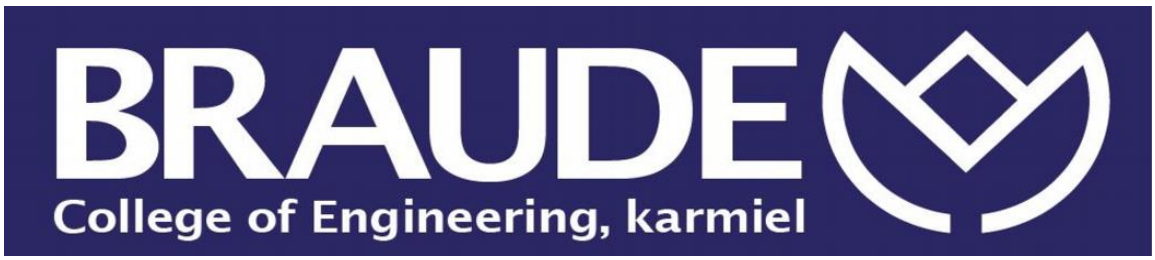


Capstone Project Phase A

Carpool

Match, Drive, Share

project number
24-1-D-29

Supervisor
Dr. Zeev Barzily

Authors
Ravid Goldin, ravid.goldin@e.braude.ac.il
Ben Ben Baruch, ben.ben.baruch@e.braude.ac.il

Date: May 26, 2024

Table of Contents

Abstract

In today's world, traffic congestion and environmental concerns have become pressing issues, particularly in areas with inefficient public transportation systems.
Carpool aims to address these challenges by fostering a collaborative ride-sharing community for drivers commuting to the same destinations regularly, such as workplaces and universities. Unlike traditional carpooling apps, our approach focuses on creating permanent groups, promoting long-term connections and a sense of community among participants.
Through a fair and transparent points-based system, drivers take turns in sharing the driving responsibilities, ensuring an equitable distribution of costs and burdens. Carpool's unique feature lies in its emphasis on building lasting relationships among members, fostering trust, and encouraging sustainable commuting habits.
By leveraging cutting-edge technology and a user-friendly interface, Carpool provides a seamless experience for drivers seeking eco-friendly, cost-effective, and convenient transportation solutions.

Chapter 1
Introduction

In today's world, some places aren't well-connected by public transportation, making many people drive alone to the same destination. This leads to traffic jams, delays, and economic costs like fuel wastage and vehicle wear. Moreover, it contributes to environmental pollution and health risks.

Our app's motivation is to help people driving to the same place, like work or school, share rides and create a community. We aim to be fair to everyone, so our app has a system that takes turns deciding who drives. It makes sharing rides easy, fostering collaboration among drivers. Unlike other carpool apps, we focus only on drivers going to the same place regularly, creating permanent groups for universities or workplaces.

Our goal is to make sharing rides, building friendships, and creating a community simple for drivers who travel to the same place every day.
Our main users are drivers seeking an economic, convenient, and fair solution, and our app appeals to employers, community groups, the Ministry of Transportation, and insurance companies interested in promoting sustainable transportation. In summary, Carpool aims to design a user-friendly solution, connecting drivers, and making commuting fun, easy, and eco-friendly.

The paper is organized as follows: Section 2 presents a review of background and related work, while Section 3 outlines our goals and expected achievements. Section 4 delves into the research process, providing insights into the motivation behind our approach and potential constraints. Section 5 provides details about our software and testing methodology, and Section 6 offers an overview of the evaluation metrics employed in the study.

Chapter 2
Background and Related Work

Carpooling has recently become more popular and has increased over the past few years. Carpooling is an excellent way to make everyone's commute less expensive. Users can connect using these apps and exchange rides to a shared location. It also helps to reduce air pollution. People who commute together use fewer vehicles, which lowers traffic congestion.

2.1 Type of carpool applications

Carpool applications come in two main types, catering to distinct travel needs: periodic travel and temporary travel. Periodic travel applications are designed for users with regular commuting schedules, such as daily trips to work or school. These apps enable users to set up recurring rides, fostering a community of consistent travel companions. On the other hand, temporary travel applications serve those with one-time travel requirements. In this model, users can request rides on a case-by-case basis, connecting with available drivers heading to similar destinations. Both types offer unique advantages, with periodic travel apps providing reliability for daily commuters, while temporary travel apps offer flexibility for users with variable transportation needs, creating a versatile ecosystem to accommodate a wide range of users and travel scenarios.

