Independent Project - IT Systems, 15c
Master Programme in Sociotechnical Systems Engineering
Uppsala University
2017-06-02

UPPSALA
UNIVERSITET

Final thesis

The CityWalks project

Under supervision of:
Dave Clark, *Course Director*
Anton Axelsson, *Assistant*
Davide Vega D'aurelio, *Assistant*

Presented by:
Markus Elfving
Mathias Fejes
Johanna Fyrvald
Ilkka Mansikkamäki
Natalie Poli
Saranya Silawiang
Malin Sjöberg
Teddy Wickström

Abstract

This thesis is the final part of the independent project for the Sociotechnical Systems Engineering program at Uppsala University. The project's purpose was to develop a system in form of either a web page or a mobile application. The requirements for the system was that it needed to have a social component and consist of both front-end and back-end coding. The team consisted of 8 persons and there were many different ideas on what kind of application to develop. After the first week it was decided that all team members liked the idea of making a map-based application and finally it resulted in developing an application that could provide the users with routes for walking created in different ways. After specifying the details, it was decided that the application would fit best for city environments, since it would focus on town attractions, parks and such, and with that in mind the name grew to be CityWalks. The team was divided into two main groups which purpose was back-end development and front-end development. Besides that, there were roles for management, usability, communication and security which roles were split up between the group members. In this way, the result has been creating an innovative application with nice design and functions that have the potential of attracting a wide group of people. The system is not yet ready to release for public or commercial use but a stable ground for a popular idea has been build up and there is room for further development. The development of the CityWalks system has given all of the group members a very good experience for similar work in the future.

Table of content

1. Introduction

The concept of fitness and the knowledge of physical activity, as an important part of staying healthy, has been present within the human civilizations for thousands of years. Over the centuries, the practice of physical activity for exercise have changed a lot and taken many different forms. [1] In the modern age, much due to the improvements in technology in the 20th and early 21st century, there is a huge fitness industry that offers many different technical solutions that aims to help with personal and social physical exercise.

One of the latest trends is to record the exercise, with wearable sensors, for a clear overview of the exercise as well as for comparing with earlier exercises to show the improvements. Another part of this trend is to share this information on social platforms, for reasons such as to get motivation from other people and/or to compete with people doing similar activities. Today, many scientists warn us about sitting too much and the worldwide increase in overweight and obesity, that in turn account for about 60-80% of new cases of type 2 diabetes which kills millions of people annually. [2] [3] New technologies that help and motivate people to exercise can therefore be lifesaving. Besides the positive side of such technologies, they are also partly blamed for the increased number of people diagnosed with the eating disorder Orthorexia. People suffering from Orthorexia gets obsessed with exercise and eating healthy, and the rise of the disorder is often connected to the new phenomena of constantly seeing images and data of other people's exercises. [4] So there is a dilemma here, the new exercise technologies with social features may both give important help as well as bringing harm to their intended users.

With this information in mind, we believe that the mobile application CityWalks could provide a new and innovative way for people to get their important exercise and at the same time include social features that are not stressful or harmful. With CityWalks, users can find good walking routes in the area, create custom routes from their preferences or record and share their own favorite routes. Last but not least, CityWalks does not focus on stressful content such as how often the users are walking or how long and fast they are walking.

1.1. Overview

This report will present the work made for the independent project in the Sociotechnical systems engineering program at Uppsala University in spring 2017, which have consisted of developing a mobile application that handles user accounts and have social components. The report will focus on the planning and outcome of the project, the development of the application created, as well as the work that concerns management, programming and communicating. Furthermore, challenges changes in the work and failures will be presented. Much focus is given to the user perspective, the usability of the system and the socio-economic aspects of the application. The goal has been to create a system that is easy to use, attractive and that could have the potential of being successful on the commercial market. This have resulted in an application called CityWalks.

1.2. Purpose

The CityWalks application is developed for people in some way appreciating physical exercise and walking, mainly for the purpose of moving, get fresh air or experience a city. The purpose of the system is to provide functions for finding walking routes, either by letting the application create routes by filters, by showing popular routes from other users or by help you share and find routes from friends.

1.3. Decision of application idea

The decision process of which idea to work on for the project started the first week with a handful of ideas of different kind of systems. After discussing in the group and having a couple voting processes the choice fell on making a mobile application that could provide the user with different kind of walking routes.

The idea developed to create a map-based application for the purpose of walking in city environments. The application can be used partly for visitors in a new town that wants suggestions of routes either for getting some fresh air or to experience the city and its attractions. It can also be used for citizens in a town that just wants suggestions on a good and pleasant walking route, with the purpose of exercise or just strolling. The routes can

either be self created by choosing different filters or a preferred distance, record new route based on location position or routes can be found from friends or public top shared routes.

1.4. What is CityWalks?

CityWalks is a map-based application for mobile smartphones developed for people in some way appreciating physical exercise and walking, mainly for the purpose of moving, get fresh air or experience a city. The purpose of the system is to provide functions for finding walking routes, either by letting the application create routes by filters, by showing popular routes from other users or by help you share and find routes from friends. The user can choose preference, for example they can choose to walk in parks or pass churches and the application will provide a route based on that.

The typical usage and the groups that are predicted to be the main users for the application and have been the focus for developing is divided in three parts. These groups are stated as examples on when the system could be handy.

The first example would be for people in an unfamiliar town visiting a friend that is currently at work etcetera and that wants to take a walk for some kilometers, but doesn't really know where. Then they could get suggestions on routes nearby. Another case is for people on a vacation trip in a new town and want a good tour that passes monuments etcetera. They can create that kind of route with help from the application by choosing preferences for what they want to see. Alternatively, they can use a public route that have positive comments or rating. A third example of possible usage would be for citizen in a town that wants to find a nice walking route for a specific distance using public ones nearby or routes that could be seen that friends have walked and maybe talked well about.

All these needs can be fulfilled by the main functions implemented in the CityWalks application, which consists of "Create route" and "Record route". Together with user handling functions, friend's functions and social components such as chats and comments this is what builds the system. All these functions will be specified in this report.

2. Socio-economic aspects

2.1 Target users

The intended users of the system are mainly visitors or business travelers in new cities who wants to explore the city. But also intended for local people who live in the city in need of walking routes suggestions. Any social group and group of ages should be able to benefit from the system and therefore aims to fit a wide range of users. The users would be interested in CityWalks since it is an innovative application with another purpose than the already existing fitness applications. The user gets a lot of opportunities to create their own personal routes and can therefore customize the way they use the application to fit their individual needs.

2.2. Competitor analysis

Today, there is a huge market for fitness applications, were applications like Runkeeper Myfitnesspal and Runtastic have millions of users and are valued to and acquired by some of the biggest sports companies for millions of dollars, see *Figure 2.1*. [5] These applications have been popular for many years and the numbers of users are steadily growing. By looking at this market and researching for alternatives we came up with a missing application and a market opportunity. We discovered that when it comes to walking in cities and discovering new sights etc, there are not as many competing and established applications on the market. That is why we chose to focus on CityWalks. We saw the opportunity to create an app for people who are less interested in running and fitness and are more interested in walking in cities on walks according to their preferences and to discover new routes to walk.

	**Runkeeper** ● 50 million users ● Acquired by ASICS for $85 million in 2016
	**Endomondo+MyFitnessPal** ● 20+100 million users, ● Acquired by Under Armour for $560 million in 2015
	**Runtastic** ● 25 million users ● Acquired by Adidas for $260 million in 2015

Figure 2.1. The leading Fitness applications in today's market.

2.3. Social aspects

The users of CityWalks will be able to use the application in different social ways if desired. Users can add their friends that has downloaded the application and created an account as well as see their walked routes. A user can also share their own walks in the CityWalks community and then browse among other users routes in form of different lists, such as nearby routes and top routes. In the further development the users should also be able to sign up, sign in and interact with other users through their social media accounts, such as Facebook. This things will hopefully fulfill a need of being social when using the application and being able to interact with friends and other users. The sharing and adding friends functions are optional, so if the users want to use the application only for personal purposes that is possible as well.

2.4. Economic aspects

In the aspect of making profit from the CityWalks application there are many possible alternatives. In this part two main ways of how the application could generate money will be described.

2.4.1. Sponsored content

One alternative would be that companies and businesses can sponsor different content in the functions of the application to be more visible and attract users of the application as

customers in different ways. That could be implemented in many ways, but one suitable for the CityWalks system would be to offer companies and businesses to make their operations extra visible. For example, the brand logotype of sponsored companies could be shown on the map, so that the users will see that place stand out from others. Also when creating routes and filter preferences in different ways, the sponsored places could have an own field, or be prioritized in the search result. An ice cream shop could for example pay for the user to get an offer for their ice cream or pay for the users to more often walk by their location.

2.4.2. Premium content

Another possible alternative is to have certain features only available in an upgraded version of the system that can be purchased for a one-time fee or a monthly payment. This would be in line with a Freemium pricing strategy. This will create a need for the users who use the application frequently to be able to use the extra features or functionalities and purchase the upgraded version. In the CityWalks application this could be in the form of certain functions such as the “create route” function, to have a maximum amount of generated routes or to simply charge for the application to be downloaded in App Store or Google Play.

3. Method

3.1 Working methods and processes

When working with the project the method Scrum [6] has been chosen, which is a methodology used for process in project working. The Scrum method is used by creating so called sprints for every small part that have to be done in the project. The sprints are handed out to one or many owners, which will be the ones responsible for them. Every sprint is made so that the responsible person can focus on delivering the assignment that is specialized in the sprint. The goal is to deliver the completed assignment in the decided time. A scrummaster is chosen to be responsible for the work with Scrum, so that everyone knows their sprints and to manage the process. In this project the scrum master has been the same person as the project manager. With daily meetings the team can communicate problems in the process with the sprints that could possible prevent them delivering in time. At this meeting the sprints which time are out is checked upon there completement and crossed of if finished.

The method was chosen to organize the work in a structured way and to avoid the scenario where the largest part of the burden remained to the last weeks. The Scrum method was not something that any of the members had been working with before so it was a learning process on how to plan the work by it. This was also the reason that all the work did not follow the Scrum plan. However, it was a good way to hand out assignments for everyone and for all members to know what they were suppose to do.

Looking back now, it could have been spent more time on learning the work with Scrum and to be stricter in following the method in all the work. In this project much of what was decided on meeting and in sprints were gone from and instead the focused shifted depending on where resources was most needed for the moment. That caused for a large part of the group working on the same thing during the same time and was not that efficient. The split up between working on deliverables and working on the system worked better however, so in many times the group was divided into two parts working parallel.

Besides the everyday meetings, each monday has started with a larger group meeting where all that concerns the project have been discussed and decided. On this meeting there have also been written protocols for all members to take part of.

3.2 Team structure

Initially, the roles were set based upon the team members preferences. The roles were decided based on that there would be one project manager and all of the team members would be assigned one developing role, front-end or back-end as well as one minor role. In the beginning, we realized that we needed a few more people on back-end to get the functionalities working, and a few less on front-end, since we would focus on the design part more later on in the project. The front-end developers focused on creating mock-ups and think about color scheme etc. in the beginning. During the project the roles changed back and forth, we realized that we needed a person in between front- and back-end as well, since it was crucial to connect the front-end and back-end design. The roles were assigned according to the following scheme:

Role	Responsibility	Team member(s)	Required skills & tools
Project Manager	Project planning, Trello, deadlines	Markus Elfving	Project mgmt, Trello
Communications Manager	Presentations, reports	Saranya Silawiang	Microsoft Office - PowerPoint, Word
Usability team	Usability design and testing	Johanna Fyrvald Saranya Silawiang	Design-thinking
Front-end Developers	Set up client side	Markus Elfving Mathias Fejes Johanna Fyrvald Natalie Poli	Ionic, HTML, CSS, JavaScript

Back-end Developers	Set up server side	Malin Sjöberg Saranya Silawiang Teddy Wickström	Node JS, MongoDB, JavaScript
Combined Front & Back-end	Connection between client and server	Ilkka Mansikkamäki Mathias Fejes	Combined front-and back-end
Technical manager	Set up computers, solve technical issues	Teddy Wickström	Technical skills, Installation of programs
Security Expert	Handle security & privacy issues	Ilkka Mansikkamäki Mathias Fejes	Security & privacy
Database designers	Design the structure of the database	Teddy Wickström Malin Sjöberg	MongoDB

Figure 3.1. Division of work and team roles

3.3 Project plan

This project plan was created initially to structure all the activities that needed to be completed during the project. The project plan covers what team members are involved in the activity, the dependencies between activities, the estimated time for each activity and the estimated resources needed. The activities were visualized by a GANTT chart, see *Figure 3.2*, together with a description of the various activities. The milestones of the project are marked in bold in the GANTT chart.

The plan was seen as a living document during the project and we realized that the differences between the planned and actual time each task took was quite significant. In the beginning, with the project start up and the planning we were following the plan. Although, as the time went by a few parts got delayed. For example, in the design part, the setup of the initial front- and back-end framework as well as making the connection between front- and back-end work took much longer than estimated.

Figure 3.2. GANTT-schedule over the project activities and its dependencies

In this part the initial main activities important for the progress of the project will be presented together with its planned time for execution, its dependencies and the involved members. We followed these milestones to make sure that we specified each main activity and what was included in each step. The GANTT-schedule, is attached to visualize the activities and their relation to each other, and below these activities are described in more details. In the time scheduling, the time for learning in each area is included. We completed all of the milestones described below, although the learning time and time estimated for the different activities varied from the GANTT-schedule.

Project startup

Decision making

The decision making includes generation of project ideas and the decision of what idea to proceed with. Different technical platforms and programming languages will also be discussed and based on what platform will serve the purpose of our idea best a decision will be made. We will also specify the requirements and features of the application.
Main responsibility: All

Set up conditions

The set up conditions includes installation of computers and software as well as setup the office environment.
Main responsibility: All

Project planning

Create project plan

The project plan will be written and it will include a Gantt scheme as well as a description of the activities. The Gantt scheme will show the estimated time for each activity and the dependencies between them. The description of the activity will also include the estimated resources for the activities.
Main responsibility: Project manager

Design

Graphical profile and logotype

Decide the graphical profile of the application, which includes the color scheme and the font as well as creation of an official logotype.
Main responsibility: Front-end developers and minor design role.

Create Mockups

Create Mockups for the design of application. The mockups will be created through drawing by hand as well as some of the pages in the mockup-tool JustInMind.
Main responsibility: Front-end developers

Build front-end framework

The building of the front-end framework includes creating the official frame of the application and a common project in GitHub. The front-end code and design is also included in this part.

Main responsibility: Front-end developers

Build back-end database

The back end database will be built in parallel with the front-end framework. A database with for example user information will be designed. The integration between frontend and backend will be executed in every weekly sprint.

Main responsibility: Back-end developers

Functions

The included roles in all the functions is primarily Front-end and Back-end developers. The functions that are needed for complete the targets with CityWalks application are listed below:

Map API integration

Implement google Map API in the application interface. This will result in an application with a maps on several of the pages. Furthermore google places- and direction services will be used for the functionalities. The map is also used when recording a route, creating a route from preferences and when creating a random route.

User functions

The user functions include creation of login/logout page for users, the storage of users in a database and the user settings.

Record route

The function of recording a route when walking it through GPS will be built. This function will contain comment, recommend and save route.

Comment, recommend and saving routes

The function of comment, recommend and saving routes will be built and be visible in Record route and when the user choose to walk Top Routes.

Social media integration

The social media integration includes integration with Facebook and potentially Google+. The user should be able to choose the option to log in with Facebook and to share their routes on Facebook.

Create route

Create a function where the users will be able to create their own routes based on their preferences through using different filter options. The user will then bypass the chosen options during their walk. The user will be able to choose the walking level after ambition. The walking level is translated to different distance intervals.

Random route

With an algorithm be able to randomize a route from your current position and after the ambition of walking level. The user will be able to see the randomized route on the map and directions from start to end. The walking level is translated to different distance intervals.

Usability

User profiling & use cases

Create user profiles of the intended users as well as specific personas of each intended user group. Use cases of scenarios in the application will also be created.
Include roles: Usability expert

Create usability tests

Create usability tests that will test the system. The tests will include scenarios that the user needs to complete.
Main responsibility: Usability expert

Usability assessment

The system will be tested by 5 team members of our project group. The intention is to test the system on 5 external non-members of the project group that fits well into the intended

user of the application. These users will for example be family members and friends of the project members.

Main responsibility: Usability expert

Close the project

Final presentation

In the last week of the project there will be a presentation of the finale system, which means that all parts of the project should be finished by then. The presentation are given in a seminar together with the other teams in the course. The presentation will be 45 minutes long and consist of 30 minutes of content, 10 minutes of questioning by opponents and 5 minutes for other questions. This activity is mainly coordinated by the communication manager but will include all members in the team.

Finalize thesis

We will work on the thesis during the whole project and in the last part of this course we will finalize it. The communication manager will coordinate all the members of the team who will work on the final thesis together.

3.4 Reflections and outcome of project plan

As mentioned above much of the scheduled plan came to differ a lot from the actual performance. Here an analysis will be presented on how the work have differed from the plan and what was the outcome of it.

The project startup and planning worked out quite smoothly, the first weeks was dedicated to work on the idea and to set up the condition, together with working on the graphical profiling and set up mockups for how the application should look like.

The thing that took a lot more time than planned was the developing of a back-end database and a front-end framework. This was mainly because all the systems and languages used for coding was new to all members, why it took time to learn during all the process on how to build both frameworks and databases. Also the functions demanded time in the same way to be able to implement, and often the group member working on a

specific function needed to spend some day just watching videos or guides on how to create a desired function.

The plan was to connect the front-end framework with the database in week 17, but this took a lot more time since all the database function were not finished and the API connection was not completely set up.

Many of the functions planned was not developed in the end. For example, the social media integration and many of the user profile functions was chosen not to be implemented in the application. The reason for that was mainly a lack of time which forced some features to be unprioritized. The completed system with its functions will be presented below in this report.

All the deliverables for the specific areas in the time plan have been followed the time schedule almost completely. This was because there was a clear division between group members working on deliverables and on the system developing. Also the group decided in an early state that all the deadlines for the deliverables should be followed to avoid getting a too large workload in the end of the project.

The usability work and testing followed the time plan pretty much liked planned. The user profiling where the potential users were mapped was made in its own deliverable, and when the application was working in its early state on a mobile phone usability tests were created and the application was tested by usability professional teachers as well as by test users that matched the typical personas for the user group of the system.

Finally, one thing that demanded some extra time and resources was the method for sharing code and work parallel with the same project. This is usually done with the Git system, see more description below, and that's what was used for this project as well. Since no person in the group had worked in larger programming projects before and had not been using Git, this took some time learning and testing with different systems before finding the right way of working. In the last weeks, after connect the database with the front-end code,

the group found a good routine of working with updates on the application locally and share them to the Git project when finished.

4. Requirements

In this part of the report the requirements of the CityWalk's system will be specified based on the group's plan and wishes for what will be implemented. The requirements is set from listing what is important from the main idea combined with the demands given in the course. In the aspect of making profit from the application there is some possible alternatives for the system, as mentioned in chapter 2. The first one is that companies and businesses can sponsor routes and places to see, for example an ice cream shop can pay for the user to get an offer for their ice cream or pay for the user to walk by their location. Another possible alternative is to have certain features only available in an upgraded version of the system.

4.1 Functional features

Login/logout and create a user profile

When a user opens the application they will see a first page where the opportunity to log in with email/password or create a new user profile will be available. Creating a new profile will contain information about the user, like name, username and e-mail. When logging in to the application the user will have the option to sign out.

Show a map with GPS-function

When a user is logged in the user will see a map which will be integrated with a GPS-function. This will happen by implementing a map provider API, such as Google Maps. A map integration is a requirement for this application since all the main function will need a map.

Show routes

The user will be able to see different routes on the Maps sort by rate, distance, creator and so on. The routes will be based upon certain pinpoints on the map. This function provide the user to see all the route created by other users.

Social interaction between users

The user must be able to integrate with other users. They will be able to add their friends and share routes between each other.

Use already existing routes

The user should be able to search between and choose to walk already stored routes created by other users. The search function should have filters where the user can specify the search based on preferences such as desired distance, pinpoints or tourist attractions and create a route based on that. The application should also be able to see official routes in the area and get directions to the starting point of that route.

Comment and rate routes

The social interaction between users should be expanded through make it possible for users to write comments about as well as rate shared routes, based upon their experience while walking a specific route.

User profile page

The users should be able to see a “My profile” page where personal information such as the user’s walked routes, how many kilometers the user have walked and so on.

Social media integration

The users should be able to log in to the application with social media such as Facebook or Google+ instead of creating a new user account and login with email/password. The user should also get the opportunity to share the routes after walking them.

Use an algorithm to create routes

The users could be able to create their own routes based on their preferences through using different filter options. There could be an option to either create a route based on the user’s desired distance, for example 3km, 5km or 10km or create a route based on desired sights along the walk such as pass a church, a café, walk through a park or a specific tourist attraction in the city. An algorithm would be created for this purpose.

Optimization

- Using data from users

The application could be able to create optimized routes based on more detailed data information from the user. The application could also create recommended routes personal for each user.

- Algorithm

The algorithm that is used to create routes could be optimized and more accurate.

Visualization of data

In the user profile page the user could see data in form of graphs such as how many kilometers they have walked every day during a week, a month or a specified period of time.

4.2 Non-functional features

Security and privacy

For the security aspect it is imperative that the users feel safe when typing in their information. Therefore, the system will implement hashed and salted passwords. The system will also use Facebook and Google authentication.

Performance

When a user is interacting with the system the response time should be as short as possible, so that the user feels like the system reacts. Otherwise they might press several times and that should be avoided.

Quality of service

To make sure the quality of service for the system is maintained. The system will have a page with FAQ and a page with contact information.

User-friendly and responsive interface

The user interface should be user-friendly regarding the design, colors and so on to facilitate for the user to make the intended interactions on each page. The user interface should also be responsive and adapt to different screen sizes.

4.3 Requirement changes

During the application development there were some changing on functional features as well as non-functional features. Many of the functions planned was not developed in the

end, the reason is mainly a lack of time which forced some features to be unprioritized. Therefore, since we saw opportunity and solution to make the application more user-friendly we did some details and design changing.

We decided to change requirements when sign in and its now just username, password and password that are required. Show routes is a function where the user can sort by recommendations, date, distance, comments, username and how many user has walked the route. This allows the user to choose what they want to sort and make the application much smoother and more efficient. The user profile page was chosen not to be implemented since our mainly focus in application is to provide routes created randomly or by preferences and therefore not interested of user profile. But for future development there will contain some user profile that include more information about the user, like user's walked routes, how many kilometers they've walked and so on.

When creating the Create Route function we realized that it was much better and easier making two different function, one that create route by preferences and one by distance level, which will be presented below. As we mention before we wanted the application to be integrated with Facebook and Google+, partly for safety reasons and partly for agility but since the time hasn't been enough this function has been unprioritized.

Many systems on the web and in applications has an administrating version or a special login providing these functions. In the CityWalks system however, the application consists of different map and routing functions only and we could not see any need for having admin functions in the framework. Instead, database functions for handling users, routes and attractions that is not editable in the application is made from our developing tools, such as Node and MongoDB, see the system presentation part. Since the content of CityWalks is created of all users and every user should be able to edit its own data, there is no need for an administrator providing data or handling the content. With that said, the people working with the system can be seen as administrators that controls the content of it. This is the same structure as many other communities based systems uses, such as Facebook,

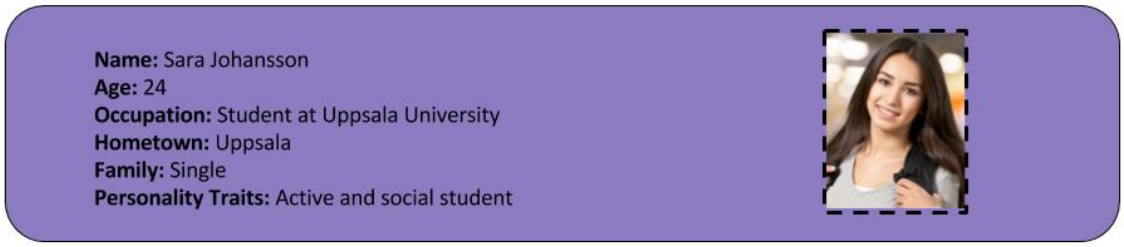
Instagram and Snapchat, which also does not have any administrator accounts, but instead control abuse or overriding of the rules of the community and offer different functions for the users to report such, something that could be implemented in the CityWalks system in a further development as well. Together with the teaching staff we agreed that this was a suitable way for our kind of system, and we also got the input that it could be more safe in a security perspective to avoid implementing too much database handling code in the front-end framework, since that could make attacks and other invasion on sensitive data easier.

5. The user perspective

5.1 Intended users

The CityWalks application aims to fit a wide range of users. Any social group and group of ages should be able to benefit from the system. However, there are three main focuses from the functions that the application provides. These benefits include finding and sharing new walking routes in local towns, finding popular walking routes when temporarily visiting a town and finding city routes with prefered attractions when touristing in a city. The challenge for the CityWalks development has been to make the application easy to understand and use for all potential users, from young to old people and from technical to nontechnical. In this part, some fictive personas which represents typical intended users who might benefit from the application is presented.

Sara, 24 - Student at Uppsala University



- ***Description:***

  Sara is a 24-year-old student at Uppsala University. She is an active student that participates in a lot of activities at different student nations. She is a social person that has a lot of friends and she is always on the run. In addition to that, Sara is also a very good student that aims for high grades and she can spend hours in the computer lab at Ångström programming. In between all the parties, events and hard studies Sara loves to go for walks to get some fresh air and alone time. She almost always walks the same route and lacks inspiration to change her favourite walk. Sara is very active on social medias and brings her smartphone along wherever she is so that she can stay connected with her friends wherever she is.

- ***User objectives:***

  - Find new routes to walk.

  - Connect with friends and get inspired from their routes.

- ***Requirements in order to use CityWalks:***

  - Suggestions of new inspiring routes.

  - Social connection with other users and via social media.

