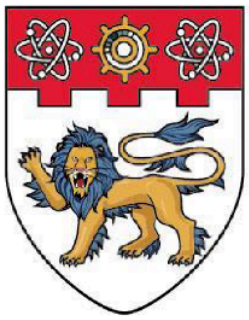


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC2006 Software Engineering

Tutorial Group SCSD

Lab Deliverables

Group Members	Matric Number
Lim Li Ping Joey	U2321331C
Roger Kwek Zong Heng	U2320447G
Matz Chan	U2321258E
Ong Hong Xun	U2321248G
Wan Li Xin Yuan	U2321934B
Cheah Wei Jun	U2320561K

Table of Contents

Table of Contents **2**
1. Introduction 3
    1.1. Purpose 3
    1.2. Product Scope 3
    1.3. References 3
    - GitHub: https://github.com/softwarelab3/2006-SCSD-C2.git 3
2. Overall Description 4
    2.1. Product Perspective 4
    2.2. Product Functions
    Bussin Buses provides the following core functionalities: 4
    2.3. User Classes and Characteristics 5
    2.4. Operating Environment 5
    2.5. Design and Implementation Constraints 5
    2.6. User Documentation 6
    2.7. Assumptions 6
3. Functional Requirements 7
4. Other Non-functional Requirements 10
5. Use Case Model 12
6. Use Case Descriptions 13
    6.1 Account Management 13
    6.2 Booking Management 17
    6.3 Bus Schedule Management 21
    6.4 Bus Management 26
    6.5 Route Recommendations 32
7. Data Dictionary 33
8. UI Mockup 36
9. Class Diagram 40
10. Key boundary classes and control classes 40
11. Sequence Diagrams of Some Use Cases 41
12. Initial Dialog Map 47

1. Introduction

1.1. Purpose

Bussin Buses is a mobile application designed to provide a seamless and user-friendly platform for NTU students who reside far from campus. The primary purpose of Bussin Buses is to provide a faster and more convenient transportation option by providing a direct shuttle service that picks up students from NTU and drops them off at their destinations without making intermediate stops. The application provides a seamless and effective commuting experience by integrating real-time tracking, seat booking and optimised routes. The app will be a useful tool for all NTU students looking for a faster and more reliable transport option.

1.2. Product Scope

Bussin Buses helps connect NTU students with a faster and more efficient commuting option while offering a range of supportive features for hassle-free travel. The app utilises GPS technology to easily track bus locations in real-time and book seats in advance, ensuring a smooth and predictable journey. The app also uses API such as OpenStreet that allows personalised route recommendations and traffic updates, helping drivers to plan their route. In addition, the app offers functions to check bus schedules, manage bookings and receive notifications for upcoming booking or traffic delays, making it a comprehensive resource for NTU students looking for a more convenient transportation option.

1.3. References

- GitHub: https://github.com/softwarelab3/2006-SCSD-C2.git

2. Overall Description

2.1. Product Perspective

Bussin Buses is a new, self-contained product designed to facilitate seat reservations for scheduled buses while providing real-time tracking, optimised route recommendations and traffic updates. This system is not a bus dispatching system but focuses on allowing Users to book specific seats on pre-scheduled buses.

The system interacts with:

A. Supabase - To handle User Registration, Login and Password Recovery.

2.2. Product Functions

Bussin Buses provides the following core functionalities:

A. User Account Management
    a. Register, Login, and Reset Password

B. Seat Booking and Cancellation
    a. Search for available buses
    b. Book and Cancel a seat on a scheduled bus
    c. View upcoming bookings

C. Real-time bus tracking
    a. Track live locations of buses
    b. View estimated arrival time

D. Route and Traffic Optimisation
    a. Provided optimised route for drivers based on traffic conditions
    b. Update estimated arrival time dynamically.

E. Driver Operations
    a. Manage start/end time of bus journey
    b. Manage bus schedule
    c. View commuters list for clarification
    d. Update any delay for traffic

F. Notifications and Alerts

2.3. User Classes and Characteristics

NTU Students

- Primarily use the app for transportation-related features such as bus tracking and seat booking.
- Authenticated with their login credentials to access the features.
- Need real-time information on bus locations and estimated arrival times to plan their commute.
- May prefer mobile-friendly, fast and easy-to-navigate interfaces as they are likely to use the app while on-the-go.

Bus Drivers

- Responsible for driving the bus on optimised routes.
- Require real-time communication tools to receive instructions about route changes or any potential incident.
- May need a simpler, clear interface to monitor student pickup and drop-off locations.

2.4. Operating Environment

The Bussin Buses app operates on smartphones, requiring an internet connection for accessing real-time bus data, location services and seat booking features. The app is compatible with <Android version 8.0 and above>. To ensure smooth functionality, users must have sufficient device storage, an active internet connection and location services enabled for accurate bus tracking and route optimization.

2.5. Design and Implementation Constraints

1. Bussin Buses requires a stable internet connection to access its functionalities as it depends on real-time data and location-based services for route planning and tracking.
2. The app utilises third-party APIs such as OpenStreet API for location services. Any changes or deprecation in these APIs may impact the app's performance or availability of certain features.
3. Bussin Buses must be installed on devices to function properly.

2.6. User Documentation

A demonstration of how the app works along with a guide to install and run the app is provided in the README.md file on the GitHub repository which contains the source code for this app (see Section 1.3 References).

2.7. Assumptions

1. Bussin Buses relies on external APIs to provide relevant real-time data and services.
2. The data retrieved from external APIs is assumed to be accurate, up-to-date and reliable.
3. In cases where the API services become unavailable, the app may not meet some requirements specified in this document.
4. Bussin Buses assumes that users have access to a smartphone with an active internet connection which is essential for the app's functionality.
5. It is assumed that users have a basic understanding of how to navigate mobile applications and can use app's features without requiring extensive technical assistance.

3. Functional Requirements

1. The booking system shall allow Users to manage their individual accounts through Supabase Authentication.

    1.1. Users must consist of Commuters and Drivers.
    1.2. The system shall allow Users to register their account in the server.
    1.3. The system shall allow Users to login into their account.
    1.4. The system shall allow Users to reset their password.
        1.4.1. For first-time users, Users can reset their password from the default password.
        1.4.2. If the user forgot their password, Users can reset their password.

2. The booking system shall allow Commuters to manage seat bookings on buses.

    2.1. The booking system shall be able to search for bus schedules

        2.1.1. The system shall display available buses based on date, time, pre-assigned pickup, and pre-assigned drop-off locations.

    2.2. The booking system shall allow Commuters to book a seat for an available bus.

        2.2.1. The system shall display a "Book Seat" button for available buses.
        2.2.2. The system will reserve a seat for the Commuter.
            2.2.2.1. If there are clashes of requests for the bus seat, the first request shall be processed and the rest shall be rejected.

    2.3. The booking system shall allow Commuters to cancel their booking

        2.3.1. The system shall allow commuters to cancel their booking up to 30 minutes before departure.
        2.3.2. Upon cancellation, the system shall update seat availability in real-time.

    2.4. Commuters shall be able to view their upcoming bookings.

        2.4.1. The system shall display all confirmed bookings with bus details, seat number, and timings.

3. The system shall allow Commuters to see and track their bus bookings.

    3.1. The system shall display individual bus schedule which is managed by Bus Driver

3.1.1. Commuters shall be able to view their desired bus timings and locations shown in the bus schedule.
3.1.2. Commuters shall be able to search for their desired bus schedule.

3.2. The system shall display the live information of the booked bus.
3.2.1. If the Commuters checked into the bus, they would be able to get live updates about their current bus status

3.3. The system shall estimate and display arrival time at each stop.
3.4. The system shall allow students to check into the bus upon arrival.

4. The system shall allow Bus Drivers to manage bus operations.

3.1. Bus Drivers shall be able to view their assigned schedules.

- The system shall display a calendar view of driver shifts.
- Each shift shall include preassigned pickup and drop-off locations.

3.2. Bus Drivers shall be able to manage bus schedules.

- Drivers shall be able to start and end their journey.
- Drivers shall update their real-time location throughout the journey.

3.3. Drivers shall be able to view their Commuter list.

- The system shall display the list of Commuters who booked a seat.
- Drivers shall confirm attendance when commuters board the bus.

3.4. The system shall display the Bus Driver’s schedule and shift information.

- The system shall display a calendar with the timings of the Bus Driver’s shifts
- Each shift shall include information about the assigned pickup and drop-off locations

4. The system shall provide optimized routes and traffic updates to Bus Drivers.

4.1. The system shall recommend the most efficient route for a given trip.
- The system shall optimize the route based on real-time traffic data using APIs (e.g., OpenStreet API).

4.2. The system shall provide real-time traffic updates to drivers.
- Drivers shall receive estimated arrival times for drop-off points.
- The system shall notify drivers of traffic incidents or delays.

4.3. The system shall allow drivers to report incidents or delays.
- Drivers shall be able to send delay notifications to the system.

- The system shall update arrival estimates accordingly.

5. The system shall allow Supabase Authentication to manage User access.

    5.1. The system shall validate User login credentials.
    5.2. The system shall handle password recovery requests.

6. The system shall assist with bus routing.

    6.1. The system shall analyze traffic conditions and recommend optimal routes.
    6.2. The system shall update estimated arrival times dynamically.

7. The system shall provide real-time traffic updates.

    7.1. The system shall track live traffic conditions and notify drivers.
    7.2. The system shall receive real-time traffic updates to recommend a route.

4. Other Non-functional Requirements

1. Usability

    1.1. The booking system should be clear and organised for users to easily navigate around (User shall be able to book a bus seating within 5 clicks .
    1.2. The UI elements (buttons, icons, colours, fonts) should be consistent and easy to read and follow
    1.3. The booking system should personalize the user experience by remembering their preferences and choices

2. Reliability

    2.1. Ensure that the real-time traffic data is accurate to provide real-time travel estimation. (System shall be able to retrieve updated traffic information 3 seconds after being queried.)
    2.2. User informations should be safe and secure (password, credit card information, etc)
    2.3. The system should have minimal downtime unless notified

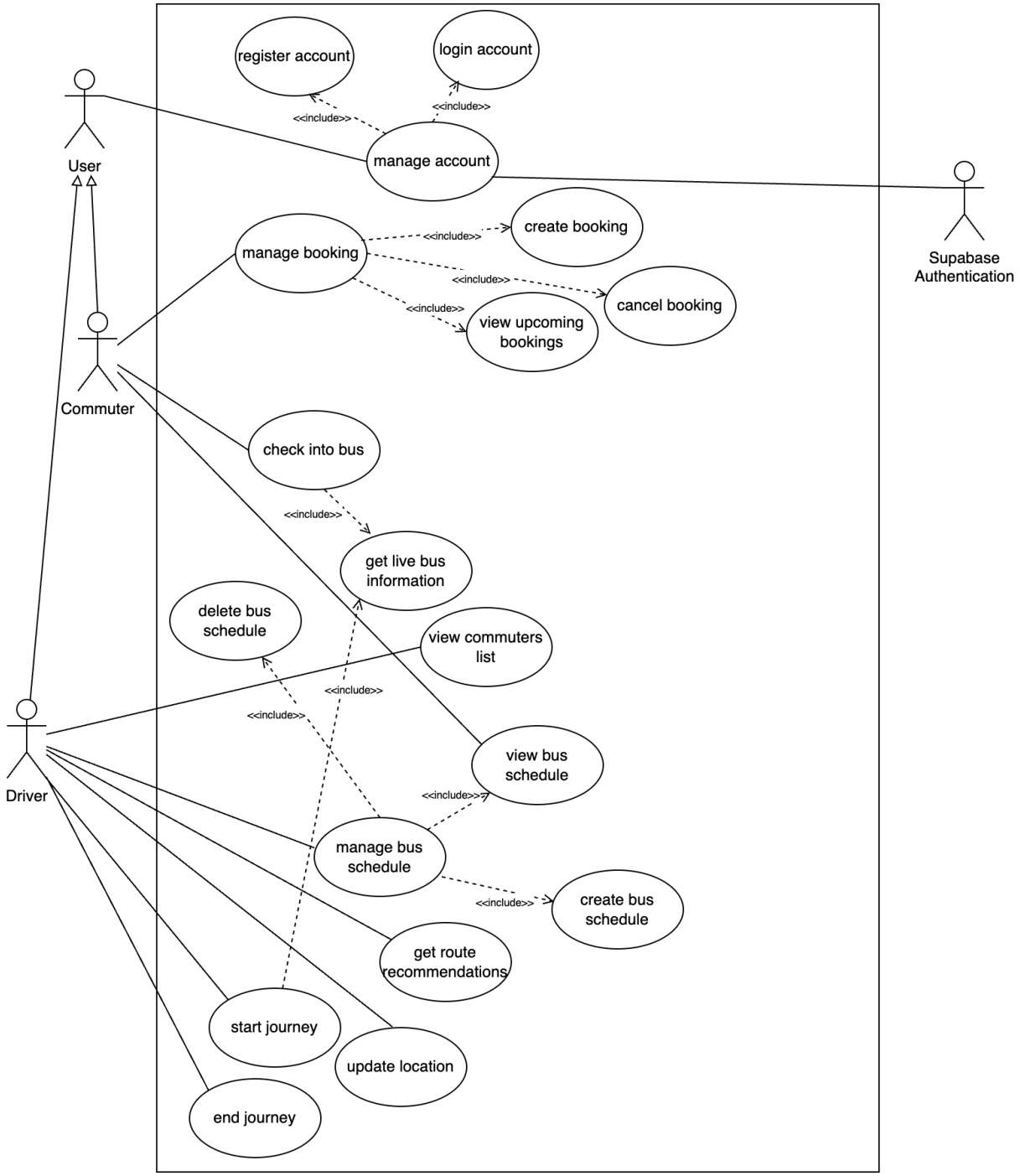
3. Performance

    3.1. The booking system should be able to handle multiple users and bookings without much significant performance degradation. (Information will be displayed within 3 seconds after user queried)
    3.2. The booking system needs to be able to handle multiple bookings while keeping real time update of the seats availability
    3.3. The GPS system needs to be accurate in tracking the bus locations to notify the users when the bus is arriving