2.2 Shared ride statistics

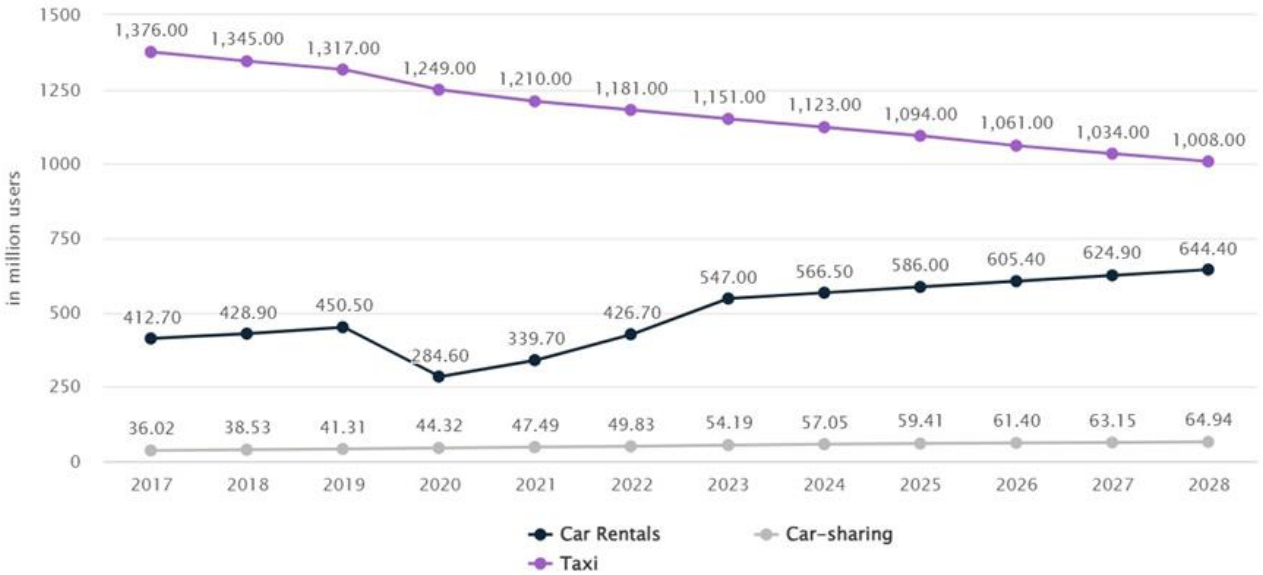


Figure 2.1: *This graph describes the amount of users of taxis, car rental and carpool applications as of 2017 and up to the future forecast for 2028.* [1] *It can be seen that the*

number of users of carpool apps is gradually increasing and is on an upward graph. The future forecast for 2028 predicts an increase of almost 2 times that of 2017.

