



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

---

# **Implementation of a Real Time Position Tracking System for a Transportation Company**

Bachelor's thesis in Computer science and engineering

Pär Aronsson, Amanda Dehlén, Jonathan Heinin,  
Edward Karlsson, Jesper Larsson, David Lidevi



BACHELOR'S THESIS 2021

# Implementation of a Real Time Position Tracking System for a Transportation Company

PÄR ARONSSON, AMANDA DEHLÉN, JONATHAN HEININ,  
EDWARD KARLSSON, JESPER LARSSON, DAVID LIDEVI



UNIVERSITY OF  
GOTHENBURG

---



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering  
CHALMERS UNIVERSITY OF TECHNOLOGY  
UNIVERSITY OF GOTHENBURG  
Gothenburg, Sweden 2021

Implementation of a Real Time Position Tracking System for a Transportation Company

Pär Aronsson, Amanda Dehlén, Jonathan Heinin  
Edward Karlsson, Jesper Larsson, David Lidevi

© PÄR ARONSSON, 2021.  
© AMANDA DEHLÉN, 2021.  
© JONATHAN HEININ, 2021.  
© EDWARD KARLSSON, 2021.  
© JESPER LARSSON, 2021.  
© DAVID LIDEVI, 2021.

Supervisor: Vincenzo Massimiliano Gulisano, Department of Computer Science and Engineering

Examiner: Arne Linde, Department of Computer Science and Engineering

Bachelor's Thesis 2021

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

SE-412 96 Gothenburg

Telephone +46 31 772 1000

Typeset in L<sup>A</sup>T<sub>E</sub>X  
Gothenburg, Sweden 2021



# Implementation of a Real Time Position Tracking System for a Transportation Company

Jonathan Heinin, Jesper Larsson, Amanda Dehlén,  
David Lidevi, Edward Karlsson, Pär Aronsson  
Department of Computer Science and Engineering  
Chalmers University of Technology and University of Gothenburg

## Abstract

This thesis focuses on the development of an intuitive real-time position tracking software system for a transportation company. Their requirement was a product that allows personnel that coordinate work assignments to oversee the location of the active drivers within the company. This was achieved by the creation of requests from the coordinators to the drivers to share their position during a work shift. Aspects like security and the integrity of the drivers were taken into consideration as well as ease of use and adaptability to the company's needs.

With the requirements as background and by using agile methods, the development was structured to align with the needs of the company. The result was a software system consisting of two applications, a mobile application for the drivers and a web application for the coordinators. The mobile app was evaluated based on the battery consumption, mobile data consumption as well as the user experience. The final product was a success in terms of fulfilling the needs of the company at a smaller scale but with the possibility of further development, for example integrating a planning system or adding a communication tool.

Keywords: computer science, engineering, real time communication, position tracking, flutter, react, firebase

## Sammanfattning

Rapporten fokuserar på utvecklingen av ett intuitivt platsdelningssystem för ett transportföretag. Deras kravspecifikation var en produkt som tillät personal som koordinerar uppdrag att överse företagets aktiva förars plats. Det uppnåddes genom skapandade av förfrågningar ("requests") från koordinatörerna till förarna om att dela deras position under ett arbetspass. Aspekter som säkerhet och förarnas integritet togs i åtanke så väl som användarvänlighet och anpassningsförmåga efter företagets behov.

Med kravspecifikationen som bakgrund och genom att använda agila metoder strukturerades utvecklingen till att vara i linje med företagets behov. Resultatet var ett mjukvarusystem bestående av två applikationer, en mobilapplikation för förarna och en webbapplikation för koordinatörerna. Mobilappen utvärderades baserat på batterikonsumtionen, mobildatakonsumtion så väl som användarupplevelsen. Den slutliga produkten var framgångsrik i bemärkelsen av att uppnå företagets behov i en mindre skala men med möjligheten för framtida utveckling, till exempel att integrera ett planeringssystem eller addera ett kommunikationsverktyg.

# Acknowledgements

We thank our supervisor Vincenzo Massimiliano Gulisano for all the help and support he has provided to us during this bachelor project.

We are grateful for ElvaEtt's project proposal as a bachelor project. They have been very cooperative, informative and understanding throughout this project which has made this experience fun and instructive to us.

Our peers that peer reviewed and gave us feedback on our report helped us greatly and we want to thank them all.

To all the individuals who took part in our system test, we want you to know that we appreciate the time you devoted into it. The feedback you gave afterwards and the bugs you noticed helped us recognize problems we might not have noticed otherwise.

# Vocabulary

**Admin** Coordinators (see below) that also have the ability to create new coordinators for the web application.

**Agile** An iterative approach to project management and software development.

**Application programming interface (API)** A software intermediary that allows two applications to talk to each other.

**Backend** A term in software development that refers to any part of the software system that the user does not see.

**Backend-as-a-service** Is a cloud service in which all the behind-the-scenes aspects of a web or mobile application is outsourced.

**Class-based programming** A style of object-oriented programming in which inheritance occurs via defining classes of objects, instead of inheritance occurring via the objects alone.

**Client-optimized programming language** A language optimized for client oriented applications.

**Cloud service** Infrastructure, platforms, or software that are hosted by third-party providers and made available to users through the internet.

**Component-based architecture** Focuses on the decomposition of the design into individual functional or logical components.

**Coordinators** Personnel responsible for coordinating the company's assignments and drivers. They have access to the web application in the Locatr software system.

**Cross-platform software** Computer software that is implemented on multiple computing platforms.

**Database triggers** Procedural code that is automatically executed in response to certain events on a particular table or view in a database.

**Domain model** Conceptual model of the domain that incorporates both behavior and data.

**Drivers** The workers who drive the artists or packages between different locations and are only able to access the mobile application in the Locatr software system.

**ER diagram** Stands for entity-relationship diagram which is used for showcasing how database entities relate to each other.

**Framework** Software that is developed and used by developers to build applications.

**Frontend** Term in software development that refers to the the visual aspects that the user sees on the screen.

**Minimal viable product (MVP)** Version of a product with just enough features to be usable by early customers who can then provide feedback for future product development.

**Model-view-controller (MVC)** Software design pattern that is commonly used for developing user interfaces that divides the related program logic into three interconnected elements: model, view and controller.

**Native code** Term that machine code is sometimes called when referring to platform-dependent parts of language features or libraries.

---

**Object-oriented programming** Programming paradigm that relies on the concept of classes and objects and is used to structure the software into simple, reusable pieces.

**Open-source software** Type of computer software in which the source code is released under a license in which the copyright holder grants user the rights to use, study, change, and distribute the software to anyone and for any purpose.

**Package** Folder consisting of one or several files containing code.

**Plug-in** Features added as a dependency to already existing software.

**Push notifications** Message that pops up on a mobile device.

**Request** Term we use to describe a work shift for drivers. A coordinator will request its driver to start sharing his or hers location where the driver accepts the request, starting the work day.

**Scrum** Framework utilizing an agile mindset for developing, delivering, and sustaining complex products.

**Separation of concerns** Design principle for separating a computer program into distinct components such that each component addresses a separate concern.

**Single-page application** Web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages.

**Snackbar** Pop up notification that comes from the bottom of the screen when inside the application.

**Software development kit (SDK)** Collection of software development tools in one installable package.

**SOLID design principles** One of the most popular sets of design principles in object-oriented software development.

**Sprint** Is a short, predetermined time period when the team works to complete a set amount of tasks.

**Stakeholder** The person the team is supposed to deliver value to. In this thesis the stakeholder is ElvaEtt.

**User experience (UX)** How a user interacts with and experiences a product, system or service.

**User interface (UI)** The space of human-computer interaction and communication in a device.

**User story** Is a short story explaining a desired software feature which someone wants to be developed.

# Contents

<b>List of Figures</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Purpose of the Project . . . . .	1
1.2 Scope . . . . .	2
1.3 Structure of the Report . . . . .	2
<b>2 System Requirements</b>	<b>4</b>
2.1 Requirements Specification . . . . .	4
2.1.1 Laws and Security . . . . .	5
2.1.2 Database Security Requirements . . . . .	6
2.2 Existing Solutions that Almost Fulfill the Requirements . . . . .	6
<b>3 Technical Concepts and Frameworks</b>	<b>7</b>
3.1 The Cloud-Based Service Firebase . . . . .	7
3.2 Frameworks and Libraries Used for the Frontend of the System . . . . .	7
3.2.1 The Library ReactJS and the Framework Next.js . . . . .	8
3.2.2 Mapbox as a Map Provider Library . . . . .	8
3.2.3 Cross Platform Mobile Environment Flutter . . . . .	9
<b>4 Architectural and Visual Design Decisions</b>	<b>10</b>
4.1 Architecture of the Technical Solution . . . . .	10
4.1.1 Domain Model . . . . .	10
4.1.2 Database Structure . . . . .	12
4.2 User Interface Design . . . . .	13
<b>5 Work Process and System Development</b>	<b>15</b>
5.1 Scrum and Agile Workflow . . . . .	15
5.2 Project Plan . . . . .	16
5.3 Changes Made Due to Stakeholder Meetings . . . . .	16
5.3.1 Changes to the Visual Representation of a Request in the Web Application . . . . .	16
5.3.2 A Request Acting as a Work Shift . . . . .	17
5.3.3 Optimizing the Request Creation Process . . . . .	18
5.4 Implementations of Features for the Mobile Application using Flutter Plugins . . . . .	19
5.4.1 Implementation of the Position Tracking Feature . . . . .	19

5.4.2	Authentication of the Mobile Application . . . . .	19
5.4.3	Sending out Notifications to the Drivers . . . . .	19
5.4.4	Uploading Data to Firestore . . . . .	19
5.5	Adding Mapbox to the Web Application . . . . .	20
5.5.1	Searching for Places by Geocoding . . . . .	20
5.5.2	Interpreting Database Documents as Markers on a Map . . . . .	20
5.6	Security of Firebase and the Role Hierarchy . . . . .	21
5.6.1	Handling Roles Using Custom Claims . . . . .	22
5.6.2	Additional Security Rules in the Database . . . . .	22
5.7	Testing the Applications . . . . .	23
5.7.1	Methods of Testing . . . . .	23
5.7.2	Testing the Web Application . . . . .	23
5.7.3	Testing the Mobile Application . . . . .	25
<b>6</b>	<b>Final System and Result Assessment</b>	<b>27</b>
6.1	Finished Product and Fulfilled Requirements . . . . .	27
6.1.1	Login Pages, Authentication and Authorization . . . . .	27
6.1.2	Web Application User Interface . . . . .	28
6.1.3	Mobile Application User Interface . . . . .	31
6.1.4	Position Sharing . . . . .	32
6.1.5	Adaption to Laws and Security . . . . .	34
6.1.6	The Flow of Interaction . . . . .	35
6.2	Evaluation of User Testing . . . . .	36
6.3	Ethical and Societal Aspects . . . . .	39
6.3.1	Driving and Using a Mobile Application . . . . .	39
<b>7</b>	<b>Discussion of the Results</b>	<b>40</b>
7.1	Reflection on Project Workflow . . . . .	40
7.2	Future Improvements to the System . . . . .	40
<b>8</b>	<b>Conclusion</b>	<b>43</b>
	<b>Bibliography</b>	<b>43</b>
<b>A</b>	<b>Appendix 1: Design suggestions</b>	<b>I</b>

# List of Figures

2.1	A design suggestion for the web application from the stakeholder . . .	5
2.2	A design suggestion for the mobile application from the stakeholder .	5
4.1	General overview of the domain model . . . . .	11
4.2	Domain model in more detail . . . . .	11
4.3	A visualization of the database as an ER diagram. . . . .	13
4.4	The mobile application mockup. . . . .	14
4.5	The web application mockup. . . . .	14
5.1	Illustration of how the visual representation of a request changed in the web application. The left visual is before while the right is after. .	17
5.2	A screenshot of the first iteration of displaying requests in the mobile application. . . . .	17
5.3	The dialog for creating a request in the web application . . . . .	18
5.4	The data flow when sending a user position to the database shown using arrows. . . . .	20
5.5	Illustration of functionality of roles. A role of higher hierarchy contains all functionality of the lower ones with additional features. . . .	21
6.1	The web application login page. . . . .	28
6.2	The mobile application login page and change password page. . . . .	28
6.3	The map page. . . . .	29
6.4	The five different cards. . . . .	29
6.5	The process of adding a location . . . . .	30
6.6	The Requests Overview where coordinators can see all the drivers and their requests . . . . .	30
6.7	The dashboard of the mobile application, displaying both tabs. . . . .	31
6.8	The profile menu in the mobile application. . . . .	31
6.9	A screenshot of a location being shared. . . . .	32
6.10	A screenshot of a position being shared on the web application . . . .	33
6.11	An image displaying the closest drivers in a list. . . . .	33
6.12	A sequence diagram of how a position is displayed on the web application started from the driver pressing the “Share position” button .	35
6.13	The lifecycle for a request starting from the creation from the web application . . . . .	36
6.14	An image of the web application during testing. The image shows the area of <i>Gothenburg</i> . . . . .	37



6.15	An image of the web application during testing. The image shows the area of <i>Västra Götaland</i> . . . . .	37
6.16	The result from how intuitive the participants found the mobile application. The x-axis represents the grade and the y-axis represents the frequency. . . . .	38
A.1	A design suggestion for the login page for the web application from the stakeholder . . . . .	I
A.2	A design suggestion for the main view of the web application from the stakeholder . . . . .	I
A.3	A design suggestion for the menu in the web application from the stakeholder . . . . .	II
A.4	A design suggestion for how drivers are to be displayed on the map in the web application from the stakeholder . . . . .	II
A.5	A design suggestion for how to send a request to a user in the web application from the stakeholder . . . . .	III
A.6	A design suggestion for a confirmation for a successfully sent request in the web application from the stakeholder . . . . .	III
A.7	A design suggestion for adding a user in the web application from the stakeholder . . . . .	IV
A.8	A design suggestion for a confirmation for a successfully added user the web application from the stakeholder . . . . .	IV
A.9	A design suggestion for the login page for the mobile application from the stakeholder . . . . .	V
A.10	A design suggestion for the request view with no active requests in the mobile application from the stakeholder . . . . .	V
A.11	A design suggestion for the request item in the mobile application from the stakeholder . . . . .	VI
A.12	A design suggestion for the request item while sharing location in the mobile application from the stakeholder . . . . .	VI

# 1

## Introduction

Today, the use of real-time position tracking in software systems has evolved and vastly increased around the world. Different companies that have begun using this technology are within businesses such as food delivery, mail delivery and taxi services. The use of this technology improves customer services and allows for a more pleasant customer experience. It increases the reliability for customers that want to track their package by allowing them to receive a better overview [1].

This bachelor's thesis is based on a suggestion from a local company. The company *ElvaEtt*, which will be referred to as the stakeholder of the project, is mainly active during the summer months and they are responsible for transporting artists to several big events and festivals such as *Way out West* and *Lollapalooza*.

Currently, they have drivers responsible for picking up artists and transport them to other locations with help from coordinators working from an office. The coordinators only know of which assignments each driver has and the estimated time duration. Due to external circumstances, such as delay because of artists and traffic, the situation and the estimated time may differ in practice. This results in the coordinators having a poor overview of the drivers' current position and the status of the assignments. If their clients need information on a specific assignment, the coordinators have to call and ask the drivers about their whereabouts. All of this shows an inefficiency in communication between the drivers and the coordinators. Another issue is that calling the drivers could result in dangerous situations since they could possibly be distracted while driving. To solve these issues, The stakeholder wanted to make the process more efficient and safe by displaying the real-time positions of their drivers on a map accessible by the coordinators.

