City EXPLORER

Authors (group 10):

John Arne Øye
Oscar Aarseth
Jaroslav Rakhmatoullin
Kristian Greve Hagen
Christian Berg Skjetne
Maria Belen Gallego Garcia

Supervisor:
Alfredo Perez Fernandez

IT2901 - Informatics Project 2
Department of Computer and Information Science

NTNU
Norwegian University of Science and Technology
May 2011

Table of Contents

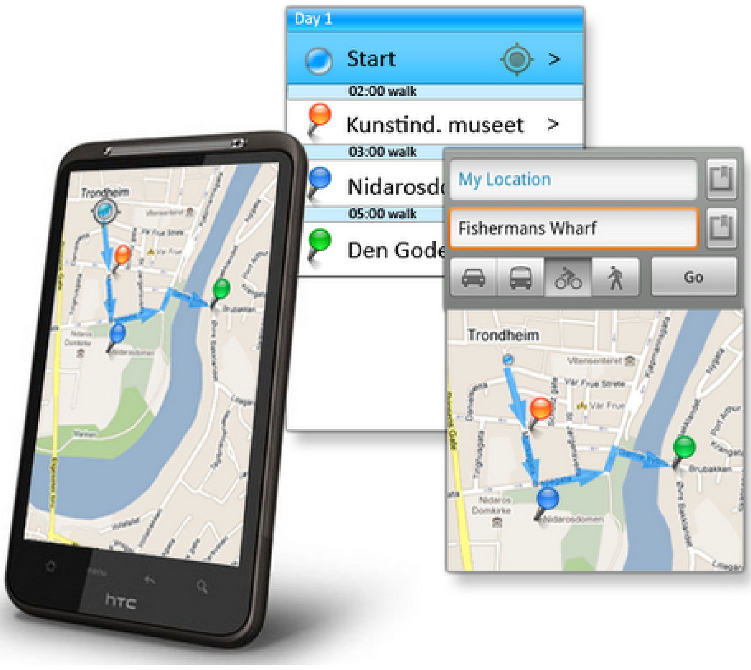
Chapter 1

Introduction

The following document is the result of the work of six students on a project centered around the science of planning and developing an information system. The project emulates a real software development endeavor because an external customer is serving the request. As such the project mandates an elaborate documentation of both the development process and the resulting product. In our case, the product is a tourist application for mobile devices running Android 2.2[1].

1.1 Product description

The core idea The City Explorer is an attempt to fill a gap in the market for mobile tourist guides. In addition to facilitating the work of tourist offices, City Explorer also enables regular tourists to contribute. The main objective is to help potential users in exploration of new cities by giving them a pool of existing points of interest and tours, as well as allowing users to create their own points of interest and tours.



Details A tour is a collection of points of interest, such as sightseeing destinations, churches or even shopping malls. The content of a tour is not limited to a specific field of interest beyond what might be associated with the provider who owns it, such as the tourist office or an architecture association. In order to obtain such tours, the user has to pick a provider from an internally stored list and browse through their repository of tours. The user may also download single Point of Interest (PoI) without selecting a particular tour. Once the user has obtained or created an interesting tour, they will be able perform a number of activities with highlights including:

- Getting directions (navigation) to a particular destination in the tour using Google Maps
- Displaying a description, a picture and other useful information associated with a particular destination
- Scheduling a tour by their preferences or by following an attached time schedule, if the creator of the tour has specified this
- Displaying a map with a path through all of the destinations as well as viewing each stop separately
- Viewing all stops in a tour simultaneously on a map without the suggested path between them
- Going through a tour (both as an actual tour or simply by browsing all the destinations on the device at home)
- Adding destinations to a tour and removing undesired destinations.

Chapter 2

Project Management

A project is usually defined as a finite event. It has objectives, tasks and limitations in human resources as well as the implicit time constraint. This chapter describes a plan of actions during the three phases our project will traverse and presents an oversight of the activities within those phases. We will also discuss the applied tools and techniques and how they have improved the outcome.

2.1 Pre-studies

According to research conducted by the client[2], there is a number of similar applications. Unfortunately, they lack in functionality on one or several of the following points:

- exploitation of potential context awareness inherent to mobile devices
- geographical wideness (support for many cities)
- quality of the user interface
- tailoring tours according to users' desires

While the collected opinions of potential users express an interest in a better application, they were not the direct cause of our client's efforts to suggest an application as specified by the requirements specification (see chapter 3). The goal of our client is rather exploration in the area of end user composition of mobile services[3]. In order to facilitate the development of the application, our client has chosen the open source Apache license.

2.1.1 Alternative Solutions

The description of the project is rather specific and the client's requests are unambiguous. There might be room for suggestions, but not misinterpretation.

Existing applications do provide interactive maps, information and descriptions of places, selection of favourites and type-based filtering of destinations. The City Explorer is not very original in that respect, but two features of City Explorer that in our opinions make this project very interesting are

- decentralized generation of contents (as in there may be several service providers and users may share their own places and tours)
- an open source model with a plug-in system as an important (future) goal.

These characteristics turn City Explorer into a potentially global application which in our view is more exciting than a regular school project.

2.2 Project Schedule

2.2.1 Phases

The completion of the project is preceded by three phases. They are the "Plan and Design", "Iterative Development" and "Last Call". All three phases are estimated to fit within the time frame of 16 weeks starting on the 20th of January and ending on the 15th of May. These dates coincide with the dates of our first meeting and the final deadline for this document.

Plan and Design The initial phase ran for three weeks, starting on 20th of January and ending on 6th of February. This time was spent working out the details of the schedule and establishing routines.

Iterative Development The main part of the project ran for 12 weeks with one week in the end set aside for testing. It started on 7th of February and ended on 8th of May. The main objective in this phase was to produce a working application. Every iteration will be referred to as a "sprint". Documentation for each sprint is worked out at the beginning of the sprint in a "sprint backlog".

Last Call The finalization of this document will find place in the last phase and final week of our schedule. It started on 9th of May and ended on 15th of May. Some important tasks will include proof reading, removal of reproduced information, review by external parts and formatting.

2.2.2 Milestones

The iterative phase of the project is defined by a few milestones. They are intermediate project objectives and help us to keep the project on schedule. A secondary purpose of the milestones is to tell the client when she can expect the various requirements to be implemented.

The milestones concerning this report are set by the course coordinator, while milestones concerning the requirements were set by us during the planning phase. Unfortunately, due to unforeseen delays, the milestones were postponed (See appendix B section B.2). The following list contains the updated dates for the completion of the various parts of the application.

31 Jan.	Delivery (16:00): Project report - VERY preliminary version
28 Feb.	Delivery (16:00): Project Report - mid-semester version
13 Mar.	Basis requriments fullfilled
08 Apr.	Delivery (16:00): Project report - for final comments from supervisor.
25 Mar.	Tailoring requirements fullfilled
15 Apr.	Sharing requriments fullfilled
02 May	Delivery of application to the client for final testing
15 May	Delivery (16:00): Project report - final version
23-24 May	Presentation of projects

2.3 Stakeholders

Since this is not a big commercial project, the number of interested parties is limited to the client and the six members responsible for the development. The client is presumably conducting market research and is looking to entice third parties in future development by demonstrating the merits of this project. Our interest is simple, we like to write Java^{TM}code[4] and aim to get a good grade in this course.

2.3.1 The Client

Our client is the independent research institute SINTEF[5] represented by Jacqueline Floch. She is responsible for the unpublished paper "A Framework For User Tailored City Exploration"[2]. In the context of this paper, she has requested this application. We are also told that this will be incorporated under the larger umbrella of the research project UbiCompForAll[3]. The idea of UbiCompForAll is about providing support to end users so they can easily compose service behaviours in ubiquitous service environments.

2.3.2 Team Organization

At the start of this project, the group consisted of the six authors mentioned on the front page. We have chosen not to assign specific roles to any of the members to divide the responsibility for the project equally among all members. This flat structure seems suitable because everyone involved will receive the same grade. Besides, when everyone shares roles, it is impossible to blame failure in a specific area of the project (e.g. management, documentation, development) on a single individual. The decision to have a flat group structure has not been without problems. This is discussed in more detail in the work process reflections section, 2.8.

2.4 Development model

A traditional Software Development Life Cycle is based on a structured step-by-step approach to developing systems[1]. The steps which may be considered normal in such a model include a preliminary investigation, requirement analysis, logical and physical design, implementation, maintenance and deployment. None of the steps may be executed before the preceding step is finished. Because of this, the whole process can be slow and cumbersome. It does not allow a team to produce prototypes in a short time period and the users of the application are involved only in the initial phases. This means that any feedback from the users has to wait until the first public release candidate, which may be too late in our case if the client finds our implementation unsatisfactory.

Rapid Application Development is a more recent and flexible approach to development, some times referred to as Agile Development models. The

[1]The paragraphs about RAD and SDLC are based on pp. 189-194[6]

main idea of RAD is to compress all of the phases found in a traditional SDLC into several short iterations. From the iterative approach (repeating all phases) follows a clearer understanding of the requirements for the system as well as a closer match of user expectations. This is not an argument against the quality of requirement analysis in traditional SDLCs, but it is very important in our case to understand the problem at hand from early on and be able to adopt to modifications in the requirements.

This point is crucial for our decision to adopt a RAD approach and to borrow elements from different Agile methodologies in our development model. Other reasons include:

- We can not spend a substantial portion of the allocated time in the beginning of the project developing an elaborate plan because of the client's request for a prototype after the second meeting
- The client has warned us that some of the requirements may change because some aspects of the system are still under investigation
- We are free to suggest and implement new functionality at any time during the project
- The client requested weekly meetings with reports and presentations of the application. For meeting minutes, see appendix A.
- Some of the group members are familiar with elements from SCRUM [7] and Extreme Programming [8]
- A rigid time limit
- The different modules of the application can be developed in parallel to each other

2.4.1 Borrowed elements

Our development model consists of several elements from Agile models such as SCRUM and Extreme Programming. We will now present a descriptions of those elements and how we have adopted them.

The Backlog The backlog is located in appendix C. The concept of backlog offered by Scrum encourages developers to record all features of an application. The content of the recorded list is usually based on use scenarios. All items in this list are prioritized according to the needs of the client while the time needed to complete each task is estimated by the team. The most important tasks are then chosen for implementation in the next sprint (RAD iteration). That point in our backlog is identical to any other backlog, but other concepts vary:

- Our items' priority is not displayed in the backlog because they are based on the client specified priorities from the requirements specification (see section 3)
- Our items represent work packages (see section 2.1) as well as individual tasks or modules of the application

Pair Programming This concept is borrowed from Extreme Programming and is rather self explanatory. Our slight modification of this concept is that we use several computers instead of one and a pair programming session may have more than two participants. We find this way of working very helpful because we are able to help each other with problems as soon as they arise.

2.5 Development environment

In order to save valuable time at the beginning of the project, we chose to write the application in the Eclipse IDE. The time saving argument is not the only one that tipped the scale. All group members were already familiar with Eclipse and we are thus able to help each other if any problems arise with either the version control back-end, or during compilation. Further, if the knowledge shared among us turns out to be insufficient to solve a particular problem, we can turn to the large community of Eclipse users and developers. This is a good option to have, because the probability of getting stuck with some problem for a long time is reduced. Furthermore, Eclipse supports the Android SDK through a plug-in which facilitates the conversion to Dalvik bytecode (Android Java VM) and the creation of binary packages for Android.

For centralized version control, we chose SVN[9] because it is provided to us by the university. SVN is also what we have used previously because it facilitates sharing of code between several developers.

SQLite[10] will be used to store the data internally in the application. It is widely used in Android development, and by using it we are adhering to the non-functional requirement 4.6 (using existing Android building blocks). We are designing primarily with a local database because the requirements for sharing amongst users of the application is not of high priority.

The application is written to run under Android on mobile phones and larger hand-held devices such as tablets. This is a requirement from our customer. The non-functional requirements can be found in the requirement specification chapter (see section 3.4).

To communicate within our team, we decided to use Internet Relay Chat (IRC), e-mail and 1 weekly group meeting. For meeting minutes, see appendix A. After trying IRC for a while, we found out that it was much easier to create a group on Facebook[11]. A short discussion of this can be found in the work process reflections section, 2.8. Communication with client is done in the weekly meetings and by e-mail. Our supervisor will meet us every other week and is also available on e-mail.

2.6 Risk Analysis

This risk list (Table 2.1) is a plan of actions to avoid risks and to minimize their consequences if they should occur. We have chosen to display the risks using a formula that states Likelihood * Impact = Severity. What this means is that the estimated probability of an accident occurring multiplied with the estimated loss of progress if the accident should occur, gives us an idea of the importance to avoid a certain risk. The higher the severity, the more important it becomes to prevent this accident and to know what adjustments to make if we are unable to prevent it.

The scale of severity has an upper bound which is the smallest square of an integer, the multitude of which will be enough to itemize all element in the table by an integer. If there are 9 elements in the list, likelihood and impact will be on a scale from 1 to 3. Where higher values have a higher importance. If there are 10-16 elements, they will be on a scale between 1 and 4, and so on. This scale makes the severity of each risk relative to the other risks.

The list also contains all the preventive actions we are taking and all remedial actions we are prepared to take for every risk.

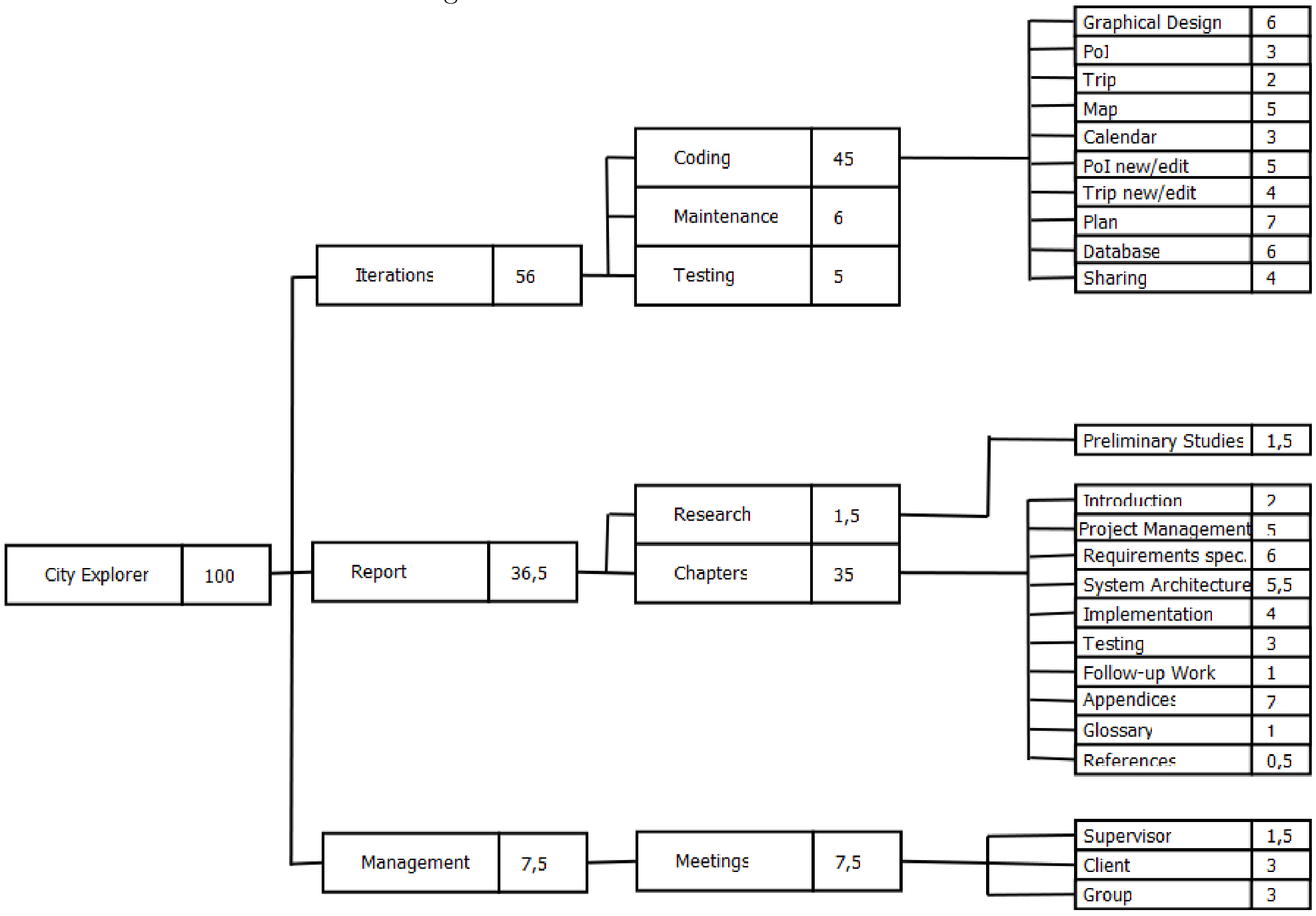
Table 2.1: Risk assessment

Description *Likelihood ∗ Impact = Severity*	**Preventive action**	**Remedial action**
Insufficient time		
$3 * 3 = 9$	Have an overview of the time we have to our disposal.	Talk to each other and keep an eye on the deadlines
Data loss		
$2 * 3 = 6$	Create daily archives from the central SVN	Recover from archives.
Incompetence		
$2 * 3 = 6$	Plan and document implementation strategy before coding, document failed implementation attempts	Help each other, try and find alternative solutions
Illness		
$2 * 2 = 4$	Avoid infecting others, dress well, keep good hygiene, get enough sleep	Wait to recover, redistribute work, work from home
Member(s) shirking work		
$2 * 2 = 4$	Deadlines within the group	Talk to each other how we are doing on our part and help each other if we are stuck at some point
Conflicts within group		
$1 * 2 = 2$	Have an open discussion. Do not overrun others' ideas	Solve the conflict or have a talk with the supervisor
Group member drops out		
$1 * 2 = 2$	Motivate, be inclusive by assigning roles, perform social activities	Notify supervisor. Reassign member's roles
Unforeseen delays		
$1 * 2 = 2$	Work more every week to keep a safe-buffer	Distribute the work to unaffected members
Server failure		
$1 * 1 = 1$	Run 'svn update' on the whole repo every day	Install git[12]

2.7 WBS

In order to estimate the time we need to complete the project, a work breakdown structure[13] graph (see figure 2.1) has been used. The number to the right of each label represents a percentage of the total time allocated to the project. Each work package is estimated to consume no more than the indicated percentage.

Figure 2.1: Work breakdown structure



2.8 Work process reflections

This chapter contains a description of the work process. It is intended that this chapter will give a more detailed view of the project. For a summary of the weekly process, see the status reports located in appendix B.

In the start of the project the group had some problems with communication. Our initial attempt at setting up a convenient communication channel for all members in the form of an IRC channel failed due to our different habits. Assuming that all members were online at all times, proved to be a mistake. While regular email got the messages across to all members, not everyone found it to be very convenient. A solution that proved successful, despite reluctance from some members was a Facebook group. We figured that posting messages there would be optimal because most members visit the site on a regular basis.

Partially due to the communication problems in the beginning, we also had a difficult time producing the first prototype. We had not devoted enough time to a discussion of what the application should be like, resulting in confusion and conflicting views among members. To resolve this, we decided to make a GUI flow chart together (see figure 4.2). This gave us a clearer understanding of what the application should be like and how we would fulfill the various requirements. Another reason for the slow start was that most of us had not worked with Android before. Things went more smoothly after one of the members, who did have previous experience with Android, wrote the start activity demonstrating the basic approach.

Oscar was at the hospital during the first three weeks of the project, and had already fallen a bit behind when we delivered the preliminary version of the report. As he had no previous experience with the Android programming platform, more time than expected was used to learn this, and his direct contribution to the group was limited as a result of this.

Still, the process was not optimal because the work load was unevenly distributed. Several factors contributed to this, with the most important being high time demands in other courses, illness amongst members and poor planning of group activities. Our self organizing of the team was apparently a bad choice. Despite the code that was available for studying in SVN, not everyone was certain about how to write the application and certainly everyone left the writing of the report to everyone else. The problem became very evident in sprint 4, which was one week before the midterm report was due. The active members' attempts to motivate the rest of the group to work with the report were not very successful during that week, resulting in a mid term report that did not meet our supervisor's expectations.

One week after the midterm delivery, Belen informed us by e-mail that she was dropping the subject. She felt incapable of contributing in a satisfactory manner, listing time constraints and lack of knowledge as reasons for this. See the status reports in appendix B for further information and consequences. One problem leading up to this was that we did not distribute responsibilities. This made it harder to figure out what each member was supposed to do at a given time. Another problem was that we did not use a lot of time to map out the members individual abilities and knowledge. This could have been very useful for addressing and avoiding these problems before they occurred. Another problem may be that the group did not discuss the individual members expectation of the course in terms of workload and grade. Having this information makes for more correct expectations of the effort of other members, and could also have helped us identify potential problems before they occurred.

After Belen left, Oscar realised that his contribution did not really outweigh hers at this time, and he asked the group if he should leave as well. But during a meeting we had with the supervisor, it got explained to us that it was possible to have a sit down with the teacher to discuss expectations for this course, and try to come up with a solution. Even though Oscar wanted to use this option, we strongly recommended him not to. As the trouble around Belen leaving had made the group look disorganized enough. Oscar have been ill for a big part of this semester, and it has greatly affected his workload. Not only has he been away when hospitalized, but his physical condition has been greatly reduced and he has technically been on sick leave for 8 weeks during this semester. He has chosen to ignore this and still tried to participate for most of these weeks. Documentation for Oscar's sick leave can be provided if necessary. Despite this setback, work on the project was stable during the rest of sprint 5.

During sprint 6 work went on as normal. The meeting with the client on 21st of March was cancelled, and at the meeting with the supervisor on the 25th of March we got criticised for not reporting all hours. We had struggled for some time to find a solution to how we could make the calendar view work. This was finally solved, and the work on making this feature started.

Sprint 7 started at the 28th of March and would primarily be about working on the report as we had our last evaluation from the supervisor at the 9th of April. We had delivered an earlier version of the report to the client and got some very helpful feedback on Monday the 4th of April. Thereafter a lot of time was spent working on her suggested improvements before we delivered the next version to the supervisor.

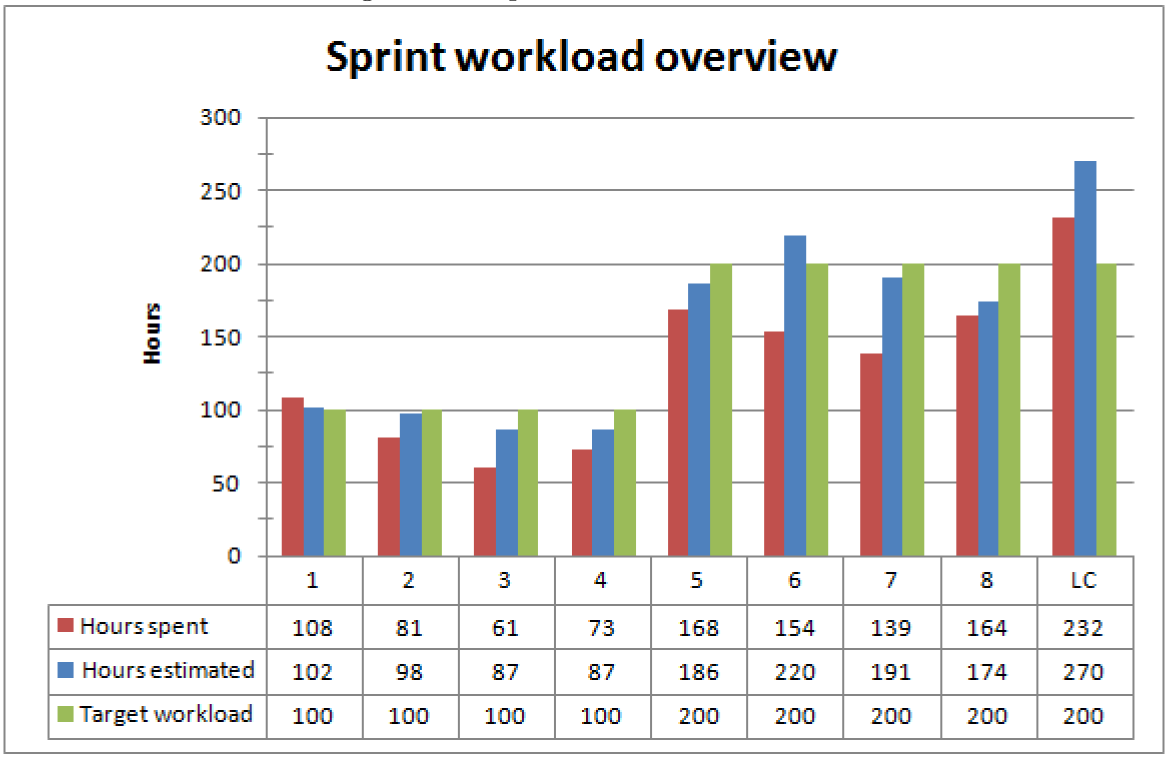
After this came the Easter holidays and even though we had not planned for this to happen, everyone took a break from work in the period of the 11th to the 24th of April.

We once again lost some manpower during this period when Oscar had to go back to the hospital for his second surgery. He scheduled his surgery to happen in the Easter holidays so he would be unavailable for as short a period as possible. Unfortunately, his surgery did not go as well as expected and he needed yet another week for recovery. Even though he informed the group about this right away, we lost valuable working hours in a crucial part of the project. At the same time we lost contact with Jaroslav. He stopped showing up at school and even though we continuously tried to contact him via email, phone and Facebook we could not reach him. As per the final delivery we have not yet been able to meet with him. Because we have not gotten a hold of him, he has not been contributing in the last weeks of the project, and the remaining group members had to take on additional work to compensate for the loss of manpower. This was the chosen solution as suggested by the risk assessment table (see section 2.6).

On the 11th of May we had a meeting with the supervisor regarding the problems we have had within the group. We explained the current situation and we all agreed that the most fair solution to our problem would be to ask for separate grading. We do realise it might be a bit late in the semester to report our troubles, but we still felt it was necessary to discuss this and ask for guidance on the matter.