Bertil, 74 - Retired man from Stockholm

- ***Description:***

  Bertil is a 74-year-old retired man based in Stockholm. He lives in an apartment in Vasastan together with his wife Gunvor and they have been married for over 50 years. Despite his age Bertil is very alert and active. He enjoys weekend travelling around Europe with his wife and wants to discover as many new cities as possible. In the different cities Bertil loves to discover the cities by foot and tick the boxes of the most famous tourist sights. Bertil also enjoys to take his wife to more quiet and closer to nature areas such as parks and places outside the most central parts. Bertil is very interested in history and has visited most of the museums around the world, but he is eager to learn more to be able to tell his grandchildren exciting stories. Bertil has a smartphone but he only uses it for the most basic purposes such as calling and texting, he has a few applications that he understands how to use and he is a fast learner.

- ***User objectives:***

  - Get routes based on preferences such as tourist attractions, museums and parks.

- ***Requirements in order to use CityWalks:***

  - Clear buttons and logical user interface.

Gustav, 35 - Travelling Businessman

Name: Gustav Ahl
Age: 35
Occupation: Travelling Businessman
Hometown: Malmö
Family: Single
Personality Traits: Ambitious & hardworking

- ***Description:***

  Gustav is a 35-year-old middle aged businessman from Malmö who works as a management consultant at a leading consultancy firm. He has a hundred travelling days per year and spends many nights in hotels or at friends houses in different cities. He likes to go out for a stroll after finishing the working day and often asks for ideas from colleagues of routes that is walking friendly. Gustav is a very hard working man which leads to that he is very short on time. When he is walking he is therefore often time pressured and sometimes he has to limit the length of his walks to be able to fit it in on the lunch break.

- ***User objectives:***

  - Find nearby routes and specify the length and environment on the routes.
  - Be able to read comments and ratings of those routes.

- ***Requirements in order to use CityWalks:***

  - A function to find other user's public shared routes.
  - The ability to filter routes based on how long the route should be.
  - The app should have short response time.

Johannes, 21 - Student at Chalmers University of Technology

Name: Johannes Attila
Age: 21
Occupation: Student at Chalmers University of Technology
Hometown: Göteborg
Family: Single
Personality Traits: Active & likes to go for walks

- ***Description:***

  Johannes is a 21-year-old student at Chalmers University of Technology in Gothenburg, interested in computers. He is an active student in contact with many friends from his hometown, why he often travels and visiting them, staying on their couch etcetera. During the days when his friends have left for school or work he often likes to take a walk in order to see the town and its surrounding, which often leads to multiple phone calls to the hosting friends with complicated descriptions of directions.

- ***User objectives:***

  - Find routes nearby that fits for walking.

  - Find popular walking routes in the current town.

  - Find routes from special preferences, i.e. Coffee Shops, parks or lakes.

- ***Requirements in order to use CityWalks:***

  - Possibility to find routes that is walking walking friendly, in difference from the ordinary map applications that often suggests car traffic roads etcetera.

  - Easy to find good routes that makes it faster than to call friend and ask for ideas.

  - Ability to find popular routes in the current city.

Alicia, 26 - Part-time worker

- ***Description:***

  Alicia is a 26-year-old part-time worker at 7-eleven from Umeå. She is interested in traveling a lot in european cities with her boyfriend. Every free weekend from work she can afford she wants to travel and explore a new city. Alicia is not that interested in seeing tourist attractions such as churches and monuments, but she likes to walk

and pass ice cream stands, cafes and restaurants. Today she uses map applications to plan her routes combined with restaurant applications such as Trip Advisor.

- ***User objectives:***

  - Create a route from different preferences of passing points such as ice cream stands and lunch restaurants, before leaving the hotel and be able to save these routes offline on the smartphone.

- ***Requirements in order to use CityWalks:***

  - Combine functions from example Google maps and Tripadvisor to plan walking routes that passes selected preferences.

  - Easy to use on the go and in stressful environments since it will be used on trips in new towns.

  - Have the possibility to save routes locally on the phone to be able to walk it offline, to avoid roaming fees abroad.

6. Design and user interface development

When the user uses the CityWalks application the first thing they will recognize is the user interface. The user interface should be effective and easy to use as possible. The aim is to make the user feels comfortable when using the application. The goal is to attract all kind of users and make the user interface user-friendly regarding to design rules and choices.

6.1 Logotype

The logotype of CityWalks was developed after several options. First of all we developed a logotype with purple background with a skyline of a city in the background and a walking man in the front, see *Figure 6.1*. After further development we had a discussion and came up with a new idea that we considered smarter and more clean than the first one. The final logotype says "CityWalks" and a walking human being is incorporated in the logo in combination with the letter "i" and "a", see *Figure 6.2.* The man is symbolizing walking, which is the main purpose of the application.

Figure 6.1. The first CityWalks Logotype

Figure 6.2. The Finale CityWalks Logotype

We also designed two different mobile logotypes for the CityWalks application. For Android we created a round logotype see *Figure 6.3,* and for iOS we decided to have the square one since Apple rounds the corners automatically, which would make it fit in with the other application logos in iOS. Initially we used the CityWalks logotype in black with white background as the mobile logotype. But as we developed and changed the design as well as

the user interface of the application we chose to change the color of the logotype to white and add a purple background instead. This made our mobile logotype follow the theme of the design of the entire application. We chose to use a lighter purple background than the original purple, because it is easier to see when it's small and the contrast with the white text made it look visually better.

Figure 6.3. Application logotype for Android

Figure 6.4. Application logotype for iOS

6.2 Color scheme

The colors were chosen from Ionic Color Classes to design the application. In the beginning it was very difficult to decide how we wanted the application to look like. It was also difficult to get everyone to agree, then someone came up with purple. Since we did not want the color that associate with other application that uses daily, CityWalks decided to have purple as main color. We wanted the purple to be associated with a positive feeling and challenge the users to share their route success. The color choice was also important not to attract any special group, since we wanted the application to be used of all different ages, genders and ethnicities. If we for example had chosen a known masculine or feminine associated colour, such as pink or blue, it could result in that group looking as the main target, and make other groups feel excluded. With the purple colour of CityWalks we hope to have found a design that is not associated with any typical group, but instead to invite all people that could have use of our product and to hopefully start associating the colour with our brand.

We also decided to have background gradient to create a smooth transition between two or more specified colors. We thought a lot about coherent design on every page, with same

color and fonts to make it easier to use the application. If pages are similar there is way more comfortable to use the application. The button colors look similar to other application that has such same functions and is presented below and are also chosen from Ionic Color Classes. Font colors is either white or black to give the users familiar and comfortable feeling which is similar to other application.

Color	Hex code	Type
CityWalks Purple	#7253C3	Main background color
Light purple	#937BD1	Background gradient
Light gray	#D3D3D3	Background gradient

Figure 6.5. Background color scheme

Color	Hex code	Functions
CityWalks Purple	#7253C3	Share, save, submit
Red	#EF473A	Back, stop, done walking, skip
Blue	#387EF5	My location, comment,
Green	#33CD5F	Create, start, walk, add friends

Figure 6.6 - Button color scheme

6.3 User Interface

6.3.1 Development progress of design and user interface

Mockups

When the idea was stated for the project and a clear picture had grown on what the system should be the group tried to gather all ideas and sat down to sketch by pen and paper on how the application should look like. When some choices were not clear or when there was different opinions, the process got further by having a vote and then move forward with the winning idea. This process was an important stepstone to get thru because it was the first

time a clear common visualization was created which gave a clear goal for the group to know what to work for.

In *Figure 6.7* and *6.8,* examples of the mockup design is shown, and all mockups created is attached in the Appendix.

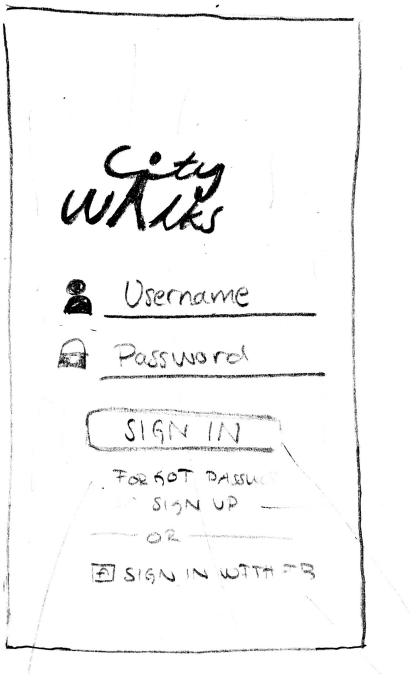


Figure 6.8. Mockup for login page

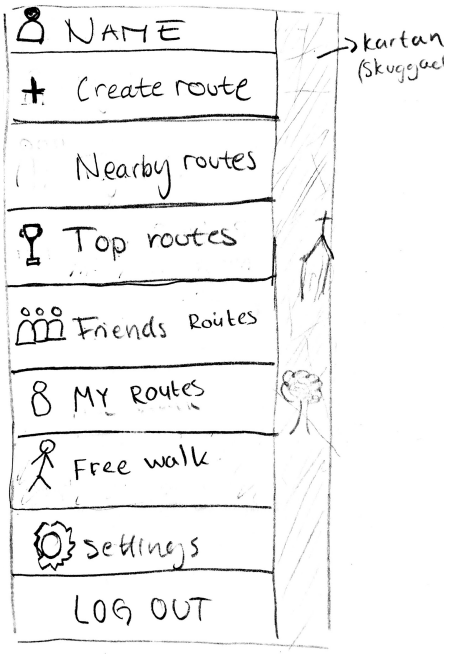


Figure 6.8. Mockup for side menu

Design Phases

By using the Mockups, the initial frame for the application was created. It consisted of a login page, and a map with a side menu when logged in. The first login page consisted of a background with a photograph of the Uppsala cathedral, something that could symbolize that the application was created of a Uppsala based team and that also showed small walking ways, a blue sky and city buildings that could associate the purposes of the product. The login fields consisted of clear forms with white background which made it easy to see the instruction text for what was to be inputted, see *Figure 6.9*.

Since the design team wanted a cleaner design the login field was later changed to be transparent which will be presented below. This was a decision that was not obvious however since it made had to compromise with the usability. The original white fields had been more clear for everyone to easy understand, but on the same time the application needed to attract all kind of groups why it also should be attractive and follow the design trends on the market.

The background picture was also a thing that was changed in the final design which will be presented below. This was done after the usability assessment (see chapter 10), because it made it clear that the picture made it to hard for the user to read the text on some places where the colors was to alike with the background. Because of that the picture was removed and instead a purple background was used on the login page. One alternative to that decision was to have the background picture dynamic based on which city the user was currently in. That would have been done with the system checking which coordinates the mobile had, and chose a picture from the database showing a city that was the closest available. This was decided not to be prioritized however and agreed upon to be a possible further development feature.

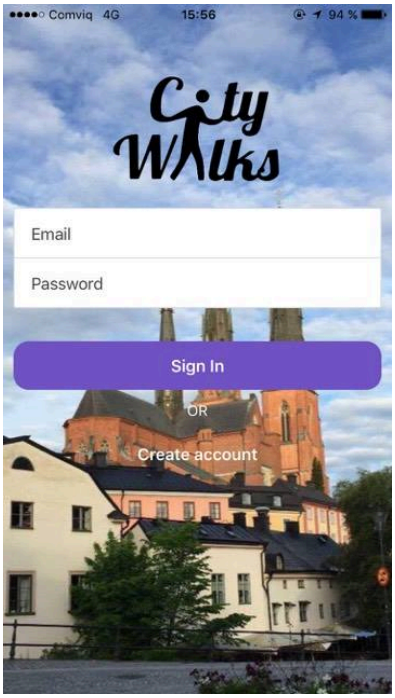


Figure 6.9. Initial design for login page

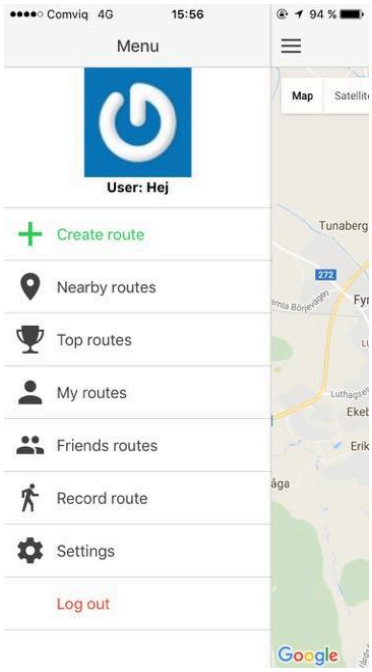


Figure 6.10. Initial design for side menu

6.4 Final design and user interface

After the different processes of design development on how the application would look like for the user, a result has been reached that the project group is very satisfied with and proud of. Of course there are many parts of the system that could be improved even more and there are a large number of ideas that still could be developed further, much of it presented in the further development chapter below. A mobile system today is also a dynamic product that continuously can be updated and get new design, and should do so keeping up with trends and new techniques available on the market.

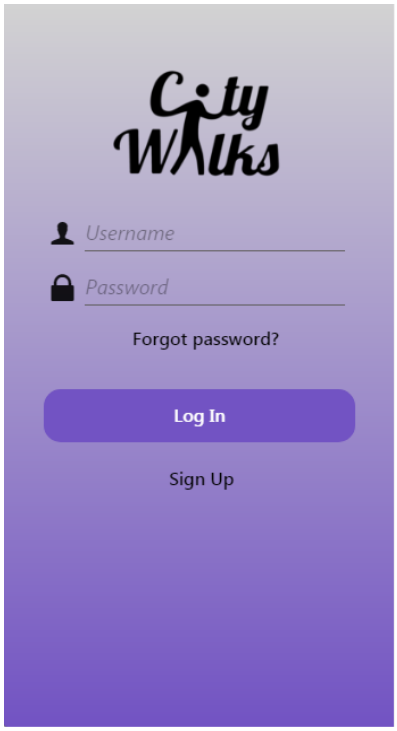
In this part however the applications design in its current state at the end of the project will be presented and explained in further detail, grouped after the different parts of it.

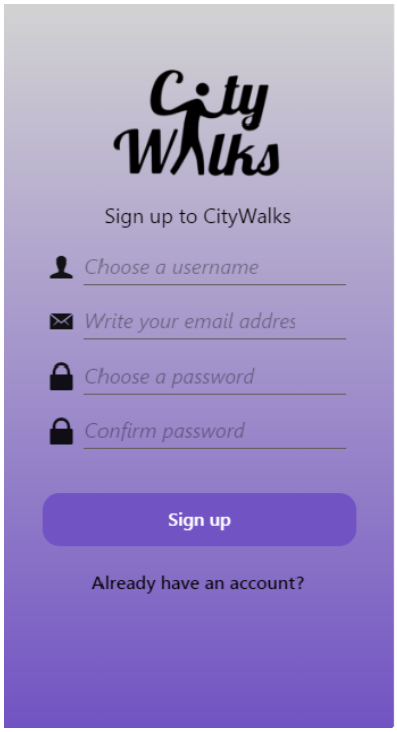
6.4.1 Log in, Sign up and Reset password

The first thing the user will see when open the application is the login page, *see Figure 6.11*, and from this page you'll be able to navigate to signup and reset password page. It's easy to log in if the user already has an account and remembered their password, otherwise they need to either sign up or reset the password.

A new user would be able to create an account which they do by clicking the button "*Sign Up*". The user will be oriented to the signup page, *see Figure 6.12*, where they need to fill in all the requested things, such as choose a username, email address, password and confirm the password they choose. By clicking the signup button the app will check if the username is already used by another user and it will also check if the password and the confirmed password matches each other.

If the user forgot their password they can easily reset it by clicking the bottom "*Forgot password?*", which will orient they to the reset password page, *see Figure 6.13*. At this page they will be requested to enter their email which they used when created an account.





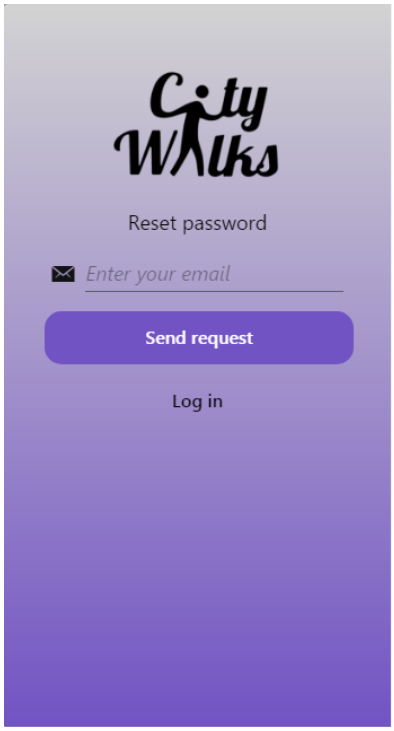


Figure 6.11. Login *Figure 6.12. Sign Up* *Figure 6.13. Reset Password*

6.4.2 Side menu

The side menu will show everything that can be accessed by the user. By tapping the icon with three lines in the left corner the user can from the side menu navigate to all the functionalities: Create route, Record route, My routes, Friends routes, Nearby routes, Top routes, Settings and Log out, *see Figure 6.14.*

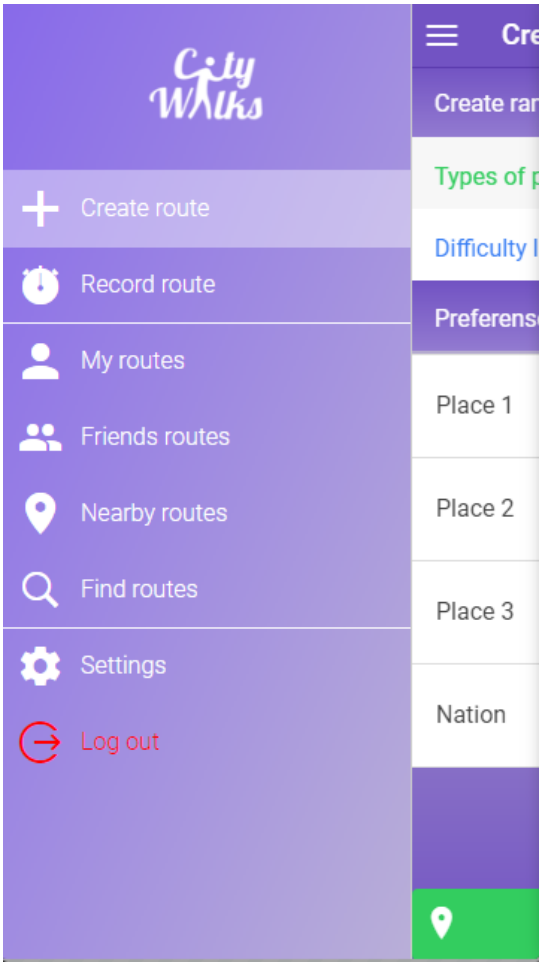


Figure 6.13. Side menu

6.4.3 Create Route

In Create Route you can either create a route by type of places or by difficulty level. When choosing the first choice the Create button will generate a route there you'll passing by every preferences you've chosen. Also distance and time to walk will appear, see *Figure 6.14* and *Figure 6.15.* Difficulty level approximate a distance and generate a route with different level choices, see *Figure 6.16* and *Figure 6.17.*

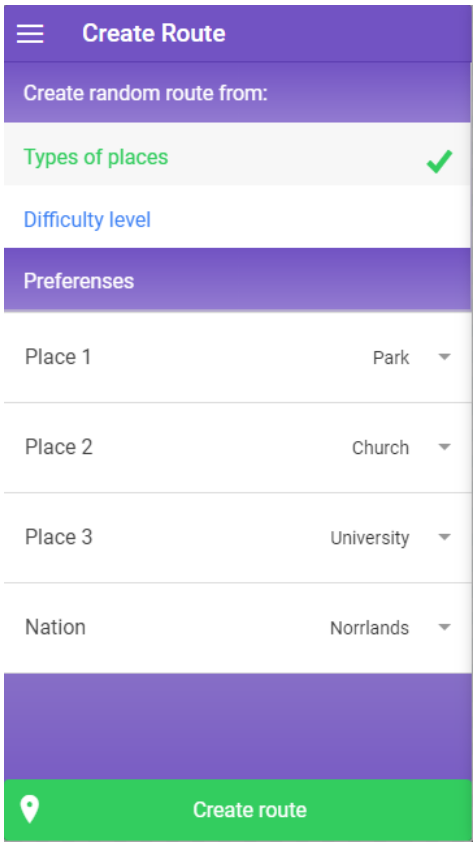


Figure 6.14. Types of places

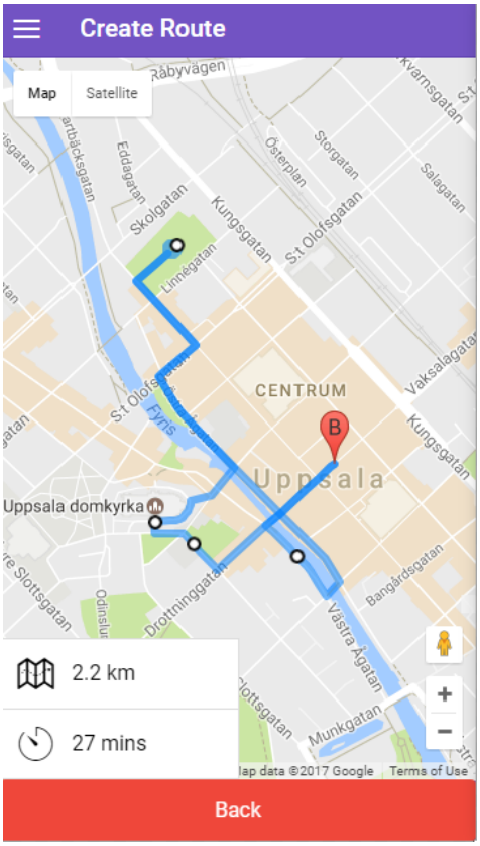
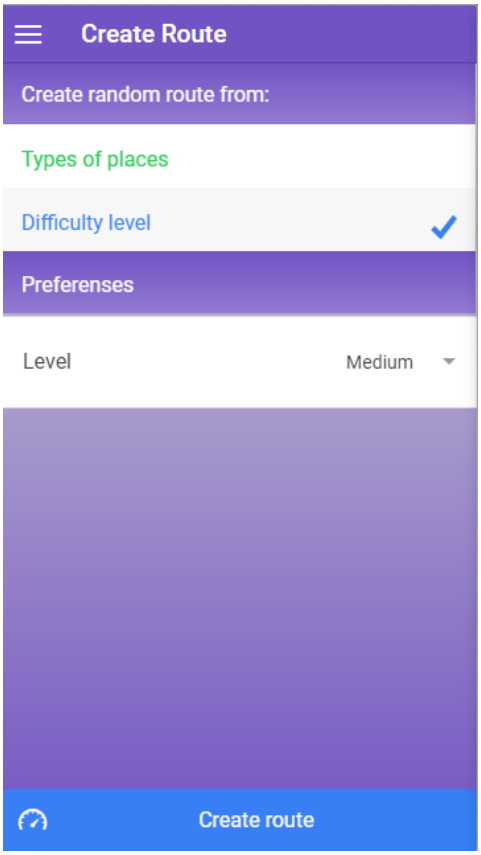


Figure 6.15. The generated route



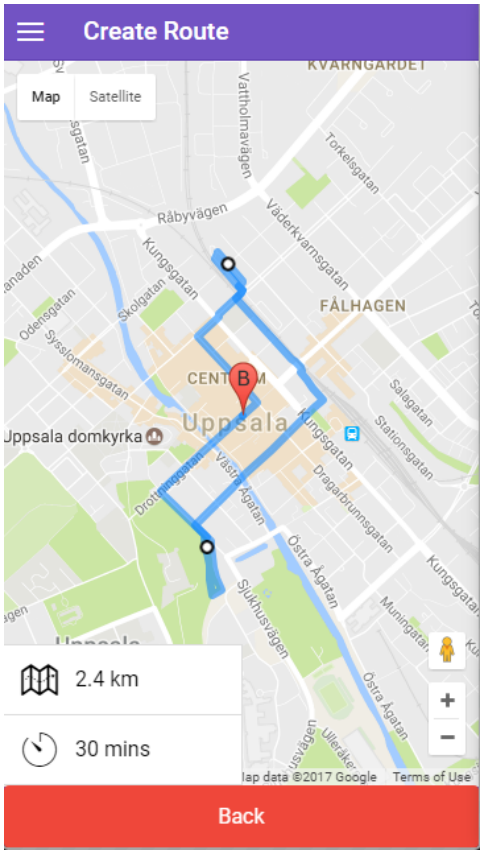


Figure 6.16. Difficulty level *Figure 6.17. The generated route*

6.4.4 Record Route

In this function the user can record their own route. By pressing on the green "Start recording" button the user will be able to start recording their route, see *Figure 6.18*. When finishing the walk, they could either choose to delete or share the route, see *Figure 6.19*. If the user want to share the route they can type in a title, a comment and also recommend it but it's not a requirement, see *Figure 6.20.*

Figure 6.18. My location

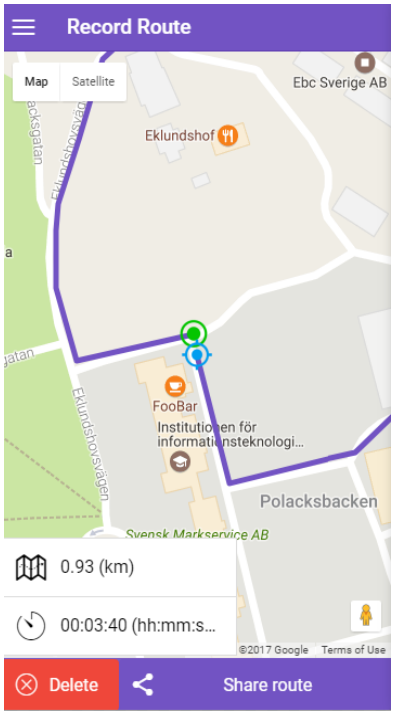


Figure 6.19. Finish recording

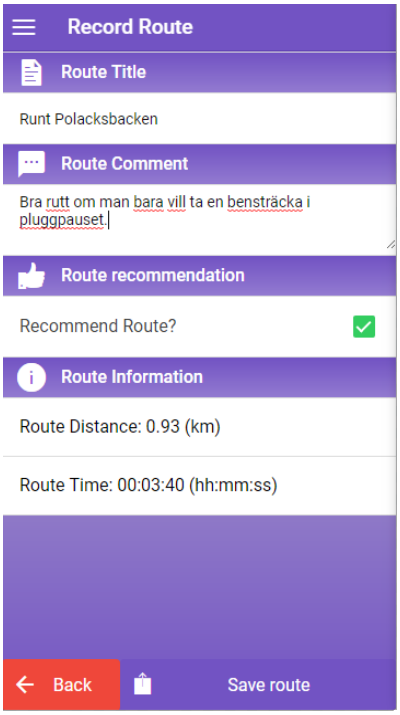


Figure 6.20. Information

6.4.5 My Routes

In this function all the users recorded routes will appear, see *Figure 6.21*. The user can choose to see more information and comments or maybe walk the existed route again, see *Figure 6.22*.

