Functional Specification

CA326 - 3rd Year Project

Project Title: Carpool App for DCU Students on Android

Students:

- Daragh Prizeman (19459734)
- George Eskander (19451972)

Supervisor: Darragh O’Brien

Date of Submission: 09/12/2021

Table of contents

1. Introduction

1.1 Overview

The system we will be developing is a carpool Android app for DCU students. This app will allow students who drive regularly to and from DCU campuses to sign up as drivers, and help fellow students with a lift to and from campus. On the other hand, students who are in need of cheaper transport to and from campus can sign up as passengers, and request to carpool with a driver to and from DCU. Drivers can then accept or decline a passenger's request. If the request is accepted, then the system will get the shortest route from the driver's starting location to the destination with a stop at the passenger's location. The system will get this route by working with an external API such as Google Maps API then the system would optimise it based on the passengers and drivers constraints.

We believe there is a need for this system as students need ways to manage their costs, to find their best route to DCU's campuses taking time and cost into account, and students look for ways to make their commute more enjoyable which could be done by socialising with other students as drivers/passengers.

On Android, the functions of the system would include DCU students being able to take a role as a passenger or a driver, input their start location and destination either to or from DCU's campuses within Ireland and their constraints for the ride, the system would then show the most optimal drivers to passengers which they pick, the driver can then accept/reject their request.

For the system, passengers and drivers may need to sign up an account and for driver/passenger roles then give access to their location in order for the system to function as intended.

1.2 Business Context

The development of this system is not sponsored by any business organization.

The system may be deployed on a cloud platform such as one like Heroku in order for students to use it online. Docker would be used to ease deployment for the development of the system.

1.3 Glossary

API - stands for Application Programming Interface, they are used for communication between different applications.

React Native - UI framework used for creating cross-platform mobile applications without having to write lots of "native" code, it is similar to React but with native components instead.

Django - backend framework for creating web and mobile applications through Python.

Encrypted - past tense of 'encrypt' meaning to convert data into code to protect information from unauthorized access.

Optimal route - This refers to the best possible route from the starting point to the destination.

GANTT chart - This is a type of bar chart, named after Henry Gantt, which illustrates the schedule of a project.

Firebase - cloud messaging service also used for creating web and mobile applications.

Heroku - cloud platform for deploying web and mobile applications.

Back4App - cloud service that can be used for backend.

Docker - platform for packaging applications into containers.

2. General Description

2.1 Product / System Functions

The key functions needed for this carpool app are as follows:

- Register
- Login
- Set user role
- Organise trip
- Input trip information
- Search for nearby drivers
- Accept or decline carpool request
- Calculate optimal route
- Update account details
- Delete account

2.2 User Characteristics and Objectives

The users of this system are DCU students who have Android devices, and are looking to carpool to and from campus with other students. Users are expected to be familiar with the Android operating system, and Android apps in general since Android was first released in 2008 and the number of Android devices has grown exponentially in recent years.

From the user's perspective, the primary objective for the system is to find other students to carpool to and from DCU campus with.

There are two types of users of this app; Drivers and Passengers.

From the user's perspective, the system is required to:

- Allow them to create an account.
- Be able to choose between passenger and driver roles.
- Allow passengers to input their starting location, destination, time of departure/arrival, and any other passengers they are travelling with.
- Allow drivers to input their starting location, destination, time of departure, available seats, and any other constraints they may have.
- Search for suitable drivers based on their route information and constraints.
- Allow passengers to request to carpool with a driver.
- Allow drivers to accept/decline a carpool request,
- Display the most optimal route for the journey based on distance and constraints
- Display driver contact information to the passenger.
- Display passenger contact information to the driver.

The users wishlist may include:

- Passengers being able to save their trip with a driver after a trip.
- Drivers may like to pick up multiple students (they must specify this if so).
- Be able to alter a saved trip.
- Be able to see the nearest carpools on a map.
- Be able to pick a nearby location close to a driver's destination (drivers may need to set what distance they consider nearby to their destination).
- Passengers and drivers may like to contact each other by phone.
- Be able to set up a trip in the future.

The above wishlist are features that are not necessary for the core functionality of the app, but are features that we will work on to enhance the user's experience if we have the time.

2.3 Operational Scenarios

Below is a list of operational scenarios that illustrate, from the user's perspective, what will be experienced when utilizing the system under various situations.