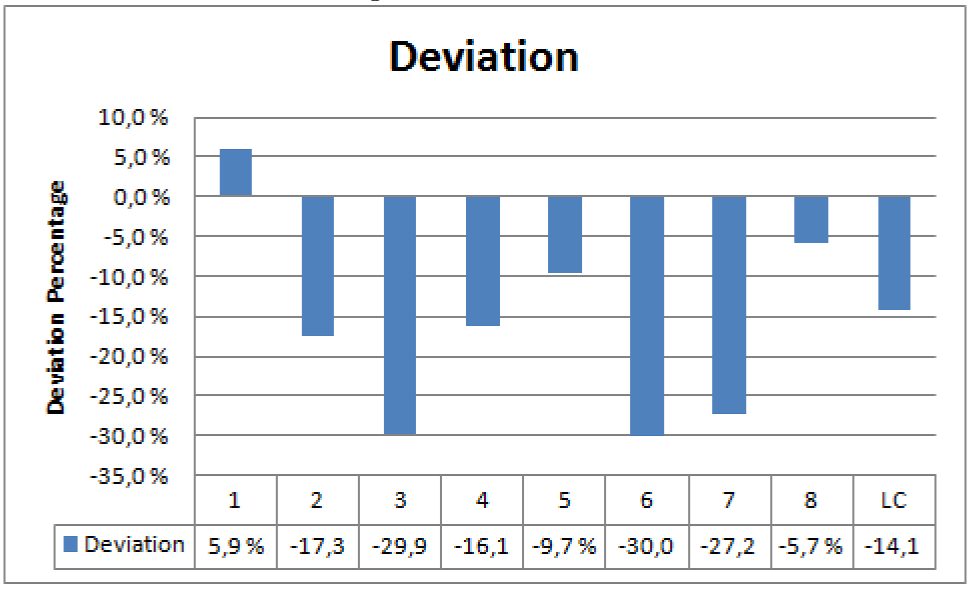
Working hours estimation To find out if our working estimates correlate to the actual hours spent we plotted the hours spent and the hours estimated, including the target workload per sprint in a chart (see figure 2.2). Starting from sprint one, up to sprint four, we had sprints lasting one week. Starting from sprint five, we increased the sprint length to two weeks per sprint. This explains the major jump in hours after week four. A better illustration of the actual deviation is shown in the deviation chart (see figure 2.3). This chart shows the deviation in percentage. If the percentage is zero, we have estimated correctly. If the percentage is above zero, we have underestimated the workload, and vice versa, if the percentage is under zero, we have overestimated the workload. As seen in the charts, we have overestimated the workload. The calculated average deviation is -16%, which means that we have to adjust our future estimates accordingly.

Figure 2.2: Sprint workload overview



	1	2	3	4	5	6	7	8	LC
🟥 Hours spent	108	81	61	73	168	154	139	164	232
🟦 Hours estimated	102	98	87	87	186	220	191	174	270
🟩 Target workload	100	100	100	100	200	200	200	200	200

Figure 2.3: Deviation



	1	2	3	4	5	6	7	8	LC
🟦 Deviation	5,9 %	-17,3	-29,9	-16,1	-9,7 %	-30,0	-27,2	-5,7 %	-14,1

Project work package distribution At the end of the project, the backlog is able to give a good indication of the how the group distributed working hours between the different parts of the project. The estimation of the workload of each part in the project was done at the beginning of the project, and is described in the WBS section of the report (see section 2.7). The two most important parts of the project are the Iterations(coding) and the report parts, estimated to 56 percent and 36,5 percent respectively. Calculations done on the backlog after the project was completed shows that the actual work done on the parts match the estimates with only small deviations. The group used 59,7 percent of the total hours on coding and maintenance, and 35,7 percent on the report. The group is pleased with the distribution of work between the parts, and the low deviation also shows that we have made a good initial estimate which we were able to follow throughout the project.

2.9 Work distribution

As mentioned in the work process reflections section 2.8 in this chapter, we have not contributed equally to the project. Because of this, we have included a table describing each person's work distribution in percentage (see table 2.9) based on the actual percentage we were supposed to work on each part of the project, seen in the work breakdown structure (figure 2.1). The top row in the table indicates the initials of the group members. C is Christian Berg Skjetne, K is Kristian Greve Hagen, J.Ø. is John Arne Øye, O is Oscar Aarseth, J.R. is Jaroslav Rakhmatoullin and B is Maria Belen Gallego Garcia.

Table 2.2: Work distribution percentage

		C	**K**	**J.Ø.**	**O**	**J.R.**	**B**
Total Percentage		29,9	29,7	16,9	6,0	16,5	1,1
Coding Percentage	45	15,8	15,4	7,1	1,0	5,8	0,0
Graphical design	6	0,5	3,0	2,0	0,0	0,5	0,0
PoI	3	0,5	0,3	0,0	0,0	2,3	0,0
Tour	2	0,3	0,1	0,1	0,0	1,5	0,0
Map	5	3,0	0,0	0,0	1,0	1,0	0,0
Calendar	3	3,0	0,0	0,0	0,0	0,0	0,0
PoI new/edit	5	0,0	3,0	2,0	0,0	0,0	0,0
Tour new/edit	4	0,0	3,0	1,0	0,0	0,0	0,0
Plan	7	3,0	3,0	1,0	0,0	0,0	0,0
Database	6	3,5	1,0	1,0	0,0	0,5	0,0
Sharing	4	2,0	2,0	0,0	0,0	0,0	0,0
Maintenance Percentage	6	0,5	0,5	2,5	1,5	1,0	0,0
Testing Percentage	5	2,0	2,0	1,0	0,0	0,0	0,0
Research Percentage	1,5	0,0	0,0	0,5	0,0	1,0	0,0
Chapters Percentage	35	9,9	10,1	4,5	2,6	7,4	0,5
Introduction	2	0,0	0,0	0,5	0,5	0,5	0,5
Project Management	5	1,0	1,0	1,0	0,5	1,5	0,0
Requirements spec.	6	0,5	2,0	1,0	0,5	2,0	0,0
Sytem Architecture	5,5	4,5	1,0	0,0	0,0	0,0	0,0
Implementation	4	1,0	2,0	0,0	0,0	1,0	0,0
Testing	3	1,0	2,0	0,0	0,0	0,0	0,0
Follow-Up Work	1	0,9	0,1	0,0	0,0	0,0	0,0
Appendices	7	1,0	1,0	2,0	1,1	1,9	0,0
Glossary	1	0,0	1,0	0,0	0,0	0,0	0,0
References	0,5	0,0	0,0	0,0	0,0	0,5	0,0
Meetings Percentage	7,5	1,7	1,7	1,3	0,9	1,3	0,6
Supervisor	1,5	0,3	0,3	0,3	0,2	0,3	0,1
Client	3	0,7	0,7	0,5	0,3	0,5	0,3
Group	3	0,7	0,7	0,5	0,4	0,5	0,2

Chapter 3

Requirements specification

This chapter contains the client's description of the application. The functional requirements describe the various activities a user can expect to perform with the application. Whereas the non-functional requirements define aspects of the application, such as the supported operating system, devices, and design guidelines. We have changed the requirements slightly according to advice from our supervisor in order to eliminate some ambiguity. However, no changes were done without the client's consent and the lists in this chapter remain close to the original ones. A listing of the modifications done on the requirements can be found in the requirements history section (3.5) at the end of this chapter.

We have chosen not to modify them significantly, because it makes the acceptance tests easier to conduct and because it makes the reporting of the progress to the client more precise. That is, when we reported to the client which functionality has been implemented, we used these requirements.

3.1 Functional requirements

Table 3.1 lists "basis city exploration" requirements. They describe the most important functionality according to the client and have the highest priority in the implementation phase. The terms fixed and free tour serve to distinguish between how a tour may be displayed. Fixed tours may be displayed in a calendar view (scheduled) and on a map with a suggested navigation path from one PoI to the next. Free tours are displayed in a map with a path through all PoIs, and without a schedule.

Table 3.1: Basis requirements

Identifier	**Requirement** The City Explorer shall provide support for...	**Priority**
S1.1	displaying information about PoIs	H
S1.2	showing PoIs in a list view	H
S1.21	showing PoIs in a map view	H
S1.3	easily switching between a PoIs list view and map view	H
S1.4	filtering PoIs according to categories (e.g. museum, landmark)	M
S1.5	navigation to a PoI from an arbitrary location (including other PoIs)	H
S1.6	showing a fixed tour in a calendar view	H
S1.61	showing a fixed tour in a map view	H
S1.7	showing a free tour in a list view	L
S1.71	showing a free tour in a map view	L
S1.8	going trough a tour	H

Table 3.2 lists the "tailoring" requirements of the application. They describe the various planning activities a user may perform and how they can modify the data. This table has medium to high priority and it is important that we fulfill these requirements as well.

Table 3.2: Tailoring requirements

Identifier	**Requirement** The City Explorer shall provide support for...	**Priority**
S2.1	marking certain PoIs as favourite	H
S2.2	adding categories to user created PoIs	M
S2.3	creating a new PoI from scratch	M
S2.4	creating a new Poi from an existing PoI	M
S2.6	creating a fixed tour from an existing tour	H
S2.61	creating a fixed tour from scratch	H
S2.7	creating a free tour from an existing tour	L
S2.71	creating a free tour from scratch	L

Table 3.3 represents the "sharing capabilities" of the application. They describe the Internet enabled functionality such as browsing and downloading PoIs and tours stored on a server and sharing PoIs and tours between

users. This functionality is of least importance to the client and should be implemented when the basis and tailoring requirements have been satisfied. In agreement with the customer, we have removed the requirement to share tours for technical reasons. For a further discussion on this, see the follow-up work chapter, section 7.3, "Sharing of Tours".

Table 3.3: Sharing requirements

Identifier	**Requirement** The City Explorer should provide support for...	**Priority**
S3.1	browsing PoIs stored on a server	M
S3.2	downloading PoIs stored on a server	M
S3.3	sharing PoIs beteween users	M
S3.4	browsing tours stored on a server	M
S3.5	downloading tours stored on a server	M

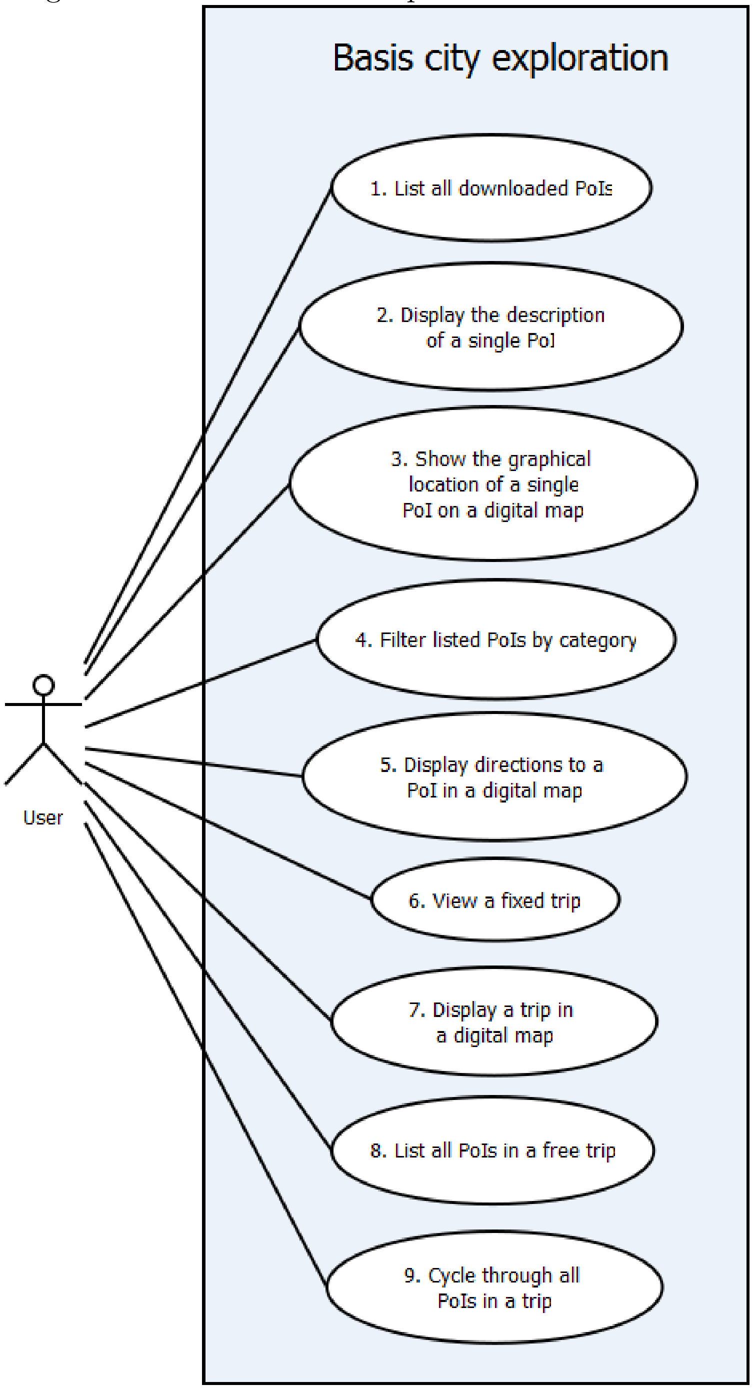
3.2 Use case diagrams

The use cases[14] explain how an end user might proceed to exploit the functional requirements. We use regular use cases for the three main aspects of the application.

3.2.1 Basis city exploration

Figure 3.1 displays an overview of the "Basis city exploration" use cases, they are described more detailed in tables 3.4 through 3.12.

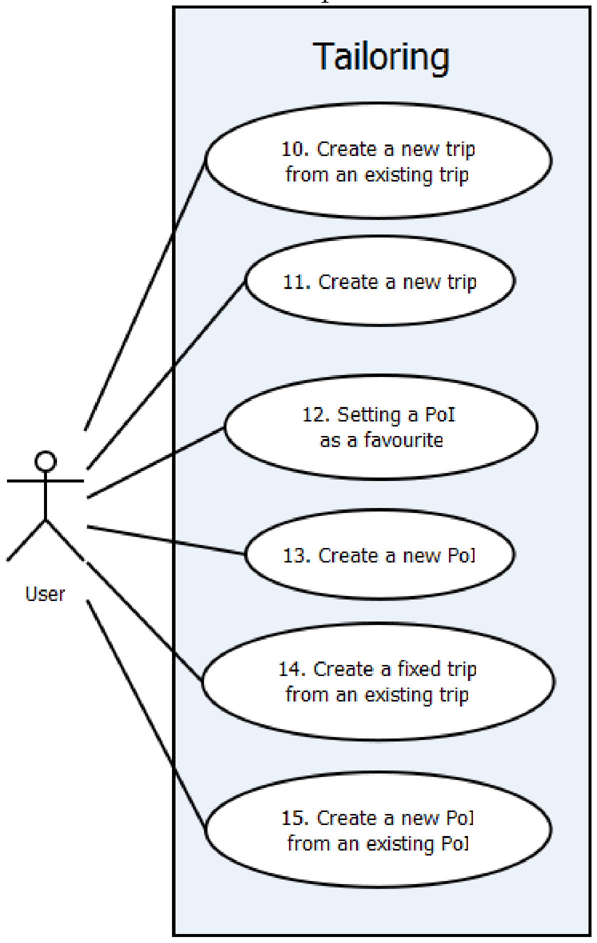
Figure 3.1: Use Cases for requirements S1.1 — S1.8


Basis city exploration
1. List all downloaded PoIs
2. Display the description of a single PoI
3. Show the graphical location of a single PoI on a digital map
4. Filter listed PoIs by category
5. Display directions to a PoI in a digital map
6. View a fixed trip
7. Display a trip in a digital map
8. List all PoIs in a free trip
9. Cycle through all PoIs in a trip
User

3.2.2 Tailoring

Figure 3.2 is an overview of the use cases which define the "Tailoring" part of the application. See tables 3.13 through 3.18 for a closer description.

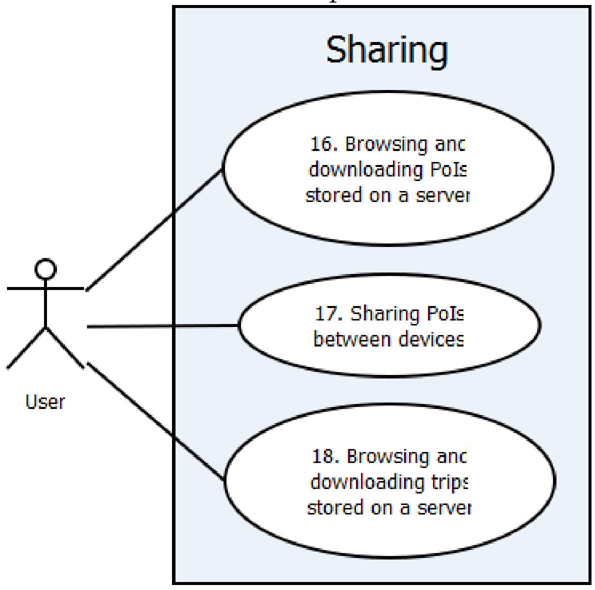
Figure 3.2: Use Cases for requirements S2.1 — S2.71


Tailoring
10. Create a new trip from an existing trip
11. Create a new trip
12. Setting a PoI as a favourite
13. Create a new PoI
14. Create a fixed trip from an existing trip
15. Create a new PoI from an existing PoI
User

3.2.3 Sharing

Figure 3.3 presents an overview of the use cases associated with the "Sharing" aspects of the application. See tables 3.19 through 3.21 for a closer description of these use cases.

Figure 3.3: Use Cases for requirements S3.1 — S3.5


Sharing
16. Browsing and downloading PoIs stored on a server
17. Sharing PoIs between devices
18. Browsing and downloading trips stored on a server
User

3.3 Textual Use Cases

Here is the textual use cases shown, which describe each use case more detailed.

Table 3.4: City Explorer use case 1

Use Case #1:	covers S{1.2, 1.3} 2011-05-04
Name	List all downloaded PoIs
Summary	The user must be able to browse places that have been previously downloaded
Actor	User
Precond.	The local database holds at least one PoI
Postcon.	Activity 2: Plan with a grouped list of all PoIs is visible. Each group is denoted by the category of the PoIs within it.
Basic flow	1. The user clicks [PLAN] in Activity 1: Start 2. The user clicks [TAB:LOCATIONS]

Table 3.5: City Explorer use case 2

Use Case #2:	covers S1.1 2011-05-04
Name	Display the description of a single PoI
Summary	Information such as an address, a picture, a description, the openening hours, a website and a telephone number that is recorded about a particular place is made visible
Actor	User
Precond.	The local database holds at least one PoI, Activity 2: Plan is active
Postcon.	The user is informed about name, description, category and address of a PoI, Activity 4: Tour view is active
Basic flow	1. Click [TAB: LOCATIONS] 2. Click an item in the list

Table 3.6: City Explorer use case 3

Use Case #3:	covers S{1.21, 1.3} 2011-05-04
Name	Show the geographical location of a single PoI on a digital map
Summary	The map activity must be able to show where a particular place is, in order to let users orient themselves and decide on whether they wish to visit the place.
Actor	User
Precond.	Internet connection is established, database holds at least one PoI, [TAB: LOCATIONS] in Activity 2: Plan is activated
Postcon.	The map activity displays a category-icon which indicates where a particular place is.
Basic flow	1. The user selects a PoI by long-clicking the corresponding item in the list 2. A list of quick actions appears 3. The user clicks [QA:SHOW IN MAP] 4. Activity 8: Directions is activated with the map centered on the coordinates of the selected Poi

Table 3.7: City Explorer use case 4

Use Case #4: covers S1.4	2011-05-04
Name	Filter listed PoIs by category
Summary	Enables the user to filter pois by category
Actor	User
Precond.	[TAB: LOCATIONS] in Activity 2: Plan is activated
Postcon.	The list of all PoIs contains only those places which meet the selected categories
Basic flow	1. The user presses the [MENU] button on the device 2. A dialog with check boxes for categories appears 3. User checks the desired categories and clicks [OK]

Table 3.8: City Explorer use case 5

Use Case #5: covers S1.5	2011-05-04
Name	Display directions to a PoI in a digital map
Summary	Launches Activity 9: Calendar View where the source and destination between which the user wishes to navigate can be selected before invoking Google Maps or other application capable of giving directions
Actor	User
Precond.	Either Activity 8: Directions or [TAB:LOCATIONS] in Activity 2: Plan is active
Postcon.	A third party application with navigation support has been launched and is showing the directions
Basic flow	1. Long click the desired PoI 2. Click [QA:GET DIRECTIONS] in quick action popup 3. Activity 9: Calendar View is activated 4. Select ”navigate from current position” 5. The user clicks [NAVIGATE] 6. Select the preferred navigation-application. 7. The directions are drawn on the display.
Alternative flow 4a	4a1. Select ”navigate from another location”. 4a2. Select location from the list of PoIs. 4a3. The user clicks [NAVIGATE] 4a4. Select the preferred navigation-applicatio . 4a5. The directions are drawn on the display.

Table 3.9: City Explorer use case 6

Use Case #6:	covers S1.6 2011-04-28
Name	View a fixed tour
Summary	Displays a tour in a time-table
Actor	User
Precond.	The database holds at least one fixed tour containing at least one PoI, [TAB:TOURS] in Activity 2: Plan is activated
Postcon.	The tour is shown in a time-table
Basic flow	1. Click on a fixed tour.

Table 3.10: City Explorer use case 7

Use Case #7:	covers S{1.61, 1.71} 2011-05-04
Name	Display a tour in a digital map
Summary	A user will want to see all PoIs that are associated with a tour in the map.
Actor	User
Precond.	The database holds at least one tour containing at least one PoI. Either of Activity 3: PoI details or [TAB: TOURS] in Activity 2: Plan is active.
Postcon.	Activity 8: Directions displays all PoIs of a tour in the map.
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:SHOW ON MAP] 3. Activity 8: Directions is activated and displays a category icon on the coordiantes of every PoI in the tour.

Table 3.11: City Explorer use case 8

Use Case #8:	covers S1.7 2011-05-04
Name	List all PoIs in a free tour
Summary	The user must be able to see all PoIs of a free tour in a list
Actor	User
Precond.	The database holds at least one free tour containing at least one PoI, [TAB:TOURS] in Activity 2: Plan is activated
Postcon.	All PoIs associated with a free tour are itemized in the simple list of Activity 3: PoI details
Basic flow	1. Click on a free tour

Table 3.12: City Explorer use case 9

Use Case #9: covers S1.8	2011-05-04
Name	Cycle through all PoIs in a tour
Summary	The user must be able to view all PoIs in a tour one by one, and switching easily between them
Actor	User
Precond.	Activity 4: Tour view or Activity 8: Directions has been activated from Activity 3: PoI details or [TAB: TOURS] in Activity 2: Plan by selecting a single PoI
Postcon.	Two or more PoIs in a tour have been displayed in the order they are stored in the tour.
Basic flow	1. Cycle through the tour by clicking [< –] or [– >].

Table 3.13: City Explorer use case 10

Use Case #10: covers S2.7	2011-05-10
Name	Create a new tour from an existing tour
Summary	The user may create a new free tour from an existing tour
Actor	User
Precond.	[TAB:TOURS] in Activity 2: Plan is activated
Postcon.	A new free tour with the PoIs of an existing tour is created
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:NEW TOUR] 3. Fill in the mandatory fields and choose free tour 5. Click [CHOOSE TOUR] and choose the tour you want to copy PoIs from 6. Click [SAVE TOUR]

Table 3.14: City Explorer use case 11

Use Case #11: covers S{2.61, 2.71}	2011-05-10
Name	Create a new tour
Summary	A user may create a new tour
Actor	User
Precond.	[TAB:TOURS] in Activity 2: Plan is activated
Postcon.	Newly created tour is stored in the database
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:NEW TOUR] 3. Fill in the mandatory fields, and choose whether you want a fixed or a free tour 4. Click [SAVE TOUR]

Table 3.15: City Explorer use case 12

Use Case #12:	covers S2.1 2011-05-10
Name	Setting a PoI as a favourite
Summary	The user may change a PoIs favourite status
Actor	User
Precond.	Activity 4: Tour view is active and displays the selected PoI
Postcon.	The selected PoI has changed favourite status
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:FAVOURITE] marked as a star

Table 3.16: City Explorer use case 13

Use Case #13:	covers S{2.2, 2.3} 2011-05-10
Name	Create a new PoI
Summary	The user may create new PoIs
Actor	User
Precond.	[TAB: LOCATIONS] in Activity 2: Plan is activated
Postcon.	The database contains the newly created PoI
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:NEW LOCATION] 3. Fill in the mandatory fields 4. Click [SAVE LOCATION]

Table 3.17: City Explorer use case 14

Use Case #14: covers S2.6	2011-05-10
Name	Create a fixed tour from an existing tour
Summary	The user may create a fixed tour from an existing tour
Actor	User
Precond.	[TAB:TOURS] in Activity 2: Plan is activated
Postcon.	The tour is displayed in [TAB:TOURS] in Activity 2: Plan
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:NEW TOUR] 3. Fill in the mandatory fields, and choose fixed tour 4. Click [CHOOSE TOUR] 5. Choose the tour you want to create a new from 6. Press the [SAVE TOUR] 7. Click on the newly created tour 8. Press the button [MENU] on the device 9. Click [MENU:TIME TABLE FOR TOUR] 10. Drag your finger to select times and PoIs. 11. Press the button [MENU] on the device 12. Click [MENU:SAVE TIMES]

Table 3.18: City Explorer use case 15

Use Case #15: covers S2.4	2011-05-10
Name	Create a new PoI from an existing PoI
Summary	The user may create new PoIs from existing PoIs
Actor	User
Precond.	[TAB: LOCATIONS] in Activity 2: Plan is activated
Postcon.	The database contains the newly created PoI
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:NEW LOCATION] 3. Press the [CHOOSE LOCATION] 4. Select the PoI you want to create a new from 5. Edit the wanted fields 6. Click [SAVE LOCATION]

Table 3.19: City Explorer use case 16

Use Case #16:	covers S{3.1, 3.2} 2011-05-10
Name	Browsing and downloading PoIs stored on a server
Summary	The user can browse and download PoIs stored on a server
Actor	User
Precond.	Activity 2: Plan is activated and [TAB: LOCATIONS] is selected
Postcon.	The user has browsed and downloaded wanted PoIs from a server
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:UPDATE LOCATIONS] 3. Choose the PoIs you want to download to your device 4. Press the button [MENU] on the device 5. Click [MENU:UPDATE LOCATIONS]

Table 3.20: City Explorer use case 17

Use Case #17:	covers S3.3 2011-05-10
Name	Sharing Pois between devices
Summary	The user can share PoIs with another user
Actor	User
Precond.	Activity 2: Plan is activated and [TAB: LOCATIONS] is selected
Postcon.	A user has sent one or more PoIs to another user
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:SHARE] 3. Choose the PoIs you want to share 4. Press the button [MENU] on the device 5. Click [MENU:SHARE] 6. Select the wanted sharing method and follow the on-screen directions 7. When the other user has received the file that was sent, open it using City Explorer by clicking on the file and choosing City Explorer

Table 3.21: City Explorer use case 18

Use Case #18: covers S{3.4, 3.5}	2011-05-10
Name	Browsing and downloading tours stored on a server
Summary	The user can browse and download tours stored on a server
Actor	User
Precond.	Activity 2: Plan is activated and [TAB: TOURS] is selected
Postcon.	The user has browsed and downloaded wanted tours from a server
Basic flow	1. Press the button [MENU] on the device 2. Click [MENU:UPDATE TOURS] 3. Choose the ss you want to download to your device 4. Press the button [MENU] on the device 5. Click [MENU:UPDATE TOURS]

3.4 Non-functional requirements

Table 3.22 describes the nonfunctional requirements of the application. These requirements impose certain constrains upon the architecture and implementation. Some of the requirements are neither visible to the end user, nor are they easily verifiable with tests. Instead, they need to be thought through carefully to ensure that the application meets them.

Requirement number S4.1 (implemented on Android) is not particularly difficult to verify because the code either compiles against the Android Level 8 API or it does not. The client did not specify a particular reason for choosing this version of the OS. However, a significant portion (roughly 60%) of active devices on the android market in the first quarter of 2011 were running version 2.2 [15]. We also know that Android was preferred because of support for flexible software composition and the open source Apache 2.0 license (page 2, [2]).