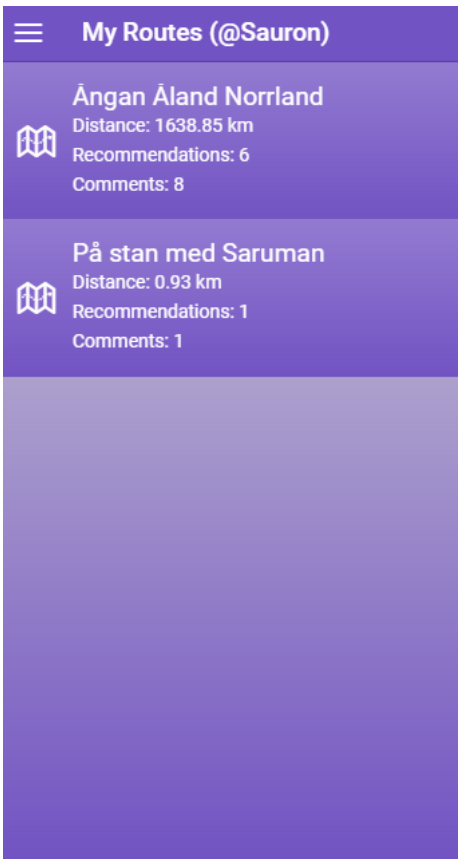


Figure 6.21. All recorded route

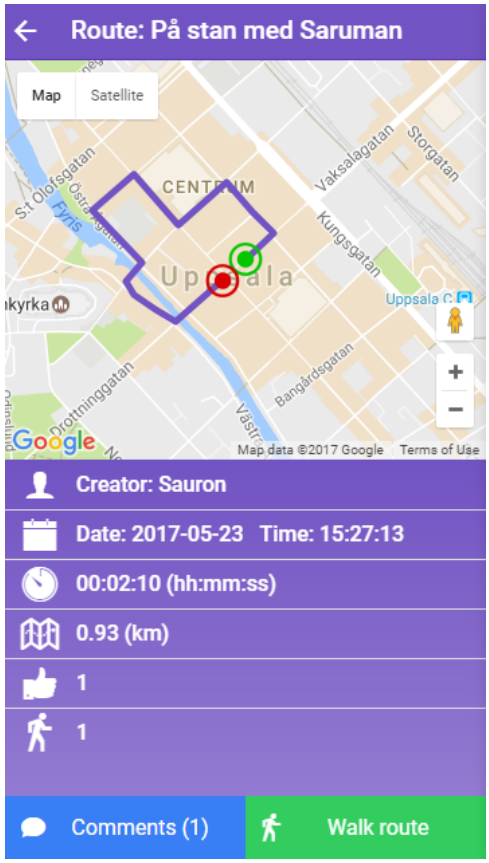


Figure 6.22. Route information

6.4.6 Comment function

From the route information page, a user can go and read the comments and also write comments and post questions, see below in *Figure 6.23*.

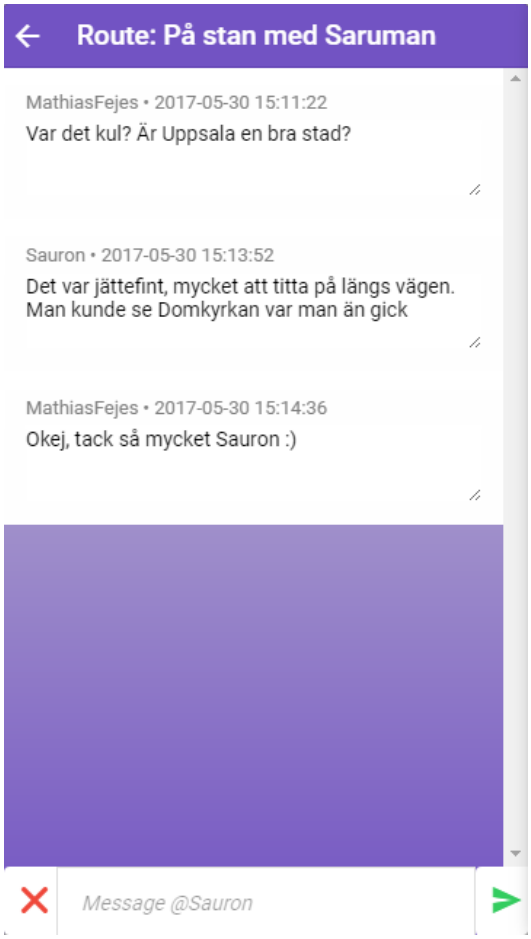


Figure 6.23. Comments on a route

6.4.7 Friends routes

In Friends routes the user can firstly display his or her friends and the number of created routes the friend has, see *Figure 6.24*. From this page the user has two options. The first one is if the user chooses to press *Add/remove friends* the user comes to *Figure 6.25*. There the user can add a friend by inputting the friend's username or the user can remove a friend by pressing the circled X next to the username. The second option is to press on a friend's username and then the user comes to *Figure 6.26.* There the user can see that friends created routes and press on a route to display the details and then walk it.

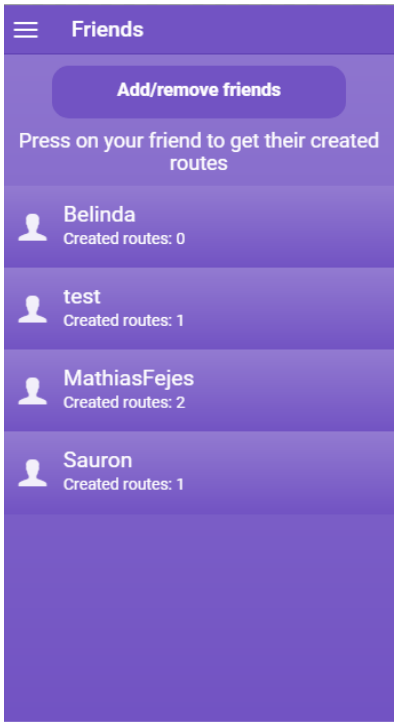


Figure 6.24. Friends page

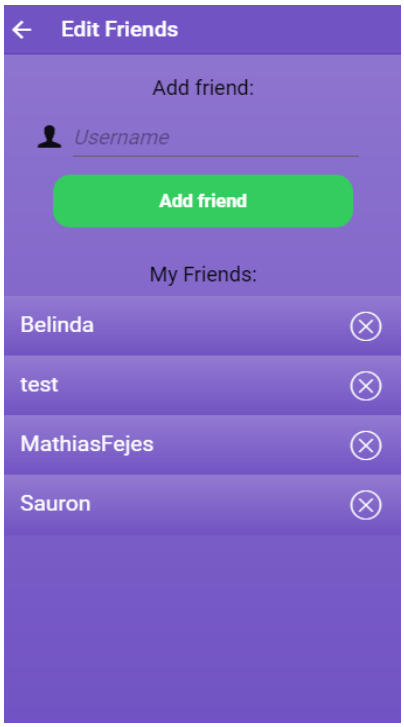


Figure 6.25. Edit friends page

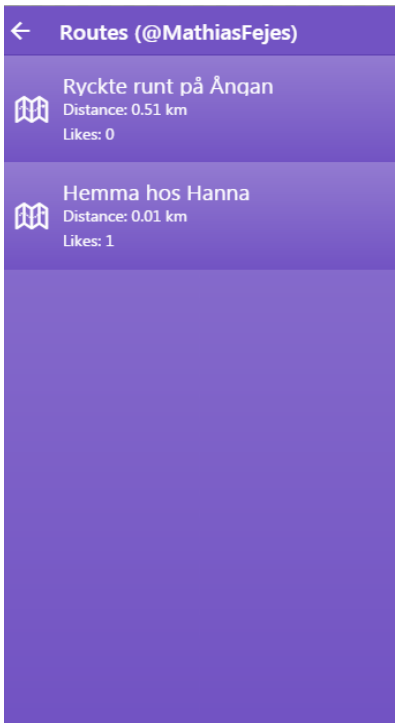


Figure 6.26. A friends's routes

6.4.8 Nearby Route

The *Nearby Routes* function displays the routes that is closest to where the user's current position is. They are sorted after the shortest distance and it is displayed as "*Distance to start position*:", see *Figure 6.27*. If the user wants to walk the route the nearest way to the start position will be provided, see *Figure 6.28* and *6.29*.

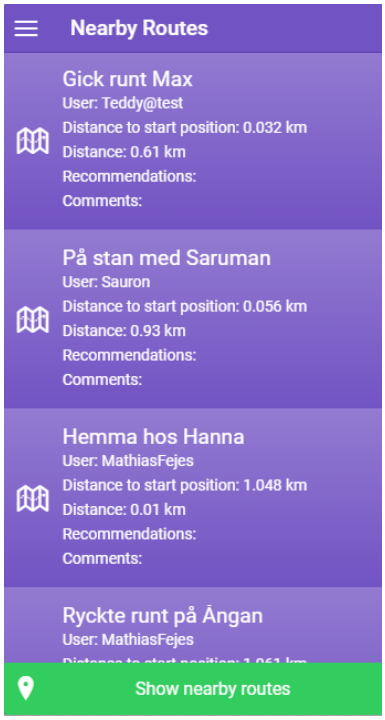


Figure 6.27. Nearby routes

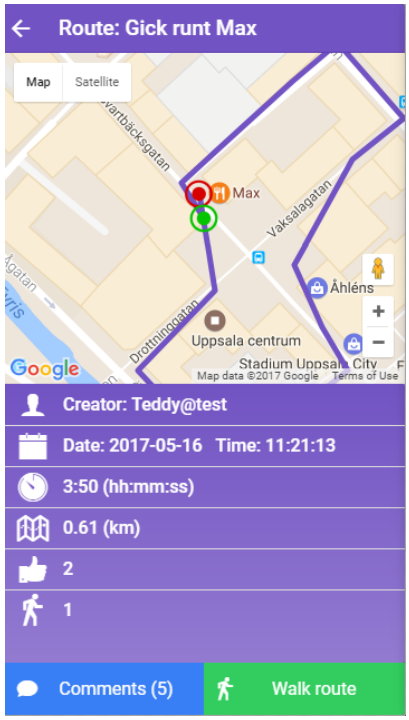


Figure 6.28. "Gick runt Max"

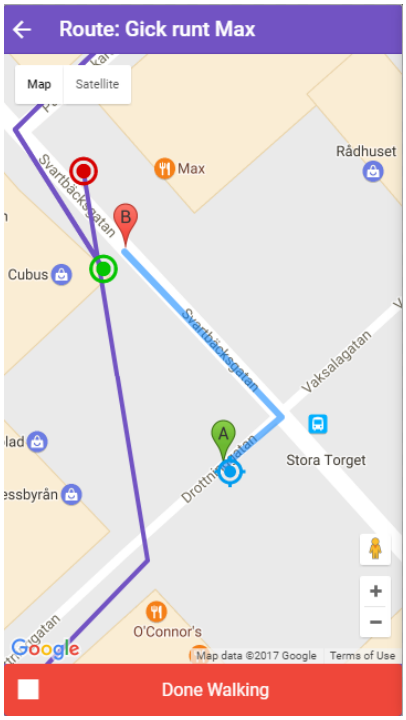


Figure 6.29. Walk route

6.4.9 Find Routes

Find routes function is a function where you can sort the routes by recommendations, date, distance, comments, username and how many user has walked the route. This function is well implemented and connected with the database in correct way. By using this function, the user can sort the routes by their wishes. How it looks like is presented below in *Figure 6.30* and *6.31*.

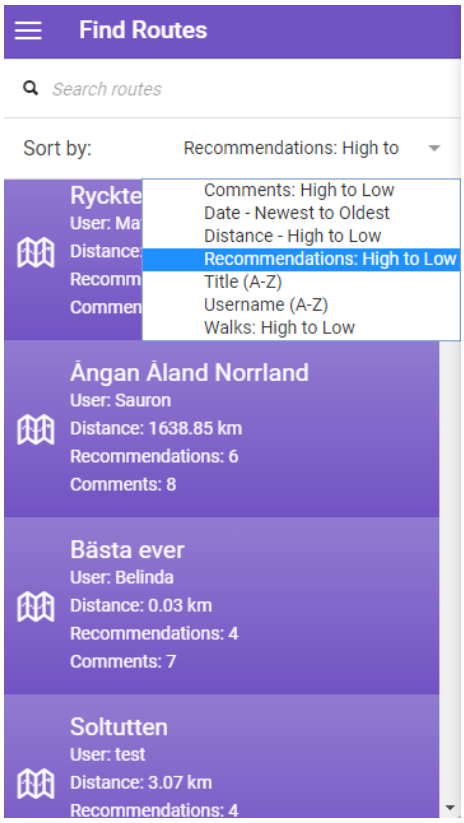


Figure 6.30. Sort by recommendation

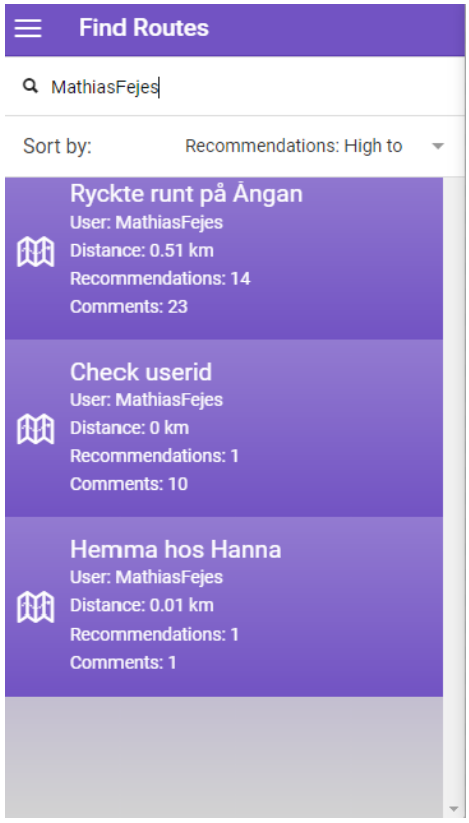


Figure 6.31. Sort by username MathiasFejes

6.4.10 Settings

In Settings the user can display his or hers account details i.e. username and email. The user can also change the password by typing the current password, a new password and confirm the new password as seen below in *Figure 6.32*. The layout on this page is under construction since we got some criticism on usability testing, but we didn't have time left so we just let it be.

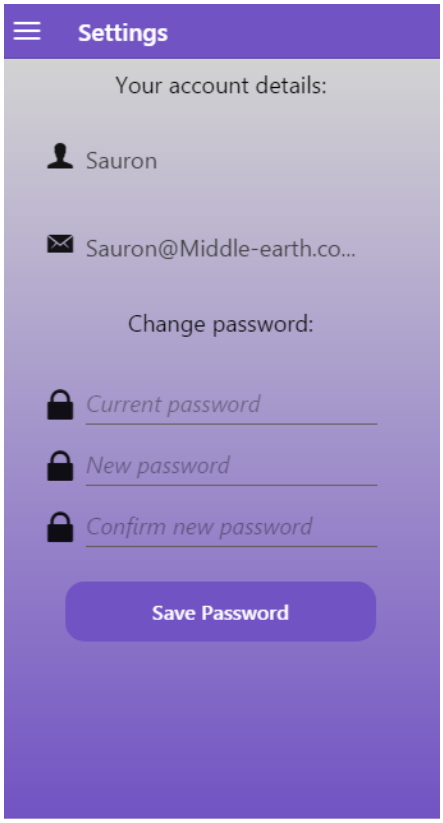


Figure 6.33. Settings

7. Technical Solutions

To build our application we're using the software bundle MEAN Stack, which is a free and open-source bundle of popular JavaScript based components used in creation of dynamic websites and in our case web application. MEAN stack hasn't been use in its original form tough. We used Ionic alongside Angular.JS in the frontend and Feathers.js as a supplement to Express in the backend to simplify the creation of the API. Presented below are these technical specifications in further detail and a figure to provide an overview of the entire system, see *Figure 7.1*.

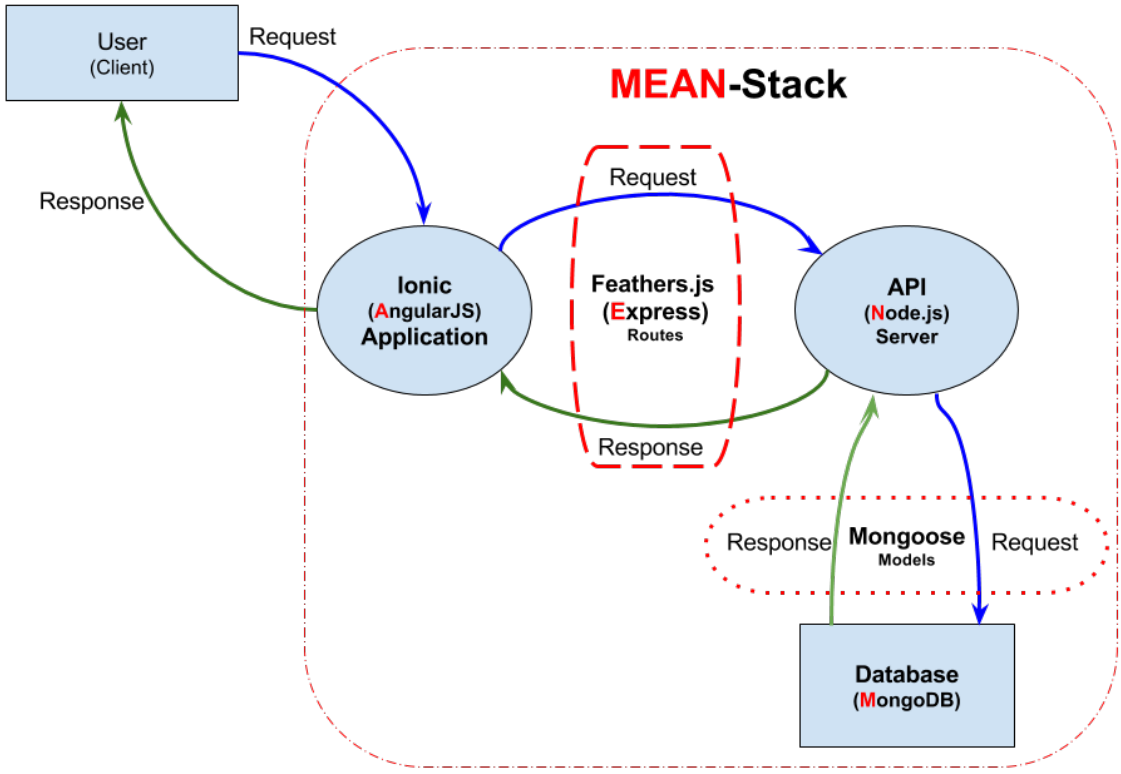


Figure 7.1. System overview, MEAN-stack

7.1 Front-end tools

To deploy our application on Android and IOS, we have been using **Android studio, Xcode** and we have mainly been using **Visual Studio** and **Brackets** to edit the code. A extension for Visual Studio was also used, called Tools for Apache Cordova. As the application becomes more complex, Visual Studio together with Tools for Apache Cordova, a simple code editor

like Brackets can become insufficient. Visual Studio's code editor is therefore optimal for a complex project since it is optimized for refactoring, auto-completion, code comprehension, debugging, previewing and testing. [9] [10] [11]

7.1.1 Ionic Framework and Ionic Creator

For the CityWalks application, the Ionic Framework have been used, which is similar to the framework Bootstrap but for mobile applications. Ionic is an HTML mobile application development framework targeted at building hybrid mobile apps. Hybrid applications are essentially small websites running in a browser shell in an app that have access to the native platform layer. This means that we can use HTML, CSS, and Javascript to build the app, and that we can target multiple platforms with one code base. To access the native layer and to run the app native on the phone, Ionic needs Cardova, which also provides plugins for the geolocation and maps that we need for our routing functions. The Ionic Framework also uses AngularJS for a lot of the core functionality of the framework. [12] The Ionic Creator is a drag-and-drop development tool for building hybrid apps with Ionic and AngularJS. This tool help makes Ionic easier and faster to develop. Ionic Creator also comes with Ionic UI elements, ready-made templates and the option to download the project as a zip-file for further development. [13]

7.1.2 Javascript

JavaScript is used on most websites and enables functionality. We choose Node.js alongside with the webApp approach to be able to use JavaScript through and across the entire creation of the application. This enabled us to smoothly transition between back and front end when extra resources were needed. [14]

7.1.3 HTML

The standard markup language used in creation of web applications and web sites. HTML provides the structure of the page using elements that divide the page into sections and subsections with different properties. [15]

7.1.4 CSS

CSS is commonly used together with HTML and JavaScript to create the interface for web-

and mobile applications. The CSS language is what describes the style and how the HTML documents are displayed including layout, fonts and colors for example. [16]

7.2 Back-end tools

When developing the application, the developers in the back-end team have been using the MEAN Stack which enable JavaScript to be run throughout the entire application, client-side to server-side. The components of the MEAN Stack are as follows:

7.2.1 MongoDB

A document oriented noSQL database, which we choose because it fits well with the MEAN stack mentioned above and we wanted to learn about the NoSQL way. In hindsight a SQL alternative would, perhaps, been a better fit for us. The data that required storing follows in some cases a natural "table like" structure. We have only tested the part, collection of our database named "checkpoints" with a large data set. And the search functions in that part are sufficiently fast. We are a bit unsure how the nested data in the collections Routes and Users that are heavily dependent on each other would hold up as the amount of users grows. All indications tell us that it would be fine but better testing protocols would have put our minds at ease. At the beginning of project, we were using a local computer as host for the database, we quickly realized that we needed the information to be available outside the university network so we created an account on MLAB.com and migrated our database to there servers in Dublin. Our intention was to move the database once more to our Digital Ocean server in Frankfurt to shorten the response time of a request to the database. Since performance, to our surprise was satisfactory and time was short we kept this setup. [17] [18]

7.2.2 Feathers.js (Express)

A node.js web application framework that helps us setup the API. We are using *Feathers.js* which uses express and grants us extra features and simplifies the CRUD (Create, Read, Update, Delete) functions of the API as well as adding support or third party authentication. Sadly we never got around to creating authentication options with google and Facebook. Although feathers.js failed in that sense, it made it easy for us to set up local strategies for authentication using JSON Web Tokens (jwt). [19]

7.2.3 AngularJS

Used in the front-end part of the MEAN stack and a JavaScript framework that runs in browser JavaScript engines. We are using Cordova Ionic that gives the extended framework for developing cross-platform mobile functionalities and apps. AngularJS uses the term “Scope” to connect variables from HTML to the JS “Controller”. All the forms and templates are created in HTML files and defined in the “Model”. The Scope function detects changes in the Model to be further used in the Controller and for example used with CRUD functions. This division gives a clear understanding of the code, where all functions are executed and what is presented in the view. Once you have learned to use Scope, Model and the Controller in AngularJS this has this proved to be a powerful tool. [20]

7.2.4 Node.js

Enables JavaScript to run on the server and uses an event-driven model that makes it efficient. It also uses a collection of “modules” that handles the functionalities such as a simplified API design. Since Node.js is event-driven and therefore uses a non-blocking system the commands are executed parallel and with callbacks instead of creating a block until completion compared to other programming languages. [21]

Regarding the server for the database the following tool has been used:

7.2.5 DigitalOcean

A cloud infrastructure provider that enables developers to scale and deploy applications. The server we are using is running in Frankfurt and this is where our server side code is implemented. DigitalOcean works as the middleman between our AngularJS Controller and MongoDB. For example, all the CRUD functions we execute in the Controller are rendered through DigitalOcean to MongoDB. [22]

7.3 Project Management tools

All communication during developing CityWalks both within the project group and with all the teachers is on Slack. Slack is a chat application that brings all pieces and project member. Even Trello has been used to set up goals and plans for the week but also for long-term plans and goals.

