Software Requirements and Design Document

for

AdventureSync-Tourism Management System

Prepared by:

Saleha Irum 22k-4360

Afsah Areeb 22i-1046

FAST Nuces

Fall 2024

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this project is to improve the Tourism sector of Pakistan. This app allows tourists to manage their trips more easily when they visit northern areas.

1.2 Product Scope

The product scope of the tourism app focuses on addressing the underdevelopment of tourism infrastructure in Pakistan by providing a seamless and user-friendly platform for both local and foreign tourists. The app will enable users to book hotel rooms, rent vehicles, and reserve seats on bus tours to popular destinations

1.3 Title

AdventureSync-Tourism Management System

1.4 Objectives

This app will allow tourists to:

- Book hotel rooms and also avail food items
- Rent cars
- Book seats in buses

1.5 Problem Statement

The problem of *underdevelopment of tourism infrastructure* in Pakistan negatively impacts *both local and foreign tourists*. They have a *hard time visiting tourist spots,* and this lack of accessible and efficient services *reduces the potential revenue* Pakistan can generate from its tourism sector. A successful solution would be *to make a tourism app.* This app will offer tourists a seamless way to book hotel rooms, find optimal travel routes, rent vehicles, and make appointments with local guides.

2. Overall Description

2.1 Product Perspective

The **AdventureSync-Tourism Management System** is a **new, self-contained product** specifically designed to address the underdevelopment of tourism infrastructure in Pakistan. This product does not replace or enhance any existing system but introduces a novel approach to providing an efficient and accessible platform for tourists. The app is standalone and caters to the unique needs of tourists visiting the northern regions of Pakistan by offering services such as hotel bookings, vehicle rentals, and bus reservations.

The system is independent and does not require integration with pre-existing systems, making it a complete solution for managing tourist-related activities.

The following diagram illustrates the major components of the system, their interconnections, and external interfaces:

- **Frontend (JavaFX):** Provides the graphical user interface (GUI) for users to interact with the app.
- **Backend (MSSQL Database):** Stores and retrieves data related to hotel rooms, car rentals, bus reservations, and guides.
- **Controller (Business Logic):** Serves as the intermediary between the frontend and backend, handling requests and ensuring proper data flow.

2.2 Product Functions

The AdventureSync-Tourism Management System is designed to perform the following high-level functions:

Hotel Room Management:
- Allow tourists to search for and book hotel rooms.
- Enable hotel owners to manage room availability and food services.

Vehicle Rental:
- Provide options for tourists to rent cars for self-guided tours.
- Allow vehicle owners to manage rental availability.

Bus Reservation System:
- Facilitate booking seats on bus tours to popular destinations.
- Allow bus operators to manage schedules and seat availability.

User Account Management:
- Allow tourists, hotel owners, vehicle providers, and guides to create and manage their accounts.

2.3 List of Use Cases

- Register account
- Rent Car
- Book a Room
- Book Seat in Bus
- Order Food
- Make Payment
- Give Feedback
- Manage Hotel Rooms
- Manage Cars
- Complete Bus Ride

2.4 Extended Use Cases

1. Rent a Car

a. Use case name

Rent a car

b. Scope

AdventureSync

c. Level

User goal

d. Primary actor

Tourist

e. Stakeholders and interests

1) Tourist: wants to rent a car for his travel
2) Travel Agency: wants their cars to be rented successfully

f. Preconditions

1) Tourist should be logged in

g. Postconditions

1) Tourist has successfully rented a car (car status is set to “rented”)

h. Main success scenario

Actor	System
1. Tourist request to rent a car 3. Tourist select the car/s	2. System display the cars 4. System assigns this car to the tourist.

i. Extensions
2a) System does not have any cars. There is nothing to display, tourist exits.

2. Book a Room

a. Use case name
Book a room
b. Scope
AdventureSync
c. Level
User goal
d. Primary actor
Tourist
e. Stakeholders and interests
1) Tourist: wants to book a room in the hotel
2) Hotel: wants the room to be booked successfully

f. Preconditions
1) Tourist should be logged in

g. Postconditions
1) Tourist has successfully booked a room
2) Hotel owner has been notified the room is booked

h. Main success scenario

Actor	System
1. Tourist request to book a room	
	2. System display the hotels
3. Tourist selects the hotel	
	4. System display the room
5. Tourist selects the room	
	6. System books room for tourist

i. Extensions
2a) System does not have hotels to display then tourist exits.
4a) System does not have rooms to display then tourist exits.

3. Book Seat in Bus

a. Use case name
Book seat in bus
b. Scope
AdventureSync
c. Level

User goal

d. Primary actor

Tourist

e. Stakeholders and interests

1) Tourist: wants to book a seat in the bus
2) Travel Agency: wants the seat to be booked

f. Preconditions

1) Tourist should be logged in

g. Postconditions

1) Tourist has successfully booked a seat
2) Travel Agency has been notified the seat is booked

h. Main success scenario

Actor	System
1. Tourist request to book a seat	
	2. System display bus and associated tours.
3. Tourist select the the bus.	
	4. System display the available seats
5. Tourist select the seat/seats	
	6. System books the seat

i. Extensions

2a) System does not have any bus then tourist exits.
4a) System does not have any seats. There is nothing to display, tourist exits.
5a) Number of seats not available which user wants

4. Order food

a. Use case name

Order food

b. Scope

AdventureSync

c. Level

User goal

d. Primary actor

Tourist

e. Stakeholders and interests

3) Tourist: wants to order some food/refreshment

4) Kitchen: wants to be informed about the order

f. Preconditions

2) Tourist should be logged in

g. Postconditions

3) Tourist has successfully ordered the food
4) Kitchen has been notified about the order

h. Main success scenario

Actor	System
1. Tourist request to order food 3. Tourist selects food from menu	2. System display menu 4. System place order for food

i. Extensions

2a) System does not have any food. There is nothing to display, tourist exits.

5. Register Account

a. Use case name

Register account

b. Scope

AdventureSync

c. Level

User goal

d. Primary actor

Tourist

(This same method will also be used by Hotel Owner, Tour Guide, Bus Driver, Travel Agency Owner)

e. Stakeholders and interests

1) Tourist: wants to register successfully
   (same applies to all other actors)

f. Preconditions

1) Tourist: verified CNIC
   (same applies to all other actors)

g. Postconditions

1) Tourist, Hotel Owner, Tour Guide, Bus Driver, Travel Agency Owner has successfully registered their account.

h. Main success scenario

Actor	System
1. User request to register account 3. User provides the detail	2. System ask details(include CNIC, name, dob) 4. System informs the user their account has been registered successfully

i. Extensions

4a) User does not provide the specified details and the account is not registered successfully.

6. Make Payment

a. Use case name

Make Payment

b. Scope

AdventureSync

c. Level

User goal

d. Primary actor

Tourist

e. Stakeholders and interests

1) Tourist: wants to ensure easy payment
2) Service Provider wants to get payment without issues. Service providers are:
    a) Hotel Owner(Payment for room/service)
    b) Travel agency
3) Payment gateway: ensure easy payment
4) Government tax agencies: ensure easy payment and obtaining taxes

f. Preconditions

1) Tourist should be logged in.

g. Postconditions

1) Money deducted from tourist
2) Money added to service provider

h. Main success scenario

(as there are multiple recipients in payment, considering only one here, i.e local guide)

Actor	System
1. Tourist makes a transaction. (for any service mentioned above) 3. Tourist chooses payment type.	2. System displays transaction details.

a) Pay by Cash b) Pay by credit card	4. Stores the transaction. 5. Generates receipt.

Section: Pay by Cash

Actor	System
1. Tourist gives amount.	2. System marks this transaction as paid.

Section: Pay by Credit Card

Actor	System
1. Tourist enters card details	2. Generates credit payment request. 3. Receives approval from the system. 4. Stores details of transaction.

i. Extensions

Pay by cash:

1a) Tourist does not have enough cash. Other payment method used.

Pay by credit card:

3a) Credit request denied, suggest different payment method.

7. Give Feedback

a. Use case name

Give Feedback

b. Scope

AdventureSync

c. Level

User goal

d. Primary actor

Tourist

e. Stakeholders and interests

1) Tourist wants to inform service providers(hotel owner, travel agency owner) about their feedback.

f. Preconditions

1) Tourist should be logged in.

g. Postconditions

1) Feedback should be stored.

h. Main success scenario

Actor	System
1. Tourist request to give feedback(of any service)	
	2. System asks for rating and comment.
3. Tourist gives rating and additional comments.	
	4. System stored individual feedback. 5. System updates overall rating of the service.

i. Extensions
4a) System fails to store feedback.
5a) System fails to update overall rating.

<u>8. Complete Bus Tour</u>

a. Use case name
Complete Bus Tour
b. Scope
AdventureSync
c. Level
User goal
d. Primary actor
Bus Driver
e. Stakeholders and interests

1) Bus driver wants to complete a tour that he has done

f. Preconditions

1) Bus owner is logged in
2) Tour was assigned to bus driver.

g. Postconditions

1) This ride is no longer associated with a bus driver.

h. Main success scenario

Actor	System
1. Bus driver requests to complete the tour.	
	2. System prints tour details.
3. Bus driver views details of the tour and completes it.	
	4. System removes the tour.

i. Extensions

2a) The bus is not assigned any tour, so there is nothing to display.
4a) Tour date is in the future so it can not be completed.