4. Supportability

    4.1. The code need to be well documented and easy to maintain
    4.2. Documentation of the booking system should be done

5. Use Case Model



6. Use Case Descriptions

6.1 Account Management

Use Case ID:	1.1		
Use Case Name:	Manage Account		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	User, Supabase Authentication
Description:	The “Manage Account” use case is essential for identifying users as commuters or drivers. Without it, users cannot access or use any system features. The use case includes registering an account, logging into an account, and resetting an account’s password.
Preconditions:	1. The system must be operational 2. Supabase Authentication must be operational 3. Users must have access to the client
Postconditions:	1. Users are authenticated 2. Users’ role must be known (commuter/driver)
Priority:	High
Frequency of Use:	Low
Flow of Events:	1. The user selects the desired action (register account / login) 2. The system executes the desired action
Alternative Flows:	
Exceptions:	1.1.EX.1 Supabase Authentication Unavailable 1. The request to Supabase Authentication times out 2. The system informs the user and tells them to try again later
Includes:	Register Account, Login Account
Special Requirements:	User must receive feedback from the client regardless of their response status
Assumptions:	1. Supabase Authentication will be a third-party service 2. There will be a user interface that brings them to select their account management action
Notes and Issues:	

Use Case ID:	1.2

Use Case Name:	Register Account		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	User, Supabase Authentication
Description:	The “Register Account” use case is used to issue a user an identity so that we know their role in the system and we can attach a unique identifier to them to store their specific data.
Preconditions:	1. The system must be operational 2. Supabase Authentication must be operational 3. Users must have access to the client
Postconditions:	1. Supabase Authentication will have a new record for the newly registered user
Priority:	High
Frequency of Use:	Low
Flow of Events:	1. The user clicks on the “Register an account” button to bring up the registration form 2. The user enters their name, email, password, confirm password 3. The user clicks on the “Register” button to confirm their registration
Alternative Flows:	
Exceptions:	1.2.EX.1 Password and Confirm Password fields do not match 1. User clicks on “Register button” 2. Client verifies if passwords match 3. Client feedbacks to user that passwords do not match
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	1.3

Use Case Name:	Login Account		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	User, Supabase Authentication
Description:	The “Login Account” use case is to authenticate the user so that they can access their user-specific data in the system
Preconditions:	1. User has an existing account recorded in Supabase Authentication 2. The system must be operational 3. Supabase Authentication must be operational 4. Users must have access to the client
Postconditions:	1. User will be considered logged in on Supabase Authentication
Priority:	High
Frequency of Use:	Low
Flow of Events:	1. The user clicks on the “Login” button to bring up the Login form 2. The user enters their email and password 3. The user clicks on the “Login” button
Alternative Flows:	
Exceptions:	1.3.EX.1 User not found in Supabase Authentication 1. The user click “Login” 2. The client verifies with Supabase Authentication 3. Supabase Authentication returns “User not found” 4. The client informs user of the exception
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

6.2 Booking Management

Use Case ID:	2.1		
Use Case Name:	Manage Booking		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	Commuter, Database
Description:	The “Manage Booking” use case is essential for allowing commuters to create, cancel, and view their upcoming bookings. It ensures that commuters can manage their trip plans and have access to relevant information about their upcoming trips.
Preconditions:	1. The user must be logged in as a commuter in the system 2. The system must have available bus schedules for booking 3. The Database must be operational
Postconditions:	1. Database is updated accordingly 2. The data present in the client is up-to-date on the modified details
Priority:	High
Frequency of Use:	Moderate to High, depending on commuter’s travel frequency
Flow of Events:	1. The commuter navigates to the booking management section in the client 2. The commuter initiates the desired action 3. The system executes the desired action
Alternative Flows:	
Exceptions:	
Includes:	Create Booking, Cancel Booking, View Upcoming Bookings
Special Requirements:	
Assumptions:	Bookings cannot be updated. They must be cancelled and re-created.
Notes and Issues:	

Use Case ID:	2.2		
Use Case Name:	Create Booking		
Created By:	Lim Li Ping Joey	Last Updated By:	Ong Hong Xun
Date Created:	31 Jan 2025	Date Last Updated:	7 Feb 2025

Actor:	Commuter, Database
Description:	The “Create Booking” use case is for commuters to book a seat on a bus during a scheduled timing
Preconditions:	1. The user must be logged in as a commuter in the system 2. The system must have available bus schedules for booking 3. The Database must be operational
Postconditions:	1. Database is updated accordingly 2. The data on the client displays the newly created booking
Priority:	High
Frequency of Use:	Moderate to High, depending on commuter’s travel frequency
Flow of Events:	1. The commuter navigates to bus schedules section in the client 2. The commuter clicks on an available schedule that they want to book 3. The commuter clicks on the “Book Seat” button 4. The commuter chooses a seat to book on the bus 5. The commuter confirms their selection 6. The client feedbacks that the commuter’s booking is successful
Alternative Flows:	AF-S3 Commuters Booking the Same Seat at the Same Time (race condition) 1. The system confirms the first request 2. The system rejects the subsequent requests 3. The system sends an error message to the rejected commuters 4. The client displays the error message 5. The client reflects the booked seat on the interface
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	2.3		
Use Case Name:	Cancel Booking		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	Commuter, Database
Description:	The “Cancel Booking” use case is for commuters to cancel a booked seat on a bus for a scheduled timing
Preconditions:	1. The user must be logged in as a commuter in the system 2. The booking already exists 3. The Database must be operational
Postconditions:	3. Database is updated accordingly 4. The data on the client removes the booking from the interface
Priority:	High
Frequency of Use:	Low to Moderate
Flow of Events:	1. The commuter navigates to the booking management section in the client 2. The commuter clicks the “Cancel” button on the booking to be cancelled 3. The client displays a confirmation message to confirm the cancellation 4. The booking is removed from the user interface
Alternative Flows:	
Exceptions:	2.3.EX.1 Commuter tries to cancel 30 minutes or less before departure 1. The commuter navigates to the booking management section in the client 2. The “Cancel” button is greyed out
Includes:	
Special Requirements:	
Assumptions:	Commuter cannot cancel booking 30 minutes before the departure time
Notes and Issues:	

Use Case ID:	2.4		
Use Case Name:	View Upcoming Bookings		
Created By:	Lim Li Ping Joey	Last Updated By:	Ong Hong Xun
Date Created:	31 Jan 2025	Date Last Updated:	7 Feb 2025

Actor:	Commuter, Database
Description:	The “View Upcoming Bookings” use case is for commuters to check which seats they booked and for which scheduled bus timing
Preconditions:	1. The user must be logged in as a commuter in the system 2. The booking(s) already exist 3. The Database must be operational
Postconditions:	
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The commuter navigates to the booking management section in the client 2a. If at least one booking exists: The table of bookings will be displayed in the booking management section 2b. If no booking exists: The booking management section will display a “No booking(s) found” message
Alternative Flows:	AF-S2a. Commuter has upcoming booking(s) today 1. Commuter opens the app 2. The client displays the home page 3. The client displays today’s upcoming booking(s)
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

6.3 Bus Schedule Management

Use Case ID:	3.1		
Use Case Name:	Manage Bus Schedule		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	Driver, Database
Description:	The “Manage Bus Schedule” use case is essential for allowing drivers to create, delete, and view their bus schedule. It ensures that drivers can manage their schedules and have access to relevant information about their upcoming shifts.
Preconditions:	1. The user must be logged in as a driver in the system 2. The Database must be operational
Postconditions:	1. Database is updated accordingly 2. The data present in the client is up-to-date on the modified details
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The driver navigates to the bus schedule management section in the client 2. The driver initiates the desired action 3. The system executes the desired action
Alternative Flows:	
Exceptions:	
Includes:	View Bus Schedule, Create Bus Schedule, Delete Bus Schedule
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	3.2		
Use Case Name:	View Bus Schedule		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	Driver, Commuter, Database
Description:	The “View Bus Schedule” use case is for drivers and commuters to view all scheduled bus rides available in the system. This is important for the driver not to clash with other drivers, and important for commuters to know when they can book a seat.
Preconditions:	1. The user must be logged in either as driver or commuter in the system 2. The database must be operational
Postconditions:	
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The driver or commuter opens the client 2. The driver or commuter navigates to the bus schedules screen 3. The system displays all the bus schedules with the schedule information.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	3.3		
Use Case Name:	Create Bus Schedule		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	Driver, Database
Description:	The “Create Bus Schedule” use case is for drivers to indicate to commuters when they are available. It also provides commuters details about the seats available, pickup point, drop off point, and departure time of the scheduled ride.
Preconditions:	1. The user must be logged in as driver in the system 2. The database must be operational
Postconditions:	1. Database is updated accordingly 2. The data on the client displays the newly created bus schedule
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The driver navigates to the bus schedules screen 2. The driver clicks the “New Schedule” button 3. The driver enters information such as the pickup point, drop-off point, departure date and time 4. The driver clicks the “Create” button 5. The client reflects the newly created bus schedule in the user interface
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	All fields are required
Assumptions:	The layout of seats will be options provided by the system, not for the driver to manually upload
Notes and Issues:	

Use Case ID:	3.4		
Use Case Name:	Delete Bus Schedule		
Created By:	Lim Li Ping Joey	Last Updated By:	Ong Hong Xun
Date Created:	31 Jan 2025	Date Last Updated:	7 Feb 2025

Actor:	Driver, Commuter, Database
Description:	The “Delete Bus Schedule” use case is for drivers to cancel their scheduled bus ride slot, in case of sudden unavailability. Due to this deletion, commuters that booked the scheduled ride will be affected.
Preconditions:	1. The target bus schedule must exist in the system 2. The user must be logged in as driver in the system 3. The database must be operational
Postconditions:	1. Bookings associated with that bus schedule is deleted from the client and Database 1. The bus schedule is deleted from the client and Database
Priority:	High
Frequency of Use:	Low
Flow of Events:	1. The driver navigates to the screen displaying only his own bus schedules 2. The driver clicks on the “Delete” button on the schedule he wants to delete in the table of schedules 3. The system prompts the driver to ask for confirmation 4. The driver clicks “Yes” 5. The system notifies all commuters that booked this bus schedule of this deletion 6. The system deletes all bookings related to the bus schedule 7. The system deletes the bus schedule 8. The client informs the driver that the deletion has been completed
Alternative Flows:	AF-S2. Driver Denies Confirmation Prompt 1. The driver clicks on the “Delete” button on the schedule he wants to delete in the table of schedules 2. The system prompts the driver to ask for confirmation 3. The driver clicks “No” 4. The system returns to the previous screen without making any changes
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	3.5		
Use Case Name:	Search for Bus Schedule		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	1 Feb 2025

