users should easily figure out where to go, which bus to take, when the bus is arriving, when to get off, where to get off, and where to go to reach their final destination. Walking and driving routes must be separated to avoid confusion. The other information mentioned must be in reach of users by simple touch gestures. Examination of similar systems presented in chapter 3, shows that there is a common use of information retrieval with touch gestures on a map view.

The last but perhaps the most important consideration is the computation time while the prototype is searching for possible journeys. If this task is time consuming users will not be using the functionality. Examining other journey planners from chapter 3, shows there is a wide variety in search time. The journey planner *Tromskortet Ruteplanlegger* is remarkably fast in computing a complete journey route including both walking and travel paths, as well as transfers if any, in just a few seconds. The journey planner presented by *AtB* in Trondheim on the other hand, may spend up to several minutes while computing a possible journey. The result by *AtB* is presented as a few lines of text, which is also far less impressive than the result presented by *Tromskortet Ruteplanlegger*, adding a more negative attitude towards *AtB's* journey planner. To minimize the search time, the

algorithm for journey calculation must be optimized.

Due to the limited capacity during the work of this thesis, the exception handling of the journey planning functionality has been neglected, as well as handling transfers. This is reasoned as the main goal of the functionality is to present the possibilities of a journey planning functionality, embedding map visualization, in the prototype. With this functionality running there is not many adjustments to be made to expand the systems to the complete functionality potential as first presented.

4.3.3 Real-time Information

The last functionality in the prototype is similar to other real-time applications that exist. The use of this functionality is hence believed to be intuitive by users. We can see the same possibilities for retrieving real-time information by bus stop in the related systems presented in chapter 3. Most systems available are already implementing a map view for bus stop selection, to obtain real-time information for a selected stop. There is however believed to be an improvement in the prototype in this thesis compared to the related systems, as the bus stops in the map view have customized icons. Visualization concepts are important aspects to be addressed to optimize the map view presentation.

In terms of search time spent to retrieve real-time information, there is not much that can be done other than selecting use of service. The possible services available must be considered in terms of usability and computation time, related to obtaining information from the service and parsing the information to a format which can be used for representation. Users of the prototype will be interested in a fast system, providing them on demand real-time data.

4.4 Navigation System on Public Transport Modes

The other system proposed in this thesis is based on the in-flight entertainment which we know from aircraft. As people navigate themselves to their environments better with a map, than with either name of e.g. places or bus stops, the idea is to embed this to public transport modes. This idea is also founded on the fact that public transport information should be available to anyone, and non-dependent on material goods. One should be able to obtain public transport information regardless of personal devices. This may be especially important for tourists and the elderly who are not as familiar with mobile applications as the younger population.

Today, most public transport modes present users with information about stops along the route. This may be done by a tube map, announcement of stops over the speaker system, next stops presented on a display within the transport mode or a combination of these. In Trondheim, *AtB* have installed at least one screen in every bus displaying the next bus stop and the second two after that one in a textual format. The next bus stop is also announced over the speakers. This may be sufficient if users are familiar with the route path, or have been given a certain stop name for where they intend to step off.

The proposed system is based on the current system of *AtB* in the city of Trondheim, by presenting current route, destination and next stop information. The idea further is to include a map view to display POIs such as the bus stops in the public transport network, the spatial location of the bus, and direction paths for the route travelled. The users will then be able to locate their surroundings on the map in real-time when riding the bus, as well as being able to see the surrounding of the next bus stop, and the distance to the next, from a spatial perspective. Users should be informed about the next bus stop by a pop-up information window displaying the bus stop name. As the bus stop is passed, the information window for the departed bus stop is closed, and the next bus stop's pop-up window is

enabled.

The issue with the today's systems, arises when people who are not familiar with the route path of the public transport mode they are on. According to the literature this is evident even for regular public transport users, for most of the routes in the public transport network. People may use only one public transport route to and from work, family, friends or similar. These routes may be well known by the users, but users will also often be in situations when they are have to travel to a unknown location, and therefore needs to take an unknown public transport route to get there. Another common scenario may be users visiting another city, where all public transport routes are new and unfamiliar.

A scenario may be a public transport user who is visiting a friend, and the only information the users have is that he should take bus number 9 and get off around the suburbs high school. The user's friend who he is visiting does not know the name of the closest bus stop, but only the surrounding landmark, the high school. This will in most cases make the user dependent on a GPS and map system to know where to get off, if he does not ask the bus driver to let you know when the bus arrives at the given stop. The proposed prototype is believed to solve this issue.

4.5 Web-Based Real-time System Considerations

The process of distributing real-time information with map visualization is a new concept for public transport modes and there are several considerations that must be made. Even though there is not any similar system known to the author as of today, there are similarities between the prototype and the in-flight entertainment on airplanes, which the idea of the prototype is build on.

The main consideration to be made regarding the prototype is the legibility of the

map view, and its visualization. The map view should have a zoom level which allows users to familiarize themselves with the surrounding areas, but at the same time be able to get some level of detail. Due to the dynamic view of the map, as the bus should always be the center of the map, a too high zoom level will be disturbing for the map viewing. If the map changes to rapidly, users will lose the opportunity to explore the surroundings of the bus and the bus path. The size of POIs should be such that the icons are legible, but without taking up too much space on the map and overplotting it.

One of the issue concerning the web prototype is the information distribution related to the next bus stop. Users must understand the way next stop is being distributed in the map view. It should be intuitive that the next stop is related to the bus stop icon it appears with, for spatial representation of the bus stop.

Finally, the largest consideration to be made is that all information on the screen should be legible for all public transport users on the bus. A bus is of considerable size, and as the information screens on board buses are places just below the roof, this adds to the general distance between user and screen.

Chapter 5

Limitations and Considerations

The development process when developing prototypes includes a wide range of decisions. These are related to everything from design, to functionality and architecture on a general level, down to specified decisions related to e.g. technology used interfaces and algorithms. In this thesis the main decisions rely on functionality and visualization of results.

Time and resources are limited during the development process of the prototypes presented in this thesis, thus enforcing limitations to the complexity and completeness of the prototypes. One limitation being that the mobile application prototype has been emphasized in terms of development. The rationale behind this choice is that the mobile application prototype is a much more complex system which can more fully be realized than the web-based system. The web-based prototype is also meant for a more visual proposal, than a ready-to-use system, on the possibilities of map integration on public transport modes.

The limitations concerned with both systems will be explained in this chapter. Some limitations will also make up some of the basis for further work, described in chapter 11.

5.1 Limitations of the Mobile Prototype

The main idea with the mobile application prototype is to explore ideas concerning a multifunctional application with a high level of map inclusion for data distribution. This indicates a main goal regarding a basic structure with some level of completeness. There are however limitations which make some functions not feasible, and the focus will be on how these could be realized with further development and the implementation of feasible functionality.

The main limitation of the prototype is that the view to obtain journey directions in a textual form from a journey planning search, has been excluded. This is due to the limited time, and that the information which the view will be based on, is distributed in the map view solution. It will hence be easy to perform the same search task, and to distribute the information in a textual way with some further work.

Another major limitation is the handling of transfers in the journey planner functionality. The handling of transfers is neglected, as the implementation will require a lot of more work concerning optimization, especially related to user of web-services and run time of the search algorithm. The idea is that the implemented solution will be a good starting point, and a good solution to show how the functionality can be used with the combination of a map view.

Additionally there are a number of other limitations that has been made due to the limited time and capacity during the work of this thesis. These are:

- Limited focus on cross mobile operating system compatibility. Development is done in an Android environment, which limited the application mobile devices with and Android operating system.

- Optimizing the code and database storage with regard to speed and data searching has been neglected.

- Limited focus on design layout and optimized icon representation.

- Error handling and exceptions are not fully implemented, as the general use and functionality is in focus.

The general idea is to create a prototype to illustrate the possibilities and the principles proposed for the prototype, and not to make a fully runnable system for all proposed functionalities. Some work will be done to make the application legible and easy to use, however a full layout design is outside the scope of this work.

The belief is that this work can shed some light on the possibilities and strengths of integrating useful tasks into one application, the use of maps for navigational purposes and information distribution, as well as the possibilities which can be realized by further work. The development and system testing of the prototype is also useful to uncover problems that may seem to work in theory but fails in a real environment. Such problems may be difficult to reveal in a theoretical approach, but is neglected in this thesis due to the limited time.

5.2 Limitations of the Web-based Prototype

The web-based system is only created for the visual aspect on how the idea of an on-public-transport-mode system could be realized with map integration. Opposed to the mobile environment prototype, the web-based prototype will not have as much functionality implemented, but will highly be based on static data and with limited functionality, for visualization purposes. That the data will be static refers to the input data being related to a given scenario, and not for ready-to-use dynamic functionality. The use of map for navigation is a main factor in this prototype as for the mobile one.

The main reason to limit the implementation of the web-based prototype is the

limited time and resources during the work of this thesis. It is also evident that most of the functionality needed in the prototype will be implemented in the mobile prototype, and thus showing how the similar functionalities in need can be used with a convention from the mobile environment to a web-based one. Another reason is that the prototype cannot be realized to the fullest due to lack of resources, such as GPS data of the buses in the bus network.

As for the mobile application prototype, the web-based prototype will be very limited in terms of layout design and optimized icon representation. The architecture and technology used, needs further implementation and development as well, as the prototype is design for visualization purposes only.

The general idea is to make a simplified prototype, which can be used to visualize the possibilities for a system to be used on public transport modes. The prototype is not to be developed as a ready to use system, but intended to visualize opportunities of map use aboard public transport modes and brighten the idea of further development of such systems.

5.3 Choices and Considerations

When developing prototypes, there are many different choices that may lead to the same results, and there may not be much variation in terms of "the better" choice. A range of considerations must however be made. The fundamental choices and considerations made for both prototypes will be elaborated in the following sections. Topics examined are choice of programming languages, API, map client, database handling and use of real-time information system.

5.3.1 Choice of Programming Language

Choice of programming language for the prototypes make up the basis for further technology and design choices, based on what the chosen language can offer of service and compatibility. One of the main concerns is to choose a language that offers easy map-client handling, to configure the behavior of the client-side. The prototypes are also in need of a server-side for real-time handling.

The mobile prototype is formed for the mobile environment, which makes the Android API the initial choice. This is based on the wide library embedded in the Android API and also it's relation to Google Maps API, which would be a good solution for map-client in both prototypes. Another option could be to develop the mobile application prototype with the iOS API. The iOS API is less documented as Android and as of today it restricts map usage to Apple Inc.'s own map, which is inferior to Google Maps. Based on this, and the authors experience with Android development, Android is the chosen environment platform. The related programming language for the mobile application prototype is thus Java, which is the main programming language used within the Android environment.

The web-based prototype is more open for programming language options. There are many options such as PHP, .net, HTML5, and JavaScript could be suited for most of the required functionality. If the chosen map client should be Google Maps, JavaScript would be a good choice. One of the strengths with JavaScript is that the web-page's client-side editing allows content to be updated without reloading all page components. As the map view will take up most of the view in the web-based prototype, JavaScript is chosen as programming language for the map-client and its overlays. HTML5 will be used to design the remaining layout.

5.3.2 Choice of API

When developing Android applications there are two APIs for use. These are the Android Open Source Project's Android API and the Google Inc.'s Google API. The Google API contains android functionality, hence making the Google API similar to an Android API extension. The Google API will hence be the foundation for the implementation of the mobile application prototype.

The Google Locations Service API, which is a part of Google Play Services provides a powerful high-level framework which is ideal for location services. It includes location provider choices and power management (Android Developers, 2013a), which is desired. As opposed to the Android framework, Google location services also provide new features such as activity detection. The service is also strongly recommended for developers featuring location-awareness by Android Developers (2013a). The Google API supports both Google Maps Android API V2 and Google Play Services which are used as a additional library and added to the Android SDK for accessibility.

The Google Direction API was chosen for use to obtain direction distances and direction polylines for route paths through HTTP calls. The API is developed for both the mobile environment and for websites using Google Maps API. The service have restricted usage limits, which has to be considered when used. The API is subject to a limit of 2,000 direction query requests per day. The API URLs are restricted to 2048 characters, before URL encoding (Google Developers, 2013), which one should be aware of, as many Directions service URLs may involve many locations along a path. In terms of waypoints, the use of these are restricted to 8 intermediate for each request. The limits may be loosened by upgrading to Google Maps API for Business customers.

The same qualities offered by Google within the mobile environment is of desire in the web-based prototype, and the chosen API is thus the Google Maps Javascript API Version 3, which is as of today the official Javascript API featured by Google.

This API allows developers to embed Google Maps their own in web pages. Additionally, the API is designed for fast and applicable use on mobile devices, as well as the traditional desktop browser applications. The Javascript API is a free service as the Android API.

5.3.3 Choice of Map Client

The choice of API is a good indication for the selection of map-client. With the Google Maps Android API one may add maps based on Google Maps data to their applications. The API handles access to Google Maps servers, data downloading, map display, and touch gesture responses on the map automatically (Android Developers, 2013a), making it a superior choice for programming purposes. The API also features the option of drawing overlays on the map such a routes, polygons, icons, changing the map, panning the map and presenting the map in 3D for some cities, to mention some of the features. The chosen map used is thus Google Maps.

The key class when using the Google Maps Android API is the MapView. This class displays the map based on Google Play Services data and handles network requests for additional maps tiles. This is also the class which handles the touch gestures and key-presses. Panning and zooming is automatically embedded with touch control and additional zoom buttons if preferred. The MapView thus provides all the UI elements necessary for user control of the map.

Another positive attribute with the use for Google Maps for map view is the readability of the map. Google Maps map view is used in several applications and web sites, which makes it an attractive map which users are used to. The highlighting of places, landmarks etc. is another strength with Google Maps. An alternative could have been Open Street Map. As with Google maps, Open Street Map also highlight and mark places, landmarks etc, but the use of the API is less intuitive and documented as with Google Maps.

An alternative to the map view could have been the use of a custom made map. This allows for custom and personalized visualization of map objects and vector layers. One may e.g. only draw the main streets which are used by the public transport mode designed for. A custom map may be stored as tiles, and loaded onto the devices SD-card on installation. OSMDroid (OSMDroid, 2012) provides an almost full replacement of the Android MapView class which can easily be used to map handling when using a custom map view. This restricts thus the usage of Google Maps Android API, which only support Google map data.

Another option for map rendering is Mapnik, which is an open source toolkit for rendering maps. The strenght of Mapnik (Pavlenko, 2011) is that is supports a variety of geospatial data formats, and supports flexible styling options for a variety of map design. Mapnik may be used to read ESRI shapefiles, PostGIS, TIFF rasters, .osm files and GDAL or OGR supportet formats (Pavlenko, 2011). An example of the use of Mapnik is shown in figure 5.1.

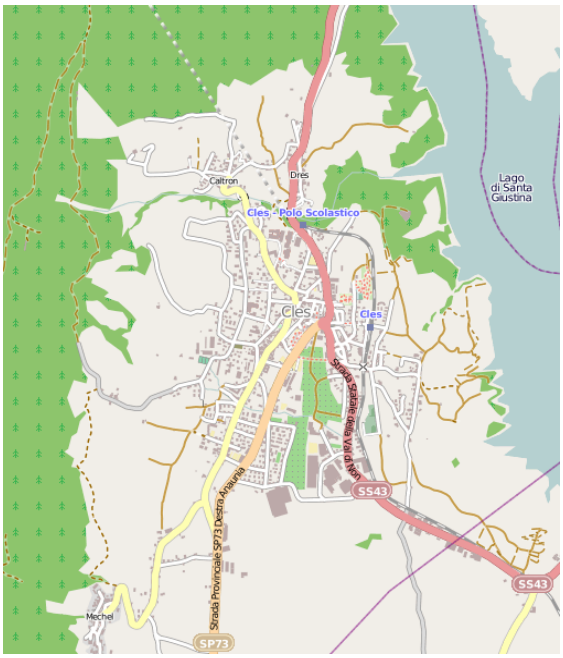

Figure 5.1: An example showing the use of Mapnik for map rendering (OpenStreetMap Wiki, 2010)

Google Maps is also the chosen map in the web-based prototype. The rationale behind this choice is based on the same arguments as for the mobile application prototype. Google Maps API provides a wide range of opportunities for map viewing, overlays and handling. It is also fast and easy to use, as much functionality is embedded in the API script.

5.3.4 Choice of Database

The embedded database used for Android applications is SQLite, which is a client-server relational database management system. As the database is embedded, it is self-contained, serverless and a zero-configuration transactional SQL database engine. The SQLite database is based on public domain code, which makes it free for use for any purpose.

SQLite reads and writes directly to ordinary disk files, and the database file format is cross-platform so one may copy a database between 32-bit and 64-bit systems or between architectures, making SQLite a popular database for Application File Format (SQLite). The SQLite also runs in minimal stack space (4KiB) and small heap (100KiB) which makes it very suitable for memory contained gadgets such as mobile devices.

The mobile application system includes a SQLite database used to store static values for easy data access and to minimize network calls. The main data stored in the database are bus stops, containing bus stop id, bus stop name, longitude and latitude.

For this use, the SQLite database is adequate, but if the application were to store map tiles for custom map rendering, or were to hold more complex geospatial data, a spatial database would be preferred. There is also need for further development of the integration of embedded spatial databases within the mobile environment for Android.

SpatiaLite is a spatial extension to the SQLite database management system, providing vector geodatabase functionality developed by Furieri (2011). The SpatiaLite database system is similar to PostGIS, Oracle Spatial and SQL Server. PostGIS could also be an alternative database if the Mapnik API were to be used, as it supports such database systems.

5.3.5 Real-time System Service

AtB provides access to their real time system for individual use. The system is based on SOAP access which is heavy for mobile platforms. The resulting data from web calls to the service are also difficult to parse to a readable formate. This has been acknowledged by other developers of realtime applications, who has thus developed new APIs which can be used in stead of *AtB's* API directly.

The first realtime API used in the mobile application prototype is developed by *BusBuddy* and distributes data from *AtB* on a more adaptable way. There are two main methods within *AtB's* public API. The first is used to gain data about bus stops and the second is used to retrieve realtime data for a given bus stop. The second API used for network calls was a API created by *Bartebuss*. This API features HTTP calls to gain scheduled departure data and also trip data for a given scheduled departure.

Chapter 6

Developing Prototypes

The development of the two prototypes is based on the concepts, considerations and limitations described in chapter 4 and 5. This development process can further be explained in terms of functionality requirements, architecture, and design choices. The idea is to give a general overview of the data flow within the system and how the prototype is build up, without presenting a step-by-step guide to the development process.

The first sections in this chapter present the functional requirements the prototypes should behold. These are derived from the ideas in the previous chapter, and serves as guidelines for the prototypes. The design overview is examined to give a general idea of the required features and how they intend to function. The architecture design will further elaborate the system flow between elements in the prototypes. Overall this chapter will present the most important components that the prototypes are build up of, and how they relate to each other. As previous, the prototype in focus is the mobile application prototype which is the most developed one. The structure of the web-based prototype will be based on many of the same principles as the mobile application prototype.

6.1 Requirements

A specification of requirements is useful guidelines for the development of a prototype. These specifications indicates the most important functionalities on a primary level, and the less important, but nor superficial functionalities on a secondary level. A functional requirements list may serve as a scenario based walk through of the prototype describing all parts that make up the prototype.

6.1.1 Mobile Application Prototype

Primary Requirements

The primary requirements make up the main functionality of the prototype. These may also be described as the main tasks the user can achieve when using the prototype. The primary requirements are labeled P1, P2, P3 and so on.

P 1: Enable user to retrieve route information.

P 2: Enable users to conduct a journey plan search.

P 3: Enable users to retrieve real-time information.

Secondary Requirements

The secondary requirement elaborates the primary requirements, explaining them in more detail. The secondary requirements are labeled S1, S2, S3 and so on.