9. Manage Cars

a. Use case name

Manage cars

b. Scope

AdventureSync

c. Level

User goal

d. Primary actor

Travel Agency Owner

e. Stakeholders and interests

1) Travel Agency Owner: wants to add, delete and edit cars without issues.

f. Preconditions

1) Travel Agency Owner should be authorized

g. Postconditions

1) Car is successfully added to/removed from/edited in the system.

h. Main success scenario

Actor	System
1. Travel Agency Owner goes to manage car selection.	2. System displays options.
3. Travel Agency Owner can: add a car, delete a car, or edit a car.	4. System displays required fields.
5. Travel agency owner enters necessary details.	6. Details are validated and saved by system.

(explanation of these sub level user goals here)

a) **Add a car**
   Travel Agency Owner enters car details, and a new car is stored in the system.
b) **Delete a car**
   Travel Agency Owner enters id of car to be deleted. If not reserved, it is deleted by system.
c) **Edit a car**

Travel Agency Owner enters id of car to be edited. Travel Agency Owner changes required details and updated car is stored in system.

i. Extensions

3a) Travel Agency Owner tries to delete a car that does not exist
3b) Travel Agency Owner enters invalid details(e.g repeating car plate number)
3c) Travel Agency Owner tries to update a car that does not exist

10. Manage Hotel Rooms

a. Use case name

Manage Hotel Rooms

b. Scope

AdventureSync

c. Level

User goal

d. Primary actor

Hotel Owner

e. Stakeholders and interests

1) Hotel Owner wants to add, remove and edit hotel room information easily.

f. Preconditions

1) Hotel Owner should be authorized.

g. Postconditions

1) Hotel room should be added to/removed from/modified in the system

h. Main success scenario

Actor	System
1. Hotel Owner goes to manage hotel room selection.	2. System displays options.
3. Hotel Owner can: add a hotel room, delete a hotel room, or edit a hotel room.	4. System displays required fields.
5. Hotel Owner enters necessary details.	6. Details are validated and saved by system.

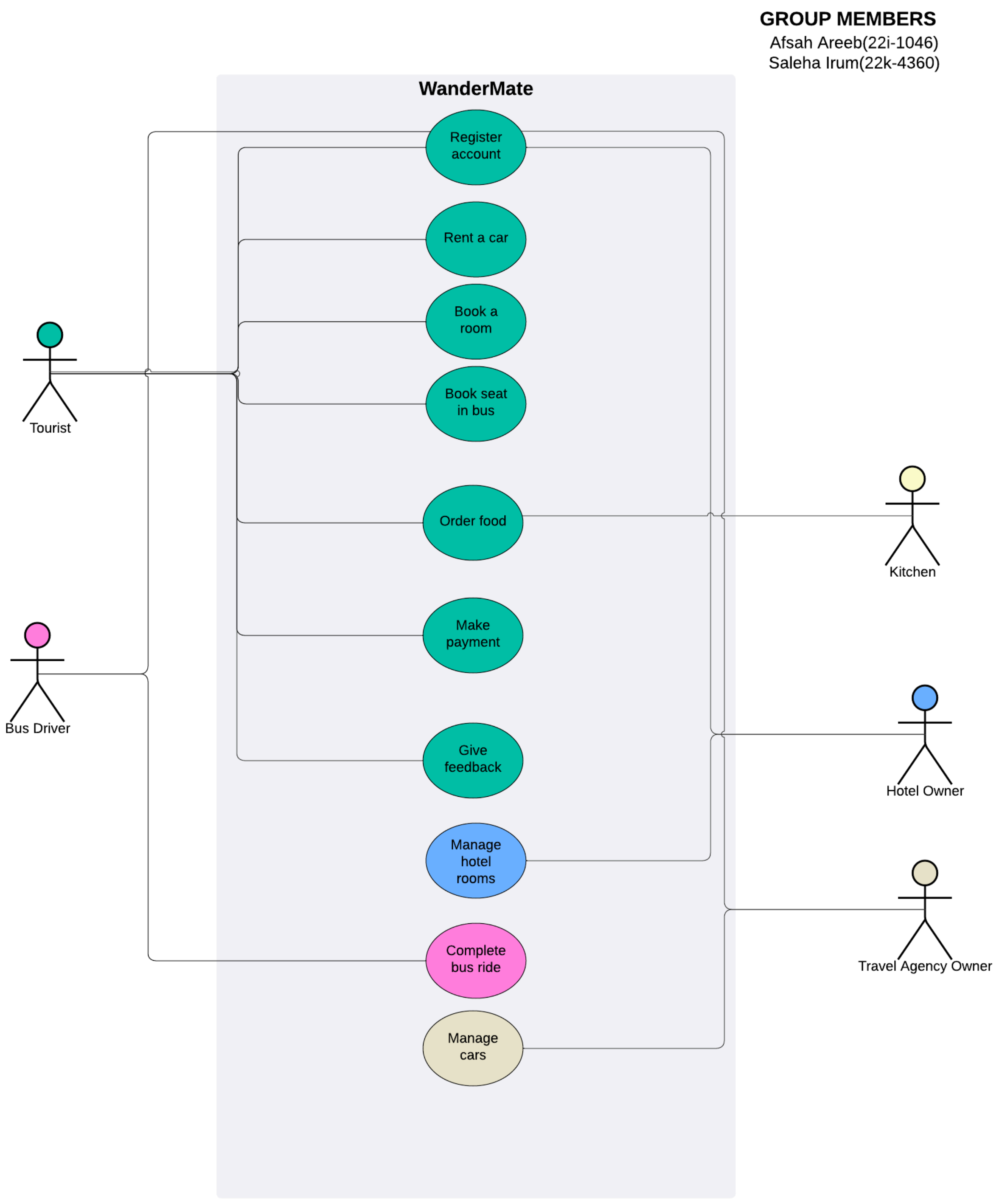
(explanation of these sub level user goals here)

d) **Add a hotel room**
   Hotel Owner enters hotel room details, and a new hotel room is stored in the system.
e) **Delete a hotel room**
   Hotel Owner enters id of hotel room to be removed. If not booked, it is deleted by system.

f) **Edit a hotel**
Hotel Owner enters id of hotel room to be edited. Hotel Owner changes required details and updated hotel room is stored in system.

i. Extensions
3a) Hotel Owner tries to delete a hotel room that does not exist
3b) Hotel Owner enters invalid details
3c) Hotel Owner tries to update a hotel room that does not exi

2.5 Use Case Diagram



3. Other Nonfunctional Requirements

3.1 Performance Requirements

For the **AdventureSync-Tourism Management System**, which operates as a simple standalone JavaFX application with a single user at a time, the performance requirements are as follows:

Response Time:

- The system must respond to user interactions (e.g., booking a hotel room, searching for a vehicle) within **1 second** to ensure a smooth experience.
- Basic database queries (e.g., fetching available rooms or vehicles) should execute within **2 seconds**.

Data Handling:

- The system should efficiently process and display data for up to **100 entries** (e.g., hotels, vehicles, bus seats) without noticeable delays.

Error Handling and Recovery:

- If an error occurs, the app should notify the user with a clear error message and allow them to retry the operation immediately.

3.2 Safety Requirements

For the **AdventureSync-Tourism Management System**, the following safety requirements are identified to mitigate risks of data loss, unauthorized access, or misuse of the system:

Data Protection:

All user data (e.g., personal information, booking details) must be securely stored in the database to prevent unauthorized access or data breaches.

Error Prevention:

Input validation must be implemented to prevent invalid or harmful data from being entered into the system. For example, ensure that booking dates are valid and car rental availability aligns with actual data.

3.3 Security Requirements

The AdventureSync-Tourism Management System must adhere to the following security requirements to ensure the protection of user data and prevent unauthorized access:

User Authentication:

- Implement a secure login system using unique usernames and passwords for all users (e.g., tourists, hotel owners, and administrators).
- Enforce password policies such as a minimum length of 8 characters, and require a mix of letters, numbers, and special characters.

Access Control:

- Differentiate user roles (e.g., admin, tourist, service provider) and restrict access to specific features and data based on roles.
- Ensure that no user can perform unauthorized actions, such as deleting records or accessing another user's private data.

Input Validation:

- Validate all user inputs to prevent injection attacks (e.g., SQL injection, XSS).

3.4 Software Quality Attributes

The AdventureSync-Tourism Management System is designed with the following software quality attributes to ensure it meets user and developer expectations:

Usability:

- The system must provide an intuitive user interface that is easy to navigate, requiring no more than 15 minutes for a new user to become familiar with the core features (e.g., booking hotels or renting cars).
- Include clear error messages and prompts to guide users through issues or invalid inputs.

Correctness:

- All operations, such as booking availability checks and financial calculations, must provide accurate results.
- Validation rules must ensure data integrity, such as avoiding double bookings or scheduling conflicts.

3.5 Business Rules

The AdventureSync-Tourism Management System adheres to the following business rules to ensure proper operation and user access management:

User Roles and Permissions:

- **Tourist**:
  - Can browse and book hotels, rent vehicles, and reserve bus seats.
  - Can view and edit their own profile and bookings.
  - Cannot modify or access other users' bookings or data.
- **Service Providers (e.g., Hotel Owners, Vehicle Rental Agents):**
  - Can manage their specific offerings, such as updating room availability or car rental inventory.
  - Can view bookings related to their services but cannot access unrelated data.
- **Booking Rules:**
  - Double-booking of the same resource (e.g., room or vehicle) for overlapping times is not allowed.
- **Payment Policies:**
  - All bookings require confirmation through payment. Payment status must be marked as "Paid" for bookings to be finalized.

3.6 Operating Environment

Hardware Platform:
The application is designed to run on personal computers or laptops.

Software Requirements:
- Java Runtime Environment (JRE): Java 11 or higher installed on the host machine.
- JavaFX SDK: JavaFX 11 or later for GUI development and execution.
- Database Server: Microsoft SQL Server (SQL Server Express or higher) with required drivers configured.