### 1.1 Purpose of the Project

The main goal of the project is to develop a software system that tracks drivers' real-time position. The challenge is to create a system that allows the coordinators to have a geographical overview and send requests to their drivers about sharing their positions during a work shift. The system needs to be adapted to the different users and their needs. Since the drivers will use the product in a vehicle, the most suitable tracking device is a smartphone for its portability. For the coordinators, a large screen is necessary to have a good overview of the map while still having

space for additional components and navigate easily. With this in consideration, the project will result in two applications that adapt to these needs. The first is a web application for the coordinators that will show the positions of the drivers in real-time. It includes features such as sending requests to the drivers, displaying an overview of the active and pending requests as well as other possible administration tasks needed. The second is a mobile application for the drivers to share their position.

Alongside the development process, useful knowledge will be obtained about web and app development which, while not being the main goal, can be regarded as a motive for this project. The subject and the content of the thesis might be of interest to businesses within a different type of logistics. This could for example be companies that could make use of having an overview of the positions of their users or objects. In the field of the shipping industry, this could be useful to track the positions of packages.

When looking at other systems that today use a similar GPS tracking system, as the focus of this bachelor thesis, one can see several benefits of such systems. One aspect is the communication between the navigator and taxi driver, where the navigator would have to call and find an available taxi driver for a client. With the proposed system the coordinator can instead check the map with all of the available drivers in a client's area. This also allows faster service for a client in need of a taxi since the time spent finding a suitable driver is decreased, resulting in a more time efficient trip [2].

## 1.2 Scope

The project is dependent on delivering value to the stakeholder. A delimitation of the project itself would therefore be to focus on a *Minimal Viable Product* (MVP). The MVP of the project is to create a product where the drivers of the company can be displayed on a map at request from the coordinators. The product will be limited by the features and design given by the stakeholder. Any additional features needs approval from the stakeholder before implementation. It was also decided by the stakeholders that the hardware used will be a smartphone that uses either Android or iOS as an operating system, and a computer with a browser installed.

## 1.3 Structure of the Report

The structure of this thesis will follow a sequential flow where it will start by explaining the requirements set by the stakeholders along with some of the challenges for the development. Afterwards comes a theory chapter aiming to give the reader the necessary information to understand concepts used in the project. After explaining the needed concepts, challenges and requirements, the workflow and results will

be presented which will include images of the developed system and explain how it works. These chapters will include how the initial system was created followed by changes made during the work process as different challenges appeared such as errors during testing. In the last chapters of this thesis, a deployment ready product will be presented and explained how it should be used. The testing of the system will be explained afterwards. Additionally, a conclusion of the project is provided, mentioning what improvements could be made and what the team could have done better.

# 2

## System Requirements

This chapter will in depth state the requirements for the system as well as discuss briefly why some related products will not work for the stakeholder's needs.

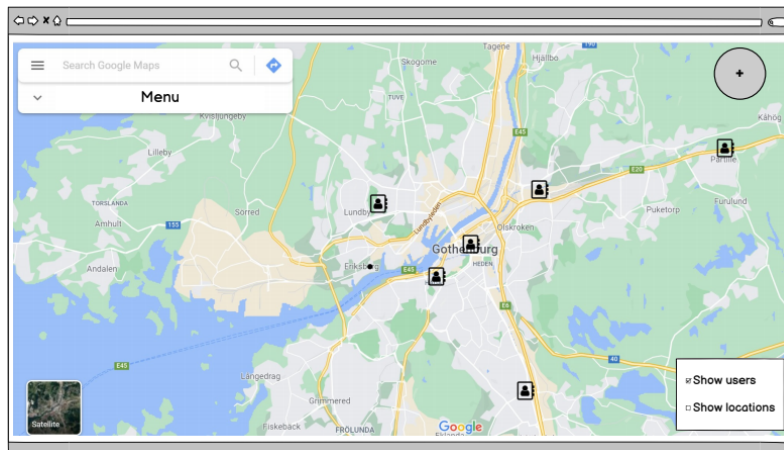
### 2.1 Requirements Specification

The project description from the stakeholder stated the fundamental requirements of the final product. One of the requirements was to have two different applications, one for the coordinators and one for the drivers. The coordinators' application should have a login page and a main page displaying a map with active drivers as markers that update as they move. This application should also be able to create driver and coordinator accounts, send requests to drivers and display the drivers closest to a location. The drivers' system should have a login page and on the main page display all upcoming requests with scheduled time to share position. Within the time span, the driver of the request will be able to share their position. When the request time comes to an end the application should automatically stop sharing the position.

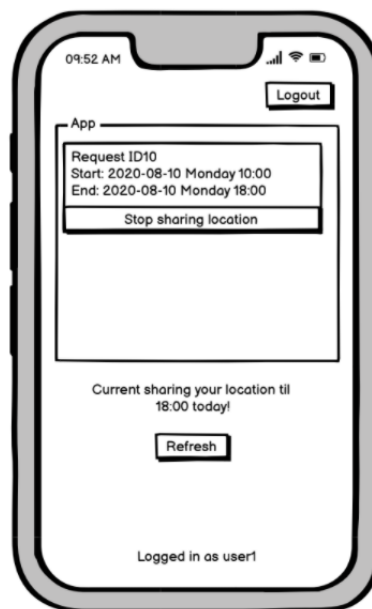
The drivers are of varied ages and levels of digital background. A too complex user interface might inhibit rather than favor an efficiency upgrade for the system. The mobile application must therefore be simple, easy to use, and not expose redundant information to the user. Additionally, the position of the user must be accessible at all times during work shifts. This means opening other applications or locking the mobile phone must not interrupt the position sharing.

Since the employees of the stakeholder has both iOS and Android devices the mobile application must be deployed for both platforms. The security of the system should prevent driver accounts from signing in to the web application, and only coordinators should have access to view the driver's positions. Data management should be strict and only stream the driver's position data when the person is sharing the position, or save it anonymously.

Included with the requirements were design suggestions for both the web application and the mobile application as seen in Figure 2.1 and Figure 2.2. All design suggestions from the stakeholder can be found in Appendix A.



**Figure 2.1:** A design suggestion for the web application from the stakeholder



**Figure 2.2:** A design suggestion for the mobile application from the stakeholder

### 2.1.1 Laws and Security

Accessing the geographical position of a person comes with a required management of personal data. There are several guidelines and requirements when handling personal data and neglecting these can lead to judicial issues.

A law related to the subject is the *General Data Protection Regulation* (GDPR) [3]. GDPR is a data privacy and security law which concerns collected personal data related to people in the *European Union* (EU). Personal data refers to any information that relates to an individual who can be directly or indirectly identified. The handling of personal data is regulated by GDPR and it includes hundreds of pages of requirements [3]. The most essential is that the data collected from a person must be absolutely necessary for the purpose and collected with consent according to

GDPR. One can only store personally identifying data for as long as it is necessary and no longer [3].

Organizations must not only *think* that they are GDPR compliant but also be able to actually prove it. A simple and effective way to do this is by maintaining detailed documentation of where, how, and when collected data is stored. Failing to obey GDPR might lead to heavy fines for the company [3] and it is in the interest of both the company and the system developers to make the system GDPR compliant so the product is legally usable.

### 2.1.2 Database Security Requirements

To keep the personal data safe and to maintain a robust system, the system follows a demand on a secure database. There are many ways to make a database safer but with the limited time, only the most common and crucial measures will be taken into consideration. The database is required to be unreachable for any unauthorized person and be protected from injection attacks. Injection attacks in this case refers to when an unauthorized user modifies content in the database by inputting malicious commands through the application interface.

## 2.2 Existing Solutions that Almost Fulfill the Requirements

There are plenty of products on the market for real-time tracking on a map. The stakeholders searched the market for a software system that would fit their needs before deciding to reach out to Chalmers. Listed below are the different software systems they looked into and what they lacked.

- **TaxiCaller** - An overall suiting software, the main issue is the impossibility to schedule a request to a driver ahead of time.
- **TaxiAdmin** - Same situation as with TaxiCaller.
- **Google Maps** - Position sharing works fine, but it is only possible to see one person's position at a time which makes this system unusable.

Since none of these existing software products can be customized to fit their needs they wished for a tailor made software system that perfectly suits them.

# 3

## Technical Concepts and Frameworks

This chapter will describe the different frameworks that have been chosen for the development of the system along with reasoning for why they were chosen. Concepts that are necessary for this report will also be described in this chapter.

### 3.1 The Cloud-Based Service Firebase

Instead of locally hosting the required computing infrastructure and applications, different companies provide hosting services of their resources like servers and software through the internet. The name cloud computing comes from the fact that the service is being processed in a remote location [4].

The cloud-based service Firebase provides the backend-as-a-service and is accessed through their services where the underlying processing is handled by them. It is a development platform developed by Google that provides a cloud service with multiple functionalities that can be implemented in software projects. The functions that are used in this project are Cloud Firestore, Cloud Functions, Cloud Messaging, and Firebase Authentication [5].

Cloud Firestore is a cloud hosted realtime NoSQL database that lets the developers use it to store data in the form of collections of documents that contain data. Cloud Functions is used to write and deploy code that will run on the Google servers. Cloud Messaging provides the ability to receive push notifications on a mobile device. Firebase Authentication is used for handling the authentication of users. In this case, the feature helps to separate the coordinators from the drivers by giving their accounts an admin status [6]. This was mainly why Firebase was chosen, and for the fact that it is simple to setup and deploy [7].

### 3.2 Frameworks and Libraries Used for the Frontend of the System

There are several frameworks for frontend development that cover all the different platforms; web, mobile, and desktop. Since the goal of the project is to develop a



web- and mobile application, the selection of frontend frameworks had to fit this requirement.

#### 3.2.1 The Library ReactJS and the Framework Next.js

ReactJS is a component-based Javascript-library for single page applications. It is broadly used within web-development due to its reusability and ease of use. The reusability is accomplished by the core concept of “Components” in *ReactJS*. Every component consists of the code concerning its behavior and supports the design principle *Seperation of Concerns*. The component can then be instantiated and used everywhere in the application [8].

ReactJS was chosen mainly because it is easy to learn, its component based structure and for its fast development. The library ReactJS also has strong community support which helps if one gets into a coding problem [9]. Other alternatives would have been Angular or Vue but those were ruled out because of the reasons mentioned before and that one of the team members had previous experience with ReactJS and could therefore teach the basics to the others. Furthermore, ReactJS is one of the best performing out of the three, and its learning curve is the lowest [10].

*Next.js* is a framework for *ReactJS* that helps create a production-ready application [11]. The three main benefits of Next.js are Server Side Rendering (SSR), Automatic Code Splitting, and Hot Module Replacement (HMR). Server side rendering refers to the ability to let the server render the HTML page and send it to the client. This allows the possibility to let the server render the main page, and then for the client to only send requests for the components in the website that are changed. Hot module replacement allows for the management of modules without having to fully reload the web page. Automatic code splitting enables only loading the needed Javascript and CSS for the requested page [11]. This will allow the application to boot up faster and run smoother in the client window.

#### 3.2.2 Mapbox as a Map Provider Library

Mapbox is a competitor to the well known map library Google Maps. Mapbox provide maps from OpenStreetMap and has a high level of customizability regarding design and map layers [12]. A map layer can for example be a collection of points that can be applied to the map. The high level of customizability was the main reason that Mapbox was decided to be used.

The map provider has a large variety of application programming interfaces (API) to provide an application with different data manipulations and actions. For example the GeoCoding package that efficiently provides searching ability for places and addresses all over the world. Mapbox also provides a cross-platform software development kit (SDK) which enables an easy way of using the map-solution across multiple platforms for further development of the system [12].

### 3.2.3 Cross Platform Mobile Environment Flutter

Flutter is an open-source mobile SDK that can be used to build native-looking Android and iOS applications from the same code base. The use of widgets are the central idea behind Flutter. By combining different widgets, the developer can build an entire UI. Each widget has a structural element, a stylistic element, a layout aspect, and many other features [13].

Flutter uses the programming language Dart, that compiles ahead of time into native code for several platforms. This leads to easier communication with the platform [13]. Dart is a client-optimized programming language for apps on multiple platforms. It is an object-oriented, class-based and garbage-collecting language with C-style syntax [14]. It can compile to either JavaScript or native code, which machine code is sometimes called when referring to platform-dependent parts of language features or libraries [15].

The main reasons Flutter was chosen was that some group members had previous knowledge of the framework and it supports both Android and iOS which was one of the requirements. With members having previous knowledge it was easier and faster to start developing. Flutter can also manipulate the UI in more effective ways compared to other similar frameworks such as React Native and is also more forgiving when refactoring [16].

# 4

## Architectural and Visual Design Decisions

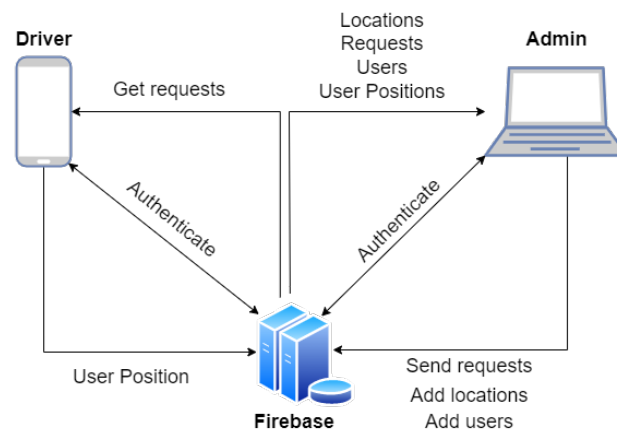
This chapter will talk about the architectural and visual design decisions taken when developing the system. It will present the domain model, an ER diagram of the database and the prototype of the user interface design of both the mobile and website applications.

### 4.1 Architecture of the Technical Solution

The system is dependent on a secure data transfer between two different applications. The communication is handled with a cloud-based database and cloud functions as backend. All communication between the driver's device and the coordinator's pass through the database first. What the devices do to stay updated is to subscribe to the database's collections, for example when a coordinator creates a work shift for a driver, the driver's application will notice that creation and will then read that data.