S 1: **Departing routes can be retrieved from the map view.**
Users may choose a bus stop from the map view which then presents users with the departing bus routes from the selected stop.

S 2: **Departing routes may be visualized as directions on the map.**
The chosen route may be visualized as a direction line from the chosen bus stop and to the routes end destination.

S 3: **The journey plan may be based on custom departure and destination locations.**
The locations may be either an address, place or the users spatial location.

S 4: **The journey plan is the most suitable route.**
The proposed best journey route is the least time consuming route and the least stressful in situations where more routes presents as good options. The variables in question are walking distances, wait time, and travel time.

S 5: **The proposed best route is visualized with a map view.**
The proposed best journey route is visualized with direction lines for walking distances and direction lines for the bus route path. Walking directions are visualized with blue lines while travel distances are visualized with a red line.

S 6: **The best proposed route is explained in details.**
The proposed best journey route is described in details by selecting the icons along the travel route. The bus icon at departing bus stop holds information about departing bus stop, route to travel with, and departure time. The location icon describes user's position. Destination bus stops describes bus stop's name and scheduled destination time. Final location icon describes the destination location.

S 7: **Realtime by bus stop may be chosen by several options.**
Bus stops selected to retrieve real-time information may be selected by a search for bus stop names, from a list of nearby bus stops based on user's location, or by touch gestures on bus stop icons from a map view.

S 8: **Users may always locate their current location.**
The current location of the user is to be visualized with a location icon at

all times when a map view is presented.

6.1.2 Web System Prototype

Primary Requirements

The primary requirements make up the main functionality of the prototype. These may also be described as the main tasks the user can achieve when using the prototype. The primary requirements are labeled P1, P2, P3 and so on.

P 1: Display route information

P 2: Visualize direction route path on a map

Secondary Requirements

The secondary requirements elaborates the primary requirements, explaining them in more detail. The secondary requirements are labeled S1, S2, S3 and so on.

S 1: **Route information distribution.**
The users should have an overview of common route information such as route or line number, destination or end stop, and the current time.

S 2: **Map visualization**.
The direction route should be visualized on a map with addition POIs such as the real-time location of the bus and bus stops in the public transport network.

6.2 General Design

This section is intended as a general overview of the different components that make up the prototype and their interactions. These are described as the general design of the prototype. This section must not be confused with the actual examination of the finished prototype, which will be presented in chapter 8.

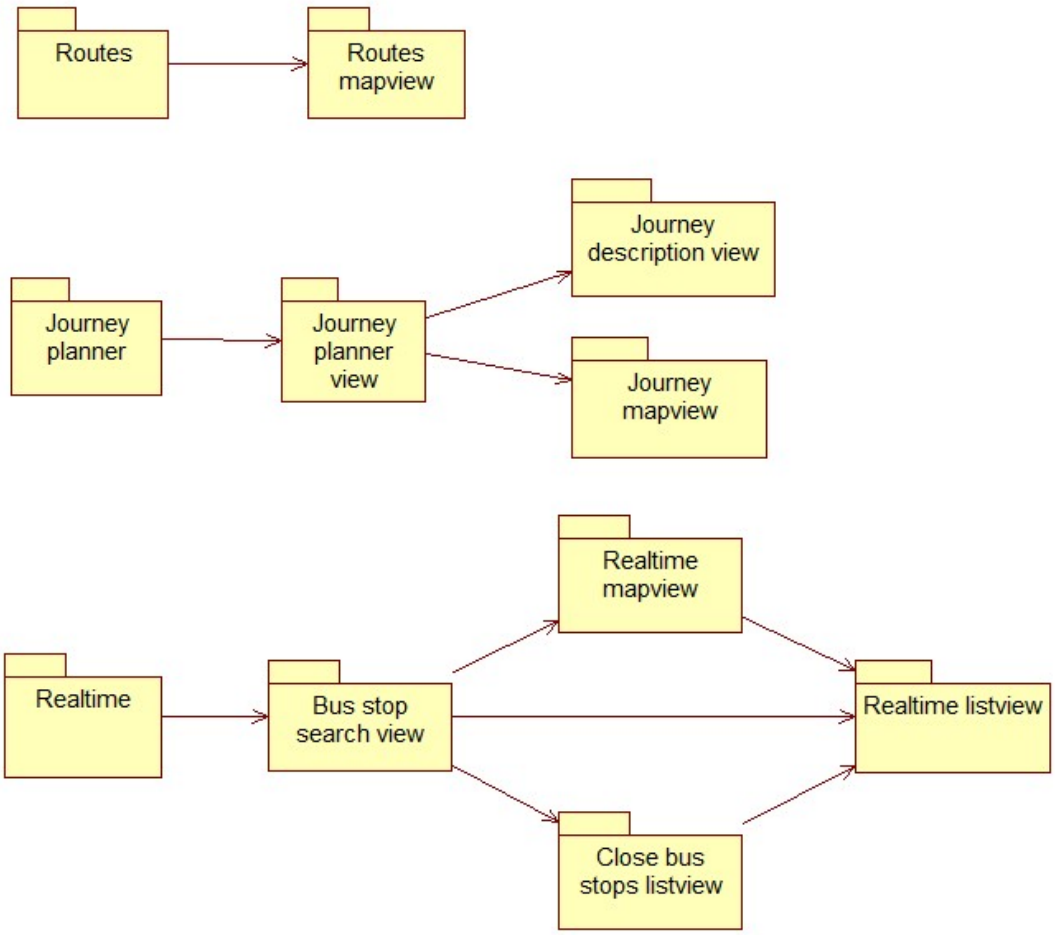

Figure 6.1: The general view of the views related to the three functionalities in the prototype from a user perspective

The main overall design of the prototype is based on Android fragments and the implementation of tabs to navigate between functionalities. These create the basic layout of the prototype, and allow users to switch between the different tabs at any point. The tabs consists of three major parts dividing the three major functionality actions related to the primary requirements.

The android fragment structure is based on one activity that holds fragments. A Fragment (Android Developers, 2013b) represents a part of the user interface in an activity. The use of multiple fragments creates a multi-pane UI, and each fragment exists with its own lifecycle, inputs and events. All view classes in figure 6.1 represents fragments in the prototype, all of which have a related view adding them to the view group.

The main activity class defines the tabs presented in the prototype, their related fragments and the activation of those when selected. It also handles the initialization process when the prototype is first installed on a mobile device and later on, when the application is opened on the device.

Route tab. This tab is the selected tab when the prototype is first initialized, and holds a map view. When selected, the map view fetches the map fragment from its XML layout. The map is then stored as a Google Map object, which is initialized for map editing and overlays. User's location and bus stops are drawn as custom icons on the map, with defined descriptions and related actions when selected. If a bus stop is selected the prototype conducts two network calls to the *Bartebuss* API, one to get the departures from the given stop, and one with the trip id's for all the first departures of the different routes, to get trip information for all departing routes.

Journey planner tab. This tab holds the journey planner view. The location buttons in the view fetches the most recent location data, which is stored in the application, when selected. If an address is typed in to one of the location fields, the application makes a connection to Google Direction Services to receive a JSON

output with related location attributes to the given address. The only attributes stored in the prototype from the network call, is spatial location data given in latitude and longitude.

Realtime tab. The real-time tab consists of three fragments. The initial fragment performs a search through the bus stop table in the prototype's database, for bus stops with similar name fields as the name input. The search is conducted for every letter typed in to the search field. The fragment holding the list view of nearby bus stops, makes a direction search to Google Direction Services to calculate the distance between the location of the user and the nearby bus stops. The map view displays a similar map as in the route tab. All selected bus stop actions navigates to the same view, the real-time list view. This view makes a network call to the *BusBuddy* API to retrieve real-time and scheduled data for the selected bus stop.

6.3 Android Class Structure

The architecture of the mobile application prototype is based on a object oriented approach. The prototype holds 25 classes representing fragments, views, the database and objects. This is made worthwhile examining on an overall view for class representation and the interaction between the classes.

A high-level UML class diagram is shown in figure 6.2. The diagram shows all classes with the exception of one, in the mobile application prototype and the relationships represents which classes triggers or contain other classes or fragments. The MainActivity class is the primary class which is launched when the prototype is initialized. It is also the main context for the whole prototype and the activity manager. The interface for location changed event is excluded in the diagram for simplified legibility of the diagram, as it is implemented by all fragments in the prototype.

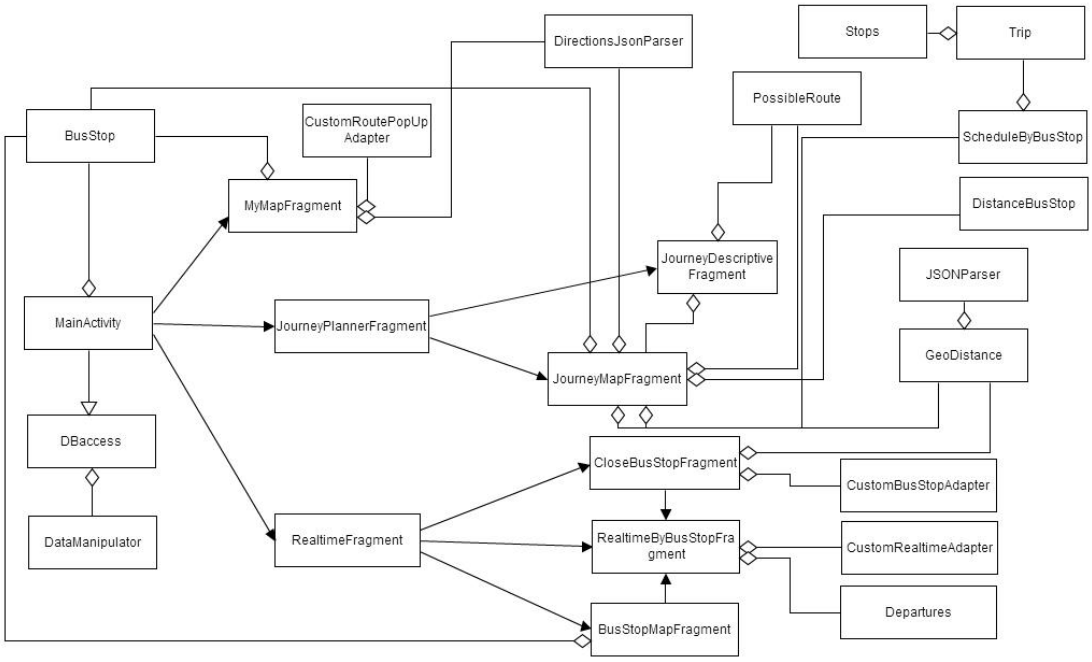

Figure 6.2: High-level UML class diagram for all classes in the mobile application prototype

6.4 Architecture

There are many components that have to interact and work together when developing the prototypes. Again, the mobile application prototype will be the prototype in focus. The architecture that applies to the web-based system will be similar to the architecture presented for the mobile application prototype. The annotation of prototype in the following subsections refers thus to the mobile application prototype.

The main intersection between the different fragments in the prototype is the location data, and the data stored in the database. These are both handled by the prototype's main context and then distributed to the subscribing parts. Another important interaction is between the different parts and the third part interfaces

accessed through HTTP network calls, such as Google Maps API, Google Direction Services API, *BusBuddy* API and the *Bartebuss* API.

The specific architecture of each of the main components of the mobile application prototype will be carried out in detail in the following sections. These are architectures concerning the location manager, the map client, the geocoder, Google Direction Services and the real-time services.

6.4.1 Initialization architecture

When the application is first installed on a device it creates the database and its defined tables. Following is a HTTP call made to the *BusBuddy* API to retrieve a JSON containing all bus stops. These are parsed to variables and each is stored in a bus stop object. Each object is then stored in the database's bus stop table. On later initialization of the prototype the application will check if the database exists and hold values. If this fails, the application repeats the installation process and retrieves the bus stops from the web after the database has been created.

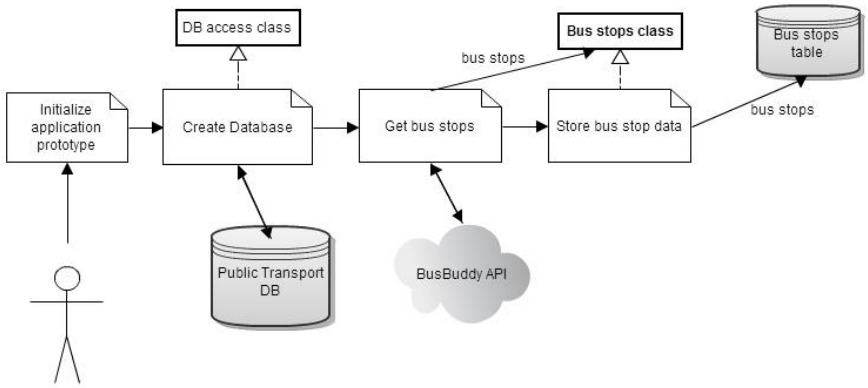

Figure 6.3: Flow chart for the initialization process

Under normal circumstances the database will be stored on the mobile device after installation. The application will then make a SQL call to the database's bus

stop table and store all values in a list holding bus stop object. The initialization process is shown as a flow chart in figure 6.3.

6.4.2 Location Manager

The main activity class is responsible for the location-based service of the prototype. The prototype is designed to handle locations from both the embedded GPS sensor and from a WIFI Network. The initialization of the Location Manager is shown in listing 6.1.

Listing 6.1: Initialization of the Location Manager

```
LocationManager service = (LocationManager)
   getSystemService(Context.LOCATION_SERVICE);
boolean enabledGPS = service
        .isProviderEnabled(LocationManager.GPS_PROVIDER
           );
boolean enabledWiFi = service
        .isProviderEnabled(LocationManager.
           NETWORK_PROVIDER);

Criteria myCriteria = new Criteria();
myCriteria.setAccuracy(Criteria.ACCURACY_COARSE);
locationManager = (LocationManager) getSystemService(
   Context.LOCATION_SERVICE);
provider = locationManager.getBestProvider(myCriteria,
   false);
Location location = locationManager.
   getLastKnownLocation(provider);
```

The Location Manager sets the criteria for location accuracy, and one may also set the preferred location source. The location source can be to either obtain location data through a network or using the embedded GPS of the device.

To retrieve location data the location manager must request locations updates. These include parameters regarding minimum time interval between location updates and minimum distance between location updates.

In order to distribute the location data from the prototype's main activity, a location listener is implemented. All fragments using location data are subscribers to the location listener actions when activated, and de-subscribes if paused or destroyed. When the main activity receives a location update, it creates a location changed event, which distributes the updated location data to all active listeners. The process of notifying listeners when a location changed event is present is shown in figure 6.4.

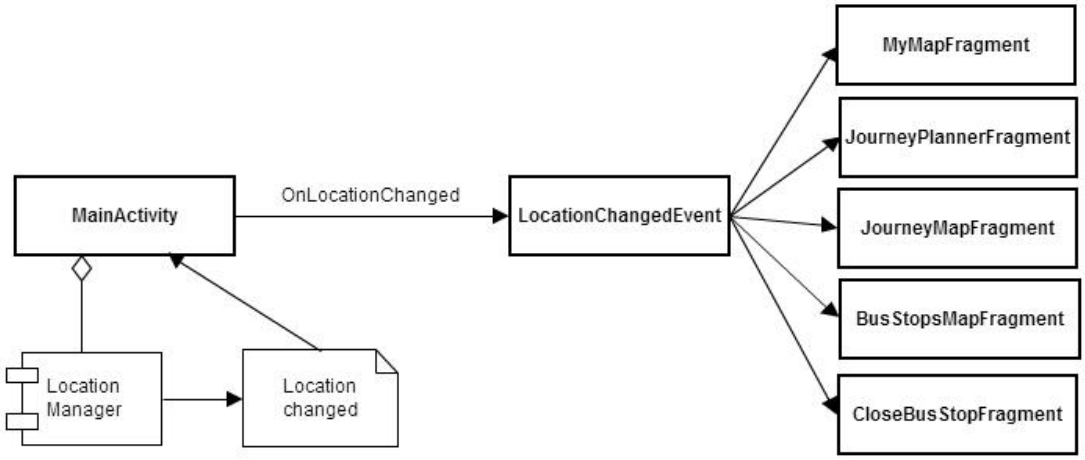

Figure 6.4: Flow chart for the location changed event

6.4.3 Map Client Architecture

As described earlier, Google Maps was chosen for map client in both prototypes. The map handles initialization and map rendering automatically when the map object has been retrieved. The mobile application prototype includes the map through a XML fragment, as provided in listing 6.2, and then load the map in to a Google Map object by pointing to the XML fragment. The web system prototype

loads a map object with a Google Map similarly by including a JavaScript containing map code, and then referring to a Google Map canvas. Listing 6.3 displays the minimum JavaScript code elements in a HTML5 document to display a plain Google Map view.

Listing 6.2: XML for Google Maps Map Fragment

```
<fragment
        android:id="@+id/map_fragment"
        class="com.google.android.gms.maps.MapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
/>
```

Listing 6.3: Code segments for HTML+JavaScript for Google Map view

```
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&
    sensor=true"></script>

var mapOptions = {
  zoom: 15,
  center: new google.maps.LatLng(63.43202, 10.392315),
  mapTypeId: google.maps.MapTypeId.ROADMAP
};

var  map = new google.maps.Map(document.getElementById('map-
    canvas'),
     mapOptions);
```

A general overview on how a mapview in the mobile application prototype fetches a Google Map object from the Google Maps Android API through the web is illustrated in figure 6.5. In the mobile application prototype this process is conducted when the map fragment in the XML file (listing 6.2) is activated to be stored in a map object. For the web-based prototype this is similarly done when the additional script is used to initialize the map variable (see listing 6.3).

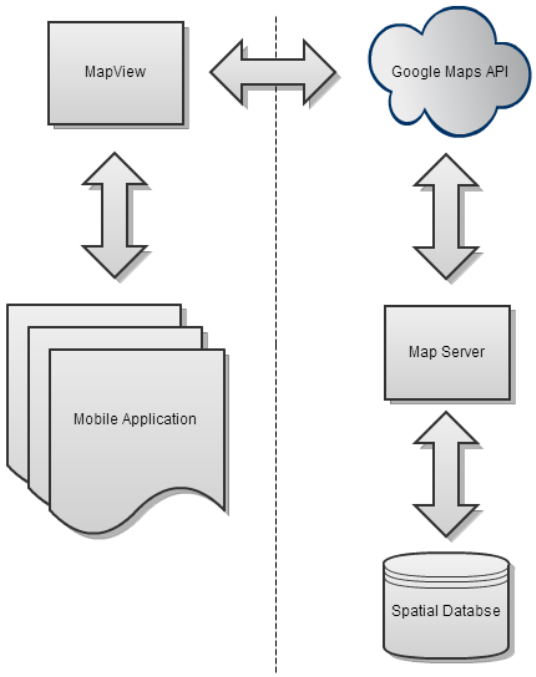

Figure 6.5: Google Maps Android API architecture

6.4.4 Geocoding

The mobile application prototype uses geocoding to handle the transformation of a address search into a location given on at LatLng format, holding latitude and longitude coordinates. To use the Geocoder class successfully one needs a backend services, and this is not included in the core android framework (Android Developers, 2013c)

Listing 6.4: Android Geocoder

```
Geocoder geocoder = new Geocoder(mContext);
        List<Address> addresses = null;

        try {
        // Getting a maximum of 3 Address that matches the
           input text
            addresses = geocoder.getFromLocationName(
               locationName[0], 3);
        } catch (IOException e) {
              e.printStackTrace();
        }
```

The mobile application prototype explained in this thesis covers only the city of Trondheim and its bus network. The geocoding search is thus limited to addresses which only exist in the municipality of Trondheim. This is conducted with the additional code: *if(address.getSubAdminArea().equals("Trondheim"))*. This is conducted because address names in Trondheim may be similar to address names in other cities. As there is not implemented any functionality for users to select among address results, the limitation makes sure the resulting address is within the municipality of Trondheim and the result is then most likely the correct result intended by the user.