Development Environment:
Developed and tested on:
- IDE: Eclipse IDE for Java Developers with e(fx)clipse plugin installed.
- JavaFX Scene Builder for designing the graphical user interface

3.7 User Interfaces

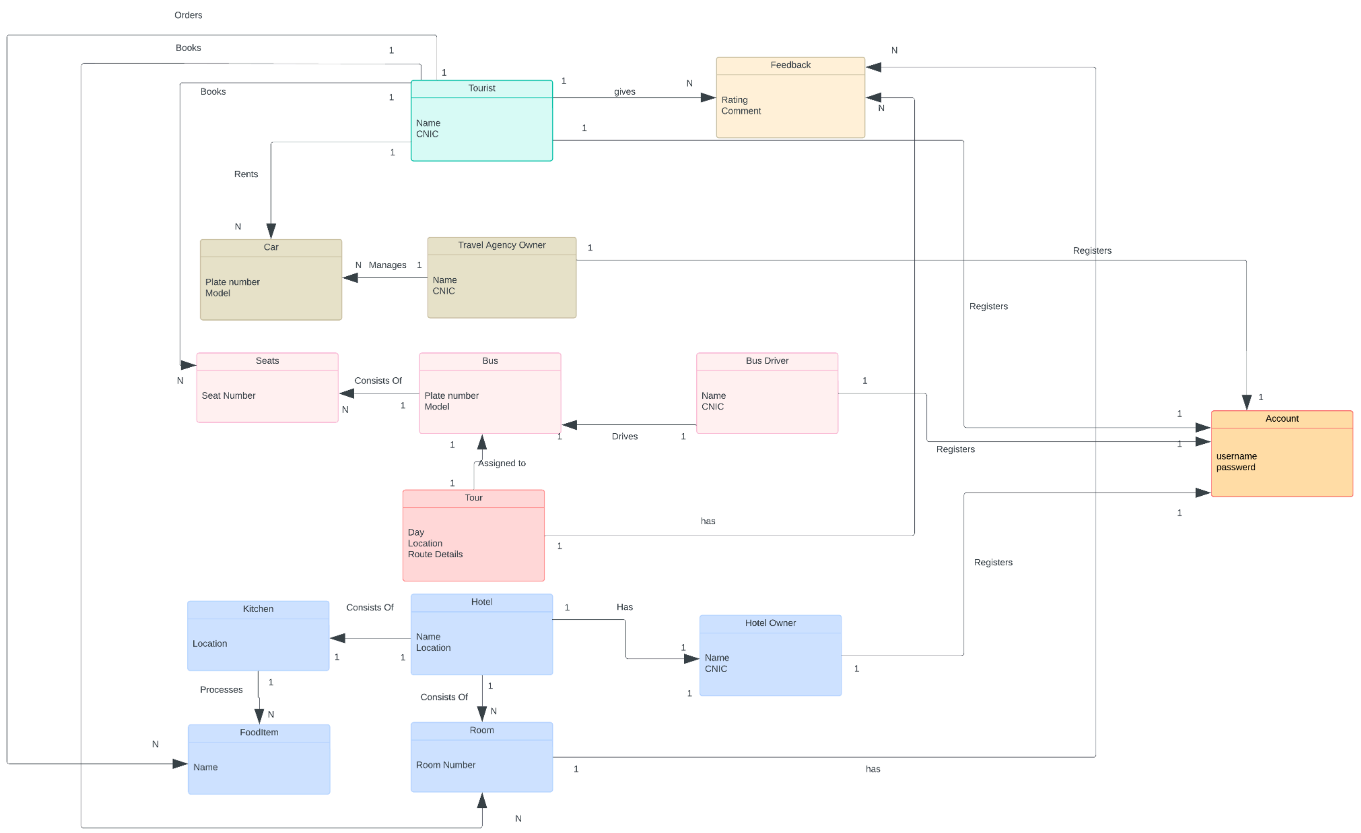
System provides a graphical user interface (GUI) designed to be intuitive, user-friendly, and visually appealing. Below is an overview of the logical characteristics of the user interfaces:

GUI Standards:
- Consistent font styles, colors, and themes across all screens.
- Simple and minimalistic layout with clear navigation menus.

Home Screen
Provide a dashboard view summarizing key actions.

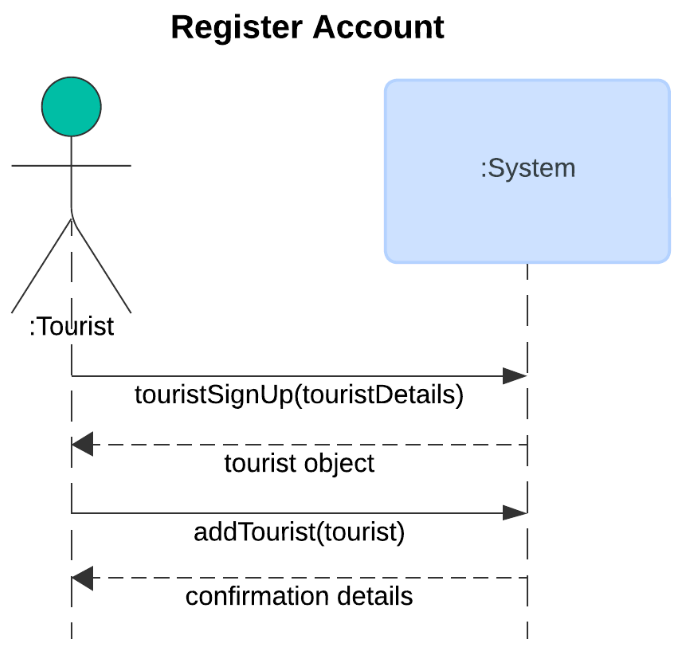
Error Standards:
Errors are displayed in red, with a concise and clear description (e.g., "Invalid date selected.").