Our client has provided two development devices running Android 2.2 that are at our disposal at all times. The two devices are the Google Nexus One and the Samsung Galaxy Tab. The application is developed on both devices and tests are performed on both to ensure that we meet S4.2 (tested on both moblie and tablet Android platforms).

We have met requirement S4.3 (follow Android recommendations for application interface) by adhering to the recommendations mentioned in the Google I/O Video "UI Patterns" [16]. For more details, see the implementation chapter, section 5.2.1.

Perhaps the most difficult non-functional requirement to fulfill is S4.4 (offline functionality). During the planning phase and early in the project we have discussed how we might implement map functionality without an active Internet connection using Openstreetmap.org. Those plans were abandoned due to the complexity of such a solution and the limited amount of time at our disposal. Nevertheless, our application does function offline, but maps and navigation are not available in this specific mode, including browsing and downloading of PoIs and tours stored on a server and other functions that clearly needs an Internet connection to fully function.

To meet requirement S4.5, the application should be able to integrate with the composition tool developed by UbiCompForAll.

We have met S4.6 (reusing existing android building blocks) by including no external libraries in our application except Google API version 8. We have used the integrated Google Maps for displaying our maps, and added the opportunity to open an URL in the default browser.

In agreement with the customer, we have decided to remove the requirement that accommodates new types of PoI entities(Plug-in). For a further discussion on this, see the follow-up work chapter, plug-in system (section 7.6).

Table 3.22: Non-functional requirements

Identifier	**Requirement** The City Explorer client...	**Priority**
S4.1	shall be implemented on the Android platform (TBD Android version 2.2)	H
S4.2	will be tested both on the Android mobile and tablet platforms provided by the customer	H
S4.3	should follow Android recommendations for application interface	M
S4.4	should provide partial functionality when no Internet access is available	H
S4.5	should be extendible to accommodate composition support	H
S4.6	should use existing Android building blocks when possible (e.g. browser and navigation support)	H

3.5 Requirements History

Modifications

Here we will list and explain all edits done on the requirements during the course of the project. By discussing the requirements among ourselves and with our client, alot of changes have been proposed while developing the application.

The following requirements were split into two. This was done because they explained two different aspects of the application in one requirement. Therefore, referencing to one of these requirements then got more specific after they were split.

- requirement S1.2 was split into S1.2 and S1.21
- requirement S1.6 was split into S1.6 and S1.61
- requirement S1.7 was split into S1.7 and S1.71
- requirement S2.6 was split into S2.6 and S2.61
- requirement S2.7 was split into S2.7 and S2.71

The following requirements were dropped because detailed discussion over the requirements proved that they were already covered through earlier requirements.

- requirement S2.2, "Showing favourite PoIs in a list view or in a map view", was dropped because it was already covered in S1.2.
- requirement S2.3', "Easily switching between a favourite PoIs list view and map view", was dropped because it was covered in S1.3.
- requirement S2.4, "Filtering favourite PoIs according to categories" was dropped because it was covered in S1.4.
- requirement S2.5, "Navigation to a favourite PoI from the current location or any other place", was dropped because it was covered in S1.5.
- requirement S2.8, "Showing a fixed trip created by the user in a calendar or in a map" was dropped because it was covered in S1.6 and S1.61.
- requirement S2.9, "Showing a free trip created by the user in a calendar or in a map" was dropped because it was covered in S1.7 and S1.71.
- requirement S2.10, "Going through a trip created by the user", was dropped because it was covered in S1.8.

Additional changes were also made:

- "Adding categories to user created PoIs" was added at S2.2.
- requirement S2.11 was moved to S2.3.
- requirement S2.12 was moved to S2.4.

The non-functional requirement S4.5, "Should be extendible to accommodate composition support", has not been fulfilled. This is because the UbiCompForAll composition tool and back end system does not yet exist. This makes it hard to implement the application with composition support.

Completion

This section contains a history of our progress on completing the requirement specification list. The orange colour means that we have started implementing the requirement. The yellow colour means that the requirement has been partially implemented. The green colour means that the requirement has been fully implemented. The dates in the tables specifies at which date the requirements was started, partially implemented and fully implemented, respectively. Since the requirement specification list was updated each week, some of the requirements were fully implemented during this period. This explains why some status cells does not have a date. As in the functional requirements section (See section 3.1), table 3.23 lists the history for the “Basis city exploration” requirements, table 3.24 lists the history for the “Tailoring” requirements and table 3.25 lists the history for the “Sharing” requirements.

Table 3.23: Basis requirements

Identifier	**Requirement** The City Explorer shall provide support for...	**Status**
S1.1	displaying information about PoIs	🟧 🟨 01. Mar 2011 🟩 07. Mar 2011
S1.2	showing PoIs in a list view	🟧 01. Mar 2011 🟨 🟩 07. Mar 2011
S1.21	showing PoIs in a map view	🟧 01. Mar 2011 🟨 🟩 07. Mar 2011
S1.3	easily switching between a PoIs list view and map view	🟧 🟨 🟩 01. Mar 2011
S1.4	filtering PoIs according to categories (e.g. museum, landmark)	🟧 🟨 01. Mar 2011 🟩 07. Mar 2011
S1.5	navigation to a PoI from an arbitrary location (including other PoIs)	🟧 🟨 07. Mar 2011 🟩 28. Mar 2011
S1.6	showing a fixed tour in a calendar view	🟧 01. Mar 2011 🟨 🟩 09. May 2011
S1.61	showing a fixed tour in a map view	🟧 🟨 🟩 01. Mar 2011
S1.7	showing a free tour in a list view	🟧 01. Mar 2011 🟨 28. Mar 2011 🟩 04. Apr 2011
S1.71	showing a free tour in a map view	🟧 01. Mar 2011 🟨 28. Mar 2011 🟩 04. Apr 2011
S1.8	going trough a tour	🟧 🟨 🟩 28. Mar 2011

Table 3.24: Tailoring requirements

Identifier	**Requirement** The City Explorer shall provide support for...	**Status**
S2.1	marking certain PoIs as favourite	🟧 01. Mar 2011 🟨 🟩 07. Mar 2011
S2.2	adding categories to user created PoIs	🟧 07. Mar 2011 🟨 🟩 02. May 2011
S2.3	creating a new PoI from scratch	🟧 07. Mar 2011 🟨 🟩 02. May 2011
S2.4	creating a new Poi from an existing PoI	🟧 07. Mar 2011 🟨 🟩 02. May 2011
S2.6	creating a fixed tour from an existing tour	🟧 🟨 28. Mar 2011 🟩 09. May 2011
S2.61	creating a fixed tour from scratch	🟧 🟨 28. Mar 2011 🟩 02. May 2011
S2.7	creating a free tour from an existing tour	🟧 🟨 28. Mar 2011 🟩 02. May 2011
S2.71	creating a free tour from scratch	🟧 🟨 28. Mar 2011 🟩 02. May 2011

Table 3.25: Sharing requirements

Identifier	**Requirement** The City Explorer should provide support for...	**Status**
S3.1	browsing PoIs stored on a server	🟧 🟨 🟩 09. May 2011
S3.2	downloading PoIs stored on a server	🟧 🟨 🟩 09. May 2011
S3.3	sharing PoIs beteween users	🟧 🟨 🟩 09. May 2011
S3.4	browsing tours stored on a server	🟧 🟨 🟩 09. May 2011
S3.5	downloading tours stored on a server	🟧 🟨 🟩 09. May 2011

Chapter 4

System Architecture

Decisions regarding the design and implementation of the application are discussed in this chapter.

4.1 Overall Architecture

Figure 4.1: Overall system architecture

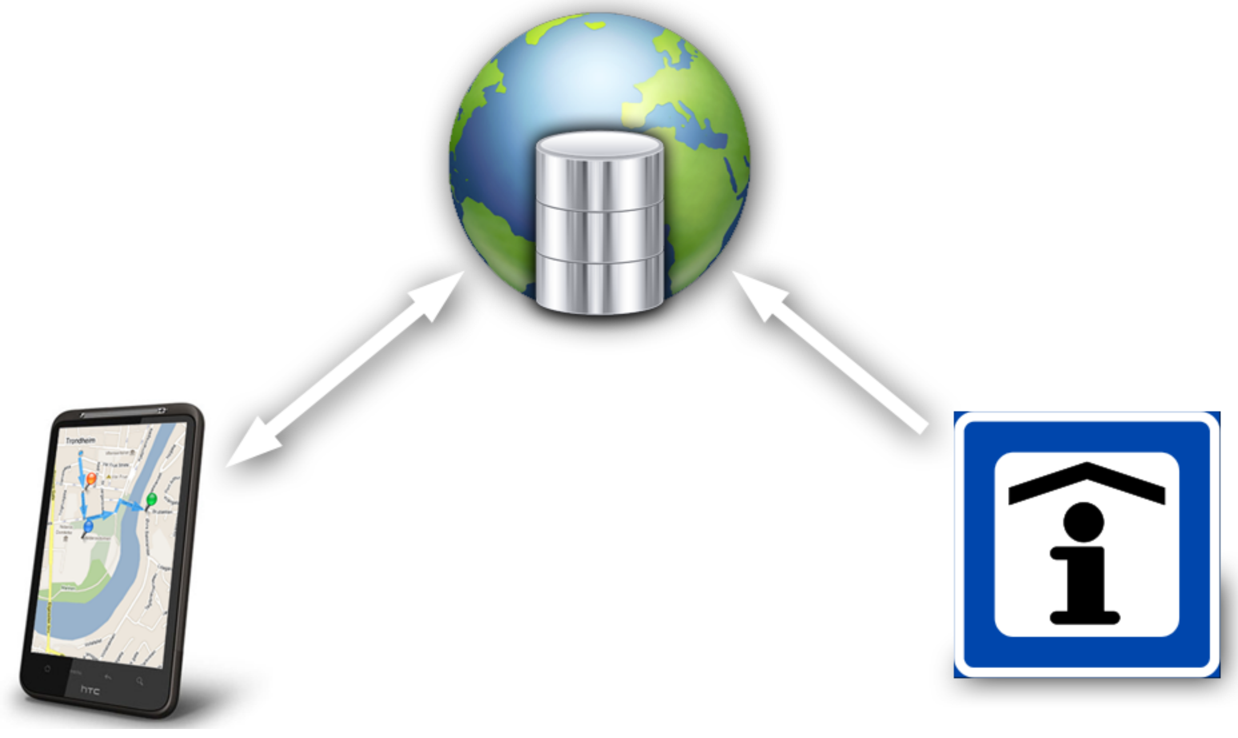


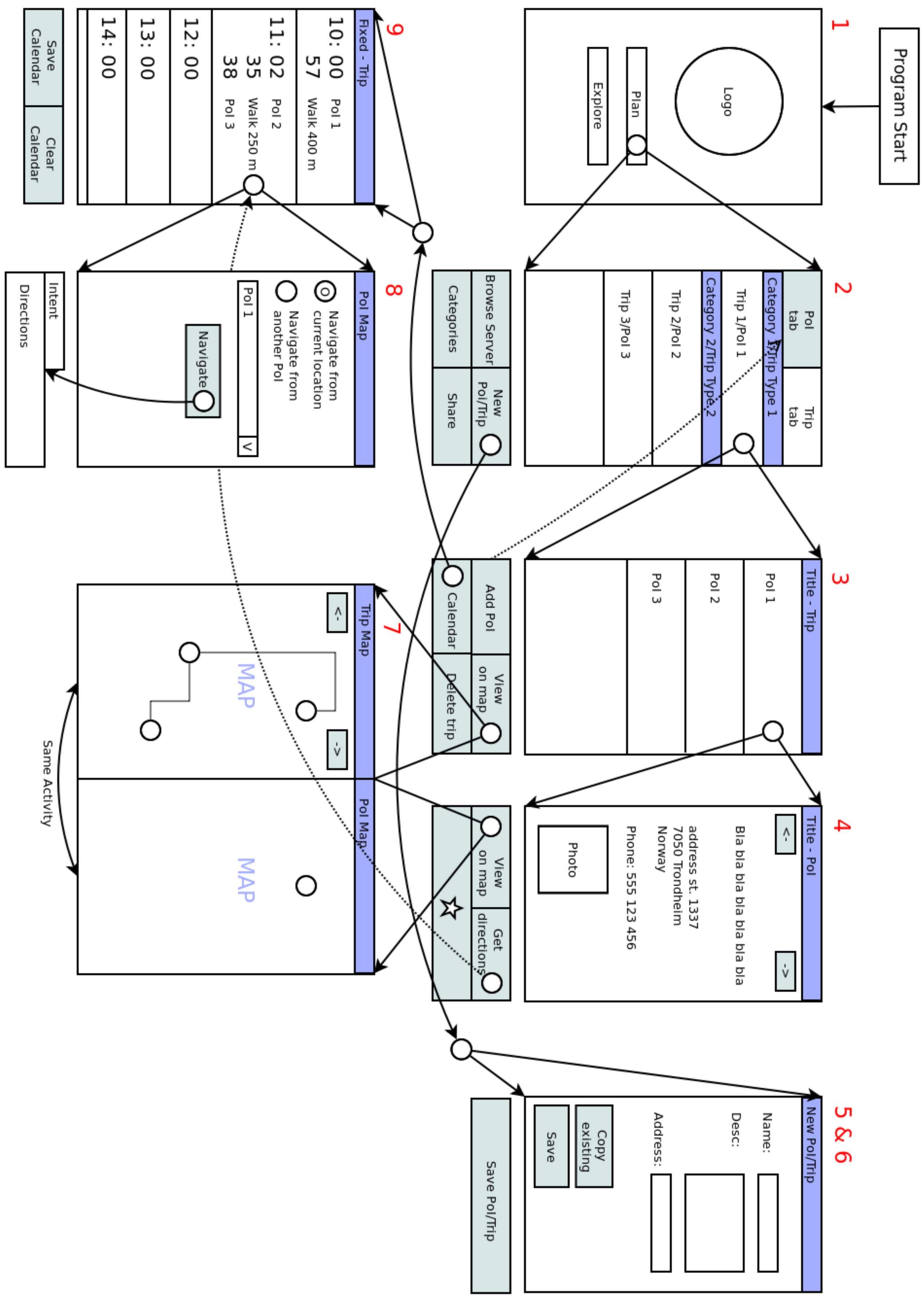
Figure 4.1 shows the overall architecture of the system. The main part of the system is divided in to a back-end server and a client. Only the client

is part of this project. The server is going to be created at a later time. Since the client software does not have a back-end server to test against, a simple server has been created for testing purposes. The server is described in this report for consistency, but it is not part of the specification. The idea is to have a central server hosting PoIs and tours. The PoIs can be added by different institutions or individuals (for example tourist office, school, museum). The client part of the system is able to download this information from the server and display it to the user. The system also supports sharing and creating of information by users of the client.

4.2 Architecture design

The user interface is separated in to the screens shown in the figure 4.2. The figure also shows the navigation flow in the GUI. The GUI is further discussed in section 5.2.1.

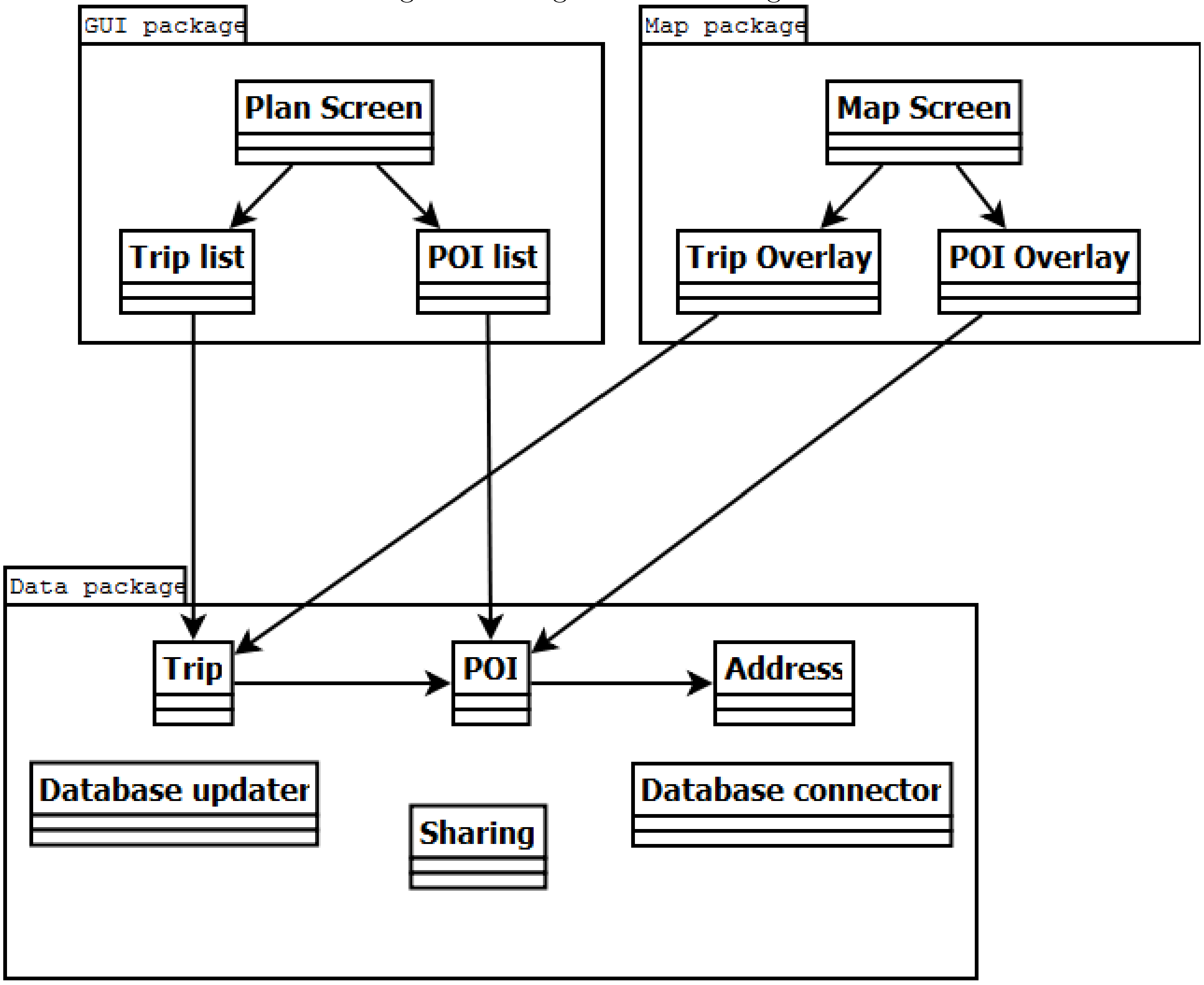
Figure 4.2: User interface navigation flow chart



4.2.1 Architecture classes

The class diagram [14] in figure 4.3 shows a high level description of the class structure of the program. The data package reflects the content of the database (see section 4.2.2). The gui package reflects the components of the GUI (see 5.2.1). The map package contains the classes necessary to show a map with different overlays. This is a high level representation and is only intended to give a simple overview of the relation between the classes in the program.

Figure 4.3: High level class diagram

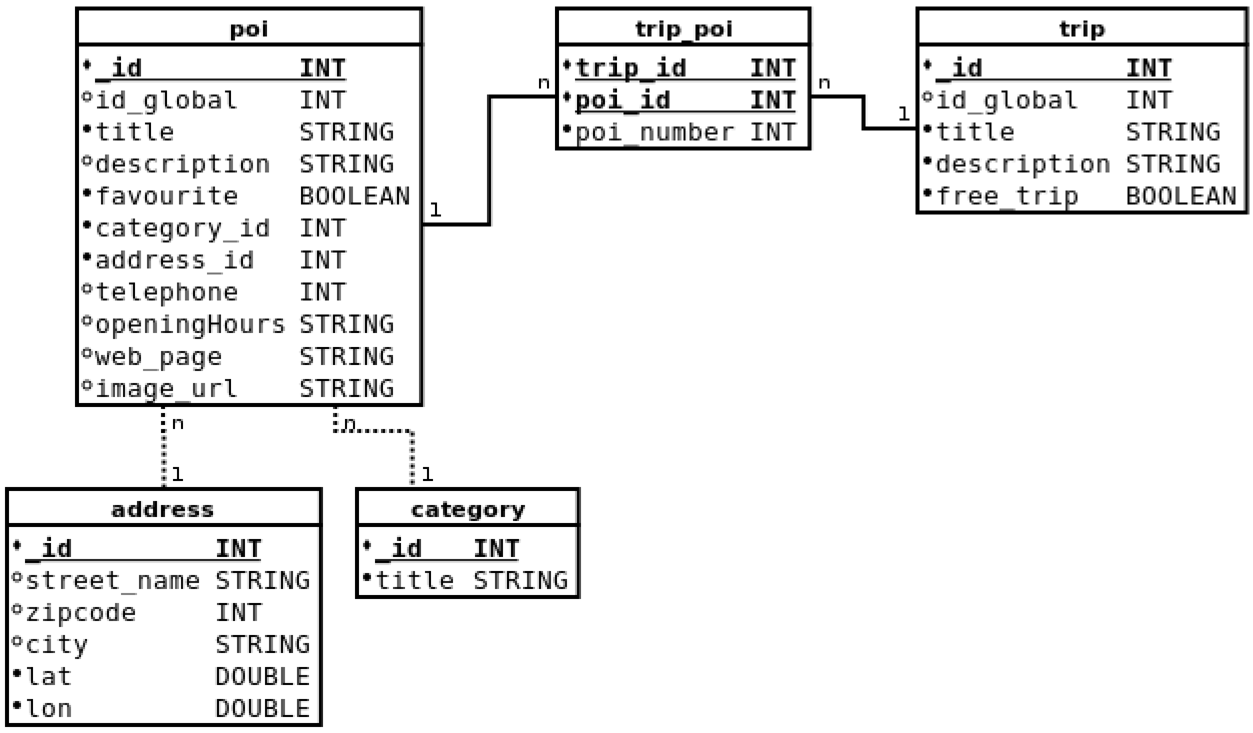


4.2.2 Architecture database

The ER-diagram [17] in figure 4.4 shows an overview of the database structure used in the program. Since the database back end system is not part of the specification, this information is mainly given to help the reader understand the relation between the different data objects used in the program. An online database and a local database is required to allow full functionality when the phone is both connected and disconnected to the Internet. The two databases is similar in design, but the implementation may be different. This will be further discussed in section 5.2.5

The diagram shows the tables and fields of the database, as well as the connections between them. The poi table shows us the fields that a particular poi should have. This is similar with the tour table. The trip_poi table connects the poi and tour tables in a many-to-many connection. This means that a PoI can be included in many tours, and a tour can include many PoIs. The address table is a helper-table that contains an address for a PoI. The category table is a simple table that holds all the categories, and a PoI can be assigned to one category.

Figure 4.4: ER-diagram



4.2.3 Architecture sequence diagrams

The sequence diagram [14] shown in figure 4.5 shows an example of a GUI element changing the favorite status of a PoI. After the update has been sent, the component updates the list of PoIs.

Figure 4.5: Sequence diagram: favorizing a PoI

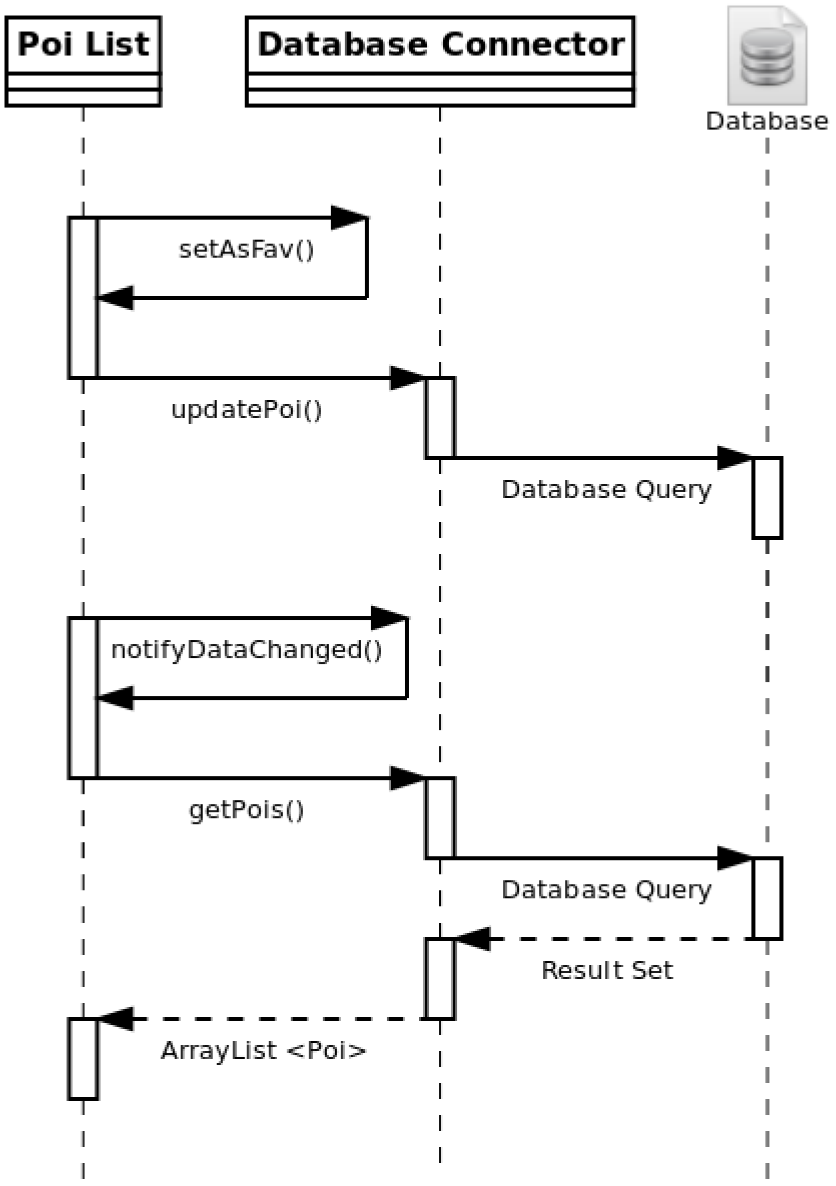
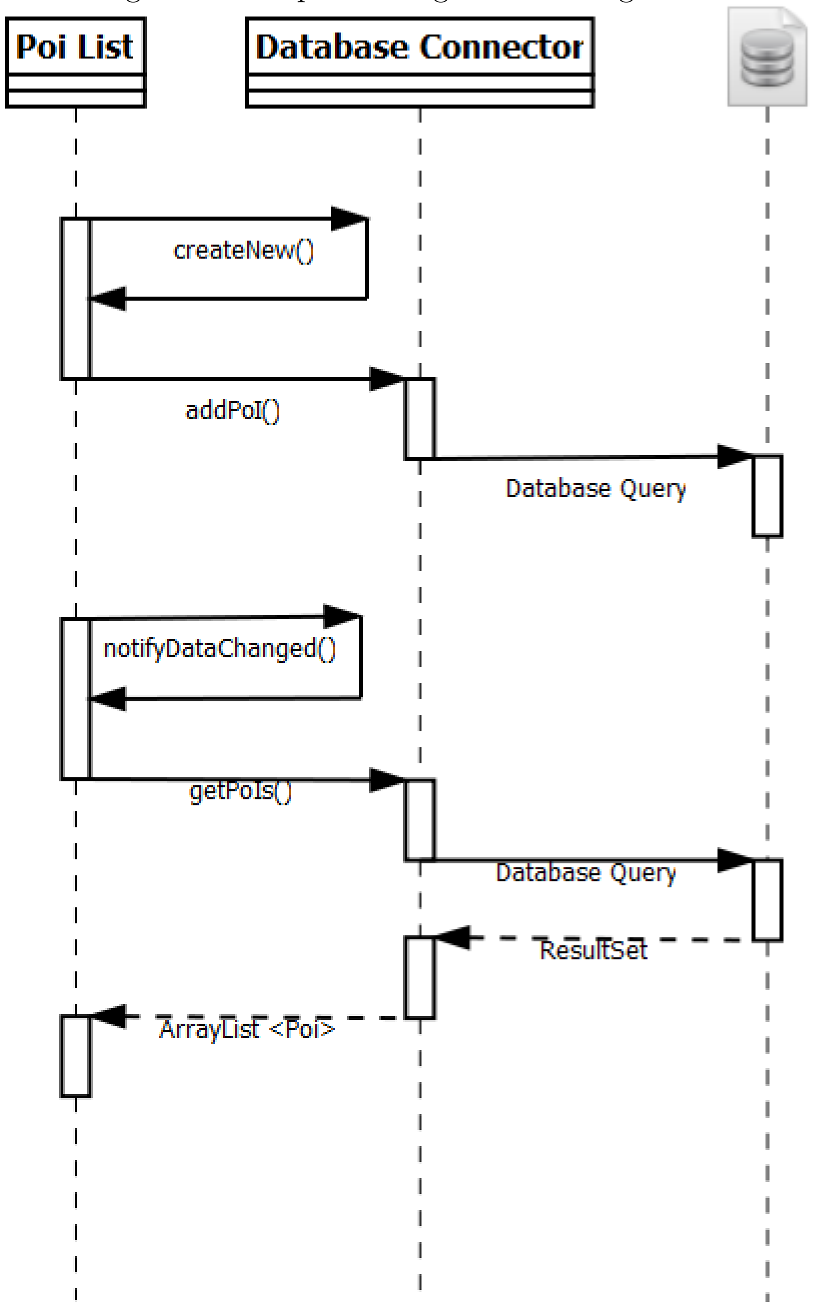


