



## ***UE21CS352B - Object Oriented Analysis & Design using Java***

### **Mini Project Report**

### **“Ezigo Car Rental Service”**

*Submitted by:*

***Srinidhi Somayaji P PES1UG21CS622***

***Srujan Muralidhar PES1UG21CS627***

***Srikanth Adithiyaa PES1UG21CS617***

***Trisha Songra PES1UG21CS677***

*6<sup>th</sup> Semester K Section*

**Bhargavi Mokashi**

**Assistant Professor**

**Dept. of CSE**

**January - May 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**FACULTY OF ENGINEERING**

**PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

# Table of Contents

PROBLEM STATEMENT .....	3
MODELS: .....	4
USE CASE DIAGRAM .....	4
CLASS DIAGRAM .....	5
STATE DIAGRAMS .....	6
ACTIVITY DIAGRAM .....	7
ARCHITECTURAL PATTERN .....	8
DESIGN PATTERNS .....	9
CREATIONAL DESIGN PATTERN .....	9
1. Singleton Pattern with @Service Annotation: .....	9
2. Factory Pattern for Creating Different Car Types: .....	9
STRUCTURAL DESIGN PATTERN .....	9
1. Facade Pattern for Separating Controller and Service: .....	9
GITHUB REPOSITORY .....	10
SCREENSHOTS: .....	10

## PROBLEM STATEMENT

Ezigo Car Rental Service aims to address the inconvenience and inefficiencies associated with traditional car rental processes. Despite the widespread availability of car rental services, customers often encounter challenges such as lengthy paperwork, limited vehicle options, unclear pricing structures, and inadequate customer support. Additionally, car rental companies struggle with managing their fleet effectively, optimizing utilization rates, and providing a seamless booking experience.

### Description:

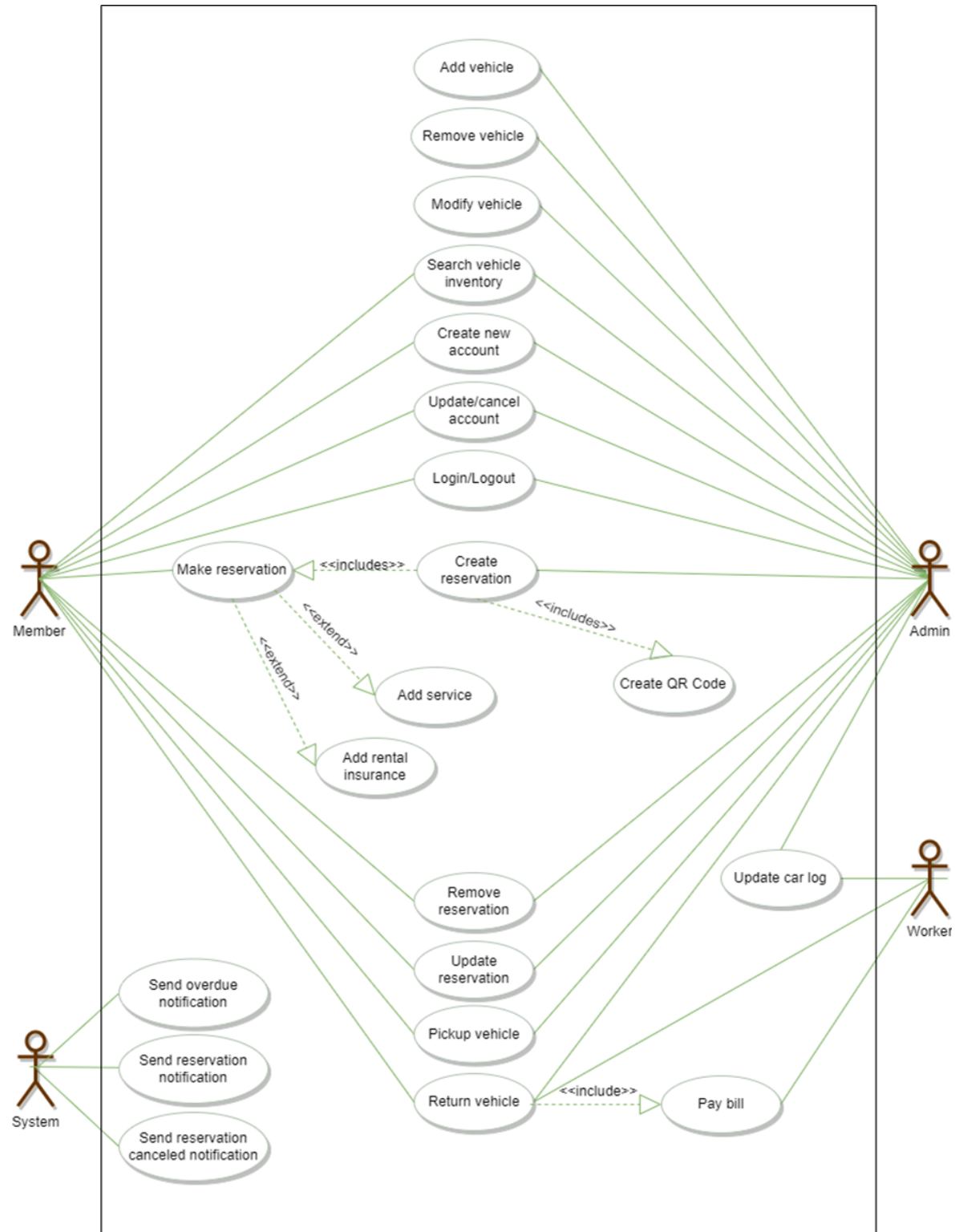
Ezigo Car Rental Service seeks to revolutionize the car rental industry by offering a user-friendly platform that streamlines the entire rental process, from booking to returning the vehicle. Through our innovative approach, customers will enjoy the convenience of accessing a diverse fleet of well-maintained vehicles at competitive prices, all while benefiting from transparent pricing and flexible booking options. Our platform will prioritize customer satisfaction by providing round-the-clock support and ensuring a hassle-free rental experience.