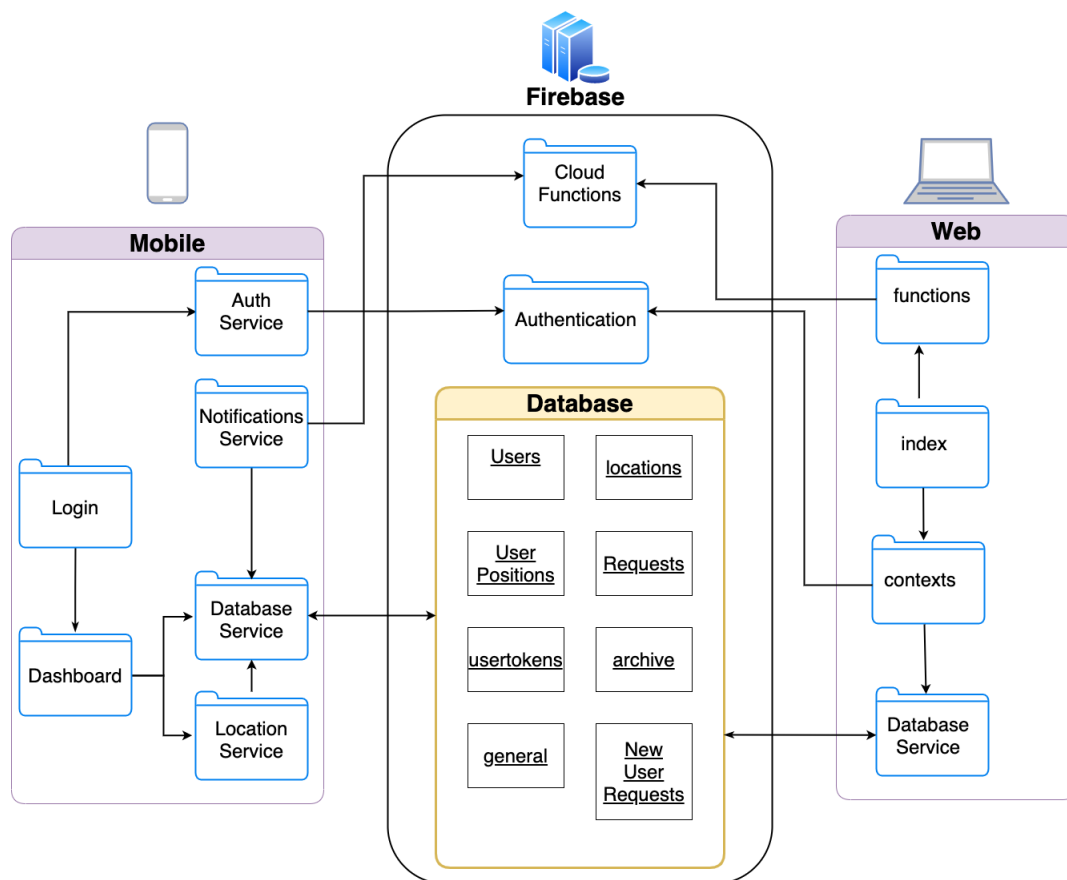
#### 4.1.1 Domain Model

Figure 4.1 shows a general overview of the project's domain model. The model showcases the relationship existing within the system and provides an accurate representation of how data is being sent through the system. As shown, the applications only interact via Firebase and has no direct contact between each other. This leaves the domain open for extension.



**Figure 4.1:** General overview of the domain model

Figure 4.2 shows the system architecture in more depth and how it follows the SOLID design principles, as there is a clear separation of concern of the driver application modules. The “dashboard” and “login” modules handle the visuals (UI) and user interaction (UX), and the service modules handle logic and communication with the database. The same principles apply for the web application, as it also has modules connected to it and later sends information to the user interface.



**Figure 4.2:** Domain model in more detail

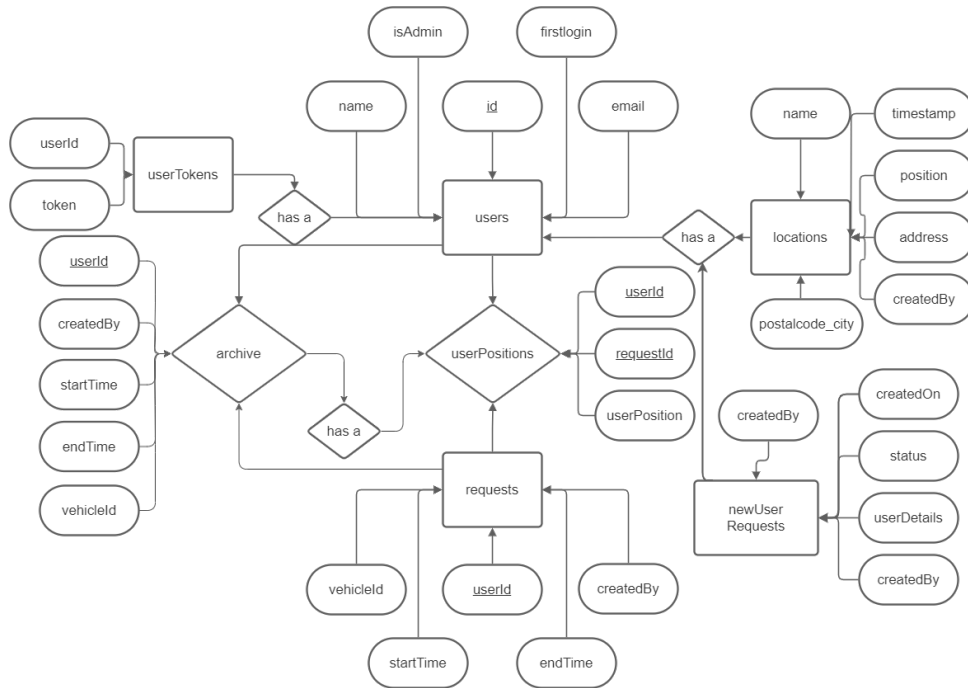
Focusing on the internal structure of the domain model, the internal structure of the mobile application follows a modified Model-View-Controller (MVC) pattern [17], where the view and controller are combined as its own entity. The reason for structuring it this way is that the program is not that large, therefore separating the view and controller functionality would only make the development slower paced, since extra measures to structure the code would be needed. The view-controller consists of the “dashboard” and “login” modules. The remaining modules are regarded as the model, as they do all the calculations, data handling and communication for the mobile application.

The internal structure of the web application is based on components where they handle their own data, following separation of concern. The main packages for the file structure are “context”, “pages”, “components”, “lib” and “functions”. In the “context” package is the file “ApplicationData.js” that is used to wrap the application with the data that constantly is changing to provide one single entry point for communication with the database. The “pages” package contains folders responding to different URLs and javascript files specifying what components that should be rendered for the page. The components are contained in the “components” package and specifies the behaviour of a specific component. Data and actions to the database that does not need real-time updates is contained in the “lib” package. The “functions” package contains all the cloud functions that is executed on the server and whose internal structure cannot be exposed to the client.

An alternative to this structure would be to implement a backend application that handles all communication with an arbitrary database that the mobile- and web application would communicate with. This might help provide a more secure data handling environment. Despite this, it was decided to use Firebase because of all its extra features it could provide, for example the authorization features. However, the most important reason was that the frontend’s user interface and user experience was the focus of the project. Since Firebase was easier and faster to setup and could be used as a backend-as-a-service it was decided to be used.

### 4.1.2 Database Structure

The database schema became relatively small with only seven entities being needed to fulfill the project goal. The idea behind it was to reuse data when possible and only create new entities if it supports the desired functionality, which resulted in the seven entities seen in Figure 4.3. Something to notice from the ER diagram is that all of the collections have some arrow pointed to the two entities “users” and “requests” which shows that these are the main concern of this project. Those two entities need to exist for the applications to work as they are also used to map a “userPosition” to a “user” and the user’s “request” enabling a coordinator to monitor which driver is working on what “request”.



**Figure 4.3:** A visualization of the database as an ER diagram.

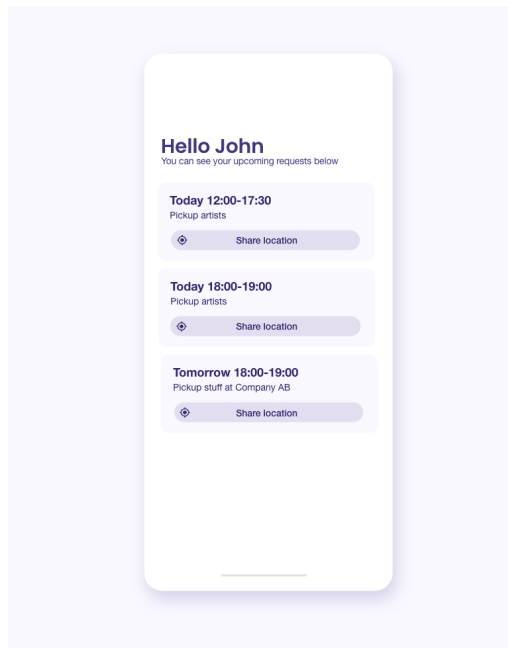
## 4.2 User Interface Design

Based on the prototype provided by the stakeholder, a prototype of higher fidelity was created to achieve a greater understanding of the system and to work towards a common goal. Having a prototype of higher fidelity also help support discussions with the stakeholder and potential users. These prototypes are shown in Figure 4.4 and Figure 4.5.

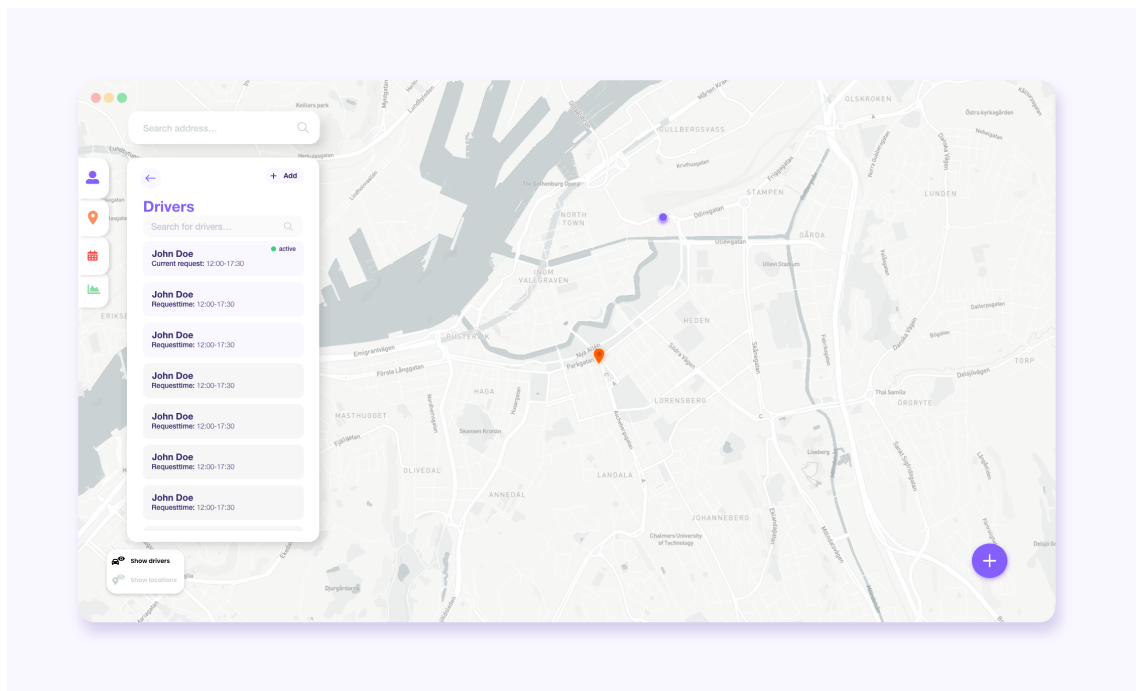
The design for the mobile prototype is motivated by the requirements of being easy and simple to use but still deliver the functionality stated in the problem description in Chapter 2. The coordinating side is dependant on the drivers using the mobile application correctly and all unnecessary details are left out to avoid confusion. Therefore the application will feature a simple and basic interface with only the most important functionality.

## 4. Architectural and Visual Design Decisions

---



**Figure 4.4:** The mobile application mockup.



**Figure 4.5:** The web application mockup.

The users of the web application are considered more digitally experienced and can handle a more complex user interface. The first draft of the user interface is kept simple and contains what is most important for the functionality seen in Figure 4.5.

# 5

## Work Process and System Development

This chapter will explain the entire realization of the project, from what was carried out first to what was done last. It will begin with explaining the working method, project plan, changes made due to stakeholder meetings and then continue with the system architecture, followed by how data is stored. At the end it covers how authentication of different roles is made, security, and the testing of the applications.

### 5.1 Scrum and Agile Workflow

This project used an agile workflow to easily organize the work and provide structure. Each member was assigned a role within Scrum with their own respective responsibility. The roles assigned are listed below:

- **Scrum Master:** Leads the planning of sprints and the division of work.
- **Two Product Owners:** Maintained contact with the stakeholders to ensure the scope was correctly aligned
- **Custom role:** One that had responsibility for the wellness and work progress of the group, including keeping track of stress levels and comparing the planned amount of functionality to the finished developed functionality.
- **Secretary:** Took notes for all weekly meetings. This role was rotated every week.

The sprints were done in intervals of two weeks starting from Monday and ending on Fridays the following week. The sprint started with a planning session where user stories were selected to be completed for the sprints. From the selected user stories, tasks were derived and measured using a measurement that gives every task a number to indicate the time spent and the difficulty of the task. Tasks also consisted of bachelor assignments which includes writing on the report. The total points for a whole sprint varied depending on the workload in other courses or upcoming assignments in the bachelor course. The goal was to have around 120-150 points in total which would end up with around 20 to 30 points per member in the group. After each sprint two members of the group, one from the web application and one from the mobile application, were chosen to proceed with a code review. The code review was a way to control that the chosen user stories were implemented and worked as intended. At the end, a sprint review was held that functioned as a way to discuss



possible bugs from the code review as well as discuss how the sprint was in general. It also ensured that the workload was not too large for the sprint or for specific members.

Every week, three recurring meetings occurred which purpose were to update other members of what tasks had been completed and the status of other tasks. A recurring meeting with the stakeholder was later introduced, which was held every other week with the intention to ensure that the project was aligned with the scope the stakeholder had in mind.

### 5.2 Project Plan

Since it had been decided to use scrum, the most logical way of putting up a plan was to create a product backlog with user stories, and in each sprint implement some of them. In total there would be eight sprints over the course of 16 weeks.

For the project plan to work, the sprints needed to have some kind of prioritization. A rough estimate was done for what was going to be done each sprint:

- **Sprint 1:** Research different frameworks and cloud based databases to use. Set up the working environment for both the web application and the mobile application. Integrate the working environments with GitHub and start working on a prototype, domain model, and database schema.
- **Sprint 2:** Finalize the prototype, domain model, and database schema. Get feedback from the stakeholders to check if they are satisfied with the prototype. Start working on the web app and mobile app.
- **Sprint 3 - 6:** Complete the web app and mobile app, start testing.
- **Sprint 7 - 8:** Make finishing touches on the web app and mobile app; performance upgrades, bug fixes, visual enhancements.

### 5.3 Changes Made Due to Stakeholder Meetings

The fortnightly stakeholder meetings have been of great value for everyone involved. It has been discovered that it is as difficult to understand the needs of a company as it is for the company to explain them. Beyond this experience came some suggestions from the stakeholders which later, after some discussion, were implemented.

#### 5.3.1 Changes to the Visual Representation of a Request in the Web Application

There has been a substantial amount of changes to how a request should be displayed, especially in the web application interface. The first assumption was that a request represents a task which led to the requests having names such as “Deliver stage materials” and “Pick up DJ”. It was later understood that a request should rather represent a whole work shift, hence the title of the request was changed to

only be the name of the driver.

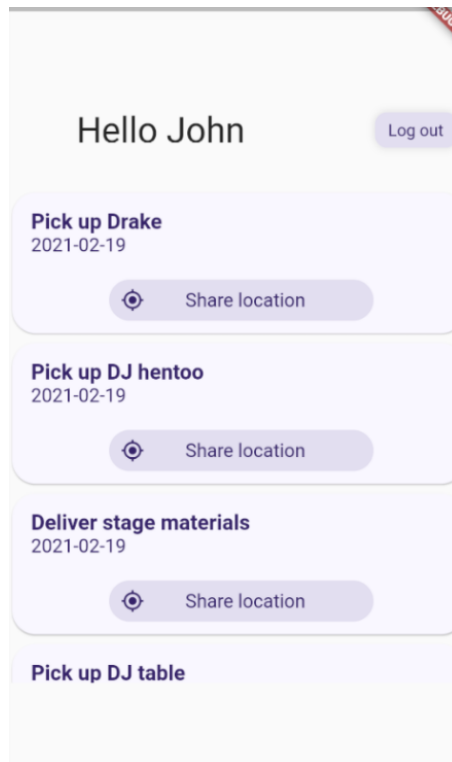
The stakeholder has experienced a lot of trouble in the past when trying to track who created a certain request in their current scheduling system. They therefore asked for a way to see who created the request and at the same time they proposed that a request should have a vehicle number to facilitate their planning. Figure 5.1 shows the visual changes made.