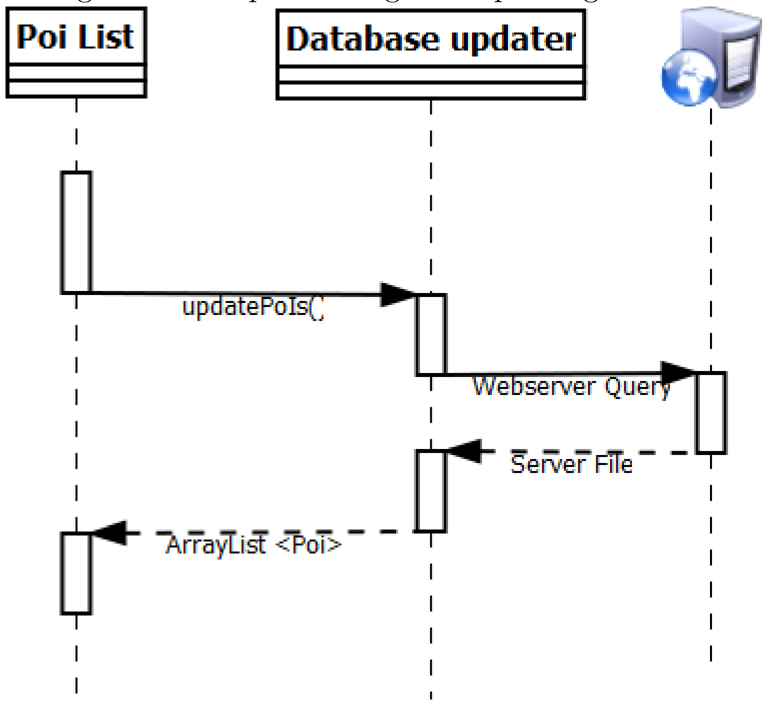
Figure 4.6 show the similarities between changing favorites status of a PoI and creating a new PoI. All other functions that calls the database will be very similar to these two sequence diagrams, so we have only included those to illustrate how the program classes communicate with the database.

Figure 4.6: Sequence diagram: creating a PoI



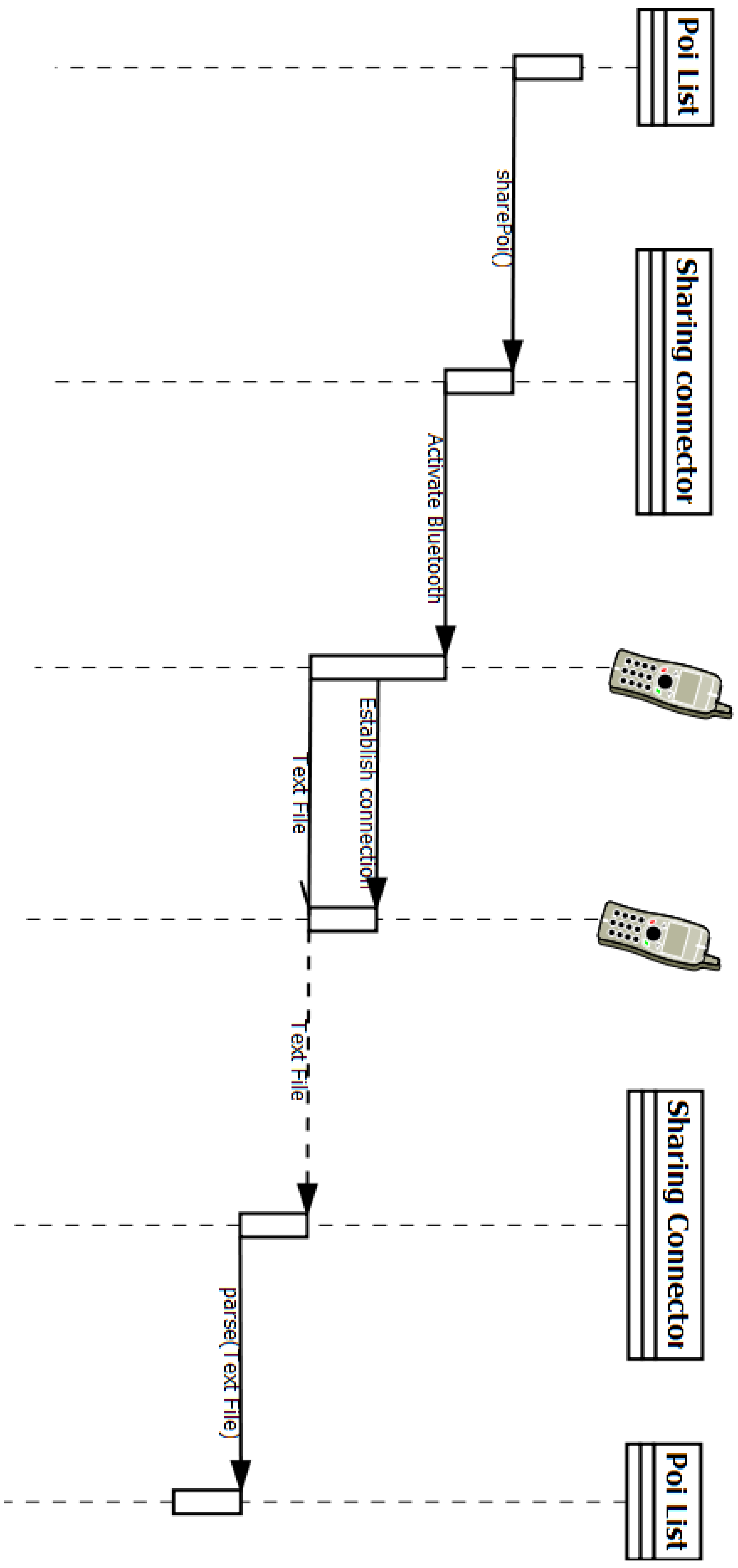
The sequence diagram [14] in figure 4.7 show the process of updating the device's local storage with information contained on a webserver. When the user request his list to be updated, a message is sent to the server, which is sent back, and parsed, before being sent back as a list of PoIs.

Figure 4.7: Sequence diagram: updating PoI List



Viewing figure 4.8, one can see the sequence of classes called when the user prompts to share one or more of his PoIs with another android device through a Bluetooth [18] connection. The PoIs are sent as object between the classes in teh City Explorer, but are parsed in a connector class and sent as text files over the connection between the devices.

Figure 4.8: Sequence diagram: sharing a PoI



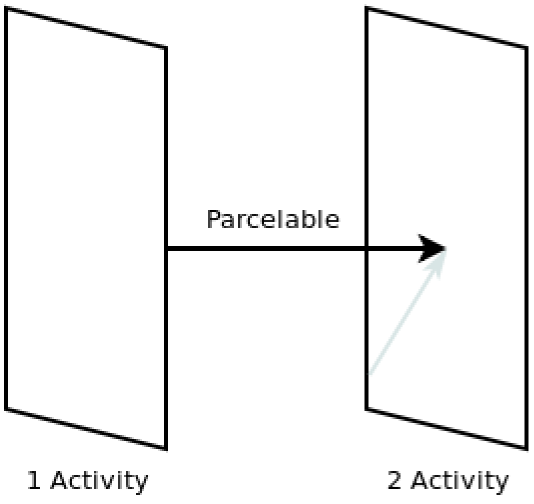
Chapter 5

Implementation

Since this project mostly involves user interface development, the architecture is simple. Also, since the platform running the system is Android, many of the design practices is forced by the platform design.

We have chosen the software implementation based on the recommended practices for Android. Some of the effects of this is a flat layered design, with minimal communication between the different screens in the program. The reason that the Android system uses this approach has to do with the way the Android operating system handles multitasking. In short, the effect is that no persistence should exist between the screens because they can be destroyed and garbage collected at a time chosen by the operating system, i.e. if it is not an active activity. This has the effect that we can not have a core class tying the entire user interface together.

Figure 5.1: Communication between activities



The passing of data between the screens is done by making the objects that we wish to send parcelable to a set of primitive types. When the sender initiates the handover, it first packages the object and sends it via an intent. When the receiving screen (or Activity, as they are called in Android) starts up, it can retrieve the package from the intent that started it, and unpack the package to build a new, identical object. (See figure 5.1)

5.1 Packages

`project.CityExplorer.gui`
This package holds the Activities for the project (see figure 4.2) and helper classes for creating lists and other gui components.

`project.CityExplorer.data`
This package holds the classes for mirroring database content in memory. (For example PoI, tour, address) it also holds the database interface and the database connectors.

`project.CityExplorer.map`
This package holds the classes needed to show the information on a map.

`project.CityExplorer.map.route`
This package holds the classes for showing a route on a map.

5.2 Implementation design

Figure 4.2 shows all of the different Activities of the application. Here is a short description of each activity:

- Activity 1 is the starting screen. This is the first thing a user sees after opening the application. The screen has two buttons ”Plan” and ”Explore”. The plan button launches activity 2. The explore button launches activity 7, showing a map with the users current position and nearby POIs.
- Activity 2 is a tab view, showing either PoIs or tours. The PoIs are listed by categories, always showing the favourites first. The tours are listed in categories ”‘empty”’, ”‘fixed”’ and ”‘free”’.

- Activity 3 opens when a PoI is selected. The activity shows a detailed view with all of the information about the PoI.
- Activity 4 opens when a tour is selected. The activity shows a view of all of the PoIs in a tour.
- Activity 5 allows creating a PoI.
- Activity 6 allows creating a tour.
- Activity 7 shows a map. The activity can be used to show a map with either a tour or PoI.
- Activity 8 opens the directions activity, allowing you to get directions from one PoI to another.
- Activity 9 opens the calendar view, allowing you to see or change times for your fixed tour.

5.2.1 User interface

The GUI is divided in nine main `Activity` elements. They are the centerpieces of the application because all functionality is exposed through them. While designing and implementing the GUI, we have followed some of the recommendations mentioned in the Google I/O video [16]. We found elements (such as "Quick Actions") especially useful and they are described more closely later in this sub chapter.

5.2.2 Quick Actions

Parallel functionality to a "context popup" which appears whenever a user right-clicks on an object in a GUI of a regular computer application is offered through a list of quick actions. A quick action can be shown after long-clicking an object in the interface. There is no clear way of knowing whether an object will produce a list of quick actions short of trying to long-click it. However, items in the lists of Activity 2 as well as PoI icons in the map, will have quick actions.

A list of quick actions may be longer than the width of the display. The user can scroll the list horizontally in those cases by holding one finger on the list and dragging it to the left.

5.2.3 Activities

An overview of the elements discussed in this section, is available in figure 4.2. Following each Activity, there are listings of items associated with it such as visible buttons, quick actions, source file classes and menu buttons. Buttons referred to in the use case section (3.2) are defined in those lists.

Activity 1: Start

This is the first visible activity after launching City Explorer. Button [PLAN] launches the tab view showing either PoIs or tours (Activity 2). Button [EXPLORE] launches the map (Activity 7) showing the current position of the device and nearby PoIs.

Involved classes:	**Buttons:**
`StartActivity.java`	[PLAN] —> Activity 2: Plan
	[EXPLORE] —> Activity 7: Map

Activity 2: Plan

Plan consists of two tabs named “Locations” and “Tours”, which list all currently downloaded PoIs and tours, respectively. The list of PoIs is divided into several sections, where each one is presented with a header line that is the name of the category to which the PoIs in the section are assigned. That is, PoIs are grouped in sections labeled by the primary category of the PoIs within. The list of tours is also divided into sections, ”‘empty”’, ”‘fixed”’ and ”‘free”’. The tours are placed under each section according to what kind of tour it is.

Involved classes:	**Buttons:**
`PlanActivity.java`	[LOCATIONS] —> PoIs listing
`PlanTabActivity.java`	[TOURS] —> tours listing
`PlanTabPoi.java`	[ITEM:POIS LIST] —> Activity 3: Poi Details
`PlanTabTrip.java`	[ITEM:TOURS LIST] —> Activity 4: Tour view
`SeparatedListAdapter.java`	
`Section.java`	
`PoiAdapter.java`	
`Poi.java`	
`TripAdapter.java`	
`Trip.java`	

Quick Actions:

tours tab

[QA:ADD LOCATIONS] —> Locations in Activity 2: Plan

[QA:SHOW ON MAP] —> Activity 7: Map

[QA:DELETE] Delete selected tour

Locations tab

[QA:STAR] Toggle favourite status of PoI

[QA:ADD TO TOUR] —> Tours in Activity 2: Plan

[QA:SHOW ON MAP] —> Activity 7: Map

[QA:GET DIRECTIONS] —> Activity 8: Directions

[QA:SHARE] Share a PoI with another person

[QA:DELETE] Delete selected PoI

Menu buttons in Locations tab:

[MENU:NEW LOCATION] —> Activity 5: Create PoI

[MENU:SHARE] —> Locations in Activity 2: Plan

[MENU:FILTER] —> Filter PoIs according to categories

[MENU:UPDATE LOCATIONS] —> Locations in Activity 2: Plan

Menu buttons in Tours tab:

[MENU:NEW TOUR] —> Activity 6: Create tour

[MENU:UPDATE TOURS] —> Tours in Activit 2: Plan

Activity 3: PoI details

PoI details displays information about a selected PoI. A user can activate this Activity in at least three ways:
1. by clicking an item in the list of the tab “Locations” in Activity 2
2. by clicking an item in the list of Activity 4: Tour view
3. by clicking a PoI in Activity 7: Map view

The buttons [< –] and [– >] appear only when the user navigates to this activity from Activity 4: Tour view.

Involved classes:	**Buttons:**
`PoiDetailsActivity.java`	[< –] show previous PoI
`Poi.java`	[– >] show next PoI

Menu buttons:

[menu:Show on Map] —> Activity 7: Map

[menu:Get directions] —> Activity 8: Directions

[menu:star] Toggle favourite status of PoI

Activity 4: Tour view

Activity 4 opens when a tour is selected. The activity shows all of the PoIs in a tour in a list view.

Involved classes:

TripList.java

Trip.java

Buttons:

[item:pois list] —> Activity 3: Poi Details

Quick Actions:

[qa:Delete from tour] —> Delete PoI from the current tour

[qa:Show on Map] —> Activity 7: Map

[qa:Get Directions] —> Activity 8: Get directions

Menu buttons:

[menu:Add location] —> Locations tab in Activity 2: Plan

[menu:Show on Map] —> Activity 7: Map

[menu:Delete] —> Delete the current tour

[menu:Time Table for Tour] —> Activity 9: Calendar view

Activity 5: create PoI

Activity 5 allows you to create a new PoI.

Involved classes:

NewPoiActivity.java

Poi.java

Buttons:

[category drop down] Choose category

[search] Opens the browser and searching for images with your given name with Google Images

[Choose Location] —> Locations in Activity 2: Plan

[Save Location] Save the PoI in the database

Menu buttons:

[MENU:SAVE LOCATION] Save the PoI in the database

Activity 6: Create Tour

In activity 11, the user can make a new tour. The user can set the name and whether or not it will be a fixed or a free tour. It will start by being an empty tour, so you can add PoIs later on, or you can choose to derive the PoIs from an existing tour.

Involved classes:	**Buttons:**
`NewTripActivity.java`	[CHOOSE TOUR] —> Tours in Activity 2: Plan
`Trip.java`	[SAVE TOUR] Save the tour in the database

Menu buttons:

[MENU:SAVE TOUR] Save the tour in the database

Activity 7: Map view

Activity 7 shows a map. The activity can be used to show either a PoI or a tour in a map, using Google Maps. The buttons [< –] and [– >] appear only when the user navigates to this activity from Activity 4: Tour view.

Involved classes:	**Buttons:**
`MapsActivity.java`	[< –] show previous PoI
`MapIconOverlay.java`	[– >] show next PoI
`MapTripOverlay.java`	[A POI] opens up the Quick Action for the selected PoI.
`PoiToPoi.java`	

Quick Actions:

[QA:DETAILS] —> Activity 3: PoI Details

[QA:GET DIRECTIONS] —> Activity 8: Directions

Activity 8: Directions

The activity where the user can select locations between which they wish to see navigation help.

Involved classes:
`NavigateFrom.java`
`Poi.java`

Buttons:
[PoI DROP DOWN] Choose PoI
[NAVIGATE] Opens up Google Maps, which gives you navigation support

Activity 9: Calendar View

The Calendar View is a view that lets you add PoIs in a tour in a time-specific order. It also displays the amount of time the user has to spend walking between each particular PoI in the tour.

Involved classes:
`CalendarActivity.java`
`Trip.java`
`Time.java`

Menu buttons:
[MENU:SAVE TIMES] Save the created times in the database
[MENU:CLEAR TIMES] Clear the times in the view

5.2.4 Implementation Classes

The class diagrams[14] shown in figure 5.2, 5.3, 5.4, 5.5 and 5.6 shows the implementation of the diagrams in the architecture chapter (see section 4.2.1). Figure 5.2 shows the classes involved with the creation of PoIs and Tours, figure 5.3 shows the classes involved in showing tours and PoIs in a list. The classes in figure 5.4 handles the calendar view, whilst figure 5.5 shows the classes for viewing tours and PoIs in a map and finally. The classes in figure 5.6 handles sharing and the downloading of new PoIs and tours.