7.3.1 Trello

Trello is a project management tool that is available online. Trello has been used to visually organize the roles in the team, the activities and the time for each activity. Used Trello to organize our work in Sprints and to-do cards. Our goal in the beginning of the project was to follow the project plan and divide each week into a sprint with a goal of what would be completed in the end of each sprint. We started by setting up a sprint for the first week, during the week we realized that the time to set up all the tools and the computers took longer than expected. This resulted in that we had to plan more time for the first sprint. We had this in our minds during the project and tried to plan our sprints in a more realistic way. [23]

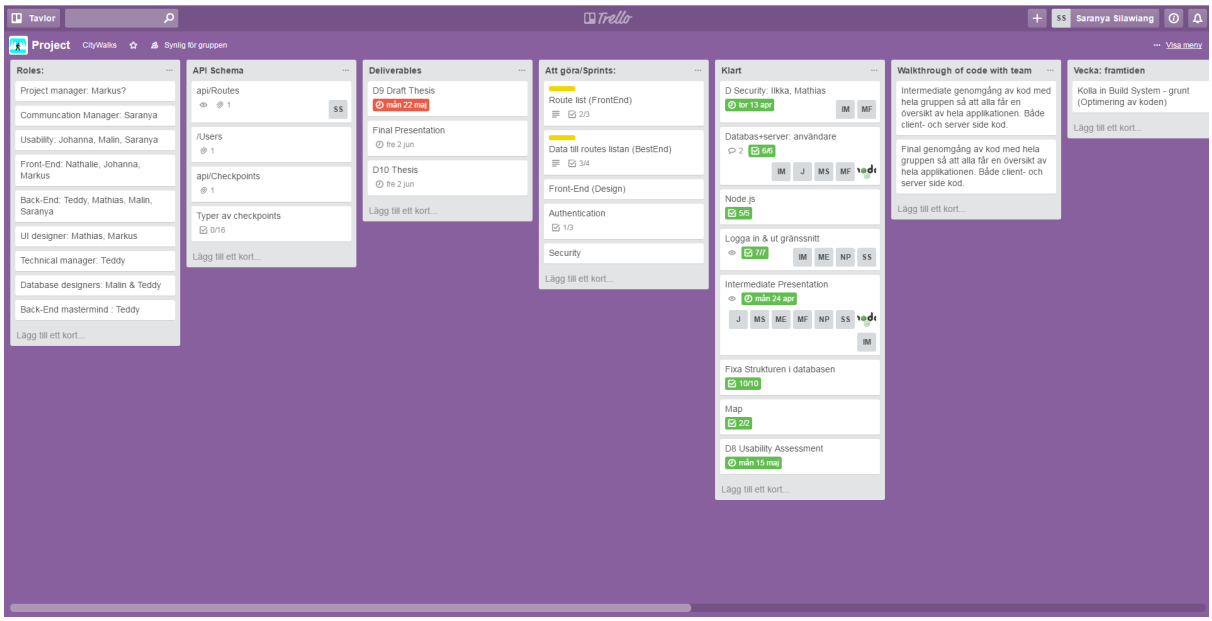


Figure 7.2. Trello created by CityWalks

7.3.2 Slack

Slack is an application that is used for professional communications within teams (Slack, 2017). We used slack as the major tool to communicate with each other during the project. We organized the team's conversations in different channels such as #general, #database, #practical, #GitHub etc. This helped organize our communication and make sure that everyone on the team was on the same page and received all the important messages. [24]

7.3.3 GitHub

Github is a web-based version control repository that is being used for source code management. We used GitHub to be able to collaborate and work with the same code when developing the application. With github education we got some free software licenses, private repositories on github as well as 50$ on digital ocean. So they bought us fair and square, these perks could not be competed with. We all had some experience with GitHub so setting up a repository went quickly to set up and creating different branches helped us a lot to experiment with different versions and functionalities of the application. [25]

7.3.4 SourceTree

SourceTree is a free Git Client that provides a more graphical interface for repositories than a command line. With the functionalities Commit, Push, Branch and Merge we have been able to keep the team up to date with the latest code. From beginning we found it hard to understand, because this type of program hasn't been used before. It took a while to understand how to use all the functionalities that it provides. But we figured it out and this tool was used diligently as one could effectively share the code. What was extra good is that you could sit on your own branch and code, but if you're wrong, you can always go back to the latest version of the code. At first we used *GitHub desktop* but we soon realized that the merges overwrote previous changes in a menacing and non desirable way. [26]

7.3.5 Google Drive

Google Drive, before called Google Docs, is a free web-based word processor used for storing and sharing documents. Google Drive has been using frequently, most for writing all the deliverables that has to be handle in during the course. We also using it for organizing all the documents, derivables, meeting protocol, PowerPoints and images into different maps to make it easier for everyone to find. [27]

8. Back-end development

The back-end development has consisted of designing the database and setting up the connection from the client to the database, the API. So in this chapter the work with the database and the API will be described. The first designs and how these evolved to our final version will be presented.

8.1 API and Database design

In earlier courses we have only learnt about SQL databases. Therefore, when we had the opportunity to choose which kind of database to use we chose a NoSQL database since we wanted to learn more on how these were structured and used. We decided to use the document oriented database MongoDB, where every collection consists of JSON objects with the same structure. As it happened MongoDB fitted nicely with the type of app we were planning to make, a web application, since the M in MEAN (see the previous chapter) stands for MongoDB and it is a commonly used and appreciated database solution.

For MEAN applications the structure of the database is a integer part of how to setup the API. In the API we decide how and what data to be stored and the database follows these rules. The rules are called models and represent the structure (Schema). Every JSON object that is added, through the API, to the collection follows these rules. Therefore, we will describe the development of these components simultaneously.

When building a MongoDB you can choose to handle relationships in your data in two different ways:

- **Embedded Relationships:** Nest the related information inside the actual object, for example having all the routes, walked and created by a certain user, and all the information about them stored in that user. The drawback is that if the documents grows to much, as we in our case due to the massive predicted popularity of our app, read and write functions can be slowed down.
- **Referenced Relationships:** This approach uses the unique auto generated ID created for each new object and whenever previously stored information needs to relate

somewhere it is referenced by it's ID. The drawback is that two database requests has to be made. One to get the desired ID's and one to find the desired information corresponding to those ID's.

After careful consideration we chose the **Referenced** approach, mostly due to the drawback described above but also because it seemed like the "right and proper" way to go. We suspected that this way required more work but would pay of at the end. We also knew if we would fail we could always swiftly create an **Embedded** version. During the development different features required us to change the setup of the API. These changes will be described in the following sections. [7]

8.2 Initial design

We started out using our previous knowledge of databases from a course in SQL design. No relational tables was created, since we knew those were not used in noSQL other than that our knowledge was limited. Our first database design had four collections which was *Route, User, City* and *Attraction* as seen below in *Figure 8.1*.

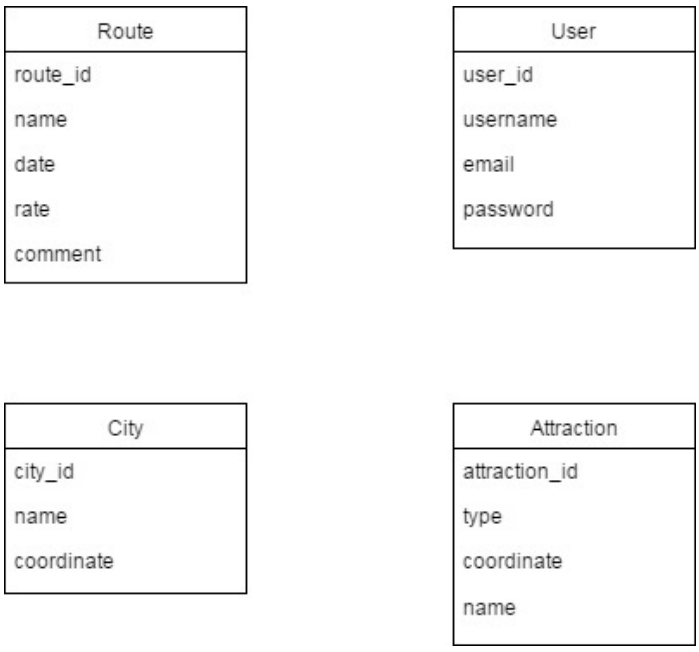


Figure 8.1. Initial document oriented NoSQL database design

The first version was created as a permutation of one of the many tutorials we tried, that one was built with Node.js using Express, Mongoose and MongoDB (MEAN) and was originally designed to store information about bears. The database was at first located on one of our computers (localhost) but later moved to an MLab server. Express was used to manage the URLs (what was going to happen when you sent a request to http://localhost:3000/api/bears). Mongoose handled the MongoDB, connection and the models required to properly store data in the correct, desired structure(schema) in the database. This example system was then redesigned to match our specifications seen in the figure above. The models used by Mongoose were primitive, designed almost solely for testing and all the information was stored as type *String, Number* or *Array of Numbers.*

This version served as a base for the back-end part of the first version of the app, the first major sprint, and it enabled the front-end to start working on elemental CRUD functions. It lacked a lot of features, for example the authentication of the user with a hashed and salted password. At this time, we used the Ionic Auth service to fill this gap, described later on in the Ionic framework part of the Front-end development chapter (9.1.1).

Worth noticing is that in this first version each route required the creation of separate functions for each kind of request it was supposed to manage, which we found tedious and identified as a possible source of future errors. This in combination with the somewhat poor structure inherited from the tutorial project.

8.3 Work in progress

As the project went on and we learnt more and more about noSQL databases and the needs our application have. We felt that our database was lacking in some areas, so the responsible team sat down a couple of weeks into the project to revise our first design. The first change in the database was that the collection City was removed because it felt redundant. We realised it was unnecessary to have a collection with the cities since the system uses coordinates. The code itself checks all the attractions and if they are close enough to implement in the route. If we later on want to sort out routes depending on which city they are in, we could just add another key in the collection *Routes* that would

provide the same results. Several group members did not understand the name of the collection *Attraction*, they felt we needed a clearer name of places the user wanted to see on his or hers route. Therefore Attraction was changed to Checkpoints.

We realized that our application was in need of relationships between the collections. For the social interaction, every user should be able to add and remove friends. Thus there should be a relationship between every user and their friend. As we chose to work with referenced relationships we added an array to the collection Users with IDs of the user's friends.

To connect the user with the route he or she created, the collection *Users* also got an array with the IDs of the created routes. For us to easily get information we also added the user's ID to the collection *Routes* and named it creatorId. The collection Routes got a couple of more changes. As we wanted to have a counter on the routes with how many people had walked the route we added an array of walker's ID to the collection *Routes* with the IDs of the users who had walked it. In *Routes* we also changed score to an array of JSON objects, where every object consisted of an auto generated object ID, the ID of the user who gave the score and the score itself. We chose to add a reference in the JSON object to make sure an user could only like the route once, and the user could consequently also remove the like. The array comments also became an array of JSON objects consisting of an auto generated object ID, the ID of the user who commented, the date of the comment and the comment itself. This so that every comment can have the username of who wrote it and the date it was written displayed next to it.

The last thing we changed was that we added a referenced relationship from the collection *Routes* where we added an array for the IDs of the checkpoints the route passes. We needed this to display which types of checkpoints the route has when the user are looking at other people's routes.

When we did these changed we wanted to use the format GeoJSON for the coordinates of the route, this seemed to be a useful way to store the coordinates, as it had a lot of benefits.

As mentioned above, in the first version we used Ionic Auth. That meant that the users and their passwords were stored on Ionics servers, which was not ideal. To solve this problem, we had to create our own authentication provider and find a good and secure solution for storing of the passwords and other private information and also be capable of using third-party-Authentication like Facebook or Google. After some research and a tip from a friend we found feathers.js and tried it out. Feathers.js turned out to be even more capable, it provides a function called services for the routes which handles all the CRUD functions. Because of that we changed our previous Express routes to the Feathers.js handled routes. The problem of creating a proper authentication still remained though, and would be solved later on when a complete rebuild of the system took place, described in the final design part.

We struggled with the relation between the collections, and the model on the API needed to be able to handle the *type: objectId* instead of the previously used *type: String.* After some work we managed to fix this but during this time we constantly evaluated the possibility of changing to the *embedded* approach.

8.4 Final Design

After working on the project for 10 weeks we now have the three collections *Users*, *Routes* and *Checkpoints* in our final database, see *Figure 8.2*. These are not set in the database, they are defined by the API.

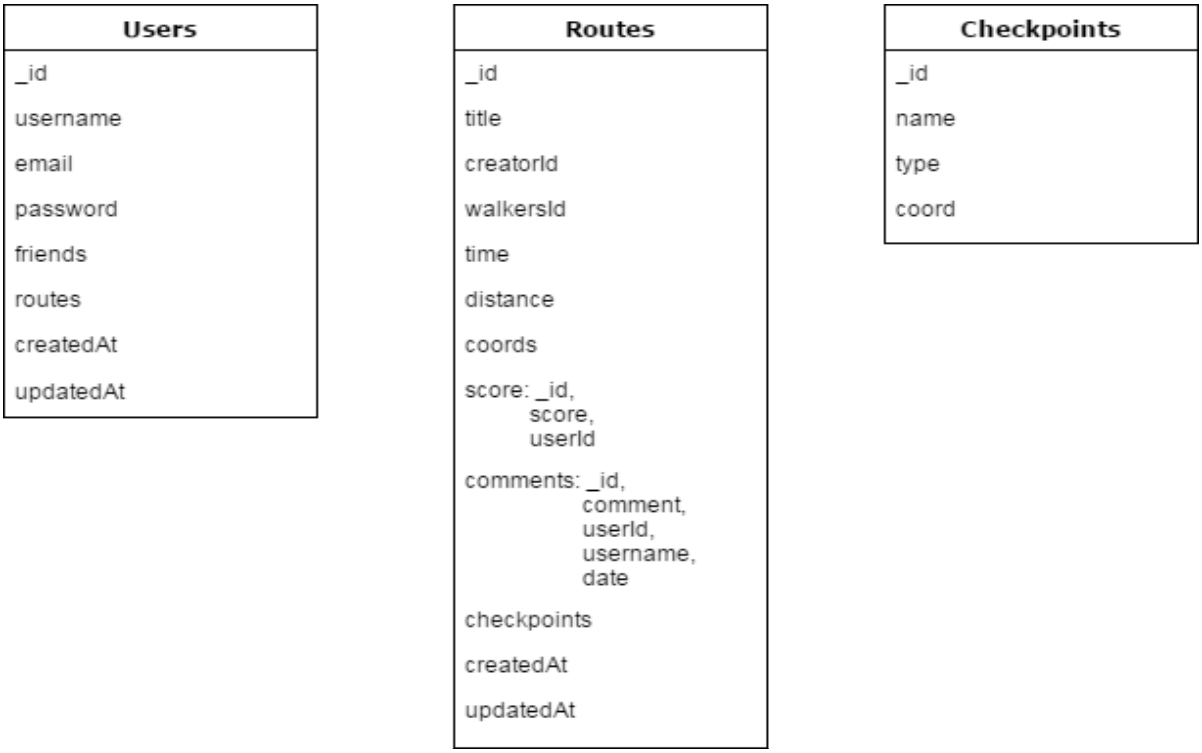


Figure 8.2. Final Database design

To take a closer look how the data is stored we have one example of JSON objects from every collection. Below in *Figure 8.3.* you can see one user in the *Users* collection. The user has an unique auto generated object ID called _id. The user also has an username, email, password, friends, routes, createdAt and updatedAt.

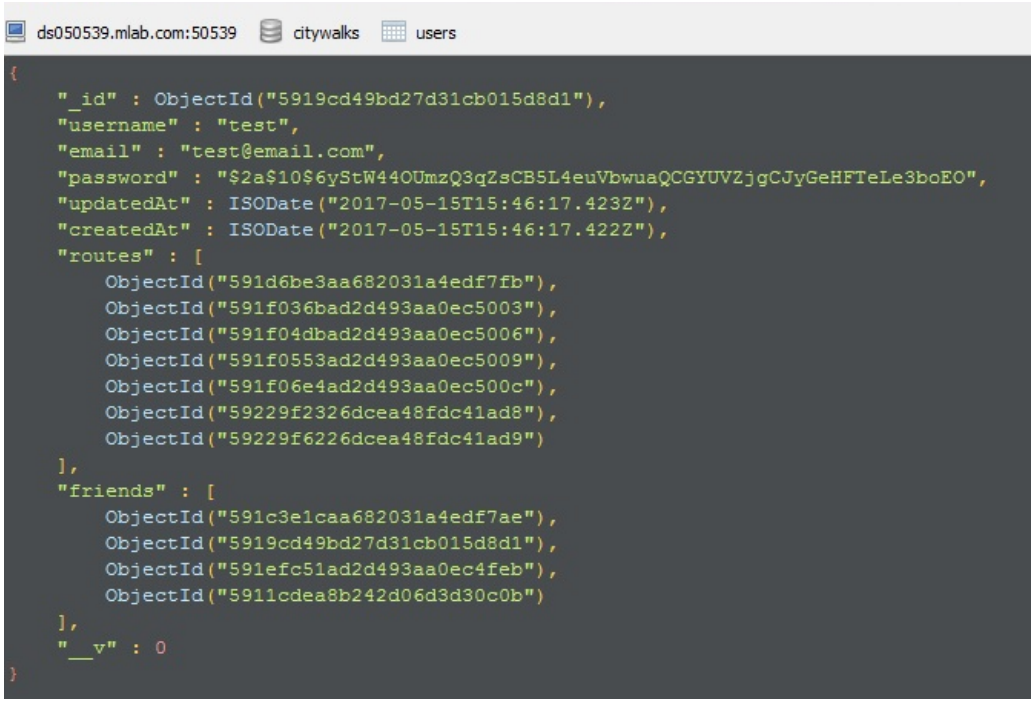


Figure 8.4. user from the database collection Users

Below in *Figure 8.5.* it is viewed one route in the *Routes* collection. The route has an unique auto generated object ID called _id. The route also has a title, creatorId, walkersId, time, distance, coords, score (a JSON-object that consists of _id, score and userId), comments (a JSON-object that consists of _id, comment, userId, username and date), checkpoints, createdAt and updatedAt.

ds050539.mlab.com:50539 citywalks routes


```
{
    "_id" : ObjectId("592438b126dcea48fdc41af5"),
    "title" : "På stan med Saruman",
    "checkpoints" : [],
    "creatorId" : ObjectId("5911cd9d8b242d06d3d30c09"),
    "createdAt" : ISODate("2017-05-23T13:27:13.467Z"),
    "updatedAt" : ISODate("2017-05-23T13:27:13.468Z"),
    "coords" : [
        [
            59.858832747181,
            17.6395351886724
        ],
        [
            59.858832747181,
            17.6395351886724
        ]
    ],
    "walkersId" : [
        ObjectId("5911cd9d8b242d06d3d30c09"),
        ObjectId("5919cd49bd27d31cb015d8d1")
    ],
    "time" : "00:02:10",
    "score" : [
        {
            "userId" : ObjectId("5911cd9d8b242d06d3d30c09"),
            "score" : 1,
            "_id" : ObjectId("592438b126dcea48fdc41af7")
        }
    ],
    "comments" : [
        {
            "_id" : ObjectId("592d701026dcea48fdc41b33"),
            "userId" : ObjectId("5911cd9d8b242d06d3d30c09"),
            "username" : "Sauron",
            "comment" : "Det var jättefint, mycket att titta på längs vägen",
            "date" : ISODate("2017-05-30T13:13:52.705Z")
        }
    ],
    "distance" : 0.93
}
```


Figure 8.5. route from the database collection Routes

In *Figure 8.6.* you can see one checkpoint in the *Checkpoints* collection. The checkpoint has an unique auto generated object ID called _id. The checkpoint also has a name, type and coord.

ds050539.mlab.com:50539 citywalks checkpoints


```
{
    "_id" : ObjectId("5911c9c48b242d06d3d30bfb"),
    "name" : "Södermanlands-Nerikes nation",
    "type" : "Nation",
    "coord" : [
        59.858875,
        17.630224
    ],
    "__v" : 0
}
```


Figure 8.6. checkpoint from the database collection Checkpoints

When we were done with the design we wanted to populate the collection *Checkpoints* with places we could use for the routing algorithm. We downloaded a CSV-file with over 90 000 coordinates of places in Sweden from GeoNames.org. The places were lakes, castles, mountains, parks etcetera and the file had the names, types, type description, coordinates and much more. Because not all the information in the file was useful we only took the objects that were suitable, which gave us a CSV-file consisting of 32 076 objects. We populated the collection with the file and tested how fast different operations on this collection, through the API, would be. None of the runs relevant to the app took too long so with regards to the overall performance of our app this was good news. Even if this part has the most simplistic structure in our database we still got a first intuition on how fast the back-end handled searches in larger datasets.

When redesigning the database structure we also rebuilt the API. This time we used the entire feathers.js framework. This gave us a clearer and better structure and helped us to a fresh start on the authentication problem, which as it turned out was a lot easier at this point. In a matter of hours Authentication with JSON web token(jwt) was live. When a request is sent to the /Authentication part of the API with the correct *username* and *password* a (jwt) is returned to the client, this token is added to the header of following requests and grants the client access to certain otherwise restricted sections. The check if the token is valid is done in a hook on these restricted routes on the API. Hooks are another feature we got from feathers.js, it enables us to manage the request before the service that connects to the database is fired and manage the response from the database before it is sent back to the client.

Initially the plan was to use these hooks to populate the response to the client with usernames and route titles instead of the id's. As an example whenever you want to access information about your friends, these are stored in the database as id's but the client is only interested in their names.

Although third-party-authentication was set aside due to lack of time, the required foundation is in place at least on the server-side. This also applies to a number of neat, not yet implemented functions like reset password and hooks for specific requests, these will be covered in the section about *further development*.

Feathers.js acts as a fully compatible wrapper over Express so even though this might seem like a big change it still follows the same MEAN structure we had utilized to this point. Since the Feathers.js service is able to utilize Mongoose models, we should be able to store the coordinate objects as GeoJSON types in the database. This would have enabled us to use the database to find adjacent geographical objects, which seemed like a nice feature. This might have some drawbacks though, It might put a heavy load on the server if not properly designed. Sadly, we never got to try this due to lack of time, we abandoned the GeoJSON plans and worked with coords in longitude, latitude format as Numbers. Which meant that the entire location logic would be done on the client. In the end, lack of time also pressed us to abandon the hook plans for improved responses, so transforming id's to corresponding names had to be made on the client, everywhere except for the comment route section where we made the username alongside the userId a fixed part of the mongoose model. How this was done is described in the *Front-end, social features, friends section* and the consequences are discussed in the *testing results* and *General improvements* sections.

9. Front-end development

The front-end team's development of CityWalks can be divided into two main parts, firstly creating the functionality of CityWalks features and secondly implementing the visual part of the application. Therefore, the focus changed constantly between integration of newly created functions with the back-end team's work and adjusting the design of the functions according to the usability team's work. In the beginning of the development, the main tasks were broken into smaller parts that could be tackled by only one developer. But as the tasks became more complex and time-consuming over time, the front-end team was divided into groups of two or three often together with people with other roles. Since much of the work in the front-end development was done in collaboration with people from the back-end team and the usability team, many of the predefined roles also changed and transitioned over time. In the end of the development it was therefore hard to distinguish people's roles as only front-end, back-end or usability. Even though the collaboration went all right with people from the back-end team and the usability team, the collaboration also had some outcomes that were undesirable. Since the development was divided into many tasks were handled separately and by different people, this lead to a lot of repetition of the styling CSS and JavaScript code. Therefore, there is also not much separation between the CSS and HTML code, which is not favorable for a good code hygiene. A better solution to this issue would be to have a configuration file with the different styles and colors that was used throughout the application. Similarly, the repetition of JavaScript code could be solved with more service functions, that are global functions and accessible throughout the application.

To achieve the initial goals of this project and finalizing a functional and user-friendly application, the front-end team began with an early plan. This plan divided the work into three main sprints with regular milestones, and the plan was supposed to be followed after a predefined amount of time in a linear fashion. But as most projects that involve several people and many unknown elements, the development encountered several issues that resulted in detours which required that the initial plan had to reevaluated and changed. Because of this, the front-end development became partly nonlinear but the three initial sprints are at the same time the after the development noticeable. These three sprints can

be summarized as firstly setting up the Ionic Application with a connection to the database, secondly adding the map-features and thirdly adding the social features. In this chapter the front-end team's development will be described in three parts, ordered after the sprint in which the features were created.

9.1 Ionic application and end-to-end connection

The first main sprint in the front-end development was about four weeks long. To goal was to have a complete setup of the Ionic application on a physical mobile device. In addition, the application was also supposed to have a connection to the database as well as the implementation of a basic function for each one of the CRUD features.

9.1.1 Ionic Framework and Ionic Creator

Early in this project, the Ionic Framework was chosen as the framework for the CityWalks application. This choice was mostly based on that Ionic Framework could target multiple platforms, specifically iOS and Android, with the same code. Besides this, the Ionic Framework was also based on familiar code languages and had support for several Cordova plugins that could access the geolocation on the physical mobile device. The first task in the front-end development was therefore to create a basic Ionic application. This was done quite easily with Microsoft's Visual Studio, that was installed with an extension specifically for Cordova/PhoneGap and Ionic. This Visual Studio extension came with a couple of Ionic templates, and the "*side-menu*" template was chosen that consisted of a HTML-page with a side-menu, a JavaScript controller page for the side-menu and the thousands of files that an Ionic application requires. To make sure the the Ionic template was working properly, it was then run in the Visual Studio Android emulator and on a physical Android device, through Visual Studio together with Android Studio.

Since none of the members of front-end team had any experience with Ionic applications and its structure, the front-end team then spent several days on going through Ionic's documentation and watching relevant tutorials on both Youtube and Ionic's website. By doing so, the front-end team also found a web-based tool created by Ionic called *Creator*. The *Creator* tool comes with all the basic components for an Ionic application that simplifies

the process of creating and connecting different HTML-pages as well as adding components, such as buttons, with Angular and standard JavaScript functions. With *Creator*, the main structure of the CityWalks application was then created, consisting of a Log in/Sign Up-page and a Side-menu with five different HTML-pages. Two other components from the *Creator* was also used, the Ionic add-on called *Auth* and the Cordova plugin *Google Maps.* The *Auth* add-on made it possible to have Ionic take care of the authentication, storage of user information and a way to reset the password via e-mail. It should be emphasized that the Ionic Auth was later removed due to a few issues, described in the Back-end development, chapter 8.3. The plugin *Google Maps* made it possible to initiate and display maps on the HTML-pages. After this was done, the *Creator* tool was after mainly used by the usability team to quickly and effortlessly try different designs of the HTML-pages.

9.1.2 End-to-end connection and CRUD functions

After the main structure of the application was done, the front-end development focused on getting a end-to-end connection with the database that the back-end team had developed. The first task was to get the CRUD function *Read* to work and read information, created by the back-end team, stored on the database. The whole front-end team worked on this, which proved to be harder task than suspected, and it was not completed until the end of the third week. The initial structure of the end-to-end was then a HTTP request of a collection in the database, which response was displayed in a list division on a HTTP page. After this, the front-end team split up and worked in pairs on the remaining CRUD functions *Create, Update* and *Delete.* These three CRUD functions proved much easier to create, since their structure resembles the *Read* function, and were completed after only a few days.