6.4.5 Real-time Service Architecture

Similar to the other third party web service APIs, the real-time APIs are accessed through HTTP calls and returned with a JSON result, which is then parsed. To cover all real-time information needed; both the *BusBuddy* API and *Bartebuss* API were used. The resulting JSON from the real-time method calls, holding real-time data, are shown in listings 6.5, 6.6, 6.7 and 6.8

HTTP method calls to the BusBuddy API

The two next listings are results from the bus stop method call and the real-time method call included in the *BusBuddy* API.

Listing 6.5: Resulting JSON from the bus stop method call

```
{
"busStops": [
{
      "busStopId": 100000,
      "name": "Oie skole",
      "nameWithAbbreviations": "Oie skole",
      "busStopMaintainer": "AtB",
      "locationId": "16538544",
      "longitude": 10.254138,
      "latitude": 63.32273
    },
    ...  ]
    }
```

Listing 6.6: Resulting JSON from the real-time method call

```
{
"isGoingTowardsCentrum": true,
  "departures": [
   {
      "line": "3",
      "destination": "Munkegata - M5",
      "registeredDepartureTime": "2011-09-11T22:46:00.000",
      "scheduledDepartureTime": "2011-09-10T22:41:00.000",
      "isRealtimeData": true
   },
   ... ]
   }
```

HTTP method calls to the Bartebuss API

The two next listings are results from the scheduled departure method call and the route trip method call included in the *Bartebuss* API.

Listing 6.7: Resulting JSON from the scheduled departure method call

```
{ "busStopName": "Gloshaugen Nord",
 "schedules": [{
     "line": "5",
     "destination": "Dragvoll",
     "departures": [ {
         "tripId": 1176,
         "operator": "AtB",
         "scheduledDepartureTime": "2011-09-11T22:43:00.000",
         "isRealtimeData": false
       },
       ...
     ]},
   ...
 ]}
```

Listing 6.8: Resulting JSON from the route trip method call

```
{ "line": "3",
  "destination": "Carl Schjetnans veg",
  "lineName": "Sjetnmarka",
  "stops": [{
      "scheduledTime": "2011-09-11T23:05:00.000",
      "destination": "Lade alle 80",
      "municipality": "Trondheim",
      "locationId": "16011041"
    },
    ...
  ]}
```

The second API used for network calls was a API created by *Bartebuss*. This API features HTTP calls to gain scheduled departure data including a trip id which can be used in another HTTP call to get travel path data for a given bus trip. The methods within this API is presented in listing 6.7 and 6.8

Originally *BusBuddy* distributes the same method calls as *Bartebuss*, but during the work of this thesis, the two last methods resulting in listing 6.7 and 6.8 were not available. This is the reason behind the use of *Bartebuss*, which features similar method calls within their API. The only difference in resulting data is the variable names, and that *Bartebuss* distributes real-time data for several hours in the future, while *BusBuddy* only distributes the next five real-time departures.

6.4.6 Google Direction Services Architecture

The mobile application prototype uses one service through the Google Services API for several methods and functionalities throughout the prototype. This is as described earlier, used for distance calculation and route direction service. Both of these are resulting features from the same service, the difference lies in the parsing of the resulting data. The web system prototype uses only the route direction service.

Distance calculation

The distance calculation is made with a HTTP call to Google Services with two locations. The HTTP call is shown in listing 6.9. The method performing the HTTP call holds values for the variables: "startLat", "startLong", "endLat" and "endLong", which is the latitude and longitude values for the departing and destination locations respectively. The Google Services was chosen to obtain distances based on walking paths, and not distance in crow flies, for a correct measure of distance.

Listing 6.9: HTTP call for directions

```
JSONObject jsonObj = JSONparser.getJSONfromURL("http
    ://maps.googleapis.com/maps/api/directions/json?
    origin="+ startLat +","+ startLong +"&destination
    ="+ endLat +","+ endLong +"&sensor=true"+"&mode="+
    mode);
```

The process of obtaining distance values is shown as a flow chart in figure 6.6. The distance values are based on bus stop locations and the location of the user, for the scenario of requesting nearby bus stops. Resulting values are stored in a object combining the distance value with the related bus stop and the user's spatial location.

To limit the search of nearby stops and to avoid unreasonable long walk paths, a maximum distance from the user's location was selected and checked for all bus stops in the database. Only bus stops within the given range were subjected a distance measure. In the prototype this was set to be three kilometers.

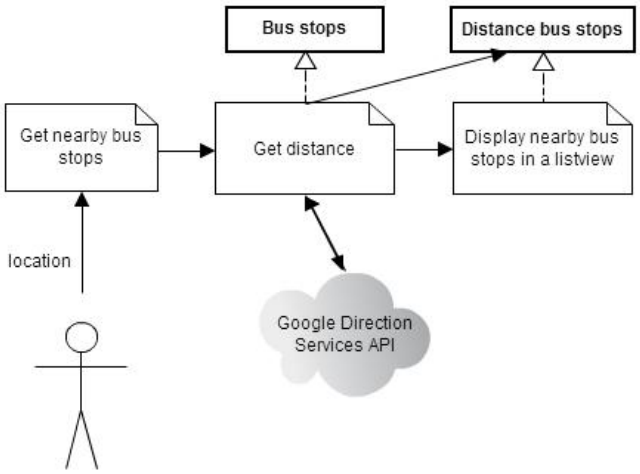

Figure 6.6: Flow chart of the process of obtaining distance values

Route Direction Architecture

The route direction architecture is used by both prototypes. In the mobile application prototype it presents itself as the most complex network service call, in terms of parsing the resulting JSON data. The initial network call to the web service is the same as used for distance calculations shown in listing 6.9. The resulting points are added to polylines which can be drawn as overlays on the map view. The process of obtaining and parsing direction polylines by a given scheduled route trip is shown in figure 6.7.

For bus routes in the prototype we are interested in a specified route made up between bus stops along the route's trip path. A direction path between start bus stop and final bus stop along a route, may result in a different route than the actual one travelled by the bus, as bus routes are usually not made up of the shortest route possible between two location. To draw polyline directions for a bus route, the bus stops along the route are looped and the direction route between each bus stop and the next are used as departure and destination locations. Together these route brackets make up the total travel path of a bus route. This practice avoids the limitation with the Google Map service when retrieving directions for lengthy paths, as the bracket paths are fairly short.

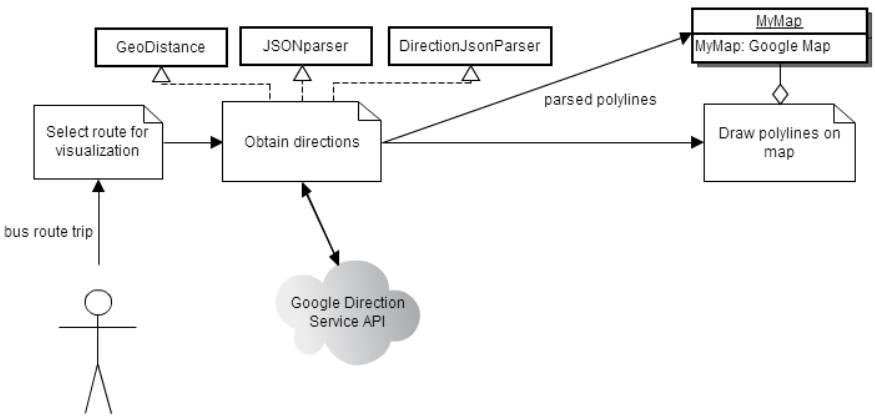

Figure 6.7: Flow chart of the process of obtaining direction paths and parsing them to be drawn on the map view

The decoding of the polylines is shown in listing 6.10. The method takes the polyline string as input and returns a list with location based points which make up the polyline on the format of LatLng. LatLng is a location object holding latitude and longitude data in the WGS84 datum.

Listing 6.10: Method to decode polylines from a direction JSON

```
private List<LatLng> decodePoly(String encoded) {
    List<LatLng> poly = new ArrayList<LatLng>();
    int index = 0, len = encoded.length();
    int lat = 0, lng = 0;
    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (
           result >> 1));
        lat += dlat;
        shift = 0;
        result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (
           result >> 1));
        lng += dlng;
        LatLng p = new LatLng(((lat / 1E5)),
                ((lng / 1E5)));
        poly.add(p);
    }
    return poly;
}
```

Bus Route Travel Path Architecture

The general process of visualizing a bus route's travel path consists of five parts and three HTTP calls to web service APIs. The user selecting a bus stop triggers the process which is based on the id of the chosen bus stop. This id is used in the HTTP call for the *Bartebuss* API, which returns scheduled departure data. These are stored as departure objects and then iterated through. For each departure, the object's trip id is used in another HTTP call for the *Bartebuss* API to retrieve trip data. The trip data is stored as trip objects containing trip information such as direction, departure time and bus stop objects which make up the actual travel path.

The process of displaying a bus route's travel path on the map view is shown as a flow chart in figure 6.8. The process of obtaining and decoding polylines has been simplified in the flow chart, as the process is fully covered in figure 6.7.

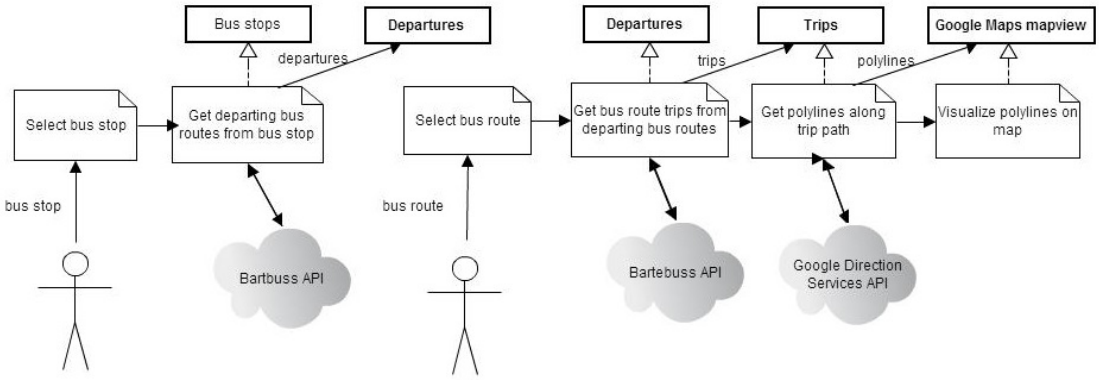

Figure 6.8: Flow chart of the bus routes trip path process

6.5 Best Journey Travel Route Algorithm

To optimize the complete travel route, a custom algorithm was created to search through the quickest travel route from start to destination with a shortest path and a weighted overlay evaluation. Shortest path is defined as the shortest path possible between two nodes. In this algorithm the nodes are the departure and destination locations.

Possible Routes Calculation

There are several steps conducted to find possible routes in the journey planning algorithm. The first is to locate nearby bus stops to the departure and destination location. This is conducted with the same approach as presented for the close bus stop fragment in the real-time tab.

The nearby bus stops of the departing location is then looped, and for each stop, scheduled real-time data is retrieved from the real-time API. For each resulting departure, a trip call is made to the real-time API and the scheduled data with related trip information are stored in the application context.

All scheduled departures are then looped through and each are checked if they contain one of the nearby bus stops of the destination location in their trip list, containing bus stops along the route path. If there is a match, the current time and the departure time of the scheduled bus is then compared with the estimated time spent walking from the departure location to the departure bus stop. If the time spent walking in less than the time before the bus departs, the user should make it to the bus stop before the bus departs. The route is then considered to be a possible route, and the necessary data is stored in the context as a possible route object.

Shortest Path

In order to make a shortest path search, the travel values of the possible route variables must be converted to the same scale. This is conducted through reclassification of the values. The chosen scale is selected to be time in minutes. Distances from locations to bus stops were reclassified with a walking speed of 5 kilometers per hour, based on the comfortable walking range presented by Bohannon (1997). He found that the slowest comfortable walking speed were about 4.58 kilometers per hour for women in their seventies, while the fastest were about 9.12 kilometers per hour for men in their twenties. The fastest comfortable speed for women was however only 6.30 kilometers per hour(Bohannon, 1997), and 5 kilometers per hour was hence selected as a speed which would suit most travellers as a comfortable walking speed. A full optimization of walking speed is outside the scope of this thesis.

Wait time for bus and travel time with bus are given as time stamps (e.g. 17:40), and were converted to minutes based on the difference between two time stamp. An example of such a difference is the departure time of bus from departing bus stop and the arrival time of bus at the destination bus stop.

All of these time measures are added together to a total time variable. The possible routes are then searched through to find the routes with the shortest total travel time. These are selected as a selection of the best possible routes.

Weighted Overlay

Weighted overlay is commonly used to solve multicriteria problems such as suitability modeling and optimal site selection. This is because various factors assessed for suitability have different value scales and do not allow for direct addition. Suitability models may also have factors of different importance.

To introduce how personalization can be used in the journey planning algorithm, a weighted overlay analysis is implemented in the prototype. It is based on the selection of the best suited possible routes. These are chosen for optimizing of the comfort of the travel route, as walking is more strenuous than riding a bus or waiting for a bus.

The different influence weight values were selected based on their importance, where bus travel time and wait time at bus stop were given most influence, as these were the most important factors in terms of a comfortable journey. The influence distribution is presented in table 6.1.

Table 6.1: Percentage of importance given to the different factors included in the weighted overlay

Factor	**Influence weight %**
Time spent walking from departure location to departure bus stop	10 %
Time spent waiting for bus at bus stop	40 %
Time spent travelling with bus	40 %
Time spent walking from departure bus stop to final destination	10 %

The calculation of the total suitability, based on the equation by Eastman (2001), is the sum of each factors individual suitability. The individual suitability is a measure of the multiplication of the factors values and its influence weight in percentage. The suitability equation is listed below in equation 6.1

$$S = \sum w_i x_i \tag{6.1}$$

Where,
w_i = weight values of the ith factor

x_i = influence percentage of each factor

S = Suitability index

In this journey algorithm we are interested in the best suitable travel path which is both time saving and comfortable to conduct. The final equation for each possible path is calculated using equation 6.2

$$S = DT1_f DT1_c + WT_f WT_c + BT_f BT_c + DT2_f DT2_c \tag{6.2}$$

Where DT1 and DT2 represents distance walking time from departure location to bus stop and distance walk time from bus stop to destination location respectively. WT represents wait time for bus to arrive, while BT represents bus travel time. The subscript letters 'f' and 'c' represent respectively the influence weight for the time factor and its value given in minutes.

6.5.1 Validation of the Weighted Overlay Analysis

To test the weighted overlay several possible routes were created and tested with the influence weights selected. An example of two of the test routes in shown in table 6.2.

Table 6.2: Validation of the Weighted Overlay Analysis

Walking	**Waiting**	**Bus**	**Walking**	**Total Time**	**Total Weight Value**
6	2	13	10	31	2.6710
3	5	13	10	31	2.8548

The two routes presented in table 6.2 have the same total travel time, but as difference in the time spent walking compared to waiting for a bus or riding a bus. The first route has a total walking time of 16 minutes, compared to 13 for the other route. Using the weighted overlay, the second route obtains the highest weight value, and is hence selected as the best possible route, based on the given preferences. Based on the idea behind the algorithm, that users would prefer waiting instead of walking, we can validate the use of the weighted overlay.

Chapter 7

Examination of the Mobile Application Prototype

The purpose of this chapter is to give an introduction to the final prototype and how it works. This is done with a walkthrough of the prototype, examining all user interactions possible. The overall walkthrough is based on the view diagram presented in figure 6.1.

The walkthrough starts with an examination of the functionality tab initially selected, and its containing functionality. The first tab is the bus route path visualization. Then the same is conducted for the next tab, the journey planner. A selected route from the testing of the functionality will be presented for validation of the algorithm. Finally the walkthrough will cover the real-time tab, and examine all the three possible ways to retrieve real-time data.

The total examination will give an indication on how fulfilling the prototype is, based on the requirements presented in chapter 6, and how the various architectures and implementations work in the final prototype.

7.1 Initialization

When the application is first installed on a device, the prototype must go through a first-time initialization process. To avoid the application from just "hanging" during this process, a splash screen was implemented. The splash screen, which can be seen in figure 7.1.b, was added as a theme, instead of a time set screen. This means that the screen will last for as long as the application needs to initialize the database, store values in it, and fetch the map fragment from Google Maps API. When these tasks are finished the view is released from the themes splash screen.

On later initialization of the prototype, when it is already installed on the device, the splash screen will naturally last for a shorter time. The splash screen will however still be visible during some initialization tasks, which has to be made on every application start.

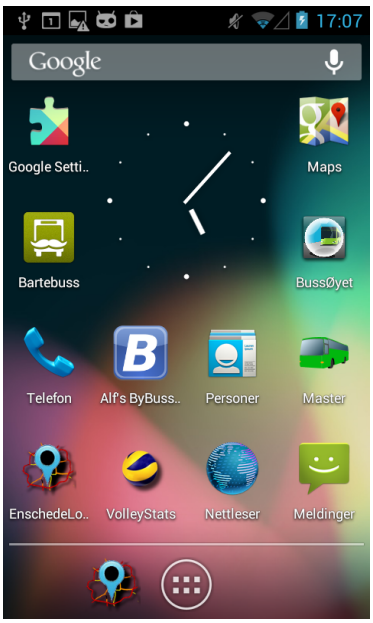

Figure **7.1.a**

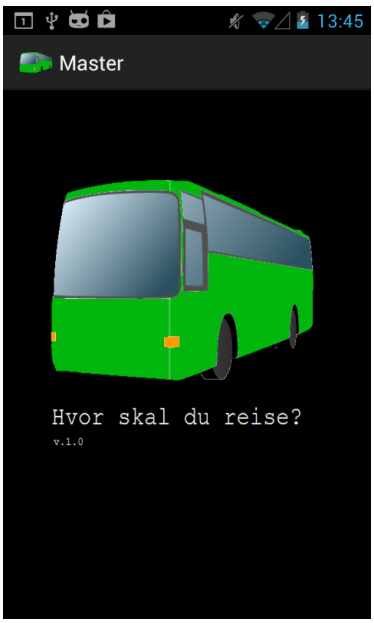

Figure **7.1.b**

Figure 7.1: The prototype application icon on the testing device (a) and the splash screen when the application is started (b)

7.2 Container Layout Design

The layout container consists of two upper rows in the testing device's window. The first row is an actionbar. The action bar is usually used for global navigation or information distribution. For example may this row contain an exit button to end the application, a search button to search through contents or a about button.

The second row is a tab bar for navigation between the main fragments of the application. The selected tab has an underline of a blue line to notify users about which tab is selected at all times, as seen in figure 7.2.

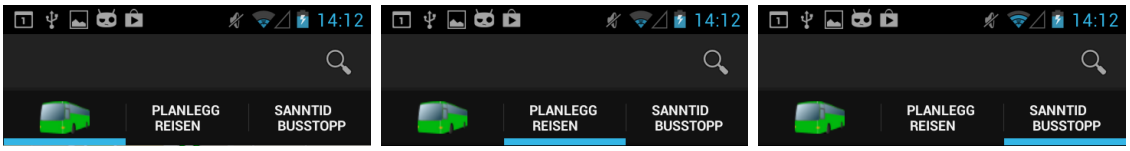

Figure **7.2.a** Figure **7.2.b** Figure **7.2.c**

Figure 7.2: Selected tab view for tab 1 (a), tab 2 (b) and tab 3 (c)

For all tasks conducted in the prototype which changes the prototype's view, the view is featured with a progress dialogue. This freezes the application while the dialogue is visible, and informs users about the task being conducted.

