CA326 - Functional Specification

Adrian Irwin - 20415624
Afolabi Fatogun - 20409054

Template Used:

IEEE Std 830-1998 IEEE Recommended Practice for Software

1. Introduction

1.1 Purpose

The purpose of this document is to provide an overview of the system requirements and analysis for a DCU carpooling application. This document will give examiners and our project supervisors all the necessary information about the system design of the project, also it will outline the requirements for the completion of the project.

1.2 Document Conventions

Term	Definition
The Software/ The app	In reference to the carpooling application
API	Software which allows two applications to communicate.
Algorithm	A set of rules to be followed in calculations or other problem-solving operations.
Dijkstra's algorithm	Algorithm we will use for routing
Constraint	A limitation or restriction
GDPR	General data protection regulations
Android / IOS	Mobile operating systems
Manual	Set of instructions for operating the software.

1.3 Intended Audience

This document is to be examined by the project supervisor and course examiner

1.4 Product Scope

The scope of this product is to develop a Carpooling software which would help DCU students get to and from campus quickly and efficiently. The application will be easy to understand and navigate, and will be available to use on all mobile devices via a mobile web application.

The Software would facilitate communication between drivers and passengers using constraints that will be outlined later in this document. Also the application is only to be available for DCU students.

2. Description

2.1 Product Perspective

The app is a new project originating from a project idea given by the Project supervisor. It aims to be fully independent in its operations and unlike its competitors it is designed for DCU students.

2.2 Product Functions

The Application will be developed using open source API. Using these APIS we will be able to introduce the following functions.

- Log In
- Log Out
- Display mapping
- Calculate and display routes between places
- Passenger enters ride details
- Display Passenger listings to Driver
    - Constraints
- Submit Driver/Passenger feedback

API's to be used in the application are:

- Openstreetmaps [1]
- Openrouteservice [2]

2.3 User Classes and Characteristics

The user classes are:

- Driver: This user class is the more important of the two. The interaction between both classes is directed by the driver who gets a selection of passengers to choose from.
- Passenger: This user class is required to place an advertisement with their preferred constraints. This information is then displayed on the application.

2.4 Operating Environment

The app will run on any phone running either Android or iOS.
The minimum version that the application will run on is Android 11 and iOS 13.

2.5 Design and Implementation Constraints

During the development of this application we would have to work with/around the following constraints:

- GDPR: Because of laws in Europe we are limited to the amount of information we can take/store from users and the amount of time we can do so.
- The project will be tested and developed using mock up user data, locations, and scenarios. Ethical approval will be needed for testing the app with real users even if they are using mock up user data.
- The following technologies will be used in the project:
    - VS Code
    - React Native
    - npm
    - MongoDB
    - Express
    - Node.js
- The database (MongoDB) will be hosted on Mongo's own cloud database service using their 'Shared' tier. [3]

2.6 User Documentation

A user manual will be provided in the application.

3. Functional Requirements

The following are the functions the app will support in order of criticality:

3.1 Login

Description:

The application must log the user into their personal account to distinguish between themselves and other users. User login data will be stored in the database to determine who the user is.

Criticality:

User login is essential to identify who the user is. This allows the system to differentiate between a user that is a driver or a passenger. A user login allows the user to access the rest of the app.

Technical issues:

- User is unable to access the system after log in
- User cannot log in

3.2 Logout

Description:

The application must facilitate the user to log out of their accounts to switch accounts.

Criticality:

User logout is not an overly critical feature but is essential to facilitate customer satisfaction.

Technical Issues:

- User is unable to log out
- User is unable to log back into their account after logging out

Dependencies:

Log out would require the user to be logged in first

3.3 Display mapping

Description:

The application will display an adaptive map which users will use to visualise their position and will aid in routing to and from the university.

Criticality:

Mapping is an intricate feature of the application as the idea is to make the application as user friendly as possible. Hence display mapping is of high critical importance.

Technical issues:

- Map does not display
- User is unable to zoom in and out of the map
- User is unable to move rhythmically around the map

Dependencies:

User would have to be logged in

3.4 Routing

Description:

The application must support visible routing between different points on the map. The application will use “Dijkstra's algorithm”- Used to find the shortest path/routes between nodes on a plane. Using this algorithm we can find the shortest path to get from the driver’s location to DCU efficiently given multiple deviations.

Criticality:

It is critical to the project that the routing system is fully operational as it will be frequently used by all user classes.

Technical issues:

- Wrong routing provided by the application
- Routing does not scale with the map
- Routing is not displayed at all

Dependencies

Users would have to be logged in.