**Figure 5.1:** Illustration of how the visual representation of a request changed in the web application. The left visual is before while the right is after.

### 5.3.2 A Request Acting as a Work Shift

Because of the changed interpretation of a request, the design was modified. Initially, the mobile application followed the same assumption as the web application had, which was to display the task to be made at a certain period of time. Figure 5.2 shows the iteration of the design that follows the suggestion from the stakeholder as seen in Figure A.11.



**Figure 5.2:** A screenshot of the first iteration of displaying requests in the mobile application.

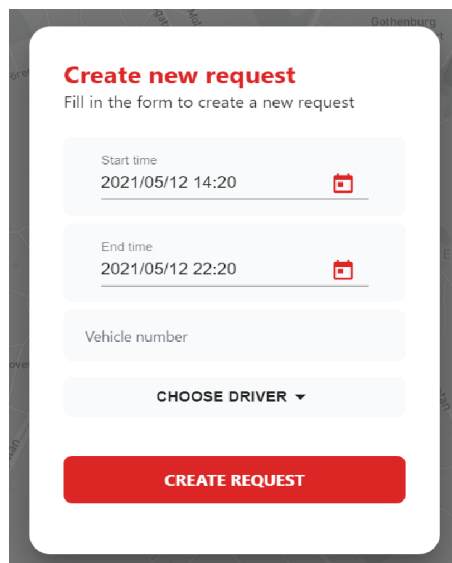
However, having multiple requests where each have a “Share position” button would cause information overload and possibly confusion for the user. Removing these position sharing buttons and only having one instead, would make it difficult to understand which request that single button would refer to. This reasoning, combined with a request acting as a work shift led to a makeover of the main page of the mobile app, which can be more thoroughly seen in Section 6.1.3.

### 5.3.3 Optimizing the Request Creation Process

A question that came up was if there was any way to optimize the process of creating requests. The stakeholder has mentioned that they have approximately 70 drivers hired for each event with each driver having on average four requests which leads to a total of 280 requests for an event. Creating this many requests is time consuming and an optimization would be efficient for the coordinators. Two suggestions were:

- The possibility to send a request with the same time interval to several drivers.
- The possibility to send several requests with the same time interval to one driver over several dates.

The stakeholder’s scheduling process is too irregular and unpredictable for the feature to add value. However, a feature that would facilitate the process was the end time to dynamically change to eight hours after the start time when creating a request, since the usual working shift lasts for eight hours. Figure 5.3 shows the dialog for creating a request. The end time automatically sets to eight hours after the start time whenever it is changed.

The image shows a mobile application interface for creating a new request. At the top, it says "Create new request" in red, followed by "Fill in the form to create a new request". There are three input fields: "Start time" with the value "2021/05/12 14:20", "End time" with the value "2021/05/12 22:20", and "Vehicle number". Each time field has a calendar icon to its right. Below these fields is a button labeled "CHOOSE DRIVER" with a dropdown arrow. At the bottom is a large red button labeled "CREATE REQUEST".

**Figure 5.3:** The dialog for creating a request in the web application

## 5.4 Implementations of Features for the Mobile Application using Flutter Plugins

The mobile application was implemented with the use of several plugins, that resulted in saved time and effort when implementing functionality that are needed for the app to work. How these were used and implemented in the project will be discussed in this section.

### 5.4.1 Implementation of the Position Tracking Feature

The most important functionality needed was for the mobile application to be able to send its current position to the database, even when the phone is tucked away. Since creating a position tracking feature from scratch would take a large amount of time to research, the flutter plugin “background\_location” was chosen for its desired functionality. By using this plugin, one of the requirements for the project was completed, that the driver’s phone should be able to send its position to the coordinator continuously during a workday, even when idle.

### 5.4.2 Authentication of the Mobile Application

A requirement needed was for the application to be secure, a driver should feel safe when using the application and not worry about someone accessing their account. For implementing this feature, the Firebase plugin *firebase\_auth* that handles authentication was used. It works as follows: When a driver signs in to the application with their email and password, their password becomes encrypted and sent for verification. The coordinator side cannot find out the driver’s password by looking in the database. They can only find out whether the sign in attempt was successful and if so allow the driver to sign in to the dashboard page.

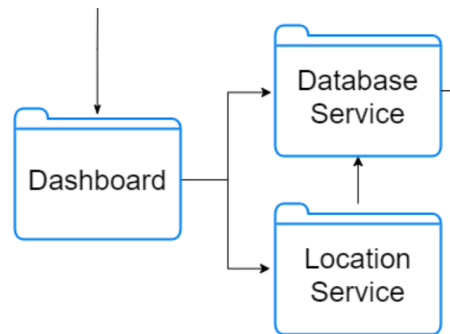
### 5.4.3 Sending out Notifications to the Drivers

An extra feature proposed by the team was to implement push notifications to the drivers. A push notification in this case works by using Firebase cloud functions which trigger when data changes in the database, for example, when the position tracking are discontinued. If that happens, the database sends a push notification using Cloud Messaging to the affected driver informing that the position tracking has automatically been stopped. The purpose of the notification feature is to send reminders or updates to drivers when a work shift is created, edited or has ended.

### 5.4.4 Uploading Data to Firestore

Sending the position data to the database was another issue encountered. The chosen database service was Firebase and it offers its own classes and methods one can interact with within Flutter. Following the pattern from previous services, all of the Firebase calls were separated to their own service module which can be seen in Figure 4.2. The service then handles all of the reads and writes to the cloud

database. If for example a signed in user wants to share their position, the data flows from pressing the share button which calls the location service and then use the database service to upload data to the cloud. The flow is shown in more detail in Figure 5.4 where the arrows represent the flowing direction of the information.



**Figure 5.4:** The data flow when sending a user position to the database shown using arrows.

## 5.5 Adding Mapbox to the Web Application

To follow the separation of concern a new component called “FullscreenMap” was created. This component has an absolute positioning to fill the entire screen of the client window and the z-index of zero to be rendered below other components along the z-axis. The package “mapbox-gl” provided by mapbox for javascript applications was added to have access to instantiating methods. A div element was created in the window containing a reference whose content is rendered to be the map at runtime. When the map is done rendering, a method will run to render all the additional components to the map.

### 5.5.1 Searching for Places by Geocoding

Provided by Mapbox is the “MapboxGeocoder”. This component has a search function which, depending on the search query, returns a list of suggestions. For the map in the project, two geocoders were instantiated. One to be able to perform search queries and another for reversed geocoding, which allows coordinates as queries and getting the address as a response when left clicking on the map.

### 5.5.2 Interpreting Database Documents as Markers on a Map

The application is wrapped by the context “ApplicationData”. This context is constructed by an initial state for the application that is populated once with data from the database. This data can then be used throughout the application and is updated as soon as the database documents are changed.

The Firestore database has a default type for locations called “GeoPoint”, which consists of longitude and latitude values. They needed to be adapted to a desired format for the markers layer to be compatible and applicable to the map. The desired format is called “GeoJson”. The document should be of the following form:

---

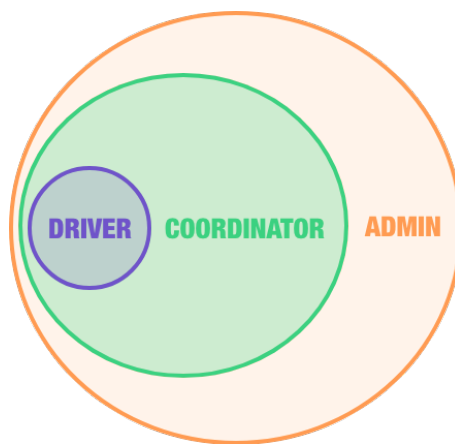
```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [longitude, latitude]
  },
  "properties": {}
}
```

---

The context is then used by the map component to access locations on this form. Additional data is provided within the “properties” parameter, which in this project contains the name of the location, its address, postal code and city.

## 5.6 Security of Firebase and the Role Hierarchy

To provide the security stated in the requirement chapter (Chapter 2) and still be able to use the same authorization throughout the system, a role hierarchy was introduced. The functionality of the roles are based on the minimum access level and ability for data mutation, while still fulfilling the desired purpose. The hierarchy is divided into three different roles where a higher role has all functionality of the lower ranked roles.



**Figure 5.5:** Illustration of functionality of roles. A role of higher hierarchy contains all functionality of the lower ones with additional features.

More specifically the roles will contain the following functionality:

- **Driver:** Allows signing in to the mobile app, sharing position data, and read data about itself.
- **Coordinator:** Allows signing into both the web app and the mobile app. In the web app, this role allows creating and deleting of drivers, requests, and stored locations.
- **Admin:** Same authority as the coordinator but with the extension to create new admins and coordinators, editing the office phone number and clearing the archive.

### 5.6.1 Handling Roles Using Custom Claims

The handling of the roles is performed by implementing cloud functions served in Firebase, using custom claims. Custom claims are additional parameters that can be added to an instance in “Firebase Authentication” from a server using the “firebase-admin” package. These parameters are hidden and can only be accessed by server-side calls. This is to prevent exposing the internal structure to the client and to inhibit security breaches through modifying code locally in the browser. The actions executed on the server are triggered by http calls, taking a data object and the context from which the call was made from, as parameters. The cloud function executed on the server is using the given parameters and the client receives a response. The different actions are:

- **Creating a driver:** Checks if the call is made from a context where the caller has the custom claim of “coordinator” or “admin”. If security checks are passed the driver is created in Firebase authorization as well as a user instance in Firestore.
- **Creating a coordinator:** Checks if the call is made from a context where the caller has the custom claim “admin”. If security checks are passed the coordinator is created in Firebase authorization with the additional custom claim of “coordinator” as well as a user instance in Firestore.
- **Check if access to web application:** Called when trying to sign in from the web application. Takes an email as a parameter, checks if the user with this email has the authority to sign in, and returns a boolean response or corresponding error.
- **Check if admin:** Checks if the signed in user has custom claim “admin” to permit extra features on the web application.

### 5.6.2 Additional Security Rules in the Database

Within Firestore, the data can be protected by security rules. To apply a certain level of security for the documents in the database, every collection has its own rules. To be able to supply the desired functionality and not expose anything that is not needed the collections got the following rules:

- **Collection of users:** Can only be read by a user with the hierarchy of coordinator or higher. If the user has the role of a driver it can only read the

document in the collection that match with the signed in user's id. If the call is done from a not signed in context or the driver's id does not match then reading and writing are forbidden. Writing to the collection can only be done by a signed in user with the role of coordinator or admin.

- **Collection of requests:** Applies the same rules as reading and writing to the user collection. Coordinators and admin accounts can read and write to the collection while drivers only can read documents containing their own user id.
- **Collection of user positions:** Can be read if the call is done from a signed in context where the user has the role of coordinator or admin. Writing can only be done by a signed in context where the user has the role of driver and the parameter "userId" of the document match the id of the signed in user.
- **Collection of locations:** Can be read and modified by signed in users with the role of coordinator or admin.

## 5.7 Testing the Applications

In this section, the process of testing the applications will be presented. It will include what methods were used to gather unbiased data.

### 5.7.1 Methods of Testing

The several test cases which will be mentioned in Section 5.7.2 and Section 5.7.3 were continuously evaluated during code reviews to see whether features were still upheld. If more testable features came up, they would be added to the list of test cases and ensure to keep the system at a high quality and not to interfere with already implemented code. Additionally, a user test was conducted on a larger group of people. More detail about the user test can be found in Section 6.2.

### 5.7.2 Testing the Web Application

The main functionality of the application, showing the drivers' position on the map, can only be tested together with the mobile application. Testing this is done by checking that the driver marker appears on the map when the driver shares position and checking that the marker moves when the driver moves. Another important case to test is if it performs as intended when multiple drivers share their positions.

Apart from the functionality that can only be tested in conjunction with the mobile application, there is some functionality that can independently be tested on the web application.

The system relies heavily on interacting with the database, more specifically writing data to it and listening to changes in it. When it notices changes it updates its visuals to correspond to what is on the database. There are five functionality aspects that are directly linked to the database and that can easily be tested. These are:



- **Drivers:** Creating new drivers writes to the database and updates the drivers list, which is represented visually in the web application.
- **Requests:** Creating new requests or editing existing ones results in a modification to the database and updates the requests list, which is represented visually in the web application.
- **Locations:** Creating new locations or deleting existing ones results in a modification to the database and updates the locations list, which is represented visually in the web application.
- **Signing in/Signing out** When signing in, the web application must contact the database which in turn must use its authentication to check if the login details correspond to a coordinator or an admin.
- **Deleting user account:** By pressing the delete account button the account gets deleted on the database and the web application redirects to the login page.

To test the three first functionalities, one could create a new request for example or edit an existing one, and then check whether the visual list on the web application has been updated. If it has been updated correctly then one knows that it is working. Each of these lists has a search bar that when searching for a request, filters the list by the search query.

For testing the last two functions one can try signing in with both a valid and invalid login details, and check if one gets signed in correctly. Lastly, one can press the delete account button and check whether the account gets deleted on the database and that the web application redirects to the login page.

Beyond the database relied functionality there is functionality that is handled by the web application alone. These functionalities are listed below:

- **Maneuvering on the map:** Pressing the left mouse button and dragging the map should move the view of the map. Zooming by using the scroll wheel on the mouse should zoom in or out on the map.
- **Showing/Hiding locations:** The location markers on the map that have been created can be toggled to be visible or hidden by pressing the assigned button that is situated on the lower left corner of the screen.
- **Right click on list items:** The four different lists which are the drivers list, requests list, locations list and archive list, show all of their content as list items. These are right clickable, which when doing so creates a little menu with the choices edit, delete or only delete, depending on what item it is.
- **Search bar for addresses:** When searching for an address the search bar should give suggestions on addresses similar to what is written, and when searching for an address the map will move the vision to that location.

The specific web application functionalities listed above can easily be tested by trying them and checking whether they result in the expected outcome.

### 5.7.3 Testing the Mobile Application

As previously mentioned, the mobile application relies heavily on interacting with the database and also communicate with the web application through it. The major functionality of the mobile application is to provide a way for the driver to share their position to the coordinators. Testing this could be done either by using the web application or more directly by checking the database. One should be able to clearly see when a driver starts sharing its position, stop sharing, and if it is updated continuously.

There are eight functionality aspects that are directly linked to the database and that can easily be tested. These are:

- **Showing the driver's request:** Only the requests that the driver is assigned to should be shown in the driver's application.
- **Start sharing position:** The position sharing should only be able to be done if it is during the request's duration. When sharing of the position is started, the user's position is written to the database and it should be clear visually that the position sharing is active.
- **Stop sharing position manually:** When sharing of the position is stopped, the user's position must not be written to the database anymore and it should be clear visually that the position sharing is not active.
- **Stop sharing position automatically:** When the duration of the request has passed, sharing of the position will automatically stop and the driver's position must not be written to the database anymore. This should also be clear visually as well as a notification will be sent to the driver's mobile.
- **Signed in:** When signing in, the application must contact the database which in turn must use its authentication to check if the login details correspond to a user.
- **Signed out:** When signing out, the application must contact the database to sign out. The position sharing should automatically be stopped.
- **Changing of password:** The first time a driver signs in, they must change their password to a personal one. They are later able to change it.
- **Signaling the status "isLate":** When sharing the user's position, the driver has the option to signal that they are late, which is shown for 30 minutes.