4. Domain Model

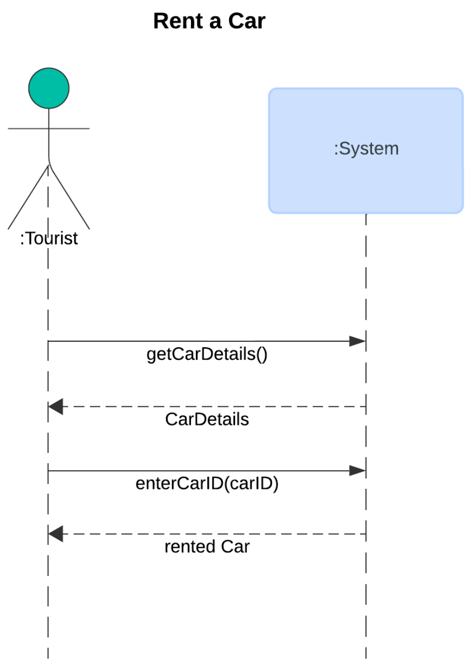


5. System Sequence Diagram

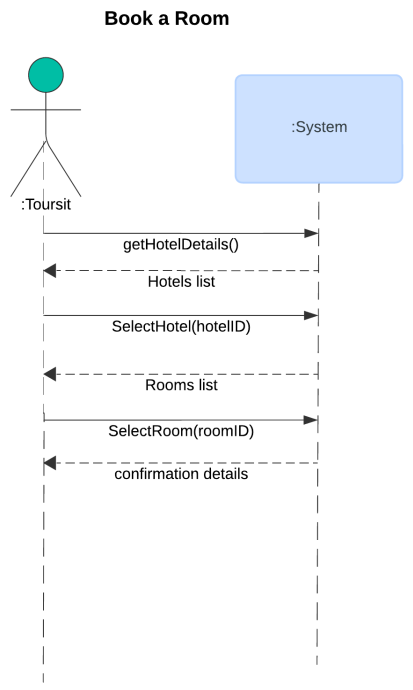
Register account



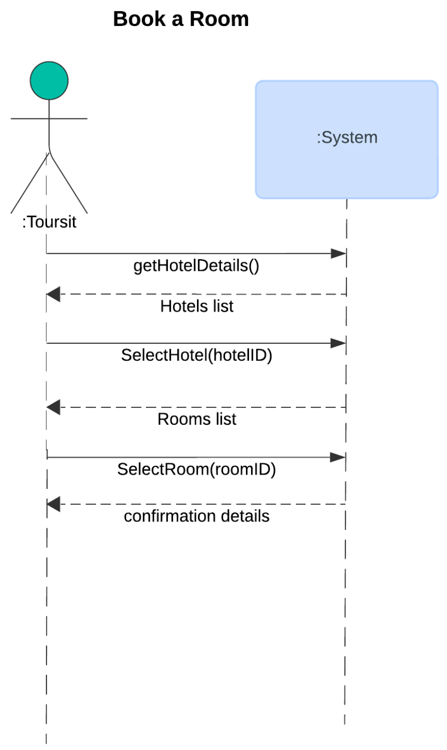
Rent Car



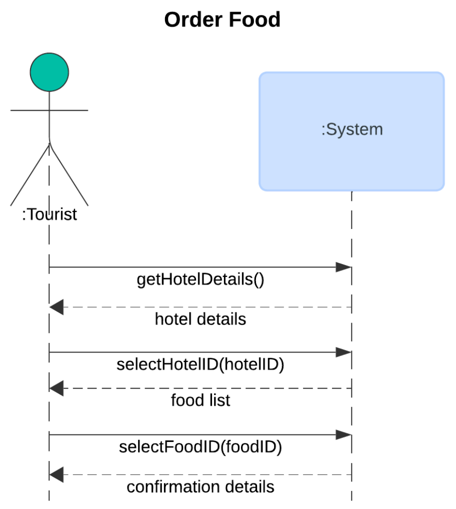
Book a Room



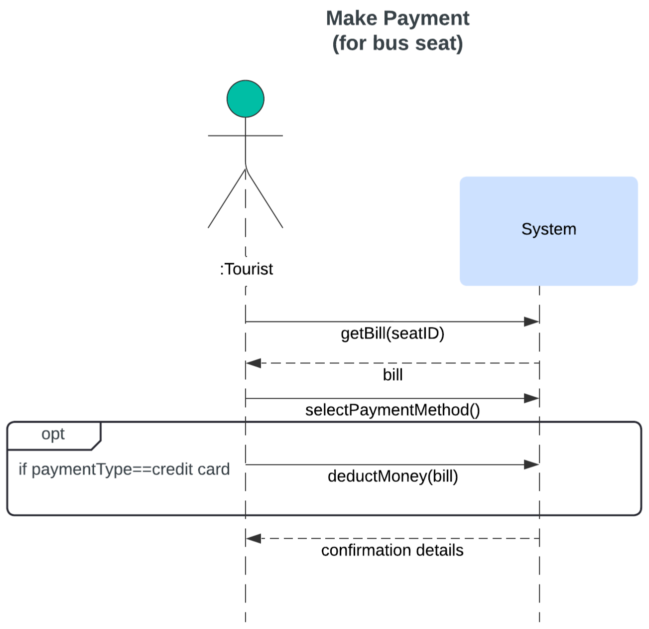
Book Seat in Bus



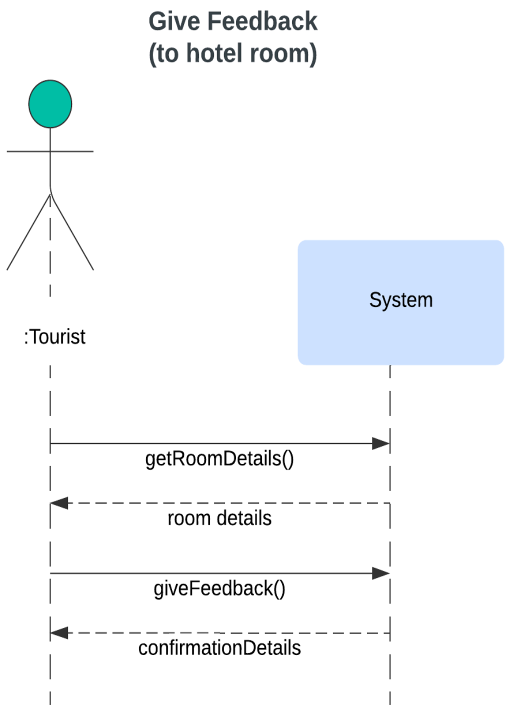
Order Food



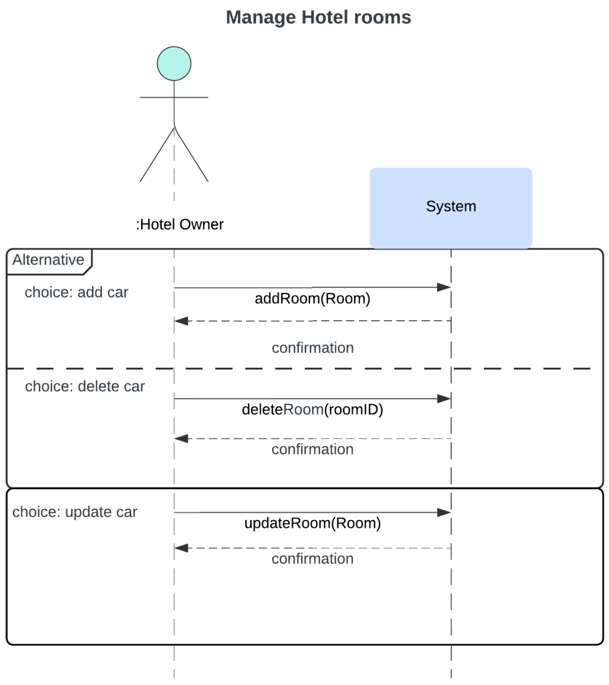
Make Payment



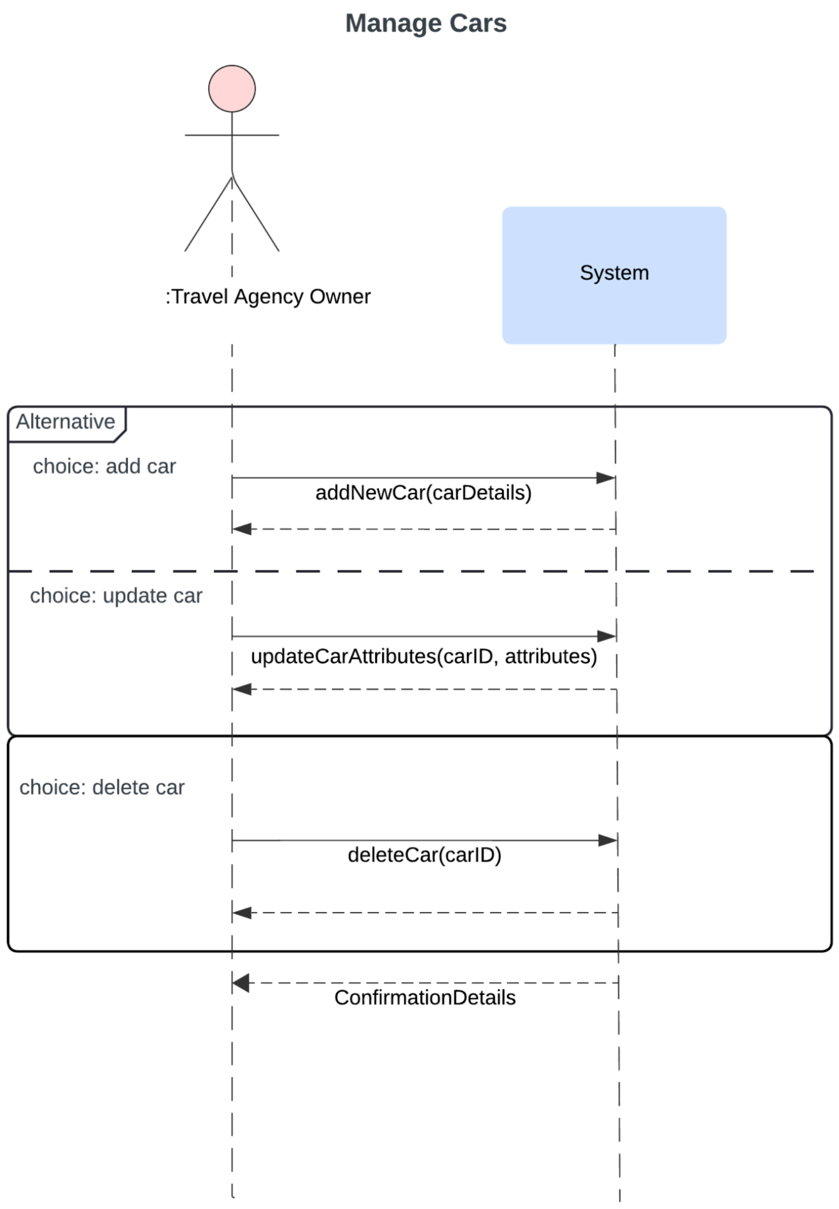
Give Feedback



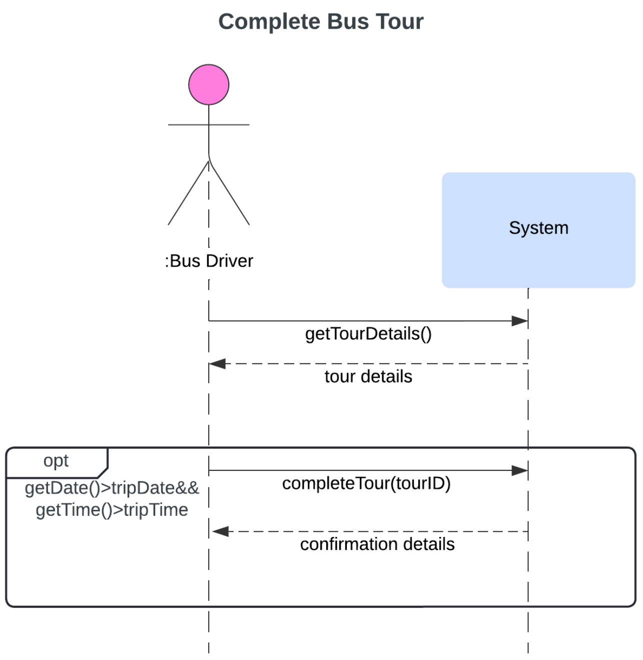
Manage Hotel Rooms



Manage Cars

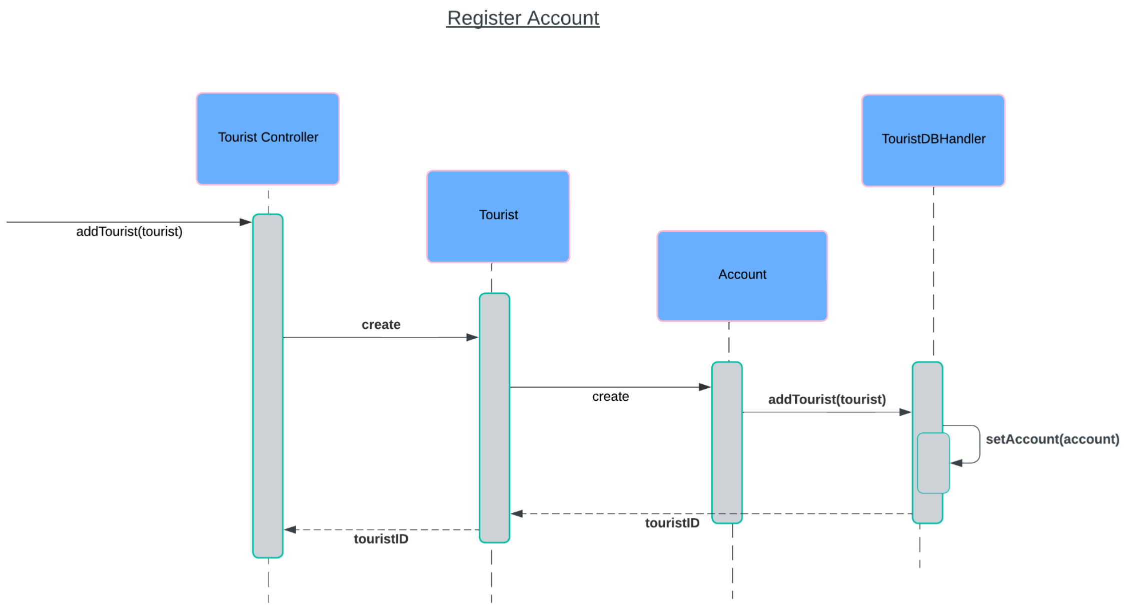

Manage Cars
:Travel Agency Owner
System
Alternative
choice: add car
addNewCar(carDetails)
choice: update car
updateCarAttributes(carID, attributes)
choice: delete car
deleteCar(carID)
ConfirmationDetails