Figure 5.2: Class Diagram 1

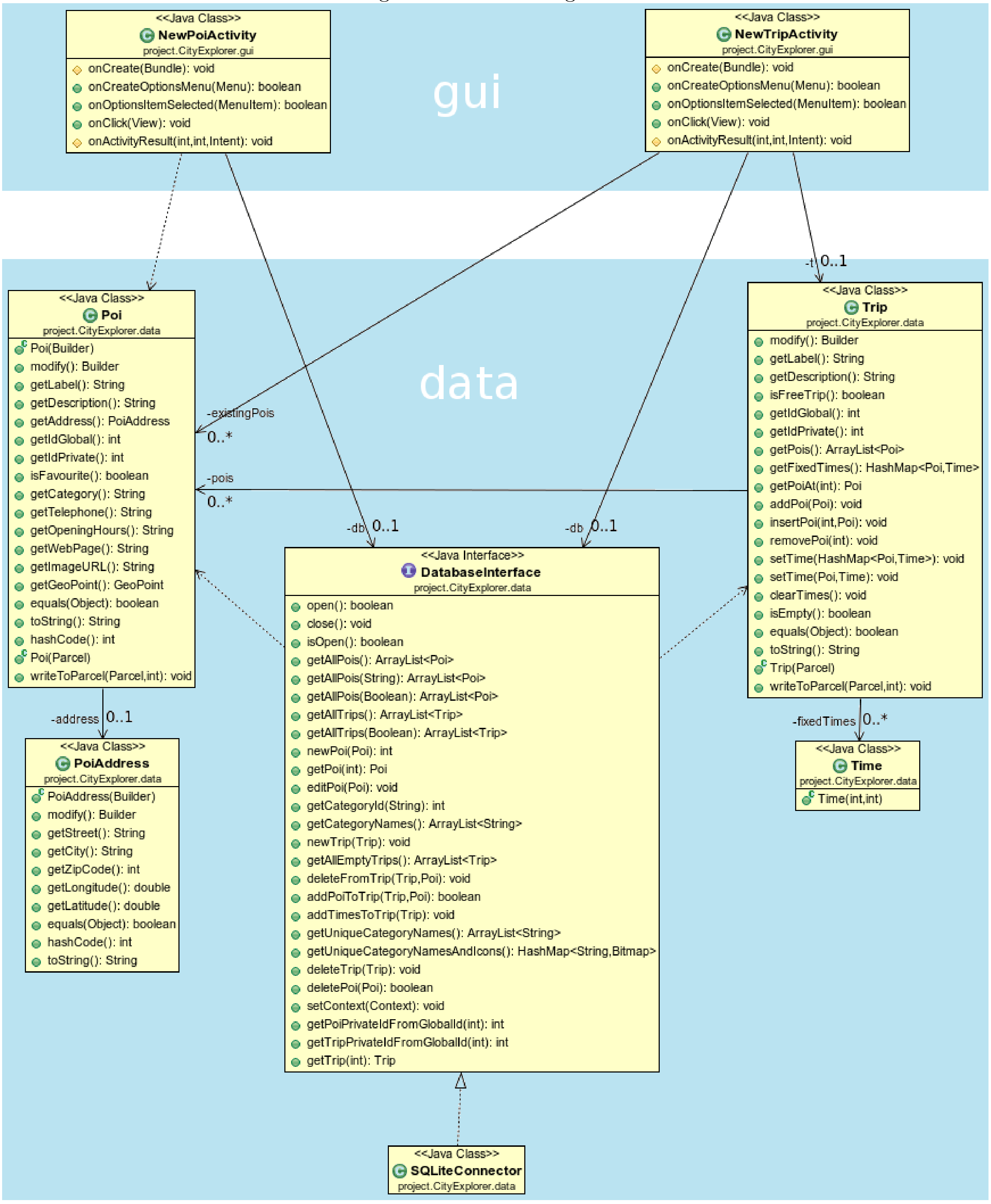


Figure 5.3: Class Diagram 2

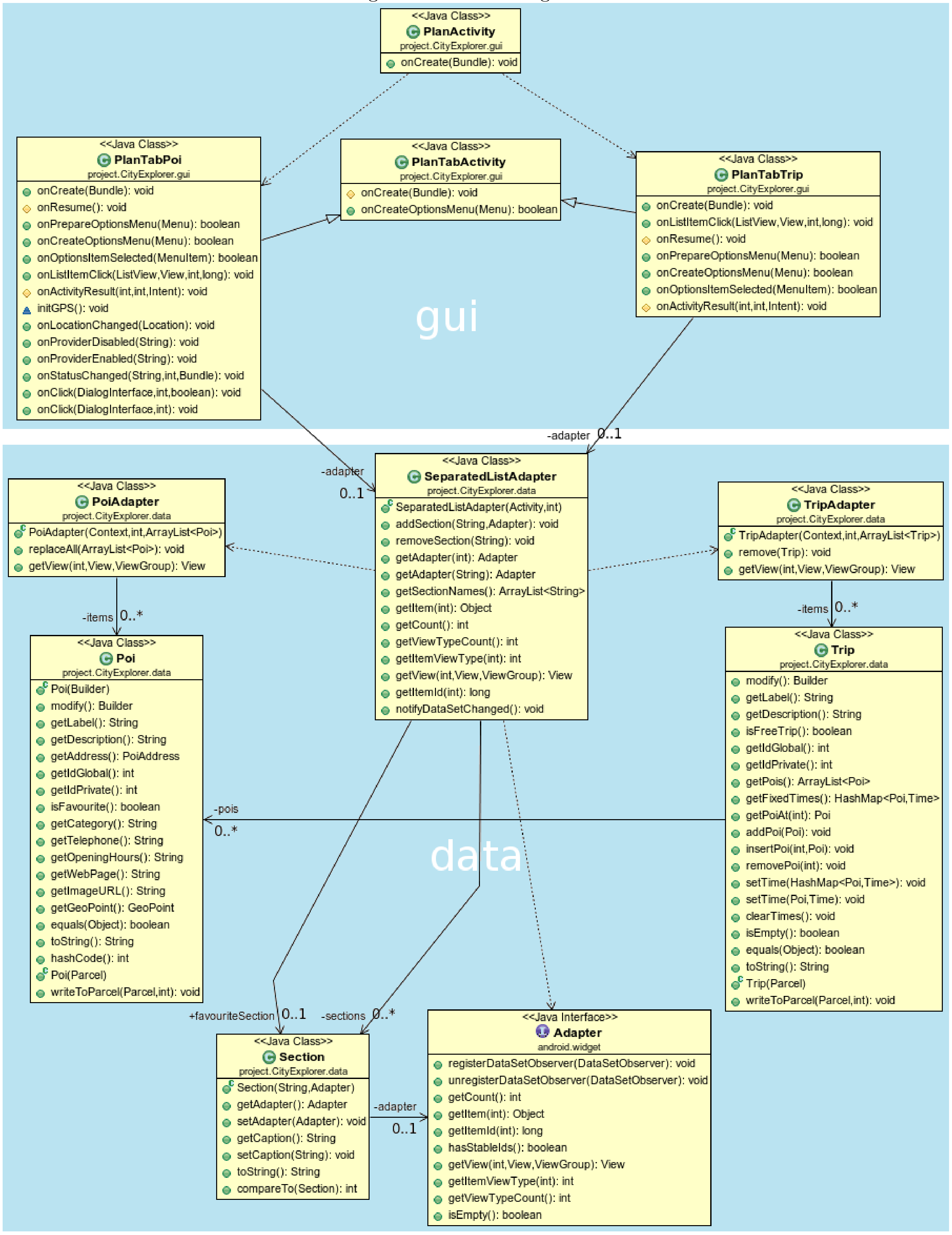


Figure 5.4: Class Diagram 3

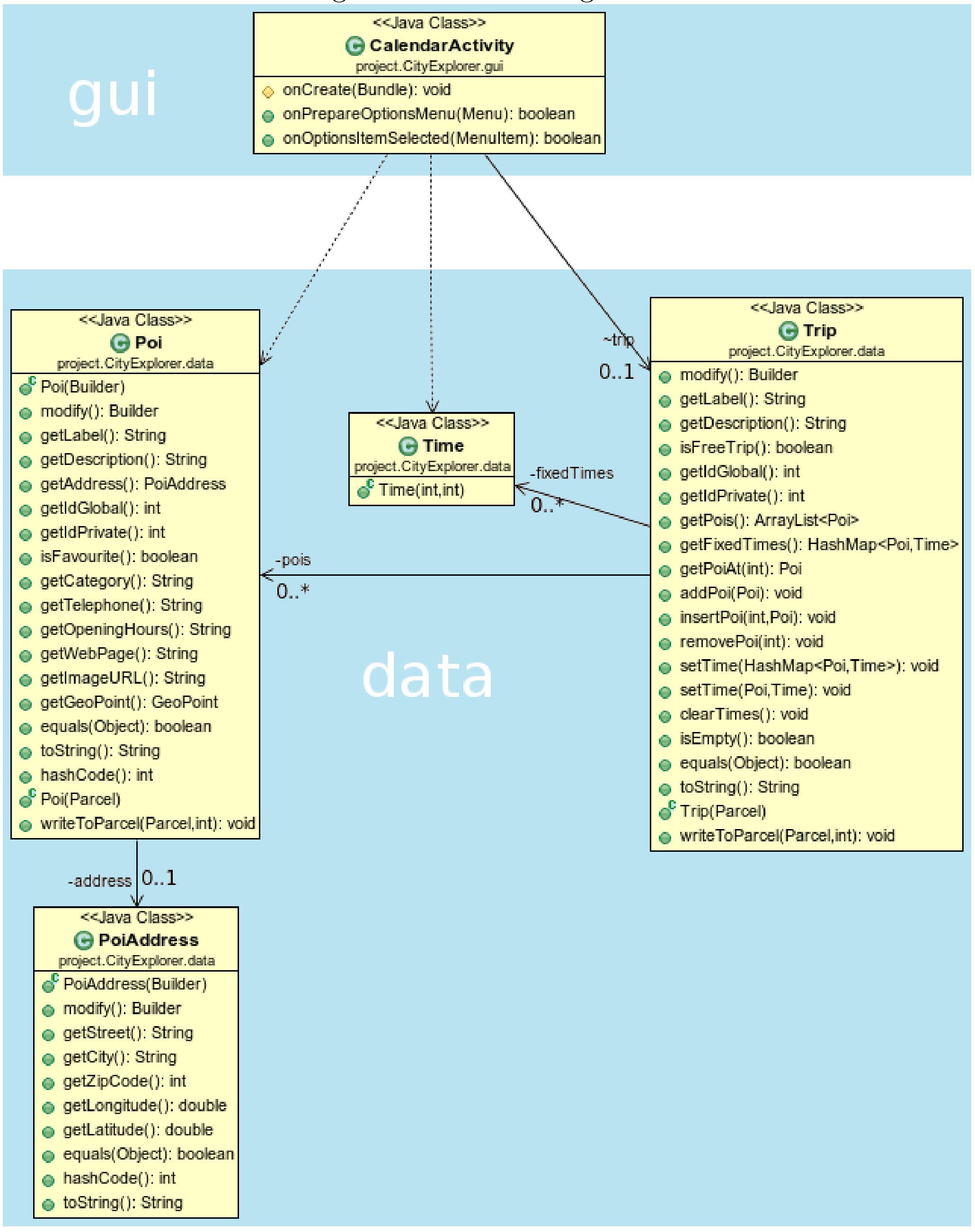


Figure 5.5: Class Diagram 4

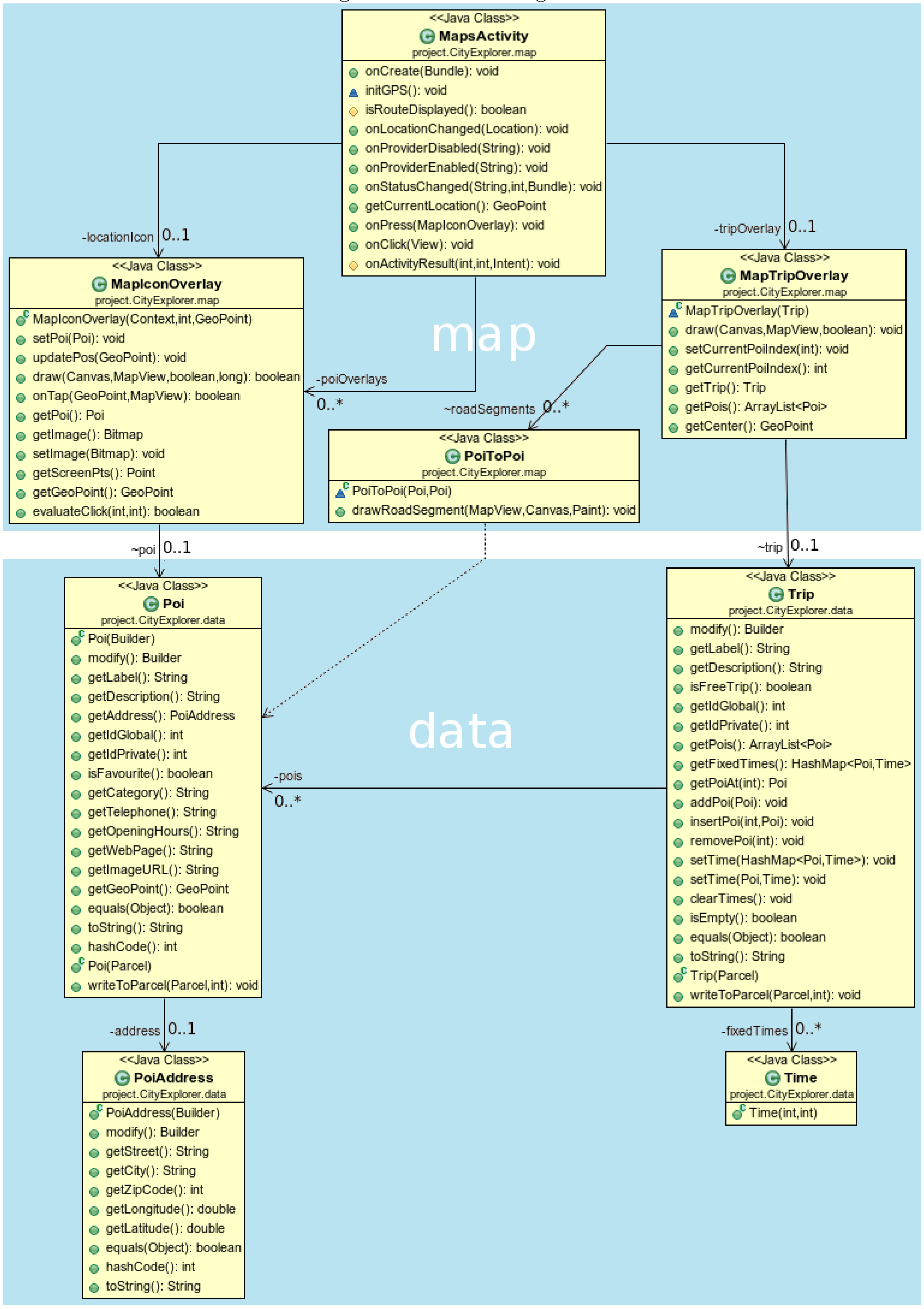
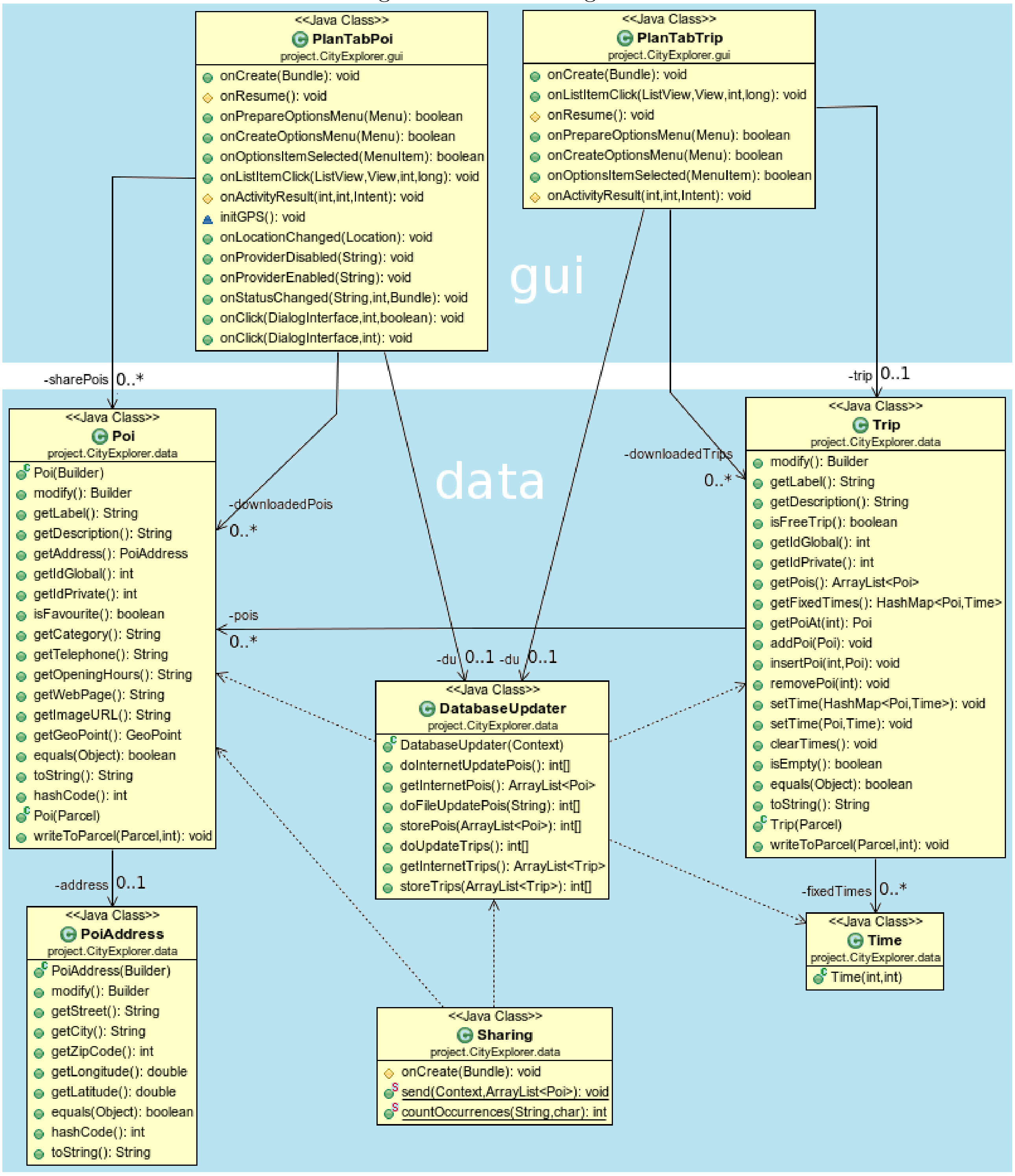


Figure 5.6: Class Diagram 5



5.2.5 Implementation Database

The actual implementation of the databases described in section 4.2.2 is realised as two different databases. The online database is realised using a simple Http call using a Php web server. The local server is realised by a SQlite server on the device. The reason for choosing this solution is because the Android operation system has built in support for SQlite. The web server is chosen because of the unstable connection nature of a mobile phone. A database system without a persistent connection is recommended.

5.3 Implemented default Android components

As stated in the non-functional requirement specification (see section 3.4), the use of default Android building blocks is encouraged. In the layout design, we have chosen to use almost exclusively default Android components. The only places the graphics of the Android look-and-feel has been changed are on the start up screen buttons (see section 5.2.3), and on the tabs in the plan activity (see section 5.2.3). Other default components we have used is the intent-filtering system. We have implemented this to give the user the ability to get directions to and from a PoI, and to send and receive PoIs between users. Other default components are the implemtation of the SQLite[10] database and the Google Maps components (see section 5.2.3) from the Google API. The quickaction bar is a gui component implemented after suggestion on the Google IO conferance[16]. The calendar view (see section 5.2.3) was inspired and partly built on the AnCal application[19].

Chapter 6

Testing

An important part of the agile development model we have used is iterative testing. Therefore, we have used the weekly meetings with the customer to test the different aspects of the application. Because of this, we have decided to conclude it all with a customer test, using all the use cases. Since the use cases cover all of the functional requirements, a test of the use cases will ensure that we fulfill all the functional requirements specified by the customer. The scenarios section 6.2 is a collection of three scenarios that will fully capture all of the requirements.

6.1 Client testing

Over the course of the project, we have demonstrated the application weekly for our client. This gave us the opportunity to fix minor flaws and change behaviour according to the client's wishes. By doing this, we were confident that all the requirements were fulfilled, and the final client test could be completed successfully. The application was delivered to the client on 11th of April for testing, and the final client testing was performed on 2nd of May and 9th of May.

6.2 Scenarios

Each of the three scenarios in this section will include multiple actions the user can do.

Scenario 1 The user wants to make a tour from scratch.

Start by creating a new empty and free tour called My Tour. Filter the list of PoIs by three categories, look through the PoIs in the three categories and choose one PoI from each category to the newly created tour. Now you should have a tour consisting of three PoIs. You would like to see that this is true, so you will go to your tour, and check this. To make sure you can use this for something useful, view your tour on the map. Loop through the PoIs so you can clearly see that you have exactly those three you wanted.

Scenario 2 The user wants to plan and take a tour.

Start by selecting the fixed tour you would like to take through the city. The screen with the list of all the PoIs contained in the tour will bve shown. Review them to verify that this is the correct tour. Open the menu and go into the timetable for this tour. On the timetable are some already set times for the different PoIs, however, you would like to set new times. To do so you open the menu and clear the time schedule for the tour. You then set new times for each of the PoIs in the tour. Save the time schedule. Now, the time has come to take the tour, so then you open its time schedule. You press the heading of the first PoI you are supposed to visit; its information and address appear. You get directions to the PoI and visit it. When the time has come for you to leave and visit the next PoI you repeat this procedure, until you have gone through your whole tour.

Scenario 3 The user wants to create a PoI, and download another from a server, then share them both with his friend via Bluetooth.

You find yourself sitting at a beautiful cafe, wanting to share this PoI with other people. Start off by opening the PoI creation menu via the menu in the “Locations” list. Fill in the correct information in the fields and select “Cafe” in the “Category” field. Save the PoI. Verify that your new PoI has appeared in the “Locations” list. Then open the update list through the menu to download the other PoI you are goind to share. Select the PoI from the list of available PoIs on the server and select to update them. This PoI should now also appear in your “Locations” list. To share these two newly added PoIs with your friend, you open the sharing list from the menu in the “Locations” list. Select the two newly added PoIs and start sharing them. Doing so will prompt a menu where you can select different means of sending the information. Select the e-mail function, fill in your friends e-mail address and send it. Your friend should now receive the PoIs on his Android device.

6.3 Test Cases

These are the test cases for our application. They are based on the use cases in the requirements specification chapter (see section 3.3).

Table 6.1: City Explorer test case 1

Test Case #1:	covers S{1.2, 1.3} 2011-05-04
Name	List all downloaded PoIs
Summary	The user must be able to browse places that have been previously downloaded
Status	Passed

Table 6.2: City Explorer test case 2

Test Case #2:	covers S1.1 2011-05-04
Name	Display the description of a single PoI
Summary	Information such as an address, a picture, a description, the openening hours, a website and a telephone number that is recorded about a particular place is made visible
Status	Passed

Table 6.3: City Explorer test case 3

Test Case #3:	covers S{1.21, 1.3} 2011-05-04
Name	Show the geographical location of a single PoI on a digital map
Summary	The map activity must be able to show where a particular place is, in order to let users orient themselves and decide on whether they wish to visit the place.
Status	Passed

Table 6.4: City Explorer test case 4

Test Case #4:	covers S1.4 2011-05-04
Name	Filter listed PoIs by category
Summary	Enables the user to filter pois by category
Status	Passed

Table 6.5: City Explorer test case 5

Test Case #5:	covers S1.5	2011-05-04
Name	Display directions to a PoI in a digital map	
Summary	Launches Activity 9: Calendar View where the source and destination between which the user wishes to navigate can be selected before invoking Google Maps or other application capable of giving directions	
Status	Passed	

Table 6.6: City Explorer test case 6

Test Case #6:	covers S1.6	2011-04-28
Name	View a fixed tour	
Summary	Displays a tour in a time-table	
Status	Passed	

Table 6.7: City Explorer test case 7

Test Case #7:	covers S{1.61, 1.71}	2011-05-04
Name	Display a tour in a digital map	
Summary	A user will want to see all PoIs that are associated with a tour in the map.	
Status	Passed	

Table 6.8: City Explorer test case 8

Test Case #8:	covers S1.7	2011-05-04
Name	List all PoIs in a free tour	
Summary	The user must be able to see all PoIs of a free tour in a list	
Status	Passed	

Table 6.9: City Explorer test case 9

Test Case #9: covers S1.8	2011-05-04
Name	Cycle through all PoIs in a tour
Summary	The user must be able to view all PoIs in a tour one by one, and switching easily between them
Status	Passed

Table 6.10: City Explorer test case 10

Test Case #10: covers S2.7	2011-05-10
Name	Create a new tour from an existing tour
Summary	The user may create a new free tour from an existing tour
Status	Passed

Table 6.11: City Explorer test case 11

Test Case #11: covers S{2.61, 2.71}	2011-05-10
Name	Create a new tour
Summary	A user may create a new tour
Status	Passed

Table 6.12: City Explorer test case 12

Test Case #12: covers S2.1	2011-05-10
Name	Setting a PoI as a favourite
Summary	The user may change a PoIs favourite status
Status	Passed

Table 6.13: City Explorer test case 13

Test Case #13: covers S{2.2, 2.3}	2011-05-10
Name	Create a new PoI
Summary	The user may create new PoIs
Status	Passed

Table 6.14: City Explorer test case 14

Test Case #14: covers S2.6	2011-05-10
Name	Create a fixed tour from an existing tour
Summary	The user may create a fixed tour from an existing tour
Status	Passed

Table 6.15: City Explorer test case 15

Test Case #15 : covers S2.4	2011-05-10
Name	Create a new PoI from an existing PoI
Summary	The user may create new PoIs from existing PoIs
Status	Passed

Table 6.16: City Explorer test case 16

Test Case #16: covers S{3.1, 3.2}	2011-05-10
Name	Browsing and downloading PoIs stored on a server
Summary	The user can browse and download PoIs stored on a server
Status	Passed

Table 6.17: City Explorer test case 17

Test Case #17: covers S3.3	2011-05-10
Name	Sharing Pois between devices
Summary	The user can share PoIs with another user
Status	Passed

Table 6.18: City Explorer test case 18

Test Case #18: covers S{3.4, 3.5}	2011-05-10
Name	Browsing and downloading tours stored on a server
Summary	The user can browse and download tours stored on a server
Status	Passed

Chapter 7

Follow-up Work

This chapter contains suggestions for developers wanting to extend and improve on the work done during this project.

7.1 Back end system

Since the specification did not include the construction of a back end Internet server system, the testing system implemented in the program is very simple. A more complex system should be implemented in the future. The current system is based on csv-files[20] simply because they are very easy to parse. The new back end system could possibly communicate with the program using another file format, since the csv file format is less standardised and has less support for meta-data than for example the xml[21] file format. The back end implemented by the group for testing is based on static csv-files hosted on a web server. The future back end should implement a proper database system. The important thing to consider is to base the protocol used in the communication between the back end system and the program on a protocol with a non-persistent connection. This is discussed further in the implementation chapter (see the implementation database section 5.2.5).

7.2 Web Portal

One of the more novel features of City explorer is the ability to create and share user created content with other users of the software. A web portal for the program's user community would allow users and tourist offices to create and update content themselves. To facilitate this, the ability to upload tours and pois directly to a web server could be integrated.

7.3 Sharing of Tours

One point on the original specification list that was removed was the ability to share tours. The reason for this is quite technical. The pois in the local database holds two identifiers, one private ID which identifies the poi in the local database, and one Global ID which identifies the pois in the local database that has been downloaded from the Internet. When a poi is downloaded, it is assigned a private ID. The problem is when you transfer a poi without a Global ID there is no way to distinguish the newly added poi from other pois, since it is automatically assigned a private ID by the SQLite database. Since the private ID is the ID used to assign a poi to a tour, a newly imported poi without a global ID can not be assigned automatically. A temporary identifier could be used to get a handle to the imported poi. But since this method could not eliminate pois already in the database from being duplicated, the decision to remove this specification was reached in agreement with the customer. A method for comparing pois based on content, and a temporary ID could be implemented to resolve this issue in the future.

7.4 Advanced Rating System

One of the ideas discussed in the early parts of the project was to implement an advanced rating system, storing several bits of information about a review and the reviewer. By doing some simple statistics, a system could make very good recommendations for other users based on the user's data (age, interests etc.)

7.5 Social Media Pages

This type of application could benefit from social media site integration. The information shared on these pages could automate an advanced rating system (see section 7.4). The sharing of tours and pois could also be integrated in to the site or sites. Automatic ”check-in” (sharing of a user's location) could also be integrated.

7.6 Plug-in System

The ability to associate behaviors to PoIs (e.g. games) through the plug-in of PoI-specific components was a point on the specification list that has been removed in agreement with the customer. The reason for the removal of this point was uncertainty about how it should be implemented. Two solutions were discussed. First a solution is a pure Android based plug-in system[22]. The problem with this is that deployment becomes difficult, since the developer of the plug-in has to know how City Explorer works. Another problem is that the plug-in software will only be usable together with City Explorer. The last downside is that the plug-in has to be downloaded separately from the application in the Android Market. On the other hand, one upside to this solution is that it is very flexible and fast, allowing for use of all the features of the Android system, as well as allowing for off-line use. The other solution discussed was a web based solution. This allows for easy deployment and updating, no separate downloading required, and the plug-in code can be reused for other web based content. The downside of this solution is that it requires the phone to be connected to the Internet. Both of these solutions could be implemented in the future.

7.7 Support for non-SI units

The specification did not mention the use of units, so SI units of measurement[23] and the 24h time was implemented. The reason we chose this implementation is because most of the people in the world are using SI units. The units are hard coded in to the application, but can easily be changed. A setting to choose the units should be implemented to accommodate different locales.

7.8 Translation

The specification did not mention the use of language, so English seemed to be the most suitable language. The reason we chose this language is because English is understood by most of the people in the world. Most of the text written in the application is written in a separate string.xml file, so it is easy to translate. Some of the text is hard coded, but this should also be easy to change.

7.9 Integration with UbiCompForAll composition tool

When the UbiCompForAll composition tool and back end system has been developed and deployed, some changes to the software have to be made. The integration with the composition tool is on the non-functional requirement list located in the requirement specification chapter (see section 3.4), but since neither the tool, nor the back end was deployed at the time of development, the requirement could not be met.

Appendix A

Meetings

This appendix includes all of our meeting dates, and some meeting minutes which shows examples of how the meetings were conducted. The table A.1 shows an overview of all the formal meetings we have had. The reason we only had three formal group meetings is because our working sessions at school served as forums for our discussions, and we did not find it necessary to document all those discussions. The formal group meetings (see section A.1) includes the meeting minutes for those that were documented. Regarding the meetings with the client (see section A.2), only some examples are documented, because they were performed very similarly all the time. Most of these meetings we were going through the requirement specification list and discussed and tested which requirements that was done. The meetings with our supervisor (see section A.3) is also some examples, because these meetings were almost the same everytime as well. We were going through the report, and got feedback on what we could do better.

Table A.1: Our meetings

Group	Client	Supervisor
20. Jan. 2011	24. Jan. 2011	24. Jan. 2011
26. Jan. 2011	31. Jan. 2011	7. Feb. 2011
11. Mar. 2011	7. Feb. 2011	2. Mar. 2011
	14. Feb. 2011	11. Mar. 2011
	28. Feb. 2011	16. Mar. 2011
	7. Mar. 2011	25. Mar. 2011
	14. Mar. 2011	1. Apr. 2011
	28. Mar. 2011	11. May. 2011
	4. Apr. 2011	
	11. Apr. 2011	
	2. May. 2011	
	9. May. 2011	

A.1 With Group

A.1.1 20. Jan. 2011

Present: Jaroslav, Kristian, Christian, Belen

General thoughts and suggestions Either Google docs or a flat file in a SVN repository may be used to store the project documentation. So far we have one vote for either option and two neutral votes.

We will most likely use an agile methodology to work with the project. Scrum was suggested because some of the members are familiar with it. In reality we will probably have to assimilate elements from other models and remove or alter elements of Scrum because of the constraints of our environment (students working on a project).

A meeting with our supervisor was proposed to be held on Monday 24. Jan 2011. None of the attending members of the group could think of any questions for him. We have to just meet him and listen to any advice he might have for us. It is probably wise to speak with him before we meet the client.

We need to find a date to meet with the client. She has sent us an email with a request to fill out a "doodle" in order to figure out a time which is good for everybody.

A series of questions was asked during the meeting which the non attending members are urged to think about. Some of them include: Do you have an Android phone? Have you got any experience with the platform? Do you have any preferences regarding the development model for the project? Do you have any questions for the client?

Because of the clumsy nature of email when there are more than two participants in a conversation, creating an IRC channel was suggested. Some good clients are X-chat, mIRC, Irssi and for those of you with smart phones there are probably good alternatives in your favourite app store. Have a look and tell us what you think.

Questions for the client

- Do we get a SVN repository?
- Are there any competing applications?
- Has any research been done among potential users?
- Do we get one or more Android phones to develop on?
- Which version of the platform are we supposed to develop on?
- Is there a back end for the information to be stored?
- If a back end exists, what formats does it use? Is there any documentation?
- Do we need to write an application for PCs and tablets?
- Why is it a good idea to develop this application? (For the report)
- Are we trying to provide an alternative to something?
- Is there huge money to be made?
- Someone's personal dream?
- Does the intended functionality of the application (e.g. navigation using Google Maps) violate some license?

Preliminary report due on 31. Jan. We had a look at the requirements for the preliminary report defined in one of the documents on It's Learning. The following points must be documented:

- Development model (waterfall, scrum, XP)
- Description of the environment
- Definition of the problem
    + Description of the application
    + Functional and non-functional requirements

- Possible obstacles we are facing: map license (means we will use openstreetmap.org), inadequate knowledge of the Android platform
- Brief outline of alternative solutions
- High level outline of the architecture
- Team description (roles and members)
- Basic time plan

Homework Think about what to ask the client and the supervisor, read the documents attached in the email from the client and read up on Android.

A.1.2 6. Jan. 2011

Present: Jaroslav, Christian, Belen

Short summary We continued more or less were we left off the last time which means that the main focus of our discussion was centered around the report. We worked out the table of contents in more detail and talked about what the different sections should contain. The structure presented in the "Preliminary report table of contents" below does not have to be how we put the document together in the final version, but it would be nice to have a logical flow.

Preliminary report table of contents

- 1.0 Introduction to the project - problem definition - high level diagram to show our "corner" of the client's system

- 1.x Introduction of the group
  - roles and responsibilities
  - scrum has a flat structure and we think it fits our group well
  - difficulty of assigning a leader is present even more than in any other situation be cause we do not know each other - we are not avoiding responsibility, but sharing it because
  the grade is shared and because it is too easy to blame
  some particular role in a project. Everyone has every role.

