

Deliverable #2 Template

SE 3A04: Software Design II – Large System Design

Tutorial Number: T02

Group Number: G3

Group Members:

- Mya Hussain
- Moamen Ahmed
- Jinal Kasturiarachchi
- Nivetha Kuruparan
- Aadil Rehan

IMPORTANT NOTES

- Please document any non-standard notations that you may have used
 - *Rule of Thumb*: if you feel there is any doubt surrounding the meaning of your notations, document them
- Some diagrams may be difficult to fit into one page
 - Ensure that the text is readable when printed, or when viewed at 100% on a regular laptop-sized screen.
 - If you need to break a diagram onto multiple pages, please adopt a system of doing so and thoroughly explain how it can be reconnected from one page to the next; if you are unsure about this, please ask about it
- Please submit the latest version of Deliverable 1 with Deliverable 2
 - Indicate any changes you made.
- If you do NOT have a Division of Labour sheet, your deliverable will NOT be marked

1 Introduction

The SRS for a mobile taxi-sharing application aims to provide a convenient way for customers to arrange taxi carpools in order to minimize the cost of a trip. The application is being developed for the local taxi company, CarpoolClan, to attract more customers and increase revenue in the long term.

1.1 Purpose

The purpose of the SRS is to serve as a basis for communication between the developers and stakeholders to ensure that all parties have a clear understanding of the application's functionality, features, and technical requirements. The SRS will serve as a blueprint for the developers and acts as a reference throughout the process and testing phases, making sure that the final product meets the customer's needs and expectations.

The intended audience for SRS is the stakeholders involved in the development and implementation of the mobile taxi-sharing application, including:

1. Developers: The developers, software architects, and engineers who will be responsible for designing, building, and testing the app.
2. Customers: The customers who will be using the application to arrange taxi carpools, including information about destinations, taxi IDs, and estimated fares.
3. Project Manager: The person responsible for overseeing the process, ensuring the project stays on schedule.
4. Taxi Company: The local taxi company that has commissioned the development of the application and will be using it to attract more customers and increase revenue.
5. Investors: Anyone who has provided financial support for the project and wants to understand the requirements for the application.