Map will need to be fully functional

3.5 Enter Ride Details

Description:

Passengers are able to enter the details of their ride including:

- Their location
- Constraints i.e male or female driver etc
- Expected time of arrival at DCU

Criticality:

This is of utmost critical importance because The passenger has to enter details of their ride to be displayed to drivers or the interaction between both classes can not occur.

Technical Issues:

- Passengers cannot enter their ride details
- Ride details not displayed properly or at all

Dependency:

User would have to be logged in

3.6 Passenger Listings

Description:

As one of the main features of the application, passengers would be able to enter information about their trip including constraints. This information is then tabulated and displayed to the Drivers.The driver can then accept suitable passengers for their journey.

Criticality:

The passenger ride data has to be displayed to Drivers or the interaction between both classes can not occur.

Technical issues:

- Passenger listings not displayed
- Driver can not accept a passenger

Dependencies:

User would have to be logged in

3.7 Submit Feedback

Description:

After a transaction has been completed both user classes are able to give one another feedback as well as a rating.

Criticality:

This aspect of the application bears little criticality to the whole project as core functionality is able to be achieved with or without it.

Technical Issues:

- User classes are unable to give feedback

Dependencies:

A ride will need to have taken place/occured.

3.8 Constraints

The app will have multiple constraints that the developers have to work with:

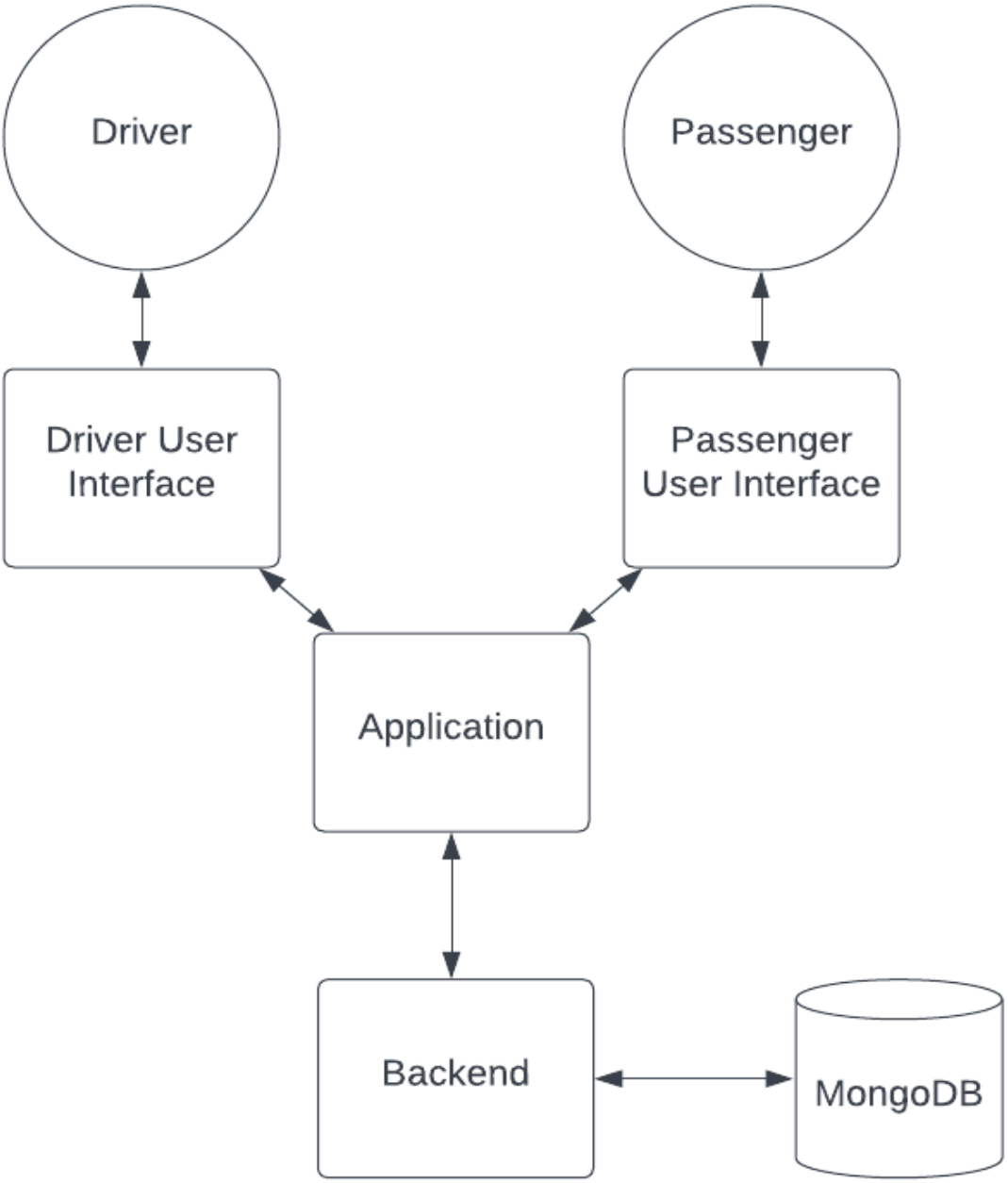
Security Constraints:

- Gender Filtering - Drivers and Passengers may want to filter the gender of the people that will ride them.

