Final Year Project:

Clever Carpooling

Functional Specification

Authors:

Shaun Carey *16450454*

Nigel Guven *14493422*

Date of Completion:

20th November 2019

Supervisor:

Ray Walshe

Table of Contents

1. Introduction

1.1. Overview

This application is being developed for Android devices and perhaps iOS devices as well, to cater for 90% of the handheld mobile operating systems market. The specific area that we are focusing on involves navigation, peer-to-peer ridesharing, and ultimately our goal is to create a new and innovative method of transport options.

The application is called Clever Carpooling, and we plan to involve the use of the OpenStreetMap API. Users can sign up as either a driver, passenger, or both. As a very brief example, if a driving user is going from point A to point B tomorrow morning at 9am, they can post that fact. Not as a social-media-styled post, per se, but a searchable 'entry' into the application. They will also be able to post the route that they are taking. Any passenger user who lives along or close to that route, and whose destination is also point B at around 9am, can search for these parameters and request a lift from the driving user.

We expect main users to be commuters for obvious reasons, but the app will be usable by the majority, if not all, of the population, for one-time lifts and such. Ultimately, this application aims to invigorate the industry by offering a non-monetary alternative to Uber, where anyone can sign up to offer lifts, provided they pass security measures. Clever Carpooling has many applications to our society as a whole, including helping attack the climate problem, and making student/commuter life easier, to name but a few.

The application is designed to be as user-friendly as possible. A messaging system will be included so that users may contact one another inside of the application without having to leave it for a third-party communications system. There is also to be incorporated, a prediction system which will examine the user data and provide interesting notifications to the user such as telling them how popular their route is with drivers, what would be the best route to take and also weather updates. This would provide a useful function for both drivers and passengers.

1.2. Business Context

The potential for commercial usage of Clever Carpooling is immense. Influenced by mobile transport applications like Uber, Lyft and MyTaxi/FreeNow, which all make considerable profit from taking a percentage of their drivers' income, our application instead is free to use, and it is up to the driver to charge a fee to their passengers. The use of tailored advertisements could be implemented into the system to create some revenue.

For each region, be it Ireland and Britain, the United States or Japan, no single passenger transport company has dominance over the market which makes our application quite useful. The goal is to provide seamless, rapid service commuting for a population which will use up empty space in cars and remove more vehicles off the road.

1.3. Glossary of Important Terms

Android :

Android is a mobile operating system which was developed by Google and based upon the Linux Kernel. It is specifically designed for handheld devices like smartphones, smart watches and tablets.

iOS:

iOS is the mobile operating system created and developed by Apple, and the main rival of the Android OS. It presently powers many of the company's mobile devices, including the iPhone, a smartphone, and the iPad, a tablet.

Xamarin:

Xamarin is a cross platform development platform for building Android and iOS applications under a .NET and C# framework.

Android Studio :

Android Studio is the official IDE for the Android operating system. It comes included with an interface design package and the Android SDK.

Visual Studio:

Visual Studio, also known as Microsoft Visual Studio and VS, is an integrated development environment for Microsoft Windows. It is a tool for writing computer programs, websites, web apps, and web services. We will hopefully be using this for the iOS side of the project.

Java:

Java is a high-level programming language which is object-oriented and follows the 'write once, run anywhere' memorandum which allows it to run on any platform which supports Java without needing recompilation.

Firebase:

Firebase is a mobile and web application development platform which provides a database for applications which is located on the Firebase cloud. Objects are stored as JSON files.

Data Mining:

Data mining is the process of uncovering patterns and finding anomalies and relationships in large datasets that can be used to make predictions about future trends.

OpenStreetMap:

OpenStreetMap is the free wiki world map, an open volunteer-driven initiative to collaboratively create a map of the world and release the map data under a free and open license.

Functional Specification

API:

A set of functions, tools and procedures that allow the creation of applications which access features of an operating system or another service.

IM:

Stands for Instant Messenger, a type of online chat that offers real-time text transmission over the Internet. Users will have access to an in-application IM on Clever Carpooling.

UMLET:

A toolkit for developing Structured Systems Analysis Design Method or SSADM diagrams, charts, tables etc.

Polylines:

Polylines in computer graphically created lines which consist of many connected lines. Poly means many in Greek. They are found on the OSM map as route directions where each line is a street interconnected from a source and destination

Gitlab:

Gitlab is a development tool which provides hosting of repositories, CI/CD and is essentially ran entirely on a web platform without the need of a project to be stored on a local machine

Swift:

Swift is a high-level programming language developed by Apple Inc. which is primarily used for creating applications for iOS, macOS and is similar to Python in its typeset but it is primarily known for being faster in execution time.

ML Kit:

ML Kit provides Android and iOS applications with the ability to recognize text, faces, barcodes and object detection using Google machine learning and Android Neural Network API's.

JSoup:

JSoup is a package in Java which allows for the parsing of XML and HTML documents, it is useful for web scraping and data mining from websites where it can process tags to extract specific information

SwiftSoup:

SwiftSoup is the counterpart to Java's JSoup or Python's BeautifulSoup which allows web scraping functionalities in the Swift language. A URL is requested and given a variable name, the various functions of the SwiftSoup library can perform extraction of information from the tags littered throughout a webpage.

2. General Description

2.1 Product and System Functions

The following is a list of the functions of our application split up into the relevant sections. Each function will be elaborated in more detail in the functional requirements section.

2.1.1 User functionality

On first open:
- Accept Privacy/Plain Statement Agreement
- Sign Up
- Get verified as a driver

Normal Use:
- Login
- Post a trip (as driver)
- Search & Request a lift (as passenger)
- Instant Messaging (IM) System
- Rate driver/passenger

2.1.2 System functionality

- Firebase user storage
- Continuous use of OSM.
- Image Recognition
- Prediction System
- Dynamic Web Scraping of API data

2.2 User Characteristics and Objectives

The application will be available to Android users (70% of the handheld market), and iOS users (20% of the handheld market), who also have internet access. The target audience is pretty much any Android/iOS user who lack another way to get to their destination and require a lift, although we expect the vast majority of users to be commuters and students.