To test the four first ones, one must create several different requests using the web application that gives several different scenarios, such as requests with other users, requests with the duration before or after as well as requests that are currently able to be activated. The ability to share position should only be possible when certain requirements are fulfilled, like no other device can be signed in to the same account and share their position and the time has to be within the request's time span.

Most of the mobile application is dependant on the database and the web application, but there is some functionality that can independently be tested.

Additional tests that can be conducted without the use of the database or web application could also be relevant for example battery consumption which is important to keep track of. This type of test could be conducted by sharing position using the mobile application. Another test is trying to maneuver the different pages of the application to check whether it is intuitive for the targeted audience. The targeted audience being the stakeholder's employees.

# 6

## Final System and Result Assessment

This chapter will cover the final result, explaining the capabilities of the system and the visual representation of the mobile and the web application.

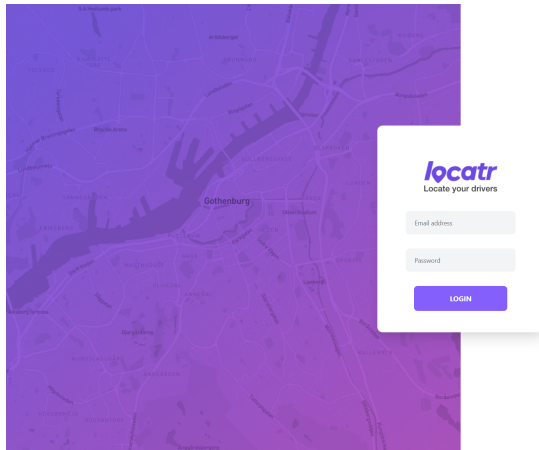
### 6.1 Finished Product and Fulfilled Requirements

Developing according to the requirements of the stakeholder and listening to their current procedures has been crucial to understanding their needs throughout the development process. This helped to avoid the already limited time from being spent on features that will not be of any value to the stakeholders.

This presentation of the finished product will be referring back to the requirements described in Section 2.1 with the additional features that were implemented. There is also a section covering the adaption to the laws and security that are discussed in Section 2.1.1.

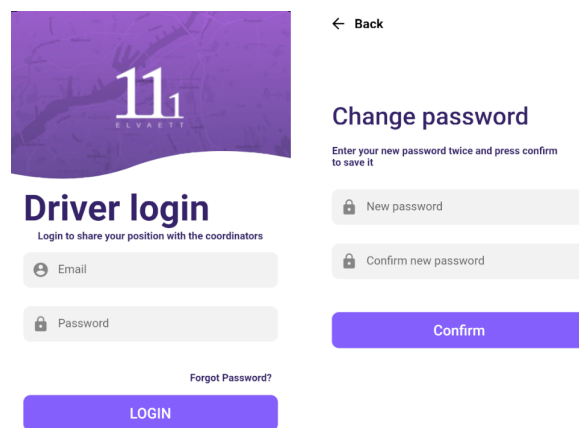
#### 6.1.1 Login Pages, Authentication and Authorization

If a user enters either the mobile application or the web application without authentication, they will automatically be sent to the login page and not be able to enter any of the other pages. When the user has put in their username and password and has clicked the login button, the application sends the data to the database for authentication. The authentication is handled by the Firebase Authentication service and authenticates the person that is trying to sign in. Firebase also handles the authorization for the different roles which was discussed back in 5.6. Admins and coordinators are authorized to sign in to both of the applications while drivers can only access the mobile application.



**Figure 6.1:** The web application login page.

When a new user is created in the web application, an email is sent to the user with a auto generated password. The first time the user signs into the mobile application, they are prompted to change the given password to their own to make it easier for them to remember their password. They are also given the option to change their password in other situations (see Figure 6.2). This is done securely by using Firebase Authentication that does not allow the changing of passwords unless recently signed in and will reroute the user to the sign in page if needed.



**Figure 6.2:** The mobile application login page and change password page.

### 6.1.2 Web Application User Interface

The web application has, beyond the login page, one main page where the map is located. The map page has a search bar on the upper left corner and five tabs on the left. Clicking on the tabs opens up an oblong card on the left side of the page to the right of the tabs, that shows the selected tabs' content.

Figure 6.3 shows the map page which is mainly covered by the map navigation since the main functionality of the web application is to track drivers. Tracking drivers and seeing their positions in relation to the position of locations were also the main requirements from the stakeholder. This is implemented by displaying drivers as purple dots and the locations as orange markers on the map.

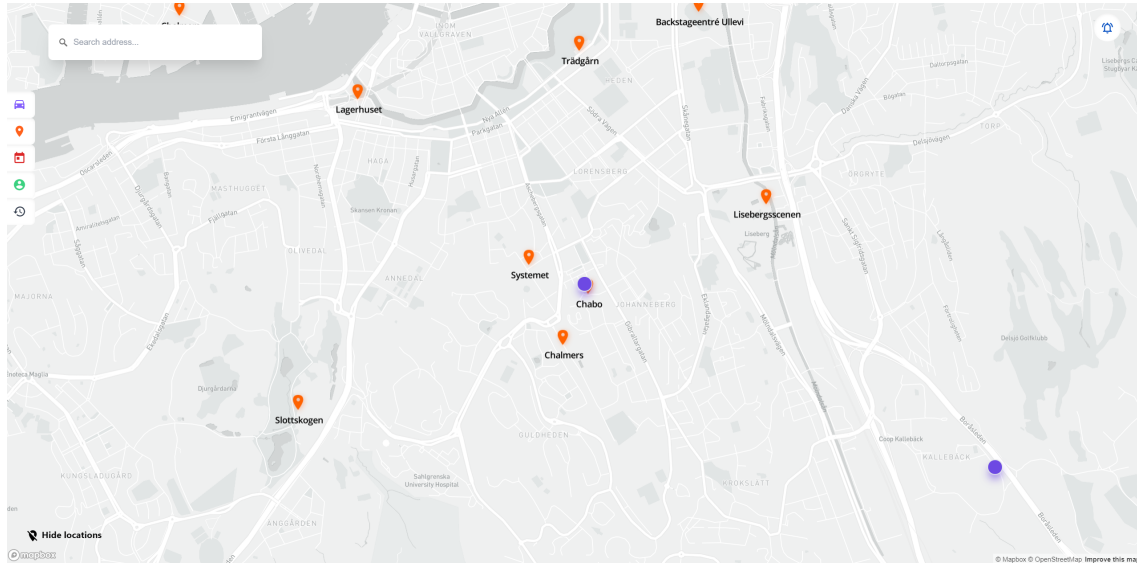


Figure 6.3: The map page.

The five different cards shown in Figure 6.4 are: drivers, locations, requests, account details and archive. All the cards except the account details card show a card with a search bar and a list of all the items of the selected category. The archive contains requests which are expired and these could assist the stakeholder with the payroll management. The account details card shows a smaller card with information of the currently signed in account. This card has buttons to sign out and delete account and, if signed in as an admin, also a button to create a new coordinator and a button to edit the office phone number. In the drivers and locations cards there are buttons to add drivers and requests respectively.

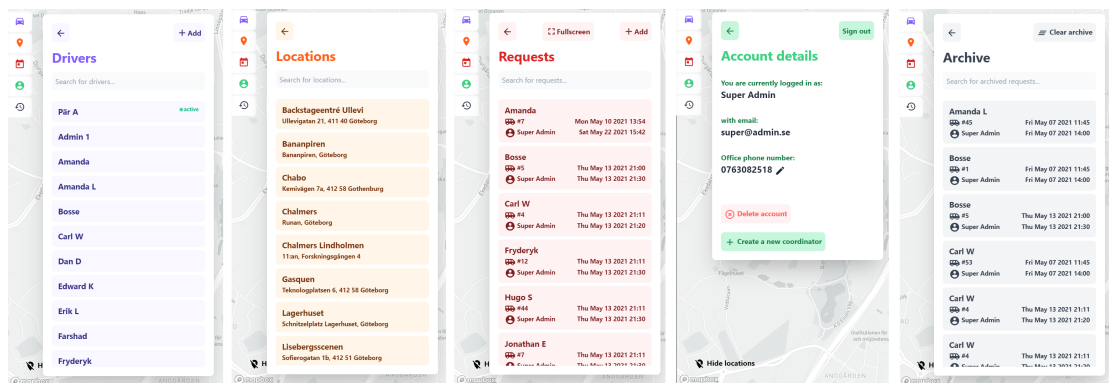
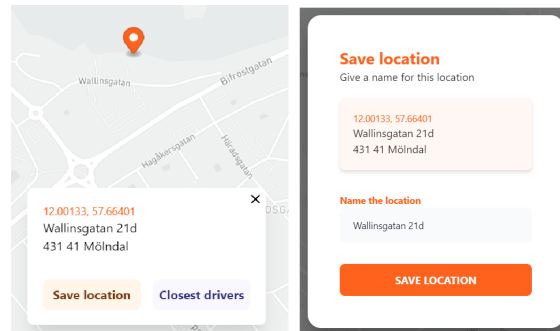


Figure 6.4: The five different cards.

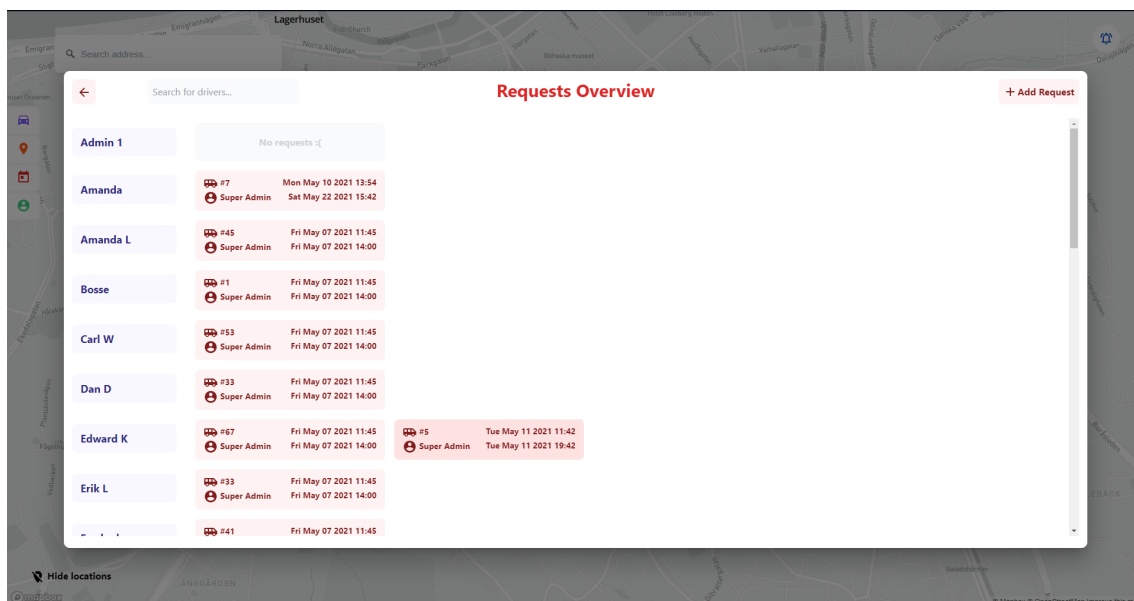
## 6. Final System and Result Assessment

Adding a location is possible by left clicking anywhere on the map which will then open a snackbar that shows the option to save those specific coordinates as a location. Figure 6.5 shows the snackbar and the location saving dialog which also gives the user an option to input a custom name for the location alongside its address.



**Figure 6.5:** The process of adding a location

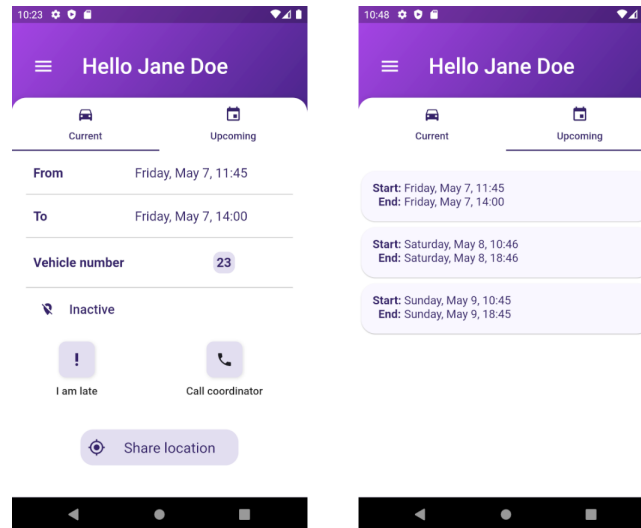
The requests card has a full-screen button, which opens a requests overview on top of the map that covers 80% of the page. It shows all the drivers in alphabetical order in the column to the left with their respective requests horizontally. There is a vertical scroll to the right that scrolls through the drivers and a horizontal scroll on the bottom that scrolls through the requests of the visible drivers to the left. The requests are positioned chronologically, so the request that is upcoming is closest to the driver box and after that comes other requests in time from soon upcoming to furthest away in time. All of this can be seen in Figure 6.6 below:



**Figure 6.6:** The Requests Overview where coordinators can see all the drivers and their requests

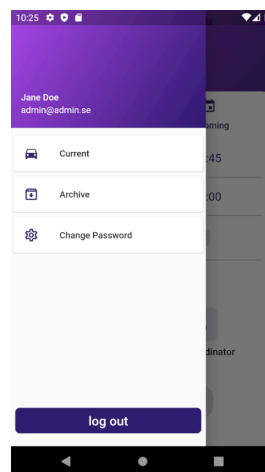
### 6.1.3 Mobile Application User Interface

The phone application consists of two main pages, the login page and the dashboard page. The dashboard is intended to display the most vital information to the driver.



**Figure 6.7:** The dashboard of the mobile application, displaying both tabs.

The dashboard page works as a schedule for the driver, displaying which time the driver is supposed to share position and which vehicle to use for the requested work shift. The request on the “Current” tab will be the next upcoming one and will be displayed 24 hours before the shift starts. Pressing the “I am late” button will send a notification to the coordinators with a timestamp when the button was pressed. This is to ease the communication between the coordinators and drivers. The dashboard also displays the coming requests for a driver in the “Upcoming” tab.



**Figure 6.8:** The profile menu in the mobile application.



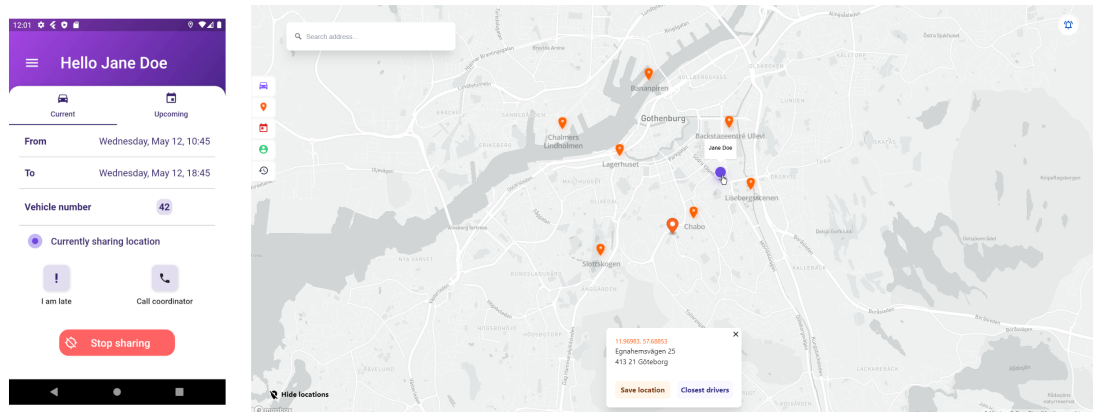
A profile menu is included to show user information with options to check archived requests and to change password. The log out button is in this menu as well.