1.2 System Description

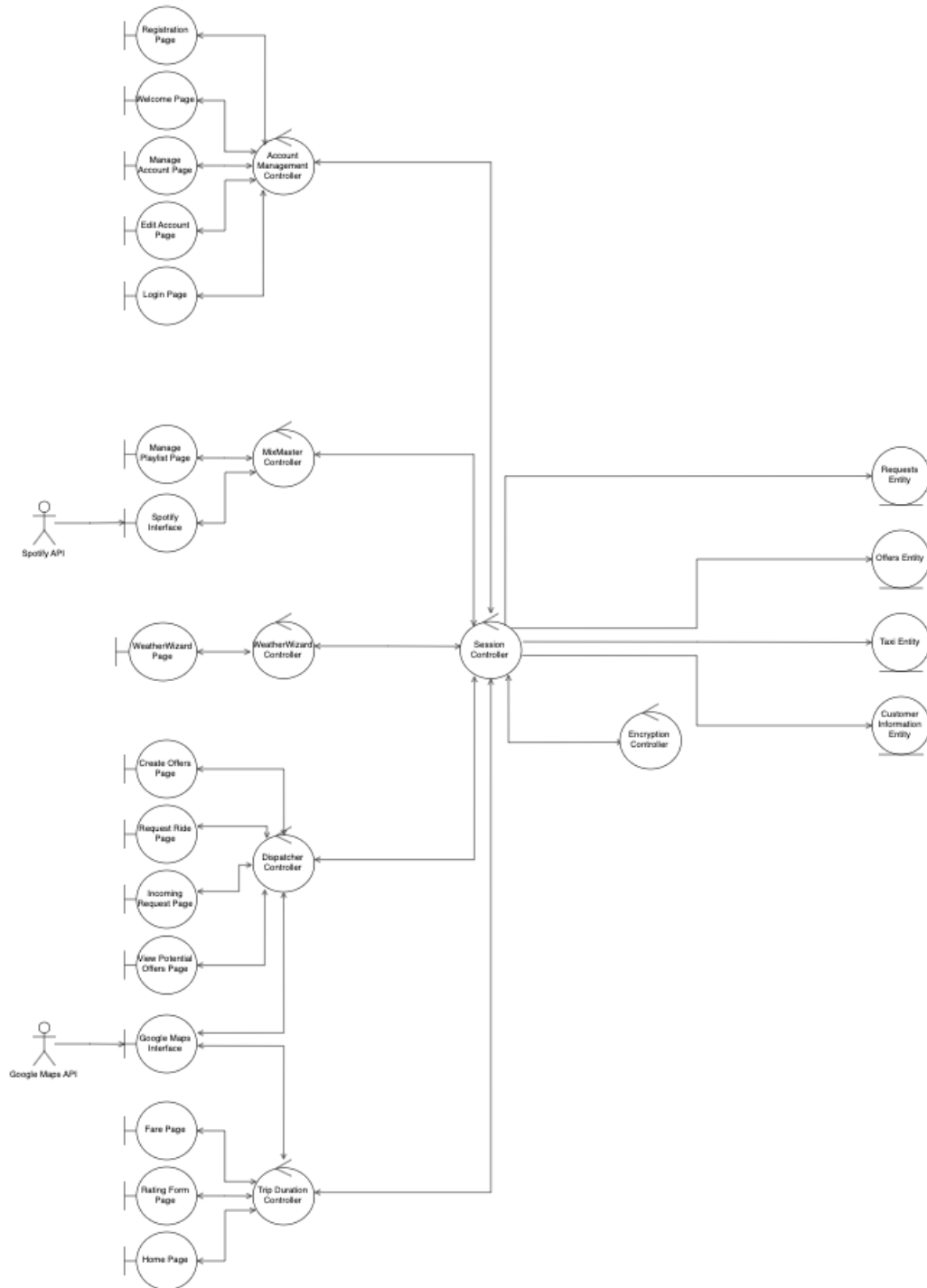
A mobile taxi carpool application for the local taxi firm, CarpoolClan, is being developed. The app aims to offer an affordable and straightforward solution for customers to arrange shared taxi rides. It will incorporate several software components, including online mapping, carpooling features, and secure data encryption to protect user privacy. The app will provide many benefits, such as increased efficiency, a user-friendly interface, cost savings, reduced environmental impact, and improved customer satisfaction.

The application will enable users to create accounts and select their preferred co-riders. All information will be protected with AES encryption. Users can both request and offer ride-sharing services, with the option to choose from a range of potential routes. The app includes the MixMaster feature, allowing for music to be queued during the ride, as well as the WeatherWizard feature, enabling adjustments to the taxi's air conditioning. The main aims of the app are to satisfy customer expectations and needs, and to provide a secure platform for the storage and transmission of sensitive data.

1.3 Overview

The SRS document comprises a class analysis diagram, an overarching structural software architecture, and CRC cards, which will facilitate further object-oriented analysis. The class analysis diagram provides a comprehensive breakdown of the system's boundary, identity, and control classes, all derived from the use cases. The diagram also illustrates the connections between each class and its attributes. The application has been designed using the MVC architecture, which offers benefits such as maintainability, scalability, and reusability; other architectures were also considered but were ultimately dismissed. Additionally, the subsystems of the application are listed below to provide an overview of the individual features and their respective functions.

2 Analysis Class Diagram



3 Architectural Design

Our CarpoolClan application follows the MVC architecture, which divides the application into three interconnected parts: Model, View, and Controller.

- The Model is responsible for managing the data and business logic of the application. It stores and retrieves data from the database and provides the necessary data to the View component. Some components that the model would include:
 - **User profiles:** This would include user information such as name, phone number, and email address.
 - **Ride offers/requests:** This would include the details of users who are offering/requesting a ride.
- The View component is responsible for presenting the data to the user in a user-friendly manner. It consists of the user interface and interacts with the user to receive input. Some views that the controller would include:
 - **Login page:** This page is to allow users to log in.
 - **Registration page:** This page is to allow users to create an account.
 - **Ride offers/requests Page:** This page is to allow the users to offer/request a ride.
 - **Carpool Matching Page:** This page displays a list of potential matches of rides.
- The Controller is responsible for handling user input and controlling the flow of data between the Model and the View. It receives input from the user through the View and uses this input to manipulate the data in the Model. The Controller then updates the View with the new data. Some components that the controller would include:
 - **Profile:** This controls the user's information like name, phone number, and email address.
 - **Authentication:** This controls user authentication, including login and sign-up.
 - **Ride offers/requests:** This controls the creation and retrieval of ride offers/requests.
 - **Dispatcher:** This controls the matching of rides.