Complete Bus Ride

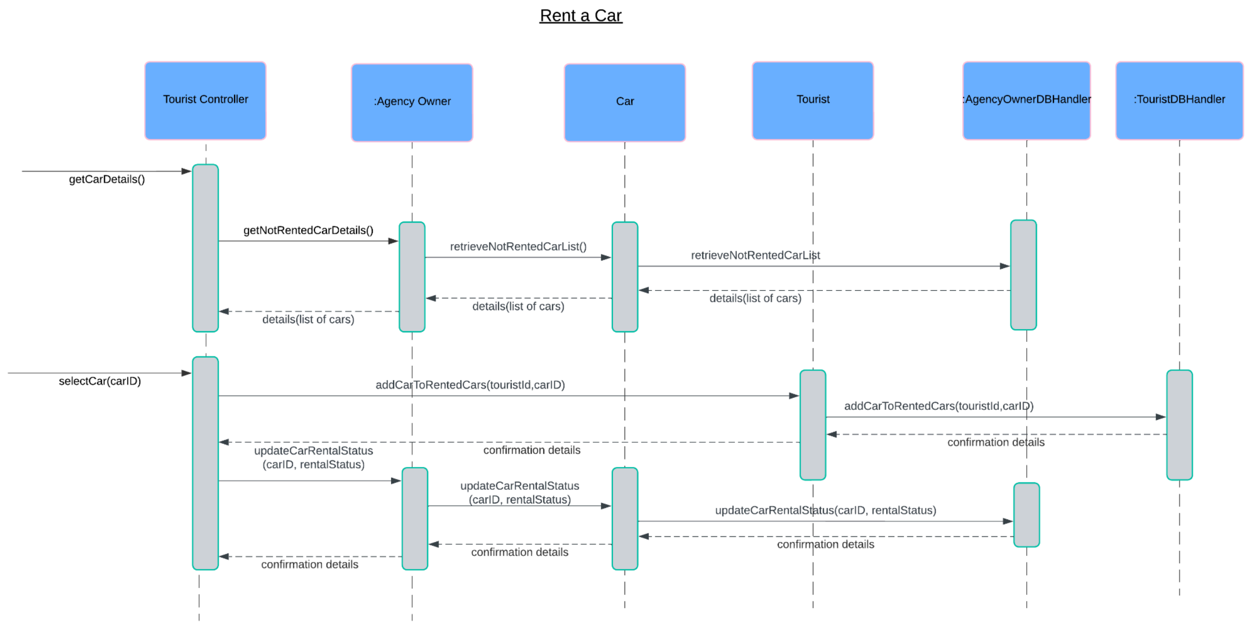


6. Sequence Diagram

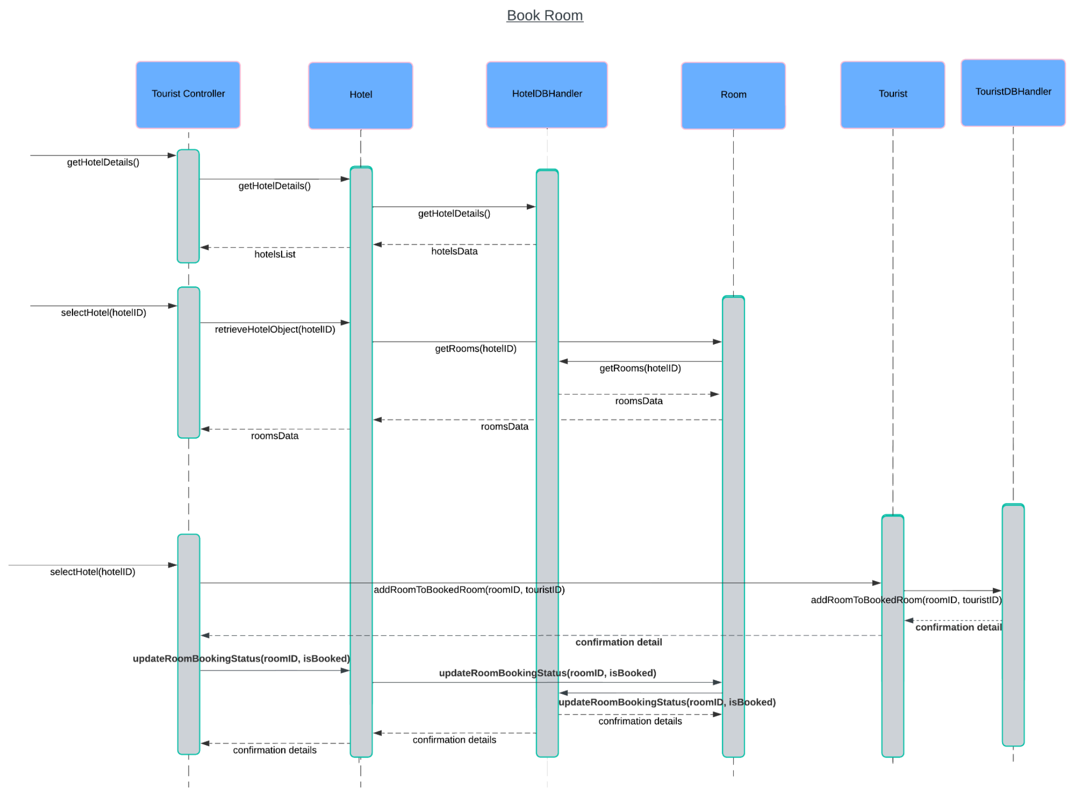
Register Account



Rent Car

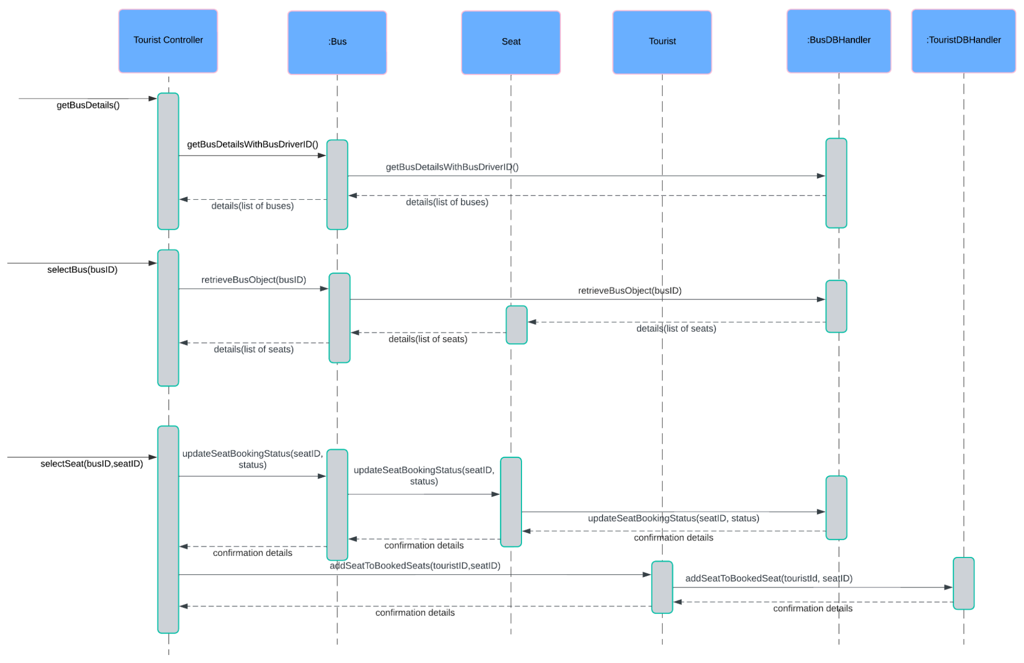


Book a Room



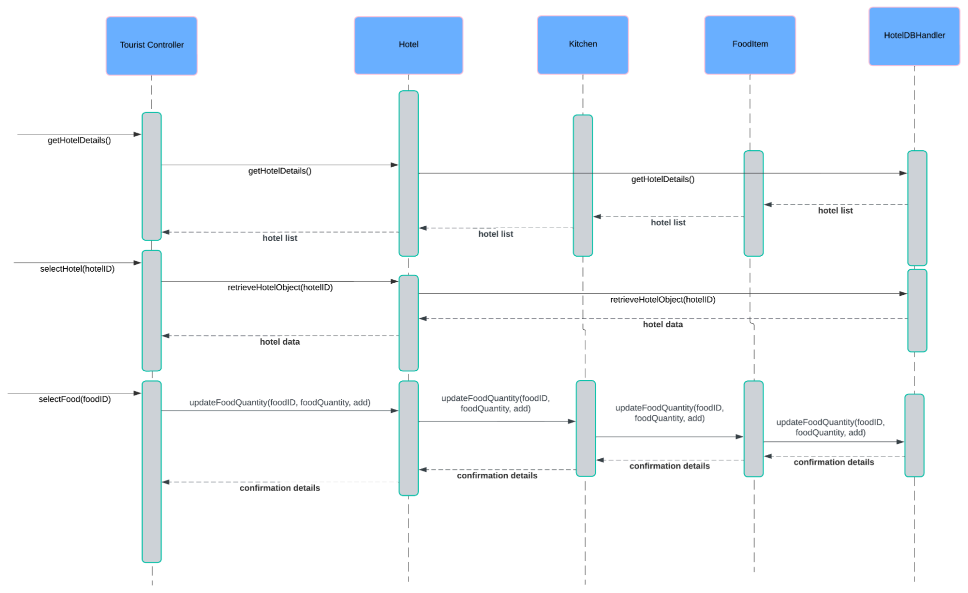
Book Seat in Bus

Book Bus Seat

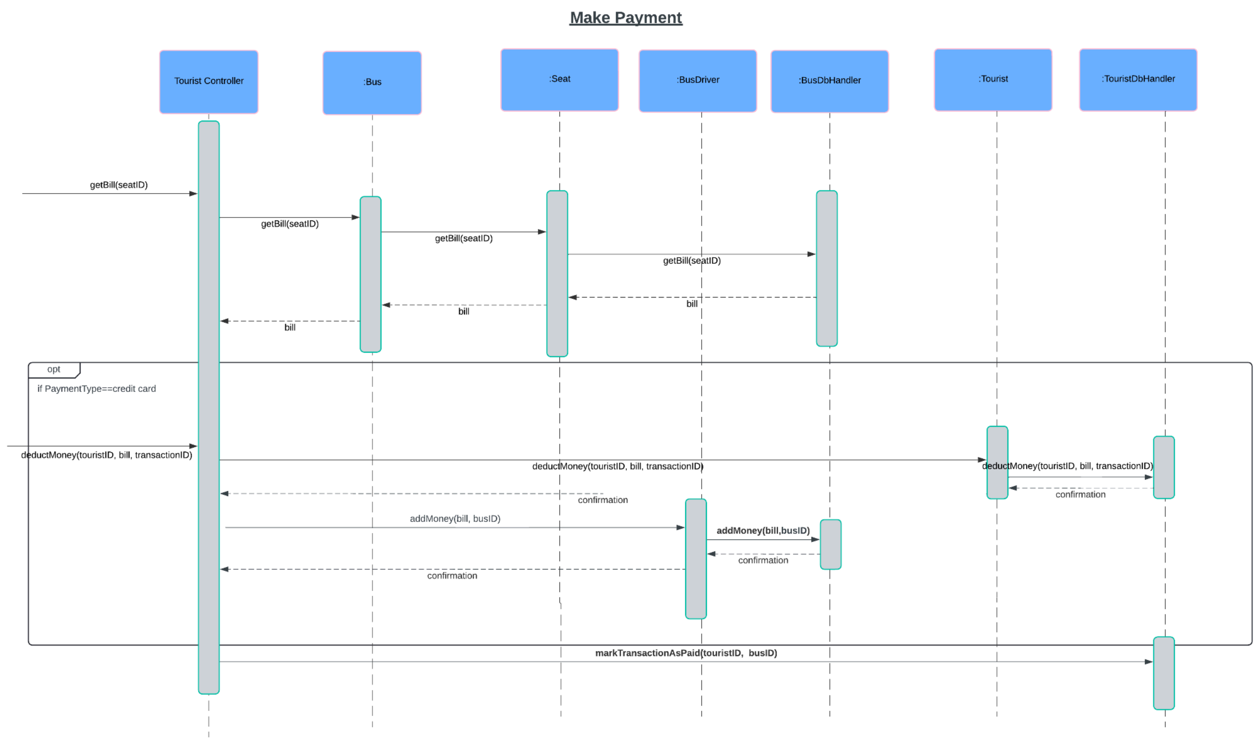


Order Food