### 6.1.4 Position Sharing

To share a position, the driver must press the “Share position” button within the time duration they are expected to. This will create a position variable for the user in the database that updates in fixed intervals.

The position sharing will automatically stop after the end time is reached. The driver cannot share the position until 10 minutes before the start time. The driver can also choose to stop the position sharing at any point.

While the position sharing function is turned on, a purple point of the driver’s position will be displayed on the web application as seen in Figure 6.9. Hovering over the purple point on the map will display the driver’s name.



**Figure 6.9:** A screenshot of a location being shared.

The “Driver” card will display who are currently sharing their position as seen in Figure 6.10. Pressing any active driver’s name on the menu will fixate the map on their position and follow the driver until any other action is done. The bell top right will alert the coordinators if a driver is late and will include the timestamp for when it was pressed.

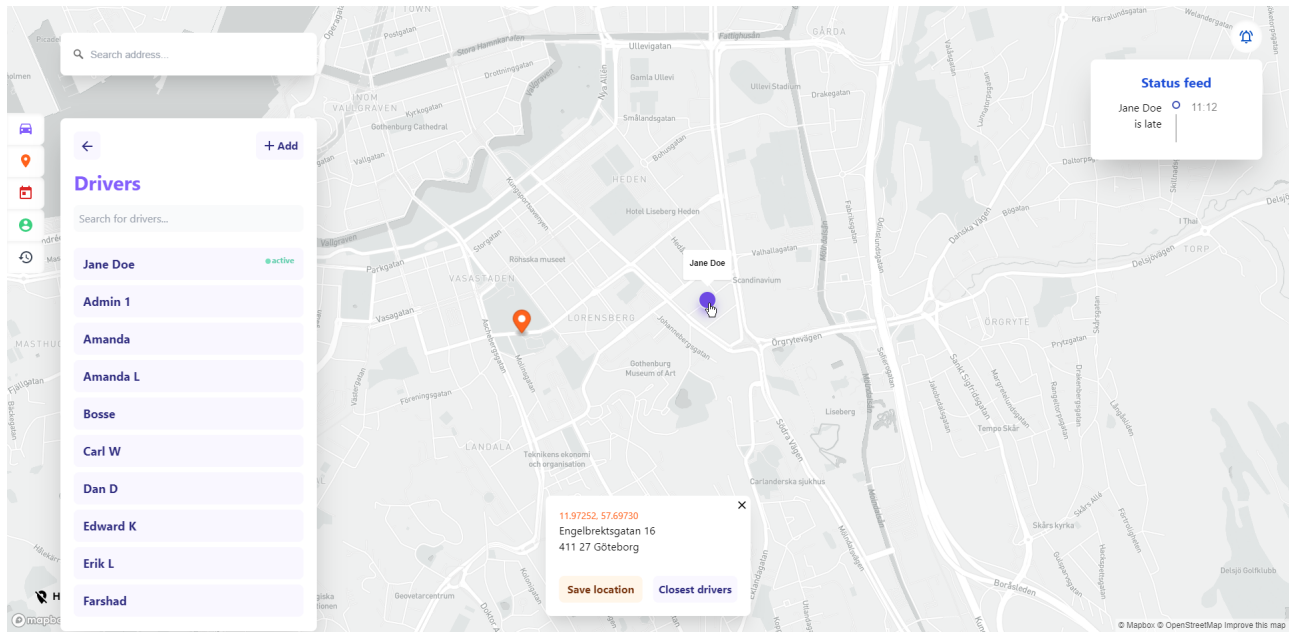


Figure 6.10: A screenshot of a position being shared on the web application

The closest driver function, as seen in Figure 6.11, was a requested feature from the stakeholders. The purpose of it is to give an indication of which driver is closest to a chosen location. This is for the coordinators to quickly assign a spontaneous task to the closest available driver.

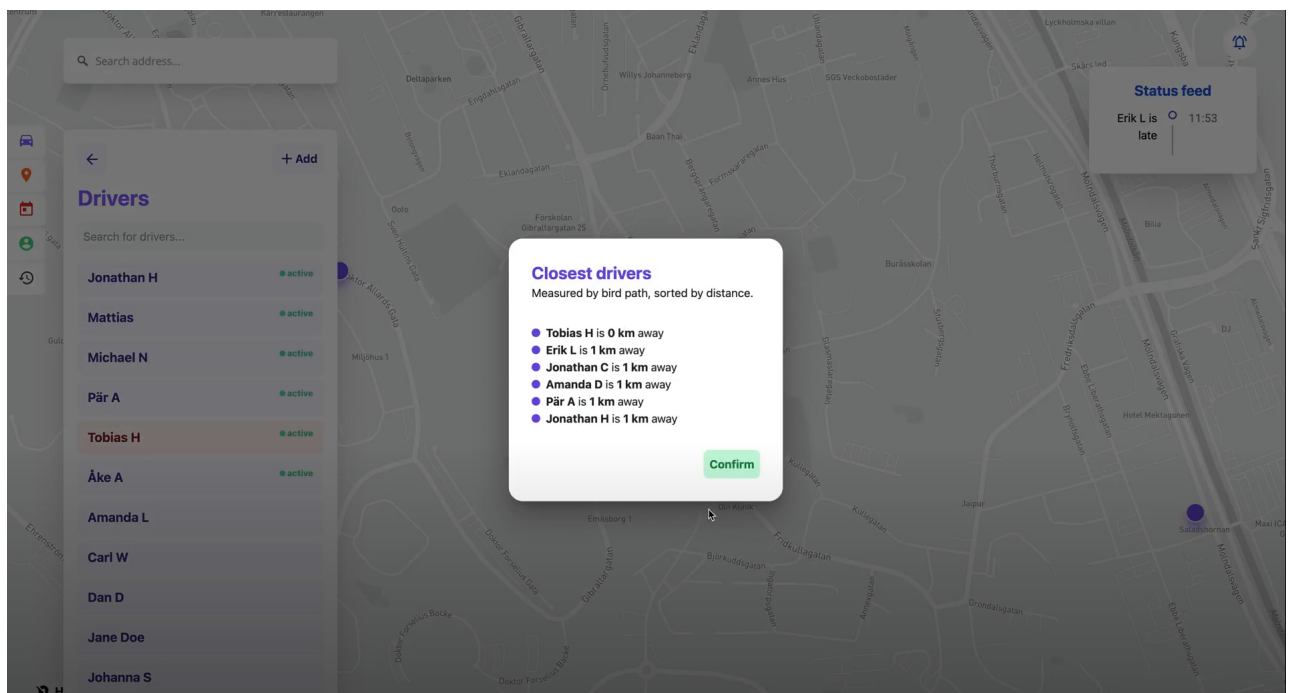


Figure 6.11: An image displaying the closest drivers in a list.

### 6.1.5 Adaption to Laws and Security

As mentioned in Chapter 2.1, the system needs to follow certain laws which primarily was the General Data Protection Regulation (GDPR). GDPR, as discussed, imposes requirements on organizations everywhere, as long as they target or collect personal data related to people in the European Union [3]. Examples of personal data that are used in this system are names, email addresses, position information and drivers' work shifts.

In the development of the system, GDPR had to be taken into consideration to make the system useful for the stakeholder by following the rules and implementing the appropriate methods. This makes the system useful but the responsibility for following GDPR and making sure everything works correctly is in the end the company's responsibility.

A requirement given by GDPR is that the organization should only collect and process as much data as absolutely necessary for the specific purpose, and only store it as long as it is necessary [3]. An example of how this have been taken into account in this product is that the position of a driver is only stored until the next position is shared, since the previous position is then no longer necessary and should be removed. The position of the driver is also removed when the position sharing is inactivated, whether it is done manually on the driver's phone, automatically by the work shift ending or the driver signing out of their account.

The position of the driver is not the only personal data that are saved. The users also have a profile connected to their account with their name, email address, tokens used to send notifications to their mobile phone as well as requests for sharing position with their future and past work shift. This information is saved until a user is deleted, which can be done by the coordinators in the web application. When this happens, most of mentioned data is removed. An exception is the requests for the past work shifts as well as the token. These are yet to be included in the process of the deleting a user, resulting in it being mentioned in future improvements (see Section 7.2). This personal data can still be deleted but one have to do it manually in the web application for the past requests and directly in the database for the token. This complies with the law that a user should be able to ask for the removing of all their personal data [3].

Another requirement of GDPR is that the processing and storing of personal data must ensure appropriate security (see Section 5.6), integrity and confidentiality [3]. The access to the web application is restricted to only coordinators and administrators (see Section 5.6), in an effort to further ensure the privacy and security of the drivers. This means that the drivers are not able to get access to any information about other users or their position, which complies with the requirement of GDPR to process and store personal data in a way that ensure confidentiality.

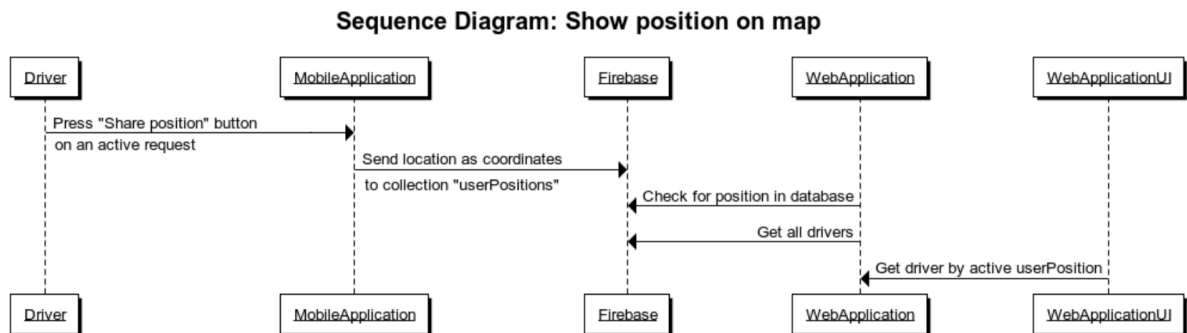
As previously mentioned, an aspect that needs to be taken into consideration to follow the rules of GDPR is the integrity of the drivers. One needs to consider their right to privacy during their spare time. This means it is important that the posi-

tion sharing is only active during their working hours. This is achieved by letting the driver activate the position sharing themselves, turn off the position sharing and restrict the position sharing feature to the time interval of a request. The requests, as previously mentioned, corresponds to a work shift and when the work shift is over, the position sharing will automatically be turned off. By doing this, the position sharing should never be active during the driver's spare time and the driver has control over when the position sharing is active.

Since the system processes and stores personal data, its subject needs to have given an informed, specific and unambiguous consent to the company's processing of their data [3]. This could be done by creating an user agreement which include information of how the data is collected, used and what rights the user has. The company is responsible for collecting consent and it is not something that is included in the system.

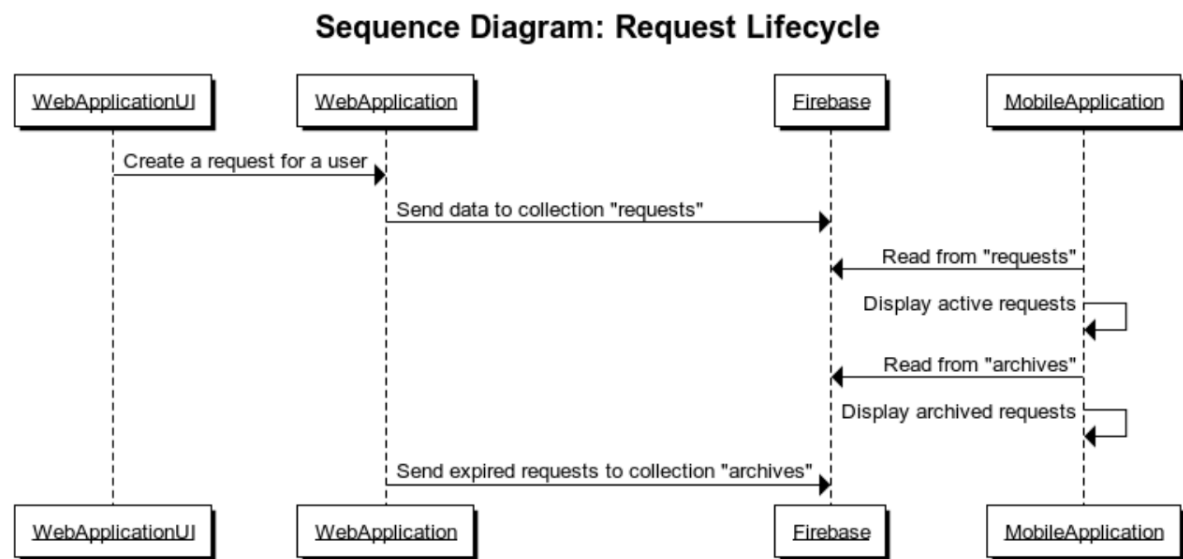
### 6.1.6 The Flow of Interaction

The sequence in Figure 6.12 shows the basic idea of communication between the two applications. The database works as a common endpoint for the communication. Methods to check for positions to display are only triggered on changes in the database to prevent constant unnecessary reads.



**Figure 6.12:** A sequence diagram of how a position is displayed on the web application started from the driver pressing the “Share position” button

As seen in Figure 6.13, the lifecycle for a request follow the same pattern as sharing a position. The communication is through Firebase and any updates in Firebase will trigger changes on the mobile application, for example adding a new request.