1. User enters the app

UI will load, the user will get a pop up asking if the app can access their location. If access is given, it will show the register/login component otherwise they will be asked to turn it on again in order for the app to function.

2. User registers

A user clicks the register button, fills in their details, and they are added to the database as a user if those details don't exist.

3. User logs in

A user can log in to their account once they are registered.

4. User selects a role

When a user logs in for the first time, two options are available on screen. A user can either select to be a passenger or a driver. The user can change their role at any time in the settings menu.

5. User with Driver role organises trip

The Driver will input whether or not this is a new trip being created and then whether or not this trip is organised now or later in the future. If it is a saved trip, they can skip all the steps below or choose to change its details.

If the trip is being organised now, they will select if they are going to one of DCUs campuses or leaving one of them. If they are leaving campus, they will select the campus as their starting location. Otherwise, if they are going to one of DCUs campuses their current location will be selected as the start location.

If the trip is being organised for the future, they will input the time they will begin the journey, then choose whether or not this is from DCU then manually select starting location and the destination.

The Driver can save the trip at any time.

6. User with Passenger role enters trip information

The Passenger will input whether or not this is a new trip (this could be a trip in the future agreed with a Driver), if it is a new trip, they will select whether or not they are leaving campus, if they are leaving campus they will select the campus and their destination, otherwise their current location will be selected and they will pick a campus.

The Passenger would then try to find a driver in scenario 7.

7. Passenger searches for nearby driver

The Passenger can search for nearby drivers after inputting trip information in scenario 6. A list of suitable nearby drivers will be displayed on screen to the Passenger, based on their trip information and constraints. The Passenger can then select a driver from the list and request to carpool with them.

8. Driver accepts/declines a carpool request from passenger.

After a passenger requests to carpool with a particular driver, a notification is sent to the driver. The driver has the option to either accept or decline the carpool request. If the driver accepts the carpool request. The passenger’s contact information is displayed, and the driver’s route is updated to include the passenger's stop. If the driver declines the carpool request, a notification is sent to the passenger to inform them and encourage them to find another driver.

9. Find the optimal route to passenger

The optimal route is calculated based on either the shortest route or both the shortest route and the constraints involved. Once a driver accepts a request in scenario 8, the app calculates the optimal route from the driver's starting location to the driver's destination with a stop along the route for the passenger.

10. Update account or role details

The user can update parts of their account details such as their phone number, password and also update details for roles such as car details for drivers in the settings menu. Users cannot change their email address for their account.

11. User deletes their account

When a user presses the "delete account" button in the settings screen, a warning message will be displayed in case they pressed it by accident. If they confirm their desire to delete the account, their account is erased from the database, and the user is redirected to the homescreen.

2.4 Constraints

Below is a list of constraints placed upon the design team:

- The app should be compatible with devices running Android, including mobile phones and tablets.
- The team has no prior experience with some technologies such as React Native for Android and building mobile applications, time will be required to learn these.
- This project is constrained by time as the final deliverables are to be submitted by the 4th of March. Refer to our preliminary schedule in section 6.

3. Functional Requirements

These are the functional requirements listed which describe what the system needs in order for the carpool app to work.

3.0. Set location permissions

Description

When a user enters the app for the first time, the app must ask for permission to access the user's location data from the user. A user has the option to either accept or deny permission to their location data. A user should be able to revoke permission to location data at any time via the settings menu, however, this will prevent them from using the app until it is turned back on.

Criticality

This is critical, the app will not be able to function as intended without location data from users.

Technical issues

We must ensure that users' location data is kept securely.

Dependencies with other requirements

N/A

3.1. Register

Description

Before a user can login, they must first register for an account. A user can register by tapping the "register" button on the homepage. This will redirect the user to a registration page, where the user will fill in their details, such as name, email, and password which are required. If the inputted email is unique and other fields are valid, the user's account should be added to the system database and the user is then successfully registered. The user should then be redirected to the login page. An error message should be displayed if the registration is unsuccessful.

Criticality

This function is necessary as the user will not be able to login to the app without first registering for an account. It is for this reason that we will be setting up the registration and logging in functionality for this app first.

Technical issues

User's data must be kept securely in our database. Users' passwords will be encrypted for this reason.

Dependencies with other requirements

Users must have accepted location permissions. See requirement 3.0.