The user will only be expected to have basic knowledge in the use of touchscreen applications (which should be a given if they own a touchscreen handheld device), as we hope to design it with the most user-friendly interface possible. In the rare case that a user does not have much expertise on navigating touch-screen apps, it is hoped that this basic interface will accommodate its users by not being too complex, but while still being perfectly fit for purpose. We expect that users who are familiar with similar applications such as FreeNow or Uber should have no problem with this app.

In terms of UI interaction, our desired characteristics include relaying as much necessary information to the user as possible. This will be done via the predictor module and we will design it so that it displays relevant information in a clear concise format, while avoiding the problem of overloading information.

2.3 Operational Scenarios

First time opening:

The app will welcome the user, briefly tell them what to expect from the app, and then ask them to sign up. They will then be asked to accept a privacy agreement as is standard in most applications, especially those which stores personal information from users and is heavily location-based. This privacy agreement contains information which allows the application to receive a location and also enforce the user to connect to the internet in order for the application to function. Each user will be given access to every user function from this point onwards.

Deletion and Reinstallation:

If a user decides to delete or remove the application from their phone, their data will be retained on the application database unless otherwise requested by GDPR regulations. They will be emailed by a customer service contact with a user feedback form to fill out if they wish to do so.

If they decide to reinstall and log back into the application, a splash screen will display which will display their name and welcome them back to the application.

2.4 Constraints

The following is a list of constraints which we thought of whilst coming up with the design of our project, which might hinder us further down the line. Other than the expected time constraints such as the project deadline, we've also thought of a few more:

- *Ethical plausibility:* As we are dealing with and storing user's personal information, as well as the fact that they will be sharing their location, ethical forms will have to be completed before the development of this project.

- *User feedback:* Unless a few willing participants are willing to take part in a feedback form, we probably won't be able to generate much feedback from users who would be willing to use an app such as this on a regular basis. The application is built around users and can become 'dead' if there are no daily logins.

- *Web Scraping Legality:* Although we are confident of the legality of this project, there are issues over the fact of whether taking this information from other sites is possibly illegal. After emails for permission, and reading of some terms of services, we are confident that we can continue forward but may have to reinitiate plans to use an authentic API which may require costly fees to produce information.

- *Memory/Storage:* One of the main problems we face whether we have the storage space on our database to include every function that we wish to include. Not being able to include some of our features could also be considered another user drawback. With regards to efficiency, the image recognition module must be able to quickly read and save a photo of a driver's license.

- *Protection against criminality:* A problems that might come upon users is the presence of criminality. To circumvent this, we must implement checking blocks to protect our users. This could be in the form of checking and verifying car registrations, credit cards and driving licenses. Login information provided will be linked to social media so that the identity of a user will be known but also protected. We believe such techniques should manage to curb any malicious attempt by a criminal.

3. Functional Requirements

3.1 User Functionality

3.1.1 Login/Sign-Up

Description

For users to access they application, they must create an account within the application system. This requires the user to list the following personal information:

- Name
- E-Mail
- Mobile Number
- Password
- Address
- Date of Birth
- Profile Picture
- Driver Licence (Optional)

Users must also sign a plain language statement, which is necessary given some terms and conditions, the majority of which are GDPR related acts. Some other items relate to user security while on the app.

There are two forms of users, passengers and drivers. Passengers may only use the app to request a lift from a driver. To become a driver on the application, one must email proof of driving license and business insurance documents to clevercarpoolingsupport@gmail.com. This covers the driver if they are in an accident during a carpool. Drivers may pick up passengers and have more control over the application usage. The driver's license is mandatory for drivers but unnecessary for passengers to have. An image recognition system checks the driver license to make sure that it is legitimate. For this it reads the driver number and examines that a user is a fully licensed Irish driver.

Criticality

This step is essential to the application and the user as it acts as a gateway into the main functionality of the system. Without it, there is no secure way of establishing client-client communications

Technical Issues

To allow this function to proceed, users must have listed each the relevant information and any lacking details may render user access inoperable.

Dependencies with other requirements

This module has dependencies with the image recognition system for reading drivers licenses. Also, data must be uploaded to a Firebase database which holds all user information. The Instant Messaging module requires to users to communicate with each other and therefore these are mutually dependent on one another.

3.1.2 Rideshare

Description

Users who are looking for a lift should request a rideshare. A rideshare is essentially a notification that is listed on the OSM map which lists the requirements of users. These requirements would be a location that a user needs to reach and at a certain time. This is provided as a destination coordinate and timestamp with an optional list of notes that a user may provide if they wish.

Criticality

This module is essentially the main function of the application. With a list of items that a passenger provides a driver may accept or decline a passenger's rideshare request. Displayable on the map for all users to see, this is essential for the application to proceed as planned

Technical Issues

There are no technical issues with the design and implementation of this requirement.

Dependencies with other requirements

This has strong links to the OpenStreetMaps API which displays rideshare notifications.

3.1.3 Instant Messaging

Description

For a passenger to communicate with a driver and vice-versa, users need to have a means of communication within the application. Rather than forcing users to use third-party communications, the application will incorporate an instant messaging system for users to coordinate plans. A simple chat room for two or more parties will suffice. This will also feature sending location and VOIP call features over a solid Wi-Fi connection.

Criticality

The importance of this IM module is not critical to the overall system as users may coordinate their means of service differently. This module is meant to improve user experience within the application.

Technical Issues

This module has one major potential technical issue. Users will need to have access to Wi-Fi for this module to function properly. Without it, users will be unable to communicate or use the VOIP additions.

Privacy Issues

This module contains hazardous privacy issues. Users have control over whom they are in contact with and may block an unknown device. Users may provide their location, but this is optional and is meant for ease-of-usage for drivers to pick up passengers.

Dependencies with other requirements

The instant messaging system communicates mainly as an attachment to the maps API.

3.2 System Functionality

3.2.1 OpenStreetMap API

Description

The OpenStreetMap API will form the centrepiece of navigation in the application. Entirely a volunteer-based project, the application has support for the likes of GPS, Route planning and geocoding. The system will require users to reveal their location on the map as well have features to control map elements such as changing from satellite to navigation layers and to search for a location on the map. The API should be able to map a route from a driver to a passenger and should also list possible passengers in the area that a driver may pick up.