Actor:	Commuter, Database
Description:	The “Search Bus Schedule” use case is for commuters to quickly find the bus schedule they need.
Preconditions:	1. The user must be logged in as a commuter in the system 2. The Database must be operational
Postconditions:	
Priority:	Moderate
Frequency of Use:	High
Flow of Events:	1. The commuter filters the search by pickup point, drop-off point, departure date and time 2. The commuter clicks “Search” 3. The client displays bus schedules that match the filters
Alternative Flows:	AF-S2. No Exact Matching Schedule(s) 1. The commuter filters the search by pickup point, drop-off point, departure date and time 2. The commuter clicks “Search” 3. The client displays bus schedules that are similar to the filters
Exceptions:	3.5.EX.1 No Near Matches or Exact Matches Available 1. The commuter filters the search by pickup point, drop-off point, departure date and time 2. The commuter clicks “Search” 4. The client displays a message informing the commuter that there is lack of bus schedules matching their needs
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

6.4 Bus Management

Use Case ID:	4.1		
Use Case Name:	Get Live Bus Information		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	2 Feb 2025

Actor:	Commuter, Driver
Description:	The “Get Live Bus Information” use case is for commuters to track where they currently are during the trip
Preconditions:	1. The user must be logged in as a commuter in the system
Postconditions:	
Priority:	Low
Frequency of Use:	Moderate
Flow of Events:	1. The commuter checks into the bus 2. The driver starts the journey 3. The bus information is streamed from the driver 4. The commuter’s client displays all relevant information about the current bus ride
Alternative Flows:	
Exceptions:	4.1.EX.1 Commuter does not check in 1. The driver starts the journey 2. The commuter will be considered as “absent” on the ride and will not receive any live bus information
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	4.2		
Use Case Name:	Update Location		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	2 Feb 2025

Actor:	Driver
Description:	The “Update Location” use case is for drivers to update their locations for commuters to see the location of the bus relative to the pickup and drop-off points
Preconditions:	1. The user must be logged in as a commuter in the system 2. The driver must have started the journey
Postconditions:	
Priority:	Low
Frequency of Use:	Moderate
Flow of Events:	1. The driver starts the journey on the client 2. The client will retrieve the driver’s location 3. The client will broadcast the location to the commuters’ clients 4. Steps 2 and 3 will repeat every 10 seconds until the driver ends the journey on the client
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	4.3		
Use Case Name:	Check into Bus		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	2 Feb 2025

Actor:	Commuter, Database
Description:	The “Check into bus” use case is for commuters to confirm that they are on the bus before departure. By checking into the bus, the commuter will be able to subscribe to the live bus information
Preconditions:	1. The user must be logged in as a commuter in the system 2. The Database must be operational 3. It must be 15 minutes or less before departure time
Postconditions:	1. The Database is updated accordingly
Priority:	Moderate
Frequency of Use:	Moderate
Flow of Events:	1. The commuter opens the client 2. The commuter clicks the “check in” button 3. The client displays a confirmation message
Alternative Flows:	
Exceptions:	
Includes:	Get Live Bus Information
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	4.4		
Use Case Name:	Start Journey		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	2 Feb 2025

Actor:	Driver
Description:	The “Start Journey” use case is for drivers to broadcast to the commuters’ clients that it will be sending live bus information updates. It also starts the routing recommendation system.
Preconditions:	1. The user must be logged in as a driver in the system 2. It must be 5 minutes or less before departure time
Postconditions:	
Priority:	High
Frequency of Use:	Moderate
Flow of Events:	1. The driver opens the client 2. The driver clicks the “Start Journey” button 3. The client displays a screen for route recommendations
Alternative Flows:	
Exceptions:	
Includes:	Get Live Bus Information
Special Requirements:	
Assumptions:	The bus driver can start the journey from 5 minutes before departure time onwards
Notes and Issues:	

Use Case ID:	4.5		
Use Case Name:	End Journey		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	2 Feb 2025

Actor:	Driver
Description:	The “End Journey” use case is for drivers to end the trip and stop broadcasting the live bus information. It stops the route recommendations.
Preconditions:	1. The user must be logged in as a driver in the system
Postconditions:	
Priority:	High
Frequency of Use:	Moderate
Flow of Events:	1. The driver opens the client 2. The driver navigates to the route recommendations screen 3. The driver clicks on the “End Journey” button 4. The route recommendation screen is destroyed 5. The driver is brought to the Home Screen
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	The driver will be able to end the journey at any time, regardless if they are close in proximity to the dropoff point. This is to accommodate situations where a diversion of path is needed due to bus faults, etc.
Notes and Issues:	

Use Case ID:	4.6		
Use Case Name:	View Commuters List		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	2 Feb 2025

Actor:	Driver, Database
Description:	The “View Commuters List” allows the driver to see who booked the scheduled ride and who has checked in to gauge whether he can leave early or has to wait a bit longer.
Preconditions:	1. The user must be logged in as a driver in the system 2. The Database must be operational
Postconditions:	
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. The driver opens the client 2. The driver clicks on “View Commuters” button 3. A list of commuters who booked the scheduled time slot, along with their checked in status will be displayed
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

6.5 Route Recommendations

Use Case ID:	5.1		
Use Case Name:	Get Route Recommendations		
Created By:	Lim Li Ping Joey	Last Updated By:	Lim Li Ping Joey
Date Created:	31 Jan 2025	Date Last Updated:	10 Feb 2025

Actor:	Driver
Description:	The “Get Route Recommendations” use case is for the driver to get a suggested route to get to the dropoff point, based on their current location
Preconditions:	1. The user must be logged in as a driver in the system
Postconditions:	
Priority:	High
Frequency of Use:	Moderate
Flow of Events:	1. The driver starts the journey 2. The Route Recommendation screen shows up 3. The screen shows a recommended route to the driver
Alternative Flows:	AF-S1 The system detects a disruption and a new route was found 1. The system finds a new route 2. The new route is sent to the driver’s client 3. The route is updated on the driver’s client 4. The client informs the driver of the disruption and the reason for the rerouting
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	The driver only gets route recommendations once at the start, and if there are any subsequent disruptions to traffic
Notes and Issues:	

7. Data Dictionary

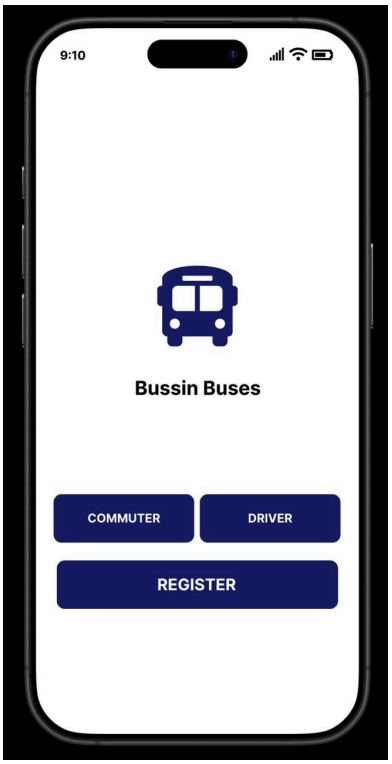
Term	Definition
Account	A registered user's personal profile associated with an application. It may include personal information, contact details, etc.
Application	A mobile application installed on users' smartphones that relies on an internet connection to fully utilize all of its intended features. The app may request certain system permissions to enable specific functionalities.
AuthenticationServer	A server responsible for managing user login credentials, verifying identity and handling password recovery requests.
Availability	Bus seats currently available for booking.
Booking	A confirmed reservation of the bus seat including details such as destination and time.
Bus	A transport vehicle that follows a fixed schedule and route to shuttle NTU students.
BusDriver	A designated individual responsible for operating the bus according to the assigned schedules and routes.
BusFare	The cost charged to users for booking a seat on a bus, based on factors such as route, distance, and demand.
BusSchedule	A predefined timetable specifying bus departure and arrival times along different routes.
Commuter	A NTU student using the application to book and track buses.

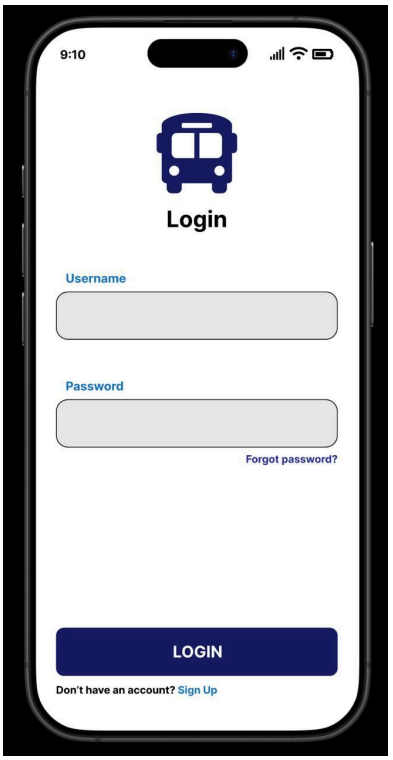
Credentials	Unique identifiers (e.g., username, email, password) used to verify a user's identity for login.
Database	Database refers to an organized collection of structured information, or data, typically stored electronically in a computer system.
Driver	The individual responsible for operating the bus along the assigned route and schedule
Destination	The final drop-off point for a bus journey as determined by a commuter’s booking.
Drop-off-Location	The designated place where a commuter exits the bus.
EstimatedArrivalTime	The estimated time for the bus to arrive at its drop off location.
GPS	GPS, or the Global Positioning System, is a global navigation satellite system that provides the bus location, speed and time synchronization.
LiveTracking	A feature that allows commuters to monitor the real-time location of their booked bus.
Notification	An alert sent to uses about bookings, updates, delays or cancellations
PickupPoint	The designated spot where a commuter boards the bus.
RealTimeUpdates	Instant information provided to commuters and drivers about bus location, estimated arrival time and delays.
Route	The bus's predetermined route, which includes destinations, departure locations, and traffic-adjusted directions.

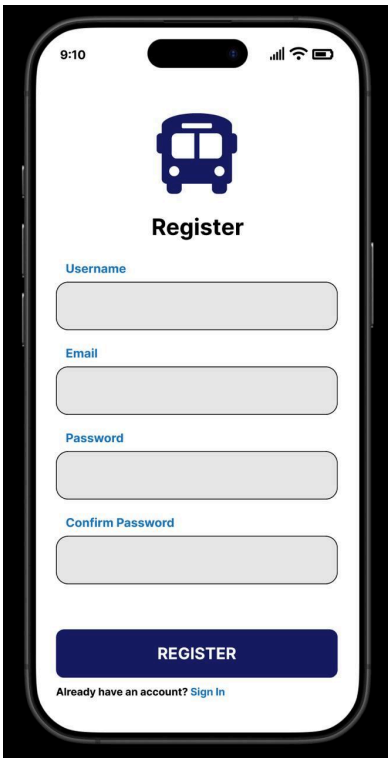
Status	Current status of the bus status such as “delayed”, “cancelled”, “available” and “unavailable”.
TrafficMonitoringSystem	Real-time traffic data used to optimize bus routes and provide accurate estimated arrival times.
User	Anyone who has created an account and uses the bus booking mobile app.