When a button is pressed by a touch gesture, it is displayed with a blue background colour to inform users about the user-button interaction. The normal and pressed state design of the location buttons in the journey planner tab is shown in figure 7.3.

Figure **7.3.a** Figure **7.3.b**

Figure 7.3: Normal state for the location button (a), and its pressed state (b)

7.3 Bus Route Path Visualization

The initially selected tab is the tab for the bus route path. The initial view is a Google Maps, map view, with bus stop icons and user location icon. The map view is initially centered on the user's location and zoomed in to zoom level 15. This zoom level gives a fair overview of the surrounding bus stops, without overpopulating the map with icons as seen in figure 7.4.a. The views in figure 7.4.b and 7.4.c are visualized with one zoom level lower than initially as shown in figure 7.4.a.

The map view is presented with zoom buttons, but the map may also be zoomed and panned with touch gestures. Additionally a location button is added to the view. The location button allows users to zoom and center the map view to their location at any time, as conducted on initialization of the functionality.

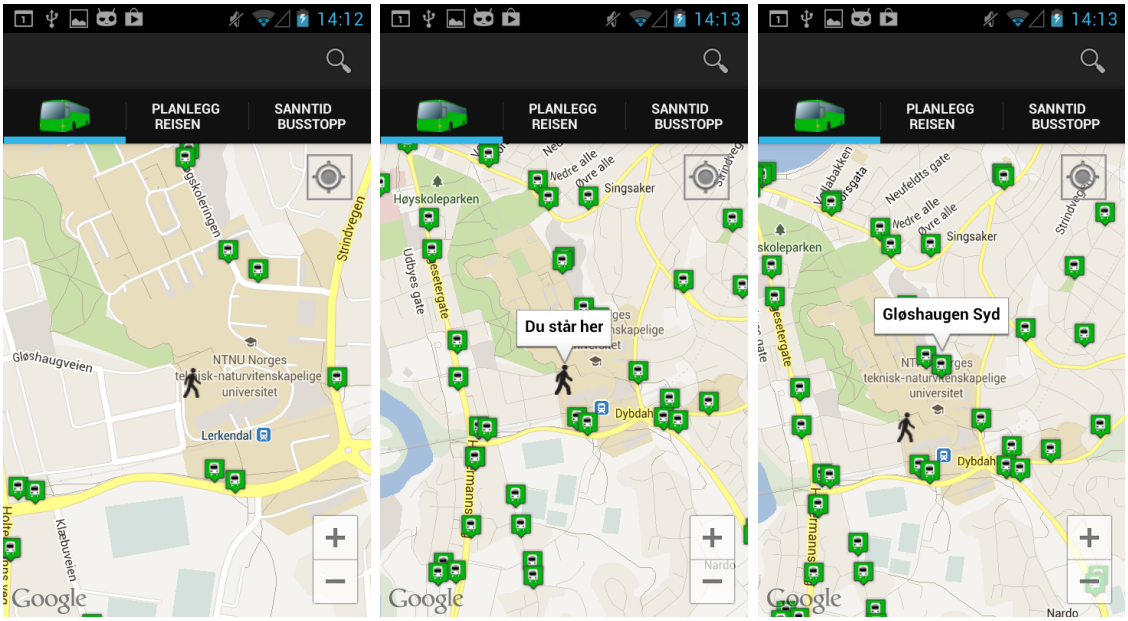

Figure **7.4.a** Figure **7.4.b** Figure **7.4.c**

Figure 7.4: Interaction for the bus route path tab: initial view (a), touch gesture on location icon (b) and with touch gesture on bus stop icon (c)

Touch gestures on POIs presents users with textual information about the selected icon. A touch gesture on the location icon informs users that they are located at that location, as seen in figure 7.4.b. This is implemented for informational purposes, as all users may not understand the icon and its purpose on the map view. Touch gestures on bus stop icon informs users about the bus stop name, which can be seen in figure 7.4.c.

Users may conduct two touch gestures related to the bus stop icons. A second touch gesture, on the selected bus stop's pop up window, performs a departure search. While the prototype is searching for departing routes from the bus stop, the application features a progress dialogue informing users of the search process. The progress dialogue is shown in figure 7.5.a. The use of progress dialogues is based on good user-system interaction methodology, to always inform users of processes and interactions.

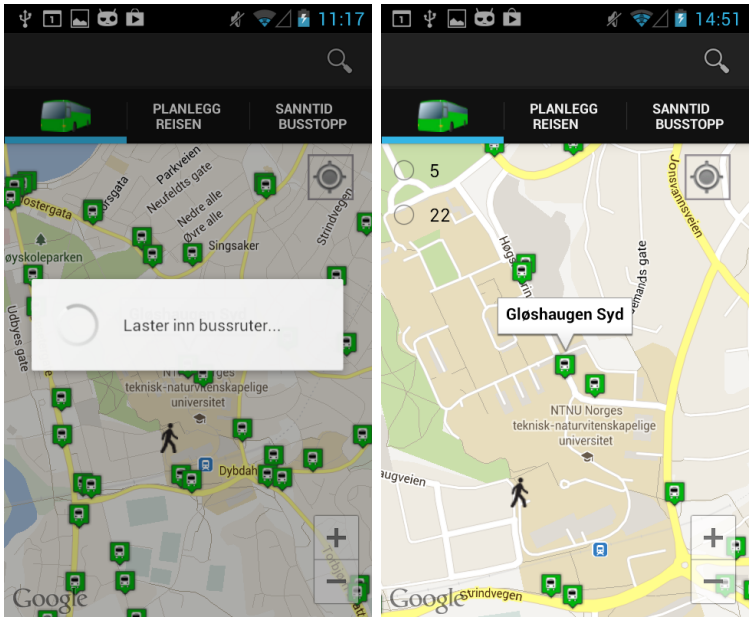

Figure **7.5.a** Figure **7.5.b**

Figure 7.5: View displaying the progress dialogue while conducting a departure search by selected bus stop (a) and the resulting view when the search is finished (b)

After the departing bus routes search, results are listed as radio buttons on the upper left side of the map view, as seen in figure 7.5.b. Users may choose one of these at a time, to visualize the route path for the chosen bus route. The direction path is visualized from the selected bus stop and to the end destination of the bus route. When a route is selected, a progress dialogue appears to inform users that the application is searching for the related trip path. The progress dialogue includes information about the selected route and it's direction, as seen in figure 7.6.a. The resulting view from the trip search is shown in figure 7.6.b.

The selected bus route is visualized with a blue dot in the radio button for continuity of the layout design. Users may at any time select a new bus stop and conduct the same process of displaying either departures or routes paths again. A new search will remove the old direction path and the departing bus icon from the previous search.

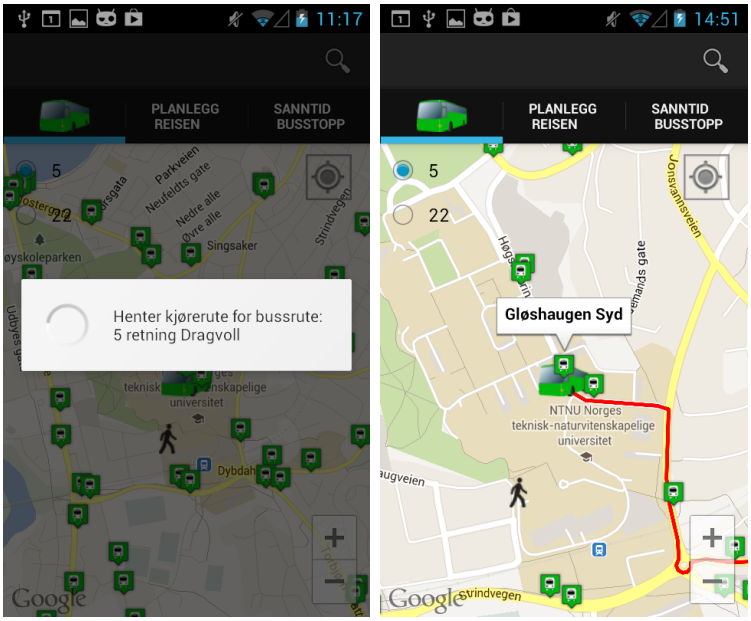

Figure **7.6.a** Figure **7.6.b**

Figure 7.6: View displaying the progress dialogue when conducting a trip search for a selected bus route (a) and the resulting view when the search is finished (b)

7.4 Journey Planner

The second tab is the journey planner tab. This tab allows users to plan their journey ahead to find the best possible route to get to a desired destination from a given departure location. Users are given the option of either typing in an address, place, and use their current location or a combination of these. The initial view when the tab is selected is shown in figure 7.7.a.

For the visualization of the functionality, a journey plan was conducted with the current location of the author at the time and the address "Moholt Alle 1" as destination location. The view with test data input is shown in figure 7.7.b. When departure and destination locations are selected, the user may navigate further with two buttons, providing journey travel information on a textual form or by a map view.

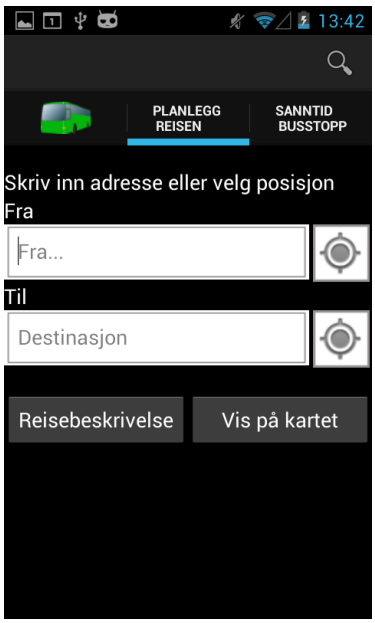

Figure **7.7.a**

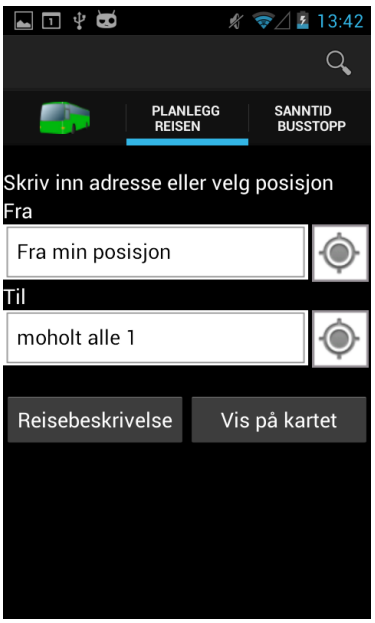

Figure **7.7.b**

Figure 7.7: The initial view of the journey planner tab (a) and the same view with departure and destination locations (b)

The feature providing the journey planning result on a textual form has not been implemented as outlined in chapter 5. The rest of the walkthrough for the journey planner tab will hence focus on the resulting map view representation of the best journey route.

After the map view button is pressed, users are informed that the application is loading the map. The progress dialogue displayed when the map is loading in the background is shown in figure 7.8.a. After the map view has been loaded, the dialogue changes to inform users that the application is computing the best journey route. The progress dialogue while computations are being conducted is shown in figure 7.8.b. We can see from the background of the screen behind the progress dialogue that the view has changed as the map is now loaded.

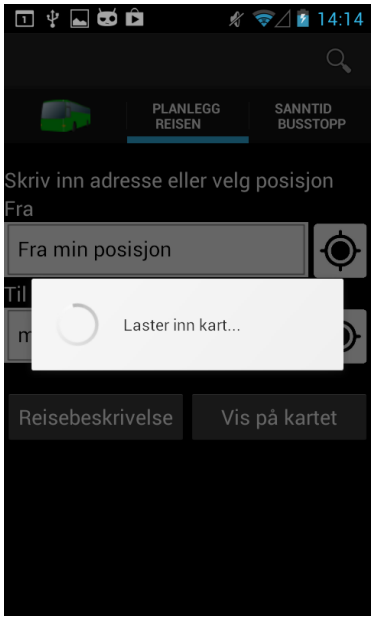

Figure **7.8.a**

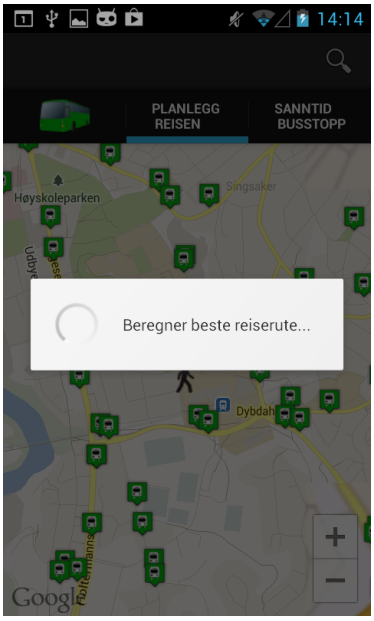

Figure **7.8.b**

Figure 7.8: Informative progress dialogues after selecting journey planner map view in respective order of (a) then (b)

For the map representation, users are given an overview of which bus stop they need to go to, which bus route to travel with from the bus stop, where the given

bus route's travel path is ahead, which bus stop to get off on, and finally where to walk to, to get to their final destination. Paths for walking and bus riding are visualized with lines along the road. The paths for walking are visualized with a blue line and bus travel drive paths are visualized with a red, slightly thicker, line.

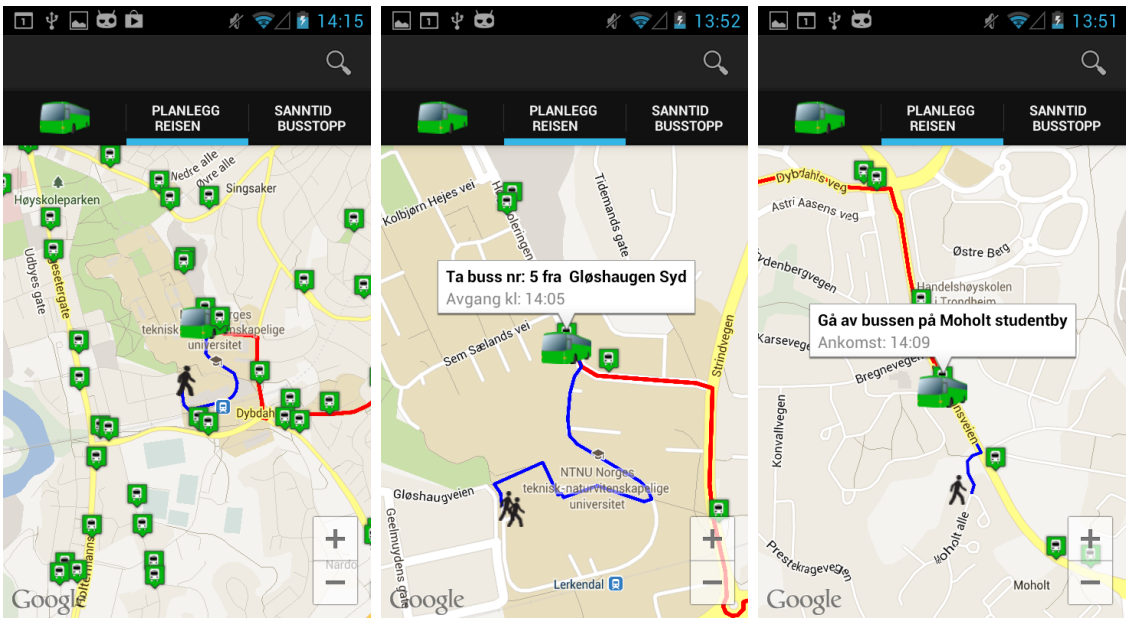

Figure **7.9.a** Figure **7.9.b** Figure **7.9.c**

Figure 7.9: Interaction for the bus route path tab: initial view (a), touch gesture on the departure bus stop icon (b) and with touch gesture on the destination bus stop icon (c)

7.4.1 Validation of the Journey Planner

To validate the use of the journey planner, the functionality was tested thoroughly and examined. The examination process is based on debugging the process to locate the distance to the nearby bus stops, and checking the real-time system separately, to cross-check the next departures. The manual examination is then compared with the journey proposed by the system. An example of this testing is presented as a scenario below.

Scenario

User is travelling from current position to *Moholt Alle 1*, as visualized above. Current time when the test was conducted was 13:49, and the day of the week was a Sunday.

The nearby bus stops of the departure and destination location is listed in table 7.1 and 7.2. Some bus stops are listed twice because the search has found the similar bus stop in both directions. This is validated as the algorithm as presented earlier does not know which direction the destination is, and the departures from the bus stops are checked if they contain one of the bus stops close to the destination location.

Table 7.1: Nearby bus stops of the departing location

Name of bus stop	**Distance to bus stop**
Gløshaugen Syd	0.6 km
Gløshaugen Syd	0.6 km
Lerkendal Stadion	0.6 km
Lerkendal Stadion	0.6 km
Lercehndal Gård	0.4 km
Lercehndal Gård	0.4 km
Prestegårdsjordet	0.4 km

Table 7.2: Nearby bus stops of the destination location

Name of bus stop	**Distance to bus stop**
Brøsetvegen	0.4 km
Moholt	0.1 km
Moholt	0.3 km
Moholt Storsenter	0.5 km
Moholt Studentby	0.4 km
Moholt Studentby	0.2 km

After examination of the possible routes, the actual possible bus stops close to the departure and destination location is listed in table 7.3 and 7.4.

Table 7.3: Nearby bus stops of the departing location in the right direction

Name of bus stop	Distance to bus stop
Gløshaugen Syd	0.6 km
Lerkendal Stadion	0.6 km
Lercehndal Gård	0.4 km

Table 7.4: Nearby bus stops of the destination in the right direction

Name of bus stop	Distance to bus stop
Moholt	0.3 km
Moholt Studentby	0.2 km

The selected journey by the application is to take bus number 5, departing at 14:13 from bus stop: *Gløshaugen Syd* to the bus stop *Moholt Studentby*. Examination of the possible routes, shows that the user could take a bus from the bus stop *Gløshaugen Syd*, *Lerkendal Stadion* or *Lercehndal Gård*. Bus routes departing from *Lerkendal Stadion* departs also from *Lercehndal Gård* just a short while later. The next three bus routes heading in the direction of the destination are bus route 5 departing at 14:13 from *Gløshaugen Syd*, and bus route 66 departing at 13:53 and 14:53 from *Lercehndal Gård*.

The time spent walking to the bus stop *Gløshaugen Syd*, is calculated to be 7.2 minutes from the departing location, compared to 4.8 minutes to the bus stop *Lercehndal Gård*. The first departure is 4 minutes after the current time of the test and the time spent walking to the departing bus stop is set to be 4.8 minutes. The user is thus not able to make it to the bus stop in time according to the algorithm, and the route is not defined as a possible route. The second two departures are feasible for the user. The time measures of the two possible routes are presented in table 7.5.

Table 7.5: Journey data for the two best possible routes

Route	Walking	Waiting	Bus	Walking	Total Time
1	7.2	16.8	4	2.4	30.4
2	4.8	64	3	2.4	74.2

We can see from the two possible routes examined from table 7.5, that the first route presented is clearly the best choice. The first route has a total travel time of 30.4 minutes compared to the second one with a total time of 74.2 minutes.

Examination of the selected bus stops validates that the closest bus stop possible has been chosen by the application. For the second possible route, the user could have walked to two bus stops, which would have resulted in the same bus route for travelling, and both bus stops was feasible departing locations, in terms of walking to the bus stops in time before the bus would depart. The bus stop chosen for the possible route was *Lerchendal Gård*, which we can see from table 7.3, is the best choice as it is closer to the location of the user than *Lerkendal Stadion*.