Criticality

The OSM is essential for the navigational systems to work on the application. Without a properly functioning map or users not allowing their location to be revealed, the application will not be able to route a destination

Technical Issues

There are some issues regarding location of the user. If a user disables their location, then the system must request that they do reveal their location or exit the map.

Dependencies with other requirements

Given that the OSM is already dependent on the location of the phone system, there are also requirements to the prediction system which must derive from OSM data, the number of users who travel between different cities, towns etc.

3.2.2 Image Recognition

Description

Image Recognition is a module appended to the Login/Sign Up system which reads in a user's Irish driving licence. The user must submit their licence under the camera which is then read by the module and if found to be illegitimate or erroneous, presents the user the option to retake the photo or decline the request to become a driver.

Criticality

This is only an issue for users who wish to become drivers on the application.

Technical Issues

The main problem that might arise from this is that a user's camera is not of enough quality or lacking the feature altogether. This would prevent users from becoming fully fledged drivers on the application. A workaround would require the user to submit their license picture to a support email whereby they may be added to the driver system by an administrator manually. This is time-consuming but given that nowadays, most phones come with cameras installed.

Dependencies with other requirements

This functional requirement has several dependencies with the underlying hardware of the phone. It has semi-dependent relations with the rideshare portal and the OpenStreetMap API. This nomination is selected because the system will need to undergo this dependency to allow users to become drivers.

3.2.3 Prediction System

Description

The prediction system is a data mining tool embedded into the application which retrieves interesting patterns and displays them to the user. The specific data could be telling the user how many cars travel to a specific location which gives the user an idea of their route plan. Also, users will be told which routes are the most popular given that the user selects a route. The data is retrieved from the database of the application located at firebase.

Criticality

This function is not critical on the overall system. It does require data from other module and therefore contains dependencies, but it is not critical for the application to run.

Technical Issues

There are issues with establishing a solid connection to firebase. If the module cannot connect to firebase, then its data mining function will operate erroneously. We plan to save data to the user's phone in this event, and such data will be updated once a connection is re-established.

Dependencies with other requirements

This prediction module is dependent on retrieving data from the firebase and is therefore dependent on the rideshare portal. The rideshare portal stores its data on firebase which the prediction system must analyse and return to the user in a human-readable format.

3.2.4 Firebase DevOps Platform

Description

The Firebase platform provides two major items in the Clever Carpooling application, a real-time database which operates on the back-end of our application framework and provides ML Kit which essentially provides machine learning for mobile applications. The database will store user information and save routes to a separate table which in total can be analysed using the prediction module to provide interesting information back to a user. This is further made possible by the strength of Firebase in being highly scalable as a storage service.

Secondly ML Kit will be used to examine driver licences and bank cards from pictures taken from a single user's phone camera. ML Kit has powerful text recognition API features which will be beneficial in reading information from a picture.

Criticality

The Firebase system is extremely important in the application utilisation particularly at the back-end of the system. Without ML Kit, drivers would not be able to register as the picture recognition system must decide if a single licence is valid or not.

Technical Issues

Technical issues may arise where the user is unable to take a picture or is entirely unable based on their mobile phone's hardware. To work around this, users will have to login with a different phone or contact the support email to register with relevant bank/driver information attached.

Dependencies with other requirements

The platform has dependencies on the overall system whereby, the prediction module analyses it and it must also collect information from both the OpenStreetMap API and Registration system.

3.2.5 Dynamic Scraping of Web Data

Description

This function is being operated in the back-end of the application. There are two scripts assembled from two different libraries, JSoup and SwiftSoup for their respective application systems. JSoup is for Android systems and SwiftSoup is for iOS systems. The web scraping involves taking an Irish driver registration number which must include the year of make, the county that this specific vehicle is registered to and a certification number.

A driver must enter this registration number upon signing up to the application. The two scraping libraries will create a URL search parameter and take data from the MotorCheck API and upon finding the vehicle as a match, will then save that vehicle to the driver. This will allow passengers to know the make and type of a car so that they may verify upon arrival on a driver. This essentially enhances the security for both drivers and passengers.

Criticality

This function collects driver registration from a car registration API and it is necessary for drivers to have their car information registered on the application for security and verification purposes so therefore it is vital to the application.