These initial four CRUD functions were then the foundation of all the other features, created in the second and third sprint, that had some sort of connection to the database. The *Create* CRUD function was in the end used in the Signup and Create Route features. The *Read* CRUD function was in the end used in, among other, the Friends, Top Routes, Find Routes and Create Route features. The *Update* CRUD function was in the end used for, among other, the Add/Remove Friend, Add Comment/Recommendation and Password Update features.

The *Delete* CRUD function was in the end not implemented in any features, but could in a future development be used for features such as Delete Route and Delete User Account.

9.2 Map features

In the second sprint the focus of the front-end development shifted to creating the map features. In the initial plan two different features were thought of, one that would record a walked route and one that would create a route from different preferences. During the development in the second sprint, the front-end team started working together with the general implementation of the Google Map API and the Cordova plugin *Geolocation*. After this was completed, the front-end team split up into two groups where one group worked on the create route feature and the other group worked on the record route function. These two functions proved to be complex, involving algorithms and calculations, and this development took several weeks and the majority of the time of the second sprint.

9.2.1 Google Map API and Cordova Geolocation

The development of the map features started with going through documentation for the Cordova plugin *Geolocation* and examples of JavaScript functions that were using it. After a while, the front-end team realized that most of the examples used *Google Map API for JavaScript* and not the Cordova plugin *Google Maps*. Since the documentation for the Cordova plugin *Google Maps* also was limited compared to *Google Map API for JavaScript*, the front-end team made a decision to develop the map features with the later and completely drop the Cordova plugin *Google Map.*

The CityWalks application rely on a lot of different map features to be able for example to show created routes, nearby routes and your current position. First of all, the CityWalks project was registered in the Google API Console to receive a API Key. With this key and the standard plan for Google Maps JavaScript API Services we were granted 25.000 free requests per day. During the time of this project this limit has not been exceeded since all of the requests have been generated from daily and frequent testing by CityWalks developers and not from real users of the application. CityWalks also holds own Checkpoints from the database and own API that are used moderately at the moment when for example creating

a route from types of places and from the option Nations. The Checkpoints are coordinates that CityWalks can create and read from, and together use with Google Maps. This gives the project and application fewer restrictions, possibility to grow on own and the need not to rely only on Google Maps API.

A function in Google Map API is the Places Library that enables an application to search for places given a certain area around a fixed point with different options. This function has been handy in our feature Create Route when creating the map by users preferences and giving the user fixed options. This is done by first checking the user's position and after that given a radius, search nearby for the chosen preferences that are send as different variables in the request to the API. It is also possible to search for multiple points of interest that are arranged in waypoints along the route and gives the user a complete route to walk with the end destination at the user's origin position. The request requires a type value parameter that is built-in in the options list and we use eight out of the approximately two hundred different types. Some of the ones that currently exists in CityWalks are park, museum and cafe. In the Create Route feature the places are mapped out after coordinates given latitude and longitude. After setting up the points a polyline is drawn to connect all the places to a complete route. Some of the built-in functions in the request that are made to the API are travelmode, distance and time. The travelmode have been set to walking which gives the maps and routes extended ways to travel compared to driving for example. With distance and time obtained from the JSON object we can display some further information about the created route the user is about to walk.

To fetch the position and coordinates from the mobile device’s GPS, Cordova’s plugin *Geolocation* was used as mentioned above. This add-on was added easily through Visual Studio, since the extension pack for Cordova/PhoneGap and Ionic included the 20 most common Cordova plugins. The plugin *Geolocation* has two main functions, “get position” and “watch position”. To get the position, the javascript function “navigator.geolocation.getCurrentPosition” is called and it returns the latitude and longitude values, and other related information, one time. To watch the position the

javascript function " navigator.geolocation.watchPosition" is called and it returns the same information as the "get position" function, but continues to return this information in a defined interval until the javascript function "navigator.geolocation.clearWatch" is called. The position data from these functions can then be used together with the Google Map API by converting the position data to the format that the Google Map API uses. To convert a single set of coordinates, the JavaScript function "google.maps.LatLng" was used. To draw a line between and convert a collection of coordinates, the JavaScript function "google.maps.Polyline" was used. Another related function that was used in the map features was the JavaScript function "google.maps.Marker", which places a marker with information on a specific coordinate. To get a direction from one coordinate to another coordinate, the JavaScript function "google.maps.DirectionsService" and "google.maps.DirectionsRenderer" was used. The final part of each map function is to make a connection between the JavaScript controller and the HTML document. This is done by first creating a map in the JavaScript controller with the function "google.maps.Map" and then giving this map an identification value that is called upon in a division or section in the HTML document.

9.2.1 Algorithms and Calculations

Record Route

One of the features in CityWalks that requires calculations is the Record route feature. To simplify, this feature records the path when walking which can then be shared to a public database of shared routes. More about the share functionality can be found below, in Social features (chapter 9.3). The first part of the Record route feature is a full screen map that shows the current position with a marker. To start a recording, a "Record route" button is pressed, which is connected to a watch position function that fetches the position every fourth second and stores the latitude, longitude and time values in a array. The position values in this array is then converted and displayed as a polyline on a map. Two markers are also placed on this map, one green marker placed statically on the first position and one blue marker dynamically placed on the latest coordinate. When a "Stop recording" button is pressed, a function is started which calculates the total time and the total distance of the recorded route. The time calculation compares the time value of the first and the last object

in the array and calculates the time difference between them. The resulting total time value is then converted into a readable format in hours, minutes and seconds. The total distance calculation compares the position values in the array, calculates the distance between them in a for-loop and adds each distance to a total distance value. After this is done, a clear watch function is initiated which stops the watch position function.

Create Route from Places and Level

Some of the features in CityWalks like Create Random Route from types of places and from difficulty level have functions to calculate distance, algorithm to randomize points in a defined area and order places after the “traveling salesman problem”. In the Create Random Route from types of places we have calculated distances between all the places and applied the traveling salesman problem in order to make the route more “flowing” and to some extent avoiding the route to cross itself and create a “zigzag” pattern. The zig zag pattern occurs when the points are send to Google Maps API that without any specific order. When creating the route after difficulty level an algorithm to generate random coordinates have been applied to get different routes regarding position.

The calculations made in the distance function results in as the crow flies, in other words a straight line between two coordinates that contains latitude and longitude and is measured in kilometers. In the calculation the earth's radius is incorporated and degrees are converted to radians to be able to get the distance. When creating this function for measuring distance an important question have been kept in mind, and that is that the users can’t walk as the crow flies. In this case a route will be drawn along a possible walking path given on the map between the points and will almost every time result in a longer route between the points than what the straight line would that we measure. Using the measured distances this way and not getting the real or exact distance between the points is sufficient enough for CityWalks and the generated routes. First, since the actual total distance will be given from the JSON file from the Google API callback when all the points and map have been generated. Second, because the main point is not to optimize and produce the shortest path

between the points and create the shortest possible route, an approximation of the distance is most of the time all that is needed to give the route an arbitrary flow and avoiding zigzag.

The traveling salesman is problem that is about a salesman that needs to visit a number of cities and this in the shortest possible way and distance. The cost is given between each city and the problem is to find the route that costs the least. To find the minimum cost might be time consuming if there are a lot of cities because the possibilities are given by (n-1)! and with eleven cities we would end up with 10! possible solutions. This well known problem have partly been applied in the application and in the Create Route feature since the routes are not in need of optimization or finding the shortest path between the points. But when creating ways between the points that are close to each other the zigzag patterns are somewhat eliminated because points that are close to each other doesn't cause big jumps and therefore minimizing the zigzag pattern, see *Figure 9.1*. A "greedy algorithm" of the traveling salesman problem have therefore been adapted where the algorithm in the end doesn't care if the solution is the most cost effective with distance in mind. Given a point, the modified greedy algorithm only checks which of the remaining points are closest to it and iterates the points this way until all points have been checked. This is done with help of the function that calculates the distance between coordinates.

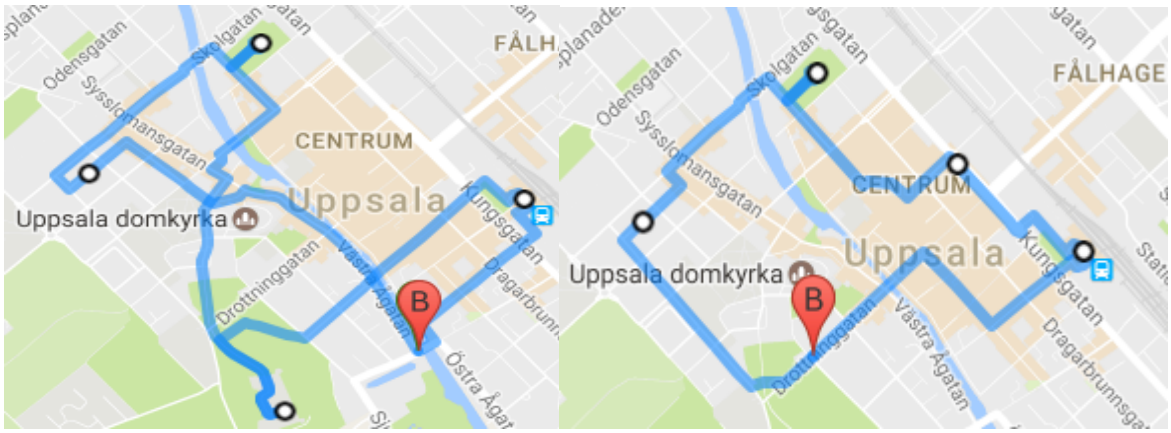


Figure 9.1. Example of before and after greedy algorithm implementation.

When choosing the preferences by difficulty level an algorithm that randomizes coordinates is adapted when generating the map. This algorithm have been created so that regarding of

the user's position a new route will be generated. The one main parameter that the algorithm uses is the one given by the user in the level option list, and the different levels are translated into a radius in the function. For instance, the option “Hard” is translated into a radius of 1000m and will generate a route of total interval between 3.0 - 5.5km, the option “Easy” is translated into a radius of 100m and will generate a route of total interval between 0.5 - 1.5 km, see *Figure 9.2*. Given that the radius is the only changing parameter, the results or intervals of the levels are quite large but somewhat controllable. The levels and intervals have been created because of the difficulty to obtain exact distances and given it more flexibility to randomize more routes given the same position. A backside to this is that a larger radius for instance used in the Expert level also has the possibility to generate the same distances as the Easy level. This is possible since the radius doesn’t have a lower limit when the radius expands. To get the random coordinates initial position is always calculated together with the radius. With the initial latitude and longitude two new coordinates x and y are generated within a given angular of the circle, after this the function adds the new coordinates to the initial coordinates and then adds these points to a array. Then the algorithm generates two new coordinates x and y in a different angle than previously and adds the new couple of coordinates to the array. The algorithm does this until every angle of the circle is covered and all new possible coordinates are added to the array. From this array the algorithm uses a random function to pick two pairs of coordinates that will be used to produce the random route.

Level	**Interval (km)**	**Average distance (km)**
Easy	0.5 - 1.5	0.75
Medium	1.5 - 3.0	2.50
Hard	3.0 - 5.5	4.25
Expert	5.5 - 12	8.75

Figure 9.2. Levels explanation

Walk Route

A couple of different algorithms and calculations is included in the Walk route feature. To simplify, this feature shows a map with directions to a already created route as well as the route itself and a marker on the current position. When a “Walk route” button is pressed on the page with information about the route, a function starts which creates a map centered on the current position. A marker is then placed on the current position as well as the recorded route. After this, the Google Map API Service calculates the optimal walking path from the current position to the first position on the recorded route. This direction path is then placed on the map and is dynamically changed when the current position changes. The Google Map API Service also estimates the walking time of the path and calculates the path distance.

Find and Nearby Routes

When searching for recorded routes in the application two important features is to find the closest and the most popular routes. Both of these features required and was made possible with different calculations. The calculations behind the Nearby Routes feature starts with fetching the current position of the phone. This position is then compared to the first position of all other routes, and the distances between them are calculated and added to an array together with the corresponding route data. This array is then sorted according to the distance between the current position and the start positions of the routes, and the 15 closest routes are displayed in a list.

To simplify the Find Routes feature, it makes it possible to search for specific details about the routes and sort the routes after, among other, most recommendations and most comments. The calculations behind the functionality of the Find Routes feature partly depend on the design of the Recommend and Comment Route features, see more below in the part Social features (chapter 9.3). When going to the Find Routes page, a Angular function initiates a JavaScript function that first gets all the routes from the database. After this step, the function calculates the number of comments, recommendations and walkers. Then these values are added to a object with the corresponding route data and inserted into

an array. To each object, the route creator, title and date is also added. This makes it possible to use an Angular function that can sort the routes on the HTML page, as well as making it possible to to search for specific details about the routes.

9.3 Social features

After the map features were mostly done, the front-end development shifted its focus to creating and implementing the social features from the initial plan. This part of the development can be seen as the third and last major sprint of the front-end development, and the features described in this part were under development until the penultimate week of the project. The social components and features of the application was one of the strict requirements of the project, and in the initial plan there was four different kinds of social features. All four of these social components was in the end implemented and the front-end started together with creating the Share feature. Then the front-end team split up into two groups where one group created the Friends feature and the other group the Comment and Recommend feature.

9.3.1 Share feature

As mentioned above, the Share feature was the first social feature in the CityWalks application. It was initially added to the Record route feature and was created in collaboration with the back-end team, since the the HTTP request to the server API required a specific format. The basic functionality of this feature was initially to send a recorded route to the database. Also, the recorded route required a title, set on the HTML page, before it could be sent together with the coordinates, route distance, route time and the creator's unique id. Two optional features was also added on the HTML page where the title was set, to recommend the route by activation a toggle and to add a description, the first comment, to the route. After the usability testing of the application, the toggle for recommending was changed to a checkbox since some evaluators struggled to understand the toggle. In the penultimate week, the required data of the HTTP request was updated on a new version of the server API, that now included a new object called walkersId. This walkersId object contains a list, in which the route creator's unique id is first added and

when another user have walked the route, their id is also added to the list so that other users can see the total number of different people that have walked the route.

9.3.2 Comment feature

The Comment feature in the CityWalks application lets the users comment on created routes and obviously also read comments from other users. This is an important social feature since it makes it possible for the users the ask questions about the route and to find out more than the basic information presented. Because of this, the Comment feature is implemented on three places in the application. Specifically, in the Share route, Walk route and also in a particular Comment feature. In the Share and Walk route features, the user has the opportunity to add a comment when a new route is shared or when an already created route is walked. The particular Comment feature is a own HTML page that looks and work like a common chat application, see chapter 6 for more information. On this page, all the comments of the route is listed after the create date and time of the comments. There is also a possibility, like in the Share and Walk route features, to add new comments in this feature. As mentioned above in the Share feature section, the comments are stored in a list in the object containing information about the route. In this list, the comments are stored together with the corresponding user id, the create date and time of the comment. There are a few deliberate limitations of the Comment feature. Empty comments are forbidden and can not be added, and the comments can be a maximum of 120 characters long. If a user tries to send an empty comment the user gets notified by a popup that says that the comment must include text. To show the user how many comments there are on a route, the total number of comments is displayed on each route, in the list of all routes, as well as on the button that takes the user to the particular Comment feature.

9.3.3 Recommend feature

The Recommend feature in the CityWalks application gives the users the opportunity to recommend routes that they have created or walked. This is an important feature since it makes it possible, together with the Comment feature, to find and choose the best routes. This function to recommend a route is implemented on two different places in the application. Firstly, on the page where a recorded route can be shared and secondly on the

page that is shown after a route is walked. Similarly, to the comments, the recommendations are stored in a list together with the corresponding user id. With the user id of the recommendation, it was possible to create a restriction that prevent users from recommending a route more than once. If a route is not recommended, no item will be added to the list, so that the length of the list becomes to total number of recommendations. To show the user how many recommendations there are on a route, the total number of recommendations is displayed on each route, in the list of all routes, as well as on the HTML page with an overview of the route.

9.3.4 Friends feature

The Friends feature lets the user add and remove friends. The user can then display the routes that their friend has created and walk their routes. A function where a user can walk a friend's route is a desired feature in this type of application. At the beginning the Friends feature was divided, the option to display their routes was under menu called *Friends routes*. Meanwhile the feature of adding and removing friends was separated and was a button under the *Settings*-page. This was changed for consistency in the application, all the functions regarding friends should be placed together. Now the button where the user can add/remove a friend is under the *Friends routes*-page.

As mentioned in the back-end chapter, the intention was to use hooks on the server for the friends functions. As time was pressed, this was instead accomplished by using a service. The service took an ID or an username and provided the other one for that specific friend. Every user has an array called *friends* in the database. When all the user's friends was loaded, the application sent the list with friends usernames to the service and got an array with the corresponding usernames and also the number of created routes of that user. This could then be displayed on the *Friends routes*-page. The service was also used when a user adds or removes a friend. When a user is added the username is sent to the service and the ID returned. Then the CRUD function *Update* is used to make changes to that specific array in the database. The Update function is also used in a similar way if the user want's to delete a friend, except then the splice method for JavaScript was used to remove the ID from the array.

10. Security and Privacy

10.1 STRIDE

With an overview of the CityWalk application we can identify threats and which aspects are the most pressing for our system and users. STRIDE is a method developed by Microsoft to evaluate these aspects of the system in the following way:

Spoofing identity or so called “man in the middle attack” is illegally accessing and using another user’s authentication information such as username and password. A lot of people use the same password for several or all of their accounts. So therefore we have to make sure the passwords are secure so no one can access them. Otherwise an intruder can take other users passwords and use them on other systems to get sensitive information, such as banking and e-mail information. Another problem might be that the attacker pretends to be someone else. The attacker would then be able to fake routs, comments and get access to this person's private routes.

Tampering with data involves modification of data when including unauthorized changes to persistent data into a database and alteration of data in a flow between computers on the internet. Since our app has some focus on the data that the users are submitting to the database tampering with data will be one of the more relevant aspects of our app. If maps and routes would be alternated the users would face inaccurate and misleading information and even tricked to take a route or path alternated by the intruder.

Repudiation threats are associated with users who deny performing actions without other parties having any way to prove otherwise. This happens when a user performs illegal actions in a system that can’t trace the operation. This threat is not relevant for our app since there will be no direct communication between users. And since the app focuses on describing and commenting on routes rather than direct chat between users this threat is not of prioritization.

Information disclosure involves the exposure of information to individuals who are not supposed to have access. This is when a user has the ability to read a file that they were not granted access to or read data in transit between computers. Because the CityWalk app tracks the user's routes in the database information disclosure is a relevant threat. Like with the tampering with data the information disclosure would reveal sensitive information about the user's location and therefore also a prioritized threat.

Denial of service (DOS) attacks deny service to valid users by making the server temporarily unavailable or unusable. In case of a server crash the database would be unavailable and this would make the CityWalk application completely lose its functionalities since most of the data is stored on the database. For the user that can mean an interruption in the walk. The user then has to wait for the server to start again before continuing the walk.

Elevation of privilege is a threat were an unprivileged user gains privileged access and has the possibility to destroy the entire system. It includes the situation when the attacker has penetrated the system defense and becomes part of the trusted system itself. Our app will only have a regular user and not an admin therefore an intruder cannot gain privileged access.

10.2 Potential threats and solutions

From STRIDE we identified possible security and privacy threats for our app. We have compiled a list of the threats that would affect the users the most. These are the threats that are most important for us to address and how these can be reduced.

- **Spoofing identity** is one of the threats that can affect the user the most. If the intruder finds the user's password it can have substantial consequences for the user. Therefore, our app will use Feathers authentication and encryption of the passwords.

- **Tampering with data** involves great security measures to be implemented to the database so that no unauthorized people can access sensitive user information. This could be done by a software- and hardware based security solution.

- **Information disclosure** is important for our system to maintain. It's imperative that sensitive information like the user's location is kept secure. This could be done with encrypted variables before sending these to the database and the other way around.

- **Denial of service** (DOS) attacks could be a serious threat to our application and we have come up with a few ways to deal with this. Firstly, our server is set up by a company called DigitalOcean who constantly monitor and react to such attacks and have the ability to handle traffic spikes. Secondly, we are using a production process manager, called PM2, for the Node.js server application. PM2 comes with a load balancer that makes the application able to handle a lot of traffic and automatically restarts the server application if the server crashes or reboots.

10.3 Ethical issues

When it comes to the retention of private user data we at CityWalk have a responsibility to evaluate and motivate why to keep or not to keep personal data of the users. Discarding or deleting private user data too soon would obviously result in problems for the users still using the application. Another problem when deleting data too soon could be of disadvantage of features in the application built heavily on user interactions that uses information from user. Keeping private user data for too long could increase that the information to go out of date, reliability of the information and the difficulty in accessing and retrieving data that is not needed. To ensure we have the right and enough of data from the users the time keeping the private data needs to be reviewed. Also there needs to be a purpose in retaining the data and a consideration why we need the information. One purpose to do this would be to improve the features and algorithms and by doing so getting a better application for the users. Another purpose to mine data would be to present statistics and charts regarding the routes to the users. Finally, we would need to implement a secure method of updating and deleting private user data.

11. Evaluation

This chapter will contain an overview of how the created functions was tested. Partly while developing the application and testing it, bit by bit successively. And partly by testing the whole application, a usability testing was performed. This chapter will also contain a evaluation of how the app would perform in the "real world", if it's scalable and which changes would have to be made. This will be presented later in his chapter.

11.1 Testing

During the application development we tried to test the functions that was created at the current time. Below we are presented how we test our experiment both in front-end and back-end.

11.1.1 Testing protocols

Front-end

During the development we implemented continuous testing, every time a new function was created or something was changed we ran the emulator in the browser and tested the new part as well as features had obvious relation to the changes. To get an insight into the JavaScript functions, we used console logs in various sections of the code.

We should have built the app to the phone on a regular basis. Before usability testing we had not done that in a while and some of the features that worked on the emulator did not work properly on the phone. One such issue was that the Cordova plugin Geolocation used enableHighAccuracy as a default setting. With this setting enabled the application attempted to triangulate position using several methods, one of which only worked properly when you had a stable internet connection. So it worked perfectly on the emulator, but when the phone was used the position changed without any kind of movement.

When building an application of this scale, or any application for that matter, code hygiene and *modularity,* separating functions and calling the same one for one specific task throughout the entire application, is of great importance. As discussed in the Front-end

development chapter, this would simplify the development not only in the sense that we would not waste time rewriting functions and styling but also help in the search for bugs. Sadly we only managed to do this to some extent, there are a lot of similar functions created in parallel throughout the quite large controller.js file. In addition, the HTML and CSS are to a great extent intertwined.

Back-end

Once a function or a service was created during development of the api we used Postman or a similar service to test different kind of requests to that part of the */api/**. In part to make sure that the, for example */api/routes* was working properly in all imaginable situations but also to test if restricted information could be accessed. The client (application) was not yet created so we tried to anticipate the demands of the final product as soon as possible. As the backend had to be modified every time new data needed to be stored ouch a different structure of already committed data. Once the first version of the app was created the demands of the api escalated. Suddenly we had two possible directions to take when we encountered a problem. Change the structure of the backend or handle the problem, often represented as data on a not-so-ideal format, on the client. When these problems started occurring we realized, as we had predicted from the start, that once the functions on the app were created we would be "looked" to that version of the api. To save ourselves from more possible headaches in the future we decided to take some extra time to overhaul our API and database and created version 3.0 that stands till this day.

As mentioned before the main server for the API is in Frankfurt, that version is connected to our git master-branch for the back-end. So the idea was that whenever new development was to take place it would first be tested on the local environment, your own computer. In form of a new branch that then, when the new changes have been made and tested, a merge with the git master would take place with the differences with a local and public environments ignored. We then make a git pull on the Frankfurt terminal, reinstall the the dependencies (npm install) and reboot the server. Ideally this deployment to the public server would be done automatically, to have some standard procedure that simplifies

testing and development. The local machine should also utilize the same software as the deployment terminal, either through a virtual machine or a separate Linus computer for local testing. All this to eliminate, or at least's rule out, some possible causes of errors in deployment. Sadly, these protocols were never fully utilized, at the time when these were formulated focus had already switched to close to full scale front-end development.

11.1.2 Usability testing

By organizing a usability study, usability issues, bugs and other faulties with the application can be found. Usability testing with the evaluators is needed to be done since the application should be as user friendly as possible and will help us fulfill our developing.

The usability assessments were done in week eight of our project. Five people who matched our intended user groups tested a beta version of our app. The tests were done outside in different areas in Uppsala. The test sessions will be performed by using concurrent verbalization protocol, where the evaluators need to think-aloud and let us know what's going on in their mind while using CityWalks application. The evaluators are then asked to describe their thought process while navigating through the app. Two members of our group was present at the assessment. One member instructed the evaluator and one member took notes. During the session the evaluators was asked to perform tasks:

1. Create an account and log in
2. Record a route
3. Comment, recommend and save a record route
4. Walk a Top route
5. Comment and recommend Top route
6. View comment from other user that uses a route in Top route
7. Create random route with preferences
8. Log out and log in as "test"
9. View friends and delete friends as "test"
10. Log out
11. Reset Password

Afterwards the evaluator was given a System Usability Scale (SUS) questionnaire and grade each statement will range from 0 to 4:

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think I would need the support of a technical person to be able to use this system
5. I found that the various functions in this system were well integrated
6. I thought this system was to inconsistent
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with the system

For statements 1,3,5,7 and 9 the score contribution is the scale position minus 1. For statements 2,4,6,8 and 10 the contribution is 5 minus the scale position. Multiply the sum of the scores each user has graded by 2.5 to obtain the overall value of SUS scores have a range from 0 to 100.

10.1.3 Test group

The individuals that have participated in this evaluation has tried to be matched with the typical personas for future users of the system. These five individuals are presented here with a short description of their demographic data such as their occupation and hobbies, and why they would appreciate the CityWalks application.

Belinda Pettersson Rimgard 24, Uppsala, Ph.D. student:

Belinda is a Ph.D. student at Uppsala University living in a student corridor with two other friends. When not working Belinda likes to visit her friends who live in different towns. When Belinda is in these different towns she often takes time to explore the city while her friends is at school or work.