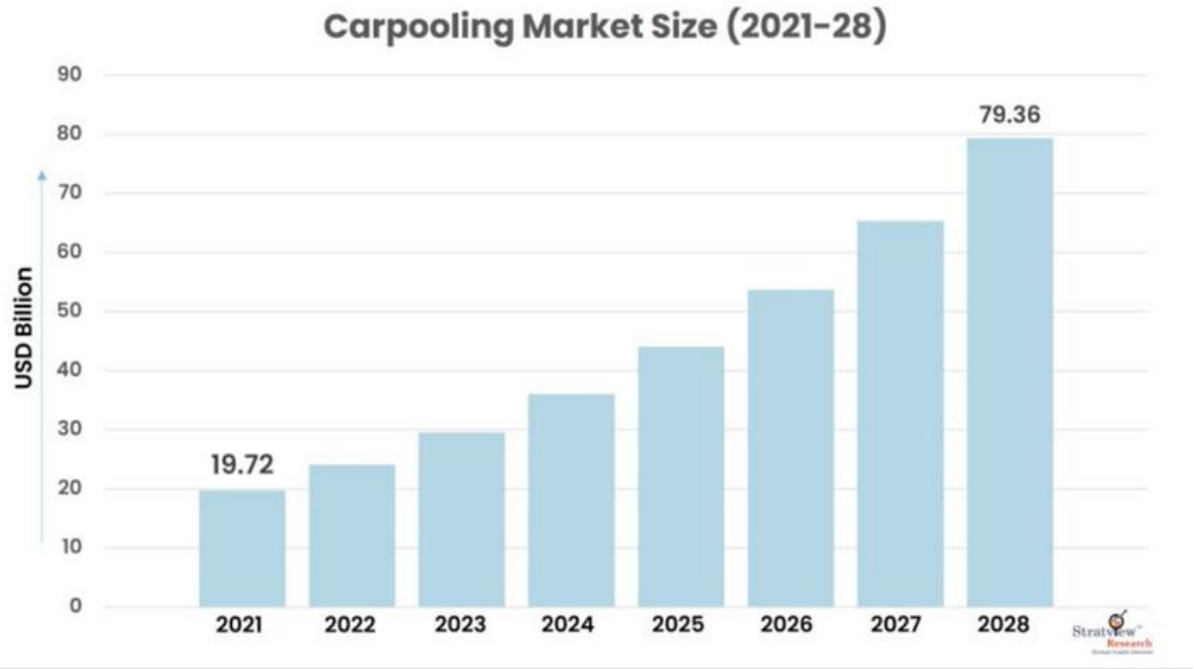


Figure 2.2: *The global carpooling market is expected to grow from USD 19.72 billion.* [2] *In 2021 to USD 79.36 billion by 2028 at a CAGR (compound annual growth rate) of over 21.93% during the forecast period of 2022-2028.*

2.3 The World's Popular Carpooling and Ride-Sharing Apps

2.3.1 Uber: Urban Mobility Leader

Uber [3], a global giant, dominates urban transportation with real-time tracking, dynamic pricing, and seamless integration with public transit. Its user-friendly platform and extensive driver network make it a top choice for commuters worldwide.

2.3.2 BlaBlaCar: Long-Distance Connection

BlaBlaCar redefines carpooling, connecting travelers across cities and countries. Its unique long-distance focus, coupled with efficient route planning, fosters a community seeking affordable and sustainable travel options beyond traditional ride-sharing boundaries.

2.3.3 Waze Carpool: Navigating Commutes Efficiently

Waze Carpool, linked with the popular navigation app, optimizes daily commutes through dynamic route planning and matching. Emphasizing reduced congestion and environmental benefits, Waze Carpool stands out for its community-driven approach to regular ride-sharing.

2.4 Algorithms exist in similar environments.

2.4.1 Type of Real-Time Tracking

It's important to choose a tracking solution that aligns with the goals of the carpooling application, considering factors like accuracy, cost, and user experience. Additionally, ensuring user privacy and complying with relevant regulations is crucial when implementing tracking technologies.
Here are a few tracking devices or technologies that can be used in carpooling applications:
GPS Trackers: Portable devices for real-time location data.
Mobile Device Integration: Leveraging smartphones' built-in GPS.
RFID Technology: Radio-Frequency Identification tags for confined areas.
Bluetooth Beacons: Proximity-based tracking in vehicles.
In-Vehicle Telematics: Using GPS and sensors in modern vehicles.
Wi-Fi Positioning: Triangulating positions based on Wi-Fi signals.
Geofencing Technology: Virtual boundaries triggering notifications.
Machine Learning Predictions: Advanced algorithms for route and timing predictions.

2.4.2 Dynamic Pricing Models

Carpooling applications like Waze employ a dynamic pricing model that takes various factors into account to calculate the cost of a ride. The pricing algorithm typically considers the distance of the journey, the duration of the ride, and the number of passengers sharing the car. Distance and duration are fundamental parameters that contribute to the overall fuel consumption and wear and tear on the driver's vehicle. Additionally, the number of passengers impacts the cost-sharing aspect of carpooling, as more passengers usually result in a lower cost per person. Some apps may also factor in real-time traffic conditions, tolls, or other expenses related to the specific route taken. The goal is to provide a fair and competitive price that encourages drivers to offer rides while ensuring affordability for passengers. The pricing algorithm is a critical component in balancing the interests of both drivers and riders within the carpooling ecosystem.
Our project doesn’t support pricing models because it's a social application for drivers only that is based on equality and fairness.

2.4.3 User Ratings and Reviews

Carpool apps commonly integrate user rating and review systems, fostering trust among users and enhancing service quality. These systems allow users to rate and provide feedback on their ride experiences, helping build a trustworthy community. Analyzing user feedback is crucial for improving the app's overall quality and informs strategies for enhancing user satisfaction. A focus on user reviews can shape approaches to address concerns, prioritize safety, and continually refine the carpooling experience.

2.4.4 Integration with Public Transportation

Carpool apps often team up with public transportation, making it easier for users to plan and combine carpooling with buses or trains. This means users can smoothly connect different parts of their journey. The apps provide real-time info on public transit schedules, help with last-mile trips, and sometimes let users pay for both carpool rides and public transit in one go. By doing this, carpool apps make traveling more convenient, environmentally friendly, and encourage users to stick with their services.
Our project doesn’t support integration with public transportation.