As the two possible routes presented are so different from each other in terms of total time measure, the weighted overlay algorithm has not been needed to select between the best. The selection has thus been conducted with the basis of shortest path only. The validation of the weighted overlay has been presented in chapter 6, and as the functionality has been validated for the foundation algorithm, the weighted overlay is thus also validated for use. This make up the total validation of the journey planning functionality.

7.5 Real-time by Bus Stop

The last tab is the real-time tab. This tab presents the users with three optional ways to gain real-time information for a selected bus stop, these are related to search for bus stop by name, choice among nearby bus stops and choice of bus stop from a map view.

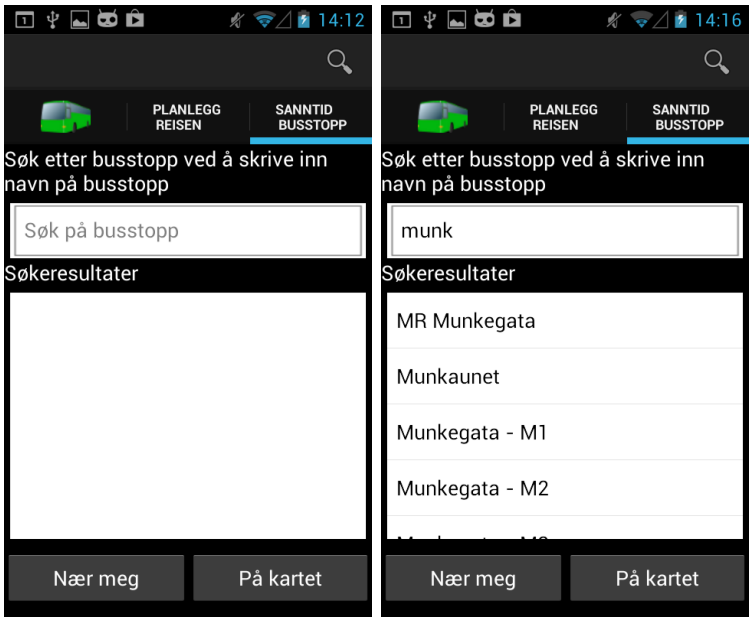

Figure **7.10.a** Figure **7.10.b**

Figure 7.10: Initial view for the real-time tab (a), and the view after typing in a few letters (b)

The initial view when the tab is selected is a view displaying a search field and the related resulting list view. Users are encouraged to type in the name of a bus stop in the search view. Simultaneously as the user is typing in letters, a search for similar bus stop names are conducted through the values stored in the bus stop database table in the database. Below the result list view from the bus stop search, there are two navigational buttons. The first button represents a search for nearby bus stops based on the location of the user. The second button represents

a map view with bus stops for selection. The initial view of the selected tab is shown in figure A.1, and the functionality of the search field is shown in figure 7.10.b.

The resulting view when selecting the *nearby bus stops* button is shown in figure 7.11. The first image, figure 7.11.a, displays the progress dialogue which appears when the button is selected and while the prototype calculates the distance to the nearby bus stops. The progress dialogue informs users that the prototype is loading nearby bus stops. The resulting view when the calculations are finished is presented in figure 7.11.b.

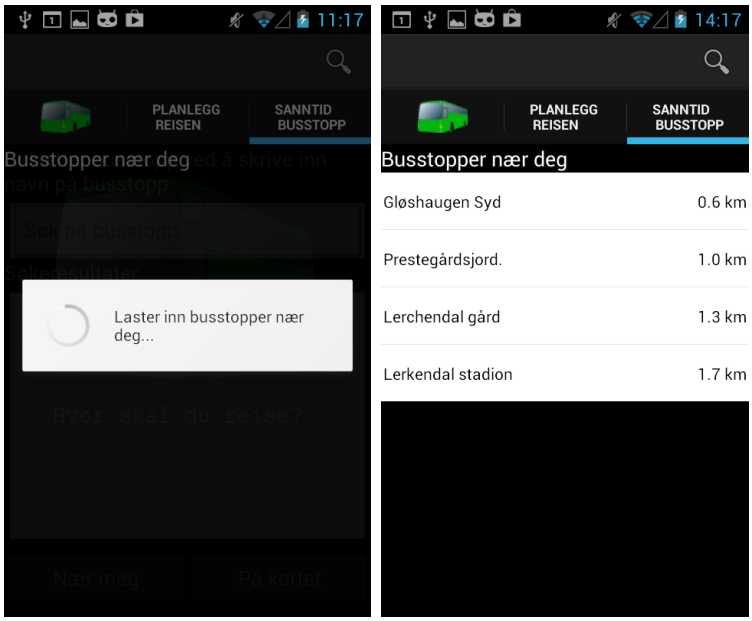

Figure **7.11.a** Figure **7.11.b**

Figure 7.11: View of the progress dialogue when pressing the nearby bus stops button (a) and the resulting list view displaying nearby bus stops (b)

The resulting view when the map view button is selected is shown in figure 7.12. The first image displays the progress dialogue presented when the button is selected. The progress dialogue informs users that the application is loading the map as seen in figure 7.12.a. The resulting view when the map is loaded is presented

in figure 7.12.b.

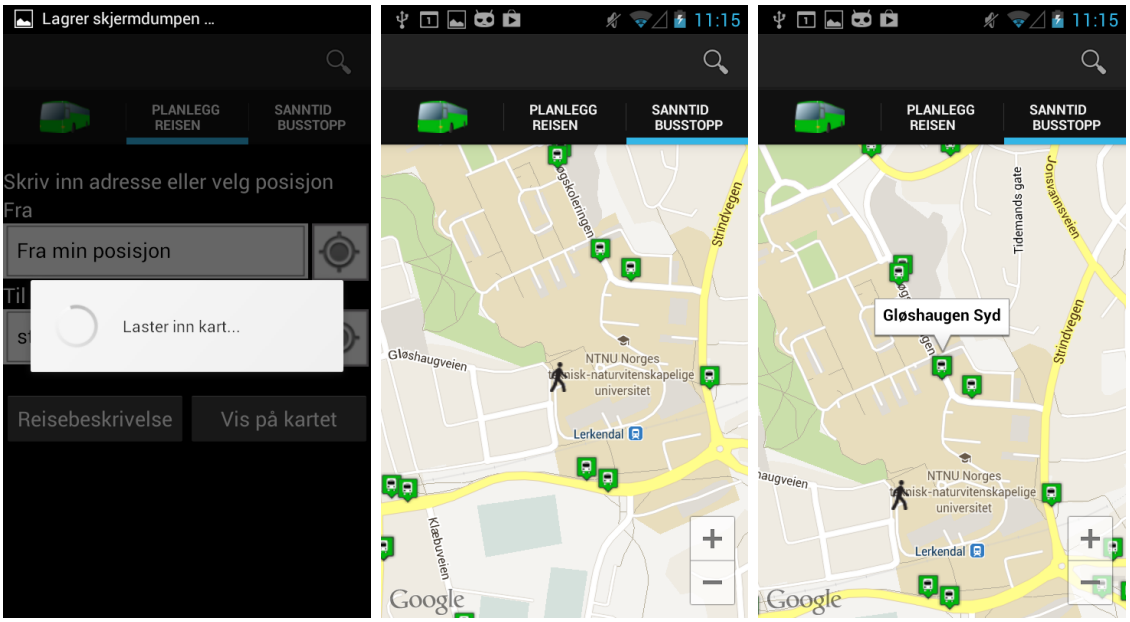

Figure **7.12.a** Figure **7.12.b** Figure **7.12.c**

Figure 7.12: The progress dialogue when selecting the map view button for bus stops (a), the resulting view (b) and the view after one touch gesture on a bus stop has been made (b)

The map view for the bus stops is identical with the map view presented in the route path tab. The only difference is the interaction between user and the bus stop icons. Touch gesture on the bus stop icons pop-up window navigates the user to the real-time fragment displaying real-time information for the selected bus stop.

Common for all three real-time options is that they all finally navigate to the same view providing the user with real-time information for a selected bus stop. This view is presented by a list where each row consists of the columns: route number, route destination, and a time column for the next departure. The resulting views are presented in figure 7.13. The image in figure 7.13.a displays the progress dialogue while real-time data is retrieved from the real-time API.

If the routes presented have real-time data available the departure time is written on the formate "hour:minute" as e.g. 14:27. If a route does not have real-time data, the scheduled departure data is listed on the same formate with the annotation "ca:" before the departure time. This is the same procedure as conducted by *AtB* to separate real-time from schedule data and the annotation is thus kept for easy interpretation for the users of the prototype. Lastly, if the real-time departure time is later than the routes scheduled departure time the real-time data is written, followed by the scheduled data written inside brackets; e.g. 14:30 (14:27).

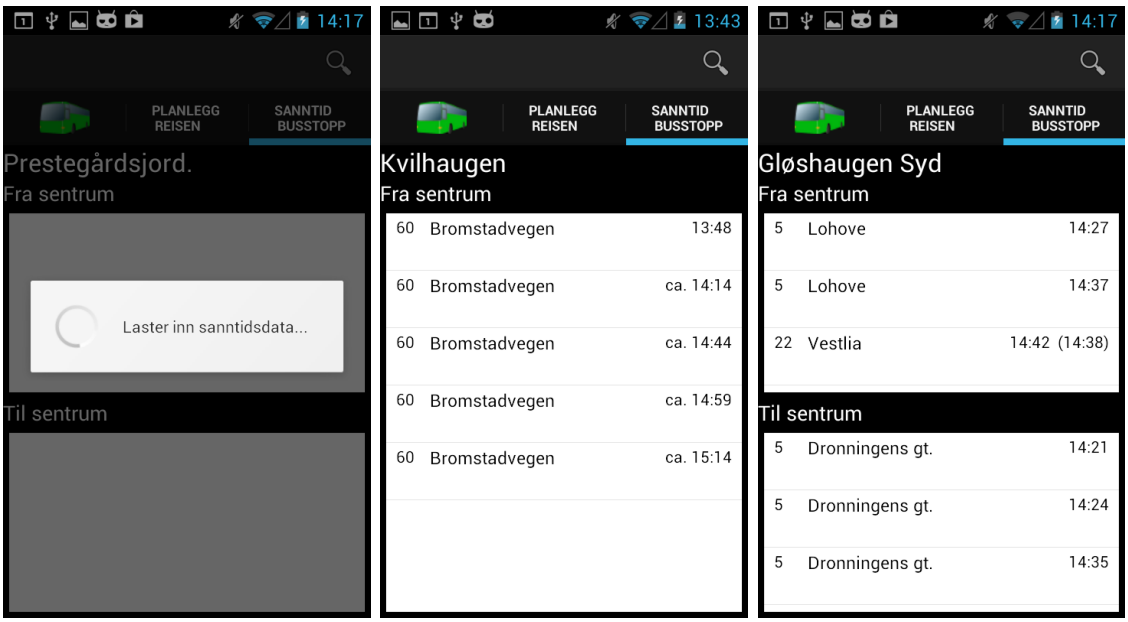

Figure **7.13.a** Figure **7.13.b** Figure **7.13.c**

Figure 7.13: The progress dialogue while retrieving real-time data (a), and the resulting real-time data for a bus stop search with bus stop in only one direction (b), and with stops in both directions (c)

When searching for bus stops either with the search window or searching for close bus stops, only one bus stop name appears in the search results, even though most bus stops have two stops for each bus stop name, one in direction city center and one in the opposite direction. If the chosen bus stop has stops in both directions the result view is divided in two parts, one with real-time data for routes going

towards the city center and one for routes going away from the city centre. This can be seen in figure 7.13.c. If the chosen bus stop consists of one stop only, the result view is a single list view and the to and from city center text is changed according to which direction the routes are going towards, as seen in figure 7.13.b.

Chapter 8

Examination of the Web-based System Prototype

The purpose of this chapter is to introduce the idea regarding the implementation of a map view on the information screens located on public transport modes, focusing on the bus system in Trondheim. This is done with visualizations of a possible system, examining the possibilities.

The walkthrough of the visualizations made with the prototype will examine how the prototype fulfills the requirements outlined in chapter 6. Additionally the general design of the prototype will be examined in terms of the concepts considerations outlined in chapter 4.

As the web-based prototype is not a system to be used in the sane matter as the mobile application prototype, the walkthrough can only be focused on the visualization part regarding the legibility of the prototype.

8.1 Test Scenario

To examine the prototype a test scenario was used for visualization. The scenario was based on the bus route number 5 with direction "Dragvoll". The time set for the scenario is a randomly selected and has no relevance to the testing scenario.

The scenario environment was set to the area surrounding "Moholt". The bus has recently departed a bus stop, and is heading for the next. The next bus stop in the chosen environment is named "Moholt Studentby".

8.2 Visualization

The initial view of the prototype is a dual view, one with route data and one containing a map view, as seen in figure 8.1.a. The route data view is presented as the top most horizontal view, presenting users with route number, end destination and the current time. The map view presents the location of the bus in real-time with a bus icon. Bus stops are selected as POIs and drawn on the map with bus stop icons. The map view is also displaying the name of the next bus stop.

The next bus stop is visualized with a custom pop-up window. The pop-up window holds the name for the next bus stop, and appears just below its related bus stop icon on the map view. The view appears with a semi-transparent background. This is chosen to not fully cover possible POIs in the background of the pop-up window. The font size for the bus stop name is similar to the route destination, and should be fairly legible for all public transport riders on the mode implementing the prototype. Font sizes are based on the font size present in the system used today.

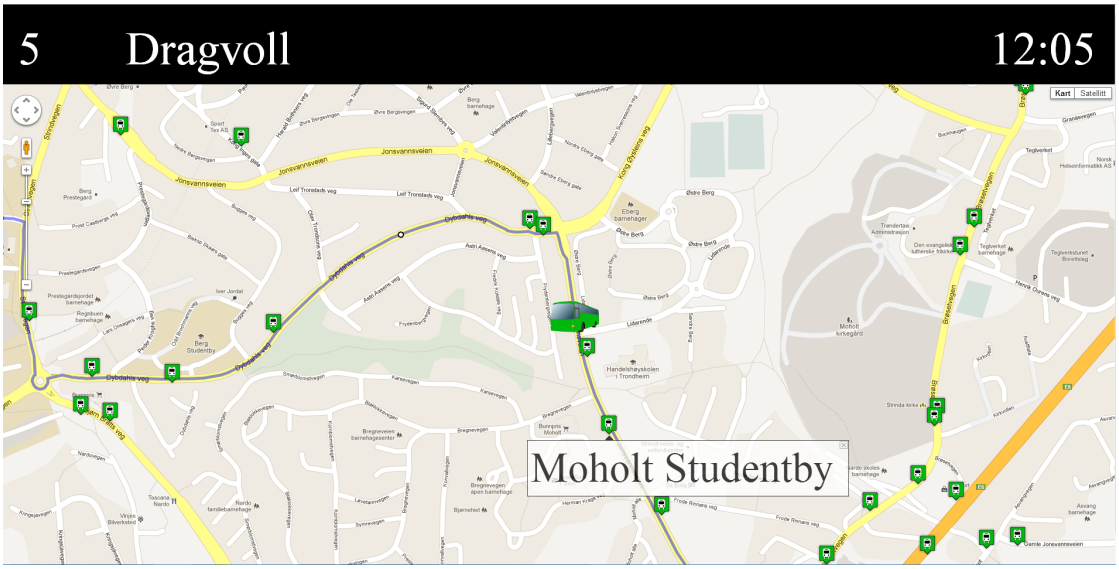

Figure **8.1.a**

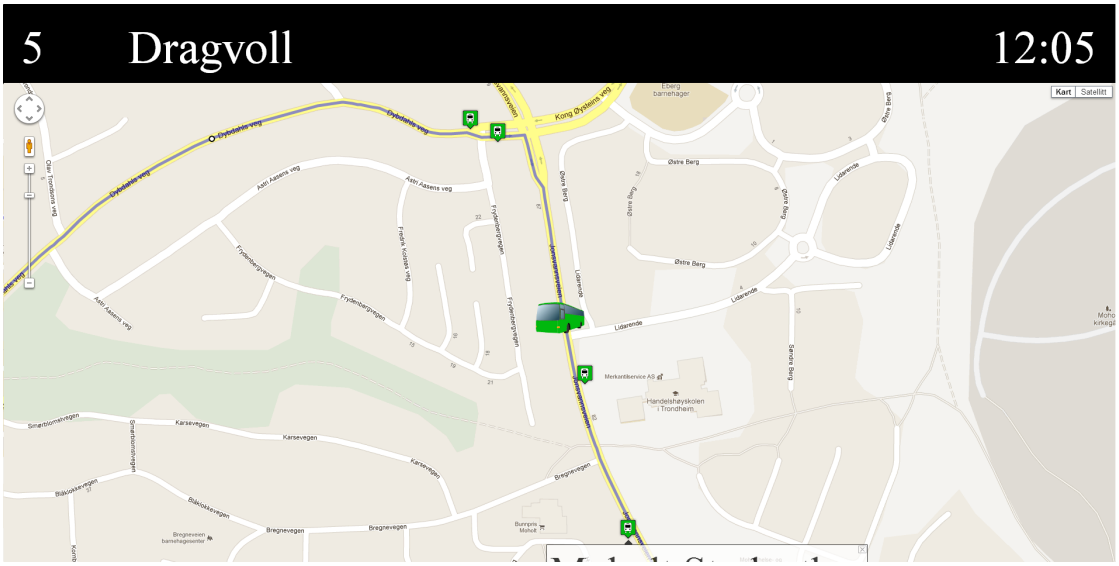

Figure **8.1.b**

Figure 8.1: Proposed zoom level for the web system prototype (a) and the view with a higher zoom level (b)

As the pop-up window displaying the next bus stop name is implemented in the prototype. it appears just below the bus stop icon. This could be improved with some additional queries. One may for each next bus stop check if the bus is north

or south of the bus stop. If the bus is located north of the bus stop, the pop-up window should appear below the bus stop icon, as visualized in figure 8.1 and 8.2. If the bus is located south of the bus stop icon the pop-up information window should appear above the bus stop icon. If this is not handled, the information window will interfere with the real-time view of the travelling bus. The same idea could be implemented to animate the bus icon according to the direction it is driving in.

The direction path of the bus is implemented similarly as for the route path visualization in the mobile application prototype. The first bus stop of the route is subjected to a web-service call to the real-time system API to obtain the trip id of the route. The trip id is then used in a HTTP call to the real-time system to obtain the trip details for the given route. A complete direction route is then drawn on the map consisting of the directions between all bus stops along the route.

The visualization of the prototype was closely related to the zoom level of the map view. The optimal zoom level proposed in this thesis is shown in figure 8.1.a. The view with this zoom level allows public transport riders to get a fair overview of the surroundings as well as some level of detail close to the bus and the nearby bus stops. The view is also believed not be to disturbed by the dynamic view triggered by the change of location of the bus. The bus icon, based on the spatial location of the bus, is set to always be the center of the map view.

In figure 8.1.b the prototype is shown with one higher zoom level than used in figure 8.1.a. This view is too close and the dynamic changing view as the bus is driving along the route path will be disturbing for public transport riders. The next bus stop in the test scenario is not visible due to the zoom level and location of the bus and the bus stop.

Figure 8.2 shows two other views of the prototype which is valued as visualizations with not high enough zoom level. The surrounding view in these figures is believed

to be too general, and thus removing the navigational possibilities of the map view. In figure 8.2.b we can also see that the map is getting close to being overpopulated in terms of the POI distribution on the map, which is distributing for the overall map view.