### Key Features:

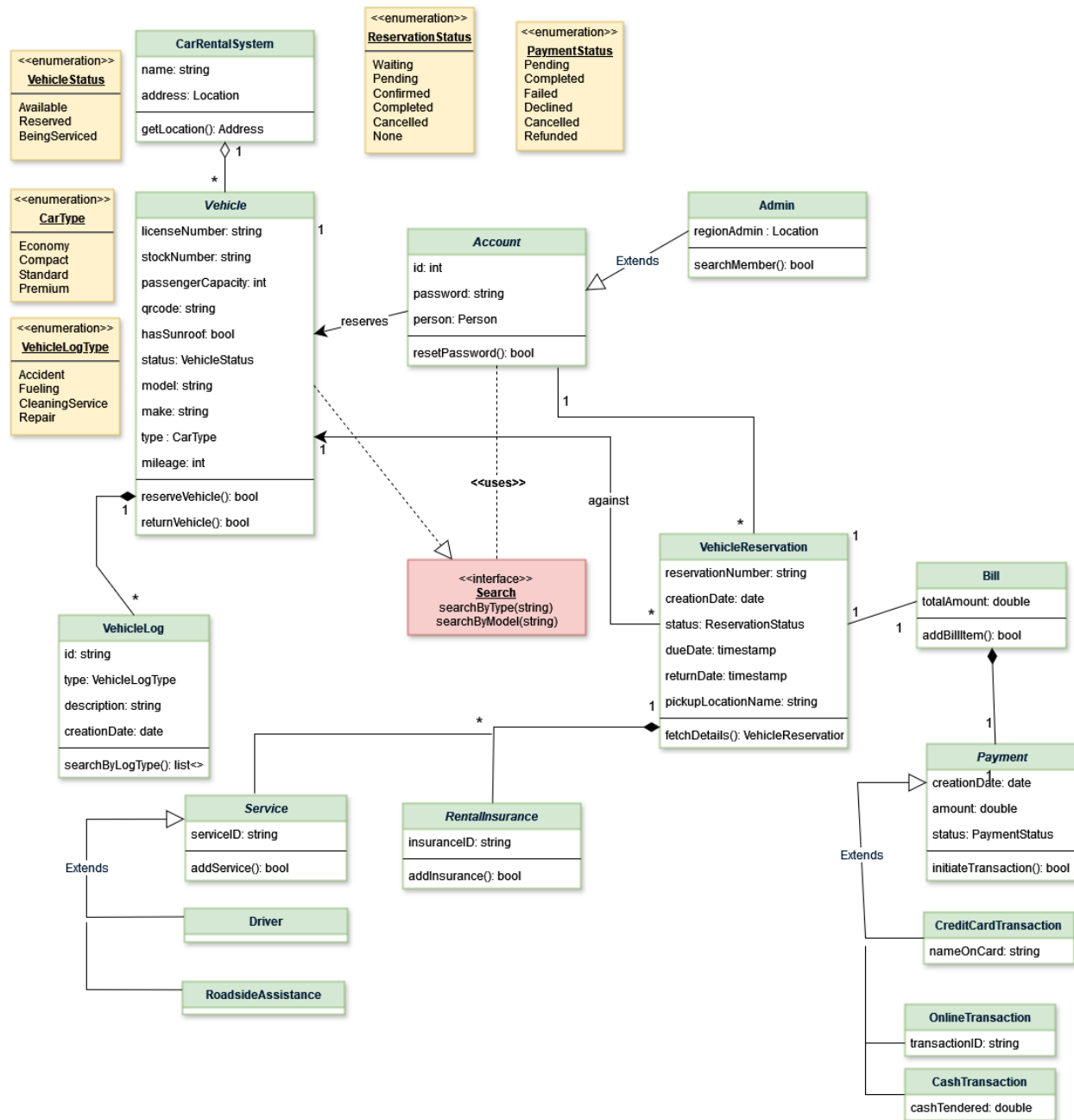
1. **User-friendly Booking System:** Our intuitive online platform and mobile application will allow customers to easily search for available vehicles, select pick-up and drop-off locations, and complete bookings in just a few clicks.
2. **Diverse Fleet:** We offer a wide range of vehicles to cater to various needs and preferences, including economy cars, SUVs, luxury vehicles, and more, ensuring that customers can find the perfect vehicle for their requirements.
3. **Transparent Pricing:** We believe in transparency, which is why we provide clear and upfront pricing with no hidden fees or surprises. Customers can view rental rates, additional charges, and any applicable discounts before confirming their booking.
4. **Flexible Booking Options:** Whether customers need a vehicle for a few hours, days, or weeks, our flexible booking options allow them to customize their rental period according to their schedule.
5. **Convenient Pickup and Return:** We offer convenient pick-up and return options at multiple locations, including airports, city centers, and popular tourist destinations, to ensure that customers can easily access and return their rental vehicles.