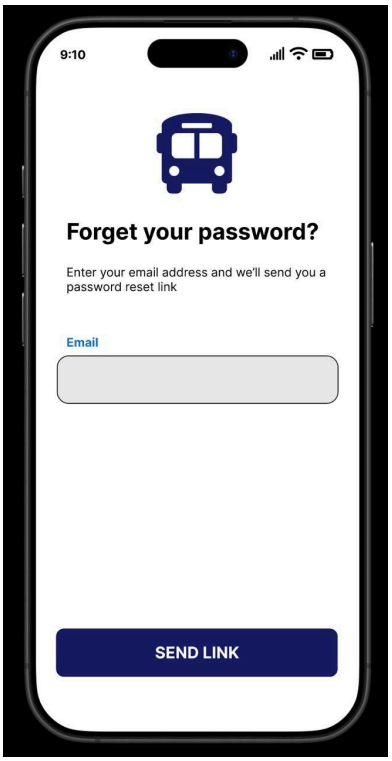
8. UI Mockup

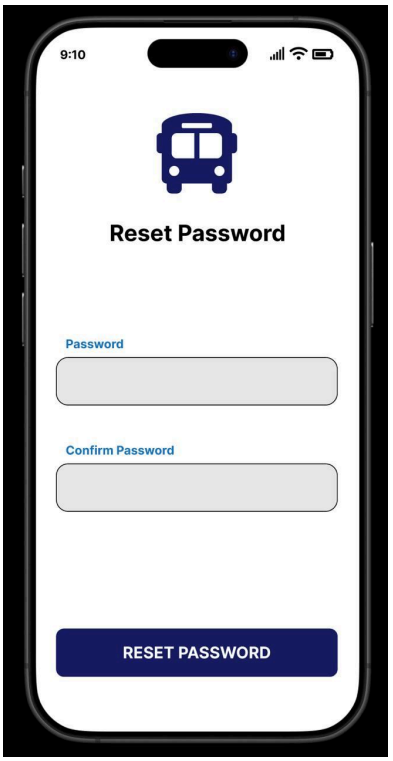
8.1 User Management



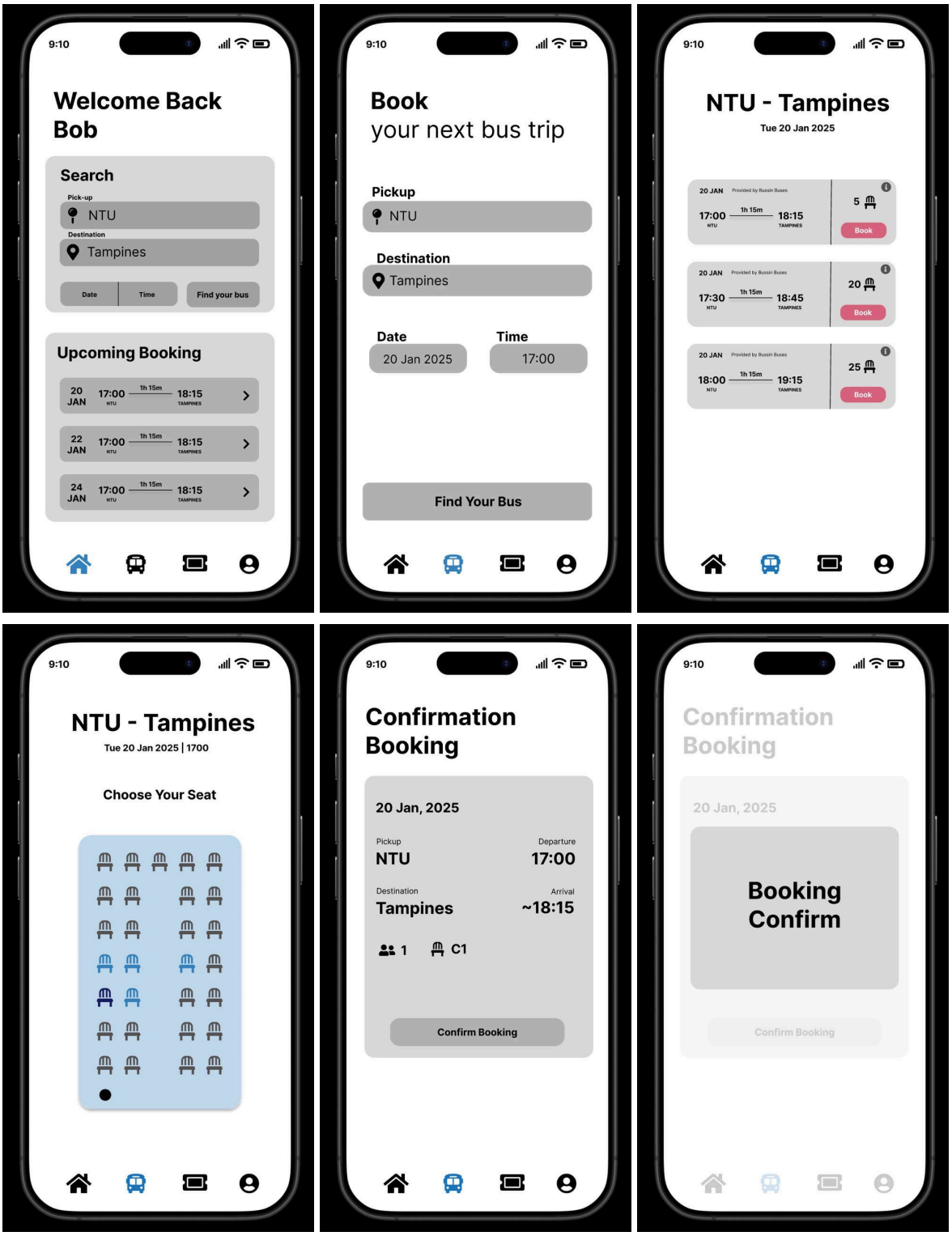




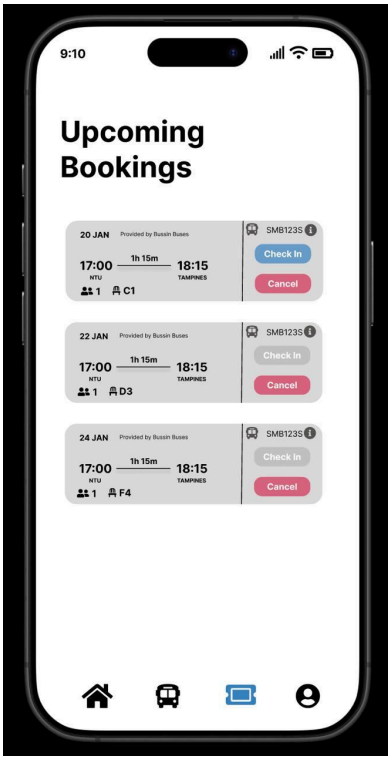


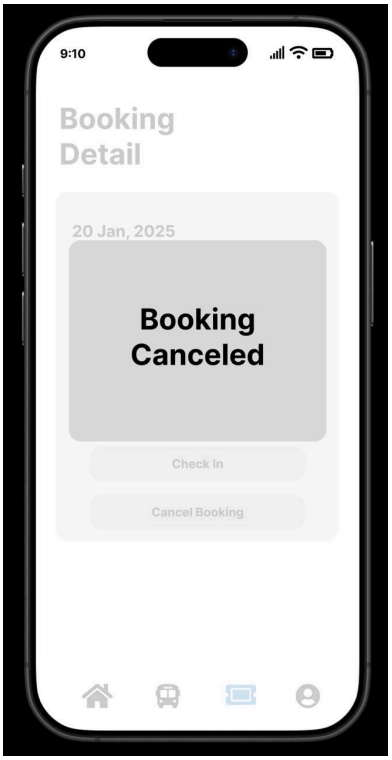


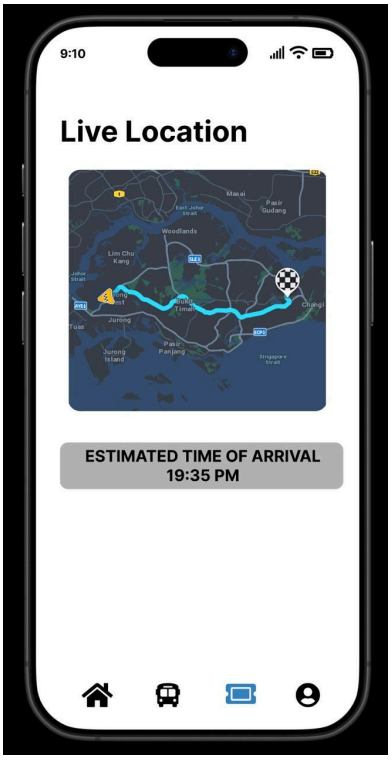
8.2 Commuters - Home Page + Booking of Bus Trip



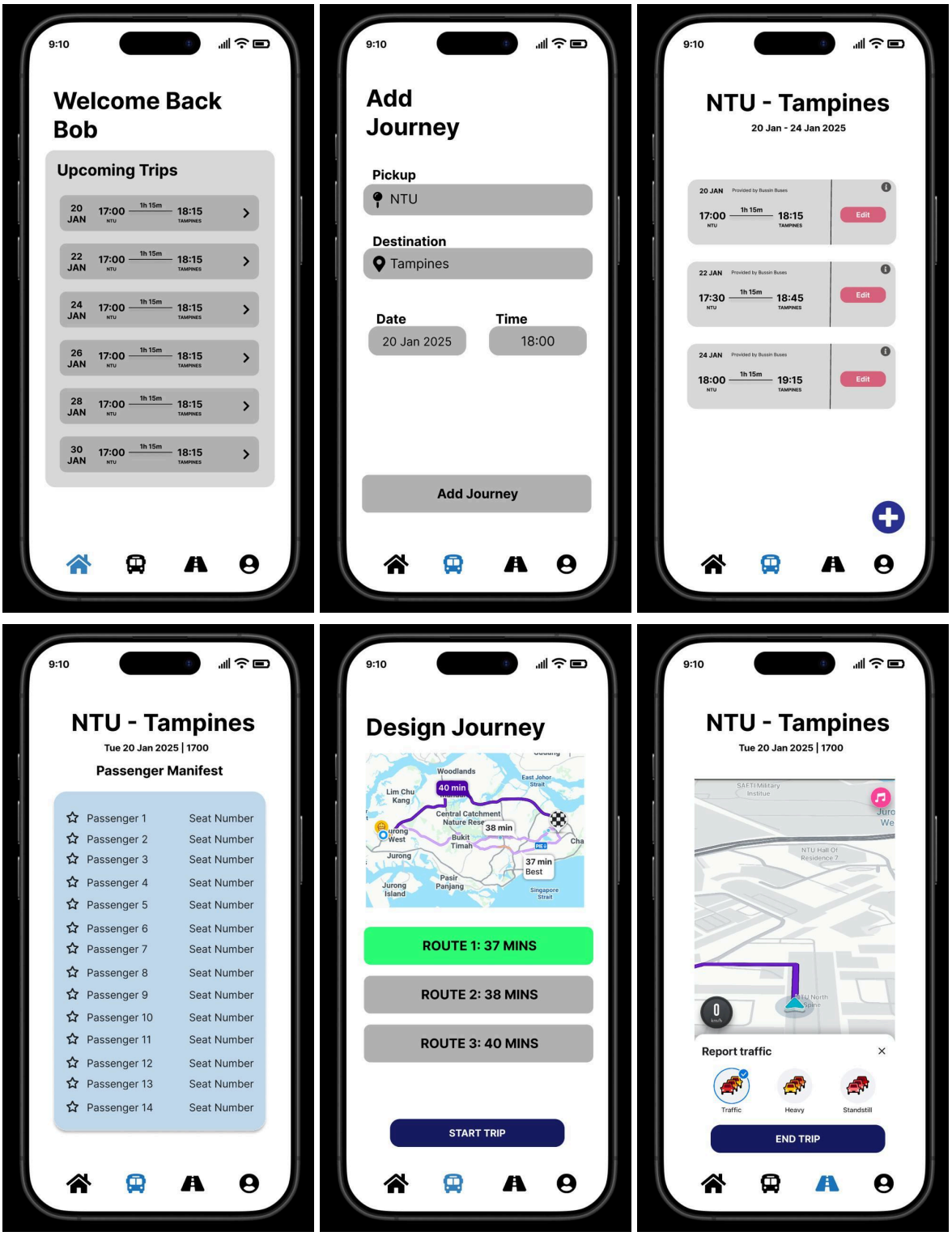
8.3 Commuter - Viewing Upcoming Booking + Live Tracking + Account Management