Usability Constraints:

- Distance Limit - Drivers will have the ability to choose how far they are willing to travel to pick up a passenger.
- Number of Passengers - The driver will have the option to choose how many seats they have free in their vehicle.

4. System Architecture



The system architecture diagram describes the architecture of the application. It is composed of the application, backend, MongoDB, Driver, Passenger, and Driver/Passenger UI.

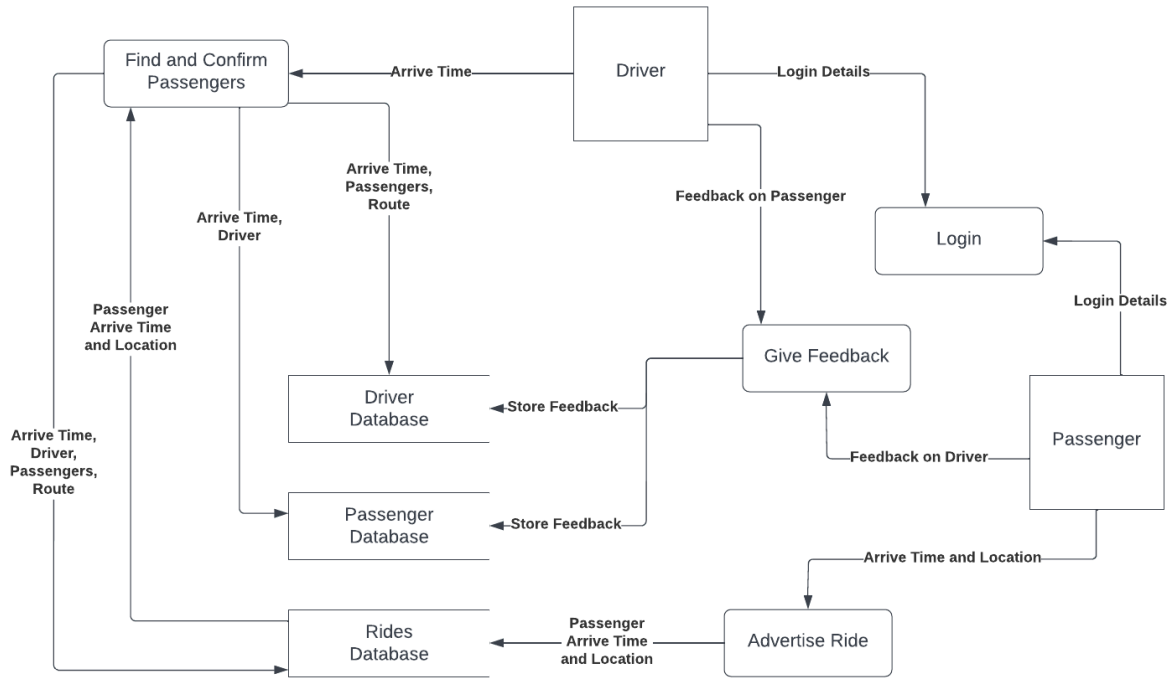
The driver interacts with the driver UI interface (to view/select passengers etc.) which inturn interacts with the application which stores/gets data from the back end which is controlled by MongoDB.

The Passenger interacts with the passenger UI interface (to enter ride information and constraints) which inturn interacts with the application which stores/gets data from the back end which is controlled by MongoDB.