2.4.5 Route planning and Matching

Route planning and matching in a carpool application involve sophisticated algorithms and mechanisms designed to efficiently pair drivers with passengers based on their respective routes and preferences. When a user inputs their starting location and destination, the app employs geospatial algorithms to calculate potential routes, considering factors such as traffic conditions, distance, and estimated travel time. Simultaneously, the system analyzes other users' journey requests in the same vicinity and identifies optimal matches. The matching process considers factors like similar origin and destination points, compatible routes, and any intermediate stops that users are willing to make. Advanced algorithms, often based on graph theory and optimization techniques, aim to minimize detours, and maximize shared segments of the route. The idea is to make the carpooling experience smooth and save time for everyone involved by connecting people who are headed in the same direction.
Our project does not use the matching algorithm because drivers join in advance the groups that match them in terms of the stations. In addition, the route is determined and displayed by a starting point, a destination point and intermediate stations.

Chapter 3
Expected Achievements

This project goal is to provide a comprehensive solution to the challenges faced by drivers seeking efficient and cost-effective ridesharing options. The development of a user-centric mobile application by using frameworks of Flutter and Firebase. The app will have a range of dynamic features including route planning and matching algorithms for optimized shared rides.

Through the utilization of Flutter, a cross-platform framework, the application will ensure a consistent and visually appealing experience for both Android and iOS users. Concurrently, Firebase will serve as the robust backend solution, facilitating secure data storage, real-time updates, and reliable user authentication. The anticipated outcome is a fully functional carpooling application that not only simplifies daily commutes but also fosters a sense of community and sustainability among its users.

The success criteria for this project are various facets, including the seamless integration of Flutter and Firebase, the creation of an intuitive user interface, and the efficient implementation of backend functionalities. Ultimately, the project aims to provide a reliable and scalable carpooling solution that addresses the diverse needs of modern commuters while promoting the benefits of shared mobility.

Chapter 4
Engineering Process

4.1 Process

We have structured our learning process and development efforts into distinct phases. These encompass organizing meetings couple times in the week, defining system requirements, and crafting diagrams to visualize the end product. Furthermore, we've conducted research on the technologies we intend to utilize, as well as the necessary development tools for building the application. This includes exploring platforms such as Flutter, Firebase, as well as APIs like Mapbox and Google Maps.

4.1.1 Engineering research and development process

In the initial stages of our engineering process, we conducted an in-depth examination of the current carpooling applications. Through this market analysis, we identified crucial features to incorporate into our system. With this knowledge, we strictly outlined the system requirements and crafted a comprehensive work plan encompassing every essential step, from the learning phase to the development stage. Additionally, alongside our engineering research, we conducted manual testing of our project concept using instant-messaging group to assess its efficacy in addressing the identified problem.

We have organized our learning and development journey into specific stages. These include scheduling regular meetings throughout the week, describing system requirements, and creating diagrams to visualize our final product. We created a sketch of how the application will look and also a flow chart and transition between the application pages. In addition, we delved into researching the technologies we plan to use, along with identifying the essential development tools required to build the application. This involves exploring platforms such as Flutter and Firebase, along with APIs such as Mapbox and Google Maps.

4.1.2 Challenges

One of the challenges was to think about how to realize the idea that the group has a number of stopping stations and show them on the map as a route. We were looking for an API that enables this and that's how we found Mapbox's service. We saw that it is necessary to create coordinates from a certain address (the stopping point) and thus integrate it into the route map of the drive group.

4.2 Product

4.2.1 Software architecture

Our software architecture is designed to facilitate seamless interaction between different components, including the user, the application, and external services such as Firebase and Mapbox.

- **User**: Users interact with our carpooling application by logging in, creating new travel groups, searching and joining existing groups, and selecting preferred stopping points for their rides. Additionally, users can view and interact with the map interface to visualize starting destinations, stopping points, this interaction is bidirectional, with users initiating actions, and the application responding accordingly by processing these requests and providing feedback to the users.

- **Application**: Serving as the central component of our architecture, the application processes user requests, manages data, and interacts with external services. It encompasses the software logic, determining how user inputs are handled and how data is presented. Additionally, the application communication with Firebase and Mapbox APIs.

- **Firebase**: [4] An integral part of our architecture, Firebase provides backend services essential for the functioning of the application. The application directly communicates with Firebase, leveraging services such as Firestore for real-time data storage, Firebase Authentication for user verification.
- **Mapbox API**: [5] We utilize the Mapbox API to integrate mapping and location services into our application. This allows for the visualization of starting destinations, stopping points during trips. The Mapbox API enhances the user experience by providing accurate and interactive maps within the application.