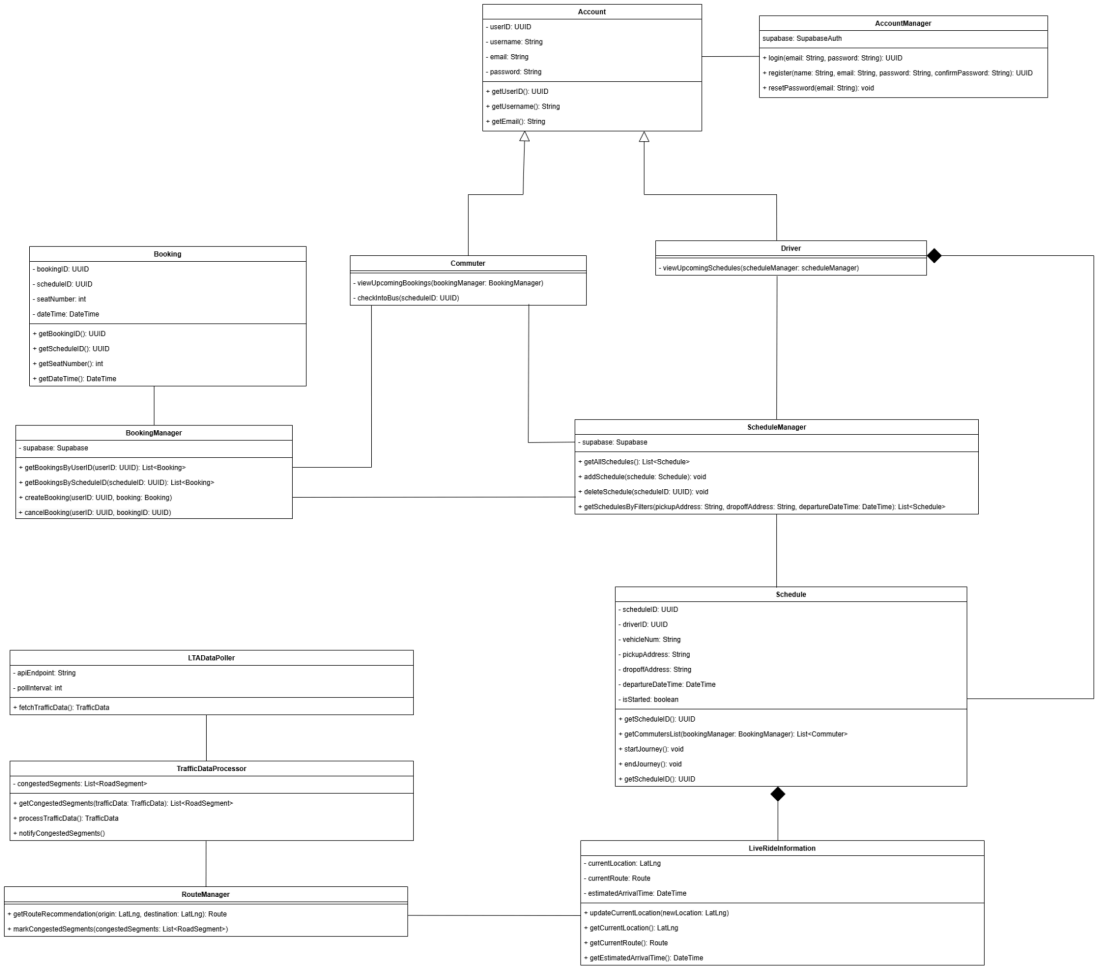




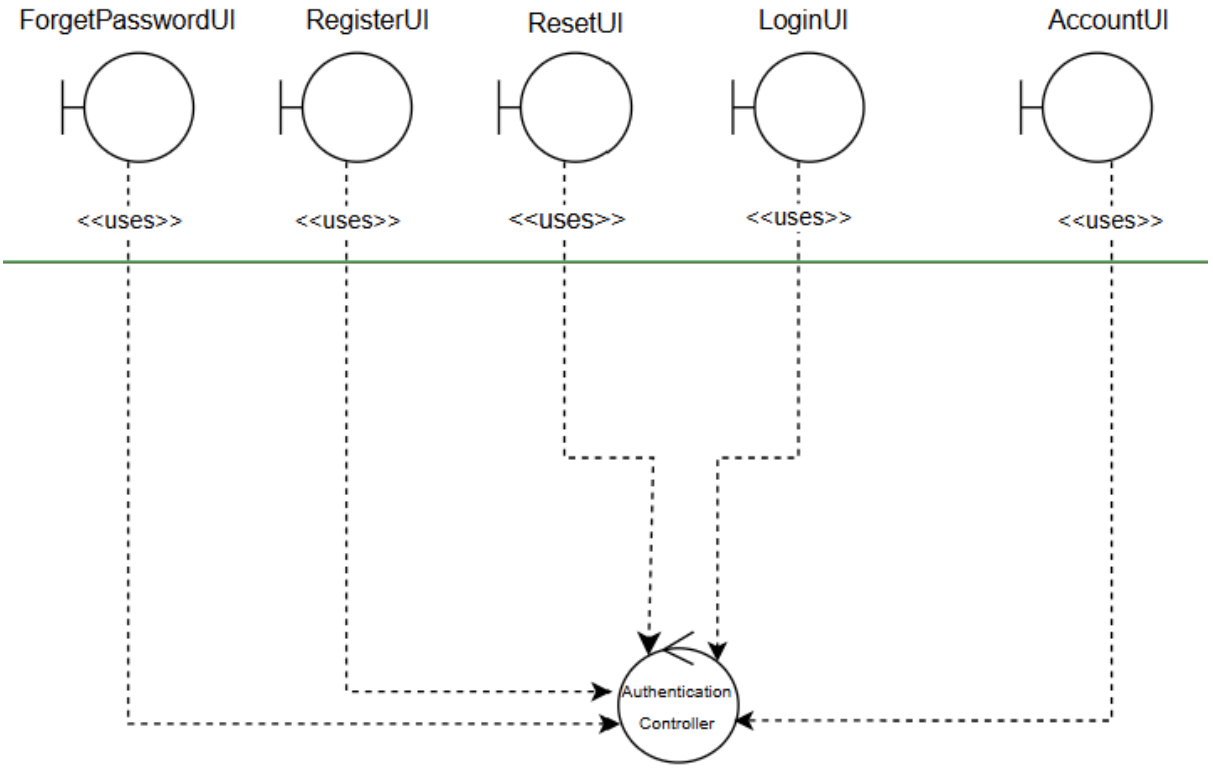
8.4 Driver - Homepage + Add Journey + Start & End Journey

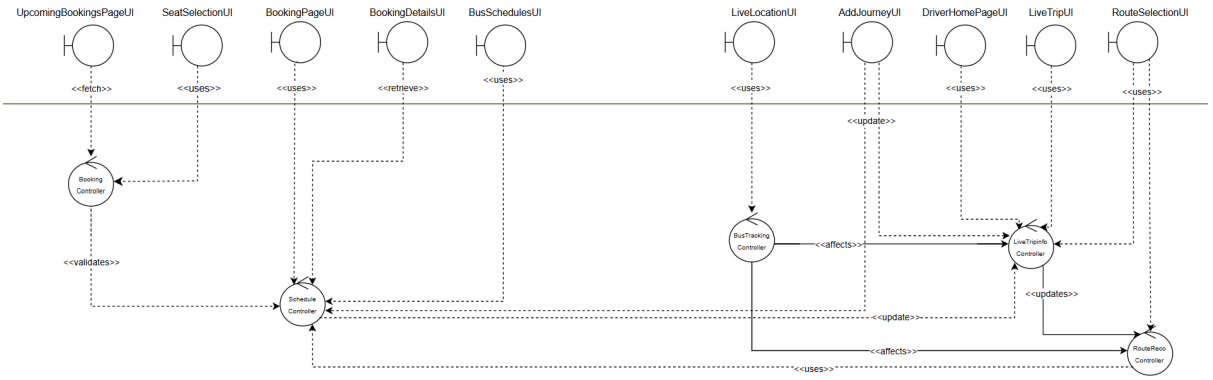


9. Class Diagram



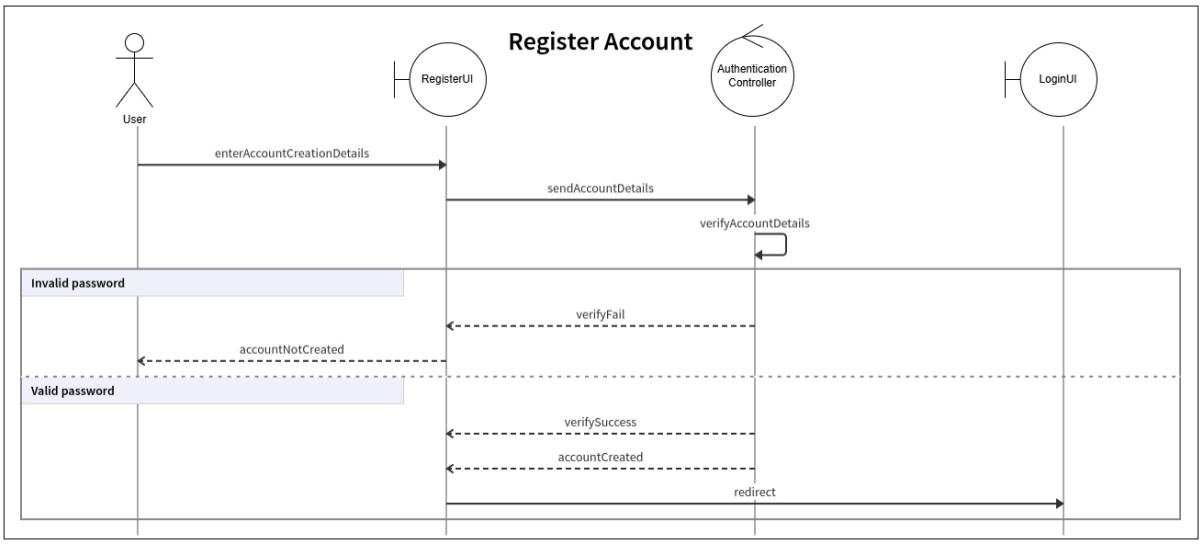
10. Key boundary classes and control classes