5. High-Level Design

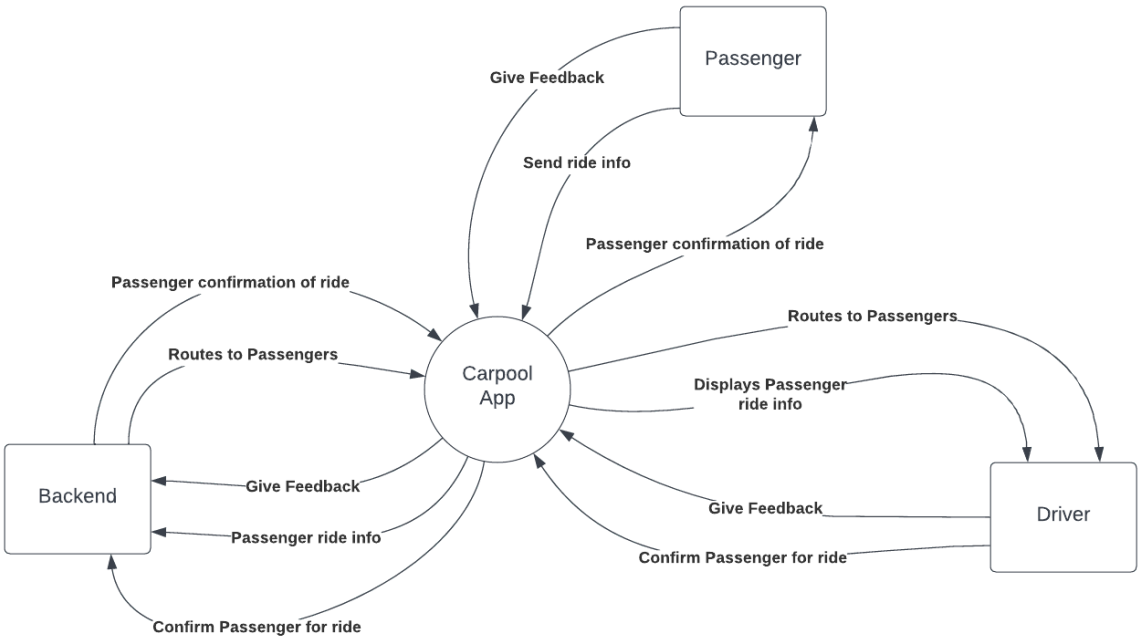
High level diagrams give context to the intricate design of the application as well as interactions between the user classes, UI, and Backend.

Data Flow Diagram



The Data Flow Diagram shows at a high level how data that has been inputted by either the Driver or the Passenger flows through our designed system after being altered in some way by an action.

Context Diagram

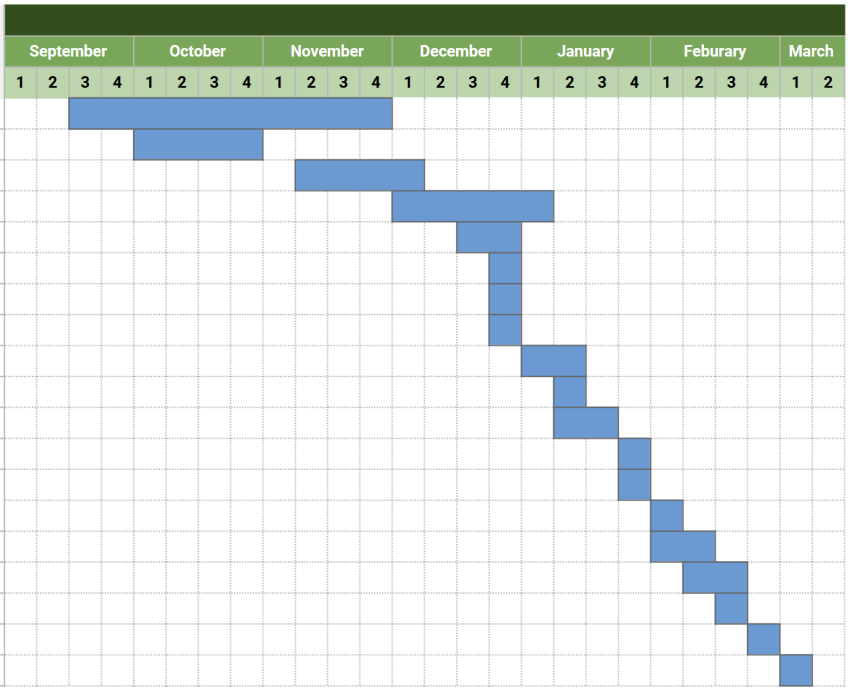


The context diagram shows the interrelationships between the passenger, driver and Backed and the Carpooling app. It highlights all the functional requirements that will be carried out by each element of the application.

6. Preliminary Schedule

The Gantt chart below shows the predicted timeline for when different aspects of the project are to be completed.

TASK ID	TASK TITLE
1	Research
2	Proposal
3	Functional Specification
4	Exam Period and Christmas Break
5	App Creation Tutorials + Practice
6	UI Mockup
7	Login
8	Logout
9	Implement database
10	Display map
11	Routing
12	Ride information(constraints)
13	Passenger listings
14	Feedback feature
15	Testing
19	Write manual
20	User Testing
21	Submission
22	Demo



7. Appendix

References

[1] “API v0.6 - OpenStreetMap Wiki,” wiki.openstreetmap.org. https://wiki.openstreetmap.org/wiki/API_v0.6

[2] “Openrouteservice.” https://openrouteservice.org/

[3] “Managed MongoDB Hosting | Database-as-a-Service,” MongoDB. https://www.mongodb.com/atlas