By emphasizing seamless interaction between these components, our software architecture ensures a robust and user-friendly experience for drivers participating in carpool

Figure 4.1: *Represents the software architecture of Carpool, detailing interactions between users, the IOS application, Firebase backend services, and MapBox API.*

4.2.2 Model-View-ViewModel

Application Architectural Pattern (MVVM): The application will employ the Model-View-ViewModel (MVVM) architectural pattern. [6] MVVM has gained prominence in mobile application development, particularly within the Flutter framework, due to its ability to enhance maintainability.

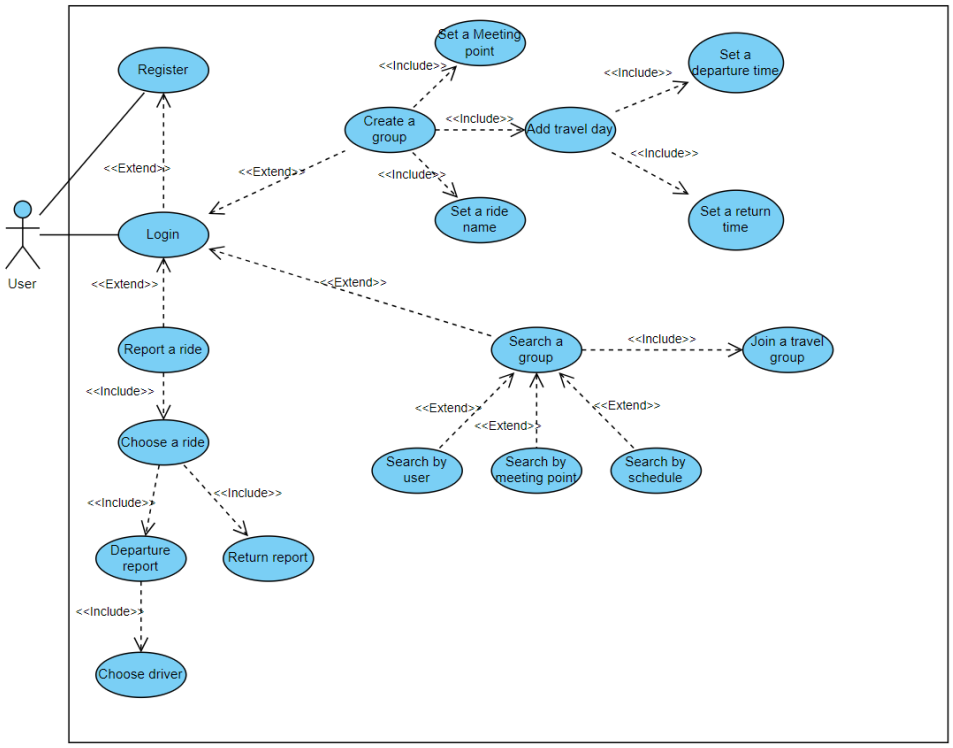
- **Model** component represents the data structures and business logic of the shared ride app. Its responsibilities include managing data retrieved from Firebase, such as user profiles, authentication information, and group details. Additionally, it handles data from the Mapbox API related to locations and routes.

- **View** component encompasses the visual elements of the app with which users interact. These include screens such as the login interface, group creation interface, and group detail page.
  The View's role is to display data provided by the ViewModel and to transmit user interactions, such as button presses, back to the ViewModel.

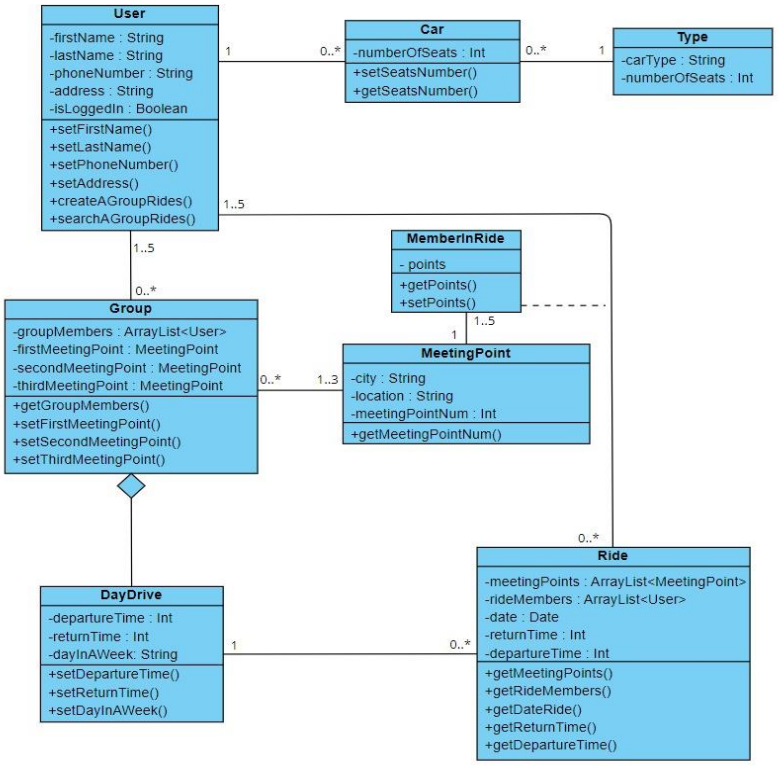
- **ViewModel** acts as an intermediary between the Model and the View. It processes data from the Model, such as filtering travel group information and formatting location data for display. Additionally, it responds to user interactions by updating the Model as necessary, such as joining a travel group or selecting stopping points.
  The ViewModel ensures separation of concerns by handling business logic and data formatting, making it easier to maintain and test.