3.2. Login

Description

After registering for an account, a user must be able to login into their account via the login page, which should be accessible to the user by tapping the "login" button on the homescreen. On the login page, the user must be able to input their email address and password, and be successfully logged into their account, provided that the email and password entered match to one in the database. An error message should be displayed if the login is unsuccessful.

Criticality

This function is necessary as the user will not be able access the app content without first logging in to their account. It is for this reason that we will be setting up the registration and logging in functionality for this app first.

Technical issues

User's data must be kept securely in our database. Users' passwords will be encrypted for this reason.

Dependencies with other requirements

Users must have accepted location permissions. A user must have first registered, in order to be able to login to their account. See requirements 3.0. and 3.1.

3.3. Set user role

Description

After logging in their account for the first time, a user should be prompted to select a role. They should have the option to either select the role of a Passenger or a Driver. Upon selecting a role, the user should then be prompted to input further details in relation to their role i.e. if a user selects Driver, they need to input the model, and colour of their car. After

successfully inputting their details, the user's role will be linked to their account in the database. A user should have the option to change their role at any time via the settings menu.

Criticality

This requirement is of high priority as users need to be able to select a role in order for the app to function as intended.

Technical issues

We must ensure that the user's role is correctly linked to their account in our database.

Dependencies with other requirements

A user must have accepted location permissions, and be currently logged in to their account in order to set their role. See requirements 3.0 and 3.2.

3.4. Driver organises a trip

Description

A Driver must be able to organise a trip by tapping on the "organise trip" button after logging in. The user should then be prompted to input the details of their trip, which includes their starting location, destination, time of departure and any other constraints they may have. After successfully inputting this information, the Driver's trip should be created and added to the database.

Criticality

This requirement is of medium priority as Drivers need to be able to organise trips, however this does not directly impact the core functionality of the overall system.

Technical issues

We must ensure that a Driver's trip is successfully linked to that Driver in our database. We must also ensure that trip information is kept securely in our database.

Dependencies with other requirements

A user must have accepted location permissions, and be currently logged in to their account and have the role 'Driver'. See requirements 3.0, 3.2, and 3.3.

3.5. Passenger organises a carpool

Description

A Passenger should be able to organise a carpool by clicking on the “carpool” button after logging in. The user should then be prompted to input the details of their journey, which must include their starting location, destination, time of departure and any other constraints they may have. After successfully inputting this information, the Passenger’s trip should be created and added to the database. The passenger should then be able to search for nearby drivers by clicking on the “search for drivers” button.

Criticality

This is of high importance as Passengers need to be able to set the constraints of their trip in order for the system to show nearby drivers.

Technical issues

We must ensure that a Passenger’s trip is successfully linked to that Passenger in our database. We must also ensure that trip information is kept securely in our database.

Dependencies with other requirements

A user must have accepted location permissions, and be currently logged in to their account and have the role ‘Passenger’. See requirements 3.0, 3.2, and 3.3.

3.6. Search for nearby drivers

Description

A Passenger must be able to find drivers within a selected distance radius and the app must show the most optimal drivers based on shortest distance and the number of Passengers constraints matched with the drivers constraints.

Criticality

This is very important, the app will not be able to function as intended without this requirement. If no Passenger can find a driver then there is no way to carpool.

Technical issues

User location data must be handled securely and when saved for a trip it must be deleted after the trip.

Dependencies with other requirements

A user must have accepted location permissions, and be currently logged in to their account and have the role ‘Passenger’ and have organised a carpool. See requirements 3.0., 3.2., 3.3., and 3.5.

3.7. Request to carpool

Description

When the user has searched for nearby drivers, they should be able to send a request to one of the drivers listed, and the system must notify the driver of their request.

Criticality

This is highly critical, the app will not function as intended if a Passenger cannot request a carpool.

Technical issues

Any details of a carpool request and notifications between Passenger and Driver must be handled securely.

Dependencies with other requirements

A user must have accepted location permissions, and be currently logged in to their account and have the role ‘Passenger’. See requirements 3.0, 3.2 and 3.3. A Passenger must have organised a carpool and have searched for nearby Drivers. See requirements 3.5. and 3.6.

3.8. Accept or decline carpool request

Description

A Driver when receiving a request to carpool must be able to see the details of the route to the Passenger, the Passenger’s constraints matched with the Driver’s constraints and the Passenger’s contact details. The Driver then must be able to choose to either accept or reject the carpool request and if accepted, have a route made for them to the Passenger, notifying the Passenger the trip has been created.