By separating the application logic into these three components, the MVC pattern achieves high cohesion within each component, as each component is responsible for a specific task. At the same time, the pattern achieves low coupling between the components, as each component only communicates with the other two through well-defined interfaces, rather than depending on them directly.

3.1 System Architecture

We chose the MVC architecture for our CarpoolClan application for several reasons. Firstly, the architecture separates the concerns of the application into three distinct parts, making it easier to maintain, modify, and test each part independently. Secondly, it is highly scalable, as each part can be modified and extended without affecting the other parts. Lastly, the architecture promotes reusability, as the same Model can be used by different Views and Controllers, and the same View can be used by different Controllers. This separation of concerns allows for the development team to focus on individual parts of the application without worrying about how it will affect the other parts. For example, changes to the UI will not affect the business logic and data storage, and changes to the data storage will not affect how the data is presented to the user.

Benefits of MVC

- **Maintainability:** The MVC architecture allows for easier maintenance and testing of the application. Since each part is independent, it is easier to locate and fix bugs or issues.

- **Scalability:** the MVC architecture allows for each part of the application to be modified and extended without affecting the others. This means that as the CarpoolClan application grows and more features are added, the Model, View, and Controller can all be modified and extended as needed.
- **Reusability:** The MVC architecture promotes reusability, as the same Model can be used by different Views and Controllers, and the same View can be used by different Controllers.

Overall, the MVC architecture is an ideal choice for our CarpoolClan application due to its ease of maintainability, scalability, reusability, and independency of concerns. It allows us to develop and maintain the application more effectively, while also ensuring a high level of flexibility for future changes and updates.

Design Alternatives:

Some of the design alternatives that we considered for our application were the three-tier architecture and the Client-Server architecture.

The three-tier architecture divides the application into three layers: the Presentation Layer, Application Layer, and Data Layer.

- **Presentation layer:** The Presentation layer handles the UI and is responsible for presenting data to the user and receiving input.
- **Application Layer:** The Application Layer, also known as the logic layer, processes user requests, applies business logic, and communicates with the Data Layer.
- **Data Layer:** The Data Layer is responsible for managing the application data, such as storing and retrieving data from a database.

The three-tier architecture provides better separation of concerns, modularity, and scalability compared to the two-tier architecture. However, it introduces additional complexity and may not be necessary for smaller applications like our CarpoolClan application. In addition, it can increase coupling between layers, making the application more difficult to maintain and modify. Therefore, we decided to use the MVC architecture for our application, which provides a similar level of separation of concerns and modularity while being more suitable for our needs.

Another alternative that we evaluated was the Client-Server Architecture, which divides the application into a Client and a Server.

- **Client:** the client sends requests to the server.
- **Server:** processes the requests and sends responses back to the client.

The client-server architecture was not chosen for our CarpoolClan application for several reasons. Firstly, the client-server architecture assumes a one-way flow of information, with the client sending requests to the server and the server processing them and sending responses back to the client. This model is not ideal for our application, which requires real-time communication and interaction between users, as users need to be able to offer and accept carpools quickly and efficiently. Furthermore, the client-server architecture has limited scalability because it relies on a single server to manage all client requests, which may cause performance issues if there is a high volume of users. In addition, transmitting requests and responses over a network can add latency and negatively impact user experience. Ultimately, we decided to use the MVC architecture for our application.