Karl-Gunnar Fejes 59, Uppsala, engineer:

Karl-Gunnar is an engineer living with his wife and daughter in Uppsala. He is traveling to Stockholm for work a couple days a week and he likes walking, either with his wife while home, or by himself when staying at hotels in new cities. He is very time pressured when he traveling in work and he sometimes relaxes through going for a stroll after work. He often asks for ideas from colleagues of routes that is walking friendly.

Ann-Christin Fejes 57, Uppsala, teacher:

Ann-Christin is a math teacher in Uppsala living with her husband and daughter. During the weekends she likes to walk with her family for fresh air, exercise and also during lunch at working days. She appreciates walking in new cities to see attractions as well as in her hometown in nature.

Maria Lindén 23, Uppsala, student:

Maria is studying to be a teacher at Uppsala University. She likes moving her body when not studying or socializing with friends and often takes a walk in the city area. She participates in a lot of activities at student nations and she is very outgoing as well as active on social media. In between all the parties, events and studies Maria loves to go for walks to get some fresh air, but she struggles a bit because she always walks the same boring route.

Ola Bengtsson 72, Stockholm, retired:

Ola is a 72 years active old man living by himself in Stockholm. He is often going on weekend holidays to European cities visiting his family and friends, often staying at hotels. While visiting the cities he often explores the town's attractions by walking and also likes to visit parks with their strolls to pass lakes and natures. He is also extremely interested in history and has visited most of the museums around the world, but he is eager to visit more.

10.1.4 Testing results

The results of the usability testing showed some functions and parts of the system that need to be improved. In this part of the report these needed improvements will be presented as well as how we already have improved some of them. The evaluators understood that our

application is an activity application when first seeing the login page and this means that we managed to develop a design that appeals to the idea we have. We fixed some bugs and faulties to make it easier to navigate. We tried to find where the users would need any information while using the application for making it easier and user-friendly. The results of usability testing and what we fixed is presented below.

- One issue that most of the test participants encountered when testing the app was that the phone's keyboard was hard to disable when the users wanted to see information behind it. This issue was probably partly connected to how familiar the evaluators were with the Android operating system that was used in the test, since 4 out of 5 evaluators had almost no experience with the Android operating system. We solved this issue by implementing scroll on the pages where this happened so that the hidden information easily can be accessed by scrolling down on the page.
- Another thing we've changed after the usability test was the placement of the options in the side-menu of the app. We regrouped these options according to their functionality and we believe that this will increase the time for the user to understand the side-menu options as well as making the the options functionality clearer.
- One of the evaluators found it difficult to operate the toggle that we were using for a "Recommend route"-function. To make this function more user-friendly we changed the toggle to a checkbox that is activated by clicking on a green circle. Similar other applications that has like-functions.
- Also since many evaluators found the "Map and GPS"-functions a bit hard to understand and quite buggy, we have implemented some additional information to help guide the user and also improved how the GPS records the position and how the map is show. It will make users more comfortable while navigating in application.
- One of the evaluators deliberately tested the system when he wrote two different passwords when creating an account. The app should double-check the password when creating a new user, but instead the app created an account for him even though the passwords didn't match. For usability sake there should be a function

that double-checks the password so the users don't accidentally write the wrong password and then later on can't log in to the application.

- After the evaluators had walked another route they noticed that they were able to recommend a route several times. We fixed this by checking if the user already had recommended the route before updating the comments JSON object. In addition to this issue, some evaluators found that they could send empty or very long comments. This issue was solved by requiring that the comment is not empty and restricting the comment to 120 characters.
- One user did not recognise the format was used for the route time and the data on the page for additional information about the routes. This issue was solved by using a standard format for the time (hh:mm:ss) and for the date (yy:mm:dd).
- The evaluators wanted additional information about the "Create route"-options, their functionality, how it's generating the random routes and how the preferences work. This could be done with a popup or other notification that explains the different choices the user can make and what those choices these might result in. The preference list that the user can chose attractions from would preferably be dynamic and update the preference list according to where the user is at the moment and only populate the list with attractions that exists nearby. Time and distance of created and random routes is merely a bug that will be fixed. At the moment the shown time and distance is shown but appears far later after the routes have been created.

Usability testing with the evaluators that matches the typical personas for future users of the system went overall very well without any bigger problems. The average score counted to 81 on range from 0 to 100 on System Usability Scale, this means that our application is user-friendly and easy to understand. The result of usability testing according to SUS is presented in *Figure 10.1* and calculation of the System Usability Scale score is presented under Appendix.

Evaluators	System Usability Scale score
Belinda Pettersson Rimgard	87,5
Karl-Gunnar Fejes	77,5
Ann-Christin Fejes	80
Maria Lindén	90
Ola Bengtsson	70
Average	**81**

Figure 10.1. System Usability Scale score by each evaluators

We can summarize that the functions were working as we wanted and that the evaluators were able to figure out how they should use all different functions we created. We also saw that younger evaluators got higher score than older evaluators, this can depend on more technical experience for the younger evaluators.

10.2 Launching CityWalks

To be able to launch this application to the open market, some changes would have to be made. At this time the application relies heavily on crude local storage functions, this approach gives us some flexibility concerning connectivity but restricts, in some sense, the refresh rate in some sections. For example, new routes created by another user won't necessarily be displayed instantly in your list of routes. When the app launches these functions need to be smarter and work in a more dynamic way, alternatively work as a backup if the user's internet connection is unstable. The performance benefits of having logic performed on the client are lost if it has to handle too large datasets, as it stands now it might even require to much from the client. However, a lot of functions that are built on the back-end are designed to smoothly accommodate further development and a node.js API is designed to be easily scalable to meet rising demands. Therefore, to use the back-end instead of local storage could make the application perform better in a bigger scale.

The intention at the beginning was to develop CityWalks for both Android and IOS. But at this point testing has only been made on android platforms. The ideal scenario would be to launch the application simultaneously on several platforms to limit the cost of marketing and further capitalize on the momentum of the launch. Therefore, more time to perfect the application to the IOS platform would be necessary.

At this point only small scale testing has taken place. Even though most of the functions, with the exception of those mentioned above, are designed to meet required standards. Larger beta testing needs to be performed to get a better overview of the entire system's performance, for example problems occurring with different OS versions and screen sizes. This would also be a good way to discover bugs and other flaws within the system that need to be improved. Smaller changes that would be made if we were to continue with the project will be presented in *General improvements* further down in the report.

12. Conclusion

During ten weeks of hard work, the CityWalks application has been developed and become a functional and innovative system for people that likes walking. Since many of the team members had little or no knowledge about how to developed an application, all team members has gathered a lot of new knowledge and learned many new things. The limitation of ten weeks also meant that we did not have time to include some of the features that was initially planned to be included in the CityWalks application. In this chapter the lessons learned of the CityWalks project will be discussed, as well as how the not included features could be added in a future development the CityWalks application.

12.1 Further development

Since the development of the CityWalks application and the creation of this thesis was limited to ten weeks, we were quite time-pressured and had to focus on the most important features. This also meant that we had to leave out some features that we wanted at the beginning and some that would be good to have for a production-ready application. In this section, we will describe the things we have not implemented, either because we have not began working on it or the function is somewhat done but not ready to implement yet.

Security and privacy

One feature we wanted from the beginning was communication directly and privately between users in the community. This function is common in applications with similar functions as ours and would give the users the opportunity to ask questions not suitable for the comments section directly to other users. This feature would also make it easy for users to meet up and walk routes together. Another feature, connected to privacy, that we wanted to implement was to add the option to have the route privately stored. This would mean that just you and your friends could see them. A function like this would also involve a feature that lets you confirm a friend-request. Right now the friends -function functions more as a follow-function of other users. However, adding these functions would also increase the problems often associated with privacy on the internet. Since this data would

be sent to our server and stored in our database, we would be have to make sure that the security of the data is at the highest standard possible for a mobile application.

Another function that should be implement is *reset password* since it's important that one user can reset their password. Although the required back-end structures are in place other pressing matters forced this function to be unprioritized and therefore was not developed in the end.

A solution to the security issue with private information described above, would be to use a third-party service that uses encryption of its messaging services and have large teams constantly improving and keeping the security. We know that there is support for integrating some of the major social platforms, some with encrypted messaging, into our application. As well as increasing the privacy, integrating a major social platform would increase the CityWalks applications visibility and make it possible for the users to share their routes with a much larger community.

Economic aspects

Future development would also involve the creation of strategies and implementation of functions for economic reasons. Firstly, to manage the payments and income we would have to register a company. As discussed in chapter 2.4 in this paper, there are a couple of options to make CityWalks profitable. One option is to add special waypoints that is always used in our "create route"-function. Businesses could then in theory pay for these waypoints to increase the business visibility and the number of people passing by it. Another function that we wanted to implement was a simple commercial banner or popup, seen on many mobile applications, on certain pages in the CityWalks application. Another way of earning money from CityWalks in the future development would be to have free and a non-free version of the application. Since there is a daily limit to the amount of requests we can send to the Google Map API, the free version would preferably be limited to a certain amount of walks or recordings per day.

Visualizing data and statistics

A feature and function that visualizes data and statistics for the users would also implement in a future development of our application. This is common in many exercise-applications and would give the users instant and comprehensible information about their activity, both for the social aspects (comments and sharing) and for the walking aspects (total distance and time walked). Implementing such a function would also give us the opportunity to add a page with statistics comparing user's activity, showing users with many recommended routes and similar features that would improve the user's experience and safety.

Algorithms and Maps

There is a lot of different aspects with the algorithms that generates maps. In general, the algorithms would preferably be more sophisticated and considering more parameters. Some of the further development opportunities would involve more accurate level options, given that no preferences are close the option, the list would change along and the created routes would take into account more data, both from Google Maps API and CityWalks own API. Also when walking the created routes, the map would preferably be more interactive and responsive to the users movements. For example, the maps would give feedback to the user regarding direction of the route and most preferably erase or blur the part of the route that already have been finished.

General improvements

In the future development we would also need to improve some of the already created functions. One example is how the information downloaded from the database and then is displayed on the application. At this stage, on the "Find routes"-page all routes are displayed. This is not a problem at the moment with less than 50 routes in the database, but if we would have a great number of routes this page could possibly be much slower. To prevent this specific issue, the information could be separated in different pages instead of being displayed in a long list.

Furthermore, we would add a *delete* function to the routes, making it possible for a user who created a route to also remove it. We would also add a *delete* function to the comments so a user can remove a comment they wrote. These are two important functions to have for the usability aspect. Partly because there is a freedom that one should have as a user and partly for reasons of confidentiality. There will be situations where the user's comment is in inappropriate, therefore a delete function should be in order for the user to be able to regret it.

Besides this, more testing of the application in general would be done. We would for instance try to make illegal and unwanted changes to the database, and also testing the limits of the core functions in the application. By testing the application with a greater number of evaluators, we would also get a better idea of what people in general find useful or dislike about our application. This would probably also give us new ideas on additional functions and features.

12.2 Lessons learned

Project collaboration

All of the team member had been working in project before since it's everyone third year of the study time. The size and length of the project however was bigger than anyone in the group had experience before. That cause for new ways of working and new methods to be learned.

The way of working shifted a lot during the project and it was not possible to completely plan the project in advance. Also a lot of group members had different time of availability and experiences of knowledge. That required flexibility and for the group to be seen as dynamic, something that have worked well and also created a useful lesson. Also the large number of group members and the fact that not everyone has been able to be gathered daily have required clarity on group decisions, the progress and the work in the project.

As mentioned in the Method chapter the work with the Scrum methodology have been a challenge for the group. This could have been dedicated more time to learn in the beginning

which would have facilitated the progress with the project, since no one had been working with that method before and it could have made the dividing of the work more efficient. Besides that, some logging could have been used to catch up with everyone's work and keep track of were doing at the moment. For example, everyone could have kept a daily or weekly log and every week could have been ended with a meeting with presenting the work.

The division of roles is also something that have been rather unclear. The initial persons that was supposed to work with front-end versus back-end have been completely mixed up during the project. That because the knowledge and time spent on different part was very varied between persons in the group. The roles could have been a little more dynamic and be changed over time as the fit for everyone was naturally clear.

The social aspect of the work has been working quite well. The communication with the teaching staff have worked well and they have constantly been around and answered questions. Of course it has been puzzle with the people having all kinds of other things to do, such as work and other courses but the atmosphere have been enjoyable and time have been spending on team building activities such as after work and group lunches.

Design and user interface

The project has also offered a new way of looking at system design and usability thinking. It's a balance to choose between what is felt to look good and what is easy for everyone or a special target group to use. Many of the team members really liked the background picture in the form of a photograph. After the usability assessment however this had to be removed due to accessibility aspects, which is an example of just that balance.

It has also been educational to use the other aspect of design, for example when using icons in the application to make more distinct functions gave some new insights on design thinking. The choices of colors in the product is also something that have been chosen not

only because of it’s nice looking but also to fit in on the concurrent market and to attract right target group of users.

Front-end

As no one in the front-end team had developed an application before the project, the front-end team members had to learn many new things before starting the front-end development. In the first two weeks, front-end team members dived into the documentation of how the Ionic Framework worked and how to create an Ionic application. This means that the front-end team now has deep knowledge about Ionic and the AngularJS functionality.

Since the front-end team also worked in collaboration with the usability and back-end team, the front-end had to learn and understand much of the other team’s respective work. This meant that the team learned the importance of an effective teamwork and that is can be crucial to give clear explaining and specifications on functions that affect several groups work. As mentioned in the Front-end development, the collaboration also resulted in some disadvantages, such as a lot of repetition of JavaScript functions and HTML styling. We therefore learnt the importance of to emphasize modularity and code hygiene from the beginning of the development, and constantly separating the CSS, HTML and JavaScript code.

Another thing that the front-end team learned was that the different social features was hard to develop. Many times the front-end team were satisfied with the social features from the beginning, thinking that the feature looked good code-wise and visually on the emulator on the computer. But it happened that the team got blinded by its own creation and overlooked issues that affected the usability. The team therefore learnt the importance of evaluating the features and exchanging ideas with people from outside, and that getting new inputs can radically change and improve the functionality and usability of features.

Generating all the maps that CityWalks uses have given the front-end team useful insight in how external API's work, handle JSON files and discover all the features that Google Maps

have to offer. From experimenting and creating maps with Google Maps API the front-end developers have been able to successfully achieve the initial plan and gained further understanding of how different functions may be assembled and applied in the right context.

Back-end

At the start of this project the first decision was what kind of application we wanted to make, to form that decision we had to estimate what was possible, plausible and reasonable without almost any knowledge of actual development. Once we agreed on the CityWalk idea, we were faced with the problem of choosing the right, or best suitable, path for us. The options were endless and at a glance they all had their "pits and perks". Because some of us wanted to develop their web development skills, we were all quickly into the idea of a web application. To avoid large difficulties down the line we spent some time getting to know the different "stacks" MERN, MEAN and variations of these. In hindsight it is hard to make uninformed decisions, and these can be critical for the future, ergo it is important to do a lot of groundwork before committing to a decision.

We have learned that the front-end is heavily dependent on the functions made in the back-end. A lot has to be rewritten or compromised if an mistake in a back-end function is allowed to become part of the front-end core.

After we reconstructed the API and got a proper structure, we saw the benefits of having a high degree of modularity. This helped us a lot when an error occurred since all functions were isolated. Therefore, we knew with near complete certainty what part of the code where erroneous.

Whenever we encountered a problem we now know that the solution is almost always, in some form, a search or question away. There is probably someone out there who has faced the same problems as us before and know how to solve it. Consequently, we have during this project searched and found a lot of answers to our questions on the internet. This also

extends to complete functions, there are frameworks and application shells that handle basic commonly used functions. Once the answer is found is only a matter of modifying the code to our needs.

12.3. Final words

The Citywalks project have been overall successful in the team's opinion and an educational and fun experience for all group members. The responses and reactions of the idea and the product developed have been over any participant’s expectations. Of course there have been challenges in different ways but the project has been finalized and the desired goals have been reached. The team is very proud of the result when it comes to the work itself as well as the product and will move forward from the project with a whole lot of new experiences useful for the future.

13. References

Internet

1. Gerald Redmond, Historical Aspects of Fitness in the Modern World. [cited 2017-05-26] Available from: http://www.nationalacademyofkinesiology.org/AcuCustom/Sitename/DAM/129/TAP_21_PhysicalActivityInEarlyandModernPopulations_03.pdf

2. WHO. [cited 2017-05-26] Available from: http://www.euro.who.int/en/health-topics/noncommunicable-diseases/diabetes/data-and-statistics

3. Mayo Clinic. [cited 2017-05-26] Available from: http://www.mayoclinic.org/healthy-lifestyle/adult-health/expert-answers/sitting/faq-20058005

4. Karin Kratina, PhD, RD, LD/N [cited 2017-05-25] Available from: https://www.nationaleatingdisorders.org/orthorexia-nervosa

5. Venturebeat. [cited 2017-05-25] Available from: https://venturebeat.com/2016/02/12/runkeeper-acquired-by-sports-shoes-giant-asics/

6. Scrum. [cited 2017-05-25] Available from: https://www.scribd.com/document/35686704/Scrum-Guide

7. Visual Studio. [cited 2017-05-27] Available from: https://www.visualstudio.com/vs/cordova/
8. Visual Studio. [cited 2017-05-28] Available from: https://www.visualstudio.com/vs/whatsnew/
9. Brackets. [cited 2017-05-26] Available from: http://brackets.io/
10. Ionic Framework. [cited 2017-05-28] Available from: https://ionicframework.com/docs/v1/guide/preface.html
11. Ionic Blog. [cited 2017-05-28] Available from: http://blog.ionic.io/ionic-creator/
12. Mozilla Developer Network. [cited 2017-05-27] Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
13. Mozilla Developer Network. [cited 2017-05-27] Available from: https://developer.mozilla.org/en-US/docs/Learn/HTML
14. Mozilla Developer Network. [cited 2017-05-27] Available from: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/How_CSS_works
15. Mongo DB. [cited 2017-05-27] Available from: https://www.mongodb.com/what-is-mongodb
16. Mlab. [cited 2017-05-27] Available from: https://mlab.com/
17. FeatherJS Documentation. [cited 2017-05-28] Available from: https://docs.feathersjs.com/

18. AngularJS Documentation. [cited 2017-05-27] Available from: https://docs.angularjs.org/guide/introduction
19. AngularJS Documentation. [cited 2017-05-27] Available from: https://docs.angularjs.org/guide/introduction
20. DigitalOcean. [cited 2017-05-28] Available from: https://www.digitalocean.com/products/storage/
21. Trello. [cited 2017-05-28] Available from: https://trello.com/about
22. Slack. [cited 2017-05-28] Available from: https://slack.com/is
23. Github. [cited 2017-05-27] Available from: https://github.com/about
24. SourceTree. [cited 2017-05-28] Available from: https://www.sourcetreeapp.com/
25. Google Drive. [cited 2017-05-27] Available from: https://www.google.com/drive/
26. Tutorials Point. [cited 2017-05-27] Available from: https://www.tutorialspoint.com/mongodb/mongodb_relationships.htm
27. GeoJSON. [cited 2017-05-26] Available from: http://geojson.org/geojson-spec.html

Code

- Back-end: https://github.com/Teddywickstrom/CityWalk
- Front-end (initial): https://github.com/MathiasJFejes/CityWalksV1
- Front-end (final): https://github.com/MathiasJFejes/CityWalksFrontEnd

14. Appendix

14.1 Reflections – Individual assessments

In this chapter, reflections from all group members will be presented. Each group member will describe what roles they had in the work, what they did during the project, what texts they have written and what they have learned during the project.

14.1.1 Markus Elfving

On the startup meeting in the first day of the project I performed my interest of being project manager. Since no one else wanted the role the group decided that I could be chosen to start as PM. However, we agreed that the roles could be dynamic and therefore changed over time, so that no one would feel that their role were set in stone, or that they missed the opportunity for some role. Besides the role as project manager I also became a part of the front-end team together with Johanna and Nathalie, but we worked a lot with the persons that worked as combined front-end and back-end, and everyone in the team since we overall were pretty mixed up in our daily work. Some of the the team members spent a lot of time finding out how to build specific design in the framework, some that worked with the functions helped a lot to implement them in the front, and everyone in the team took part on the decisions for the visible design such as colours, logotypes and other graphic. The three of us however focused on the visible parts of the development. The first week was dedicated a lot for planning and specify what we wanted to accomplish and how that could be built as an mobile application. After that we started to create mockups for how we wanted the application to look like, and to have something to work for in the developing part. When we started the actual code building we started to use the web system Ionic Creator, which allowed us to build the ground with side menu, login fields etcetera in a simple drag and drop-like way. When we were satisfied with that part we exported the code to visual studio where we could edit and add code, and emulate the application by using a Cordova plugin.

When it came to my project manager role, I saw that as an opportunity to try my leadership skills and to try having a greater responsibility, which could inspire me and make me really dedicated to the project.

I saw my part as being the one trying to organize the work and make sure the project moved forward. Furthermore I thought I should be the convener and handle the priorities of time and resources.

The part of my role that I thought was the hardest to perform right was the decision making. Since we did not have any clear guidelines or policies in the group whether I should make decisions or not, and it was not stated in the course either, it resulted in me not taking any decisions at all in the project. Also of course I had no formal position so that I could make sure that the planning was followed by the group. This leads me to the scheduling and planning of the project, which is the thing that I think was my biggest failure in the PM role. As mentioned in the report we tried to use the Scrum method to split the project down in sprints. I did not succeed as a scrummaster however to formulate sprints in an appropriate size or concreteness, something that I thought was hard to achieve. The reason for that was probably that I, together with the rest of the group, had to little experience working with this method, and looking back more time should have been spent learning this method.

In the beginning of the project it was clear that some members had a lot more experience and knowledge when it came to the programming, why this time in that area was spent by me mostly trying to keep up and understand all the parts of the ground building and setup of the system. In my role I also wanted to have an overall picture of the project, so because of those reasons I also spent a lot time working on the deliverables to be able to contribute in some way, and I was part of writing about the requirements, the user profiling and the usability assessment, and was the one responsible in the work with the project plan. Furthermore I took a big part in the presentations, and was working as the main talker in the intermediate presentation, something I thought was suitable in my project manager role. In the finale thesis I have mainly been writing in the introduction part, the design part, the

front end part and the conclusions lessons learned, but generally everyone in the group have been working on all pieces in same way and that is the case also for me.

My main focus in the development was the front-end since I was in the part team initially chosen to work on that part. In that, we started the first week on design planning, creating mockups and to build a ground for the application, using Ionic Creator as mentioned above. This part was not that technical, which allowed me to get into the understanding of the coding in a couple of weeks, as a learning period.

When we later started to work on front-end of the application by coding and connect it with the database and API functions created of the back-end team I really started to understand the languages, the coding and the systems a lot more. With this I could start working with the design part and styling of the code and try different variants of choices and therefore contribute and make progress in that area. Often we worked in pairs, spending some day just working with trying to get a certain function work by searching the web, create script functions and implement them in the visible parts of the application. This really were a good way to getting forward with the development and also to get a deeper understanding in the same time, since it is easier to collect the knowledge you need when having a specific goal and on the same time having the people you work with sharing their experiences.

My general reflection on this project is that it is the most giving thing I have done during my university studies so far. It have allowed me to work on a real life example in the very sam way as it would have been done outside of school. The thing that I have appreciated the most is just the freedom given in the course, that we did not have any boundaries on what languages or system to use, but only got a requirement on what the system would do. It have also been a really nice teaching staff which have had a close communication with us students, been offering help when needed and really been a great resource.

The thing that could have been better in the course is the group sizes. Since we were eight persons it was hard to gather everyone continuously and make sure everyone could follow

the process and move forward with decisions. If the groups had been smaller, it would have been easier to synchronize as a team. In this project, when people had been away we needed to spend time to learn each other on what had been done while gone and which decisions that had been made, and sometimes discuss the decisions again if there were something that was disagreed upon. There is also where my biggest lesson learned lays, that I should have made clear in the beginning of what my project manager role should be. Doing that it should have been specified whether I should be taking decisions or not, and if I would, which ones? That could have saved us a lot of time, getting forward with design issues or dividing of the work, instead of having many people changing the same thing over and over, or to many people working on the same assignment. Furthermore I could have spent more time learning project managing methods such as Scrum, and how to formulate appropriate sprints.

To summarize I have really gotten some large experiences from this project. I have learned the different difficulties of trying to plan and manage a project of this size and the challenges of getting the dynamic to work between eight different individuals. Also the thing that have been really giving is the chance the project has offered on how to really create a complete system that can be used for real. I have learned from scratch on how a mobile application is created and been able to combine the different experiences I have gotten from previous courses in my education which have been really educational.

14.1.2. Mathias Fejes

My role in the project is kind of hard to define since it has changed several times, and also since I have worked on tasks involving almost all the major parts of the project. During the first week of the project we decided the roles and I chose to be a part of the back-end team. This choice stemmed from that I had worked with front-end programming and design tools earlier, and I therefore wanted a challenge and to learn something new.

But after only a week I my role started shifting towards usability with designing the CityWalks logo in Adobe Photoshop and to front-end by setting up the first basic Ionic application. After this my role gradually moved to front-end development since I started to implement design choices, functions and some basic features with Ionic Creator. At the same time I continued with the back-end setting up the DigitalOcean server computer and also by being involved in development of the mongo database and getting the server API to communicate with the Ionic application. I also had the role in the usability testing area, deploying and testing the application on a physical mobile device. In addition to this, I handled the payments of Ionic Creator and DigitalOcean, and had the role as being in charge of the economic aspects of the project. So my role constantly changed during the first five weeks and my role turned somewhat to a middleman, contributing to many areas and being a communication link between different teams.

After the fifth week, the mid-presentation, my role turned more strictly to a front-end developer since I became responsible for the Record route feature and started working on styling the HTML pages in collaboration with the usability team. In addition, I had the role as a Security expert and looked over the security aspects of the system and the application. In the last four weeks, my role once again became diverse since I worked with all the teams on styling, creating features and helping with some back-end tasks.

To sum up, my role in the project was mostly as a front-end developer that a the same time worked extensively together with the usability team and the back-end developers.