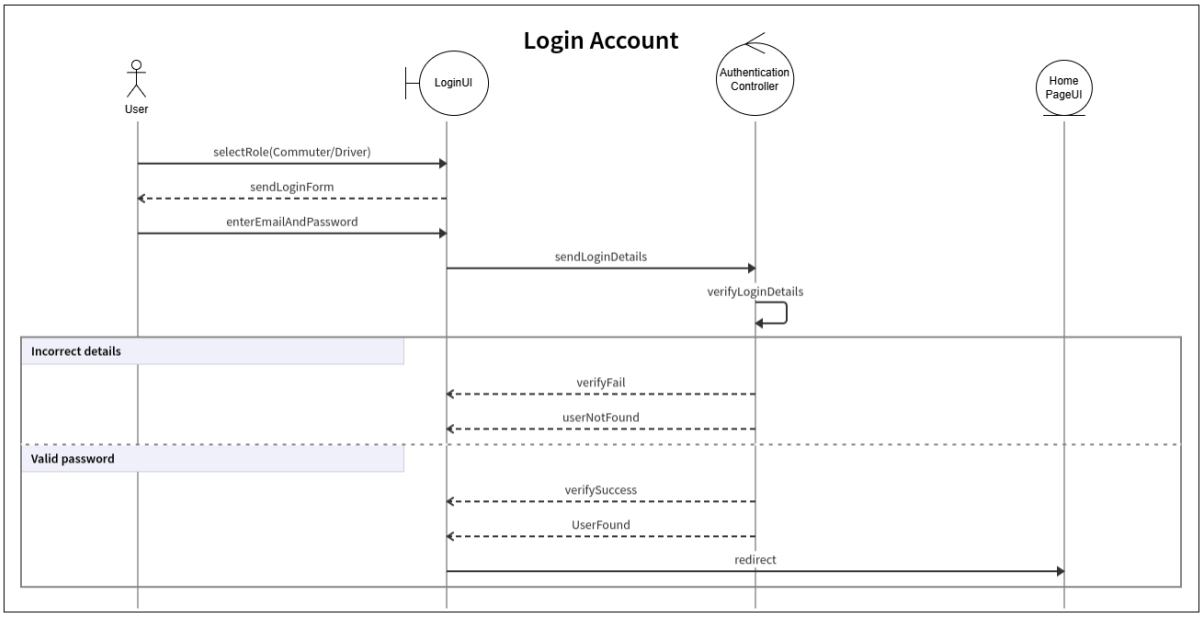


11. Sequence Diagrams

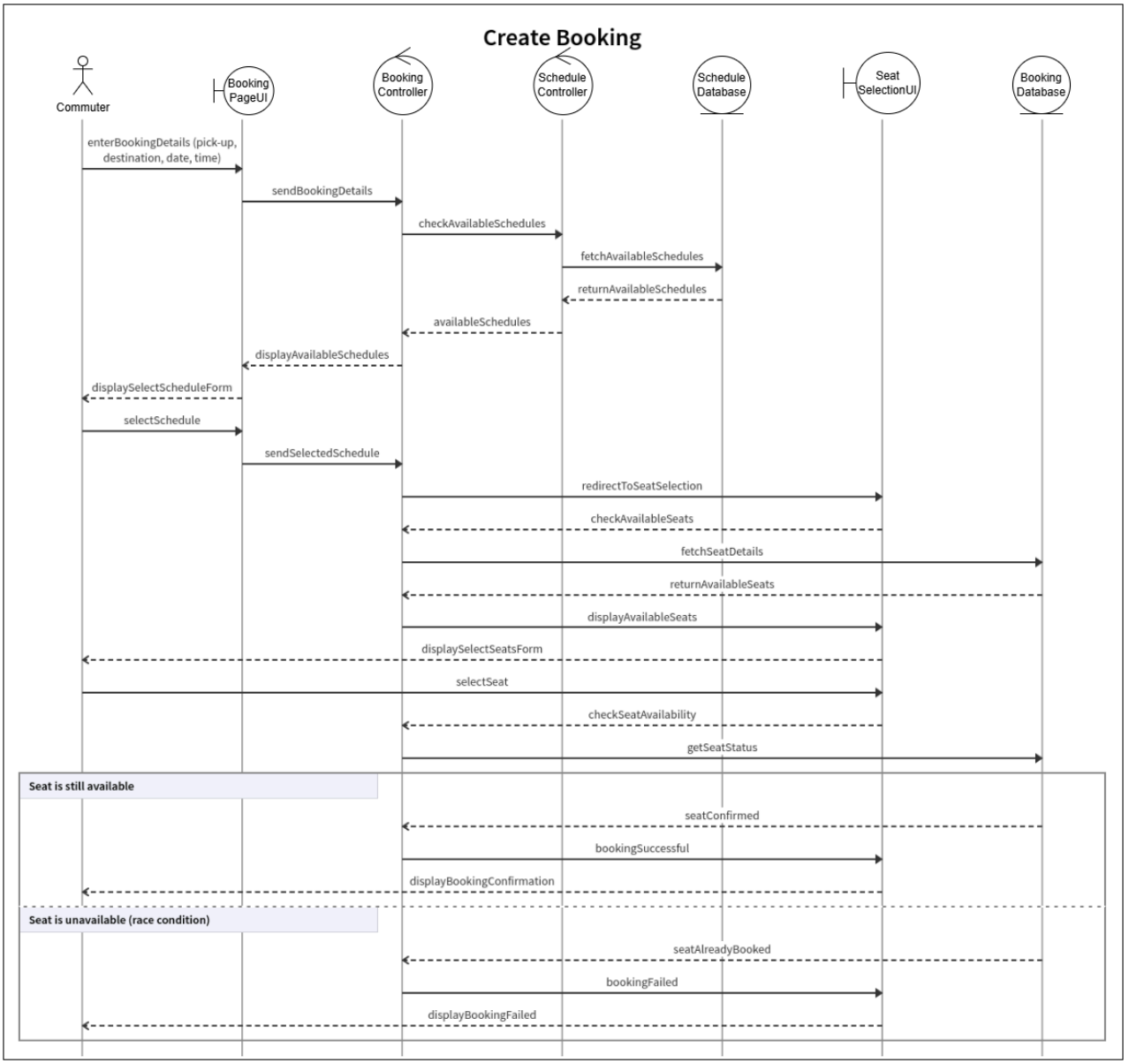
Use Case ID: 1.2
Use Case Name: Register Account



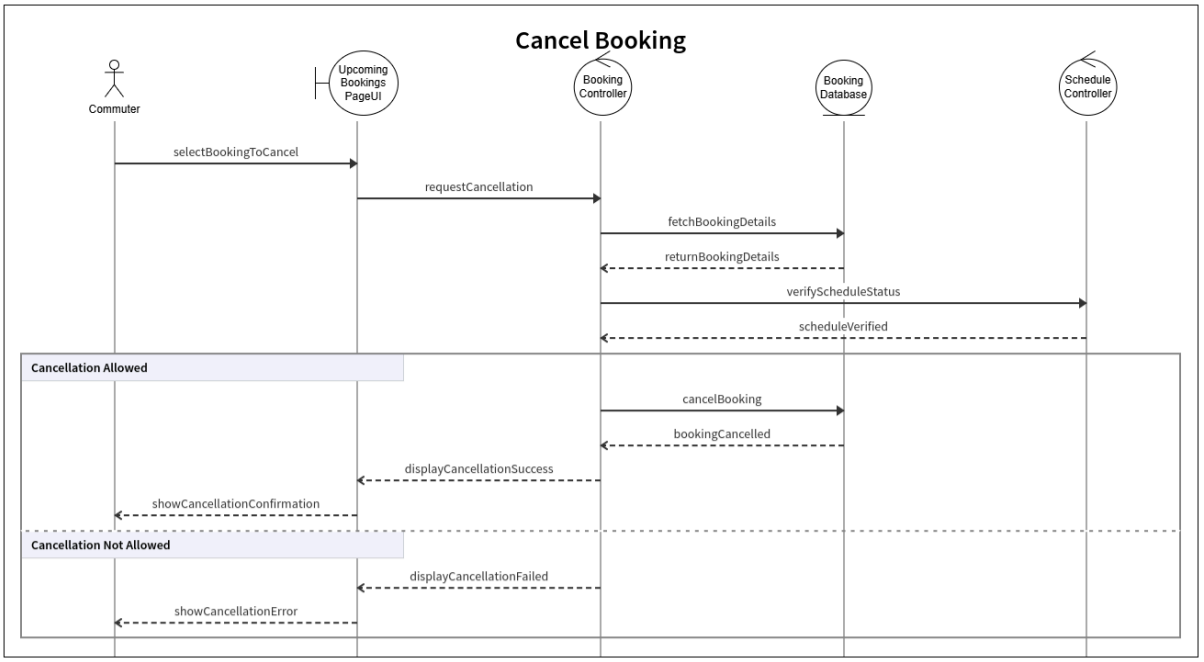
Use Case ID: 1.3
Use Case Name: Login Account



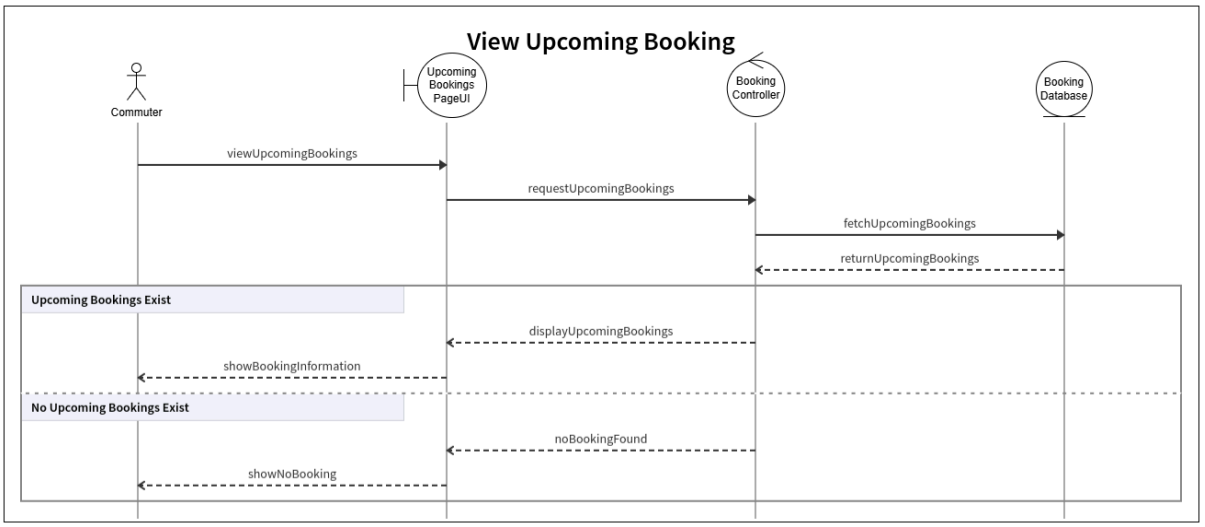
Use Case ID: 2.2
Use Case Name: Create Booking



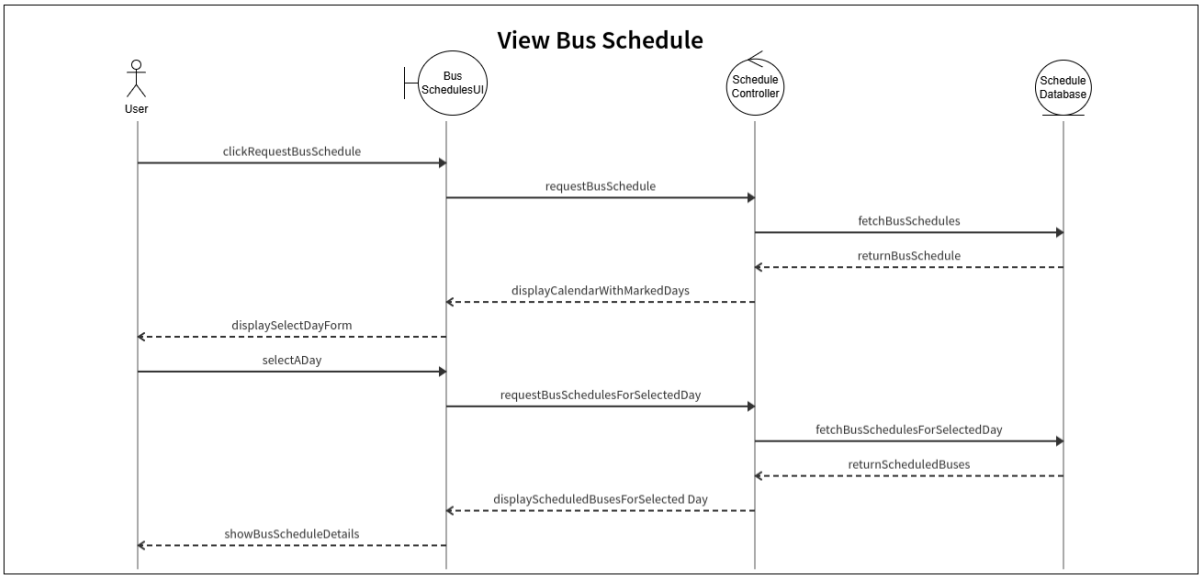
Use Case ID: 2.3
Use Case Name: Cancel Booking



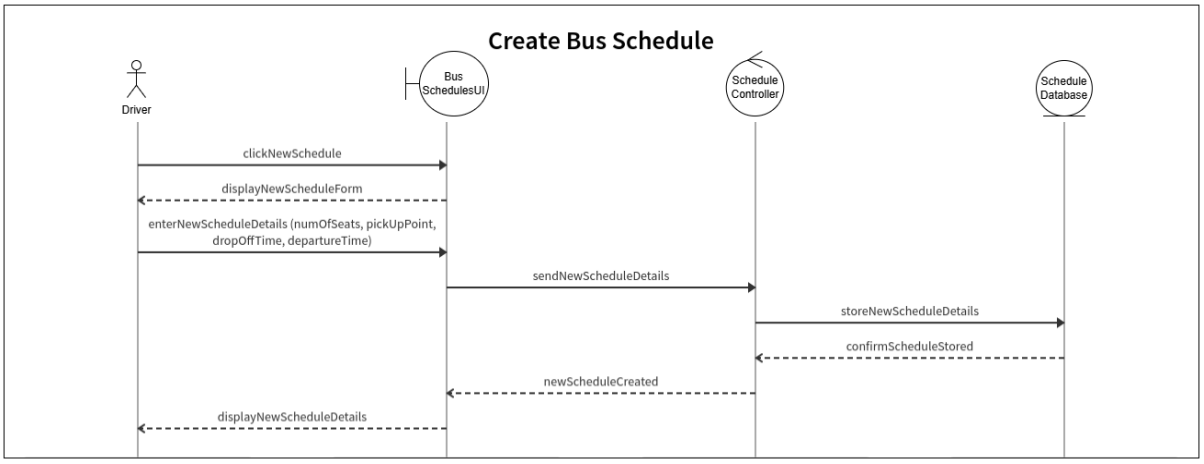
Use Case ID: 2.4
Use Case Name: View Upcoming Booking