3.2 Subsystems

All subsystems contained in our system are as follows:

- Data Encryption/Decryption Subsystem
- User Authentication and Authorization Subsystem
- Ride Request Subsystem
- Ride Offer Subsystem
- Geolocation Subsystem
- Ride Rating Subsystem
- Notification Subsystem
- WeatherWizard Subsystem
- MixMaster Subsystem

The descriptions of these subsystems can be found below:

Data Encryption/Decryption Subsystem

This subsystem deals with the security of data contained in our system. Its main functionality is to take input data (either directly from the user or from within the system itself) and encrypt it using the Advanced Encryption Standard (AES) before storing it. This subsystem is also responsible for decrypting required data from the stored location and providing it to the system where needed/requested. It is effectively related to all other subsystems since they all handle data.

User Authentication and Authorization Subsystem

This subsystem is responsible for ensuring that any user attempting to log into the application is providing correct credentials for a valid account in the system and effectively authorized to make actions such as requesting and offering rides. This subsystem is related to the Data encryption/decryption subsystem in order to compare credentials effectively.

Ride Request Subsystem

The ride request subsystem should allow users to request rides by specifying their pickup and drop-off locations. Once the request is complete, the relevant information is sent to the dispatcher in order to match the request with potential offers. It relies on the geolocation subsystem, as it needs to be able to identify the location of the user. Additionally, it is related to the data encryption/decryption subsystem, as all data pertaining to ride requests must be secure.

Ride Offer Subsystem

This subsystem is responsible for providing users with the ability to offer rides by indicating where they are able to pickup and drop-off riders. Once the offer is submitted, the dispatcher receives the offer and matches it with any potential requests that currently exist. The geolocation subsystem is relied on, as an accurate location of the user is required. The data encryption/decryption subsystem is also utilized in order to secure information regarding offers.

Geolocation Subsystem

The geolocation subsystem provides the application with locations of users. It is responsible for tracking user locations and route information. The subsystem relies on the GPS technology on users' devices and is critical for the ride request and ride offer subsystems.

Ride Rating Subsystem

The ride rating subsystem allows users to rate their ride experience after the ride is complete. It collects user feedback and driver ratings, which are used to improve the service's overall quality. The subsystem relies on the user authentication subsystem to ensure that only registered users can leave ratings.

Notification Subsystem

This subsystem sends notifications to users regarding ride status updates, such as ride acceptance or vehicle arrival time. It is responsible for ensuring that users stay informed throughout the ride process. It is integrated with the ride request and ride offer subsystems, as notifications will be associated with requests and offers that specific users submit.

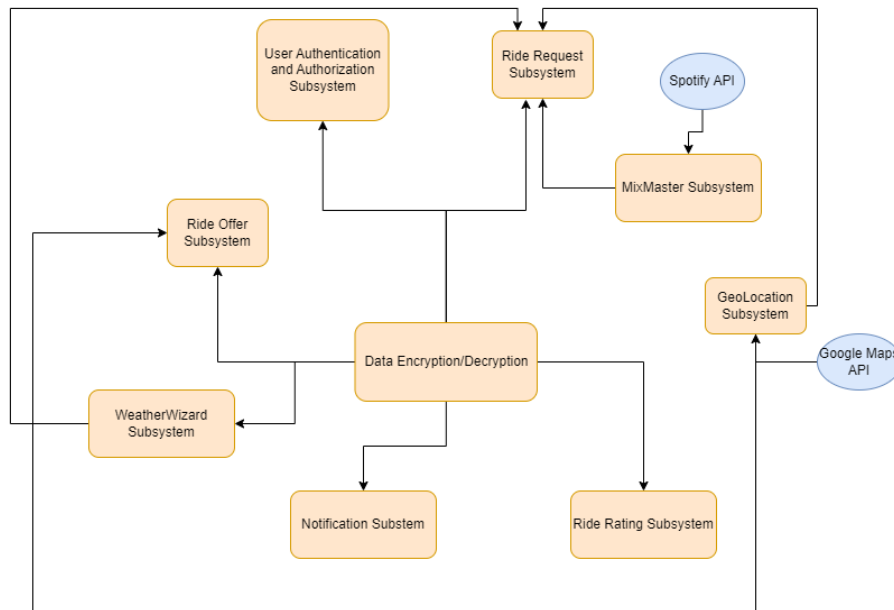
WeatherWizard Subsystem

The WeatherWizard subsystem allows for users to control the air conditioning unit of the auto-driving taxi from within the application. It takes in requests that are provided by the user on the WeatherWizard feature of the application and validates the request against certain requirements (i.e. within allowable range of temperatures). If valid, the request is then fulfilled by the vehicle toggling the air conditioning unit accordingly. The subsystem relies on the air conditioning technology in the self-driven vehicles, along with the data encryption/decryption subsystem.