As mentioned earlier, my role in this project were quite diverse and thereby I have worked on many different areas and tasks in the project. The general work that I have contributed to is for example looking over the security aspects of the system and the application with Ilkka and taken care of the payments to different systems. In the first and second week I also worked with installing all the different softwares that we needed, helping others to set up Windows and then Visual Studio with the Cordova extension as well as the Android Studio. It should be mentioned that much of my work has been centered at reading and going through documentations, instructions and tutorials for all of the different tools and code-languages used in the project.

My work also included some usability and design development. Together with Markus, we created the design of the currently used logo for CityWalks and I later created the icons that are seen on the physical mobile device. Much of my work had to do with implementing and styling the different features and HTML pages of the application with CSS code, often together with the usability team. In the early stages of the development, I worked with Ionic Creator and through it tried to give the application a clean and good-looking interface. When it comes to the usability work, I tested, together with Saranya and Malin, the application on three different people from outside the project and helped with the compiling of the tests.

In the back-end development I worked on several tasks, mostly in the first four weeks of the project. In the beginning I participated in setting up the mongo database and the initial server API. I also added Ionic Auth to the application that for a while took care of the authentication, stored the user information and a way that reset the password via e-mail. I also created a Digital Ocean account, set up a cloud computer droplet, installed node.js on it and then deployed the server API code on it. In addition, I generated an encrypted SSH-key for the server and setup the software PuTTY and WinSCP from which the server could be easily controlled and updated.

As most of my work focused on tasks in the front-end development, it's here where I think that I contributed the most. In the first weeks I worked with first understanding the Ionic Framework and then setting up the initial Ionic application, which the CityWalks application is based upon. This also meant the I had to work hard with learning and getting comfortable with the syntax and functionality of AngularJS, JavaScript, HTML and CSS. When it comes to the different features and functions of the application, I have developed and contributed to many of them. In the very beginning I worked on the initial Login, Signup, Forgot password and Create account features. At the same time, I also worked on setting up the side-menu with HTML pages connected to it. When this was done I worked on getting the end-to-end connection to work and to initially add information from the database to a list on a page in the application. This function was then refined so that it properly listed routes from the database. The next thing I worked on was to take data from only one of the items in the list to a new page. This function was after that refined, so that the second page showed all the data in a user-friendly way. In the two last weeks I worked on and created the Nearby routes and Find routes features. In addition, I worked on creating the My routes feature. When it comes to the map features, I worked on creating the Record route and Walk route features. More generally, I also worked on implementing the Cordova plugin Geolocation and the Google Map API to the application, from which maps could be added to the HTML pages and so that the functions could find the position of the phone. I also helped Ilkka with styling the Create route feature. When it comes to the social features of the application, I worked on creating the Comment, Recommended and share features.

My contribution to this thesis begins with the general introduction about social applications for physical exercise and walking (Chapter 1). After that, I have written about the technical solutions for the front-end (Chapter 7) and together with Ilkka, about the security (Chapter 10). In addition, I have written about the usability testing and the following improvements and changes to the application (Chapter 11). The majority of my contribution to this thesis is in the chapter about the front-end development (Chapter 9). Finally, I have also in the conclusion chapter written the general introduction and about the lessons learned of the

front-end development, as well as about the future development of the CityWalks project (Chapter 12).

Looking back at the ten weeks of developing the CityWalks application, I find that I have learnt many new things and that this project was very rewarding in several ways. For example, I have attained new skills such as learning new code languages and gained a deeper understanding of the dynamics of a system development process. In system development, tasks often tend to take much longer time than expected and it is therefore difficult to estimate how long a task will take to finish. As planning a project thoroughly is key when working in groups, this sometimes proved to be slightly problematic. Similarly, I now understand that learning how to develop new features from scratch can be a very demanding and complex process. At the same time, all of it was worthwhile and not to mention fun, particularly solving problems. Another important lesson is how much you can accomplish with the help of the internet and especially the StackOverflow community.

Something that I have learned as well is the importance of emphasizing modularity and code hygiene from the very start of the development process. The repetition of JavaScript code and the intertwined CSS and HTML code was a simple and fast solution in the beginning, but later proved to slow down the pace of the front-end development.

One of the most important lesson though is the dynamics of a large project that involves many people. As the project become more complex, I found that it became more difficult to have an efficient and productive teamwork, since there were so many things going on at once. I believe that our group size was a factor that contributed to this and that it might have been easier with a smaller group. Another solution to this issue could been to have more and better structured meetings where everybody could get a chance to keep up with everybody's respective work.

Overall, to work on single project for ten weeks unlike the usual study mannor was very interesting and fun to try. I feel that this project has given me valuable insights into system development that can probably be helpful and useful in the future.

14.1.3. Johanna Fyrvald

In the startup phase of the project one of the first things we decided was the different roles in the group. We decided that every team member would be either a front-end or a back-end developer as well as that there would be one person who communicated between the front and back-end developers. I stated my interest for being a front-end developer since I was interested in learning more about the design parts of developing an application as well as learn the languages HTML, CSS, Javascript and other tools used specifically for mobile application developing better. I was also assigned the role as one of the usability experts and focused therefore on coming up with personas, a usability test plan, use cases etc. I saw this as an opportunity to deepen my knowledge about the usability testing, since this was a relatively new subject for me that really interested me. The roles we set up in the beginning of the project were dynamic and a lot of group members changed roles over time between front- and back-end developing and so on. My role was also to get deliverables done and write on the thesis draft and the final report.

During the first two weeks I focused my time on setting up the computer, installing all the necessary software as well as learning how to use the different programs and tools such as Ionic, Trello, GitHub etc. This included watching a lot of tutorials online and reading documentation about the front-end tools. In the beginning we in the initial front-end team focused on creating mock-ups for the different screens in the application. Me, Malin, Natalie and Markus did the initial design mock-ups by hand. This was a really good way to picture the application and the design before actually programming it. The hand-created Mock-ups also made it easier for the decision making further on in the project, since we could always look back at the mock-ups and see how we thought about the design initially. Sometimes the functions in Ionic framework and Ionic Creator didn't match with what we had designed by hand, which resulted in that we had to change the initial design and mock-ups several times. We also started to create mock-ups in a program called just-in mind to get more interactive and professional looking mock-ups, but we realized that it was very time-consuming, and that it was a better idea to start the actual programming and make the

front- and back-end able to communicate with each other. That was a process that took much longer than planned.

I also worked together with Markus and Natalie in Ionic with the front-end design, we worked with designing the log-in page, the sign-up page, the reset password page as well as created and designed the sidebar with the icons. I am really happy with how the design turned out in the end, we also changed a few core design elements such as the background picture/colour due to the results of the usability testing. We realised that using purple as the main colour in the entire application made it more clearly and visible as well as easier for the users to see text and what to do on different pages.

I was also focusing a lot on getting parts of the first deliverables done such as D2: Requirements, D3: Project Plan D4: System Design and Test Plan and D5: User profiling. I also focused a lot on the D9: Draft thesis where I created the initial table of contents as well as parts of the report that later was used in the final report. The deliverables turned out to be really useful for the final report and we could reuse and rewrite a lot of the material and use it in different parts of the report.

In the final thesis I structured the initial table of contents as well as chapter 2. Socio-economic aspects, target users, Competitor analysis, Social and Economic aspects. In chapter 3. Method, I wrote about the team structure, created the overview figure describing the roles, the project plan in form of a GANTT Schedule together with the Project Manager Markus as well as the description of all the activities in the project plan. I also wrote parts in chapter 5. The user perspective, such as the intended users, some of the personas and the information squares about each persona. In chapter 6. Design and user interface development I wrote about the logotype. We all collaborated really well in the writing of the final thesis and the different minor roles focused on writing the parts that we have gained the most knowledge about, like for example the Security expert and Usability expert.

Overall I am really happy about the way our project and final result turned out. This ten week period was a really rewarding time and I have learnt a lot about how I function and work in teams, group dynamics as well as about the whole process of how to develop an application from scratch. I realized that there is so many different choices and decisions that

can be made and so many different tools available for creating a mobile application that I did not even know existed before starting this project.
I also learned how hard it is to follow a project plan and how estimating how long time different activities require is very hard to do. Initially we tried really hard to stick to our plan and different sprints each week. We had a lot of regular meetings, one in the morning and one after lunch sometimes. But when the project moved on the structure and the planning sort of faded out and everybody just kept working in their own pace and with the groups they were in. This shows that a project is very dynamic and the roles within the group as well as the time plan changes all the time and you can either accept that the structure is hard to keep up after a while or try really hard to stick to the plan and really make it work. It depends on the group dynamics and how many group members there are.

The size of the groups, 8 team members were a little bit too much. There was so much different opinions and personalities in the group so every decision making stage was very time consuming, since we really wanted to keep it democratic and let everyone speak up and discuss different options and ideas. So for next year, if the resources is available an idea is to divide the class into a little bit smaller teams maybe around 5-6 people in one group. I think that would help the organization and the planning s well as decision making within the group run more smoothly.

To conclude I have learnt a lot during this project and I really appreciate that the project was so practical and hands on, so that we actually developed our own IT-System instead of only writing a thesis. It was really useful to work so independently during such a long time, that really taught u show it feels to be totally responsible for a project and it was really rewarding in the end when presenting the final result.

14.1.4. Ilkka Mansikkamäki

My major role in the project was to act as combined front- and backend developer with a given minor role within security and privacy. The tasks within the role as combined front- and backend developer varied a lot during the time of the project. The idea of the role was determined to act as the middleman between front- and backend, client and server. Initially, and during the first half of the project my main focus was on deciding and choosing software tools and be able to create the end-to-end connection between client and server.

Later on and during the second half of the project my role shifted to more narrower tasks within different features, maps and algorithms within the application. Overall during the whole project my role have required to work closely and jumping back and forth between the front- and backend teams. The role and tasks within security and privacy stayed more static and was conducted alongside the whole project that resulted in a privacy and security report analysis of the application.

Since my roll was as a combined front- and backend developer my work and contributions are widely spread out in the project from client to server and database. Most of my work have been in research about software tools, creating the connections between client and server, setting up the CRUD (Create Read Update Delete) functions, generating maps, building algorithms and conducting a security and privacy analysis. In the early stages of the project a lot of time was spent on figuring out which software tools to use both in the front- and backend, which tools would work together within the entire stack and for this the MEAN- stack was chosen. Alongside with setting up the MEAN- stack other software, editors, a server and plugins was installed and compared to be able to set up and start the second part of the project involving all the maps and features. Once the pieces of the stack began to fall into place I started to work on the CRUD functionalities to get a end-to-end connection for the entire application. Even though the application didn’t have the right support or setup on client or server side for the CRUD functions at the moment these were created since these functionalities would later be essential for the application to run data from client to server. The CRUD functions were created and tested one at a time and initially

only with GET and POST. Also initially with small amounts of data and later on expanded in a variety off different functions and features across the application.

After the end-to-end connection was setup with the CRUD functions ready to use I moved in the second half of the project to work with generating maps and building algorithms generated together with the maps. More specific these were the maps and algorithms used in the *Create Route* by preferences such as level and types of places. It started off with learning, testing and applying the maps to our application and accessing the Google Maps API with a generated API KEY. Later on in the progress the maps were populated with coordinates from Google and with *Checkpoints* from our own database. Using coordinates from two different sources was something I together with the backend team thought was the best way to proceed mainly because of scalability. The three main algorithms that was build for the *Create Route* feature was a distance calculator between coordinates, closest coordinate finder and random coordinate generator. Creating the two last mentioned algorithms required cutting some corners given the limit amount of time, however the result was arbitrary enough for them to operate in a sufficient way a handle some of aspects that caused problems when generating the maps.

During the project i have been part of writing the deliverable Security & Privacy Analysis. In the thesis a have mainly been part of writing *4.1 Functional features*, *7. Technical Solutions*, *9.2 Map features*, *10. Security & Privacy*, *12.1 Further development*, *12.2 Lessons learned.*

I think given the broad role as a combined front- and backend developer is something that matched my work well throughout the whole project. I especially experienced this within the first half of the project when beeing part of setting up the end-to-end connection between client and server and generating the CRUD functionalities. Being part and involved in the beginning of the project with the end-to-end connection and able to work and collaborate both with both front- and back end I feel that I have learnt more about the whole stack from client, API, server to the database and also the programming language Node.js.

During the second half of the project I had more narrower tasks involving the features and algorithms but I still experienced the need of working as the middleman and with both front- and backend team members since these features and algorithms relied on both teams. Working with more narrower tasks such as features and algorithms I received more in depth knowledge in the programming language Angularjs.

In the project given my role, the various different tasks and the cross collaborative efforts to fulfill requirements and solutions I experienced my work and the team as whole to fulfill a agile software development process. I think this was achieved regarding the phase of the project and whether working on larger or more narrower tasks. Working in a agile way is something I have learned both as an individual and as a member of a larger team during the time of this project and how to track this process with project management tools like Trello.

14.1.5. Natalie Poli

At the start of the project we decided in the group who would take which role, i did not want to take that much place in the group so i decided to not take a special role, but for not missing the opportunity for some role we decided that the roles could be changed over time if someone wanted.

Even if we divided the roles as project manager, communication manager, etcetera, i think that everyone took responsibility and worked together as a group, which i think was really good.

Besides those roles in the group we divided us into front-end and back-end teams so it would be easier to work but we all, both front-end and back-end, worked a lot combined and together so everyone would be a part of decisions of the application, like colours, design, etcetera.

I was a part of the front-end team together with Markus Elfving and Johanna Fyrvald. I wanted to be a part of the front-end team because i wanted to learn more about developing but also to learn more about the different languages we used, like Javascript, HTML and CSS, for the application developing. But i did also do it because i like to play with the design.

The developments was our first focus after we created the mock-ups. All the planning about the application, how we wanted it to look etcetera was what we started with the first week, when everything was decided and the mock-ups was done we could start with the development and had something to work with. We started to use Ionic Creator for the code building which helped us with the base of our application. We used it to make for example the side menu, the login and signup side, etcetera. When we finished the base that we wanted we took the code and exported it to Visual Studio where we could change it and add other code.

Beside the developments we also took much responsibility for the different deliverables we needed to send in under the development and working time, for example the Project plan, User Profiling, System Design and Test Plan.I think that the deliverables was a good start to write because we actually could take much use of them when writing the thesis.

The first weeks we focused on installing all the programs that we needed, like Ionic Creator, GitHub, Visual Studio, Source Tree, Slack, etcetera. We also needed time to understand how the different programs work so we watched a lot of tutorials to understand and explained to each other. When we started with the programming we saw that the whole group did not have the same experiences, some of us knew better programming than others. I was one of them who needed time to understand some things but it was not that hard to understand while working on the development.

In the development my main focus was the front-end, as said before. We started to do the mock-ups on paper so we had a first picture of how we wanted it and after we started to play with the design in Ionic Creator, which i think was fun. We learned much about how to use Iconic and understanding the code. The mock-ups was a great start because we always had them there to look back at.

After creating the most, almost all sides like login, signup, reset password pages in Ionic we connected it with the back-end team who had worked with the database and API functions. The system was now easier to understand when everything was together.
After connecting them we started to play with the code for the design and tried different stylings till we chose the one we wanted. We changed the background several times until it was user friendly and fit the usability test. We did this in groups of 2 or 3 persons to understand and help each other, we shared our experiences and knowledge which was really good because we got deeper understanding. As i wrote earlier we also took much responsibility for the deliverables, for example the Project plan, User Profiling, System Design and Test Plan. And the last was the draft thesis and final thesis which we all worked alot with. As mentioned we could take much use of the deliverables while writing the thesis, but we also had much more to write and work with.

I have worked a lot with chapter 6, the design and user interface development but also chapter 4, the requirements, chapter 5, the user perspective, chapter 9, the front-end

development and chapter 12, the conclusion. The writing of the final thesis worked well and we all worked together as a team.

At first i will say that i appreciated the freedom we had in this course, that we did not have any special system or language we had to work with, i like that we could search and choose by ourselves. We had much help from the courses we been reading earlier at the university, so we had little experiences from before.

Something that could be better is the sizes of the groups. Sometimes it can be hard to work when the groups are too big, everyone want to make decisions and it´s hard to make everyone happy all the time. A big group is useful during the thesis writing but can be hard during the programming part. Our front-end and back-end groups made it easier to work and to count everyone's opinion.
With smaller groups it would be easier for the team to work together, even if it now worked well in our group. I have learnt how to work in a big group and how the group dynamic works and I have felt that this teamwork has worked good during the whole project and I am glad over what we have achieved.

During the project i got really much experiences about how to create an application from scratch, that can be used for real by others. Expect from that i'm actually really happy about out project and how the result turned out even if it was so many different tools we used that i did not even know existed before this project. We have also learnt how to actually development our own application and IT-system, which is really useful for courses or jobs in the future. I like that we were not only writing a thesis, that we actually worked with our hands and were programming all the time.

14.1.6. Saranya Silawiang

When we starting the project I was asked to be communication manager, mostly to fill out the roles and it seemed fine by me. Being a communications manager means that you are responsible for presentation and reports that should be handle in during the course. I was expected skills in Power Point and Word which I was familiar with earlier. I needed to know when the deadline in every deliverable is, plan what should be saying in the report, all the requirements on each deliverable and also that everyone do what to be done. My role as communication manager varied a lot during the project, but I think it went very well any way since I got a lot of help from Markus and Johanna in the beginning. Many of us have participated in almost all report writing which is good and is a sign that we can work together as a team.

Beside the communication manager role, I was assigned to be a part of the front-end team, but since I was mostly sick in first week and decided instead to change to back-end team. In back-end team I worked along with Teddy, Ilkka, Malin and Mathias. I saw my chance to develop my skills in programming which is a good idea and for myself when it comes to learning. Since everyone has different programming skills we spent a lot of time on installing programs and tools that we needed. I spent the most time learning to use the different tools that the back-end team decided to use. During the first two weeks, I spent most of my time learning how to use different tools that back-end team decided to use. By watching Youtube tutorials one can follow example from scratch which I learned a lot of. Since one in back-end team create a database on mLab we needed to learn how to send and receive data from there. We worked in Brackets which is a code editor and I've been coding in Brackets earlier courses and felt familiar with it. Our database is a noSQL database which consist collections and to put j-son object in collections you can use CRUD object operation. We tried to figured it out how to use GET and POST functions by using resteasy which is Google Chrome's extensions. At the end we came along with GET and POST function but then one in back-end team decided to start with feathers that replaced express and simplifies management of CRUD operations. We realized that it was more efficiently since you need to write just one

function for CRUD operations and with express you needed to do function for each operation.

When we started with feathers it became a division in back-end team, a database team and some main application function developers which worked very good. In the beginning I was in team with Teddy trying to figured it out how to use authentication on log in page. But personally I thought it was difficult to understand how feathers worked and considering leaving to Teddy and Malin, which I thought they had more experience and knowledge about that. Instead, I started to cooperate with Mathias and in couple of days we tried to work with the record function. We started to successively finding solutions by getting a lot of help from Google search function. During developing the functions, we worked a lot with JavaScript and HTML, which I developed my skills in those programing language and Mathias helped me a lot to understand. It went very well because I wasn't that familiar with JavaScript earlier and I learned a lot. In JavaScript we can use scope which provide access of variables, functions, and objects in some particular part of the code. By using scope, we could show the value of a variable from JavaScript in a HTML code, i.e. show distances and time while stopped recording. We also spent hours on the design, which meant that we suddenly became a front-end team. As the roles were flexible, you could switch between front-end and back-end. Which meant that I learned much about HTML and we got help from W3Schools.

In the end of the project I worked a lot with the Usability Testing which made me to one of the Usability Experts. Before usability testing we write the methods about use cases, tasks to perform and select our evaluators that matched with the typical personas for the future users of the system. By using System Usability Scale, we could calculate the average score for how user-friendly our application is. Me, Malin and Mathias did the first usability test with one of the evaluators here at Ångström. During the testing with the first evaluator I act secretary and wrote down evaluator's thoughts. I was the one that calculate the scores where you can see in Appendix and with help from Malin, Markus, Ilkka and Johanna we write the report about Usability Assessment. I was a part of derivable 4 about System Design

with Malin and Teddy. I worked a lot with finale report at the end of the project, where I spent most of my time on chapter 4. Requirement (*Requirement changes*), chapter 6. Design and user interface development (a part of *Logotype*, *Color scheme*, a part of *Final design and user interface*), chapter 8. Technical Solutions (*Google Drive*), chapter 10. Evaluation (*Usability Testing*, *Test group* and a part *of Testing results*) and chapter 11. Conclusion (a part of *Project collaboration* and *Design and user interface*).

Due to the group consisted of 8 students there were disadvantages but also benefits. What has been good is that report writing has gone faster since two or more have focused just on derivable. What has been less good is about programming, since it is difficult to be 8 pieces that program when communication between front-end and back-end was not as intended. But we learned in the meantime and it worked out very well. As the roles were very flexible, the roles collided with each other, which meant that on the back-end team worked a lot with usability as well. It became more like we decided most designs together as a large group at the end which is good since everyone's opinion counts. Therefore, I have felt that this collaboration has worked very well and I am pleased with what we have achieved. I also think that this course has been very instructive, educational and there is a lot that I will take with me in the future. I have learned to work in a bigger team of different divisions which feels very useful for future work. As mentioned before the roles have been rather clear and instead, the roles could have been more dynamic and be changed over time. It feels enjoyable to do something similar to real life, since you have proven your knowledge for yourself and for others, but also less well-known things.

14.1.7. Malin Sjöberg

When we started the project I was unsure what programming role I wanted. For a long time I have liked front-end programming but during the course in database design I also developed an interest in databases, how to design and use them. In the beginning I decided to be on the front-end team since this seemed like the easier choice for me as I was more familiar with front-end programming. I worked as a front-end developer for about two weeks. But the group soon realised that we needed to be more people in the back-end team. As I still was curious about the back-end programming, I decided to switch to the back-end. I worked as a back-end developer and also took the role as database designer with Teddy. When our work with the server and the database was done and up and running, I moved on and started working as a front-end developer again. When the usability testing came I started helping out as a usability expert as well. As we have had relatively flexible roles this has worked out very well for me because I was involved in different parts of the project and got to learn quite a bit of everything. And that was my hope in the beginning of the project.

The first week as a front-end developer I just tried to familiarize myself with the tools we decided to use. I also tried to repeat and learn more HTML, CSS and JavaScript. In the front-end team we also sat down and drew mockups of the system by hand.

Then I started my work as a back-end developer. The first days in the back-end group I spent trying to catch up to the other people. I was setting up everything on my computer that was needed and asked them what they had been doing and what they thought I should have a look at, which part they were trying to do. I started watching tutorials on how to set up a connection to a database. In brackets I wrote simple models and schemas for a database structure and connected it to our database. I could POST and GET data from the database through localhost. This was the first step of our server. Then we decided to team up and code in couples. I worked with Ilkka and for a couple of days we sat and created the different CRUD functions because he felt that we would need them in the future, and so we did!

Then we switched groups, I started working with Teddy instead. We decided that we needed to have a look at the database so we took the roles as database designers and we had the main responsibility for the back-end. The first design of the database didn't meet the requirements that we started to realise our application needed. I and Teddy started to draw up a new version of the database with everything we thought we needed at this point. We revised the schemas and models on the server to match our new requirements for the JSON objects in the database. After we set up the new server on DigitalOcean, we decided that we needed to populate the collection *Checkpoints,* so the checkpoints could be used in the routing algorithm, that the other developers had started working on. Teddy found a CSV-file with locations of different places in Sweden. We converted it to an Excel-file where we could easily display the data. It took a couple of days for us to remove the data that we didn't need in our database and to import it to our database, we had a lot of small complications. But after we solved them we had 32 076 objects in our database, so we were very pleased.

After we populated the database, we felt satisfied with the server and database and could move on to help out on the front-end. I started working as a front-end developer. During this time I mostly worked with Teddy, but our work in the group was dynamic and we helped each other a lot. I and Teddy started with the *Friends*-page and the functions to display, add and remove friends. Then worked for a while with figuring out how to best display the username instead of the user's ID. After we solved it by using a service, we added the function to display your friends routes if you press on their name.
When the usability testing came it was clear that we needed more people on Usability, therefore I jumped in as well. I read up on what was needed and I wrote a testing protocol and Saranya took over and revised it when I did some more front-end programming. But then I was part of doing the usability testing as well.

Then I started working on adding feedback to our application. For *Log-in* and *Sign-up* we needed some feedback for the user for example if they typed in the wrong password. I also

added the double-check of the password on *Sign-up* and in *Settings*. Lastly I worked with Mathias on the *My Routes*-page.

I was part of the group who wrote the deliverables about the *Requirements*, the *System Design and Test Plan* and the *Usability Assessment*. In the finale thesis I and Teddy wrote the chapter about the *Back-end Development*. I also helped out writing the functions of the system in chapter 6 about the *Final design and user interface.* I was part of writing the *Evaluation* chapter (*Usability Testing* and *Launching Citywalks*) and also wrote the *Friends* section of the *Social Features* in *Front-end Development*. And lastly I and Teddy wrote *Back-end* in *Lessons Learned*.

During this project I've learned a lot of things. Besides coding in front-end with HTML and JavaScript and how to setup and design a NoSQL database, I've learned a few more general lessons about building an application. The first lesson is that there is always several ways to accomplish something. Every time we were stuck on a problem we tried to do it multiple ways. When we redesigned the API for the database we tried to make the coordinates GeoJSON objects. In the end we were not able to make it work and instead we made coords to an array of arrays. It would probably have been better if we had the coordinates as GeoJSON objects but for us it has worked out fine for us so far.

One of the lessons for me on this project have been that it is difficult but it's also very profitable and fun to work in a group. It's can be difficult to work in a group because in a large group of people there is often times when people disagree. There is always situations when people have differencing opinions. But for the most part this project has given me the notion that people work best together. Coding by myself works well and can be fast and efficient. But working with new things and not really sure how to accomplish something it is better to work together. Because you always get another input that can help you move forward.

The last and the biggest lesson I've learned during this project have been that the back-end is very important. When we designed the database, I felt like it had a really good structure. But as I worked more with the front-end development, I started to realise that it wasn't perfect. There was a lot of cases where a better designed and structured database would've made the front-end programming a lot easier. Therefore I feel like we should have done a better ground. But I also remember that I was difficult to make choices and design a database before I knew exactly what was needed from it. That is why I feel like this project and the fact that I got to work with different parts and see how they worked together was especially beneficial and fun for me.