Technical Issues

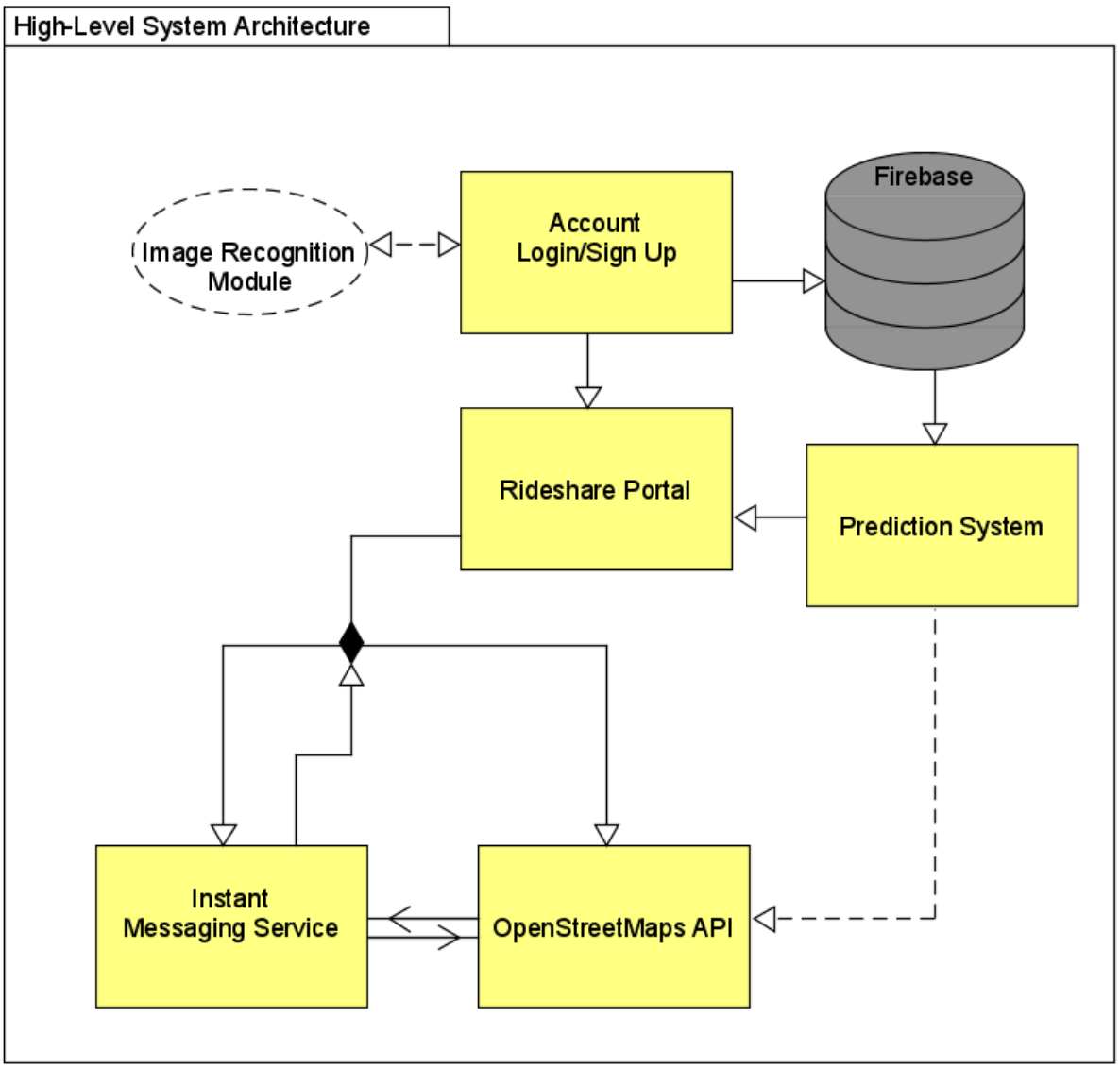
The plan to develop this dynamic web scraping system involves writing two different scripts for different systems. This is because there are two different languages for different platforms making this function platform dependent. Both of the libraries, JSoup and SwiftSoup may have different operations and optimisations which will definitively cause some differences of effect between the two applications on different operating systems.

Dependencies

This functionality requires the correct input of user information and must be correctly established with a valid URL connection to the MotorCheck API.

4. High Level System Architecture

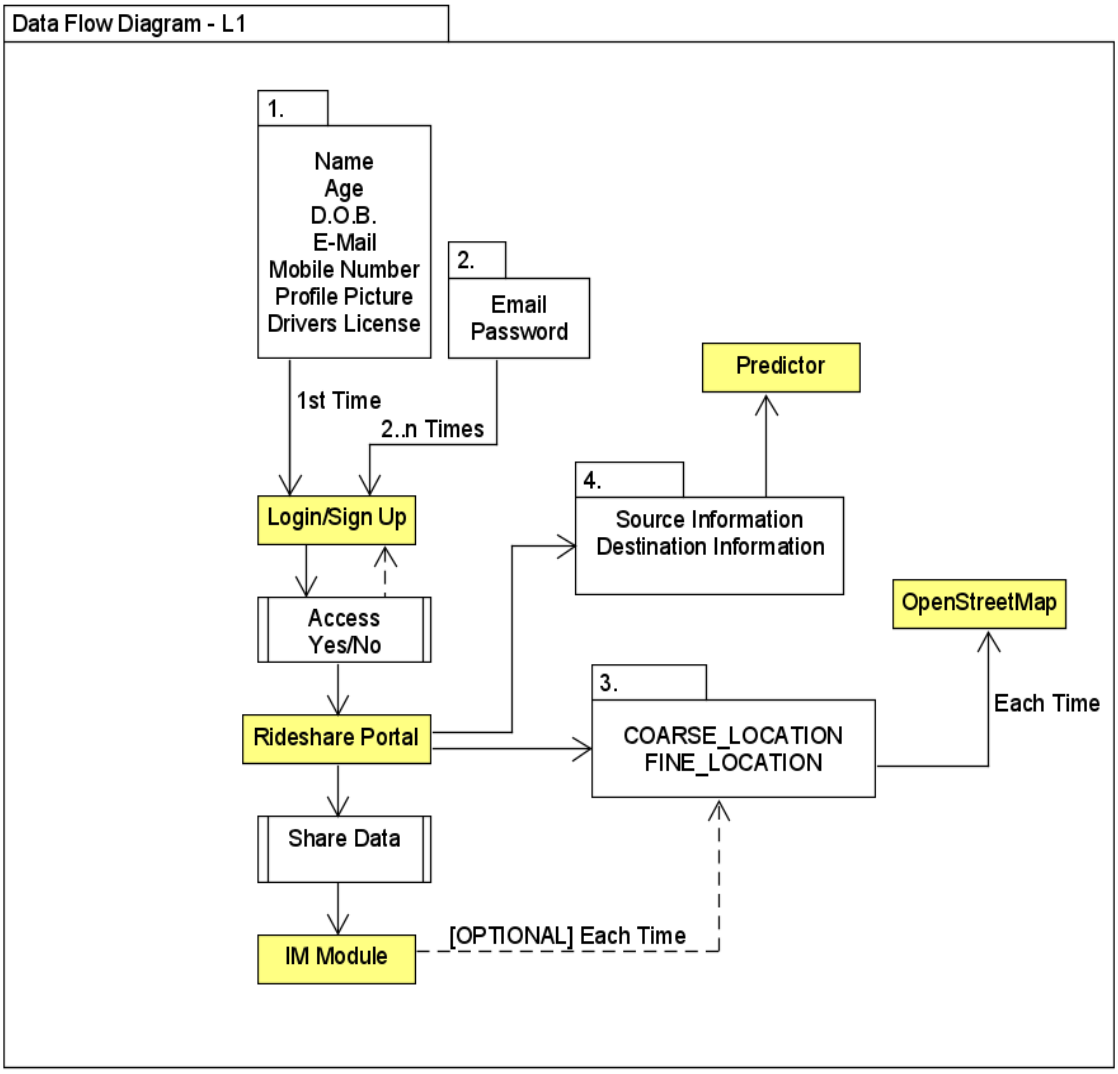
The following diagram portrays the entire architecture of the application. Starting with a login/sign up section, communications with a Firebase server and prediction module, the flow runs through the OSM mapping system (Subject to change). See Chapter 3 – Functional Requirements which describe how the functions operate over the system.