- 2.0 Model description - choice of process model - the client made it clear that changes might arise
  - we are welcome to suggest features
  - the client needs first and foremost a good GUI. Making several prototypes is an approach to GUI development with many
  practical benefits.
  - the group is comfortable with agile development because we know it from before

- 2.x Development environment
  - Eclipse, Android 2.2, SVN, MySQL
  - Why are we choosing these environments? - Primarily phones, but

also a tablet if the client provides it

- 3.0 Project description
  - more diagrams: program classes, architecture, use cases etc.
  - description of the technical issues - possibly a handbook for the application
  - features of the application
- 3.x Description of a layered app and the layers
  - UI, interface, back end and other modules
  - back end is not going to be very solid or reusable
  - focus on UI and an interface to a swappable DB

- 3.x Requirement specifications
  - functional
  - non-functional
  - some changes are justly desired because there are cases of incoherence in the list of features between some highly prioritized items and the envisioned development progress
  - the client needs to know that their list is problematic
- 4.0 Brief outline of alternatives
  - why are we writing this app?
  - reuse client's research

- 5.0 Miscellaneous
  - time plan
  - risk analysis

Notes This outline might have gone somewhat beyond what is necessary for the preliminary version. Nevertheless, members present at the meeting took the liberty of either assigning to others, or divide amongst themselves, all sections mentioned in the "Preliminary report table of contents" above:

1.0 - Everyone 1 and 2 (Group intro) - John Arne 1.0 and 3 (Diagrams) - Christian 3 (Requirements) - Kristian 3.0 (Only a start) - Belen 5 and 4 (Risk excluded) - Jaroslav

Homework You might or might not remember from English class an exercise where pupils are instructed to summarise a story or present an idea in 50 words or less. The idea is that you write half a page, figure out what is really essential, trim it down to 50 words (2-4 sentences) and discard everything else. This might be a good way to present our problem in the very beginning of the report. The assignment is for everyone to make a few sentences, so that we may pick out some good ideas and put together a mean definition of our problem in a compact paragraph.

We need to do some work before the weekend. Everyone works on their sections until 17:00 on Thursday (27. Jan. 2011) and upload what they have by then. The idea is for us to have something to talk about when we meet on Friday. The section 1.0 problem definition is for Friday.

A.1.3 11. March. 2011

Present: Jaroslav, Christian, Oscar

The report needs updating Some things that needs to be done are listed in the next paragraph, other things that were pointed out by the supervisor at the last meeting will be mailed by Christian any minute now.

Report "To do's" for Tuesday 15. Mar. 2011

- Move sprints to the appendix
- Split the following specifications: 1.2 1.5 1.6 1.7 2.2 2.5 2.6 2.7
- Risk list: write about it (Oscar)
- Architecture: write section text (Christian)
- New section: glossary
- Intro: remove "(henceforth CE)"

Poor planning In order do be able to predict the impact of someone loosing time in a sprint, we will have to assign or pick all tasks in a sprint before it starts. To solve the problem further, that is, to be able to predict the impact on the project result of a member quitting or similar, we will assign responsibilities to people. The responsibilities will be the various work-packages.

Homework See "To do's" and the mail from Christian.

A.2 With Client

A.2.1 24. Jan. 2011

- We will receive one device for development
- The client wants to see a plan of the project for the next meeting
- The project will be licensed under the Apache License
- Future meetings will find place in Sintef ITC on Mondays at 14:15
- A PoI might be/contain:
  - Description
  - Game
  - History Application
  - Quizzes
- A tour can contain several PoIs
- We were asked to think about how to make the application work in offline and online modes

A.2.2 31. Jan. 2011

Comments on the preliminary report

- Remember to write on the importance of documentation

Until the next meeting

- Make diagram for the GUI.
- Working on simple GUI components.

A.2.3 7. Feb. 2011.

Comments on the preliminary report

- Justify our choices for development environment. (e.g. MySQL)
- The architecture is not documented
- Backlog: should be more detailed and specify when the various parts will be done (does not really work that way)

Comments on the GUI (see figure 4.2)

- Button to edit a tour.
- Add poi from map (minimum time for GPS to stabilize?)

- New/edit PoI screen: enable setting the location on a map, or by using your current location.
- Tour screen:

  To-from times on each POI

  How long do we spent in each place?

  Calendar view

- How to distinguish my own content from public?

Other

- Consider using local and/or Internet database. We will try both.

A.2.4 14. Feb. 2011

Present: Jaroslav

- Non-f.req.: Should be extendible to accommodate new types of PoI entities.
  - Open a poi as a game
  - Activities related to places or cities
- The importance of modifying user created content was stressed
- Sprint 3:

  0. Last sprint
  1. PoI on map, map <-> list (poiview)
  2. Calendar view
  3. Map - fixed tour

- A local database was more important than an Internet based because the client intended to bring this prototype to possible stakeholders such as "arkitektforeningen" and ask them about making a tour out of their book about Trondheim, for example.
- We agreed on the term "tour" instead of "itinerary" which was originally proposed by our client.

A.2.5 28. Feb. 2011

Present: Jaroslav, Belen, Christian, Kristian, Oscar
Items for sprint5

- Display modes for a tour: List view and calendar view
- Create an activity for selecting or entering details about "to" and "from" before invoking Google Maps context aware (pois in a list of suggestions should be in proximity of location
- Add a menu button in tour view: [get directions]
- Add a button to QA in map-view: [navigate here]

Other

- Send .apk to the client
- Collect signatures from members who have not signed the contract
- Documentation (the most significant parts of the api are): The API (javadoc) and architecture models
- "My way is not the only way" - make suggestions to alternative approaches

A.3 With Supervisor

A.3.1 24. Jan. 2011

Present: Jaroslav, Kristian, Belen, John Arne

This meeting was just a formality so we could meet our supervisor. He talked a little bit about what he could do for us, and what he could not do. To sum up, he can only assist us with the project management and the report. He will not provide us with assistance regarding the application.

A.3.2 7. Feb. 2011

Comments on the preliminary report and work so far

- We have to provide rationale for why we will not use commandline or e.g. NetBeans instead of Eclipse.
- We have to have a status report, so we can be more aware of some issues during the project.
- We need a risk analysis.
- Meeting minutes should be in the report.
- We have to have one person that makes all contact with the customer and the supervisor.
- We need to include the difference between functional and non-functional requirements.
- Improve the structure of the report.
- All members of the group have to read through the report, so that we catch typos and poor sentence structure.
- A more objective perspective is needed when writing the report.
- Need to add references to things like SCRUM, MySQL and other tools or brands we are using.
- Need to add estimated hours and actual hours on the different phases of the project(e.g. in the sprints).
- Send everything to the supervisor at least 24 hours before a meeting with him.

Appendix B

Status Reports

Some of the column headers in the activity plan are abbreviated. Their meaning is defined to the right:

WP	Work package
Res.	Resource
Est.	Work hours planned
Hrs.	Work hours spent

B.1 Status report: sprints 1 and 2

The first two sprints of our project were spent working on a sketch of the GUI. After the GUI navigation flow was done, we worked on realising the GUI elements in terms of Android's XML layouts. We spent the first week working on the project documentation and with the backlog.

Group update We discussed the content of the preliminary report and delegated sections for everyone to work with. We abandoned the google docs repository and made a report folder in the SVN repository for the documentation with plain-TEXt files instead.

Progress summary Tasks that have been performed by the group are described in sprints 1 and 2, see tables C.2 and C.3 in section C.2.

Problems

- GUI components not properly documented
- Architecture design missing
- Routines not followed or documented (sprints, status reports)

B.2 Status report: sprints 3 and 4

City Explorer graphic user interface works and can navigate through some of the different options. The Plan activity shows a list of PoI and is able to show them on a map. The Map activity can show a tour with a path between them. A class has been written for both MySQL (JDBC) and SQLite (Android).

Group update Christian was ill during the first week of this period.

On the 20th John Arne informed us that he was leaving to attend his grandfathers funeral and would not be back until the following Sunday. Since the midterm delivery is due this week, the rest of the group has to increase their workload. The group has unfortunately not worked sufficiently evenly with the report up till now, and as an effect, we are unable to follow the risk analyses exactly, and have to increase the number of hours worked by the rest of the group.

The group agrees to work more steadily with the report to avoid this situation in the future.

During the last days of the period, some members does not contribute to the satisfaction of the rest. The rest of the group have to work even more hours to cover for the work not being done by these members in addition to John Arnes work.

The Facebook group page does not seem to be read by all members, despite them being urged to read it as well as reprimanded via mail for not doing so.

Progress summary During the period of sprint 3 and 4, alot of work has been done on the report aswell as the GUI of the application. We now have a functioning user interface, which makes it easier to do personal testing on the application. Detailed tasks are described in Sprints 3 and 4, see tables C.4 and C.5 in section C.2.

Problems

- GUI components not properly documented
- Architecture design missing
- Some members participating less than desired

  Need to discuss the situation

- Time spent on project activities not documented
- Flat group structure not working

Risk analysis updates Added an entry for illness.

B.3 Status report: sprint 5

Some items from the previous sprints have accumulated to this date. Our source code has more functionality than the GUI can show. The intention for these two weeks was to resolve those issues.

Group update A lot of programming and report work is done at the start of this sprint, mainly by 4 members of the group. Attempts to make the remaining members contribute failed due to unresponsiveness to emails and Facebook messages. After the first week of the period Belen informs us by mail that she was dropping the subject because she did not feel she was able to contribute satisfactory, listing time constraints and lack of knowledge as reasons.

The risk analyses does not give a complete picture in this type of event. In reality the consequences, and the solution is more complex. There are three possible outcomes and solutions to problem. The first is that the group have to take on the entire workload for the leaving member. The second outcome is to agree with the customer to reduce the workload by dropping some of the lower priority specs. The final solution is to compromise between the first two outcomes. The rest of the group takes on some of the responsibilities of the missing member as well as reducing the total workload.

In a usual case, the choice of solution is governed by cost, but since the only resource we have in this project is time, the increase in cost is acceptable as long as the group members are able and willing to work more. The group will probably not be able to take on all of the responsibilities and workload of the missing member, so the solution chosen is the compromise solution. The group will discuss with the client about dropping some of the requirements. We anticipate that this will be fine based on discussions on the subject in an earlier meeting.

After discussing amongst the group and having a meeting with the client, alot of changes have been made to the functional requirements, see the paragraph "Requirements update" B.3 below for details.

Progress summary Tasks that have been performed by the group are described in Sprint 5 C.6 (section:C.2).

We have had to postpone some of the milestones. One problem caused by this is that tailoring and sharing requirements have to be documented after the deadline for delivering a version of the report to which we can get comments. We will have to finish the documentation of those parts as much as possible before the implementation is completed in order to benefit from comments by the supervisor.

Requirements update

- requirement 1.2 has been split into 1.2 and 1.21
- requirement 1.6 has been split into 1.6 and 1.61
- requirement 1.7 has been split into 1.7 and 1.71
- requirement 2.6 has been split into 2.6 and 2.61
- requirement 2.7 has been split into 2.7 and 2.71
- requirement 2.2, previously being "Showing favourite PoIs in a list view or in a map view", was dropped and rewritten as "Adding categories to user created PoIs".
- requirement 2.3', "Easily switching between a favourite PoIs list view and map view", was dropped because it is covered in 1.3.
- requirement 2.4, "Filtering favourite PoIs according to categories" was dropped because it is covered in 1.4.
- requirement 2.5, "Navigation to a favourite PoI from the current location or any other place", was dropped because it is covered in 1.5.
- requirement 2.8, "Showing a fixed tour created by the user in a calendar or in a map" has been dropped because it is covered in 1.6 and 1.61.
- requirement 2.9, "Showing a free tour created by the user in a calendar or in a map" has been dropped because it is covered in 1.7 and 1.71.
- requirement 2.10, "Going through a tour created by the user", has been dropped because it is covered in 1.8.
- requirement 2.11 has been moved to 2.3.
- requirement 2.12 has been moved to 2.4.

Problems

- stil many shaky sections in the report.
- status reports not done properly.
- sprint 6 not planned.

B.4 Status report: sprint 6

Two meetings have been held with the supervisor in this period, 16th and 25th of March. We received comments on the report; improvements were made.

Group update After the mid semester report delivery and correctionsafter comments from the supervisor, the group puts the report out of their minds and starts focusing on the coding. This results in that a lot of progress is made on the program; several classes are made which makes the general outline of the program and workload easier to visualize for all of the group members. Positive feedback on the program is received by the customer from the meeting 14th of March.

Progress summary Programming is done on details of PoIs, bugs are fixed, icon support for MapView is implemented, and more. In general, alot of refining has been done on the program. See table C.7 for detailed information.

Problems

- still many shaky sections in the report.
- people are still bad at documenting work.

B.5 Status report: sprint 7

Two weekly meetings with the client has been held, 28th of March and 4th of April, and testing on the application based on the use cases was done on both dates.

Group update The group has talked about the importance of documenting work, not only the documentation done in the backlog for use in the report, but also the importance of documenting the changes comitted to the SVN server. These comments are very useful when developing, since one can easily check what has been done recently by other developers.

Progress summary In addition, research has been done on how to implement calendar times to the tours, and it seems to show that the hours on this is overestimated. See table C.8 in section C.2 for more details on work done in sprint 7.

Problems

- time constraints may lead us to not being able to implement the "Sharing requirements"(see table 3.3 in section 3.1).

B.6 Status report: sprint 8

During this period two meetings have been held with the client, on the 11th of April and 2nd of May. A meeting with the supervisor was held at 1st of April.

Group update Since the start of this period we've had trouble contacting Jaroslav, which has resulted in no work being done by this group member the last weeks. Attempts to contact him via e-mail, phone and facebook have all proven unfeasible. While the group is concerned, both about the condition of the grou member aswell as the time constraints, nothing is done on this problem except for increasing the work load on the remaining members.

Progress summary After meetings with the client, several changes were recommended. Some of the time in this sprint went to correcting them. See table C.9 in section C.2 for more details on work done in this period.

Problems

- unable to get a hold of a group member for weeks.

B.7 Status report: last call

During this period one meeting was held with the client on the 9th of May. Testing was perfomered with the client on the finished application. The client seemed to be pleased with the result.

A meeting was held with the supervisor to dicuss the uneven workload during the project, aswell as the group member who has been missing for several weeks.

Group update The supervisor recommended us to set up a work distribution chart, depicting how the total workload of the project has been distributed among the group members. Work on that was done this week and the results were sent to the supervisor.

Continuing efforts to contact the missing group member have proven to be less than successful. The group reached him on the phone at one point, but the only responce from him was that he was not very motivated to work. We have yet to reach him since. This has been grounds for concern from the rest of the group; since this was the last week before delivery there has been a lot of work to be done.

Progress summary The last planned elements of the applications were implemented before the meeting with the client at the beginning of the sprint. After that, mostly work on the report has been done.

On the report we have refined alot of the sections, and gone through the whole report correcting typos and other erroneous elements. See table C.10 in section C.2 for more details on work done in this period.

Problems

- at time of delivery, Jaroslav is still not reachable.

Appendix C

Backlog

This appendix includes the Project backlog and all of the sprints. These concepts is a part of the SCRUM development model [7].

C.1 Project backlog

Table C.1 shows us the project backlog. This is an overview of all the sprints.

Table C.1: Project backlog

					Sprint Nr.	1	2	3	4	5	6	7	8	Last call
Item title	Description	Requirement spec.			Hrs est	102	98	87	87	186	220	191	174	270
Report	Writing the documentation					49				52	64	66	18	172
Maintenance	Resolving issues, code documentation, updating elements					43		16	17	2			52	9
Management	Planning activities, meetings					16	13	2			10	3	3	7
Design	Graphical and structural design and implementation							10	6	12	16	14	12	
PoI	Displaying information about a PoI	S1.1					2	12		11	8		2	
PoI new/edit	Creating a PoI	S3.3 S2.2	S2.11	S2.12										
Tour new/edit	Creating a tour	S2.6 S.27	S2.61 S2.71	 S3.6						16	8	3		
Tour	Displaying information about a tour	S1.6 S1.7	S1.61 S1.71	S1.8					22	29	12	25	11	
Map view	Displaying PoIs and tours in a map	S1.21 S1.3	S1.61 S1.5	S1.71 S2.1			42	0	6	12	17	4		
Plan	Showing tours and PoIs in a list	S1.4 S3.4 S3.1 S2.1	S1.2 S3.5 S3.2 S1.5	S1.3			24	11	6	24			25	16
Other	Bugfixing and coding that does not fit in other categories			S3.6					4	1	19	24	36	28
Database	Store and retrieve information	S3.1 S3.4	S3.2 S3.5	S3.6 S3.6				10	12	9			5	
					Hours Spent	108	81	61	73	168	154	139	164	232

C.2 Sprints

Table C.2 through table C.8 shows us all of the sprints in the project. The sprints includes sprint items, which is small packages of work that needs to be done. It also shows us the estimated time of each individual sprint item, the persons who is responsible for doing them, and actual hours spent. The top rows in the tables indicate the initials of the group members. C is Christian Berg Skjetne, K is Kristian Greve Hagen, J.Ø. is John Arne Øye, O is Oscar Aarseth, J.R. is Jaroslav Rakhmatoullin and B is Maria Belen Gallego Garcia.

C.2.1 Sprint 1

Table C.2: Sprint 1: 31st of January - 6th of February

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K	B
Report	1	Problem description	4	3		2			2
Report	2	Architecture	5		4				
Report	3	Group organization	3	2		2			
Report	4	Requirement specifications	4					4	
Report	5	Development environment	8	2	2	2			
Report	2	Time plan	3				4		
Report	7	Alternatives	3				4		
Report	8	Sketching GUI	15	3	3	3	3	3	
Report	9	Digitalizing GUI	3		3				
Management	10	Meeting with the client	5		1	1	1	1	
Management	11	Meeting with the group	5	1	1	1	1	1	
Management	12	Backlog: Initial version	3			2	2		
Management	13	Backlog: Sprint 1	1				1		
Management	14	Backlog: Sprint 2	1			1			
Management	15	Setting up Mumble	1		1				
Maintenance	16	Environment: Eclipse	10	2	2	2	2	2	
Maintenance	17	Environment: SVN	10	2	2	2	2	2	
Maintenance	18	Environment: Android SDK	10	2	2	2	2	2	
Maintenance	19	Mobile phone drivers	5		3				
Maintenance	20	Report: Latex	3				4	6	
		Total	102	17	24	20	26	21	2

C.2.2 Sprint 2

Table C.3: Sprint 2: 7th of February - 13th of February

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K
Management	10	Meeting with the client	5	1	1	1		1
Management	21	Meeting with the supervisor	5	1	1	1		1
Management	11	Meeting with the group	5	1	1	1	1	1
Poi	22	Menu buttons for navigating to PoI details and mapview: S1.1, S1.3	3				2	
Plan	23	Showing PoIs in a listview: S1.2	20			3	3	18
Map view	24	Showing PoIs in a mapview: S1.21	30	2	15	4	5	1
Map view	25	Showing a free tour in a mapview: S1.71	30	5	5	5		
		Total	98	10	23	15	11	22

C.2.3 Sprint 3

Table C.4: Sprint 3: 14th of February - 20th of February

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K
Management	10	Meeting with the client	5				2	
Design	26	Splash screen	15					10
Database	27	Create locations for testing	10			4	6	
Plan	28	Connect poi list view to a provider	2				2	
Plan	29	Selecting favourite pois: S2.1	12		8			
Plan	30	Add support for categories	6			1		
PoI	31	Create an activity for displaying information about a place: S1.1	5				2	2
PoI	32	Create the layout of the details of a PoI: S1.1	4					8
Map view	33	Make the map pins react to actions and add support for zooming	8					
Maintenance	34	Revise the Backlog	10					
Maintenance	35	Switching views (QuickActionPopup): S1.3	8				12	
Maintenance	36	Rewrite the PoI list to use an Adapter in a ListView	2			2	2	
		Total	87	0	8	7	26	20

C.2.4 Sprint 4

Table C.5: Sprint 4: 21st of February - 27th of February

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K
Management	10	Meeting with the client	5					
Tour	37	Showing a free tour in a list view: S1.7	30				12	10
Other	38	Figure out how to pass data between activities: S1.3	5		2		2	
Plan	30	Add support for categories	6					4
Plan	39	Enable the search button to bring up a search and filter overlay: S1.4	3					2
Design	40	Theme and style touch ups	5	2				4
Map view	33	Make the map-pins react to actions and add support for zooming to them	8		6			
Database	41	Create database schema	5		4			4
Database	42	Add support for fetching data from the database	5		4			
Maintenance	34	Revise the Backlog	10	4			8	
Maintenance	43	Set up the databases: MySQL and SQLite	5		5			
		Total	87	6	21	0	22	24

C.2.5 Sprint 5

Table C.6: Sprint 5: 28th of February - 13th of March

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K
Design	32	Improve the layout of the details of a PoI: S1.1	6			8		
Design	44	Xml layouts and bitmaps: S1.5	10					
Design	45	Xml layouts and bitmaps: S2.6	8					4
Tour	46	Show fixed tour in mapview: S1.61	8		8			
Tour	47	Make a quick action for all PoIs in a list: S1.5	4			6		
Tour	25	Show free tour in mapview: S1.71	15			5	4	
Tour	37	Showing a free tour in a listview: S1.7	6			2		4
Plan	48	Show favourite PoIs in listview: S1.2	5		4			3
Plan	49	Make the PoI list respect the filter selections: S1.4	8		2			8
Plan	50	Show favourite PoIs in mapview: S1.21	5		4			3
PoI	51	Menu button in detailview: S1.5	6			6		
PoI	29	Selecting favourite PoIs: S2.1	5		4			1
MapView	52	Make a quick action for all PoIs in a map: S1.5	16		10	2		
Tour edit	53	Create New Tours activity: S2.6, S2.7	16					16
Maintenance	54	Add author field to tours and PoIs: (S2.6, S2.7, S2.10)	1					1
Maintenance	55	Update the affected tables in the database, make new primary key	1			1		
Database	56	Convert from MySQL to SQLite	6		9			
Other	57	Add support for installing to SD card	1					1
Other	58	Create intermediate activity between ”navigate to” and Google Maps: S1.5	16					
Report	59	Write status report, risks, stakeholders	16	5			16	
Report	60	Formatting and user interface	8				8	
Report	61	Use Cases	9	2			9	
Report	62	WBS	6				6	
Report	63	Requirements and sprints	4	2			4	
		Total	186	9	41	30	47	41

C.2.6 Sprint 6

Table C.7: Sprint 6: 14th of March - 27th of March

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K
PoI	64	Add support for images: S1.1	6			6		2
Tour	65	Loop through all PoIs in a tour in a list view: S1.8	6			2		3
Tour	47	Make a quick action for all PoIs in a list: S1.5	8			2		2
Tour	66	Create activity calendarview: S1.6	40					
Tour	37	Showing a free tour in a listview: S1.7	7					3
Design	67	Redesign menubuttons: S1.1	6		1	6		1
Design	45	Xml layouts and bitmaps: S2.6	8					4
Design	44	Xml layouts and bitmaps: S1.5	10					4
MapView	52	Make a quick action for all PoIs in a map: S1.5	8		8			
MapView	68	Add support for icons	5		8			
MapView	69	Show all nearby PoIs when clicking on explore	3					1
Tour edit	53	Create activity for creating tours: S2.6, S2.61	16					8
Management	70	Make a general strategy for how to create the calenderview	16		10			
Report	61	Use Cases	3				10	
Report	61	Make Use Cases for Tailoring and Sharing	20	5			10	
Report	59	Risk list: Sort by severity	1					1
Report	71	Add glossary	5					1
Report	72	Handle suggestions from the supervisor	10	4	6			3
Report	73	Make status report for sprint 4	10		2		8	
Report	74	Make status report for sprint 5	6	2	2	2	6	2
Other	58	Create intermediate activity between ”navigate to” and Google Maps: S1.5	16					12
Other	75	Bugfixing	10		4	1	1	1
		Total	220	11	41	19	35	48

C.2.7 Sprint 7

Table C.8: Sprint 7: 28th of March - 10th of April

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K
Management	10	Meeting with the client	5		1	1		1
Mapview	76	Loop through all PoIs in a tour in mapview: S1.8	4		4			
Design	67	Redesign menubuttons: S1.1	1		1	1		1
Design	77	Redesign tab buttons: S1.1	12			6		5
Tour	66	Create activity calendarview: S1.6	40		18	6		
Tour	78	Quick action for making a new tour out of an existing: S2.6	4					
Tour	79	Xml layouts and bitmaps: S2.6, S2.7	2					1
Tour edit	53	Add field free or fixed in new tour: S2.6, S2.61	2					3
Report	80	Fix the changes recommended by the supervisor	80	4	15	3	10	25
Report	81	Update database diagram with new fields and primary keys	2		2			
Report	61	Use Cases	3		1			1
Report	61	Make Use Cases for Tailoring and sharing	10				5	
Report	82	Make status report for sprint 6	16					
Other	75	Bugfixing	10		2	5	12	5
		Total	191	4	44	22	27	42

C.2.8 Sprint 8

Table C.9: Sprint 8: 25th of April - 8th of May

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K
Management	10	Meeting with the client	5		1	1		1
PoI	83	Added geocoding when adding new pois	3					2
Report	61	Use Cases	16		5			11
Report	85	Added testing section	2					2
Plan	91	Implementing sharing of PoIs	16		8			8
Plan	87	Remaking category filter	10		5			4
Tour	78	Making a new tour from an existing tour: S2.6	5					3
Tour	86	Implementing calendar view: S1.6	8		8			
Design	77	Redesign tab buttons: S1.1	3			4		
Design	89	Xml layouts and bitmaps	10			6		2
Database	88	Update database diagram with new fields and primary keys	6		2	3		
Other	92	Browsing and downloading PoIs	16		8			8
Other	93	Browsing and downloading tours	16		8			8
Other	75	Bugfixing	10			4		
Maintenance	94	Fixing Javadoc	8			5		4
Maintenance	84	Code revert and bugfixing	24		16			8
Maintenance	90	Implementing changes recommended by the client	16		5	9		5
		Total	174	0	66	32	0	66

C.2.9 Last call

Table C.10: Last call: 9th of May - 15th of May

Backlog Item	Sprint Item	Description	Hrs est	O	C	J.Ø.	J.R.	K
Management	10	Meeting with the client	5		1	1		1
Management	21	Meeting with the supervisor	5	1	1	1		1
Report	85	Writing testing section	2					2
Report	95	Finalization of the report	200	20	50	50		50
Other	75	Bugfixing	10			4		
Other	93	Browsing and downloading tours	24		8	8		8
Plan	91	Implementing sharing of PoIs	16		8			8
Maintenance	94	Fixing Javadoc	8			5		4
		Total	270	21	68	69	0	74

Appendix D

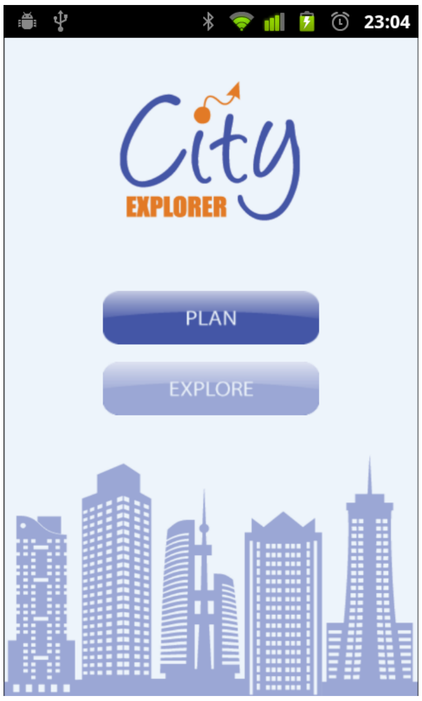
User Manual

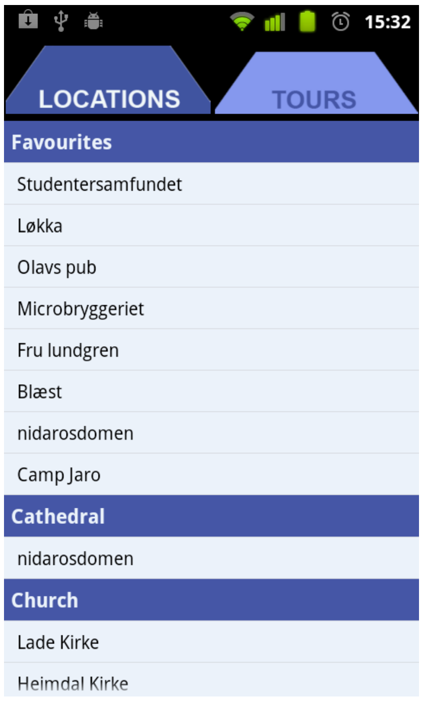
This manual goes through the different uses of the City Explorer.