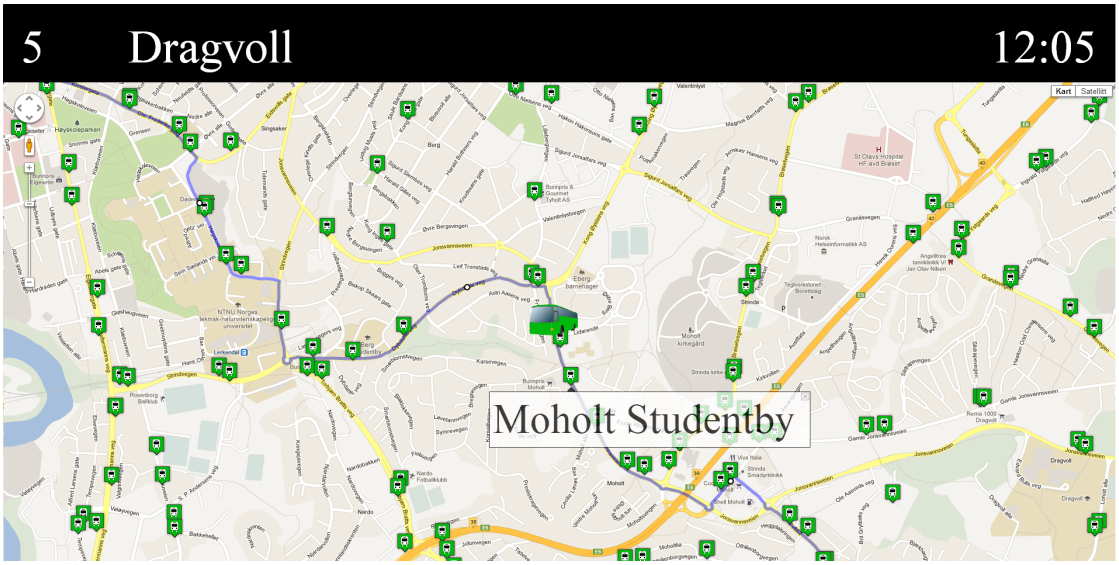

Figure **8.2.a**

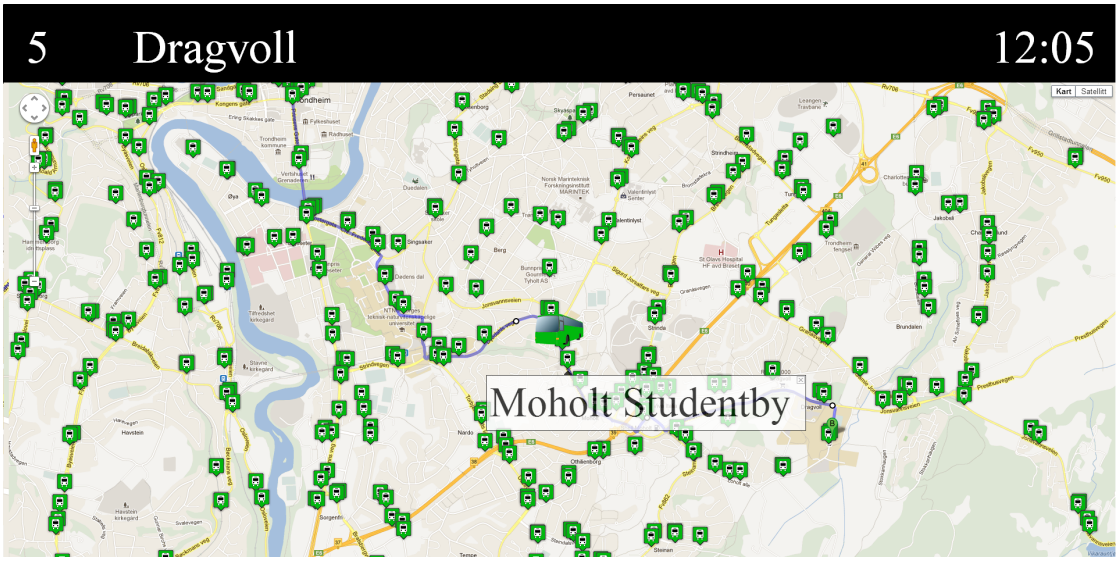

Figure **8.2.b**

Figure 8.2: Visualization of the prototype with higher zoom levels than the proposed best suited zoom level

Chapter 9

Related Problems

There are several limitations which must be addressed during the planning process for developing the prototypes. These limitations and concerns are presented earlier in chapter 5. Later in the development and testing phase some additional problems occurred that is worth elaborating. These will be presented in this chapter.

The functionality presenting most issues is the visualization of travel paths for bus routes. There are also some issues related to obtaining distance values between two locations, and the information distribution and selection handling of departing bus routes from bus stops. These topics will be examined further and discussed in this chapter.

9.1 Google Direction Services API

The Google Direction Services API was frequently used in the mobile application prototype, for distance and direction routing purposes. Both of these functionalities presented some issues for the usage and completeness of the prototype.

9.1.1 Distance Values

For the nearby distance functions, the Google Directions Services API was used, and the distance value between two locations was parsed and kept in the prototype for further use. The issue found related to this service is that some distances were not returned with a value. The parser was set to handle such distances by annotating them as "too far".

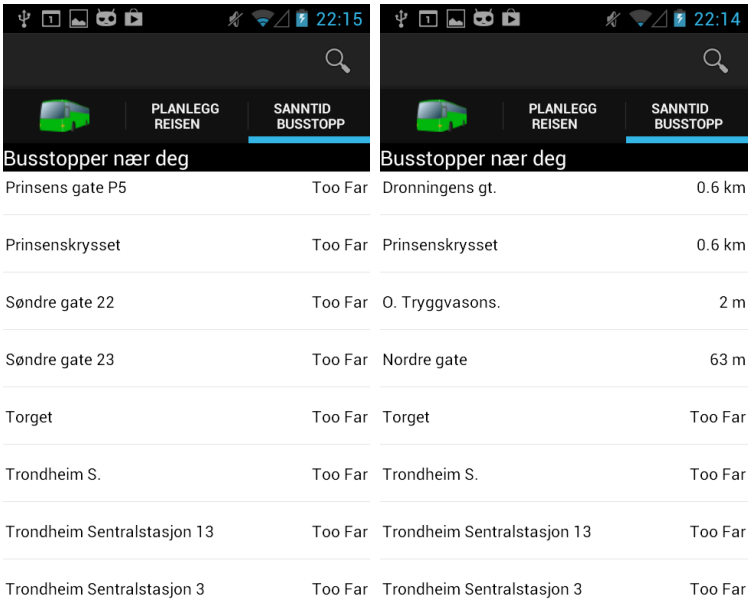

Figure **9.1.a** Figure **9.1.b**

Figure 9.1: Displaying resulting nearby bus stops (a), and the same search conducted a minute later (a) for the same location

An example of the nearby bus stop problem is shown in figure 9.1. The resulting search for nearby bus stops in figure 9.1.a and figure 9.1.b are conducted for the same location with only a minute time difference between the searches. In figure 9.1.a, we can see that the bus stop *Prinsenkrysset* is not given a distance value and is hence parsed the value "too far". In the same search conducted a minute later, the same bus stop is given the distance value 0.6 km. This indicates that there is an issue in retrieving the distance values, and not the actual size of the values.

Other test searches conducted on the nearby bus stops functionality shows that some search queries results in only a few distance measures, while some provides distance values for nearly all bus stops in question. No pattern was found for this issue during testing of the functionality.

9.1.2 Routing

Most problems with the mobile application prototype are related to the routing of bus route paths. During the testing of the application there were two main issues discovered. The first issue is that some route paths are not completely drawn upon the map view. Examination of the system code and its dynamic output when running the prototype shows that the system are receiving values for the polylines to be drawn, but all of these do not always appear. This problem is also evident even though the known problems of lengthy routes, as explained in chapter 5, has been addressed by dividing the complete route into bracket routes between the bus stops in the route. Compared to the route directions in the web-based prototype, this is not presented as an issue. The issue is thus believed to be related to a limitation within the mobile environment, and is subjected to further investigation.

The other issue related to routing of bus paths, is based on the possible driving paths for cars compared to public transportation modes. In the city of Trondheim which the prototype is developed for, there are public transport lanes, and some roads are originally one way driven, with the exception of public transport vehicles. This is not accounted for in Google Maps, which hence presents problems when visualizing route paths.

The city center presents issues for this functionality, as presented in figure 9.3. The first figure 9.3.a shows the original Google Maps view for the city center of Trondheim and the road network. The next image, in figure 9.3.b, is an illustration presenting the actual route travelled by bus route 5 with destination *Buenget*, as

it arrives in the city center and stops at the city center bus stop. The last image, in figure 9.3.c, displays the route visualization for the same route as presented in figure 9.3.b, but as realized in the prototype. We can see clearly that the map view in the prototype do not allow for driving directions where supposed in this area, and the alternative is a discontinuous route path.

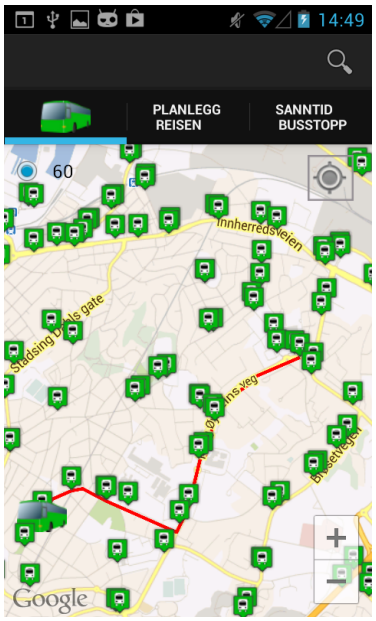

Figure 9.2: View showing the sudden end of a route path

This issue is evident for nearly all bus routes as they all pass through the city center along a route which is not marked for driving by Google Maps. Another good example of this issue can be seen for bus routes travelling from the city center of Trondheim and towards *Ila*. These routes are travelling along a road which is one way driven in the opposite direction. Public transportation is however the exception of this one way issue, and may drive along the road in both directions. This issue is shown in figure 9.4. The first image, figure 9.4.a, represents another part of the direction path of bus route 5 which was presented above in figure 9.3.c. We can see from the image that the route path is drawn to a certain point, where it turns, and continues in the opposite direction. This is believed to be due to the one way street which the bus is not allowed to travel according to Google Maps.

The route is hence drawn for another road until a turn is possible, to then finish the route on the road it was supposed to follow. The issue being it is now in the wrong direction. Finally, the route is cancelled although there is still a major part left of the bus routes travel path.

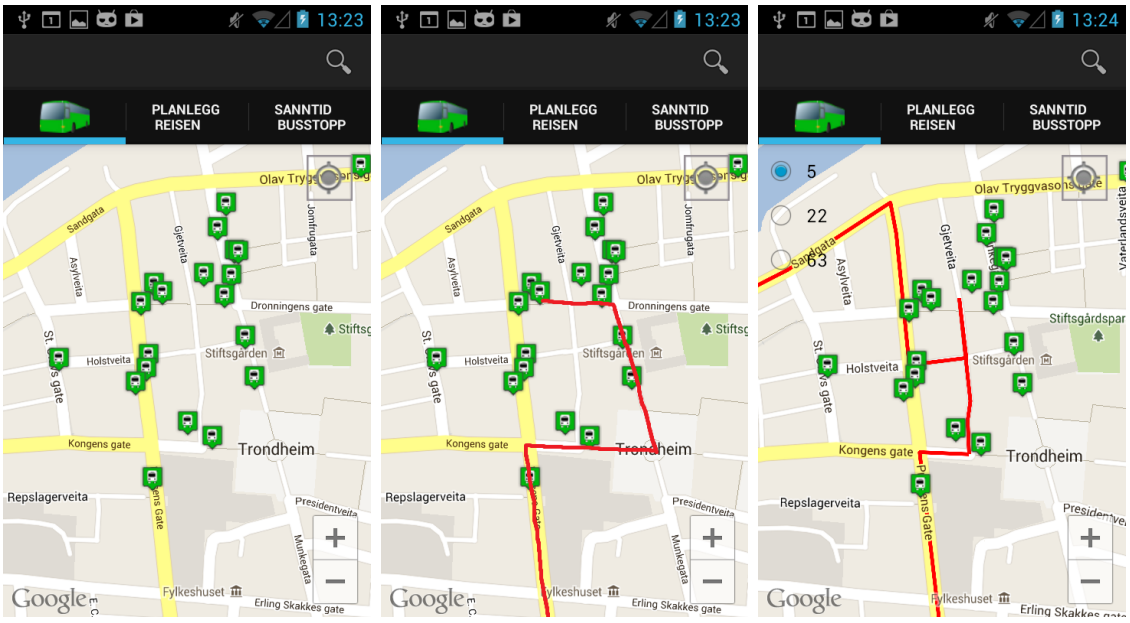

Figure **9.3.a** Figure **9.3.b** Figure **9.3.c**

Figure 9.3: Initial view of the city center of Trondheim (a), illustration of the normal route for bus route 5 in direction *Buenget* arriving in the city center (b), and the actual drawn route for the same bus route in the mobile application prototype.

The same problem is evident for bus route 63, shown in figure 9.4.a. For this route we can see the same turn action as for bus route 5. Opposed to the earlier example, this route does continue on towards its destination after the one way driven road issue. On the continued travel path the issue of non-drivable road is again encountered, resulting in a dead-end line to bus stops covered by the bus route, where the bus evidently turns and drives back, before finding a road to continue to the next bus stop along the route path.

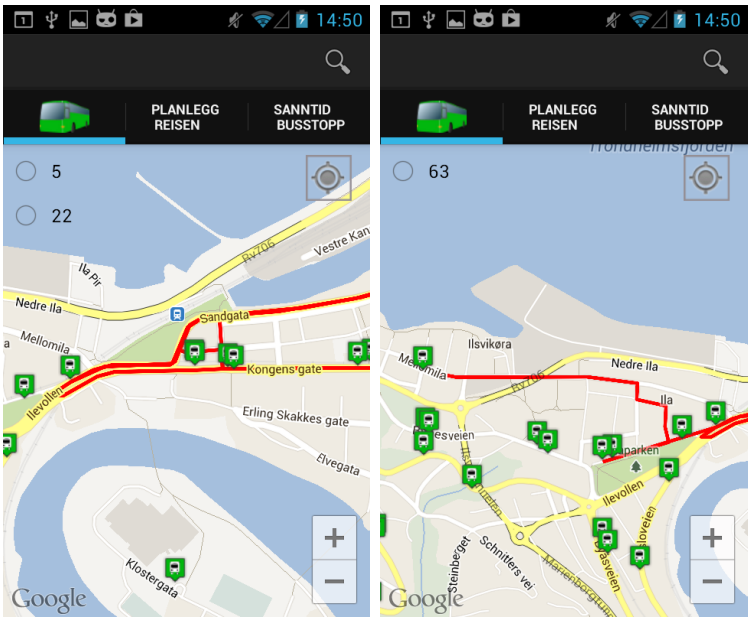

Figure **9.4.a** Figure **9.4.b**

Figure 9.4: Route direction problems the bus route 5 (a) and bus route 63 (b)

9.2 Selection of Bus Route

Another issue which can be seen in the illustrations presented in the previous section is the selection issue when selecting a bus route to visualize its direction path. The routes are displayed as radio buttons in the upper left corner of the map view. The radio button activity is based on a single selection. This means that only one radio button can be selected at once. On initialization, no button is selected. For some cases the button never receives its selected animation, which is a blue dot inside the button. Examples of this can be seen in e.g. figure 9.4.a and 9.4.b. The actual visualization when a button is selected is shown in figure 9.3.c. During the testing phase of the mobile application prototype, no pattern was found to explain this issue, and the issue seems to appear randomly.

9.3 Presentation of Bus Routes

The issue related to selecting bus routes for route path visualization is explained in chapter 7, with the walkthrough of the mobile application prototype and its functionality. The idea is not to repeat the problem, but visualize it as it is presented in the prototype.

As seen in figure 9.5 the visualization for bus routes departing from bus stop is more legible in figure 9.5.a than in figure 9.5.b. The problem presented for route selection in figure 9.5.b is that the bus stop is one of the city central bus stops which is passed by several bus routes. The list presented in the map view is not the whole list of departing bus stops, but only the ones that fit the screen size. The issue of distributing this selection listing to the users is a task for future work.

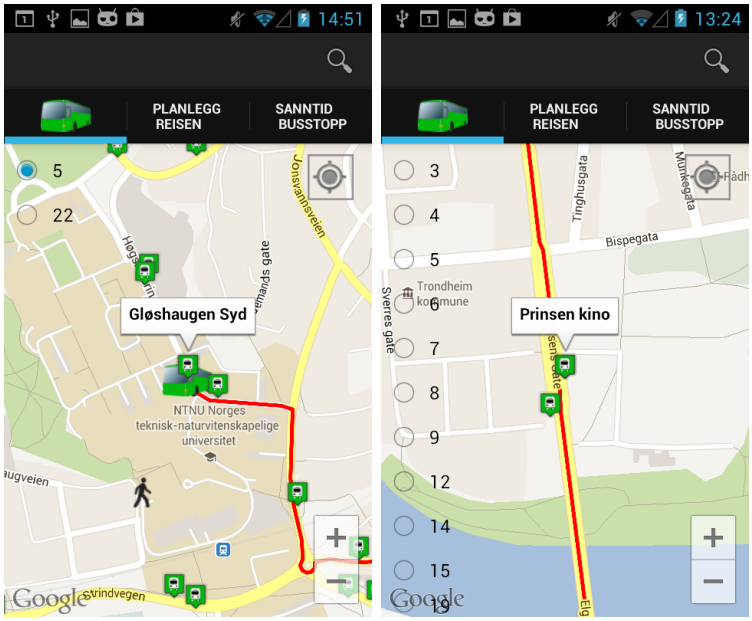

Figure **9.5.a** Figure **9.5.b**

Figure 9.5: Displaying route selection from bus stop *Gløshaugen Syd* (a) and from *Prinsen Kino* (b)

Part IV

Discussion and Further Work

Chapter 10

Discussion

The main goal of this thesis was to investigate the possibilities of higher map usage within public transport systems. The scope was limited to the development of a mobile application prototype distributing public transport information using maps, and a examination of the possibilities of map usage in systems aboard public transport modes. This chapter will examine if the prototypes developed fulfill their requirements and how they work.

The decision to use Google Maps in both prototypes proved to be a good implementation. The use of the Google Maps API presented a well-documented API which made the use of a map-client easy. The map rendering process is fast for both prototypes, and responses to touch gestures on the map are easily handled.

In general, the Google API was a good solution for the mobile application prototype. The first reason, as mentioned, being the use of Google Maps for map view. The other reason being the possibilities related with the API, such as Google Direction Services, which offers a range of possibilities that are easily implemented in relation to the map object.

The prototype in focus during the work of this thesis was the mobile application

prototype. The functionality of the prototype can be divided into the three tabs, the *route path visualization*, the *journey planner*, and *real-time by bus stop*.

The distribution of real-time information and journey planning functionality is not revolutionary in any way, and these functionalities can be found in existing systems today. The investigation in chapter 3 did however reveal some limitations and how capabilities within the existing services that may be improved. One of the aspects that could be improved in all of the systems examined is the map usage. These investigations are the foundation of the guidelines this thesis is based on. The prototypes developed are hence believed to correct the limitations and error made by other systems. This does not mean that the solutions provided in this thesis are optimal. There are still issues and concepts that need further investigating. These are described in the next chapter, which is further work.

The following sections will be based on the three main functionalities presented in the mobile application prototype, before the prototype will be assessed in relation to the given requirements presented in chapter 6. Finally, the web-based prototype will be discussed and assessed in relation to its requirements, also presented in chapter 6.

10.1 The Mobile Application Prototype

The purpose of the mobile application prototype was to create a system with high map integration for information distribution and to investigate the possibilities of a multifunctional system implementing the believed, most desired features by public transport users.