Criticality

This is highly critical, the app will not function as intended if no one can accept and create a carpool or choose to reject it.

Technical issues

Any details of a carpool request and notifications between Passenger and Driver must be handled securely.

Dependencies with other requirements

A user must have accepted location permissions, and be currently logged in to their account and have the role ‘Driver’. See requirements 3.0, 3.2 and 3.3. A Passenger must have requested to carpool with this Driver, see requirement 3.7.

3.9. Calculate optimal route(s)

Description

When a Passenger searches for drivers, the app must be able to generate a route, and the user should be able to find the most optimal route along with other routes found in order of shortest distance and constraints matched. When a Driver accepts a carpool request, an optimal route to the passenger and the destination must be generated.

Criticality

This is critical, a route must be created in order for a Driver to easily find a good route to a Passenger and the final destination.

Technical issues

Optimisation of getting routes based on shortest distances and constraints may need to be considered.

Dependencies with other requirements

A user must be logged in to their account and have one active role i.e. Driver or Passenger. The user must have either accepted a carpool request or have started searching for nearby drivers. See requirements 3.6 and 3.8.

3.10. Update account and role details

Description

The user must be able to update their account details through a settings menu, the user should at least be able to update their phone number, and be able to change role details. For the Driver role, the user should at least be able to change details about their car such as model, and colour. For the Passenger role, they should have the option to change any details such as a description of their overall appearance.

Criticality

This is a low priority since it does not affect what the app is intended for.

Technical issues

Details must be updated and handled securely in the database. Updating details must be handled carefully so that only affects the user’s data and not any unrelated data in the database.

Dependencies with other requirements

A user must have created an account and be logged in order to update their account details. A user must have set a role to change their role details. See requirements 3.1, 3.2, and 3.3.

3.11. Delete account

Description

The user must be able to delete their account and any data associated with that account such as role details.

Criticality

This is a low priority since it does not affect the core functionality of the app.

Technical issues

Deletion of data will need to be handled carefully so that it only deletes the user's data and does not affect any unrelated data in the database.

Dependencies with other requirements

A user must have created an account and be logged in. See requirements 3.1 and 3.2.

4. System Architecture

Frontend

For the frontend development of our system, we will use React Native to create the Android app for the UI. It will communicate with the backend to get data via the API set up in the backend.

Backend

For the backend development of our app, we will use Django. We are both familiar with Django from previous projects. We will make use of Django's REST framework for user authentication, and password encryption. Provisionally, we plan to host the server on a cloud service such as Heroku, Firebase, or Back4App, so it can be evaluated. We plan on using a SQL database for this application.