D.1 Exploring

This part of the User Manual involves the users ability to explore a city of previously unknown pubs, shops and other sights to see, using the City Explorer on Android.

Figure D.1: The opening screen(left) and the Locations screen(right)




15:32

LOCATIONS | TOURS

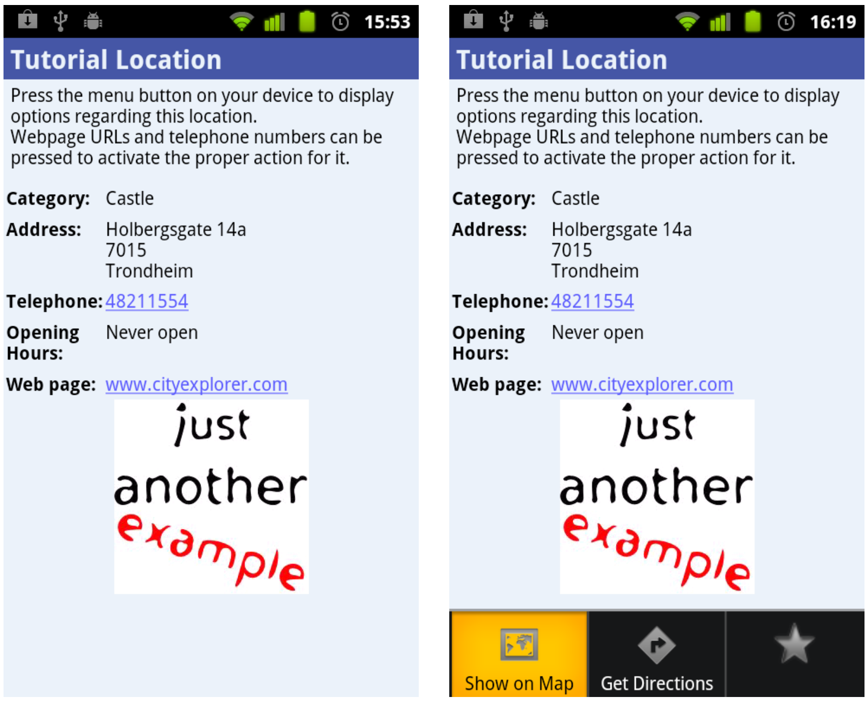
Favourites
- Studentersamfundet
- Løkka
- Olavs pub
- Microbryggeriet
- Fru lundgren
- Blæst
- nidarosdomen
- Camp Jaro

Cathedral
- nidarosdomen

Church
- Lade Kirke
- Heimdal Kirke

The screen to the left in figure D.1 is the main screen that is shown when you run the City Explorer. As you can see there are two buttons:"Plan" and "Explore". Starting off we will delve into the usage of the "Plan" button, and then explain the "Explore" button in the end of this section.

Figure D.2: The location details screen(left) and the "Show Map" button(right)



15:53

Tutorial Location

Press the menu button on your device to display options regarding this location.
Webpage URLs and telephone numbers can be pressed to activate the proper action for it.

Category: Castle

Address: Holbergsgate 14a
7015
Trondheim

Telephone: 48211554

Opening Hours: Never open

Web page: www.cityexplorer.com

16:19

Tutorial Location

Press the menu button on your device to display options regarding this location.
Webpage URLs and telephone numbers can be pressed to activate the proper action for it.

Category: Castle

Address: Holbergsgate 14a
7015
Trondheim

Telephone: 48211554

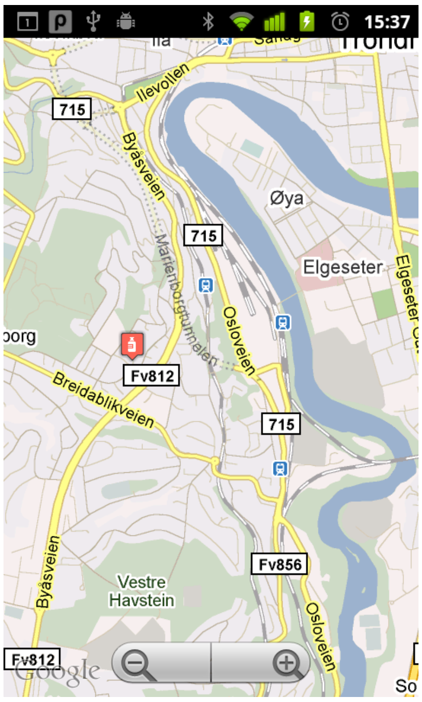
Opening Hours: Never open

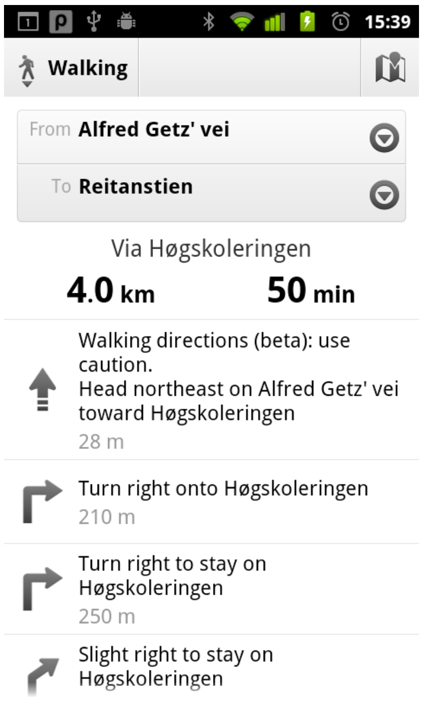
Web page: www.cityexplorer.com

Viewing locations. This is the most basic part of the application; the ability to view information about different locations in a city. The screen you see to the right in figure D.1 is the screen that appears when you pressed the "Plan" button in the opening screen. As you can see this is a list with names of different locations in the city, seperated by the categories to which they belong. If you press one of the entries, detailed information about that location will appear, as seen on the screen on the left in figure D.2.

Pressing the web page on the details screen will open the web browser with the corresponding url address. Pressing the telephone number will activate the call function and the number will be called with your android phone. As seen on the screen to the left in figure D.2, more options will be available if you press the menu button on your phone. In this image the "Map Button" is highlighted, pressing this button will show you a map with the address of this location shown, as seen on the left screen in figure D.3. On the screen beside you have what is shown if you press the "Get Directions" button, which will give you directions on how to get to the location.

Figure D.3: The map view(left) and the directions to a location(right)




15:39

Walking

From **Alfred Getz' vei**

To **Reitanstien**

Via Høgskoleringen

4.0 km **50** min

Walking directions (beta): use caution.
Head northeast on Alfred Getz' vei toward Høgskoleringen
28 m

Turn right onto Høgskoleringen
210 m

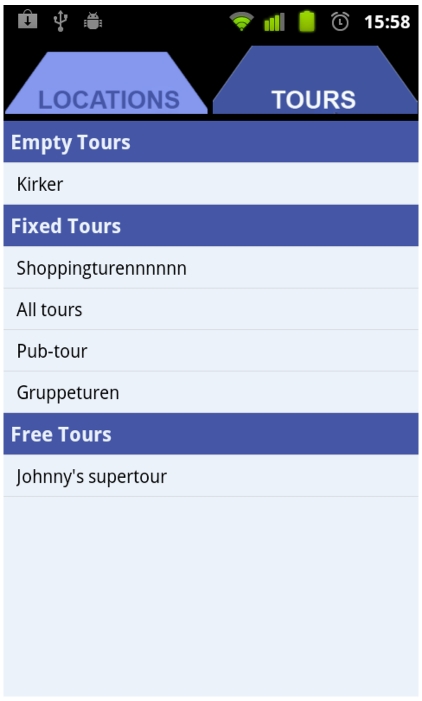
Turn right to stay on Høgskoleringen
250 m

Slight right to stay on Høgskoleringen

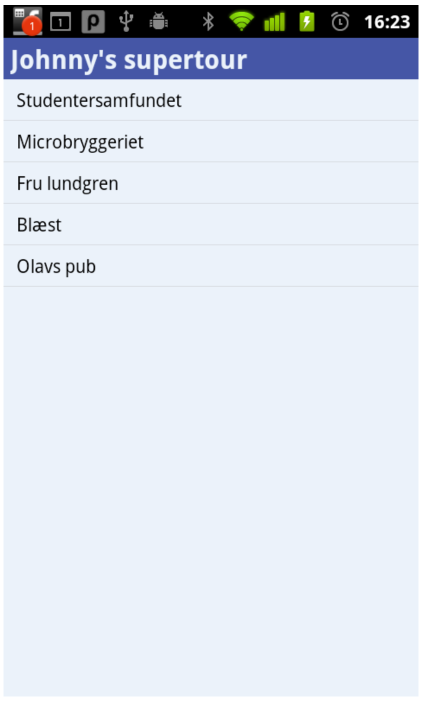
The "Star" button on the menu to the right in figure D.2, can be pressed if you want to add this location to your own personal list of favourites.

Viewing tours. This part allows you to view different tours around the city. A tour is basically just a collection of locations, often suited to a set of interests(e.g. a Pub-tour or a Shopping-tour).

Figure D.4: The tour list(left) and the list of locations in a tour(right)



Empty Tours
Kirker
Fixed Tours
Shoppingturennnnnn
All tours
Pub-tour
Gruppeturen
Free Tours
Johnny's supertour

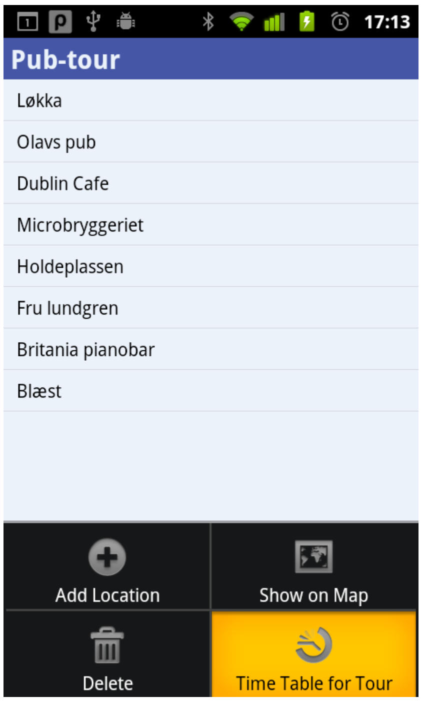


Johnny's supertour
Studentersamfundet
Microbryggeriet
Fru lundgren
Blæst
Olavs pub

To get to the list of tours you simply press the tab marked in to the left in figure D.4. Here you see a list over different tours, separated by 3 categories: "Fixed Tours", "Free Tours" and "Empty Tours". The main difference between a fixed and a free tour is that a fixed tour is set to a time-schedule while a free tour is not planned to be taken at any specific time during the day. An empty tour is of course a tour that contains no locations. The screen to the right on figure D.4 is the list you view when you press one of the tours in the previous list.

If you look at figure D.5 you can see the left screen shows the menu, with the button for showing timetable for this fixed tour highlighted. The screen to the left is the timetable that appears when you press that button. In the timetable you have the locations from the tour lined up at the time you are supposed to visit them, aswell as the time to walk between them. If you press the locations you will get to the detail screen, if you press the walking distanses you will get the walking directions.

Figure D.5: The tour list(left) and the list of locations in a tour(right)


17:13

Pub-tour

Løkka

Olavs pub

Dublin Cafe

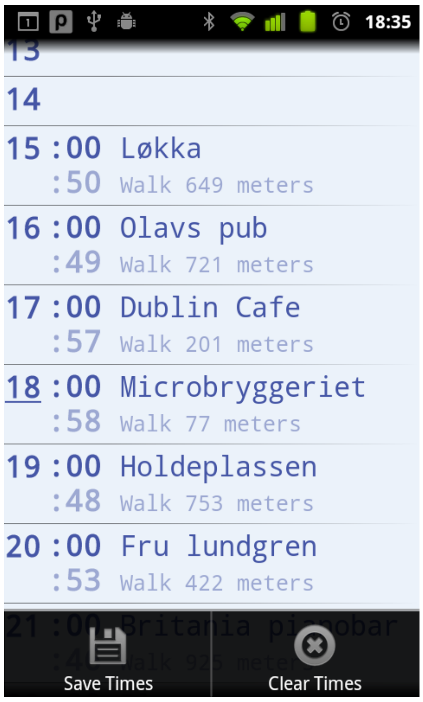
Microbryggeriet

Holdeplassen

Fru lundgren

Britania pianobar

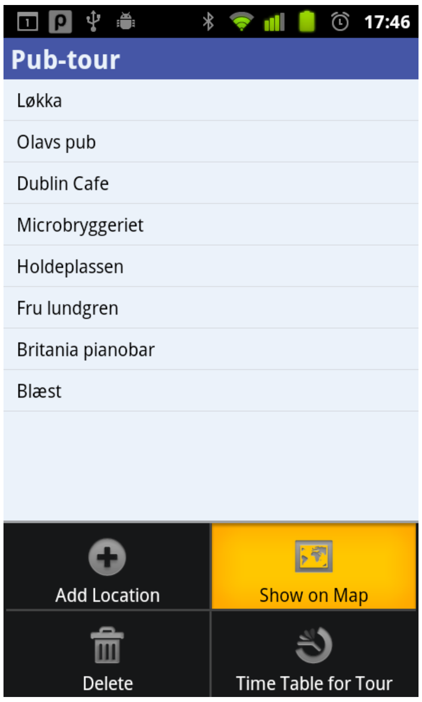
Blæst


18:35

```
13
14
15 :00 Løkka
   :50 Walk 649 meters
16 :00 Olavs pub
   :49 Walk 721 meters
17 :00 Dublin Cafe
   :57 Walk 201 meters
18 :00 Microbryggeriet
   :58 Walk 77 meters
19 :00 Holdeplassen
   :48 Walk 753 meters
20 :00 Fru lundgren
   :53 Walk 422 meters
21 :00 Britania pianobar
```

From the menu when you've entered a tour, you also have the possibility to view all the locations on a map, by pressing the button as shown in figure D.6, with directions between the locations, as shown to the right in the same figure. The arrows on top of the screen can be used to cycle through the locations.

Figure D.6: The button for opening map in a tour(left) and the corresponding map with locations(right)


17:46

Pub-tour

Løkka

Olavs pub

Dublin Cafe

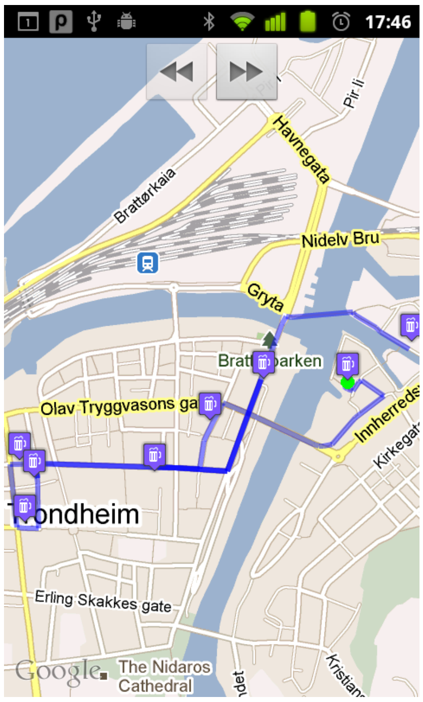
Microbryggeriet

Holdeplassen

Fru lundgren

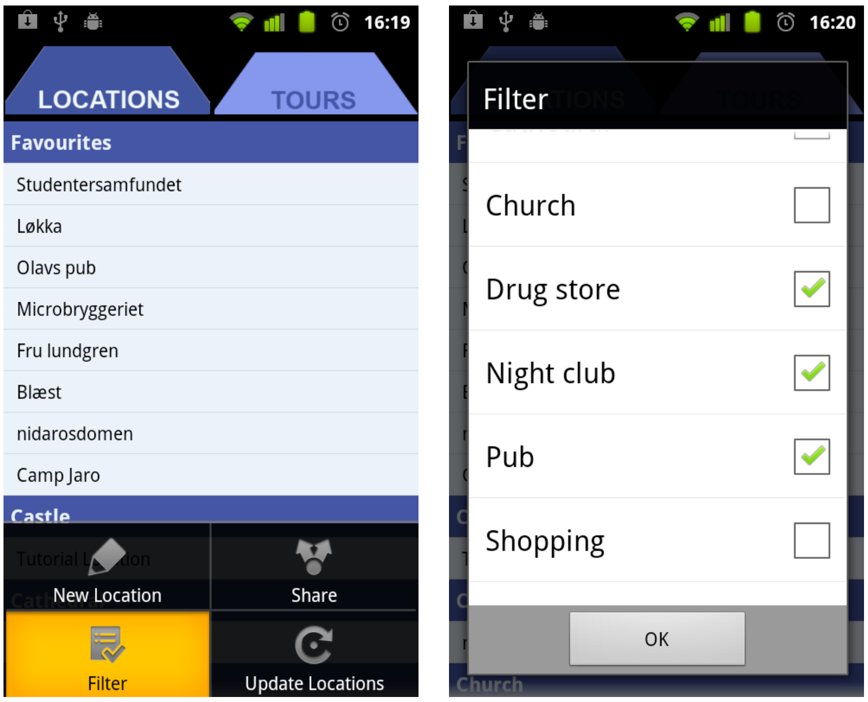
Britania pianobar

Blæst



Filtering your locations. Now we jump back to the screen shown to the right in figure D.1, to show how to suit the location list for yourself. Figure D.7 displays, to the left, the button for opening the filter list, which is displayed to the right. In this list you can check off the categories you would like to have displayed in the location list, including the Favourite category. Pressing the OK button will get you back to the location list, containing only the categories corresponding to what you checked in the filter.

Figure D.7: The button to open filtering(left) and the filtering list(right)


16:19

LOCATIONS TOURS

Favourites

Studentersamfundet

Løkka

Olavs pub

Microbryggeriet

Fru lundgren

Blæst

nidarosdomen

Camp Jaro

Castle

New Location

Share

Filter

Update Locations

16:20

Filter

Church ☐

Drug store ✓

Night club ✓

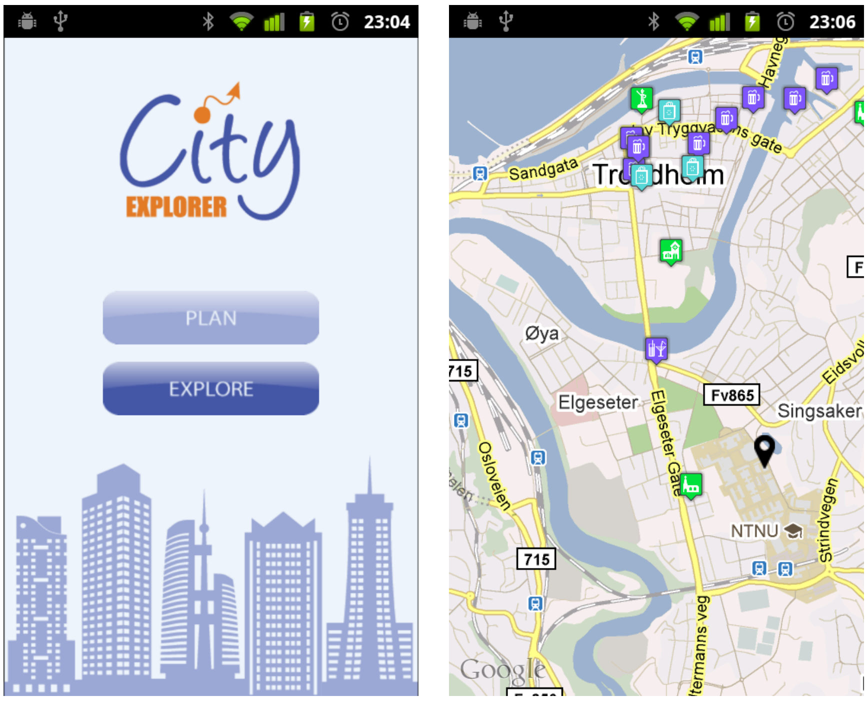
Pub ✓

Shopping ☐

OK

Church

Figure D.8: The opening screen with(left) and the explore map(right)



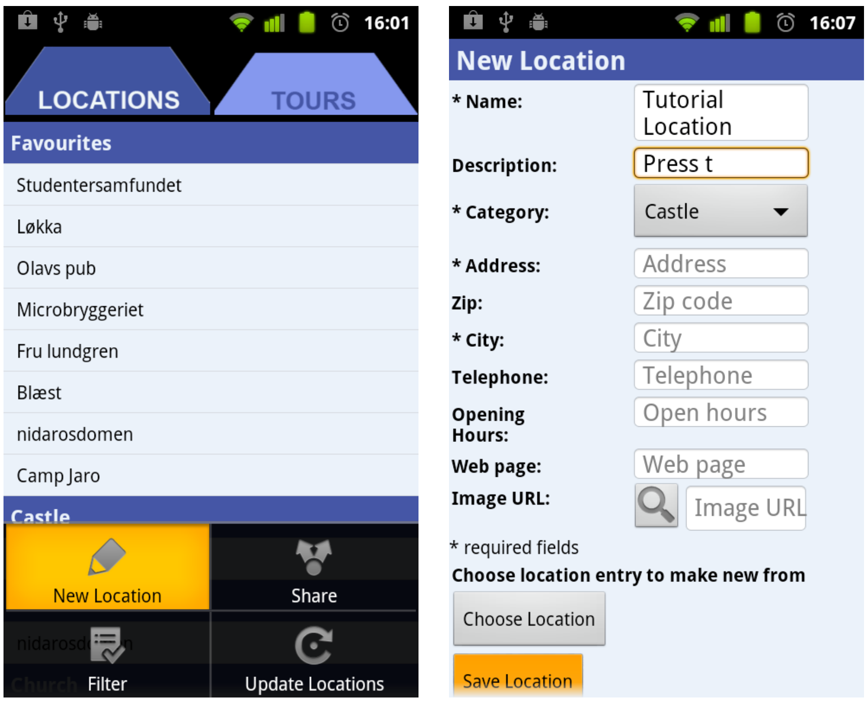
The "Explore" button. And now, as mentioned in the beginning of this section, we will explain the function of the "Explore" button in the opening screen of the City Explorer. It is quite simply the button you would like to press if you find yourself downtown and bored one day, and would like to check out the sights and shops in your close vicinity. Figure D.8 shows, to the right, the result you get when you press the "Explore" button; a map showing locations withing a 500 meters vicinity. You can press the individual marks on the map to get to the details screen for the corresponding location.

D.2 Tailoring

This part of the User Manual involves the users ability to tailor their own sights and experiences in a city, using the City Explorer on Android.

It covers the more advanced aspects: favorizing locations, creating new locations, creating tours and adding/deleting locations from them aswell as setting timetables for your tours.

Figure D.9: The "New Location" button(left) and the location creation(right)



16:01

LOCATIONS TOURS

Favourites

- Studentersamfundet
- Løkka
- Olavs pub
- Microbryggeriet
- Fru lundgren
- Blæst
- nidarosdomen
- Camp Jaro

Castle

New Location | Share | Filter | Update Locations

16:07

New Location

* Name:	Tutorial Location
Description:	Press t
* Category:	Castle
* Address:	Address
Zip:	Zip code
* City:	City
Telephone:	Telephone
Opening Hours:	Open hours
Web page:	Web page
Image URL:	Image URL

* required fields

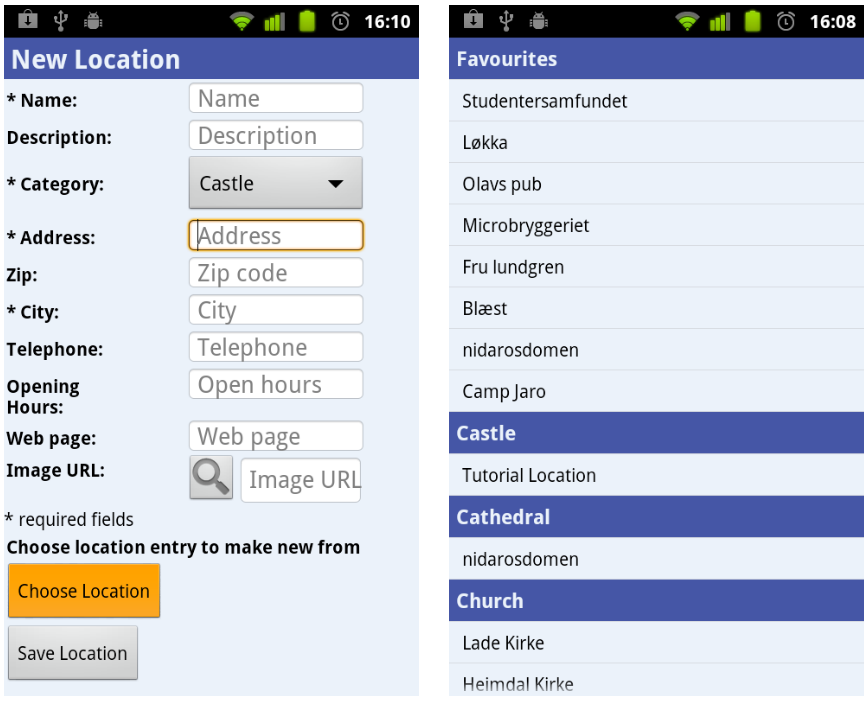
Choose location entry to make new from

Choose Location

Save Location

Creating locations. If you find yourself sitting in a beautiful cafe(for example Edgars cafe in Studentersamfundet in Trondhjem) which, for some reason, has not yet been added to the City Explorer, you might want to add this to your list of locations. Figure D.9 shows how you would go about to do this.