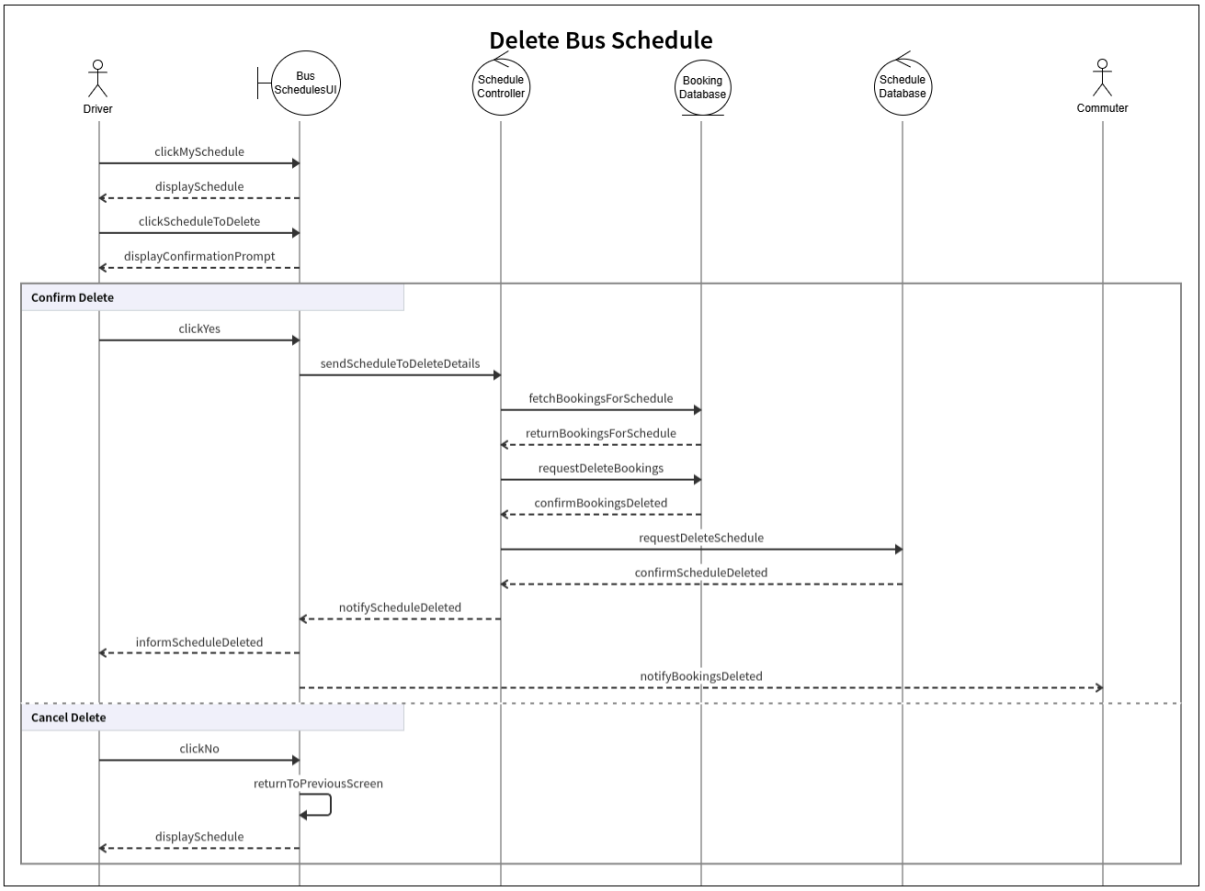
Use Case ID: 3.2
Use Case Name: View Bus Schedule



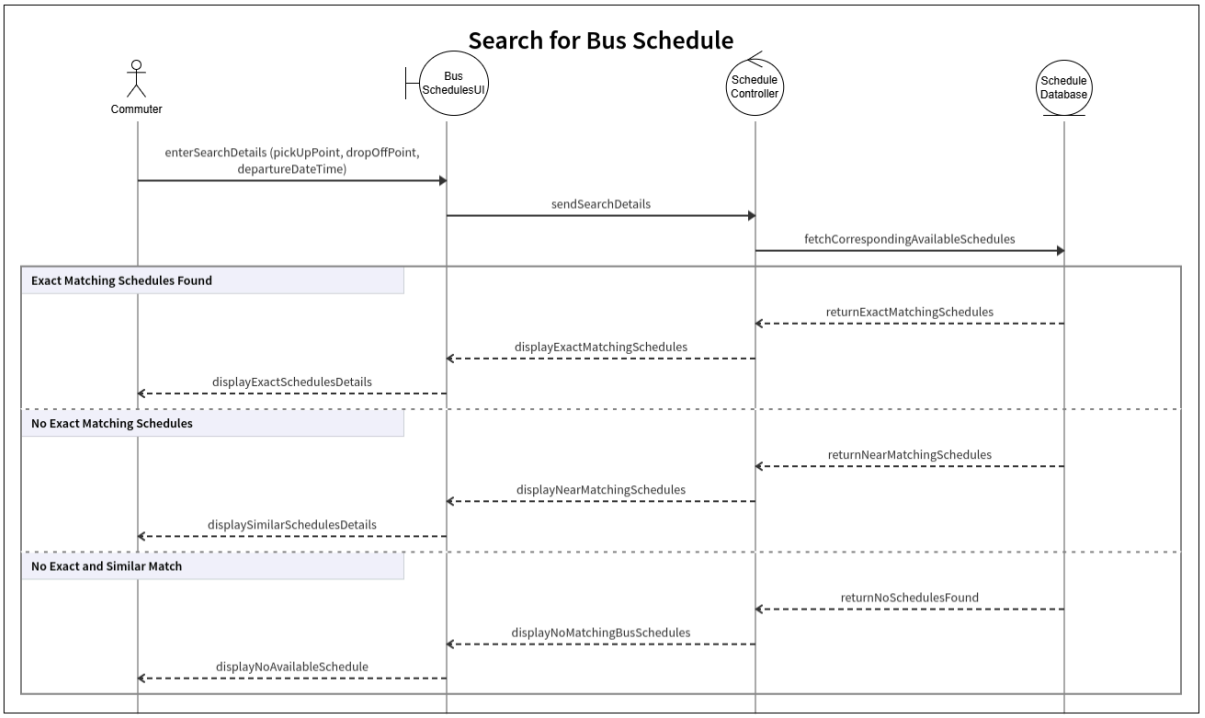
Use Case ID: 3.3
Use Case Name: Create Bus Schedule



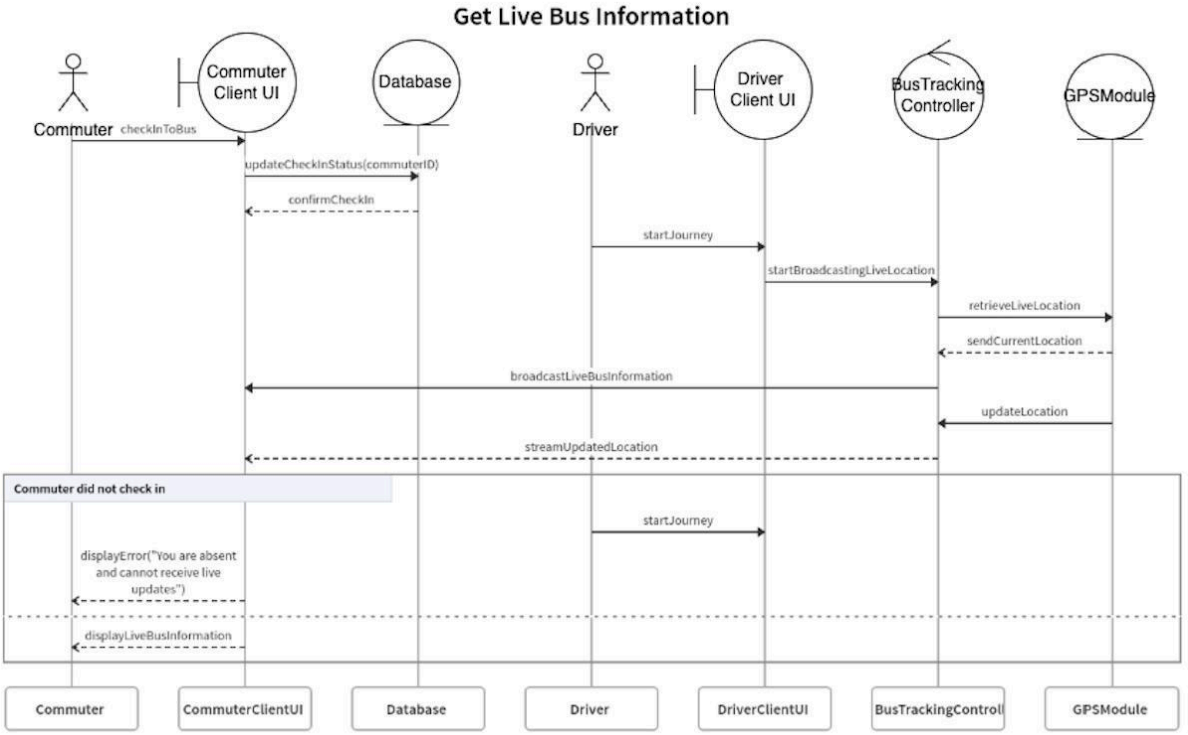
Use Case ID: 3.4
Use Case Name: Delete Bus Schedule



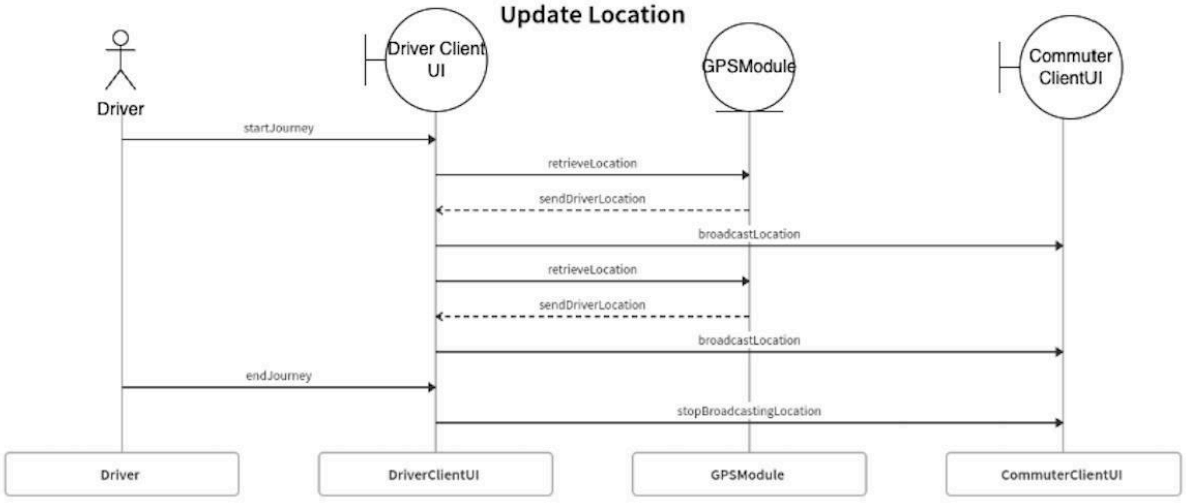
Use Case ID: 3.5
Use Case Name: Search for Bus Schedule



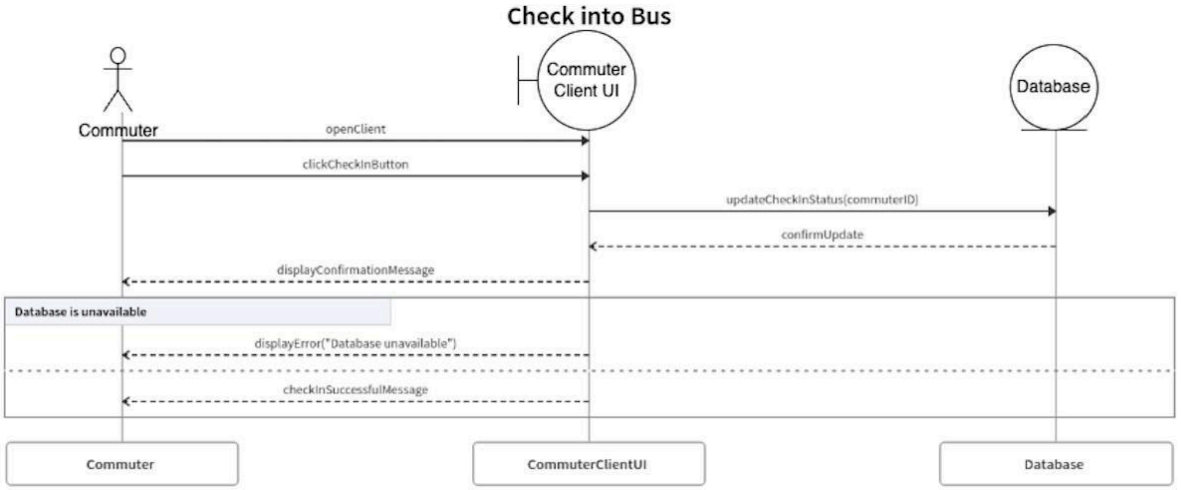
Use Case ID: 4.1
Use Case Name: Get Live Bus Information