## MODELS:

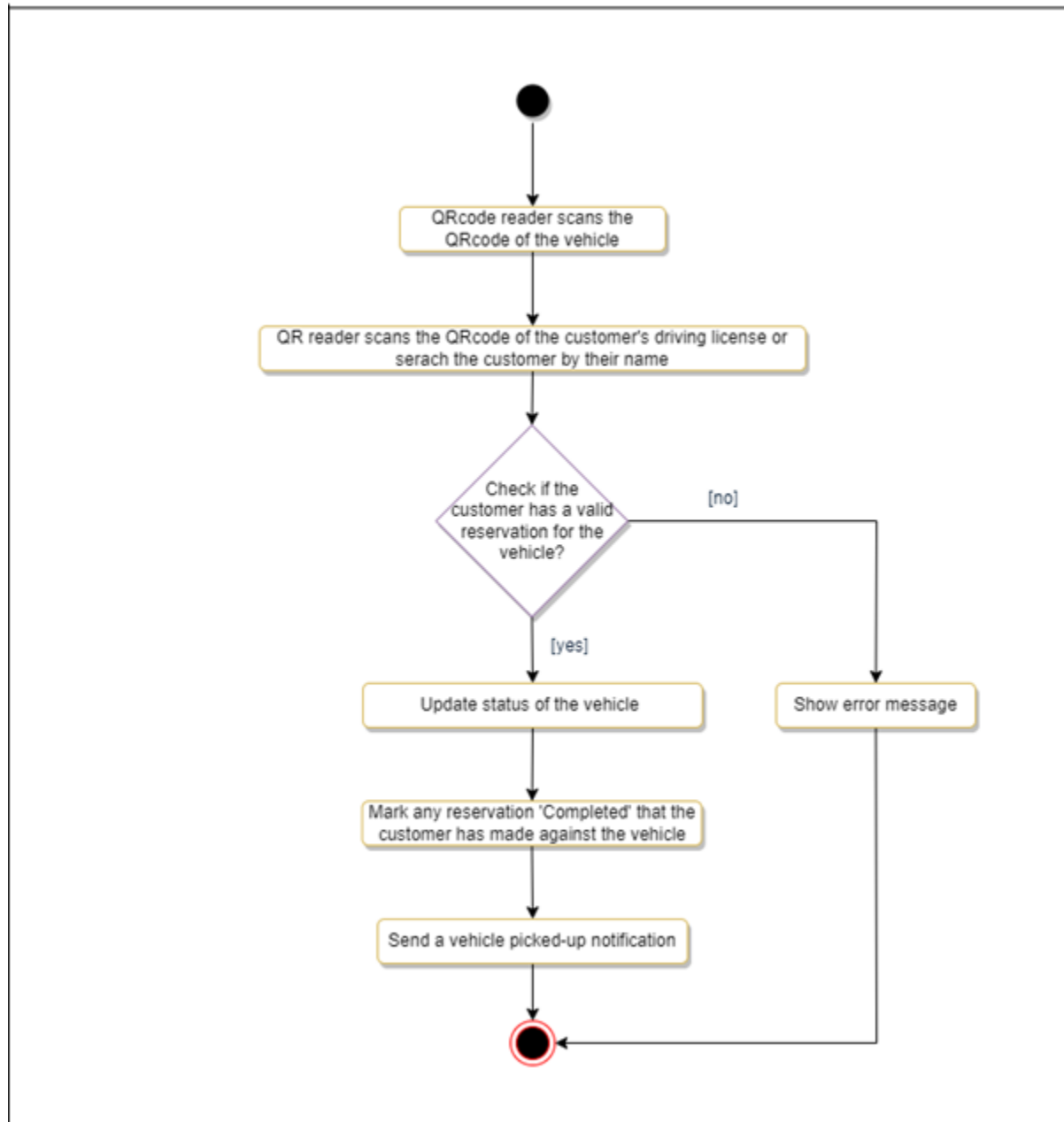
### USE CASE DIAGRAM



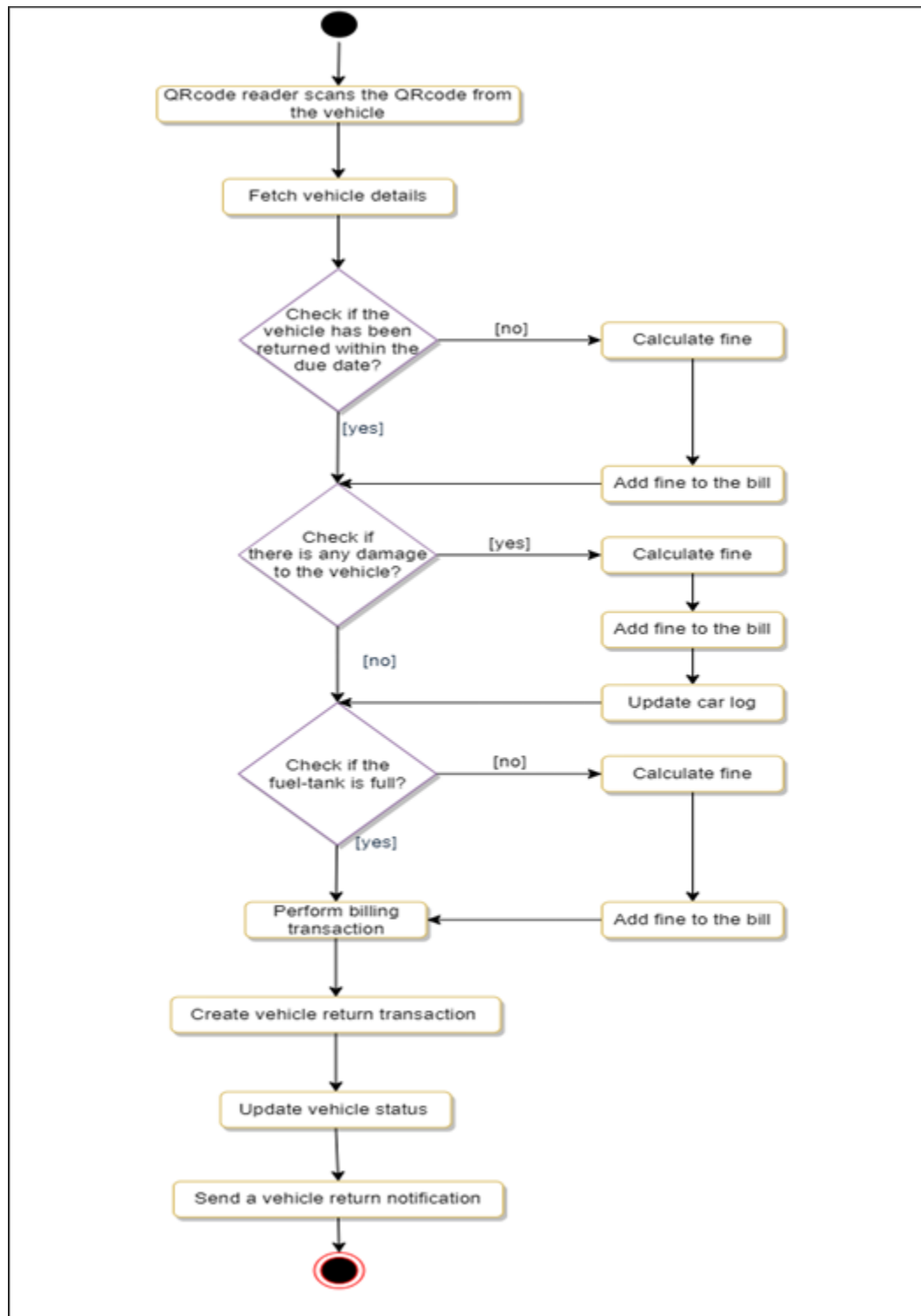
# CLASS DIAGRAM



## STATE DIAGRAMS



## ACTIVITY DIAGRAM



## ARCHITECTURAL PATTERN

In our Ezigo Car Rental Service project, implementing the Model-View-Controller (MVC) architecture can significantly enhance the organization, scalability, and maintainability of our application. Here's how we utilized MVC in our project:

### Model (M):

- The Model represents the core business logic and data of our car rental service. This includes classes and components responsible for managing data storage, retrieval, and manipulation.
- In our project, the Model layer would handle tasks such as managing the rental fleet database, handling customer information, processing bookings, and calculating rental charges.
- It encapsulates the application's data structures and business rules, ensuring data integrity and consistency.

### View (V):

- The View layer represents the user interface components of our application. It encompasses everything that users interact with, including web pages, forms, menus, and other graphical elements.
- In the context of our car rental service, the View layer would include the user interface elements displayed on our website. This could include pages for searching and browsing available vehicles, booking forms, account management screens, and rental confirmation pages.
- Views are responsible for presenting data to users in a clear and intuitive manner, facilitating seamless interactions with the application.

### Controller (C):

- The Controller acts as the intermediary between the Model and the View layers. It receives input from users via the View, processes it using the appropriate Model components, and updates the View accordingly.
- In our project, Controllers would handle user requests and orchestrate the flow of data between the Model and View layers. They interpret user input, invoke the necessary business logic in the Model layer, and determine which View should be presented to the user in response.
- Controllers play a crucial role in implementing application logic, handling user authentication, validating input data, and managing the overall application workflow.



# DESIGN PATTERNS

## CREATIONAL DESIGN PATTERN

### 1. Singleton Pattern with @Service Annotation:

- The Singleton pattern ensures that a class has only one instance and provides a global point of access to that instance. In Java Spring framework, the @Service annotation serves as a Singleton by default.
- In Ezigo Car Rental Service, Singleton pattern with @Service annotation is utilized for components that should have a single shared instance throughout the application's lifecycle.

### 2. Factory Pattern for Creating Different Car Types:

- The Factory pattern provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created. This promotes loose coupling between client code and the objects being created.
- In Ezigo Car Rental Service, the Factory pattern is employed to create different types of cars based on customer requirements or available inventory. For instance, a CarFactory class can have methods for creating economy, standard, compact and luxury cars.