By adopting the MVVM pattern, it facilitates easier modification and expansion of features while keeping the codebase organized and manageable.

4.2.3 Use Case diagram

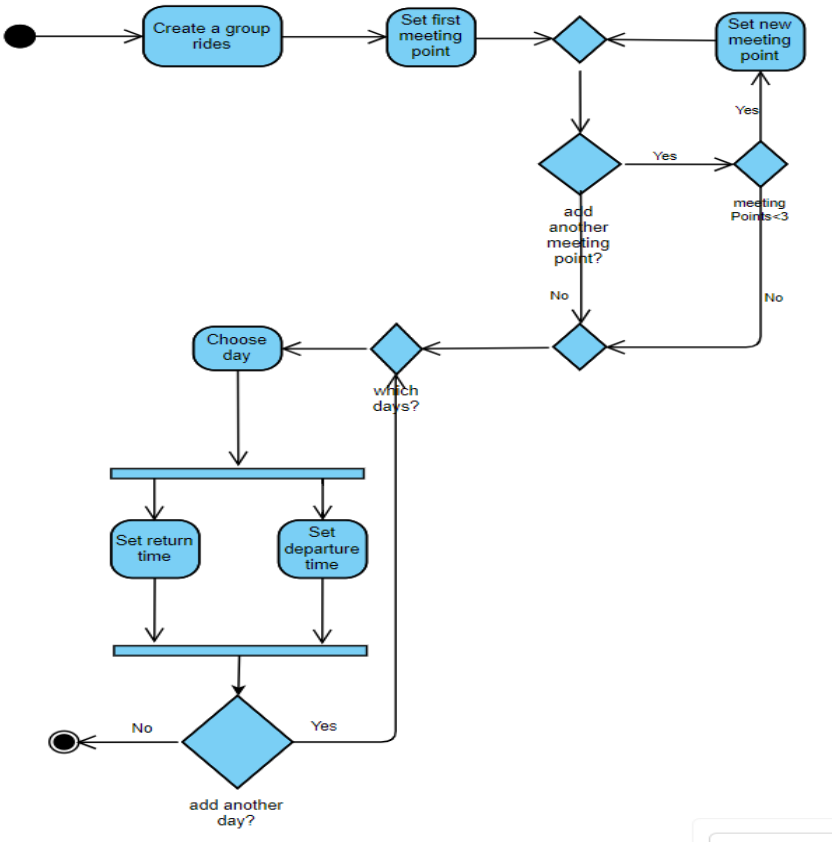


4.2.4 Class diagram



4.2.5 Activity diagram

The activity diagram outlines the process of creating a group ride. The initial step involves the user setting meeting points, where a group must have a minimum of one meeting point and a maximum of three meeting points.
After establishing the meeting points, the user proceeds to choose the schedule for the group ride. This includes selecting the days of the week and specifying the times for departure and return.



4.2.6 Technology stack

• **Flutter:** Our project utilizes Flutter as the framework for building the user interface and managing app behavior. Flutter, known for its fast development capabilities and expressive UI, enables us to create visually appealing, cross-platform mobile applications. With Flutter's hot reload feature, developers can quickly iterate on app designs and see instant results, enhancing productivity during the development process. Its extensive widget library and customizable components allow for the creation of rich and engaging user experiences across different devices and screen sizes.

• **Firebase:** Firebase serves as the backend infrastructure for our project, offering a suite of tools and services to support app development and management tasks. We leverage Firebase's Firestore Database for storing and managing app data in a scalable and secure manner. Firebase Authentication simplifies user authentication processes, enabling seamless registration and login functionality through various authentication methods, including email/password and social media accounts.