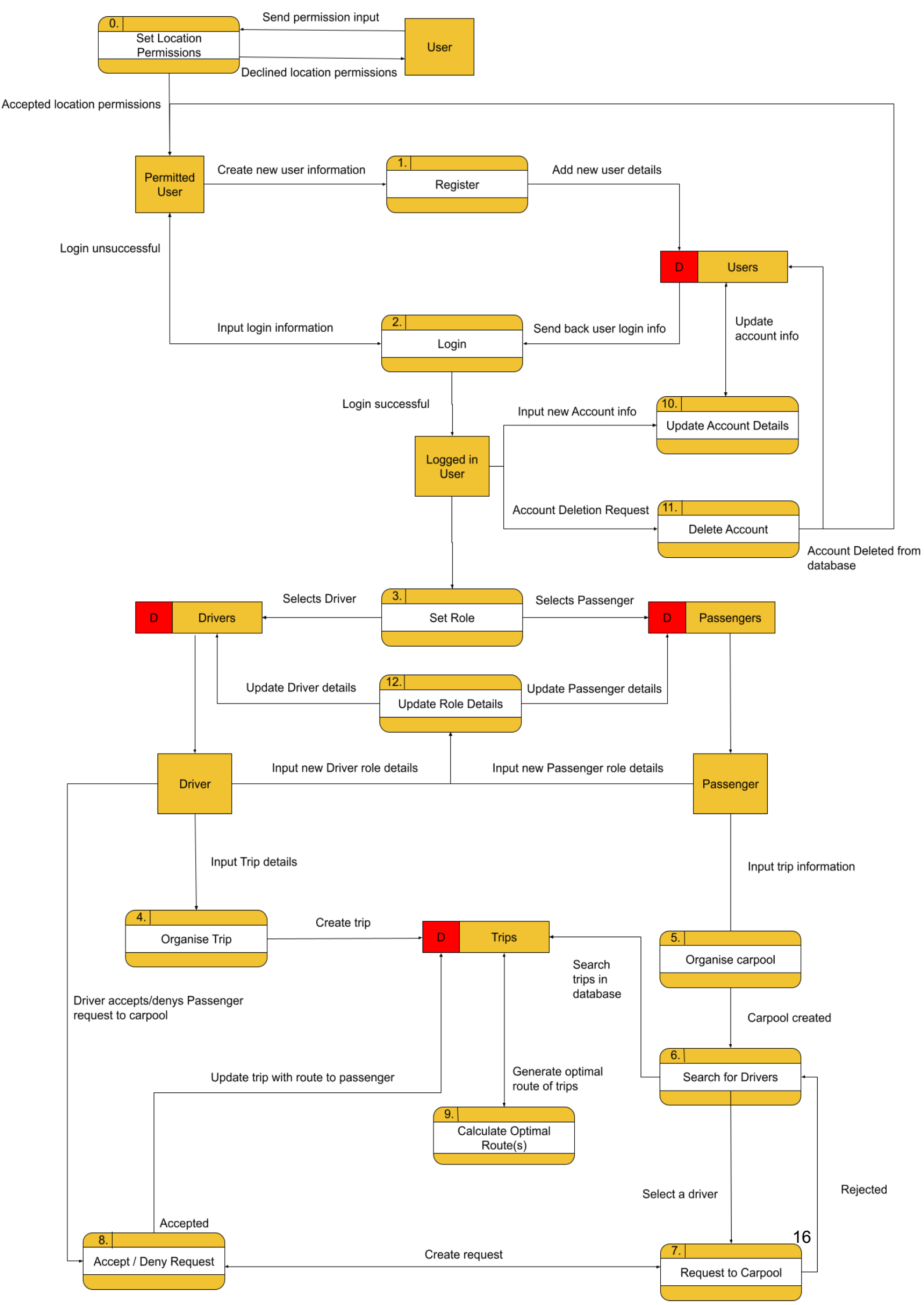
3rd party components

We plan to make use of Google Maps API in order to help in calculating optimal routes and to search for nearby drivers. See requirements 3.6 and 3.9.

5. High-Level Design

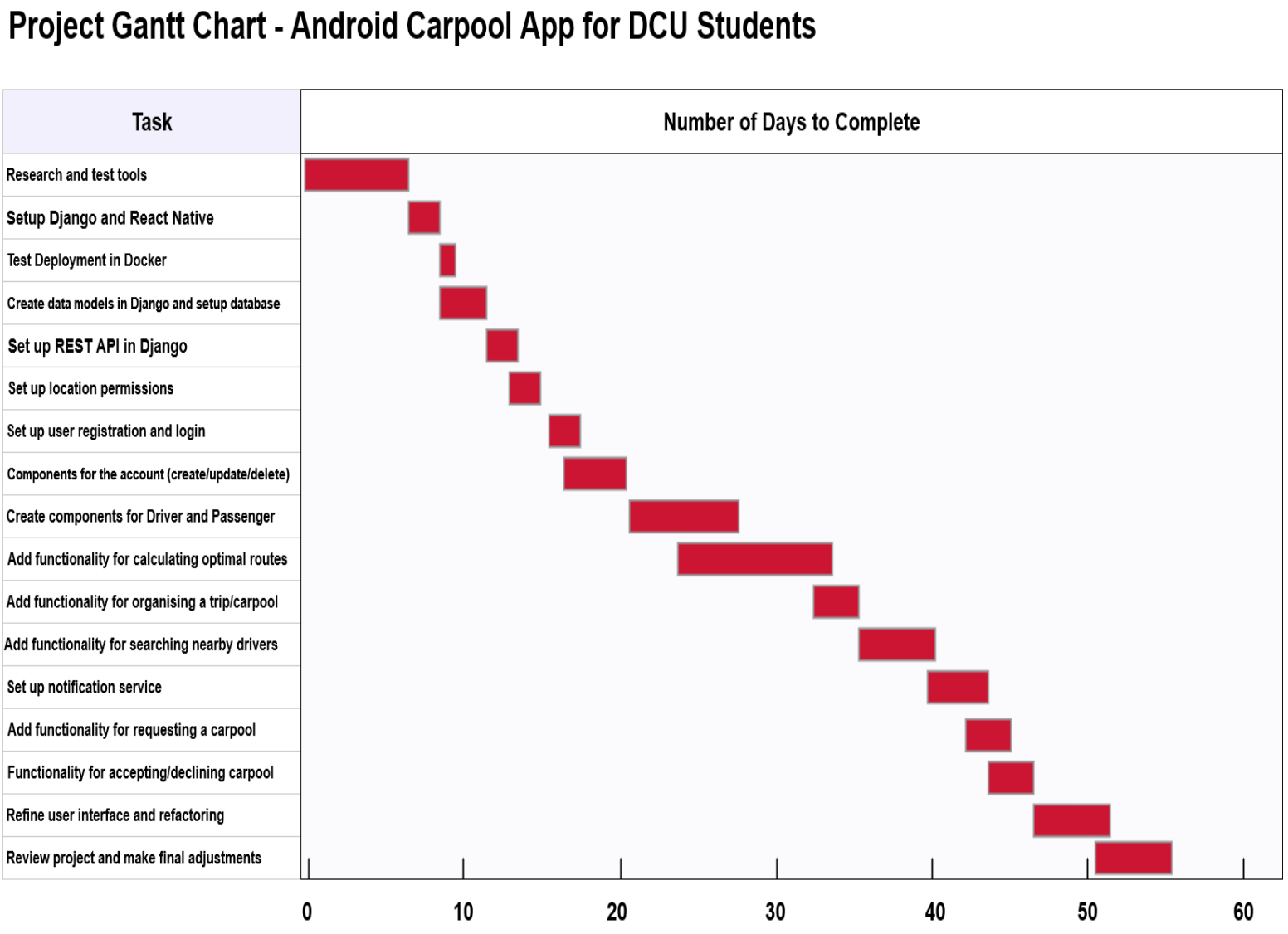
This is a provisional high-level design of the system illustrated using a data flow diagram below.