MixMaster Subsystem

The MixMaster subsystem is responsible for providing functionality for the MixMaster feature of our application. This allows users to request songs to be played throughout the duration of their ride. The subsystem relies on the Spotify API in order to validate song requests, and must integrate with the speaker system in the dispatched vehicles in order to sync the queued songs to the music that is playing during the ride.

The relationship between the subsystems are defined in the following diagram:



4 Class Responsibility Collaboration (CRC) Cards

This section should contain all of your CRC cards.

- Provide a CRC Card for each identified class
- Please use the format outlined in tutorial, i.e.,

Class Name: Customer Information DB	
Responsibility:	Collaborators:
Accomplish the following two tasks in one atomic step: (1) verify the nonexistence of supplied email and (2) if successful, insert the new email, password, name, and date-of-birth, into the database.	Session Controller
Accomplish the following two tasks in one atomic step: (1) verify the existence of a supplied email and (2) if successful, delete the corresponding email, password, name, and date-of-birth from the database.	Session Controller
Retrieve customer information from username	Session Controller
Edit customer information from username	Session Controller

Class Name: Taxi DB	
Responsibility:	Collaborators:
Accomplish the following two tasks in one atomic step: (1) verify the nonexistence of a Taxi ID (QR code) and (2) if successful, insert the new Taxi ID(QR Code), Taxi driver, and Taxi capacity into the database.	Session Controller
Accomplish the following two tasks in one atomic step: (1) verify the existence of a supplied Taxi ID and (2) if successful, delete the Taxi ID from the database.	Session Controller
Retrieve Taxi ID from username	Session Controller

Class Name: Current Rides DB	
Responsibility:	Collaborators:
Accomplish the following two tasks in one atomic step: (1) verify the nonexistence of a current ride ID and (2) if successful, insert the new ride ID and related customer information and Taxi ID into the database.	Session Controller
Remove completed rides from the database	Session Controller

Class Name
Responsibi
Scans and de
Handles click

Class Name: Offers DB	
Responsibility:	Collaborators:
Accomplish the following two tasks in one atomic step: (1) verify the nonexistence of an offer ID and (2) if successful, insert the new offer ID and corresponding route and offering customer data into the database.	Session Controller
Remove accepted or expired offers from database	Session Controller

Class Name: Session Controller	
Responsibility:	Collaborators:
Know Customer Information DB	
Know Taxi DB	
Know Requests DB	
Know Offers DB	
Know Encryption Controller	
Handle the customer registration request (creates a customer with the desired email, password, name, age, date of birth)	Account Management Controller
Handle the customer login request (verify that the email/password pair exists)	Account Management Controller
Handle the customer edits request (edits existing customer with the desired password, name, age, date of birth)	Account Management Controller
Handle the customer delete request (deletes customer from DB)	Account Management Controller

Class Name: Encryption Controller	
Responsibility:	Collaborators:
Retrieve data from Session Controller, encrypt data, and send to Session Controller	Session Controller
Retrieve data from Session Controller, decrypt data, and send to Session Controller	Session Controller
Know Session Controller	

Class Name: Account Management Controller	
Send customer registration information to Session Controller	Session Controller
Retrieves and handles customer registration validation result from Session Controller	Session Controller Home Page Registration Page
Send customer login information to Session Controller	Session Controller
Retrieves and handles customer login validation result from Session Controller	Session Controller Home Page Login Page
Send customer edits information to Session Controller	
Retrieves and handles customer edits validation result from Session Controller	Session Controller Manage Account Page
Send customer delete notification to Session Controller	Session Controller
Retrieves and handles customer delete notification result from Session Controller	Session Controller Welcome Page Manage Account Page
Know Welcome Page	
Know Welcome Page	
Know (or create) Login Page	
Know (or create) Registration Page	
Know Account Management Page	
Know Home Page	
Know Session Controller	