5. High Level Specification

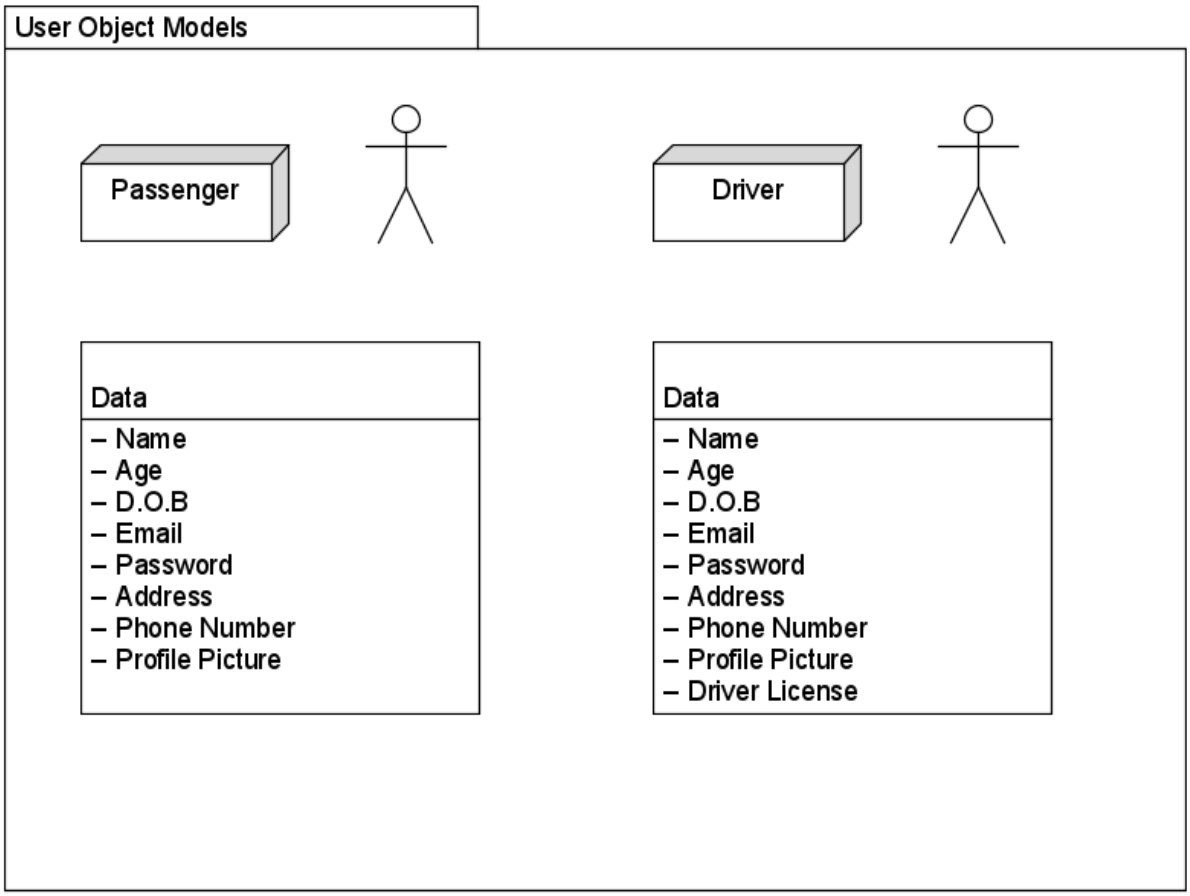
5.1 Data Flow Diagram

The diagram below describes the flow of major data items throughout the program.