• **MapBox API**: To integrate mapping and location-based functionalities into our app, we utilize the MapBox API. MapBox provides robust mapping tools and services, allowing us to embed interactive maps, display custom layers, and visualize spatial data within our application. With MapBox's versatile mapping capabilities and extensive customization options, we can tailor map displays to suit our app's specific requirements and branding. Furthermore, MapBox offers advanced features such as geocoding and routing, enabling us to provide users with accurate location information and optimized navigation experiences. By leveraging the MapBox API, we enhance the usability and functionality of our app, empowering users to explore and interact with geographical data seamlessly.

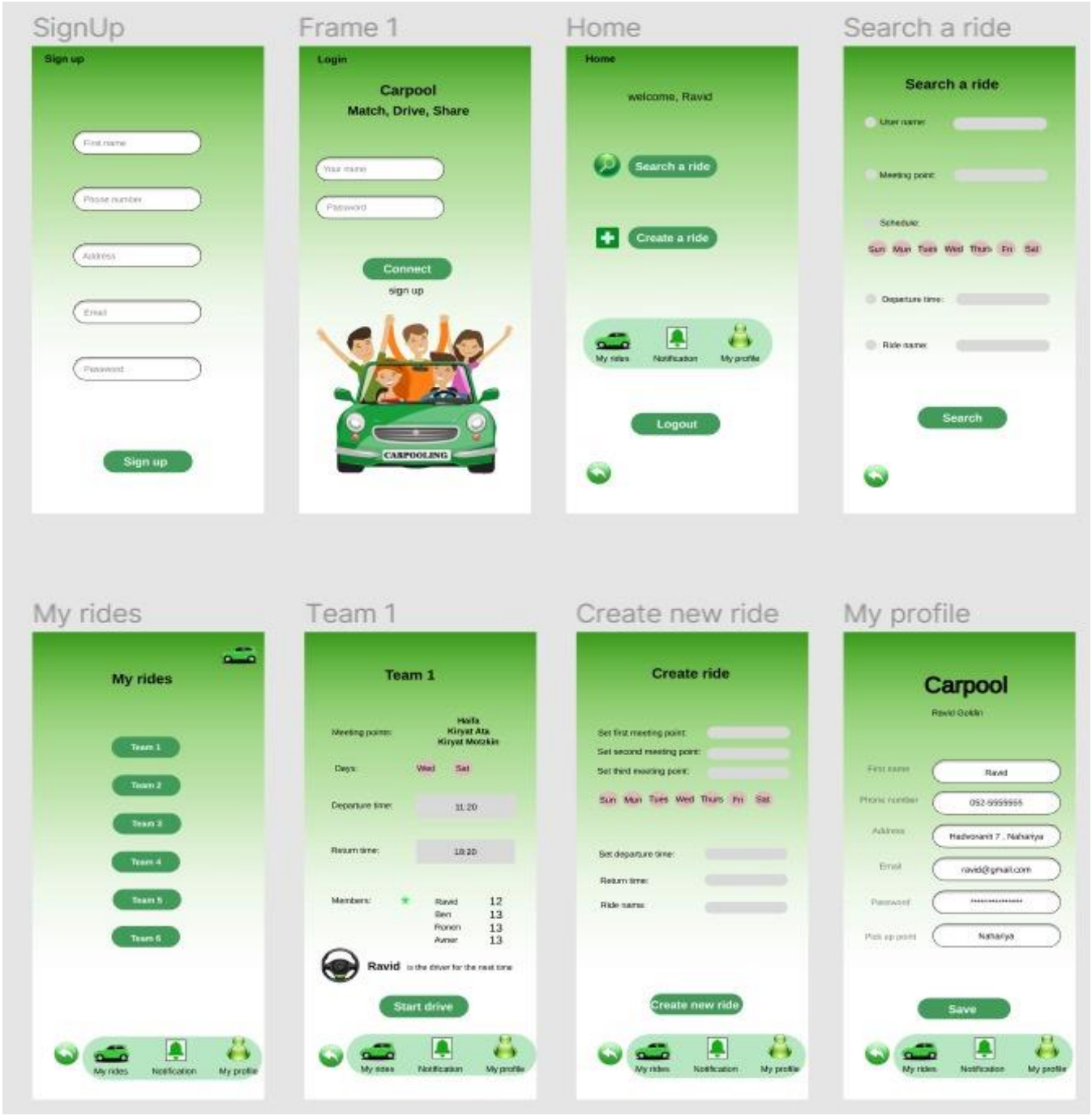
4.2.7 Functional requirements

#	Category	Description	Priority (1-5), 1-High, 5-Low
1	General	Integration with MapBox API for displaying maps with meeting points.	1
2	General	Integration with Firebase to enable real-time data synchronization.	1
3	General	Send push notifications.	2
4	General	Points algorithm.	1
5	General	The app will include a help section for user guidance.	5
6	General	Successful and failed operations should be logged.	2
7	User	Login process to the application.	1
8	User	Register process to the application.	1
9	User	Logout process from the application.	1
10	User	Manage and edit your profile.	2
11	User	View ride groups of users.	2
12	User	Search group rides by parameters.	1
13	User	Create group rides.	1
14	User	View the group's ride meeting points.	3
15	User	View the group's ride points table.	2
16	User	Report a ride is complete.	1
17	User	View My rides.	3