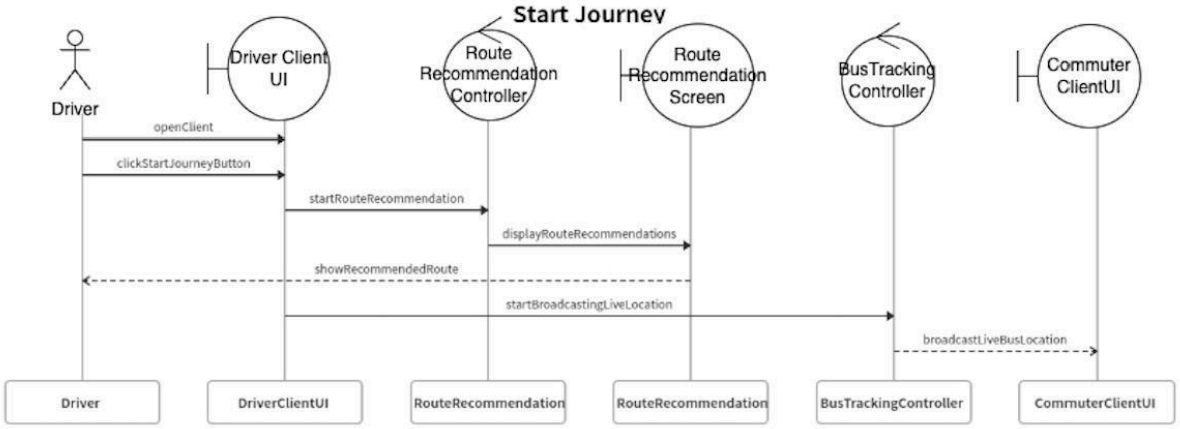
Use Case ID: 4.2
Use Case Name: Update Location



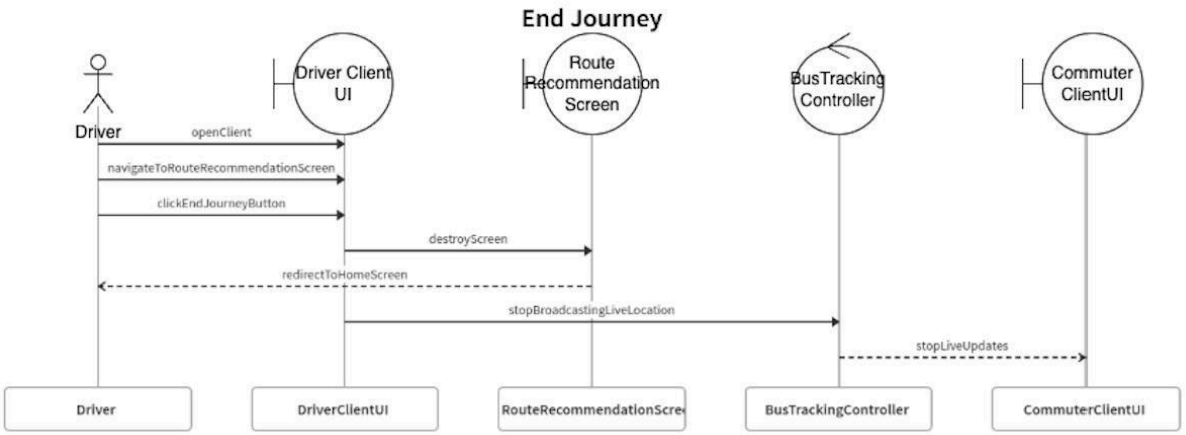
Use Case ID: 4.3
Use Case Name: Check into Bus



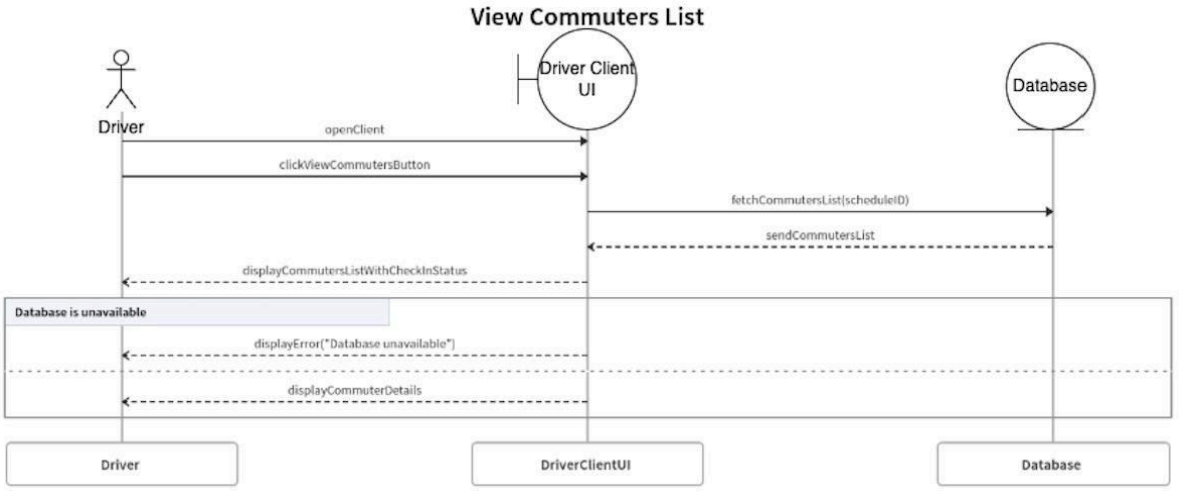
Use Case ID: 4.4
Use Case Name: Start Journey



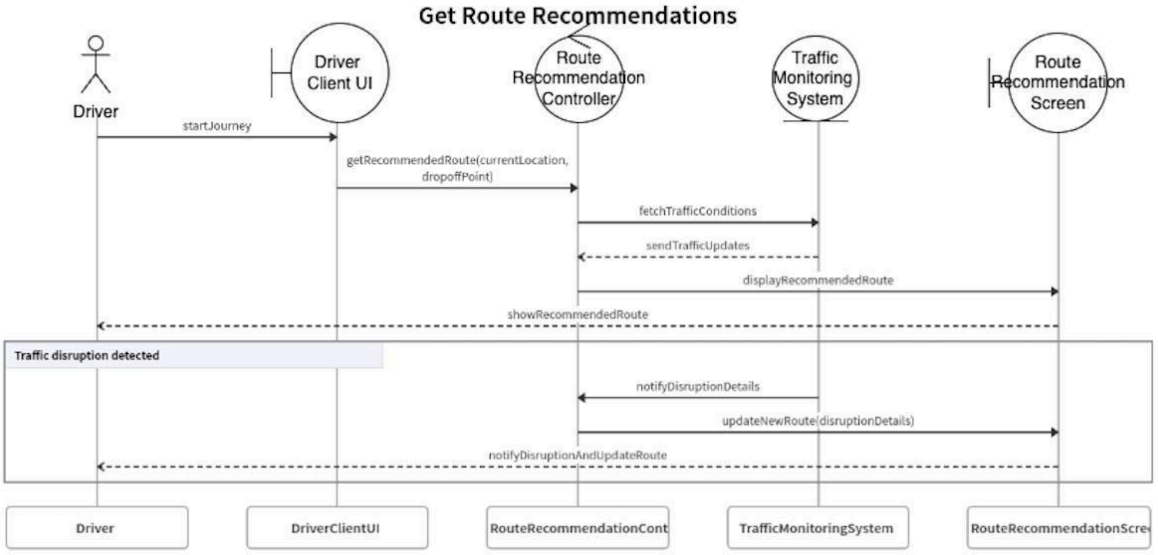
Use Case ID: 4.5
Use Case Name: End Journey



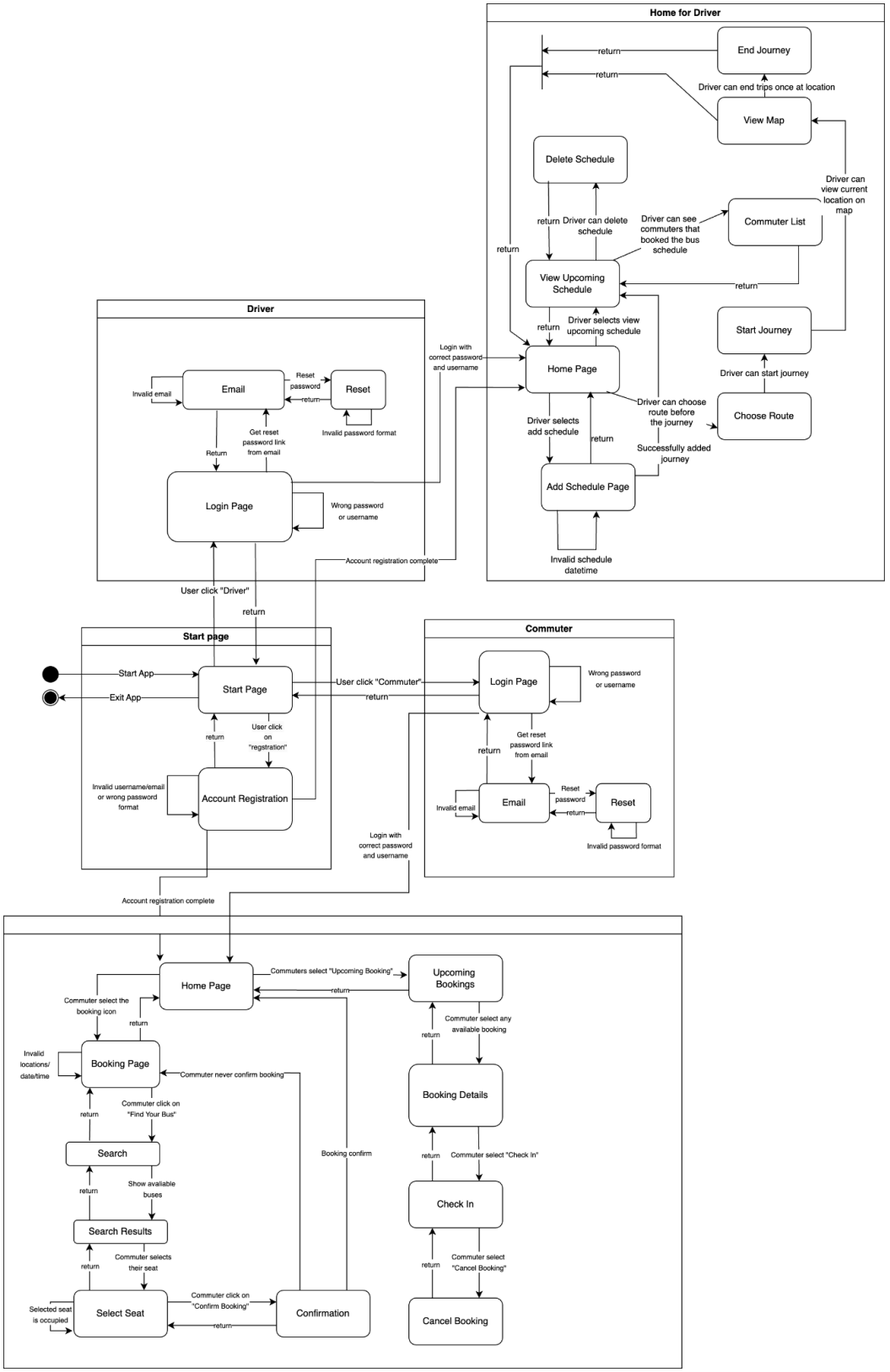
Use Case ID: 4.6
Use Case Name: View Commuters List



Use Case ID: 5.1
Use Case Name: Get Route Recommendations



12. Initial Dialog Map



13. System Architecture Diagram