Figure D.10: The "Choose location" button(left) and the location selection(right)



16:10

New Location

* Name:	Name
Description:	Description
* Category:	Castle
* Address:	Address
Zip:	Zip code
* City:	City
Telephone:	Telephone
Opening Hours:	Open hours
Web page:	Web page
Image URL:	Image URL

* required fields

Choose location entry to make new from

Choose Location

Save Location

16:08

Favourites

- Studentersamfundet
- Løkka
- Olavs pub
- Microbryggeriet
- Fru lundgren
- Blæst
- nidarosdomen
- Camp Jaro

Castle

- Tutorial Location

Cathedral

- nidarosdomen

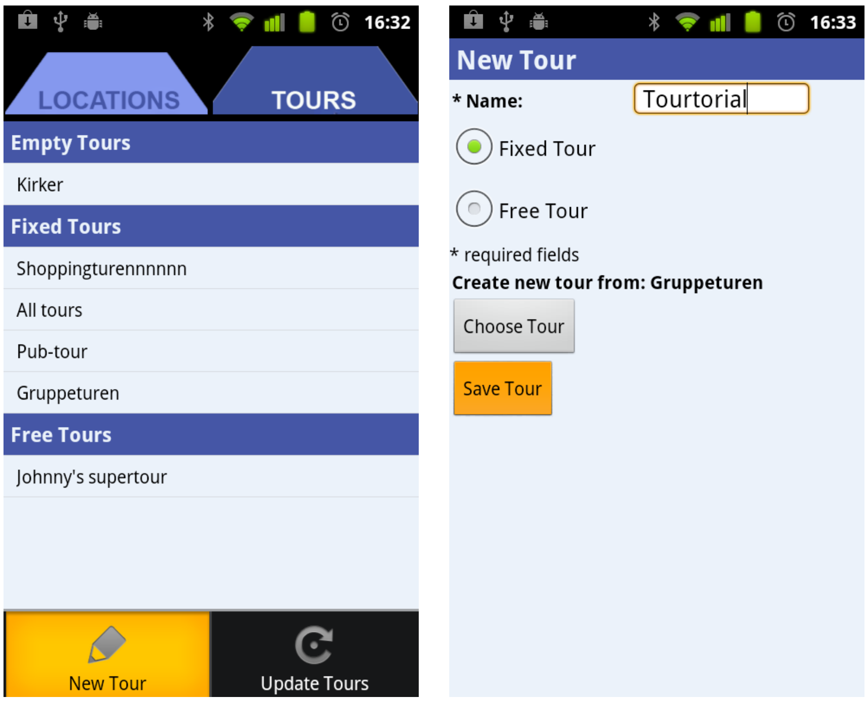
Church

- Lade Kirke
- Heimdal Kirke

As you can see, the menu in the location list contains a "New Location" button, which brings you to the creation screen. On this screen you can fill in the information regarding the location, and save it in your database. Remember to assign your location to a suitable category, by using the drop-down menu at the "Category" field.

If the location you want to create differs only a little from a location you already have, you would be smart to use the "Choose location" button in the creation screen, as displayed in figure D.10. Doing this will show you your list of locations, as seen to the left in figure D.10, choosing one of them will fill the information contained in that location into your creation fields.

Figure D.11: The "New Tour" button(left) and creation screen(right)



16:32

LOCATIONS | TOURS

Empty Tours

Kirker

Fixed Tours

Shoppingturennnnnn

All tours

Pub-tour

Gruppeturen

Free Tours

Johnny's supertour

New Tour | Update Tours

16:33

New Tour

*** Name:** Tourtorial

[✓ Fixed Tour]

Free Tour

* required fields

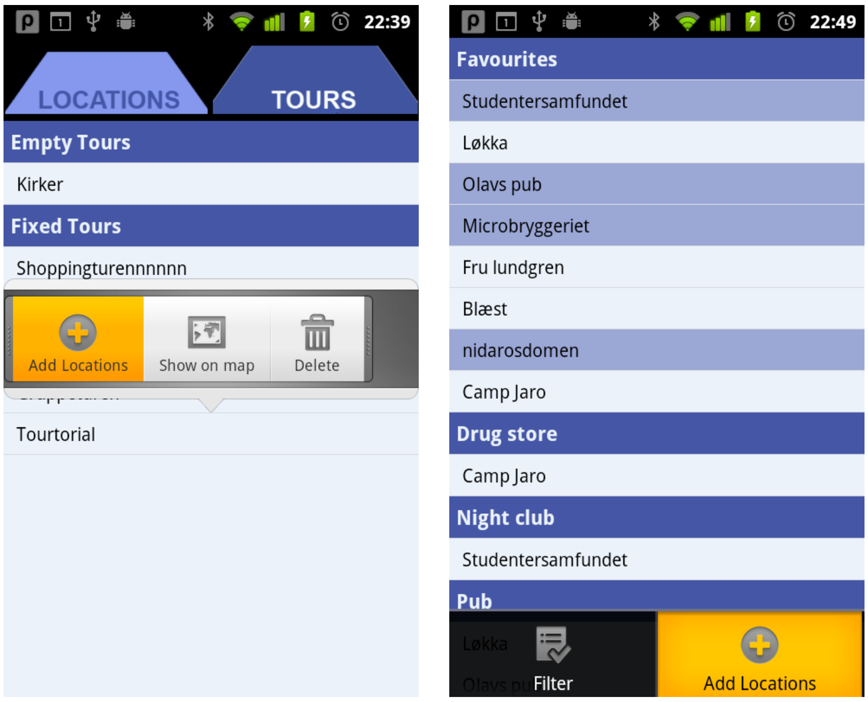
Create new tour from: Gruppeturen

Choose Tour

Save Tour

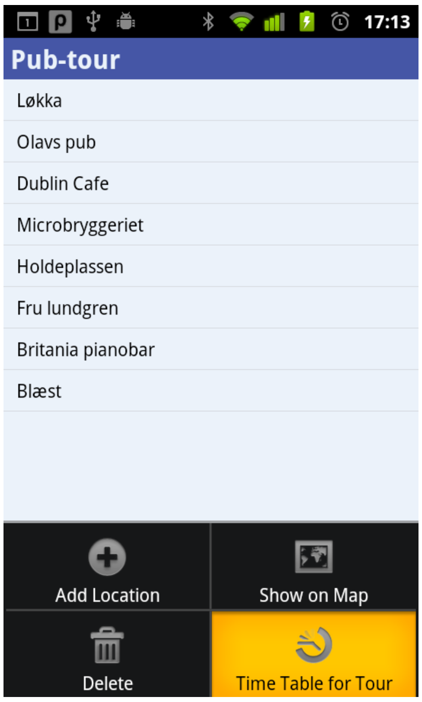
Creating your own tour. You might find that none of the tours that exist does not really suit yourself, or you might want to create a tour based on your own experiences for someone else to enjoy. Figure D.11 shows you how this would be done. To the left of the figure you can see the button for opening the creation screen to the right. Fill inn the "Name" field, choose whether you wish a fixed or free tour, press save and voila you have created a new tour.

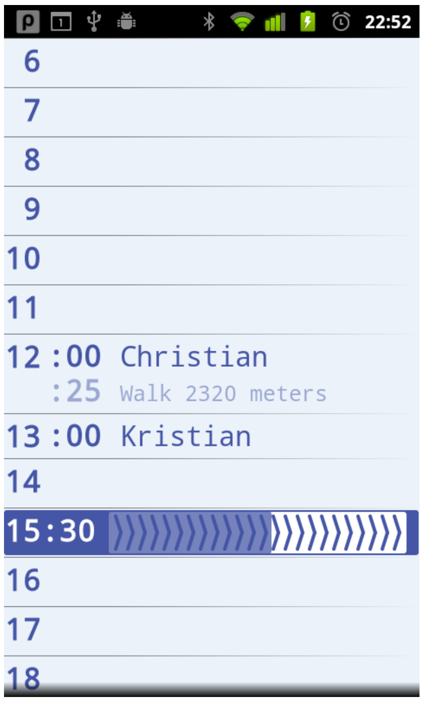
Figure D.12: Button to add locations to a tour(left) selecting locations for adding(right)



However, this tour you've just created is currently empty. How you go about adding locations to your newly created emtpy tour you can see on figure D.12. Starting off in the tour screen, you can longpress your tour to get a menu of actions on that tour. To the left in figure D.12 you can see this menu with the "Add locations" button highlighted. The list to the right then appears, containing all the locations; here you can mark which locations to add(marked locations are highlighted in light blue) and add them to your tour.

Figure D.13: Accessing timetable(left) and creating the timetable(right)

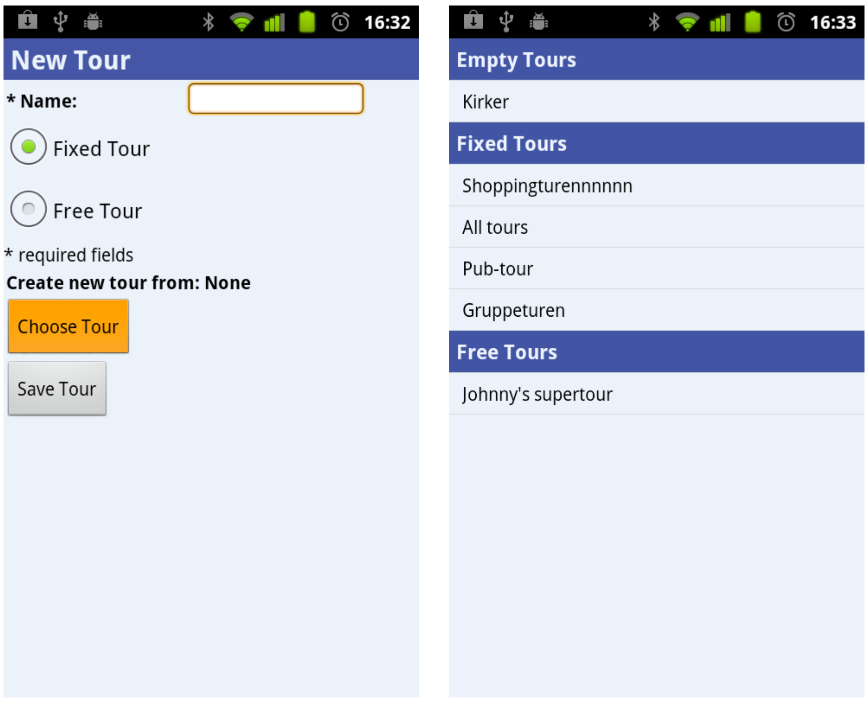




Setting a timetable for your tour. When inside the tour, you can access the timetable, as shown to the left in figure D.13. To set times for each of your locations, you have to touch the hour you want to set it to, and drag your finger across the timetable to the correct minute, as shown to the right in figure D.13. A list of the locations in your tour will appear and you must add them to the timetable(you must put them into the table in the correct order).

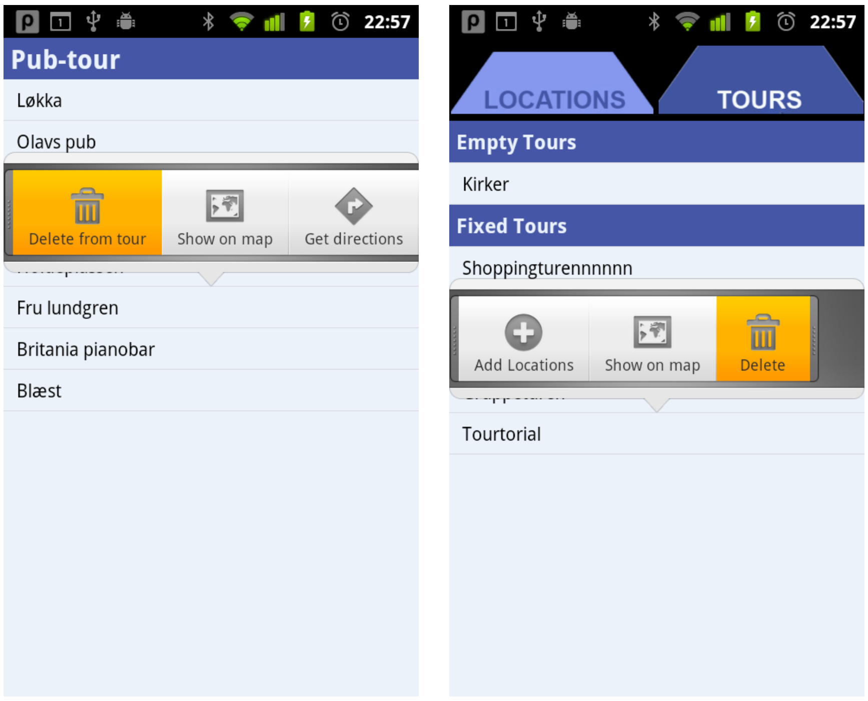
The City Explorer will calculate the time you need to walk between locations. The hour you need to leave a location to get to the next depends on the walking distance and the time you've set for arriving at it.

Figure D.14: The "Choose tour" button(left) and the tour selection(right)



Creating a new tour based on an old one. If you want to create a tour based on another tour you can use the "Choose tour" button in the creation screen, as seen in figure D.14. The locations contained in the tour you choose in the right screen in figure D.14 will be loaded into your new tour.

Figure D.15: How to delete locations from tours(left) and how to delete tours(right)



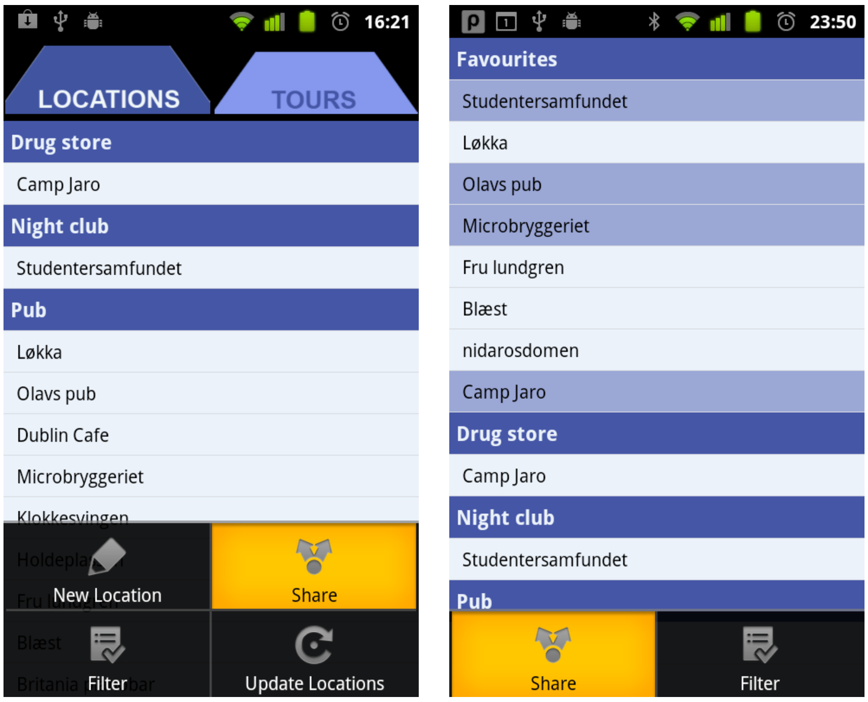
Deleting locations from your newly created tour. The process of removing locations from your tour is shown to the left in figure D.15, and deleting tours from your phone's memory is shown to the right.

D.3 Sharing & Updating

This part of the User Manual involves the users ability to share among themselves the things they might enjoy to see in a city, using the City Explorer on Android.

We will go through how you go about sharing a location with friends, updating locations from a server and updating tours from a server.

Figure D.16: Opening the sharing of locations(left) and location selection for sharing(right)


16:21

LOCATIONS TOURS

Drug store
Camp Jaro
Night club
Studentersamfundet
Pub
Løkka
Olavs pub
Dublin Cafe
Microbryggeriet
Klokkesvingen
Holdepla
Fru lu
Blæst
Britania
bar

New Location | Share
Filter | Update Locations

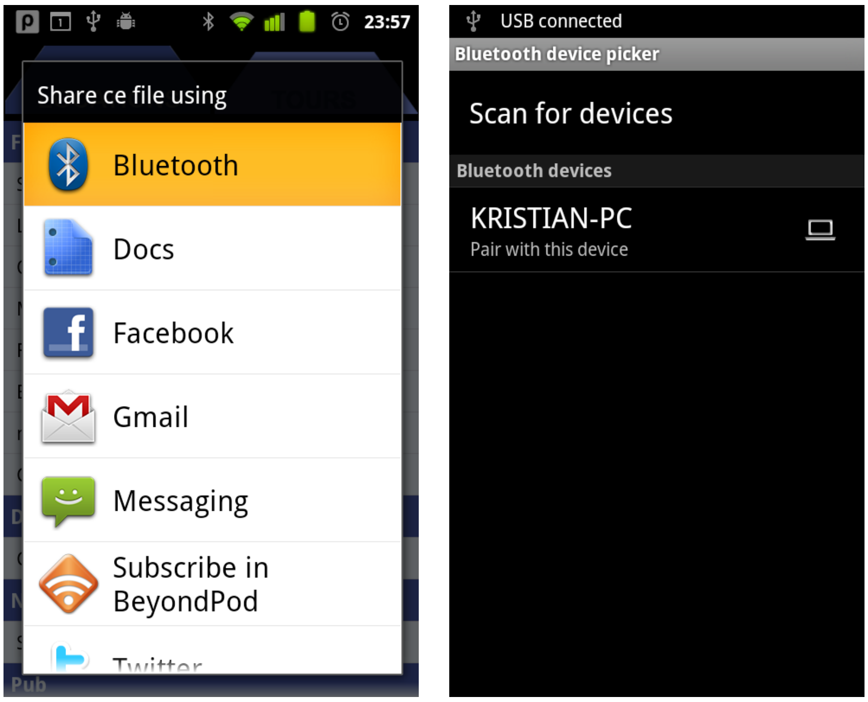
23:50

Favourites
Studentersamfundet
Løkka
Olavs pub
Microbryggeriet
Fru lundgren
Blæst
nidarosdomen
Camp Jaro
Drug store
Camp Jaro
Night club
Studentersamfundet
Pub

Share | Filter

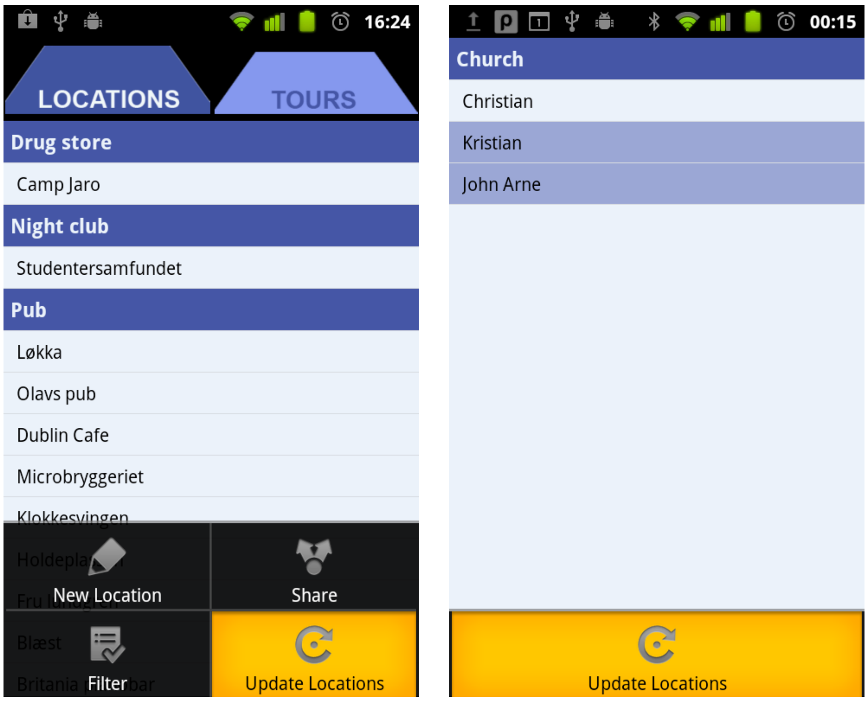
Sharing locations with a friend. The process of sharing locations with a friend is shown in figure D.16. When in the locations screen, you press the share button, as shown to the left in figure D.16, to access the list of locations to the right. Select the locations you wish to share, open the menu and press the share button.

Figure D.17: The list of ways to share locations(left) and list of devices available for sharing(right)



This will prompt the list you can see to the left in figure D.17, where you have several means of sending the location information to someone. However, the recommended ways are bluetooth and email. The next screen shows a list of devices you can send to, so select your friends android device and he will receive it.

Figure D.18: The list of ways to share locations(left) and list of devices available for sharing(right)

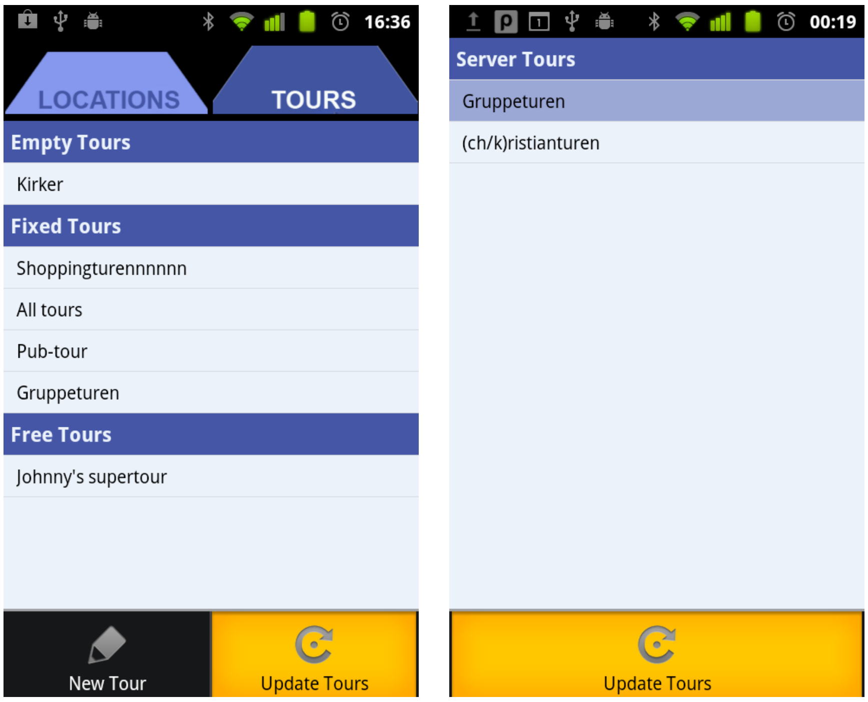


Drug store
Camp Jaro
Night club
Studentersamfundet
Pub
Løkka
Olavs pub
Dublin Cafe
Microbryggeriet
Klokkesvingen
Blæst

Church
Christian
Kristian
John Arne

Updating locations from server. If you have a look at figure D.18 you can see that the process of updating the locations stored on your phone with information stored on an online server is pretty strait forward. The first screen shows the accessing via the locations menu, and the next screen shows the selection of which locations to download from the server.

Figure D.19: Updating tours



Updating tours from server. This is the same process as when you update locations from a server, only accessed through the tour menu, as shown in figure D.19.

Appendix E

List of Figures

This appendix lists all of the figures seen in the report.

List of Figures

Appendix F

List of Tables

This appendix lists all of the tables seen in the report.

List of Tables

Glossary

Activity From the Android reference: An activity is a single, focused thing that the user can do. (...) In other words, it is a component of the application responsible for exposing functionality and for processing user input. 58

API Application Programming Interface. 39

Extreme Programming Extreme Programming is a software development methodology. See reference [8]. 13

GUI Grapical user interface. The interface shown to the user that enables the user to interact with and be informed by the program. 48, 50, 52

http Hypertext Transfer Protocol. HTTP is the foundation of data communication for the World Wide Web. `http://en.wikipedia.org/wiki/HTTP`. 70

IDE Integrated Development Environment. 14

php PHP is a general-purpose scripting language originally designed for web development to produce dynamic web pages. `http://www.php.net`. 70

PoI Point of interest is a location with attached information. 8, 48, 52, 58

SCRUM SCRUM is an iterative, incremental framework for project management. See reference [7]. 13, 101

SDK Software Development Kit. 14

SQlite SQlite is an embedded relational database management system. `http://www.sqlite.org`. 70

tour A tour is a set of one or more PoIs. 7, 25–27, 43–45, 48, 51, 58, 89, 90

VM Virtual Machine. 14

Bibliography

[1] "Google inc. android developers reference." http://developer.android.com/reference/packages.html. [Accessed 27-February-2011].

[2] J. Floch, "A framework for tailored city exploration." Paper to be published in the proceedings of Third International Symposium on End-User Development (IS EUD)., 2011.

[3] "Ubicompforall. ubiquitous service composition for all users." http://ubicompforall.org. [Accessed 13-May-2011].

[4] "Wikipedia. java (programming language)." http://en.wikipedia.org/wiki/Java_(programming_language). [Accessed 14-May-2011].

[5] "Sintef itc." http://www.sintef.no/. [Accessed 27-February-2011].

[6] G. M. Marakas, *SYSTEMS ANALYSIS & DESIGN an active approach*. McGraw-Hill, 2 ed., 2006. [international edition].

[7] "Scrumalliance - an innovative approach to getting work done." [Accessed 14-May-2011].

[8] "A gentle introduction to extreme programming." http://www.extremeprogramming.org/. [Accessed 27-February-2011].

[9] "Wikipedia. apache subversion." http://en.wikipedia.org/wiki/Apache_Subversion. [Accessed 14-May-2011].

[10] "Sqlite." http://www.sqlite.org/index.html. [Accessed 13-Mars-2011].

[11] "Facebook." http://facebook.com (Log in required). [Accessed 13-May-2011].

[12] "git - the fast version control system." http://git-scm.com/. [Accessed 6-April-2011].

[13] "Wikipedia. work breakdown structure." http://en.wikipedia.org/wiki/Work_breakdown_structure. [Accessed 7-April-2011].

[14] M. Fowler, *UML Distilled - A Brief Guide to the Standard Object Modelling Language*. Addison-Wesley Educational Publishers Inc, 3 ed., 2003.

[15] "Google inc. pie chart showing distribution of active android os version on the android market in february 2011." http://developer.android.com/resources/dashboard/platform-versions.html. [Accessed 13-March-2011].

[16] "Google i/o 2010 - android ui design patterns." [Accessed 7-March-2011].

[17] J. G. Raghu Ramakrishnan, *Database Management Systems*. McGraw-Hill, 3 ed., 2002.

[18] "Wikipedia. bluetooth technology reference." http://en.wikipedia.org/wiki/Bluetooth. [Accessed 12-May-2011].

[19] "Ancal." http://code.google.com/p/ancal/. [Accessed 14-May-2011].

[20] "Wikipedia. comma-separated values." http://en.wikipedia.org/wiki/Comma-separated_values. [Accessed 13-May-2011].

[21] "W3schools. introduction to xml." http://www.w3schools.com/xml/xml_whatis.asp. [Accessed 14-May-2011].

[22] G. Paller, "Plug-in." http://www.mylifewithandroid.blogspot.com/2010/06/plugins.html. [Accessed 11-May-2011].

[23] "Wikipedia. international system of units." http://en.wikipedia.org/wiki/International_System_of_Units. [Accessed 13-May-2011].