## STRUCTURAL DESIGN PATTERN

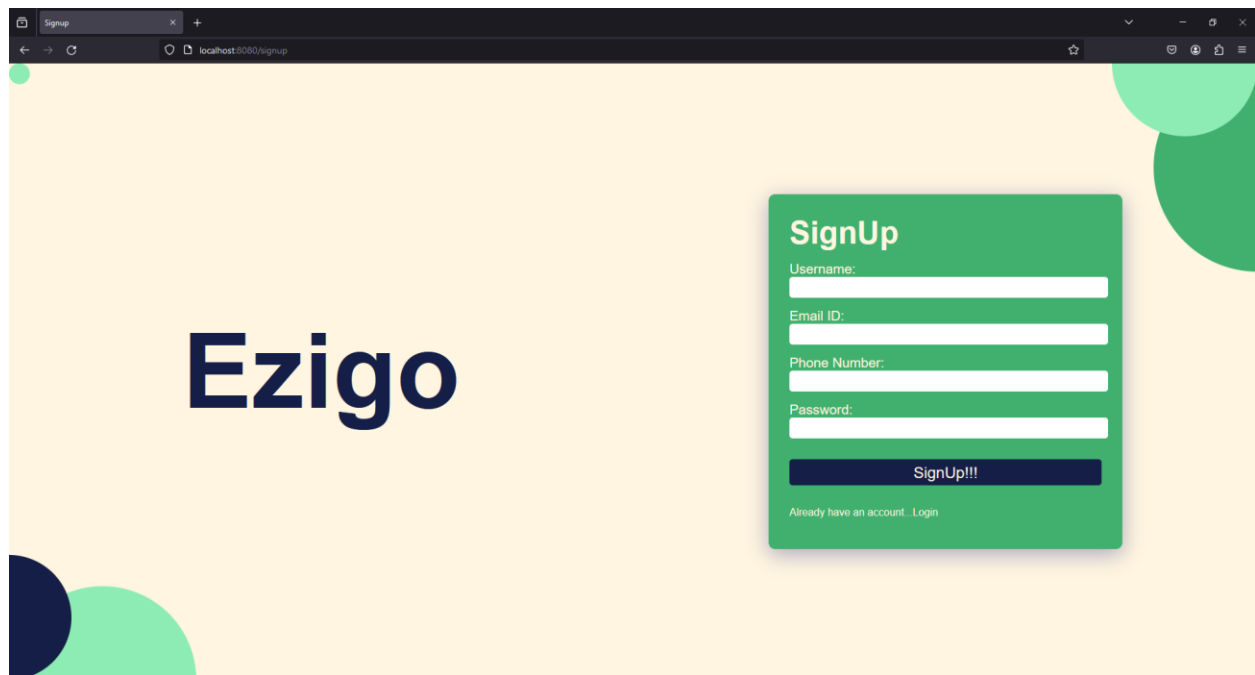
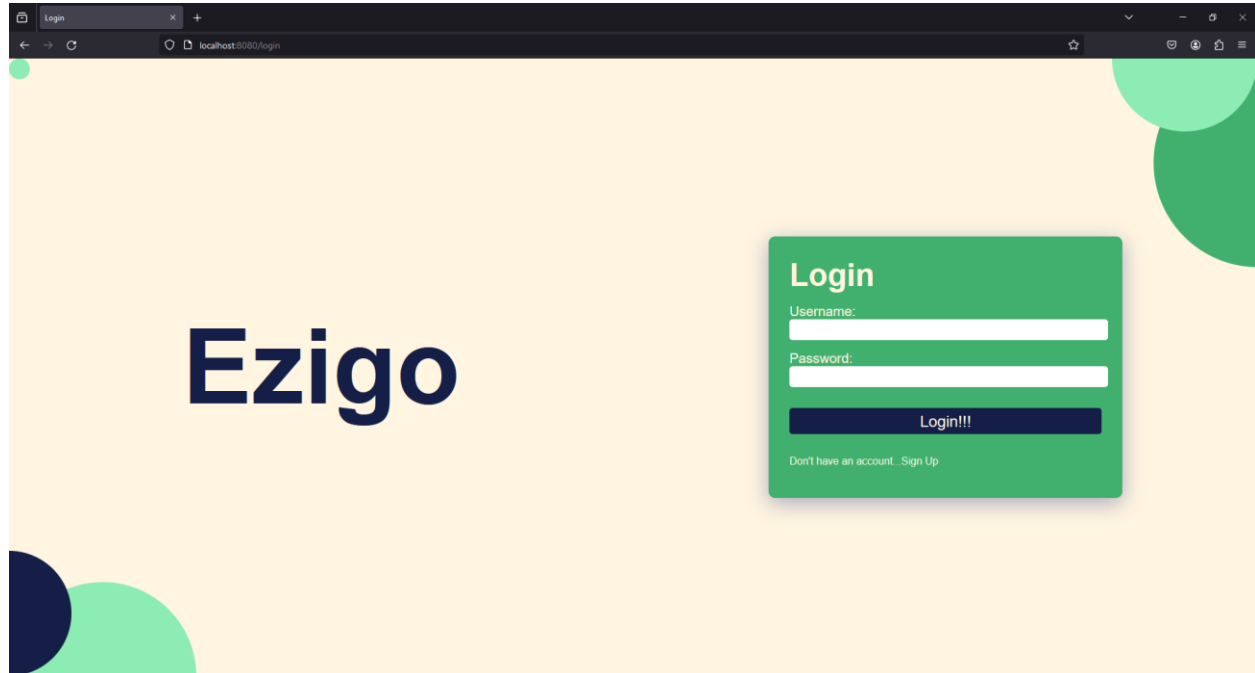
### 1. Facade Pattern for Separating Controller and Service:

- The Facade pattern provides a unified interface to a set of interfaces in a subsystem. It defines a higher-level interface that makes the subsystem easier to use, thus promoting loose coupling and simplifying client code.
- In the context of Ezigo Car Rental Service, the Facade pattern is employed to separate the concerns between the controller layer (responsible for handling user requests and responses) and the service layer (responsible for implementing business logic and data operations).
- By using a facade, the controller layer can interact with the service layer through a simplified interface, abstracting away the complexity of underlying service implementations. This separation of concerns enhances maintainability and allows for easier unit testing of both layers.

# GITHUB REPOSITORY

[Ezigo-Car-Rental-Service](#)

## SCREENSHOTS:



## List of different cars



The screenshot shows a web application interface for car rental. At the top, there is a dark blue navigation bar with a white 'Index' label. Below the navigation bar is a horizontal menu with six green buttons: 'All', 'Luxury', 'Compact', 'Economy', 'Premium', and 'Sedan'. The main content area has a light yellow background and displays three car listings. Each listing includes a car image, the car name, its category, mileage, capacity, price, and a 'Book Now' button.