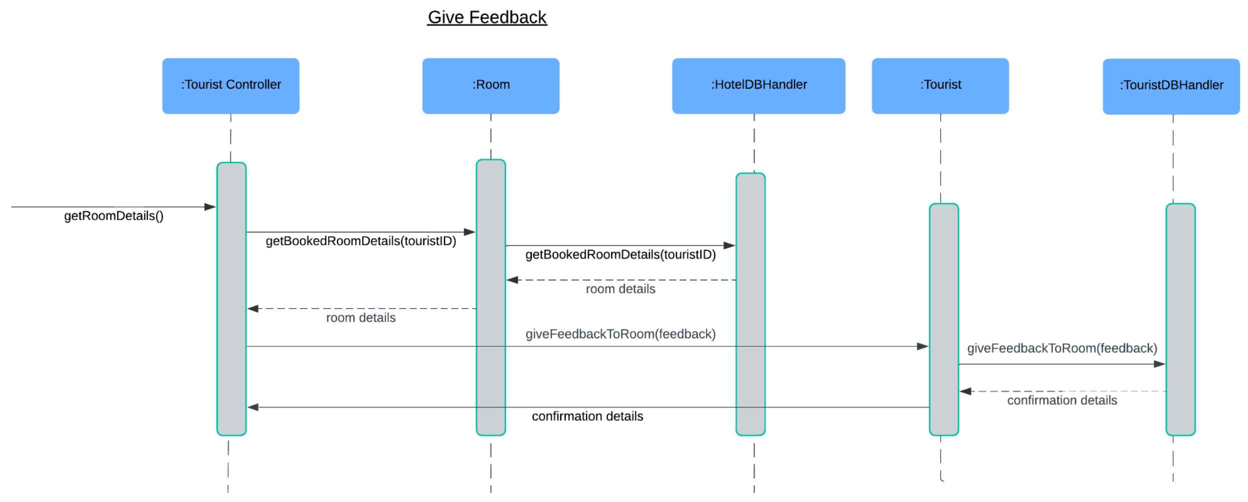
Order food



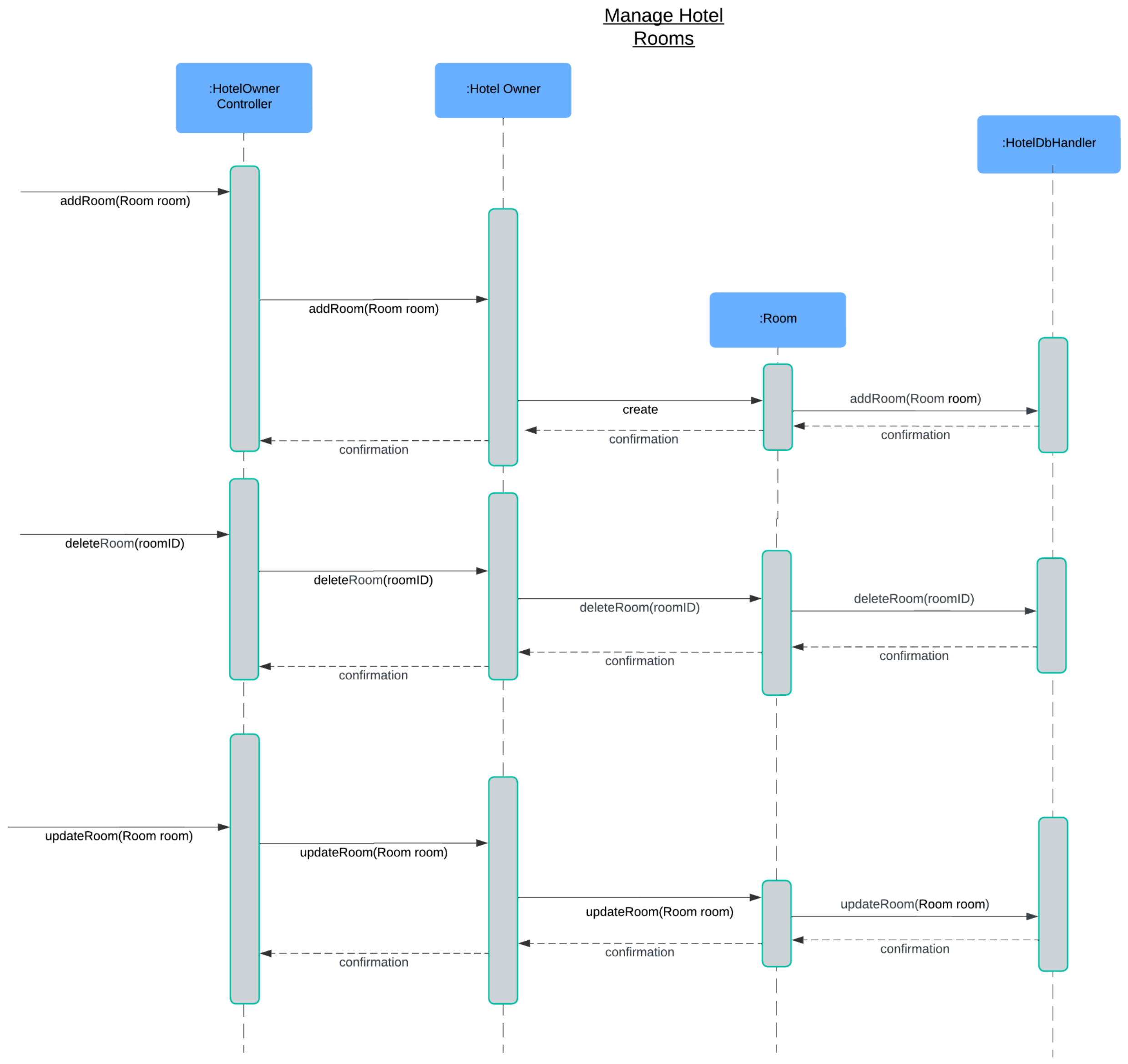
Make Payment



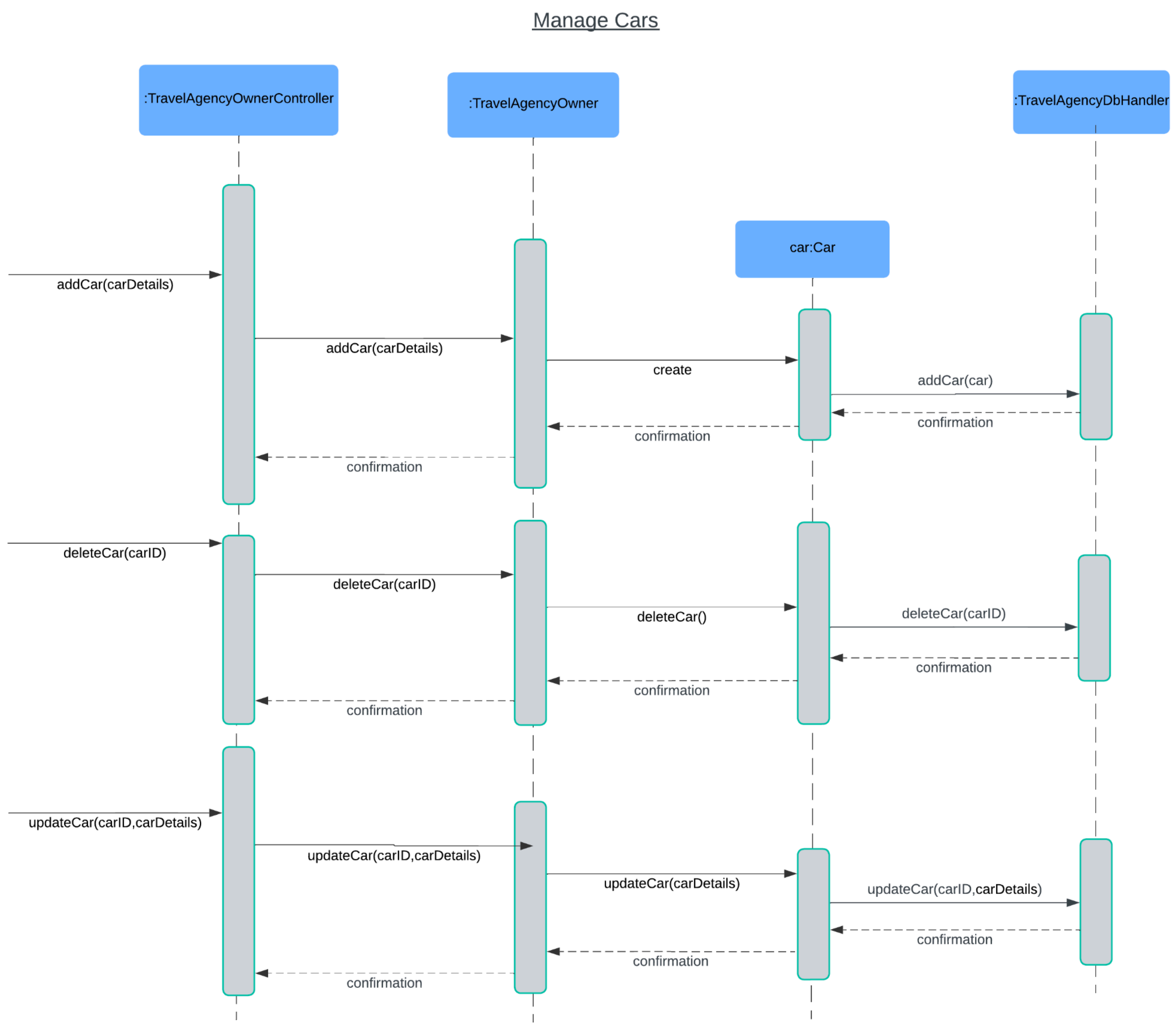
Give Feedback



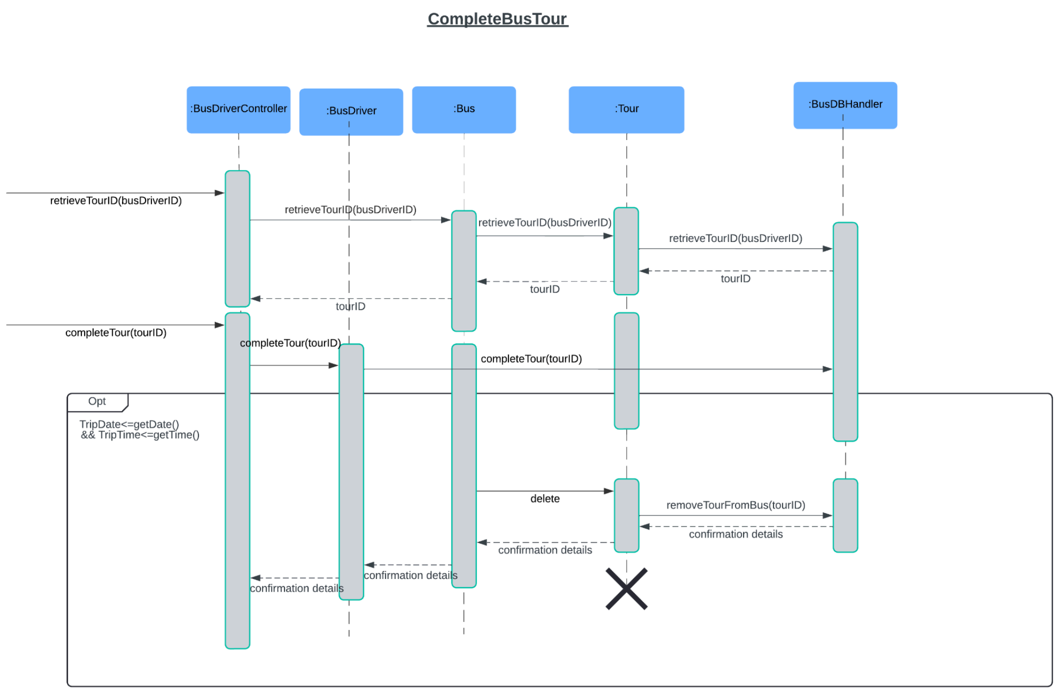
Manage Hotel Rooms



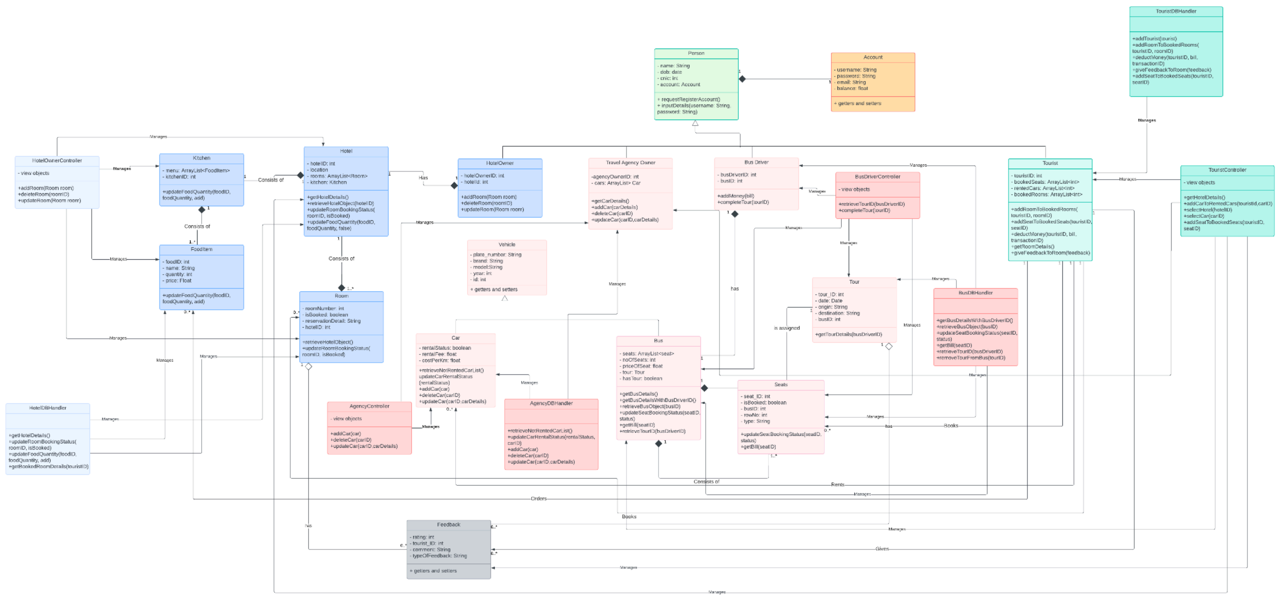
Manage Cars


Manage Cars
:TravelAgencyOwnerController
:TravelAgencyOwner
car:Car
:TravelAgencyDbHandler
addCar(carDetails)
addCar(carDetails)
create
addCar(car)
confirmation
confirmation
confirmation
deleteCar(carID)
deleteCar(carID)
deleteCar()
deleteCar(carID)
confirmation
confirmation
confirmation
updateCar(carID,carDetails)
updateCar(carID,carDetails)
updateCar(carDetails)
updateCar(carID,carDetails)
confirmation
confirmation
confirmation