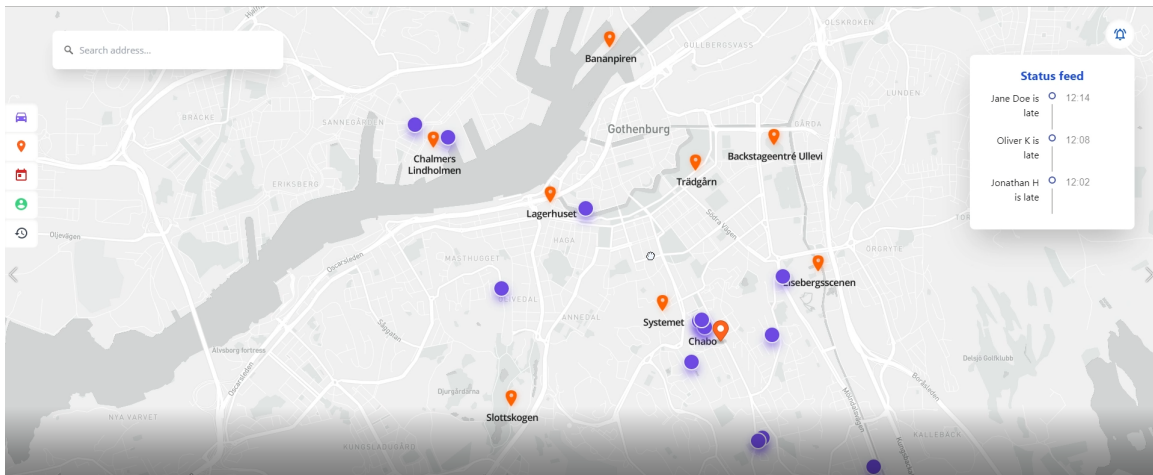
**Figure 6.13:** The lifecycle for a request starting from the creation from the web application

## 6.2 Evaluation of User Testing

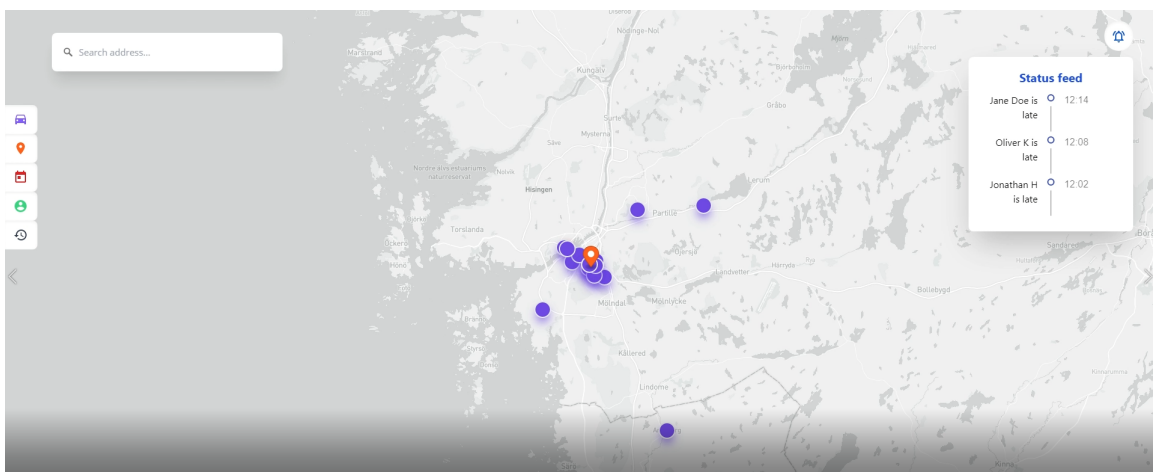
A major test was conducted close to the end of the project, to test the performance of both the mobile and web application and investigate whether the applications could remain stable with several users and also the points described in Chapter 5.7. To ensure the system's stability for the stakeholder, it was specified that about 80 people would be needed to use it at the same time to stress test the system. Due to lack of time, a smaller test was conducted to see whether the system would work when at least 20 people used it at the same time. A selection of family members and friends were asked to participate which totaled in 20 people participating. Having gathered enough participants, the test could then be executed.

The premise was that all the participants would download the latest version of the mobile application and activate the app's position sharing feature. They would share their position during a period of two hour and during this time the web application was monitored by group members to evaluate its performance.

Figures 6.15 and 6.14 show the web application during this testing period. A major issue that occurred during the testing period was that the mobile application's connection to our database was severely unoptimized because it read and wrote data to Firebase too often. The free usage of reads and writes to Firebase was exceeded after 30 minutes of up time and as a result, a decision was made to temporarily pause all interactions with the database, shutting down the test. All participants received an evaluation form after the test ended and was asked to give feedback of the experience.



**Figure 6.14:** An image of the web application during testing. The image shows the area of *Gothenburg*.



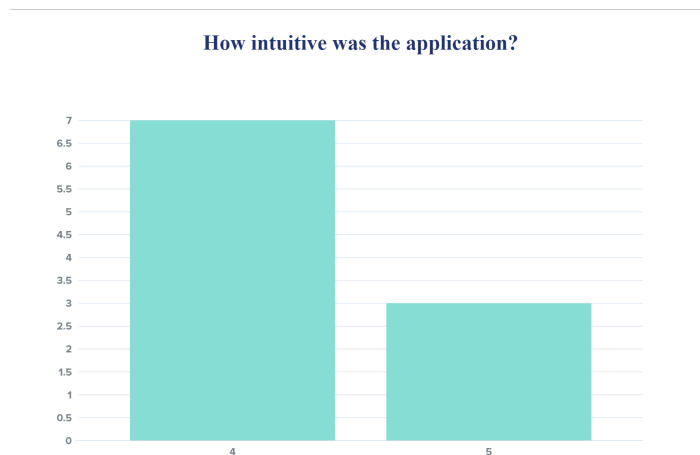
**Figure 6.15:** An image of the web application during testing. The image shows the area of *Västra Götaland*.

Despite some challenges, the test gave some valuable insights on issues that need to be solved for the mobile and web application to work properly, these will be listed below. Feedback from the evaluation form written by the test participants and other noticed issues will be included in the list. Overall, both applications performed as expected and after these issues have been solved then the system could be ready to be used by the stakeholder.

- **Read and writes to Firebase:** Need to drastically decrease the amount of reads and writes from the mobile app to Firebase.
- **Web application ran slow:** The web application worked slower than expected when there are several active users
- **Features not working as intended:** Users had issues changing their password where the confirm button did not give any response.

- **Faults with UI:** Some users thought the loading wheel was poorly placed and looked odd at some places where it was implemented.
- **Incorrect icons:** The icon that is shown in a users notifications bar did not have its right icon but and was instead a gray transparent box.
- **The phone's GPS need to be turned on:** The GPS-service are not enabled using the application, so it is possible for a driver to turn on the position sharing feature without having its phone's position sharing active.

A feedback form was sent to the participants of the test after it was finished to obtain insight on any problems and uncertainties when exploring the user interface and navigation within the mobile application. The feedback form was not filled by every participant but can still give a good insight on how they experienced the application. Figure 6.16 shows the result from how intuitive the participants found the application where a five was the highest possible grade and one the lowest.



**Figure 6.16:** The result from how intuitive the participants found the mobile application. The x-axis represents the grade and the y-axis represents the frequency.

This was clearly a positive result but worth mentioning is the lack of elderly participants. The ease of use especially for the older generations was a requirement briefly mentioned in Chapter 2.1 and a larger amount of older test participants had therefore been desirable.

Alongside feedback about the user experience were also some feedback of technical character such as battery consumption and mobile data usage. The average mobile data usage was 43.5 MB/h and the average battery consumption was 245.5 mAh after 30 minutes. It is not unusual for a mobile phone to have a battery capacity up to 4000 mAh which means that the battery can last for almost 10 hours while using the application [18]. The stakeholders provide the company cars with chargers so battery drain should not be a problem. The mobile data usage result could be more problematic since it potentially could reach up to 2 GB after a 40 hour work week. This leads to a minor cost for the drivers which would have to be compensated for.

## 6.3 Ethical and Societal Aspects

The ability to use GPS to trace the movements of people raises important ethical issues. Since the application is based on giving coordinators access to and store the geographical position of employees during their working shifts, this feature cannot be removed from the project. Personal position tracking presents a global concern and geolocation privacy legislation is in place in some countries. The protection and privacy legislation provided by GDPR is an important ethical aspect and needs to be adhered by the system. This is discussed in details in Section 6.1.5.

Another ethical aspect that needs to be considered is how the data is used. It is essential that the position is used only in purpose to help the coordinators in assigning tasks and assisting the drivers, and not to monitor their personal lives. This is something that is hard to ensure fully. There is possible ways of decreasing the risk of it happening but it would remove important aspects of the system. One example is to remove the possibility to see which driver is connected to each position. However, this would obstruct the purpose of the system since the coordinators need to be able to connect a position to a set of tasks and be able to communicate to the specific driver when needed. There is no obvious solution to this problem but since the position is not saved after the work shift is over, the risk lessens since the employers cannot observe it afterwards.

### 6.3.1 Driving and Using a Mobile Application

The hazardous nature of using a mobile application while driving is one of the motivations behind the development of this product. With this application, the coordinators will avoid calling the drivers during their work shift in the same capacity to prevent disturbing them while driving, as motivated before. This leads to a more secure working environment for the drivers and safer trip for their passengers.

The driver should start sharing their location when they begin their work shift. This should be done before they start driving and they will not have to use the application further while driving. As a result, the mobile application will not be needed to use the rest of the work shift and the driver should not be distracted by the mobile while driving.

There is one feature that is an exception and could be used while driving. It is the feature to report a delay in the mobile application. This feature needs interaction from the driver and the intention is that this should be used when parked in a pick-up place while waiting for delayed passengers. The problem that could occur is if the driver falls behind schedule due to other circumstances. This potentially leads the driver to use this feature while driving. This is not something that should or will be encouraged but the drivers still have the possibility to divert from the recommendations of the coordinators. A solution to this problem will be discussed in Section 7.2.



# 7

## Discussion of the Results

In this chapter, a discussion will take place about the results from chapter six, give the group's thoughts, and state what aspects of the work that was thought to have went well, along with aspects that could have been done differently if the project was done again.

### 7.1 Reflection on Project Workflow

From the beginning, we thought that this project would be simple considering the amount of time we had; 16 weeks with 20 hours of work per team member each week. This was not the case as it took longer than expected to implement different features. For example, to finish developing the core functionality of the system, which is being able to send a driver's position to a coordinator, took twice as much time to implement than expected. Aspects that were thought to take a small amount of time took longer and given the large amount of time the group had at the beginning of the work, the ambitions to accomplish several features became too much to handle.

The uncertain time estimation did not become an issue, however, since the work was structured to follow agile methods it was easy to set priorities to functionality that must be done and optional functionality which improves the system. Using agile methods also supported a constant dialog with the stakeholder, which enabled them to give their input on implemented features, making sure that the project was on the right path during the whole process. As mentioned in Chapter 6, many ideas changed after these meetings with the stakeholder.

An aspect of the work which could have been done better was the system testing, which could have been done at an earlier stage. The group felt that there still could be improvements to the application so the thought of testing was low prioritized resulting in some new issues that came unexpectedly after testing. If the testing had been done a few weeks earlier then the issues could have been dealt with but with the small amount of time left, the issues are instead left documented.

### 7.2 Future Improvements to the System

There is always something one can improve with software and since there is not enough time to work on the project, the group reflected on some issues that could be fixed along with future features to implement. Major issues that currently exist

within the system are those that came up during the performance testing.

One of the issues that occurred in the mobile application was that it sent too many requests, to read and write, to the database than what was necessary. It is deemed as a major issue since the mobile application will not scale well when the amount of connected users increases. This issue could be fixed by storing more data locally in a user's phone, instead of constantly fetching that same data from the database. Issues faced by the web application were similar to the mobile as a major issue was the amount of reading to the database causing the web application to run slow. A solution to this problem could be to decrease the amount of reading from the database which could also reduce the amount of mobile data used by the mobile application (see Section 6.2).

A major issue found was that the permission to use the GPS service is asked for but the application does not actively check that the GPS service is enabled. If the user's phone does not have the GPS service enabled already, the sharing position can be started with the indication that it works but position is not actually shared since the GPS service is disabled. Therefore, an important development that is essential to the usability of the system is that if the GPS service is disabled, the user must be prompted to enable it.

An important improvement that needs to be done is to make sure that in the process of deleting a user, that everything connected to the user is also removed. Currently, past requests and the token used when sending notification to the user are not included in the process and should be added to easier adhere to GDPR (see Section 6.3).

Setting the focus on future features instead of issues, there are some which have been deprioritized. One of them is to expand the use of notifications being sent to a driver. As of now, that feature became deprioritized but a code layer is already set to allow expanding this feature if one would continue developing this project.

As discussed in Section 6.3, there could potentially be a problem if the driver chooses to use the mobile application to signal that they are late while they are driving. A possible solution to that problem is to monitor the speed that the mobile, and by extension the driver, is moving in and turn off all possible interactions with the application while the mobile is moving. This could potentially stop the driver from interacting with the mobile while driving and removing that risk from their work environment.

Another significant feature discussed has been to save a driver's position history for analytical use. This feature was deemed to be implemented at a later time because the feature would only be useful for an already working system. With this data, the company could possibly receive more support in planning the work shifts and creating a more effective and well informed schedule for the drivers.

The company has a separate planning system for planning the work shifts for each

driver and the different assignments for each day. A possibility that the stakeholders have shown interest in, is replacing their planning system by adding additional features in the system developed in this project. A system that combines the developed position tracking and the planning into one product in an effort to keep all information in one place. This could also result in, that instead of receiving the schedule through a separate communication tool, the schedule for each work shift could be received through the application, and the communication could be kept in the same place as well.

To keep the communication in the same place, a communication tool could be developed inside the software system. Being able to send messages through the system, eliminates the need for the other messaging platform WhatsApp. This came into the discussion since the stakeholder stated that they currently use WhatsApp for many aspects of their work. This could further an effort to combine all the stakeholder's software systems into one.

# 8

## Conclusion

Overall, the project was a success in terms of fulfilling the requirements from the stakeholder at a smaller scale. The developed system completed the minimum viable product and implemented some additional features like notifications for being late and finding the closest driver to a given location.

For future reference, the system has room for improvement with some necessary aspects such as adapt to GDPR-rules and optimize the database usage. As mentioned in Section 7.2 the stakeholder has an interest and end goal to have a product that could replace all the current solutions they use. During development, extension possibilities were kept in mind which makes this system a good base for reaching the long term goal of further improving the system to include all of the stakeholder's system functionalities.

It is unclear whether the stakeholder will use the system, since it is unoptimized at this moment and does too many reads and writes from the database. This makes primarily the web application slow and cost inefficient when reaching a larger number of active drivers. Its performance needs to be improved in order to host as many drivers as the stakeholder might need.

# Bibliography

- [1] Vehicletracking. (Sep. 19, 2016). “Major benefits of using real time vehicle tracking system in business,” [Online]. Available: <https://www.vehicletracking.qa/blog/real-time-vehicle-tracking-system-benefits-businesses/>. (Accessed on: 2021-05-28).
- [2] V. A. Doctor. (). “How gps tracking can improve business efficiency,” [Online]. Available: [https://www.streetdirectory.com/travel\\_guide/52315/car\\_parts/why\\_gps\\_navigators\\_are\\_necessary\\_for\\_todays\\_taxis.html](https://www.streetdirectory.com/travel_guide/52315/car_parts/why_gps_navigators_are_necessary_for_todays_taxis.html). (Accessed on: 2021-03-20).
- [3] B. Wolford. (). “What is gdpr, the eu’s new data protection law?” [Online]. Available: <https://gdpr.eu/what-is-gdpr/>. (Accessed on: 2021-05-24).
- [4] S. Ranger. (Dec. 13, 2018). “What is cloud computing? everything you need to know about the cloud explained,” [Online]. Available: <https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/>. (Accessed on: 2021-02-20).
- [5] D. Stevensson. (Sep. 25, 2018). “What is firebase? the complete story, abridged,” [Online]. Available: <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>. (Accessed on: 2021-02-20).
- [6] Google Developers. (May 3, 2021). “Cloud firestore,” [Online]. Available: <https://firebase.google.com/docs/firestore>. (Accessed on: 2021-05-10).
- [7] I. Vyas. (Jan. 18, 2021). “Top 5 advantages of using firebase for mobile app development,” [Online]. Available: <https://citrusbug.com/article/5-amazing-benefits-of-using-firebase-for-mobile-app-development>. (Accessed on: 2021-05-25).
- [8] Facebook Inc. (2021). “Tutorial: Intro to react,” [Online]. Available: <https://reactjs.org/tutorial/tutorial.html>. (Accessed on: 2021-03-12).
- [9] Brainhub. (Mar. 12, 2021). “Top 6 reasons to choose react,” [Online]. Available: <https://brainhub.eu/library/reasons-to-choose-react/>. (Accessed on: 2021-05-25).
- [10] A. Pattakos. (Jan. 25, 2021). “Angular vs react vs vue 2021,” [Online]. Available: <https://athemes.com/guides/angular-vs-react-vs-vue/>. (Accessed on: 2021-05-25).
- [11] A. Duncan. (Feb. 11, 2021). “Next.js; what is it and why do we use it?” [Online]. Available: <https://www.clock.co.uk/insight/next-js-what-is-it-and-why-do-we-use-it>. (Accessed on: 2021-02-25).