4.2.8 Non-Functional requirements

#	Category	Description	Priority (1-5), 1-High, 5-Low
1	Performance	Data synced in real-time via Firebase integration.	1
2	Performance	Optimize navigation and scrolling experience within the app.	3
3	Performance	User-friendly and intuitive interface design.	3
4	Security	Secure user authentication and access control.	1
5	Security	Ensure data security through encrypted storage and transmission.	1
6	Portability	Support for cross-platform.	4
7	Maintainability	Compatibility and responsiveness with various devices' screen sizes.	2
8	Maintainability	Consistent updates aimed at bug resolution and feature enhancement.	3
9	Maintainability	Codebase designed for modularity and seamless extension.	2
10	Privacy	Ensuring the safeguarding of user data.	2

4.2.9 Graphical User Interface (GUI)



Chapter 5
Evaluation/Verification Plan

5.1 Unit testing

Unit testing refers to the practice of evaluating individual units or components of a software application separately to ensure they operate as expected. This process aids in early detection and resolution of defects during the development phase, consequently enhancing the overall code quality. Moreover, unit tests act as a safety net, allowing developers to refactor or restructure the codebase confidently, knowing that any unintended behavioral changes will be caught. The following table outlines some of the unit tests that will be performed:

Component	**Test Case**	**Expected Result**
User Registration	Register a new user with valid details	Registration has been successful. New user details stored in Firebase.
User Registration	Register a new user with invalid details such as wrong email format or short password	Registration has failed. Error message should be displayed.
User Registration	Register a new user with existing user details	Registration has failed. Error message should be displayed.
User Login	Login with valid user details	Login has been successful. Redirection to the main page of the application.
User Login	Login with invalid details	Login has failed. Error message should be displayed.

Create a new group ride	Create a new group with valid details	Creation has been successful. Redirection to page of group.
Create a new group ride	Create a new group with wrong details such as empty schedule or meeting points	Creation has failed. Error message should be displayed.
Delete group ride	Create a new group and then delete it.	Deletion has been successful. Redirection to the main page.
Delete group ride	Delete an existing group with 2 or more participants.	Deletion has failed. Error message should be displayed.
Search for groups	Search for groups based on parameters	Search has been successful, and results are valid.
Search for groups	Search for groups based on not valid parameters	Search has failed. Error message should be displayed.
Profile management	Edit user profile with valid details	Profile should be updated successfully.
Profile management	Edit user profile with wrong details such as empty fields or wrong format	Profile should not be updated, and error messages should be displayed.
Join a group ride	Join a new group ride	Users have been joined to the group, and the ranking points table should be updated.

Join a group ride	Join to full group ride	User has not joined the group because it is already full. Error message should be displayed.
Points algorithm (Report after drive)	Validate points are updated after a drive.	Points should be updated in the table of drivers group. Driver's points should be increased by 1.
Points algorithm (Choose driver)	Validate the driver is being chosen by lowest number of points.	The driver for the next ride should be the one with the lowest number of points.

5.1.1 Manual testing

Test case	**Steps**	**Expected Results**
Map creation	Create a group with meeting points and see if the map is correct (Made by MapBox API)	Map route should be correct.
User Interface	Navigation through all the pages of application (Homepage, Profile, My rides, Search)	The application should navigate to the correct pages based on the activity of the user.
Multi-platform	Validate application is working on iOS and Android.	The application should be working on both systems because it is written by Flutter.

References

[1] Shared ride statistic- https://www.statista.com/outlook/mmo/shared-mobility/united-states#revenue

[2] Global carpooling market- https://www.stratviewresearch.com/1554/carpooling-market.html

[3] Best Carpool App- https://www.konstantinfo.com/blog/rideshare-apps/

[4] Privacy and Security in Firebase - https://firebase.google.com/support/privacy

[5] Mapbox SDK- https://docs.mapbox.com/ios/maps/guides/

[6] MVVM architecture- https://builtin.com/software-engineering-perspectives/mvvm-architecture