Data Flow Diagram (next page)


0.
Set Location Permissions
Send permission input
User
Declined location permissions
Accepted location permissions
Permitted User
Create new user information
1.
Register
Add new user details
Login unsuccessful
D
Users
Input login information
2.
Login
Send back user login info
Update account info
Login successful
Input new Account info
10.
Update Account Details
Logged in User
Account Deletion Request
11.
Delete Account
Account Deleted from database
Selects Driver
3.
Set Role
Selects Passenger
D
Drivers
D
Passengers
Update Driver details
12.
Update Role Details
Update Passenger details
Driver
Input new Driver role details
Input new Passenger role details
Passenger
Input Trip details
Input trip information
4.
Organise Trip
Create trip
D
Trips
5.
Organise carpool
Search trips in database
Driver accepts/denys Passenger request to carpool
Carpool created
6.
Search for Drivers
Update trip with route to passenger
Generate optimal route of trips
9.
Calculate Optimal Route(s)
Select a driver
Rejected
Accepted
8.
Accept / Deny Request
Create request
7.
Request to Carpool

6. Preliminary Schedule

We plan to research and test the tools we will require for this project over Christmas, such as React Native and Google Maps API. Our tentative start date to begin setting up the Django and React Native is the 3rd of January. We plan on using an agile approach in the development of this project, having weekly sprints to complete the tasks. The final project deliverables are to be submitted by the 4th of March. Below is a GANTT chart which shows the major tasks to be completed for this project, their interdependencies, and their planned duration. Note that the tasks and duration of each task could change over the course of the project as certain tasks may be more difficult than we first thought.



7. Appendices

7.1 References

[1] Django: https://docs.djangoproject.com/en/4.0/

[2] React Native: https://reactnative.dev/docs/getting-started

[3] Heroku: https://devcenter.heroku.com/categories/reference

[4] Firebase: https://firebase.google.com/

[5] Docker: https://www.docker.com/

[6] Back4App: https://www.back4app.com/

[7] Google Maps API: https://developers.google.com/maps

[8] Django Rest Framework: https://www.django-rest-framework.org/

[9] diagram.net; tool used to create Data Flow Diagram and Gantt chart (See sections 5, 6): https://app.diagrams.net/

7.2. Similar Applications

[1] BlaBlaCar: https://www.blablacar.in/search-car-sharing

[2] UberPool: https://www.uber.com/us/en/ride/uberpool/

[3]: Moovit Carpool: https://moovit.com/carpool/