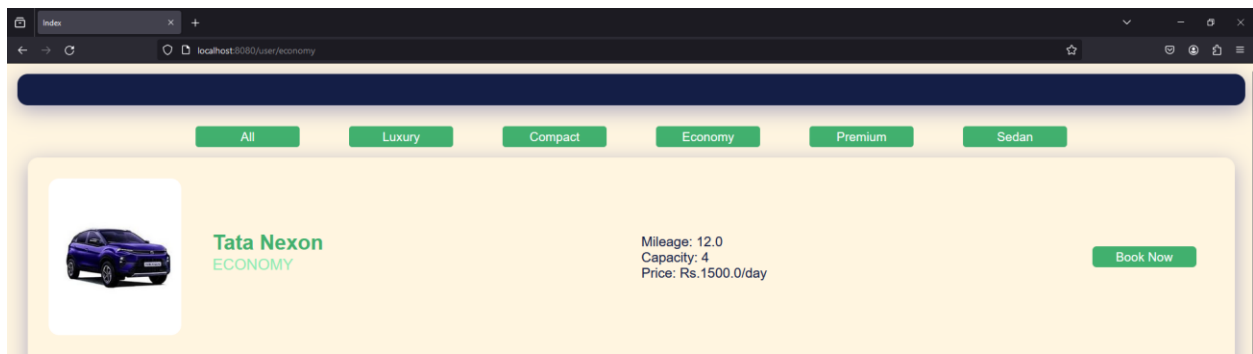
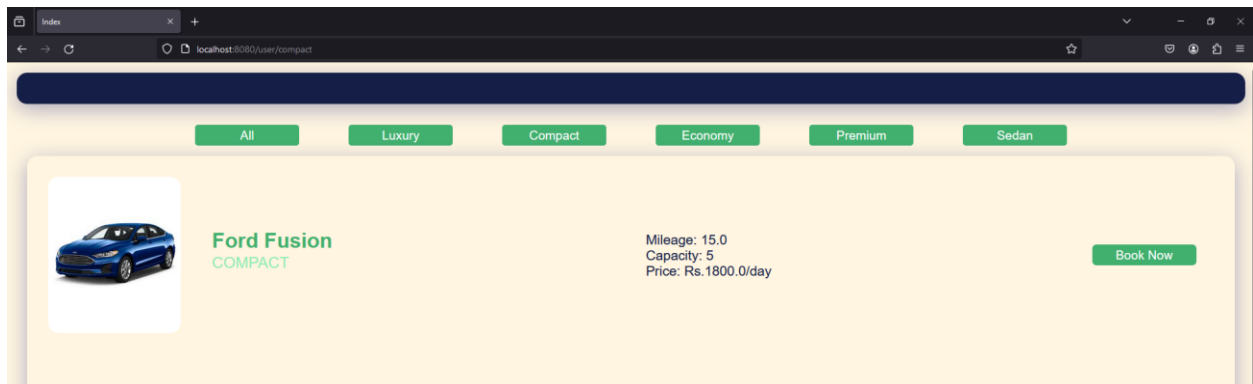
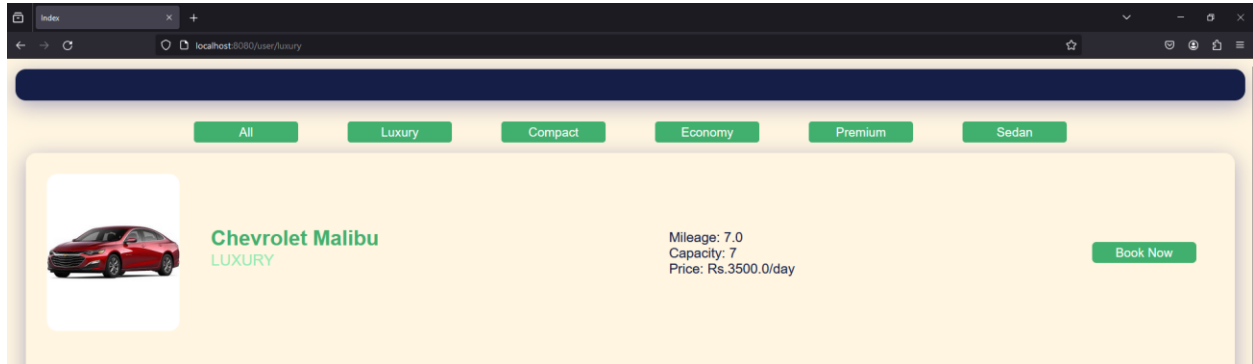
Car Image	Car Name	Category	Mileage	Capacity	Price	Action
	Toyota Corolla	SEDAN	12.5	5	Rs.2000.0/day	Book Now
	Tata Nexon	ECONOMY	12.0	4	Rs.1500.0/day	Book Now
	Honda Civic	SEDAN	11.0	5	Rs.2000.0/day	Book Now