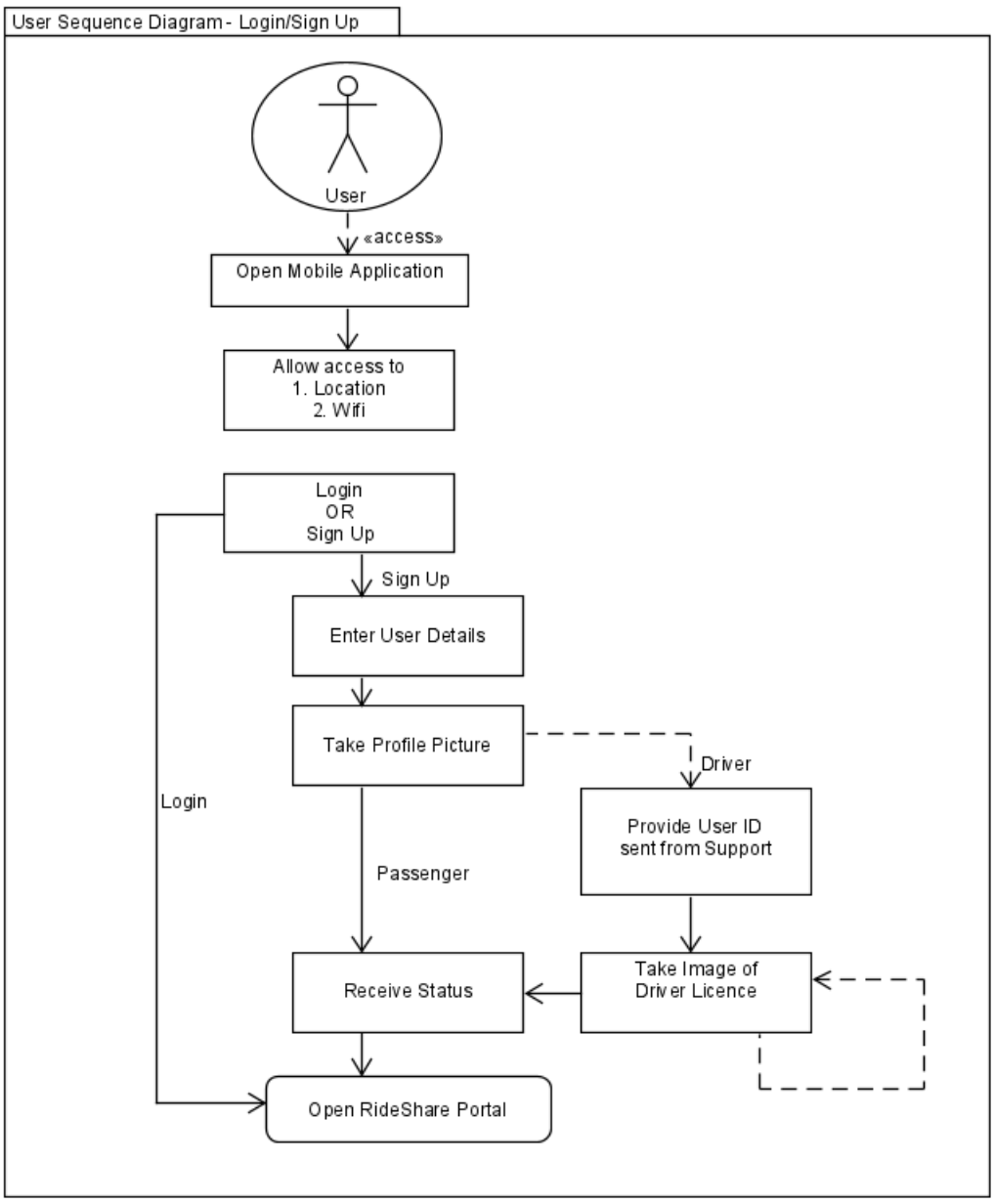
5.2 User Object Models

The following diagram contains a set of models for two different types of users, drivers and passengers and what data items they are expected to maintain in their respective classes on the application.



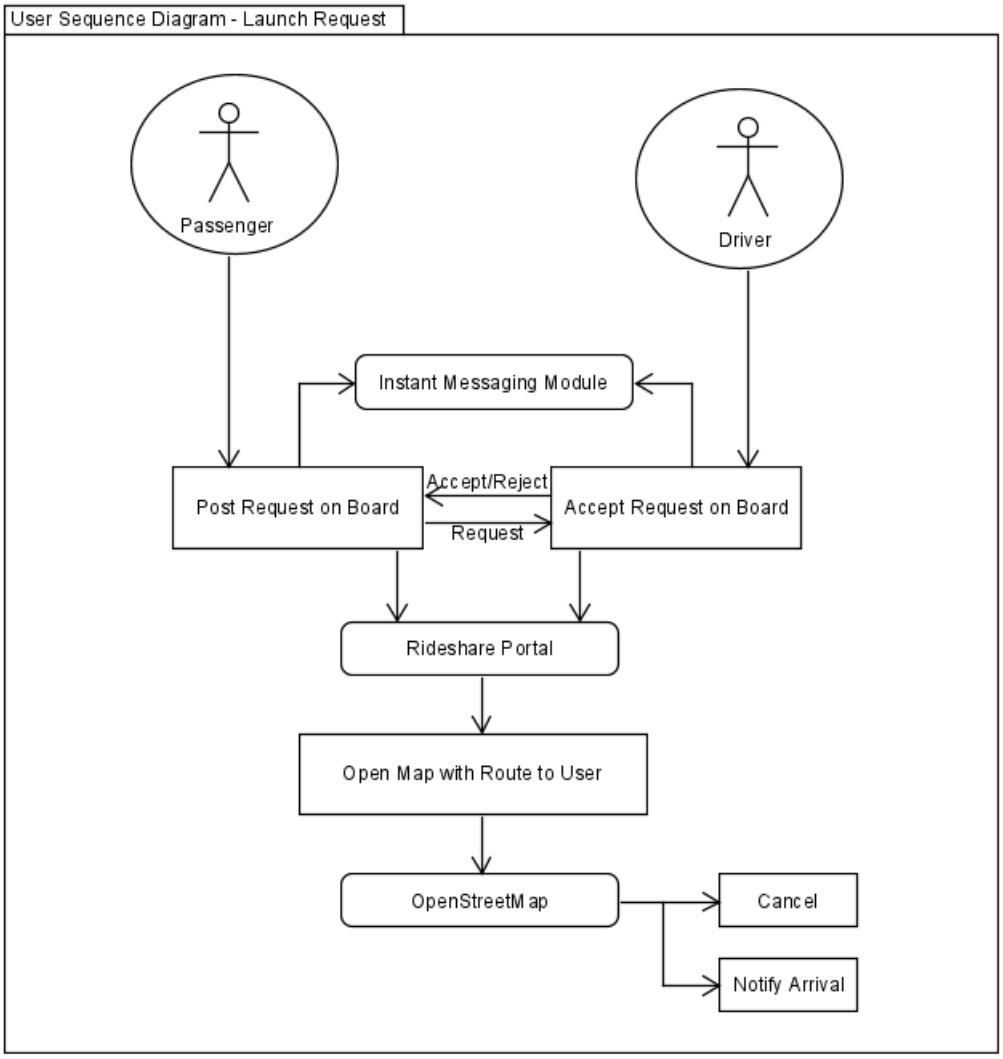
5.3 User Sequence Diagram – Login/ Sign Up

The following diagram describes the sequence of user controls over the login/sign up section of the application.