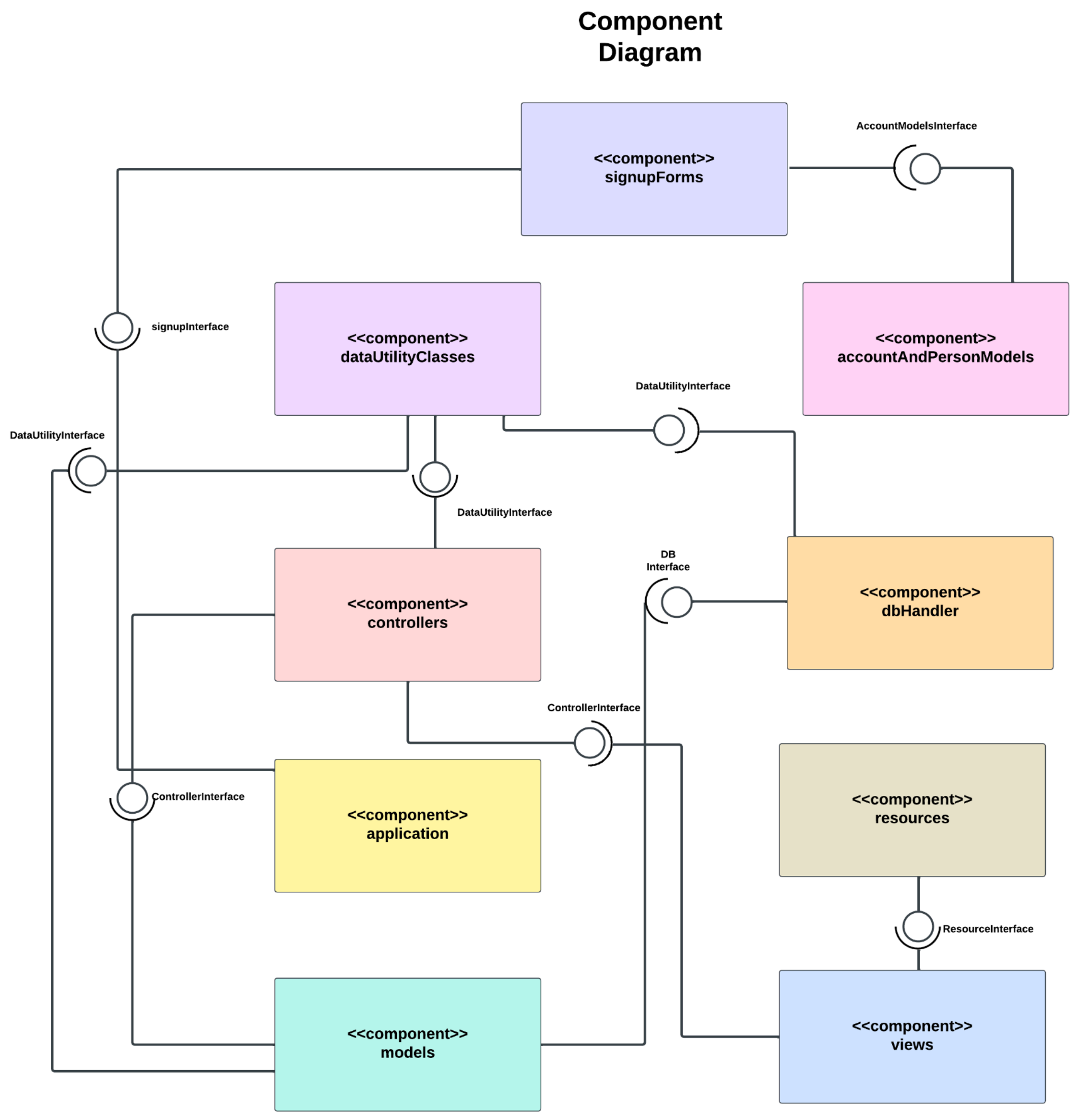
Complete Bus Ride



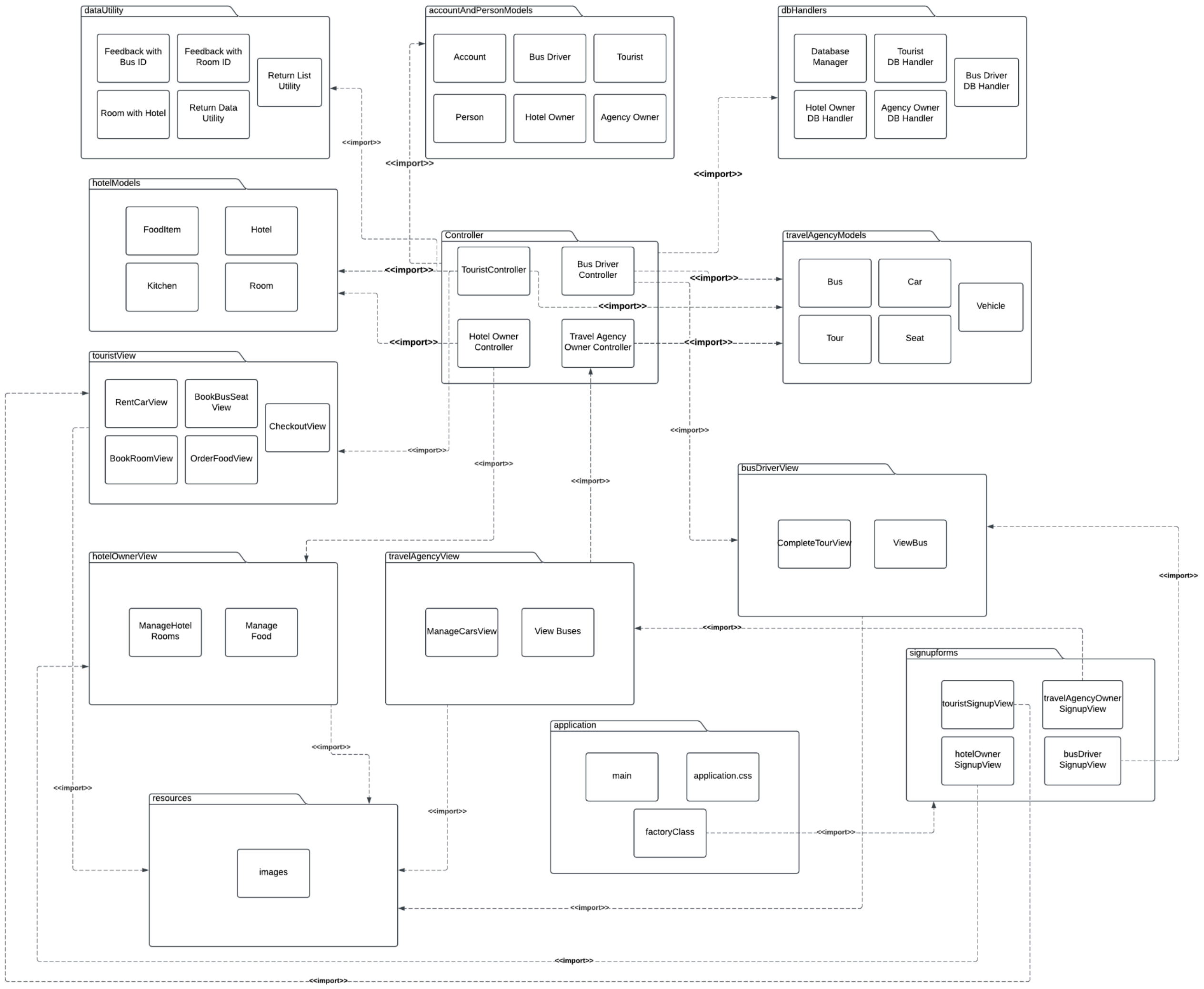
7. Class Diagram



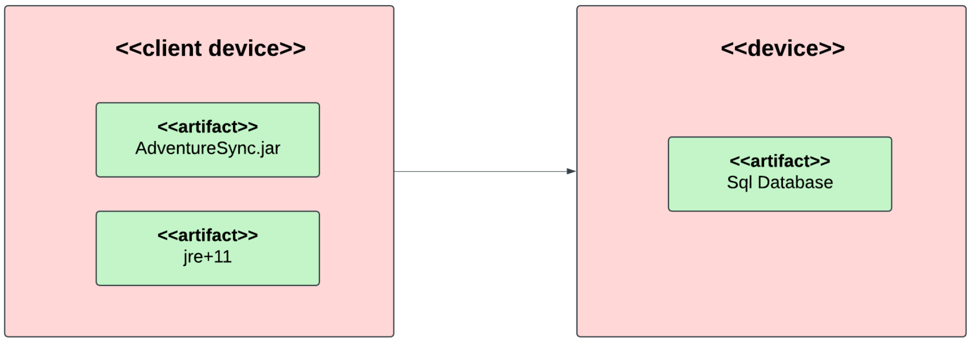
8. Component Diagram



9. Package Diagram



10. Deployment Diagram



<<client device>>
<<artifact>>
AdventureSync.jar
<<artifact>>
jre+11
<<device>>
<<artifact>>
Sql Database