Class Name: Manage Account Page	
Handles toggle-click event of 'Edit'	Account Management Controller
Know password	
Know name	
Know DOB	
Handles button-click event of 'Confirm Edits'	Account Management Controller
Handles button-click event of 'Delete Account'	Account Management Controller
Knows Account Management Controller	

Class Name: Login Page	
Responsibility:	Collaborators:
Know email	
Know password	
Handles button-click event of 'Login'	Account Management Controller
Knows Account Management Controller	

Class Name: Registration Page	
Responsibility:	Collaborators:
Know email	
Know password	
Know name	
Know DOB	
Handles button-click event of 'Signup'	Account Management Controller
Knows Account Management Controller	

Class Name: Welcome Page	
Responsibility:	Collaborators:
Handles button-click event of 'Login'	Account Management Controller
Handles button-click event of 'Signup'	Account Management Controller
Knows Account Management Controller	

Class Name: Login Page	
Responsibility:	Collaborators:
Know email	
Know password	
Handles button-click event of 'Login'	Account Management Controller
Knows Account Management Controller	

Class Name: Registration Page	
Responsibility:	Collaborators:
Know email	
Know password	
Know name	
Know DOB	
Handles button-click event of 'Signup'	Account Management Controller
Knows Account Management Controller	

Class Name: Welcome Page	
Responsibility:	Collaborators:
Handles button-click event of 'Login'	Account Management Controller
Handles button-click event of 'Signup'	Account Management Controller
Knows Account Management Controller	

Class Name: Session Controller	
Responsibility:	Collaborators:
Know Registration Controller	Registration Controller
Know Account Management Controller	Account Management Controller
Know Pickup Controller	Pickup Controller
Know Drop-Off Controller	Drop-off Controller
Know Encryption Controller	Encryption Controller
Know Dispatcher Controller	Dispatcher Controller
Know Login Page	Login Page
Know Logout Page	Logout Page
Sends registration status to Registration Controller	Registration Controller
Receives registration status from Encryption Controller	Encryption Controller
Receives customer information from Registration Controller	Registration Controller
Sends customer information to Encryption Controller	Encryption Controller
Receives customer information from Encryption Controller	Encryption Controller
Sends customer information to Account Management Controller	Account Management Controller
Receives edited customer information from Account Management Controller	Account Management Controller
Sends edited customer information to Encryption Controller	Encryption Controller
Receives deleted customer information from Account Management Controller	Account Management Controller
Sends deleted customer information to Encryption Controller	Encryption Controller
Receives ride request information from Dispatcher Controller	Dispatcher Controller
Sends ride request information to Encryption Controller	Encryption Controller
Receives potential offer list from Encryption Controller	Encryption Controller
Sends potential offer list to Dispatcher Controller	Dispatcher Controller
Receives offer information from Dispatcher Controller	Dispatcher Controller
Sends offer information to Encryption Controller	Encryption Controller
Receives potential offer list from Encryption Controller	Encryption Controller
Sends potential offer list to Dispatcher Controller	Dispatcher Controller
Receives offer confirmation from Dispatcher Controller	Dispatcher Controller
Sends offer confirmation start-ride notification to Pickup Controller	Pickup Controller
Sends offer confirmation to Encryption Controller	Encryption Controller
Receives ride-end notification from Dropoff Controller	Dropoff Controller
Sends ride-end notification to Encryption Controller	Encryption Controller
Handles Login Request	Customer Information DB
Handles Logout Request	Customer Information DB