10.1.1 Route Path Visualization

The way of presenting route directions is not commonly used by public transport systems, but the prototype proves that this could be a feasible solution to distribute route path information, with some adjustment to the related issues with the drawing of the direction routes.

All of the features in the functionality are working, but there is still work to be done. The functionality is subjected to usability issues in terms of intuitive use. The idea and implementation of the functionality is fairly new, and is hence a lot less intuitive to users than functionality such as real-time by bus stop, which one may find in several other applications. As it is implemented in the prototype, users have to be aware of the touch gestures possible on bus stop icons and their pop-up windows to retrieve departing routes and to further use the radio buttons to obtain route path directions from the selected bus stop.

As the routes which result from the departure bus stop search are presented as radio buttons, they are assumed to be intuitive in terms of selection. Radio buttons are a common concept, which are associated with the selection of one at a time. There is still one issue with the route presentation. In the prototype they were enabled on the map view in black writing. The radio buttons are legible when the higher zoom levels are used, and the legibility drops as the zoom level is reduced. This is mainly due to the interrupting view occurring when the lower zoom levels are used, because the size of the POI icons drawn on the map is not related to the zoom levels, and the map will be subjected to overplotting of these. The static size of the overlay icons on the map view is another aspect which should be handled.

The route path radio buttons were subjected to be featured in a separate pop-up window, which would make them more legible. This solution however, limits the built-in functionality with the radio buttons, which is that one may navigate between them and select one and another on turn. If a pop-up window were to be used, users would have to select the bus stop again to review the route list and

reselect a route for path visualization. The routes could be changed into a regular list instead of radio buttons to solve this issue, but the reselection issue would still be present. When reselecting a bus stop, a new HTTP call must be made to a real-time API, which again is adversely as these should be minimized to the fullest.

As mentioned earlier on in this thesis, the route path visualization should ideally also visualize other routes in the overall bus network. This would support transfer options and inform users of where these possibilities occurs. If the routes were to be stored in a database this could be feasible, and to good use for users of the application. In this work, as each route is based on a HTTP call for a real-time API, such extensive processing is foreclosed. Based on the related problems with the drawing of polygon route paths this solution would also solve the general problem with route path visualization.

10.1.2 Journey Planning

The journey planner tab was assumed to be very much intuitive. The type-in fields are well described with attributes, and the location icon is the same as used in the first tab. The location icon is also a common icon used for its purpose, making it a well-known symbol for most users. The buttons used in the journey planner are also well described and should be intuitive.

Because of the limited time during this work, the search for addresses has been conducted as simple as possible in terms of development. The only additional modification made apart from general geocoding, was to retrieve 3 address results, and check each if they are in the municipality of Trondheim, and then select the first address. More functionality related to the geocoder should have been implemented, such as distributing the other resulting addressed to the user. The user could then select the correct address in cases where there might be several possible addresses for a address search within the same city. This also introduces

personalization which is one of the social challenges presented within the mobile environment, as well as supporting the accuracy of the prototype.

The functionality presented in the real-time fragment when searching for a bus stop based on bus stop names could also been implemented for the type-in address field, presenting users with address options as they are typing in addresses. This would require a more comprehensive process as the geocode database is a lot bigger than the bus stop database used in the prototype.

The resulting view of the journey planner is well visualized with path lines and descriptions related to the presented icons. As the prototype is developed, the location icon used is the same as the departure and destination location icon. These icons could have been different from the each other, to better indicate departure, destination and current location. As the location icon is used for location only in the other features, it might be confusing when it is used for another representation in the journey planner.

In terms of validation, the functionality is successful. As we can see from the results presented in chapter 7, and the journey data presented in table 7.5. For a more advanced calculation of the best travel route users should also be able to personalize the weighted overlay influence percentages. Some users may prefer to rather walk a bit further to a bus stop, than to walk to a nearby bus stop and wait for a bus, and the maximum walking distance may vary considerably. Some users may also be disabled hence limiting their mobility possibilities. It is proven in several studies that walking speed varies with gender and age. Such personal preferences should be handled by user input and taken into concern. The rationale behind this proposal is the social challenges presented in the literature, concerning the mobile environment. The option of defining personal limitations will also be a major contribution in terms of optimization of the algorithm.

The prototype presents only the best journey route, but in most cases there will be several possible routes, and many that may not differ a lot in desired score. The

use of radio buttons presented in the route path visualization tab could have been implemented to allow users to see the three best possible routes in respectively order. Users could then easily navigate between the journeys and select the best journey route of their choice. As described in the related literature, the presented routes should also preferably have an additional visualization to reflect on how well the constraints are met.

10.1.3 Real-time Information Distribution

The real-time functionality of the prototype is believed to be intuitive, easy to use, and accurate based on the data distributed to the user. This is a good indication that users may be encouraged to use the feature. This is important in means of supporting the overall impression of the prototype.

The only adjustment proposed related to the real-time feature, is the division of direction among the bus stops, when bus stops are selected through either search by name, or search by location. The resulting list view of bus stops should have a second information tag combined with the bus stop name, being their direction. This is a limitation which is explained earlier in chapter 5, along with a proposed solution for both the data collection and the data distribution by the real-time service.

10.1.4 Fulfilment of Requirements

A total of three primary requirements and eight secondary requirements were presented for the prototype in chapter 6. In order to see how the prototype fulfills these, or fails to do so, the requirements are reviewed. The primary requirements are presented as earlier in this thesis. The secondary requirements are more closely described, and only the title of each are reproduced.

P 1: **Enable user to retrieve route information for a given route from a selected bus stop.**
This is handled in the route path visualization tab when selecting a bus stop. A selected route can further be visualized with its direction path.

P 2: **Enable users to make a journey plan.**
This is handled in the journey planner tab. The result in only implemented to be seen in a map view, but the prototype is intended to also hold a feature for turn by turn directions in a textual format.

P 3: **Enable users to retrieve real-time information by bus stop.**
This is handled by the real-time tab. Users may obtain real-time information by bus stop by three methods.

S 1: **Departing routes can be retrieved from the map view.**
Users are enabled departing routes after selecting a bus stop, in form of radio buttons. There are some limitations to the legibility of the routes in terms of the bus stop selected and the zoom level of the map, which should be further improved.

S 2: **Departing routes may be visualized as directions on the map.**
By selecting a route presented as realized through S1, users are presented the direction path of the selected route. The presented route is often not complete or accurate, due to map limitations, as explained in chapter 9. As this is a issue outside the control of the prototype, the requirement is seen as fulfilled. The feature shows the possibilities of this implementation regardless of the limitations.

S 3: **The journey plan may be based on custom departure and destination locations.**
Users may type in a address or select their current location using the location button provided.

S 4: **The journey plan is the most suitable route.**
The examination of the journey planner in chapter 7 presents the successful accuracy of the feature, and this is hence proven to be fulfilled in the prototype, based on the scope of the feature. The handling of transfers is neglected, and must be implemented for the functionality to be fully functional. Some more work should be conducted to optimize the algorithm further.

S 5: **The proposed best route is visualized with a map view.**
The map view presents users with icons for departure and destination locations, current location of the user as well as the departing and destination bus stop. All icons hold related information about the journey. The implementation lacks some of the information that should be implemented in the related icons. This information is stored for the route, and can easily be implemented in the future. The same issues as in S2 apply to this feature also.

S 6: **The best proposed route is explained in details.**
As described in S5, the icons presented on the map view hold journey details, with some limitations. The textual feature of journey planning has not been implemented in the prototype as described in chapter 4.

S 7: **Realtime by bus stop may be chosen by several options.**
This is realized as explained in P3.

S 8: **Users may always locate their current location.**
This functionality has been implemented in the route path tab, showing how it can also be applied to all map views in the prototype. Due to the limitation of the development process, this functionality is only implemented in the map view presented in the route path tab. This implementation shows how the functionality can easily be adopted by the other map view.

10.2 The Web-Based Prototype

There was not a possibility to actual test the prototype on a screen it is developed for, and in a real environment. It is therefore difficult to make any real conclusions in terms of legibility. The POI icons may be a bit small, as well as the drawn route directions. The overall impression is however that the prototype is believed to be fairly legible for bus riders.

The legibility of the map is also an important question. In order to make use of the map, it must be legible for the users. The idea is not to be able to get a complete detailed view of the surroundings, but a fairly general one. Related literature on visualization emphasizes the importance of showing the right level of detail on limited screens.

The main goal with the web-based prototype was to investigate the possibilities for map implementation on the information screen aboard public transport modes, focusing on buses in Trondheim. This is believed to be achieved with the prototype. Further work related to the prototype is needed for optimization and usability testing.

10.2.1 Fulfilment of Requirements

A total of two primary requirements and two secondary requirements were presented for the prototype in chapter 6. In order to see how the prototype fulfills these, or fails to do so, the requirements are reviewed, as conducted for the mobile application prototype.

P 1: **Display route information**
This is taken care of in the top most view in the prototype, displaying route number, direction and additionally the current time.

P 2: **Visualize direction route path on a map**

This is handled by the prototype by drawing a blue line along the direction path for the bus route. There are some limitations as explained in S2 for the mobile application prototype.

S 1: **Route information distribution.**

This is achieved as explained in P1.

S 2: **Map visualization.**

The map is presented with POIs for the bus, bus stops and direction lines as explained in P2.

Chapter 11

Further Work

There are several aspects of the two prototypes developed which can be further examined and forms the basis for further work. A good basis for further work to be done can be found in the limitations of the prototypes described in chapter 5 and the related problems presented in chapter 9. These can be classified as foreseen and unforeseen limitations and issues. Some which needs improvement by the third-party services used in the prototype, and some that can be conducted by a developer. These limitations include topics like data distributed from the web services, the route path visualization process, information distributed on public transport modes, usability concerns and scalability.

These topics will be explained in the next sections. The focus will be on the limitations of the prototypes with related propositions for improvements. The idea is to encourage further work within the area of real-time distribution with the use of GIS in applications and systems.

11.1 Data Distributed from the Web Services

The data distributed from the web service could be improved in order to improve the prototypes. Three web services were used in the development process in this thesis. All of these services presented either limitations or related problems to the prototypes.

The real-time service APIs should as described earlier aggregate some of the resulting values to avoid the frequency of web calls to the service. This is especially essential within the mobile environment, which is more limited to the extensive processes of web calls, than a web-service. The first method call proposed for improvements is the bus stop method. Most applications divide bus stop searches into bus stop names divided in two, annotated respectively with the direction of either *to* or *from* the city center. To gain direction information for bus stops one must either examine all bus stops manually, or make a HTTP call to the real-time API to check the boolean *isGoingTowardsCentrum* to check the direction of the buses passing the bus stop.

As there are 1850 bus stops in the network this direction call was excluded due to the risk of overloading the server system. Usage information on *BusBuddy's* web page (Norangshol, 2011) reveals that the daily server system calls are up to 3000 a day. A solution could have been to conduct the server call for a few bus stops per day over a larger time period, but for this project this was instead limited.

The second call which should be aggregated in terms of data distribution, is the method for real-time data. One does not retrieve trip ids when retrieving real-time data, but this can be obtained when retrieving scheduled data. This trip id should hence be included in the real-time call, making the scheduled data method call excessive. The call to retrieve real-time data can thus provide all the relevant information needed, and thus minimizing the HTTP calls conducted.

BusBuddy distributes only the next five departures, opposed to the *Bartebuss*

which distributes departure data for several hours ahead. A longer resulting JSON object from the web service will naturally be more time consuming to parse and present to the user, than a shorter result. This has implications for the usability of the real-time functionality in the mobile application prototype.

In terms of minimizing the run time of the retrieving and distribution process of real-time data, an option could be to present only the next departures on initial access, and then offer the option of retrieving more departures if asked for. A user interested in departure information ahead is more likely to be suited to wait a few more seconds for the result than a user which is checking to see if he can make it to the bus stop in time for the next departure, in a timely manner.

The other web service used is the Google Direction Services API. This API was not foreseen any limitations for the prototype, but presented issues when examined. The largest issue is the drawing of route path for public transportation. This issue seems to be mostly present when the intended routes were not feasible due to the classification of the roads intended. This issue is present in both prototypes. For the mobile application prototype there were also registered events where the whole route was not drawn, without any clear reason. This should be examined further to reveal if the problem is related to the web service, or the drawing of polylines on the map object.

A solution to the drawing of route direction could be to draw the routes manually, and store the data in the database with related trip information. On further use, the application may then retrieve the data from the database instead of conducting web calls and the parsing process of the resulting data. This solution will be extensive to conduct, but will ease the use of the feature in the future. Manually drawing of polylines will also solve the issues described above.

11.2 Route Path Visualization

One of the functionalities of the prototype is intended be a visual map representation for navigational purposes, when using public transportation. This functionality can be developed further. The idea is that the map view can hold data for all buses in the transport network, and visualize these on a map view, as conducted by *Busskartet* and *Flightradar24*. The map view should hold POIs as presented in the mobile application prototype. These POIs are intended to be bus stops and the route path for all buses in the public transport network.

The idea is that routes should be in transparent shades of grey to give a general overview of route paths, but to not steal much focus from the rest of the map view. On the selection of on bus in the network, the route path of the bus should be highlighted for information distribution of the selected bus. The route path may then be coloured in red, as visualized in the mobile prototype. For interchange options, bus stops should be selectable to obtain route information related to the stop. The selection of routes departing from stops should also include options for displaying route directions for a given bus route for general planning purposes.

The literature reveals that the most important issue for public transport users is the timing of departure and transit, which is the foundation of the idea of the route path visualization. The functionality is believed to help users with this issue, as they can then locate their own location, the location of the bus they are on, and the other buses in the public transport network. Maps covering all interchanges have also been found to be the most desired pre-trip functionality.

A lot of work must be conducted for this idea to be realized, and one needs further data such as GPS positions of all buses or a simulation as conducted by *Buskartet* must be made. The proposal of storing route paths in a database must also be realized for the idea to be feasible, due to the high amount of HTTP web-calls the prototype need to conduct if the routes are not stored. Additionally, the idea must be subjected to usability testing, to check if the information is legible and

intuitive, or if the feature will hold too much information to be usable.

11.3 Journey Planning

The journey planner in the mobile application prototype is limited concerned to completeness. The algorithm created is only searching for routes without transfers. This exception must be added to the functionality to add to the completeness and accuracy of the feature. This expansion of the functionality requires careful implementation. The algorithm is already making a high amount of network calls to web services, and the expansion increases this amount remarkably. The need for optimization of the algorithm increases accordingly. General exception handling is also neglected, which is another subject for further work.

The Weighted Overlay algorithm in the functionality in created for the use of selection among possible journeys which do not differ particularly in total travel time. The algorithm should hence be improved to handle a selection for all possible routes, and implicit select among the least time consuming routes based on the weight influence percentages.

11.4 Visualization on Public Transport Modes

As described in chapter 2, Looije et al. (2007) presents some interesting guidelines to the level of detail for limited screen systems. One of which is; detailed level dependent on location. This could be implemented for the information visualization in the web-based prototype. The detail level should be set to be high in the city center, where a higher level of detail is needed due to the higher amount of information present in the map view, compared to rural areas where the detail level should be lowered.

One may also consider the possibilities of using nested levels of details, to support the transfer possibilities when travelling with a public transport mode. With this idea, users could be informed of transfer options given as a list of departures of other bus routes related to the next bus stop. This information could additionally be given with real-time data for the possibility to plan a possible transfer.

Another idea for supporting transfer planning when travelling on a public transport mode could be the proposed idea for the route path visualization described for the mobile application prototype, described earlier in this chapter. The web-system could then hold additional bus icons to represent other buses in the transport network and their spatial location in real-time. Additionally the routes of the other buses could be drawn on the map view in transparent shades of grey.

All of the ideas presented in this section for the web-based system needs to be thoroughly tested and further examined to see if they are feasible. The ideas are only meant as pointers to a more extensive information distribution with map use and context-awareness, and has not been evaluated in regards to related issues.

11.5 Usability Concerns

Usability is a major field within computer science, and should always be considered when a product or system is intended to be used by the public. The prototypes in this thesis should hence be examined. The scope of this work was to only conduct technical tests on the functionality and testing of the feasibility of the concept.

If the prototypes are to be used as a public application and as a implemented system on public transport modes, they must be thoroughly tested for usability issues. The main topic being effectiveness, efficiency and satisfaction as described in section 2.3. Further, the six point check-list presented by Chittaro (2006) when developing mobile visualization, should be addressed. These points can be used as

pointers for the usability of the prototype and for improvements where the mobile application prototype fails to fulfil the points described.

Another important factor when testing the usability of the prototypes, in the testing environment. It is essential the prototypes are tested in a environments as similar as possible to the environment the prototypes most likely will be used in. This is due to the environmental issues present, as described in the related literature.

Both prototypes are embedded systems which combine map representation within a containing layout. This presents the prototypes with a usability concern for both the cartography of the map presentation as well as the overall usability design. The makes the prototypes subjected to usability examinations for both of these fields as well as the combination of them.

11.6 Scalability

The prototypes developed are based on the bus system in Trondheim. The use could however be expanded to cover other cities and transport modes. The use and implementation of the prototypes will be the same, the only change will be the input data, and possible changes to the handling of this, if the construction of the input data differs from the real-time and location data used in this thesis.

The literature reveals that multimodal travel is of high appreciation, and this should also be covered by the mobile application prototype. For the mobile application this will require some extensive work, as the users must be able to choose transport mode. This also presents more transfer options in the journey planner algorithm, which must be addressed. The expansion of multimodal travel will not have any impact on the web-based prototype as it is developed in this thesis, at it is non-dependent on other modes in the transport network.

Part V

Conclusion

Chapter 12

Concluding Remarks

This chapter provides concluding remarks about the thesis and the work presented. The topic explored in the thesis is the use of maps for information distribution related to public transport and real-time information. The prototypes developed are believed to act as examples on how maps can serve the function of information distribution for public transport data.

The mobile application prototype has the purpose of obtaining all needs that users may behold in terms of public transport information, embedded in one application. The focus has been on the defined concepts presented earlier in this thesis. The prototype should with benefit be seen in combination with the findings and discussion in this thesis in order to get a view of the context and rationale of choices made regarding the development.

The web-based prototype is believed to shed some light on the integration of map use aboard public transport modes for information distribution and context-awareness. As for the mobile application prototype, the web-based prototype should be examined in context of the findings and discussion presented in this thesis.

The use of mobile applications has emerged tremendously in the later years, and we can see a rise in the use of maps for information distribution with the emerge of location-based services. We can also see that there has been a rise in the integration of maps within the public transport sector. There is reason to believe that this integration will only increase, based on the related work. The idea presented in this thesis is also founded on the possibility to see this emerge on systems aboard public transport modes, for public distribution of transport data.

As discussed there are some limitations and problems related to the prototypes developed which must be addressed. These problems are believed to be carried out in detail in the thesis. The work conducted is thus meant to be a good starting point for the integration of map use in public transport systems.

There is still a lot of work that needs to be done. Both related to the development of the prototypes, and the investigation of the topics presented in this thesis. I hope that the work presented may inspire others to further investigate the topics and contribute to the development of the prototypes, or similar systems. This may lead to a step towards a more map oriented view for information distribution of public transport information. In the long run, this may also be a small step towards the promotion of public transport in general, which is known to be a good solution environmentally.

Bibliography

Abran, Alain; Khelifi, Adel; Suryn, Witold, and Seffah, Ahmed. Usability meanings and interpretations in iso standards. *Software Quality Journal*, 11(4):325–338, 2003.

Alamdari, Fariba. Airline in-flight entertainment: the passengers' perspective. *Journal of Air Transport Management*, 5(4):203–209, 1999.

Andersen, Rune M. Om bartebuss. URL `http://bartebuss.no/om`. Accessed: 2013-05-23.

Android Developers, . Location and maps, 2013a. URL `http://developer.android.com/guide/topics/location/index.html`. Accessed: 2013-05-22.