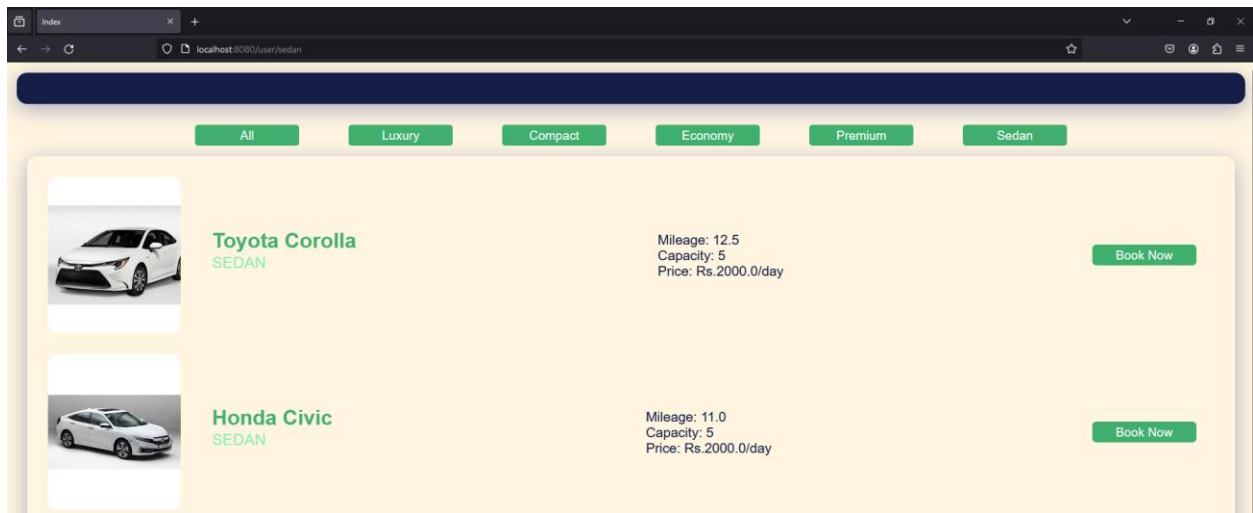


The screenshot shows the same web application interface as above, but with a different set of car listings. The navigation bar and menu are identical. The main content area displays three car listings: Honda Civic, Ford Fusion, and Chevrolet Malibu.

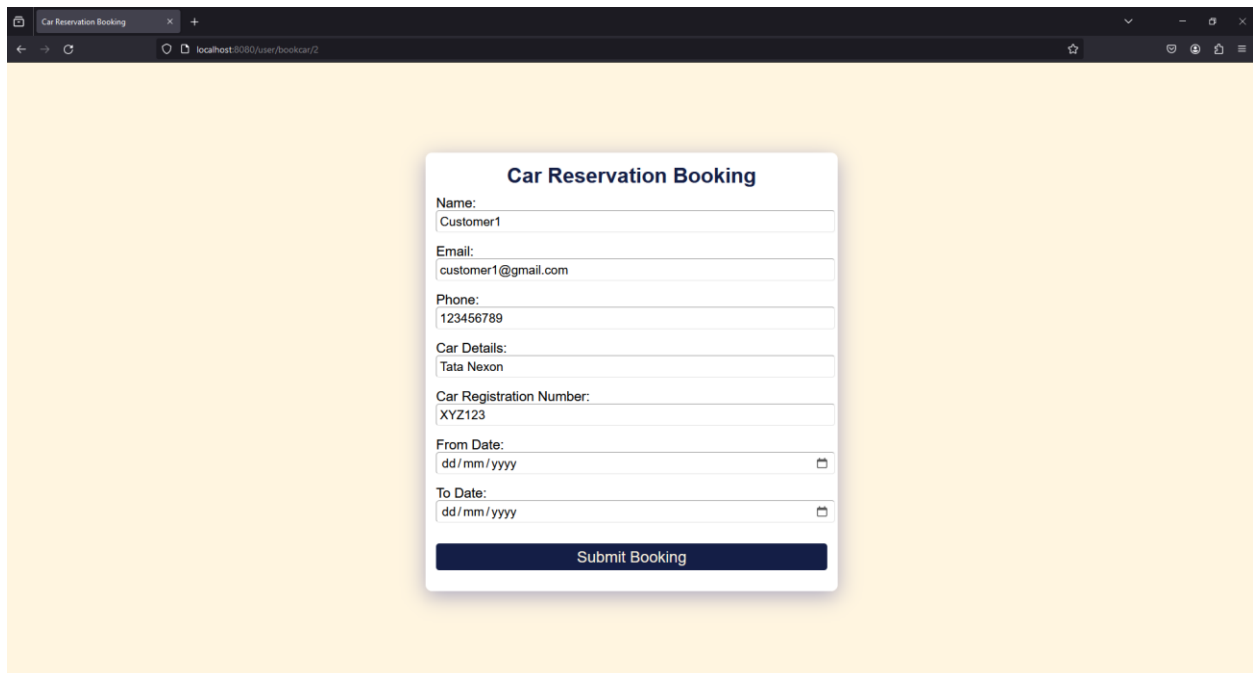
Car Image	Car Name	Category	Mileage	Capacity	Price	Action
	Honda Civic	SEDAN	11.0	5	Rs.2000.0/day	Book Now
	Ford Fusion	COMPACT	15.0	5	Rs.1800.0/day	Book Now
	Chevrolet Malibu	LUXURY	7.0	7	Rs.3500.0/day	Book Now

## Different car types





## Car reservation



The screenshot shows a web browser window with the URL `localhost:8080/user/bookcar/2`. The page displays a "Car Reservation Booking" form on a light orange background. The form includes fields for Name, Email, Phone, Car Details, Car Registration Number, From Date, and To Date, each with a corresponding input field. A "Submit Booking" button is located at the bottom of the form.

**Car Reservation Booking**

Name:

Email:

Phone:

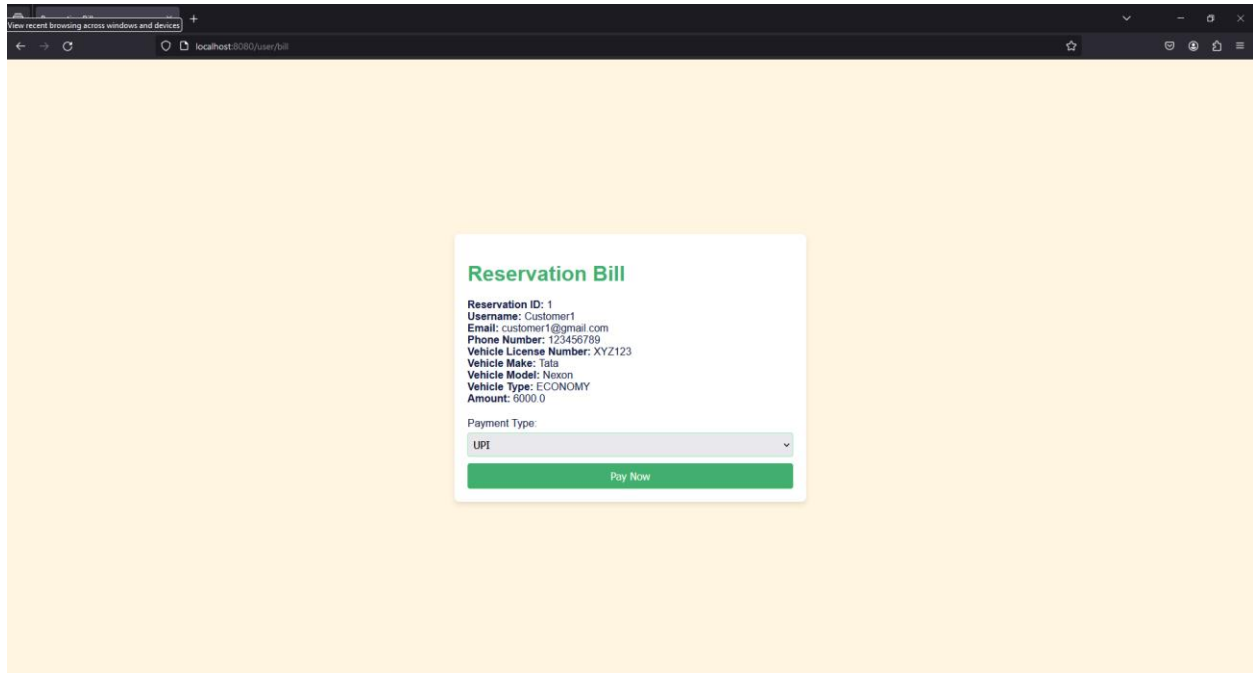
Car Details:

Car Registration Number:

From Date:

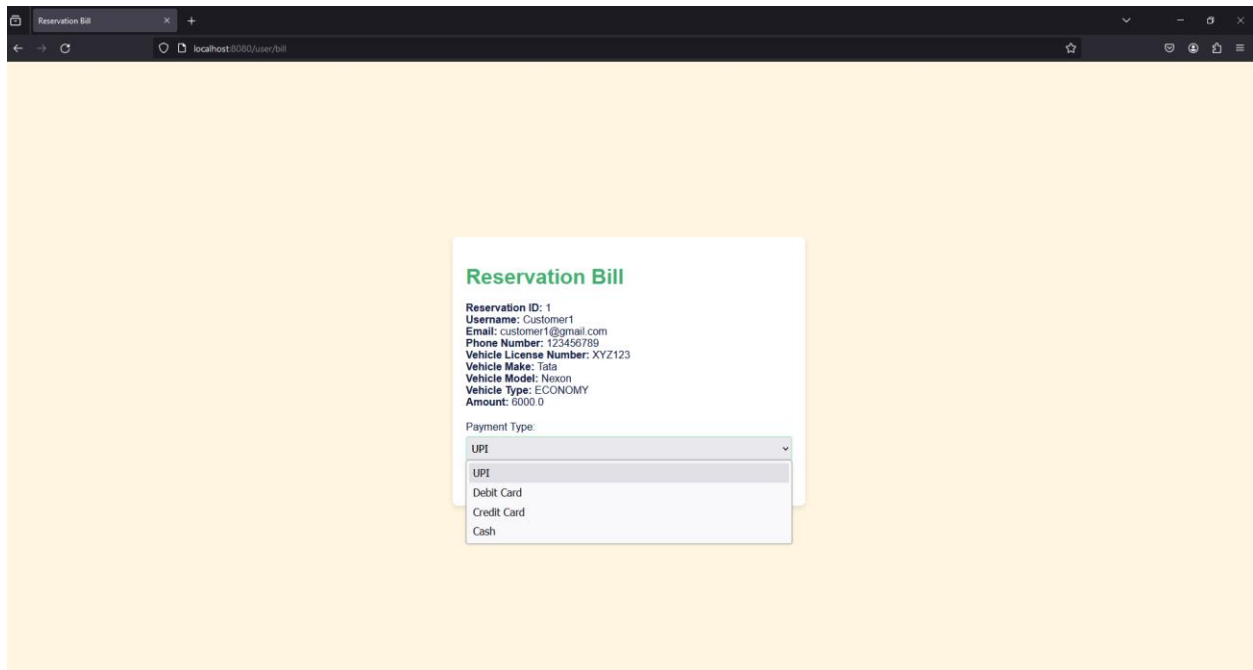
To Date:

## Bill generation



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/user/bill'. The page has a light orange background. In the center, there is a white card titled 'Reservation Bill' in green. The card contains the following text: 'Reservation ID: 1', 'Username: Customer1', 'Email: customer1@gmail.com', 'Phone Number: 123456789', 'Vehicle License Number: XYZ123', 'Vehicle Make: Tata', 'Vehicle Model: Nexon', 'Vehicle Type: ECONOMY', and 'Amount: 6000.0'. Below this text, there is a 'Payment Type:' label followed by a dropdown menu showing 'UPI'. At the bottom of the card is a green button labeled 'Pay Now'.

## Payment



The screenshot shows the same web browser window as before, but the 'Payment Type:' dropdown menu is now open, displaying a list of options: 'UPI', 'Debit Card', 'Credit Card', and 'Cash'. The 'UPI' option is currently selected and highlighted. The rest of the card and the background remain the same.