Class Name: Dispatcher Controller	
Responsibility:	Collaborators:
Know Request Ride Controller	Request Ride Controller
Know Communicate Offer to Offerer Controller	Communicate Offer to Offerer Controller
Know Offer Ride Controller	Offer Ride Controller
Know Session Controller	Session Controller
Receives ride request information from Request Ride Controller	Request Ride Controller
Sends ride request information to Session Controller	Session Controller
Receives potential offer list from Session Controller	Session Controller
Sends potential offer list to Request Ride Controller	Request Ride Controller
Receives offer selection from Request Ride Controller	Request Ride Controller
Determines optimal offerer-requester pairing using the offer selection	
Sends offer selection and optimization information to Communicate Offer to Offerer Controller	Communicate Offer to Offerer Controller
Receives offer confirmation from Communicate Offer to Offerer Controller	Communicate Offer to Offerer Controller
Receives offer denial from Communicate Offer to Offerer Controller	Communicate Offer to Offerer Controller
Sends offer confirmation to Session Controller	Session Controller
Sends offer denial to Session Controller	Session Controller

Class Name: Main Registration Page	
Responsibility:	Collaborators:
Create User Account	Registration Controller

Class Name: Registration Error Page	
Responsibility:	Collaborators:
Generate Error Object with error message	Registration Controller

Class Name: Registration Success Page	
Responsibility:	Collaborators:
Generate Success Object with Success Message	Registration Controller

Class Name: Registration Controller	
Responsibility:	Collaborators:
Know Main Registration Page	Main Registration Page
Create Registration Success Page	Registration Success Page
Create Registration Error Page	Registration Error Page
Handle requests to create a user account	Main Registration Page
Sends customer information (username, password, name, email, age, and address) from the Main Registration Page to the Session Controller	Main Registration Page
	Session Controller
Receives status of registration from the Session Controller and determines the validity of login information	Session Controller

Class Name: View Main Account Page	
Responsibility:	Collaborators:
View All Account Data	Account Management Controller

Class Name: Delete Account Page	
Responsibility:	Collaborators:
Delete a Users Account and all associated credentials	Account Management Controller

Class Name: Edit Account Page	
Responsibility:	Collaborators:
Handle user requests to edit profile data	Account Management Controller

Class Name: Account Management Controller	
Responsibility:	Collaborators:
Know View Main Account Page	View Main Account Page
Know Edit Account Page	Edit Account Page
Know Delete Account Page	Delete Account Page
Handle requests to edit a user account	Edit Account Page
Handle requests to delete a user account	Delete Account Page
Send edited user data to the Session Controller from the Edit Account Page	Edit Account Page Session Controller.
Receives customer information from Session Controller and displays information to View Main Account Page	View Main Account Page Session Controller.

Class Name: Search Ride Page	
Responsibility:	Collaborators:
Allow User to enter and submit all ride details: starting location, ending location, number of passengers, intended pickup time, and if they want MixMaster and WeatherWizard enabled	Request Ride Controller

Class Name: Potential Offers Interface	
Responsibility:	Collaborators:
Display potential ride offers to the user	Request Ride Controller

Class Name: Confirm Ride Page	
Responsibility:	Collaborators:
Allows the user to select and confirm a ride	Request Ride Controller

Class Name: Request Ride Controller	
Responsibility:	Collaborators:
Know Search Ride Page	Search Ride Page
Know Potential Offers Interface	Potential Offers Interface
Know Confirm Ride Page	Confirm Ride page
Receives a response from search ride page and relays it to the dispatcher	Dispatcher controller Search Ride Page
Receives a response from the dispatcher and relays it to the potential offers interface	Dispatcher Controller Potential Offers Interface
Receives a response from Confirm Ride Page of the confirmed ride and relays it to Routes Controller and Dispatcher controller	Confirm Ride Page Route Controller Dispatcher Controller

Class Name: Show Route Page		
Responsibility:	Collaborators:	
Displays route in text and on a map to user	Routes Controller	

Class Name: Google Maps Gateway		
Responsibility:	Collaborators:	
Sends requests and receives route data from Google Maps Direction API	Routes Controller	
Creates Map graphic from direction data		

Class Name: Routes Controller		
Responsibility:	Collaborators:	
Generates requests for routes from one location to another	Request Ride Controller	

Class Name: Pick Up Controller		
Responsibility:	Collaborators:	
Signals to Notify Customer Page when the user should be notified of Taxi arrival	Notify Customer Page	
Enables use of MixMaster	MixMaster	
Enables use of WeatherWizard Controller	WeatherWizard Controller	
Receives notification from Session Controller to notify customer their ride has arrived	Session Controller	