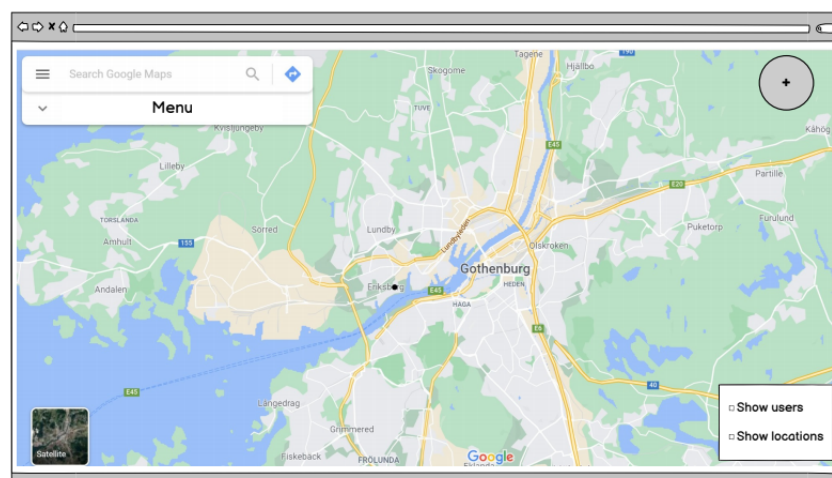
- [12] D. Bulatovych. (May 7, 2021). “Best mapping solution for logistics: Mapbox versus google maps,” Yalantis, [Online]. Available: <https://yalantis.com/blog/mapbox-maps-ready-mobile-apps/>. (Accessed on: 2021-05-17).
- [13] Concise Software. (Aug. 26, 2019). “What is flutter? here is everything you should know,” [Online]. Available: <https://medium.com/@concisesoftware/what-is-flutter-here-is-everything-you-should-know-faed3836253f>. (Accessed on: 2021-03-06).
- [14] Wikipedia. (Apr. 11, 2021). “Dart (programming language),” [Online]. Available: [https://en.wikipedia.org/wiki/Dart\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Dart_(programming_language)). (Accessed on: 2021-03-06).
- [15] —, (Mar. 11, 2021). “Machine code,” [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Machine\\_code&oldid=1022392860](https://en.wikipedia.org/w/index.php?title=Machine_code&oldid=1022392860). (Accessed on: 2021-02-15).
- [16] D. W. Bartosz Skuza Agnieszka Mroczkowska. (May 10, 2021). “Flutter vs. react native – what to choose in 2021?” [Online]. Available: <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021>. (Accessed on: 2021-05-25).
- [17] Wikipedia. (Dec. 22, 2020). “Model-view-controller,” Wikipedia, [Online]. Available: <https://en.wikipedia.org/wiki/Model-view-controller>. (Accessed on: 2021-03-15).
- [18] A. Tyagi. (Aug. 1, 2019). “What is mah in battery (milliamp hours), how much you need,” [Online]. Available: <https://www.techworm.net/2018/09/mah-battery-milliamp-hours>. (Accessed on: 2021-05-31).

# A

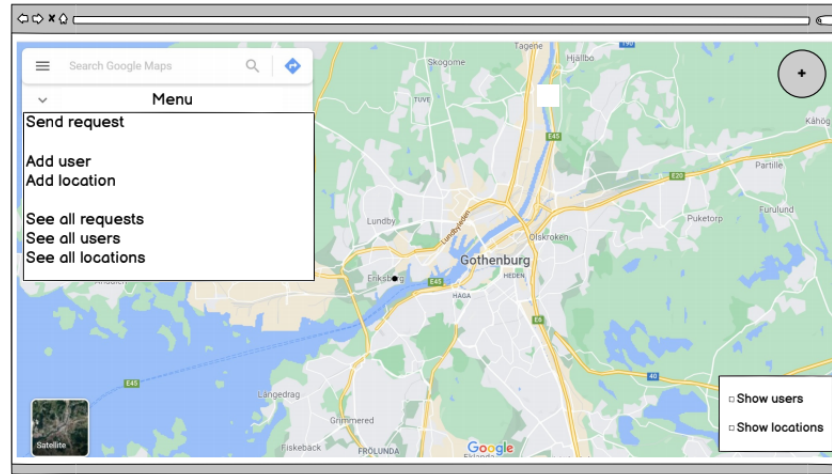
## Appendix 1: Design suggestions

A design suggestion for a login page. The page is enclosed in a browser window frame. At the top, there is a search bar with a magnifying glass icon. Below the search bar, there is a large rectangular box containing the following elements: a label 'Group Name' followed by a text input field, a label 'Username' followed by a text input field, and a label 'Password' followed by a text input field. Below this box, centered on the page, is a large rectangular button labeled 'Login'.

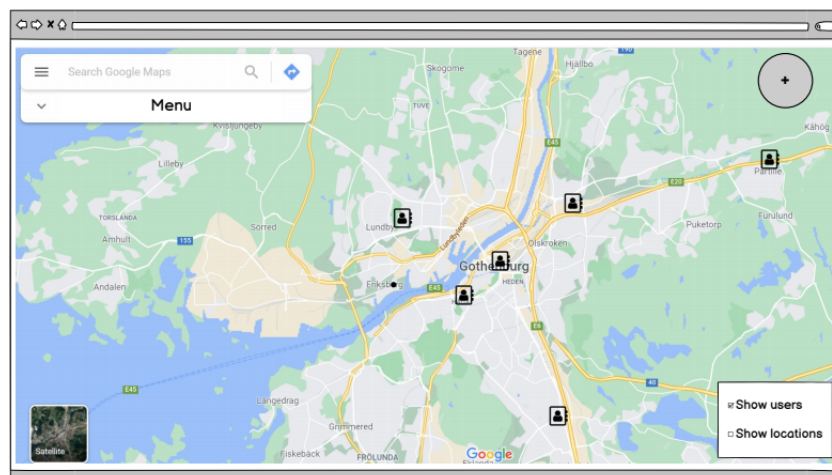
**Figure A.1:** A design suggestion for the login page for the web application from the stakeholder



**Figure A.2:** A design suggestion for the main view of the web application from the stakeholder

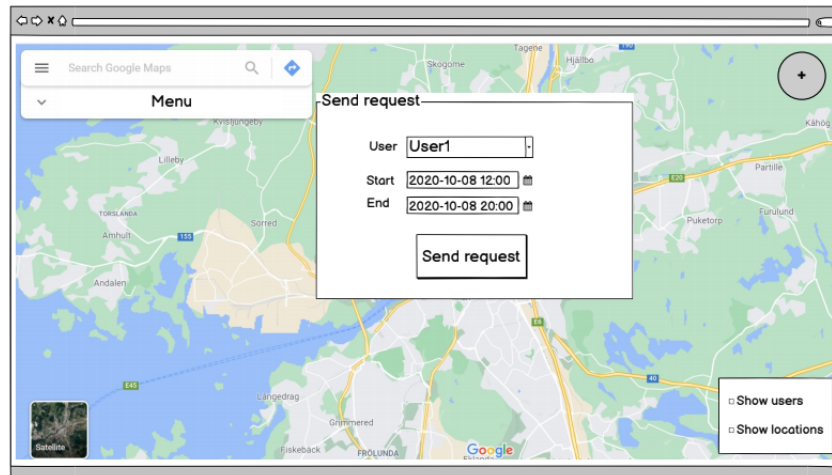


**Figure A.3:** A design suggestion for the menu in the web application from the stakeholder

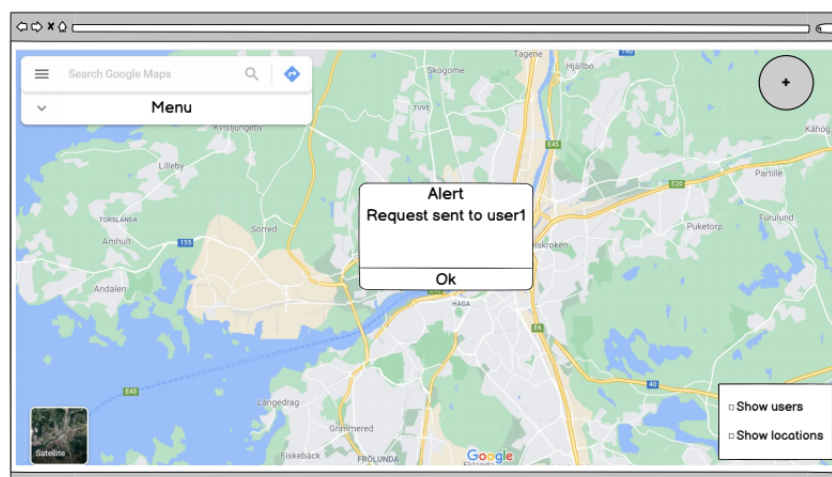


**Figure A.4:** A design suggestion for how drivers are to be displayed on the map in the web application from the stakeholder

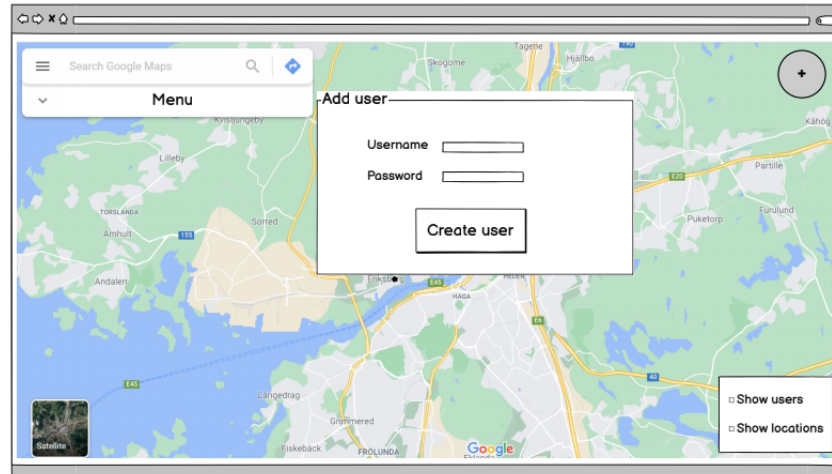




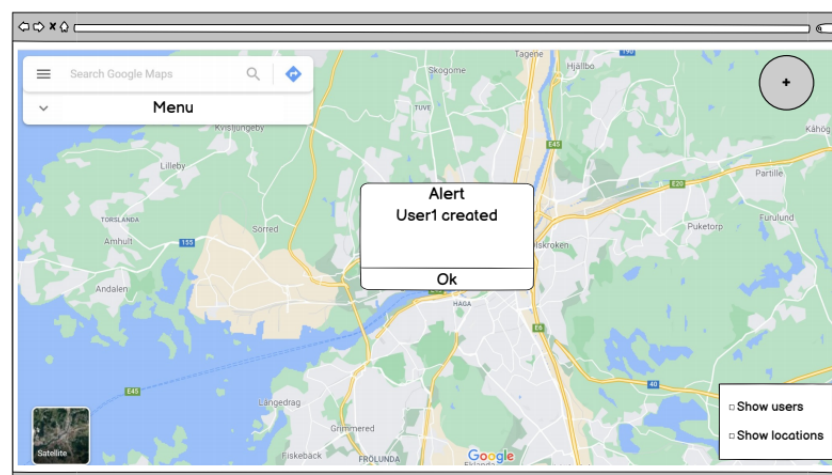
**Figure A.5:** A design suggestion for how to send a request to a user in the web application from the stakeholder



**Figure A.6:** A design suggestion for a confirmation for a successfully sent request in the web application from the stakeholder



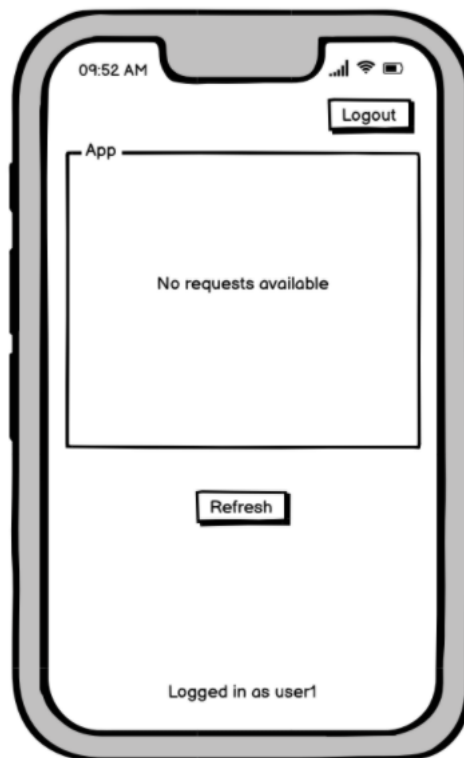
**Figure A.7:** A design suggestion for adding a user in the web application from the stakeholder



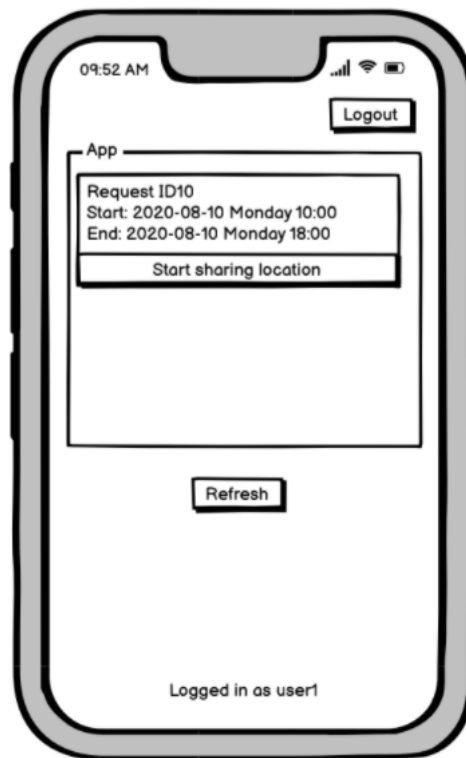
**Figure A.8:** A design suggestion for a confirmation for a successfully added user the web application from the stakeholder



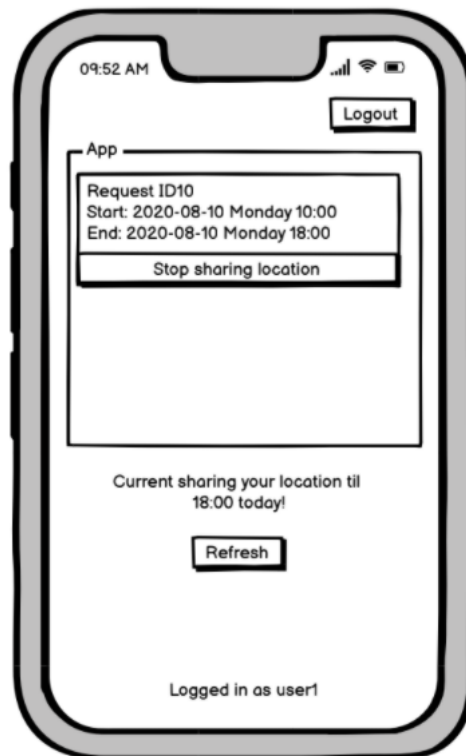
**Figure A.9:** A design suggestion for the login page for the mobile application from the stakeholder



**Figure A.10:** A design suggestion for the request view with no active requests in the mobile application from the stakeholder



**Figure A.11:** A design suggestion for the request item in the mobile application from the stakeholder



**Figure A.12:** A design suggestion for the request item while sharing location in the mobile application from the stakeholder