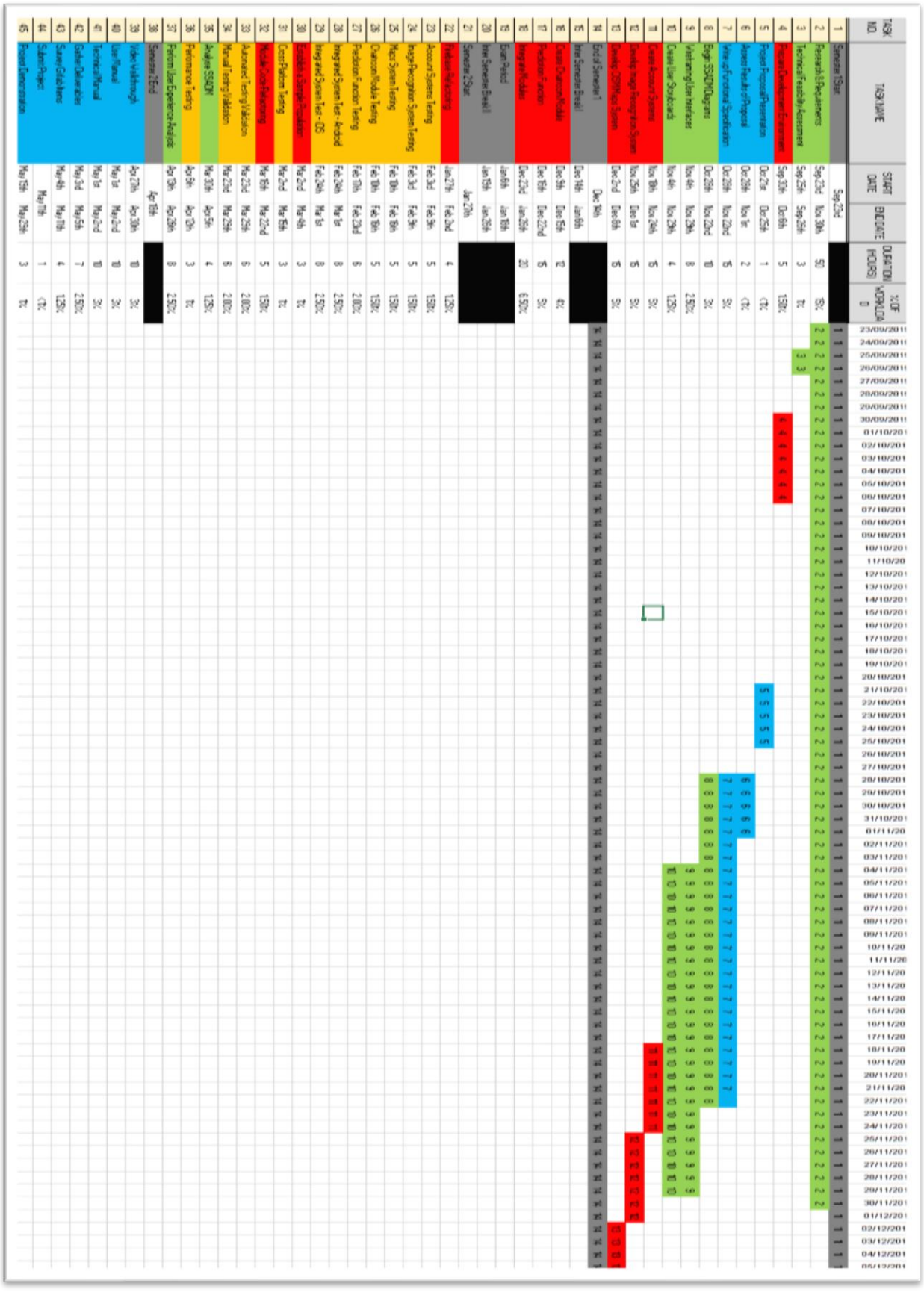
5.3 User Sequence Diagram – Launch Request

The following diagram describes the set of user controls over the sequence of setting up a rideshare request, acknowledgement and setting up of OpenStreetMap polyline routes.