Class Name: Notify Customer Page		
Responsibility:	Collaborators:	
Notifies the user when their taxi has arrived	Pick Up Controller	

Class Name: View Current Fare Page		
Responsibility:	Collaborators:	
Displays the current fare for a single user for the duration of their ride	Fare Controller	

Class Name: Fare Controller		
Responsibility:	Collaborators:	
Returns the current fare at any point in time for the ride	View Current Fare Page	

Class Name: Drop off controller		
Responsibility:	Collaborators:	
Requests Customer Payment		
Enables Rating Controller	Ratings Controller	
Sends ride end notification to session controller when ride is done	Session Controller	

Class Name: Ratings Controller		
Responsibility:	Collaborators:	
Sends out requests for ratings to each customer for each other customer	Rating Form Page	
	Drop Off Controller	

Class Name: View Rating Page		
Responsibility:	Collaborators:	
Allows users to browse their own ratings	Ratings Controller	

Class Name: Rating Form Page	
Responsibility:	Collaborators:
Allows User to submit a rating for another user	Ratings Controller

Class Name: WeatherWizard Controller	
Responsibility:	Collaborators:
Raise the Car's temperature	Edit Temperature Page
Lower the Car's temperature	Edit Temperature Page
Open the Car's Window	Open and close Window Page
Close the Car's Window	Open and close Window Page

Class Name: Edit Temperature Page	
Responsibility:	Collaborators:
Allows the user to submit a request to raise the car's temperature	Weather Wizard
Allows the user to submit a request to lower the car's temperature	Weather Wizard

Class Name: Open and close Window Page	
Responsibility:	Collaborators:
Allows the user to submit a request to open the car's window	Weather Wizard
Allows the user to submit a request to close the car's window	Weather Wizard

Class Name: Spotify Gateway	
Responsibility:	Collaborators:
Returns song name, artist, and playable file from query	MixMaster

Class Name: View Songs Interface	
Responsibility:	Collaborators:
Allows the user to view songs in the playlist	MixMaster

Class Name: Search Songs Page	
Responsibility:	Collaborators:
Allows the user to search for songs by title and artist name	MixMaster

Class Name: Manage Playlist Page	
Responsibility:	Collaborators:
Allows the user to add songs to the playlist	MixMaster
Allows the user to remove songs from the playlist	MixMaster
Allows the user to re-arrange songs in the playlist	MixMaster

Class Name: MixMaster	
Responsibility:	Collaborators:
Facilitates communication between Search songs page and Manage Playlist Page	Search Songs Page
	Manage Playlist Page

Class Name: Decline request page	
Responsibility:	Collaborators:
Allows the user to decline a request, Handles click-event of "Decline request"	Communicate Offer to Offerer Controller

Class Name: Confirm Request Page	
Responsibility:	Collaborators:
Handles click-event of "confirm request"	Communicate offer to offerer Controller

Class Name: View Request Interface	
Responsibility:	Collaborators:
Displays all requests to the offerer	Communicate Offer to offerer Controller

Class Name: Communicate Offer to Offerer Controller	
Responsibility:	Collaborators:
Communicates offers form dispatcher to Offerer	Dispatcher Controller View request Interface Confirm Request Page Decline Request Page

Class Name: Confirm Offer Page	
Responsibility:	Collaborators:
Handles click-event for offerer 'confirm offer'	offer ride controller

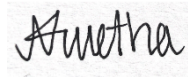
5 Division of Labour

Include a Division of Labour sheet that indicates the contributions of each team member. This sheet must be signed by all team members.

- Q3: Moamen

A handwritten signature in black ink on a light yellow rectangular background.

- Q1, 2, 4: Nivetha

A handwritten signature in black ink on a light grey rectangular background.

- Q3: Jinal

A handwritten signature in black ink.

- Q1, 2, 4: Mya

A handwritten signature in black ink.

- Q1, 2, 4: Aadil

A handwritten signature in black ink.

- Overall, the group had split into two groups to work on this deliverable. Aadil, Mya, and Nivetha all collaborated and met together to work on Q1, Q2 and Q4. Moamen and Jinal collaborated and worked on Q3 and added in input throughout Q4.