Android Developers, . Fragments, 2013b. URL `http://developer.android.com/guide/components/fragments.html`. Accessed: 2013-05-25.

Android Developers, . Geocoder, 2013c. URL `http://developer.android.com/reference/android/location/Geocoder.html`. Accessed: 2013-05-25.

Balcombe, Richard; Mackett, Roger; Paulley, Neil; Preston, John; Shires, Jeremy; Titheridge, Helena; Wardman, Mark, and White, Peter. The demand for public transport: a practical guide. 2004.

Bohannon, Richard W. Comfortable and maximum walking speed of adults aged 20—79 years: reference values and determinants. *Age and ageing*, 26(1):15–19, 1997.

Bozkurt, Mehmet; Groth, Roger; Hansson, Björn; Harrie, Lars; Ringberg, Peter;

Stigmar, Hanna, and Torpel, Karl. Towards extending web map services for mobile applications. In *ICA Workshop on generalisation and multiple representation*, 2005.

Cassidy, S and White, PR. Use and perceptions of a real time passenger information system. *Journal of advanced transportation*, 29(1):27–39, 1995.

Chittaro, Luca. Visualizing information on mobile devices. *Computer*, 39(3):40–45, 2006.

Dziekan, Katrin and Kottenhoff, Karl. Dynamic at-stop real-time information displays for public transport: effects on customers. *Transportation Research Part A: Policy and Practice*, 41(6):489–501, 2007.

Eastman, R.J. Guide to gis and image processing. page 144, 2001.

Egeler, C. Multimodal travel information service for transport in the tri-national agglomeration of basel based on real time data. In *Proceedings of the 1st Swiss Transport Research Conference, Ascona, Switzerland, 1-3 March*, volume 1, 2001.

Flightradar24 AB, . Flightradar24 live air traffic - how it works, 2013. URL `http://www.flightradar24.com/how-it-works`. Accessed: 2013-04-30.

Forsyth, E and Silcock, DT. Evaluation of real-time information on the arrival of urban public transport services. In *Seminar on public transport operations research*, 1985.

Frisco, Jeffrey A; Keen, Robert M, and Latta, Glenn S. Aircraft in-flight entertainment system providing passenger specific advertisements, and associated methods, May 2 2003. US Patent App. 10/428,234.

Furieri, Alessandro. Spatialite, 2011. URL `https://www.gaia-gis.it/fossil/libspatialite/index`. Accessed: 2013-05-23.

German Federal Ministry of Education and Research, . Mobility in conurbations - first results. Bonn, 2002. Federal Ministry of Education and Research (BMBF).

Good, M.; Spine, T. M.; Whiteside, J., and George, P. User-derived impact analysis as a tool for usability engineering. *SIGCHI Bull.*, 17(4):241–246, 1986.

Google Developers, . The google directions api, 2013. URL `https://developers.google.com/maps/documentation/directions/`. Accessed: 2013-05-26.

Gorlenko, Lada and Merrick, Roland. No wires attached: Usability challenges in the connected mobile world. *IBM Systems Journal*, 42(4):639–651, 2003.

GoTic, . Effects of real-time information in gothenburg (research report no. 19). GoTiC: Gothenburg Transport Information Centre, 2002.

Grotenhuis, Jan-Willem; Wiegmans, Bart W, and Rietveld, Piet. The desired quality of integrated multimodal travel information in public transport: Customer needs for time and effort savings. *Transport Policy*, 14(1):27–38, 2007.

Hare, K. Clearing the air. 117(6):45–6, 1997.

Harrington, Winston and McConnell, Virginia. A lighter tread?: Policy and technology options for motor vehicles. *Environment: Science and Policy for Sustainable Development*, 45(9):22–39, 2003.

Heidmann, Frank; Hermann, Fabian, and Peissner, Matthias. Interactive maps on mobile, location-based systems: design solutions and usability testing. In *Proceedings of the 21st International Cartographic Conference*, pages 10–16, 2003.

Infopolis2, . Review of current passenger information systems (deliverable 1. Commission of the European Community, 1998.

ISO, 9241. Ergonomic requirements for office work with visual display terminals (vdts), 1992/2001.

Kane, Lalit; Verma, Bhupendra, and Jain, Sanjeev. Vehicle tracking in public transport domain and associated spatio-temporal query processing. *Computer Communications*, 31(12):2862–2869, 2008.

Kenyon, Susan and Lyons, Glenn. The value of integrated multimodal traveller

information and its potential contribution to modal change. *Transportation Research Part F: Traffic Psychology and Behaviour*, 6(1):1–21, 2003.

Lehtonen, Mikko and Kulmala, Risto. Benefits of pilot implementation of public transport signal priorities and real-time passenger information. *Transportation Research Record: Journal of the Transportation Research Board*, 1799(-1):18–25, 2002.

Looije, Rosemarijn; te Brake, Guido M, and Neerincx, Mark A. Usability engineering for mobile maps. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pages 532–539. ACM, 2007.

Çöltekin, Arzu; Heil, Benedikt; Garlandini, Simone, and Fabrikant, Sara Irina. Evaluating the effectiveness of interactive map interface designs: A case study integrating usability metrics with eye-movement analysis. *Cartography and Geographic Information Science*, 36(1):5–17, 2009.

Meng, Liqiu. Ego centres of mobile users and egocentric map design. In *Map-based Mobile Services*, pages 87–105. Springer, 2005.

Nielsen, J. and Hackos, J.A.T. *Usability engineering*, volume 125184069. Academic press San Diego, 1993.

Nijkamp, Peter; Pepping, Gerard, and Banister, David. *Telematics and transport behaviour*. 1996.

Norangshol, Roy Sindre. Busbuddy, 2011. URL `http://api.busbuddy.no/`. Accessed: 2013-05-23.

OpenStreetMap Wiki, . Cles mapnik.png, 2010. URL `http://wiki.openstreetmap.org/w/images/1/17/Cles_mapnik.png`. Accessed: 2013-05-23.

OSMDroid, . Osmdroid - openstreetmap-tools for android, 2012. URL `https://code.google.com/p/osmdroid/`. Accessed: 2013-05-15.

Pavlenko, Artem. Mapnik, 2011. URL http://mapnik.org/faq/. Accessed: 2013-05-23.

Raper, J. Gis, mobile and locational based services. In in Chief: Rob Kitchin, Editors and Thrift, Nigel, editors, *International Encyclopedia of Human Geography*, pages 513 – 519. Elsevier, Oxford, 2009. ISBN 978-0-08-044910-4. doi: 10.1016/B978-008044910-4.00055-9. URL http://www.sciencedirect.com/science/article/pii/B9780080449104000559.

Reichenbacher, Tumasch. Adaptive concepts for a mobile cartography. *Journal of Geographical Sciences*, 11:43–53, 2001a.

Reichenbacher, Tumasch. The world in your pocket-towards a mobile cartography. In *Proceedings of the 20th international cartographic conference*, pages 2514–2521. Beijing,[SN], 2001b.

Sarjakoski, L Tiina and Nivala, Annu-Maaria. Adaptation to context—a way to improve the usability of mobile maps. In *Map-based Mobile Services*, pages 107–123. Springer, 2005.

Sekara, V and Karlsson, M. A field evaluation of real time information at tram and bus stops. part 2. *Traffic & Public Transport Authority City of Gothenburg*, 1997.

SQLite, . About sqlite. URL http://sqlite.org/about.html. Accessed: 2013-04-30.

Stradling, S; Hine, J, and Wardman, M. Physical, cognitive and affective effort in travel mode choices. In *International Conference On Traffic And Transport Psychology-Icttp 2000, Held 4-7 September 2000, Berne, Switzerland-Keynotes, Symposia, Thematic Sessions, Workshops, Posters, List Of Participants And Word Viewer-Cd Rom*, 2001.

Virrantaus, K.; Markkula, J.; Garmash, A.; Terziyan, V.; Veijalainen, J.; Katanosov, A., and Tirri, H. Developing gis-supported location-based services. In *Web Information Systems Engineering, 2001. Proceedings of the Sec-*

ond International Conference on, volume 2, pages 66 –75 vol.2, dec 2001a. doi: 10.1109/WISE.2001.996708.

Virrantaus, Kirsi; Markkula, Jouni; Garmash, Artem; Terziyan, Vagan; Veijalainen, Jari; Katanosov, Artem, and Tirri, Henry. Developing gis-supported location-based services. In *Web Information Systems Engineering, 2001. Proceedings of the Second International Conference on*, volume 2, pages 66–75. IEEE, 2001b.

Von Hunolstein, Stefan and Zipf, Alexander. Towards task oriented map-based mobile guides. In *Proceedings International Workshop" HCI in Mobile Guides" at Mobile HCI 2003, 5th International Symposium on Human Computer Interaction with mobile Devices and Services*, pages 8–11, 2003.

Watkins, Kari Edison; Ferris, Brian; Borning, Alan; Rutherford, G Scott, and Layton, David. Where is my bus? impact of mobile real-time information on the perceived and actual wait time of transit riders. *Transportation Research Part A: Policy and Practice*, 45(8):839–848, 2011.

Wertheimer, Max. Untersuchungen zur lehre von der gestalt. ii. *Psychological Research*, 4(1):301–350, 1923.

Zipf, Er; Aras, Hidir, and others, . Proactive exploitation of the spatial context in lbs-through interoperable integration of gis-services with an multi agent system. 2002.

Appendices

Appendix A

Retrieving Departure Data for a Bus Stop

Listing A.1: Code snippet for retrieving real-time data

```
public class ScheduleTimeTask extends AsyncTask<URI, Void,
    JSONObject[] > {
  private static final String TAG = "HTTP_TASK";

  HttpTaskHandler taskHandler;

    @Override
    protected JSONObject[] doInBackground(URI...urls) {
      HttpClient client = new DefaultHttpClient();

        JSONObject[] json = new JSONObject[urls.length];
        for (int i = 0; i < urls.length; i++) {
          HttpGet httpGet = new HttpGet(urls[i]);

          try {
              HttpResponse response =  client.execute(httpGet);
```

```
            BufferedReader reader = new BufferedReader(new
                InputStreamReader(response.getEntity().
                getContent(), "UTF-8"));
        StringBuilder builder = new StringBuilder();
        for (String line = null; (line = reader.readLine())
            != null; ) {
            builder.append(line).append("\n");
        }
      JSONTokener tokener = new JSONTokener(builder.toString
          ());
      json[i] = new JSONObject(tokener);

          } catch (Exception e) {
            Log.e(TAG,e.toString());
        e.printStackTrace();
        return null;
      }
    }
      return json;
}

@Override
protected void onPostExecute(JSONObject[] json) {
      if(json != null) {
        System.out.println("Schedule time task success");

        System.out.println(json[0].toString());

        if(schedulesByStopListFromDest.size()>0){
          schedulesByStopListFromDest.clear();
        }

          for (int j = 0; j < json.length ; j++) {
            JSONArray schedules = new JSONArray();
            String busStopId = null;
            try {
              schedules = json[j].getJSONArray("schedule");
        } catch (JSONException e) {
```

```
  e.printStackTrace();
}
    try {
  String line = json[j].getString("busstopID");
  busStopId = line;
} catch (JSONException e) {
  e.printStackTrace();
}

    for (int i = 0; i < schedules.length(); i++) {
      JSONObject c = new JSONObject();
      String linenr = null;
      String name = null;
      try {
    c = schedules.getJSONObject(i);
  } catch (JSONException e1) {
    e1.printStackTrace();
  }
      try {
        linenr = c.getString("line");
  } catch (JSONException e) {
    e.printStackTrace();
  }
      try {
        name = c.getString("name");
  } catch (JSONException e) {
    e.printStackTrace();
  }

      JSONArray departures = new JSONArray();
      try {
        departures = c.getJSONArray("departures");
  } catch (JSONException e) {
    e.printStackTrace();
  }
      length = departures.length();

      for (int k = 0; k < length; k++) {
```

```
                    SchedulesByBusStop schedule = new
                        SchedulesByBusStop();
                    schedule.setBusStopId(busStopId);
                    schedule.setLine(linenr);
                    schedule.setDestination(name);

                    JSONObject d = new JSONObject();
                    try {
                  d = departures.getJSONObject(k);
                } catch (JSONException e1) {
                  e1.printStackTrace();
                }
                    try {
                  int line = d.getInt("id");
                  schedule.setTripId(line);
                } catch (JSONException e) {
                  e.printStackTrace();
                  break;
                }
                    try {
                  String line = d.getString("t");
                  schedule.setScheduledDepartureTime(line);
                } catch (JSONException e) {
                  e.printStackTrace();
                }

                    schedulesByStopListFromDest.add(schedule);
                  }
                }
              }
            new TripTask().execute("start");
          }
      }
}
```

Appendix B

Obtaining the Distance Between Two Locations

Listing B.1: Code snippet for obtaining the distance between two locations

```
private class BusStopsDistanceTask extends AsyncTask<String,
    Void, String>{
    @Override
    protected String doInBackground(String... locations) {

            //If there is already distance values (old) in DB -
                delete these
            if(db.distanceTableHoldsValues()){
              System.out.println("Distance table holds values -
                  Deleting values");
              db.deleteDistanceEntriesInDB();
            }
            else{
              System.out.println("Distance Table in DB is empty")
                  ;
            }

            DistanceBusStops object = null;
```

```
GeoDistance getDistance = new GeoDistance();

    //Gets distances for the "from" location -
       departure location
    if(locations[0].equals("from")){
for (int i = 0; i < busStopList.size(); i++) {
        if(Math.abs((getFromPosition().latitude -
           Double.parseDouble(busStopList.get(i).
           getLatitude()))) < maxLatitudeDistance()
           &&  Math.abs((getFromPosition().longitude
           - Double.parseDouble(busStopList.get(i).
           getLongitude()))) < maxLongitudeDistance()
           ){
          LatLng end = new LatLng(Double.parseDouble(
             busStopList.get(i).getLatitude()),
             Double.parseDouble(busStopList.get(i).
             getLongitude()));
          String distance = getDistance.
             GetRoutDistane(getFromPosition().
             latitude, getFromPosition().longitude,
             end.latitude, end.longitude, "walking");

          //Add values in DB
          db.insertDistancesInDB(busStopList.get(i).
             getName(), distance, busStopList.get(i).
             getLocationId(), "from");

        }
}
    }

    //Gets distances for the "to" location -
       destination location
    if(locations[1].equals("to")){
       for (int i = 0; i < busStopList.size(); i++) {
          if(Math.abs((getToPosition().latitude -
             Double.parseDouble(busStopList.get(i).
             getLatitude()))) < maxLatitudeDistance()
```

```
                        &&  Math.abs((getToPosition().
                        longitude - Double.parseDouble(
                        busStopList.get(i).getLongitude()))) <
                        maxLongitudeDistance()){
                        LatLng end = new LatLng(Double.
                            parseDouble(busStopList.get(i).
                            getLatitude()), Double.parseDouble(
                            busStopList.get(i).getLongitude()));
                        String distance = getDistance.
                            GetRoutDistane(getToPosition().
                            latitude, getToPosition().longitude,
                            end.latitude, end.longitude, "walking
                            ");

                        //Inserting values in DB
                        db.insertDistancesInDB(busStopList.get(i)
                            .getName(), distance, busStopList.get(
                            i).getLocationId(), "to");
                    }
                }
                  }

                distanceListFromDest = db.getDistanceListByLoc("
                    from");
                distanceListToDest = db.getDistanceListByLoc("to");

                return "finished";
        }

        @Override
        protected void onPostExecute(String params) {
                tripPlanner();
        }
}
```

Appendix C

Method for Selection of Possible Journey Routes

Listing C.1: Code snippet for selection of possible routes

```
public class GetTravelRouteTask extends AsyncTask<String, Void,
    PossibleRoute> {

@Override
protected PossibleRoute doInBackground(String...routes) {

    //Get all bus stop IDs from the distance table in DB close
       to the "to" location - destination
  List<String> idToDest = db.getAllBusStopIDfromDistanceTable("
     to");

  for (int i = 0; i < schedulesByStopListFromDest.size(); i++)
     {
    for (int j = 0; j < tripsList.size(); j++) {

      if(String.valueOf(schedulesByStopListFromDest.get(i).
         getTripId()).equals(tripsList.get(j).getTripID())
          && schedulesByStopListFromDest.get(i).getLine().
```

```
      equals(tripsList.get(j).getLine()) &&
      schedulesByStopListFromDest.get(i).getBusStopId().
      equals(tripsList.get(j).getBusStopId())){

for (int k = 0; k < idToDest.size(); k++) {
  boolean containsStartAndEnd = doesTripContainStop(
     schedulesByStopListFromDest.get(i).getBusStopId(),
      idToDest.get(k), tripsList.get(j));
  if(containsStartAndEnd){

    System.out
        .println("tripid "+schedulesByStopListFromDest.
           get(i).getTripId());

    //Is a possible route
    PossibleRoute posRoute = new PossibleRoute();

    List<DistanceBusStops> distObj = db.
       getDistanceObjectByLocID(
       schedulesByStopListFromDest.get(i).getBusStopId
       ());
    String dist = distObj.get(0).getDistance();
    String res = "";

    System.out.println("dist to close from stop " +
       dist);

    if(dist.charAt(0)!='T' && dist!=null){
      //If value is given in km (e.g. 1.3 km)
      if(dist.contains(".")){
        for (int p= 0; p < dist.length(); p++) {
          res+= dist.charAt(p);
          if(dist.charAt(p+1) == ' '){
            break;
          }
        }
        double distvalue = Double.parseDouble(res);
        posRoute.setW1((distvalue/WALKING_SPEED)*
```

```
        MINUTES_IN_HOUR);
      System.out.println("min to walk " +(distvalue/
        WALKING_SPEED)*MINUTES_IN_HOUR);
    }
    //Else if value is given in meter (e.g. 70 m)
    else{
      for (int o= 0; o < dist.length()-1; o++) {
        res+= dist.charAt(o);
        if(dist.charAt(o+1) == ' '){
          break;
        }
      }
      double distvalue = Double.parseDouble(res);
      double distvalueInKm = distvalue/100;
      posRoute.setW1((distvalueInKm/WALKING_SPEED)*
        MINUTES_IN_HOUR);
      System.out.println("min to walk " +(
        distvalueInKm/WALKING_SPEED)*MINUTES_IN_HOUR
        );
    }
  }

  //Can user make it to the bus stop in time?

  //String s1 is on the format e.g. "2013-95-15
     11:53" and needs to be modified
  String s2mod =  schedulesByStopListFromDest.get(i).
     getScheduledDepartureTime().substring(11, 16);
  System.out.println("deptime " +s2mod);
  boolean isTimePassed = TimeIsPassed(getTimeRightNow
     (),s2mod);

  if(!isTimePassed){
    int minToDep = getTimeUntilBusDeparts(
       getTimeRightNow(), s2mod);

    System.out.println("min to dep " +minToDep);
```

```
                if((minToDep - posRoute.getW1()) >= 0){

                posRoute.setWait_time(minToDep - posRoute.getW1
                    ());

                posRoute.setStartBusStopId(
                    schedulesByStopListFromDest.get(i).
                    getBusStopId());
                posRoute.setEndBusStopId(idToDest.get(k));
                posRoute.setLine(schedulesByStopListFromDest.
                    get(i).getLine());
                posRoute.setTripId(schedulesByStopListFromDest.
                    get(i).getTripId());
                posRoute.setDepartureBusTime(
                    schedulesByStopListFromDest.get(i).
                    getScheduledDepartureTime());
                posRoute.setTrips(tripsList.get(j));

                possibleRouteList.add(posRoute);
                }
              }
            }
          }
        }
      }

    //Get the best route based on weight of all the possible
        routes
    PossibleRoute bestRoute = getBestPossibleRoute();
}
```