14.1.8. Teddy Wickström

Before this project I have had a desire to build, or at least learn how to build a mobile app. So at the start of the project I pushed and researched hard for that decision. After the decision was made I had a major role in the search for which technologies we would use. Alongside the technologies I also had a major role in the overall production and decision making, due to the fact that I had acquired a lot, of overall knowledge. As the technical manager I help installing the chosen software and informed group members, to my best ability, how to utilise them. I also had the main responsibility for the back-end development and as a part of that was part of the database design team. So my role changed a lot during the project, depending on what was needed.

I designed, with some help from other team members, all the different versions of the API. As a part of the backend me and Malin designed the database. After the Usability testing with Anton we shifted focus and I started working on the Front-end writing mainly functions in the controller.js file, and fixed bugs discovered during the testing phase. This shift of focus meant that we stopped development of the backend so some desired features were never fully implemented or solved in an other, perhaps less ideal manner. This was quite sad since we had worked hard to create the proper structure required for these features which would have made our application a lot better and would have granted the functionality needed to be able to handle larger amount of data.
One of which is the handleUser service I created that takes Names or Id's and matches them to corresponding id's and names from elsewhere. I later used this Service alongside malin and others to replace Id's throughout the application with userNames and routeTitels. This would, I suspect, ideally been done on the server through so called hooks on the back-end server.

Alongside Malin wrote the Back-end section of the report. I also wrote a large part of the testing protocols and the reflections concerning how we have worked and should have worked. I wrote large parts of the Technical Solutions, front-end and back-end

Large part the “ready for launch” part. Large part of the back-end lessons learned, which is why there are a lot of similarities between my lessons learned and the overall lessons learned.

I learned that even if you don’t know everything about coding you can still do stuff and it's really fun and educational way of working. All the lessons learned, or at least most of them, are these kind of hindsight scenarios were I realised that we should (could) have done something in another way, to make something work better or to give us an easier path to extended functionalities.

We spent a lot of time searching for bugs and testing in an inefficient manner. Due to poor modularity and lack of testing protocols, luckily we, who wrote most of the functions, all worked closely together, and thanks to that we had a sufficient understanding of almost every part of the code. But for an outsider certain functions and parts might not be so easy to comprehend and work with. And my desire, at least after this project, is to write code and design the structure in a way that someone else can continue my work without hours of headache, without them thinking I’m an idiot. So if I were to start this project all over again today I would spend a lot more time going over the actual structure of the app and force myself to “do the work” properly when coding, use something like a config file containing all the “permanent”, recurring properties and the same thing for recurring functions.
I have learned that if a proper structure is not achieved on the first attempt one should not be afraid to redo some parts to get there. Almost all of the work done can be reused and it will, beyond any doubt in my mind, be a good decision in the end.

When working with node.js there are a lot of available packages available through the npm (node package manager). This can at first glance seem like a good thing but sadly all of them don’t meet a high standard, some are good some are bad, some are underdevelopment and some are no longer maintained. Choosing the right ones specifically for us, without really knowing our needs was at times troublesome and a great responsibility. Especially in the beginning when I, during some periods was alone in the back-end. So if anything I have

learned the importance of the back-end and the work needed “behind the scenes” and that discussing the different possible paths ahead with someone would have been greatly appreciated. At this point I must make it absolutely clear that I was not alone in back-end development, a lot of work was done alongside and even without myself. I’m talking about the conceptual and overall design parts. Even if I wanted to be part of the decision-making-process some additional support, an extra set of eyes, to scan through this jungle of node packages available would have been appreciated.

I tried to implement protocols for the Development → Deployment → Production cycle after a few tedious deployments. I realised how much time and headache could have been saved if proper ways had been utilised from the start and if these protocols would have been followed by everyone. In relation to this I should have been better at informing and “enforcing” these rules, which I in some way knew would help us later on.

14.2 System Usability Scale Score

Test 1: Belinda Pettersson Rimgard
System Usability Scale (SUS):

1. I think that I would like to use this system frequently: 3
2. I found the system unnecessarily complex: 1
3. I thought the system was easy to use: 3
4. I think I would need the support of a technical person to be able to use this system: 0
5. I found that the various functions in this system were well integrated : 4
6. I thought this system was too inconsistent: 0
7. I would imagine that most people would learn to use this system very quickly: 4
8. I found the system very cumbersome to use: 1
9. I felt very confident using the system: 3
10. I needed to learn a lot of things before I could get going with the system: 0

On statements 1,3,5,7 and 9, calculate scale position - 1: 2+2+3+3+2 = 12
On statements 2,4,6,8 and 10 calculate 5 - scale position: 4+5+5+4+5 = 23
*Total score * 2,5* = 35*2,5 = 87,5

Test 2: Karl-Gunnar Fejes
System Usability Scale (SUS):

1. I think that I would like to use this system frequently: 2
2. I found the system unnecessarily complex: 1
3. I thought the system was easy to use: 3
4. I think I would need the support of a technical person to be able to use this system: 0
5. I found that the various functions in this system were well integrated : 3
6. I thought this system was too inconsistent: 1
7. I would imagine that most people would learn to use this system very quickly: 2
8. I found the system very cumbersome to use: 1
9. I felt very confident using the system: 4
10. I needed to learn a lot of things before I could get going with the system: 0

On statements 1,3,5,7 and 9, calculate scale position - 1: 1+2+2+1+3= 9
On statements 2,4,6,8 and 10 calculate 5 - scale position: 4+5+4+4+5 = 22
*Total score * 2,5* = 31*2,5 = 77,5

Test 3: Ann-Christin Fejes
System Usability Scale (SUS):

1. I think that I would like to use this system frequently: 4
2. I found the system unnecessarily complex: 1
3. I thought the system was easy to use: 4
4. I think I would need the support of a technical person to be able to use this system: 1
5. I found that the various functions in this system were well integrated: 4

6. I thought this system was too inconsistent: 1
7. I would imagine that most people would learn to use this system very quickly: 4
8. I found the system very cumbersome to use: 1
9. I felt very confident using the system: 3
10. I needed to learn a lot of things before I could get going with the system: 3

On statements 1,3,5,7 and 9, calculate scale position - 1: 3+3+3+3+2= 14
On statements 2,4,6,8 and 10 calculate 5 - scale position: 4+4+4+4+2 = 18
*Total score * 2,5* = 32*2,5 = 80

Test 4: Maria Lindén
System Usability Scale (SUS):

1. I think that I would like to use this system frequently: 4
2. I found the system unnecessarily complex: 1
3. I thought the system was easy to use: 4
4. I think I would need the support of a technical person to be able to use this system: 0
5. I found that the various functions in this system were well integrated: 3
6. I thought this system was too inconsistent: 0
7. I would imagine that most people would learn to use this system very quickly: 4
8. I found the system very cumbersome to use: 1
9. I felt very confident using the system: 3
10. I needed to learn a lot of things before I could get going with the system: 0

On statements 1,3,5,7 and 9, calculate scale position - 1: 3+3+2+3+2 = 13
On statements 2,4,6,8 and 10 calculate 5 - scale position: 4+5+5+4+5 = 23
*Total score * 2,5* = 36*2,5 = 90

Test 5: Ola Bengtsson
System Usability Scale (SUS):

1. I think that I would like to use this system frequently: 3
2. I found the system unnecessarily complex: 1
3. I thought the system was easy to use: 4
4. I think I would need the support of a technical person to be able to use this system: 2
5. I found that the various functions in this system were well integrated: 3
6. I thought this system was too inconsistent: 1
7. I would imagine that most people would learn to use this system very quickly: 3
8. I found the system very cumbersome to use: 1
9. I felt very confident using the system: 3
10. I needed to learn a lot of things before I could get going with the system: 3

On statements 1,3,5,7 and 9, calculate scale position - 1: 2+3+2+2+2 = 11
On statements 2,4,6,8 and 10 calculate 5 - scale position: 4+3+4+4+2 = 17
*Total score * 2,5* = 28*2,5 = 70

14.3 Initial design

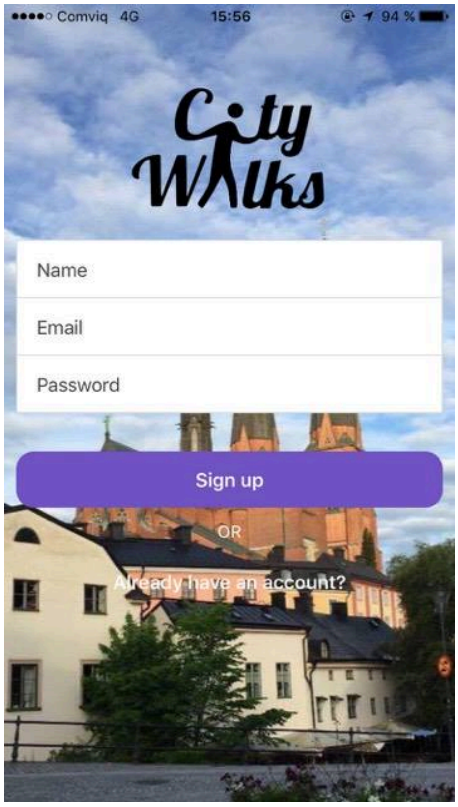


Figure 1. Sign Up page

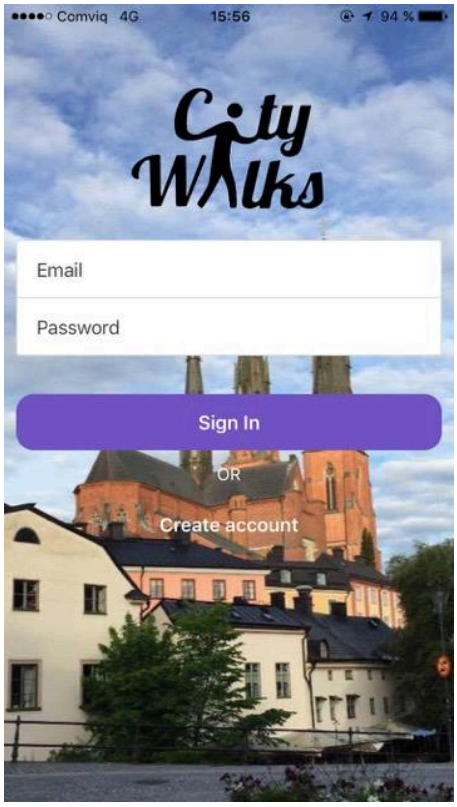


Figure 4. Log in page

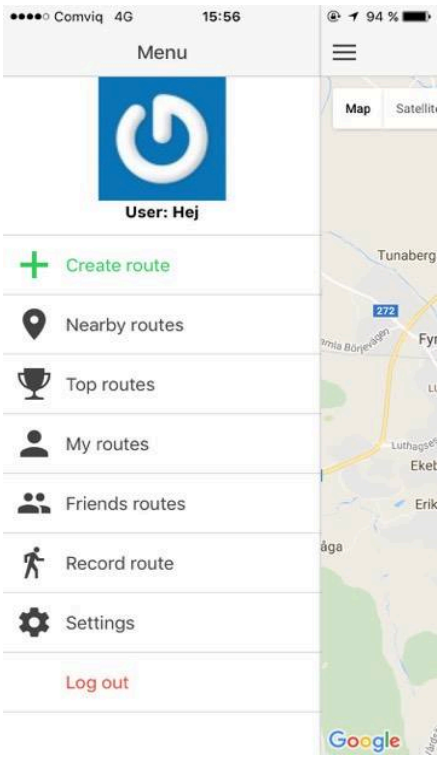


Figure 3. Side menu

Figure 4. Map integration

14.4 Mock ups

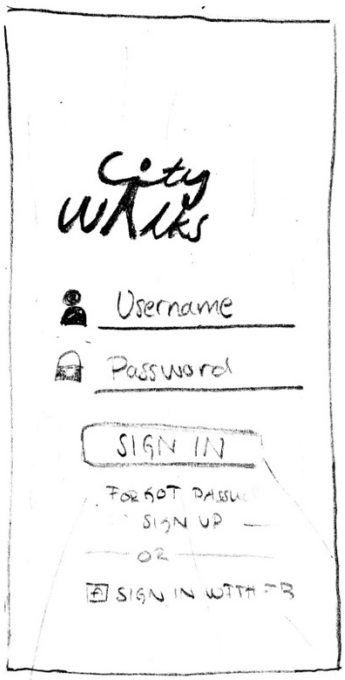


Figure 1. Log in page

Figure 2. Reset password page

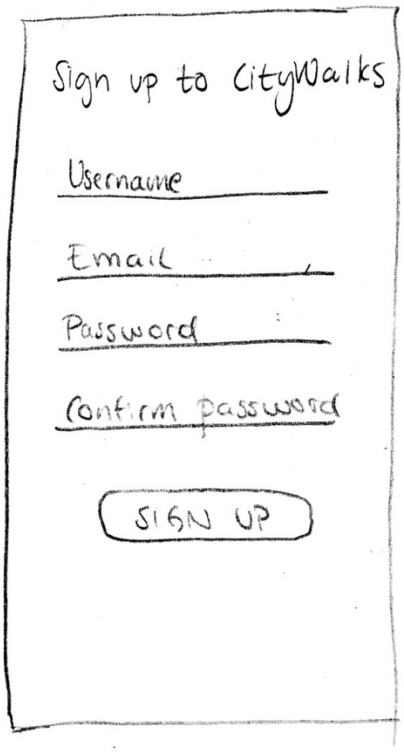


Figure 3. Sign Up page

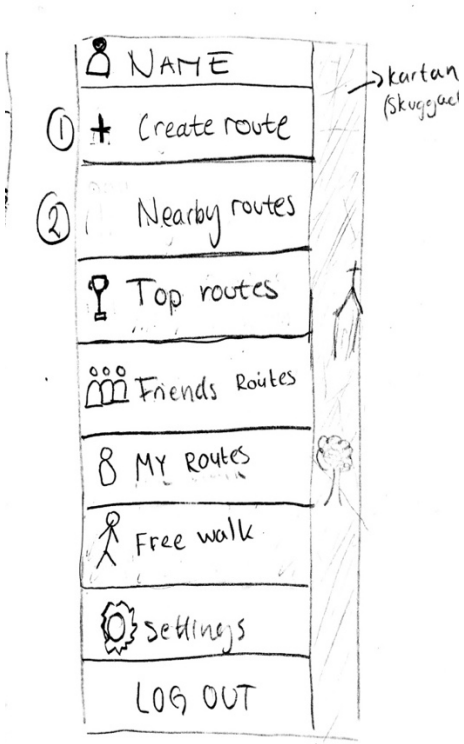


Figure 4. Side menu

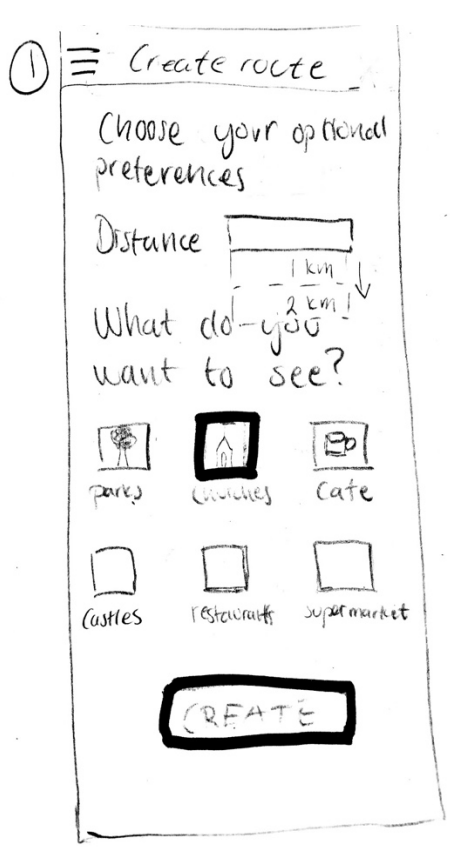


Figure 5. Create route

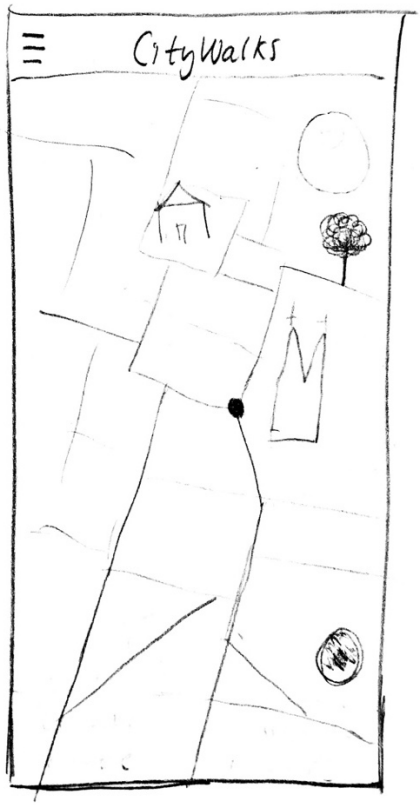


Figure 6. Map integration

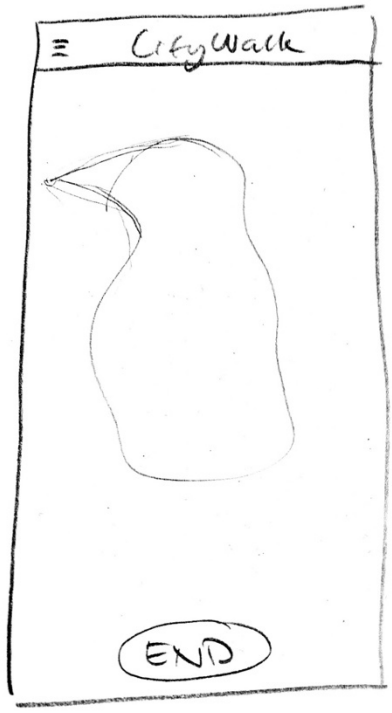


Figure 7. End walking

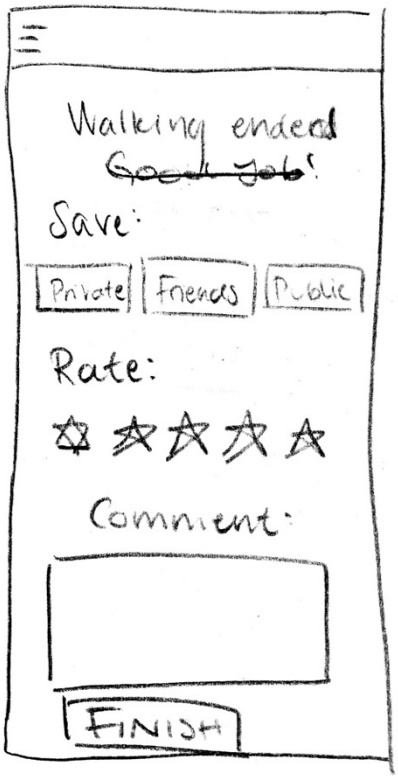


Figure 8. Comment and rate page

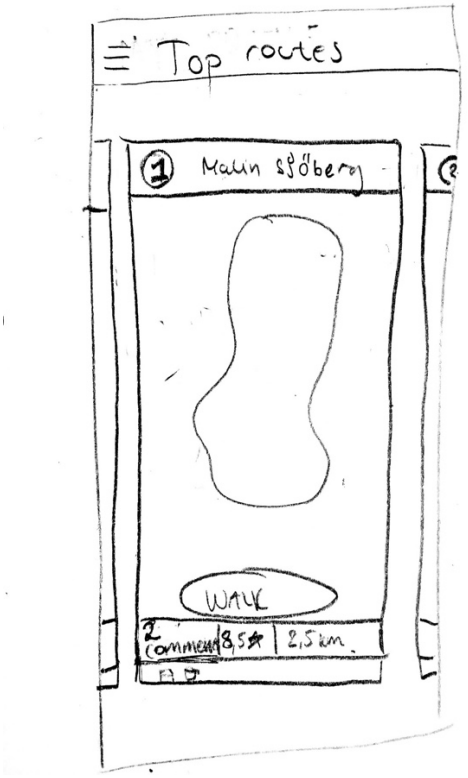


Figure 9. Top routes

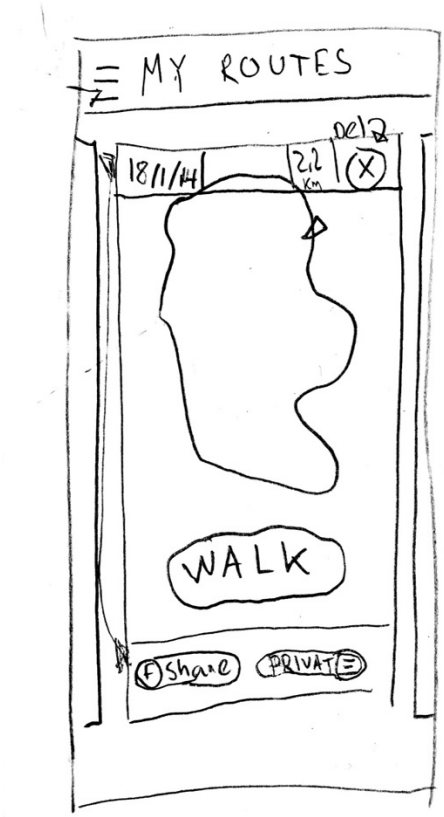


Figure 10. My routes

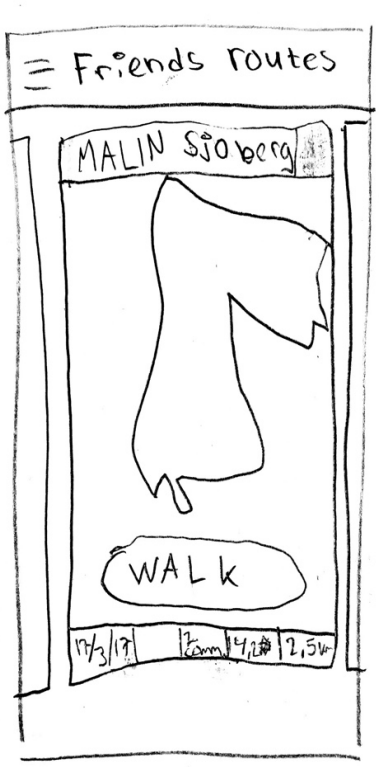


Figure 11. Friends routes

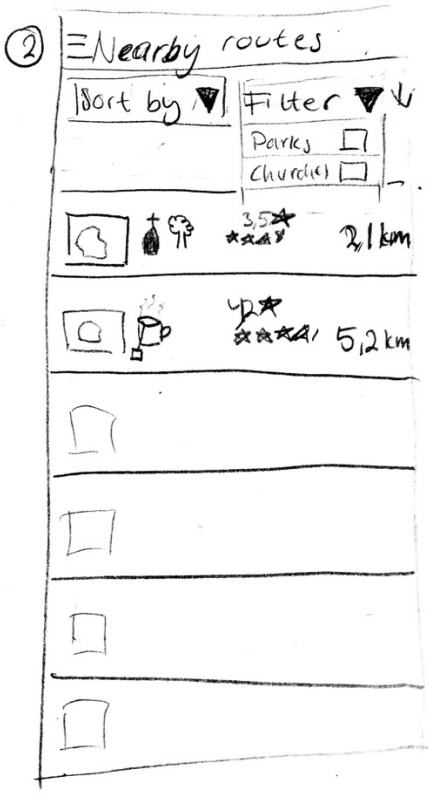
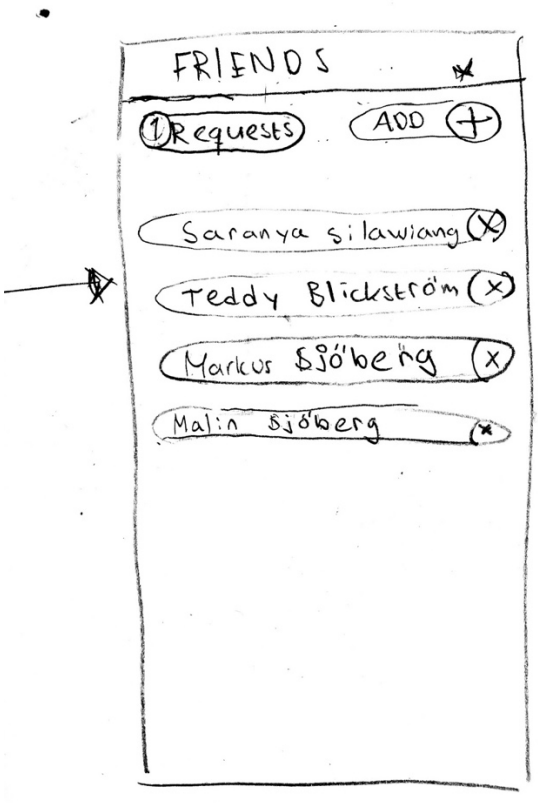


Figure 12. Nearby routes



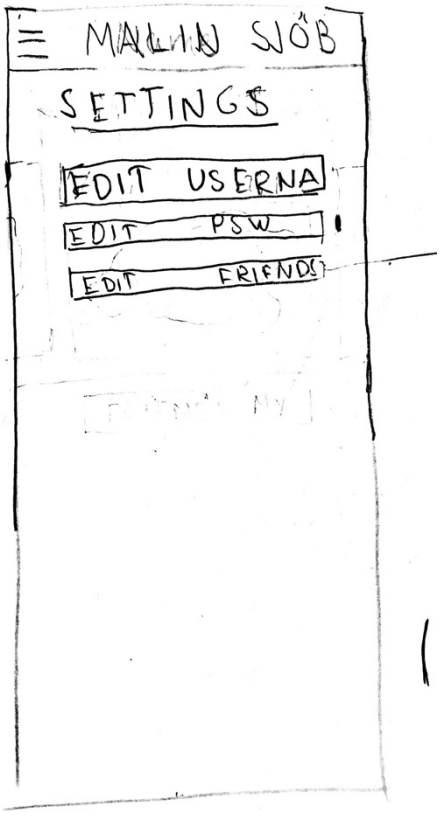


Figure 13. Friends page *Figure 14. Setting page*