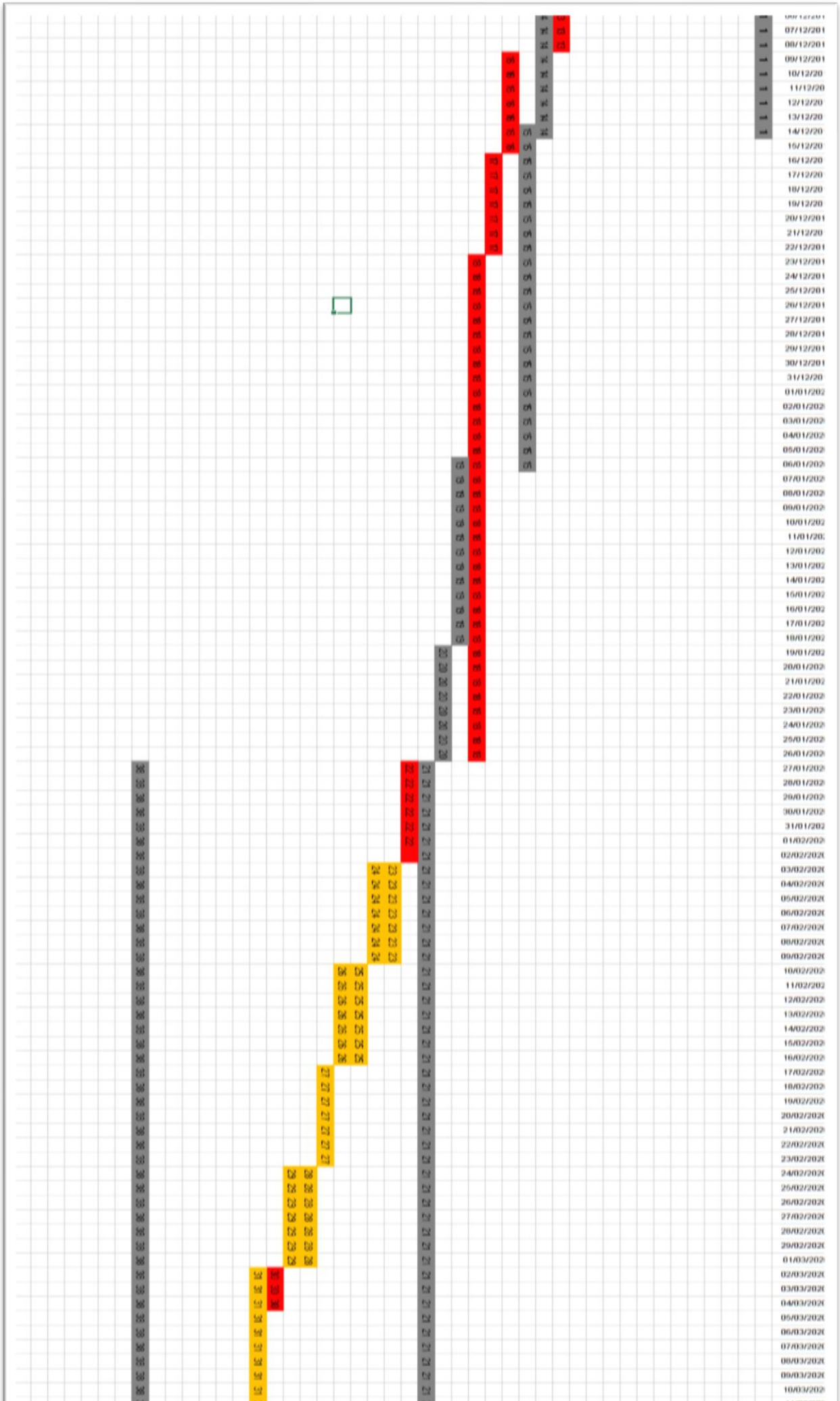


6. Project Schedule

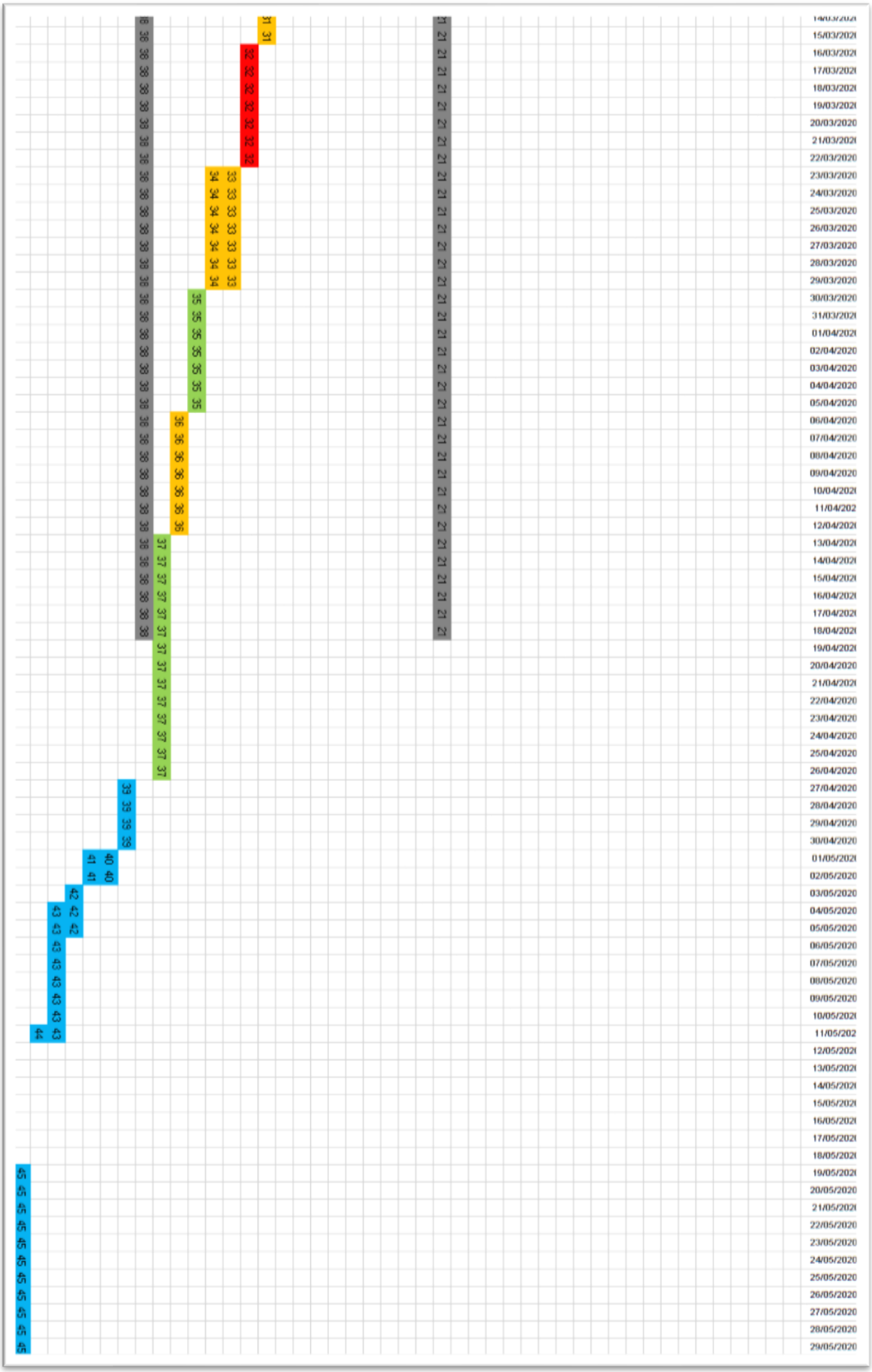
The following chart in a Gantt style workflow shows the tasks and their respective work days over the entire system lifecycle. Items are color-coded as the following: blue for deliverables, grey for time constraints, orange for testing, green for systems analysis and red for development.



Functional Specification



Functional Specification



7. Appendices

7.1. Research, Tools and References

Java	https://www.java.com/
OpenStreetMap	https://www.openstreetmap.org/
Swift	https://developer.apple.com/swift/
Android Studio	https://developer.android.com/studio
Visual Studio	https://visualstudio.microsoft.com/
JSoup	https://jsoup.org
SwiftSoup	https://github.com/scinfu/SwiftSoup
Xamarin	https://dotnet.microsoft.com/apps/xamarin
Firebase	https://firebase.google.com/
UMLet	https://www.umlet.com/
Gitlab	https://about.gitlab.com/

7.2. Sites for Inspiration

Uber	https://www.uber.com/ie/en/
FreeNow	https://free-now.com/ie/
Lynk	https://www.lynk.ie/
MotorCheck API	https://www.motorcheck.ie/api/