PROJECT REPORT

ON

OKBIKES

AT

EPTIQ Technologies, Wakad

By

RISHI RAM (12834)
MCA – II SEM – IV
(A.Y. - 2024-2025)

Under the guidance of
Prof. Neha Dhakol

Submitted to

SAVITRIBAI PHULE PUNE UNIVERSITY
In partial fulfillment of the requirement for the award of the degree of
Master of Computer Application (MCA)
Batch - 2023-2025

Through

SURYADATTA EDUCATION FOUNDATION'S
SURYADATTA INSTITUTE OF MANAGEMENT & MASS COMMUNICATION (SIMMC)
PUNE - 411021

DECLARATION

I hereby declare that the project titled "**Okbikes**", is an original piece of work carried out by me under the guidance of **Prof. Neha Dhakol.**

All the necessary information, facts and figures contained herein has been collected from genuine and authentic sources.

The information provided herein below is true and correct to the best of my knowledge and belief and nothing has been falsely stated or concealed therein.

The work has been submitted in partial fulfillment of the requirement of degree Master of Computer Application (MCA) to Savitribai Phule Pune University.

(Rishi Ram)
12834

Date:
Place: Pune

ACKNOWLEDGEMENT

I am glad to take this opportunity to acknowledge to all those who helped me in designing, developing and successful execution of my project on **Okbikes**

I would like to extend my thanks and gratitude to my project guide **Prof. Neha Dhakol** and **Dr. Vidya Gavekar Project Co-ordinator** for their valuable guidance and timely assistance throughout the development of this project.

Furthermore, I would like to express my appreciation to **Dr. Manisha Kumbhar**, **our Head of Department**, whose help and support this project would not have been possible. I am also grateful to **Dr. Shailesh Kasande, CEO, Group Director of our institute** for his valuable time and suggestions in the success of our project.

I also show my extended gratefulness towards SGI, **Chairman Prof Dr. Sanjay Chordiya** sir for his continues support and valuable guidance in the institute. Thank you to everyone involved for your valuable contributions and support throughout this work.

Thank you

Rishi Ram

Place: Pune

Date:

INDEX

CHAPTER 1 : INTRODUCTION

1. INTRODUCTION

1.1 Company Profile

Company Name: **Eptiq Technologies**

Eptiq Technologies is an emerging IT company located in Wakad, Pune, offering innovative and tailored digital solutions to a wide range of clients. The company focuses on delivering high-quality services across various domains, catering to the evolving needs of industries through technology-driven approaches.

Company wants to build a future where technology is easy to use, trustworthy, and safe, helping people in their daily lives. Our aim is to provide great services, build strong relationships with our clients, and grow by always learning and improving. By using the latest technologies and encouraging new ideas, we want to stay ahead of changes in the digital world. In the end, we hope to make a positive impact in technology, create a legacy of quality, and inspire future generations to create a better tomorrow.

The mission of Eptiq Technologies is to empower businesses with innovative technology solutions that drive growth and efficiency. We strive to understand and exceed the technological needs of our clients, ensuring they stay ahead in their respective industries. We believe in fostering a collaborative and innovative work environment where every team member can thrive and contribute to our collective success. We are committed to sustainable practices and ethical business operations, ensuring a positive impact on the environment and society.

We focus on honesty and new ideas as the foundation of our business. Honesty helps us make decisions and ensures we are open and truthful in everything we do, from helping customers to running our company. New ideas push us to find better solutions and technologies that improve our products and services, keeping us ahead in our field. By bringing together these values, we aim to not just meet but go beyond what our customers and employees expect, creating a culture of trust and quality that is important for our success in the future.

Key Services Offered:

1. Web Development:

Designing and developing responsive, user-friendly websites tailored to client requirements.

2. App Development:

Creating robust mobile applications with intuitive user interfaces across Android and iOS platforms.

3. IT Consultancy:

Offering strategic IT guidance and solutions to help businesses enhance their operations and efficiency.

4. Industrial IoT and Automation:

Providing smart industrial solutions for monitoring, control, and automation to improve productivity and reduce costs.

5. Digital Marketing:

Helping businesses grow online through SEO, social media marketing, and data-driven campaigns.

1.2 Abstract

This documentation presents a comprehensive overview of the **Okbikes Rental System**, a digital platform designed to streamline the process of renting bikes for users while providing robust management tools for administrators. The system facilitates seamless bike booking, real-time availability tracking, and secure payment processing, offering a user-friendly and efficient alternative to traditional rental services.

It highlights how the system addresses common challenges in the bike rental industry, such as manual booking inefficiencies, inventory mismanagement, and lack of user accountability.

In addition to describing system functionality, this document provides insights into the **technical architecture**, including the **front-end and back-end technologies**, **database structures**, **APIs**, and **deployment environments**.

This guide is intended for multiple audiences:

- **Users**, who will learn how to interact with the app for booking, returning, and tracking bikes;
- **Administrators**, who are provided with tools to manage bike fleets, monitor usage, and handle maintenance requests;
- **Developers**, who gain insight into the system's design patterns, code structure, testing strategies, and API endpoints.

1.3 Existing System and Need for System

Existing System

The bike and scooter rental market has grown significantly in recent years, fueled by the demand for convenient, cost-effective, and sustainable transportation options. However, despite the increased availability of these services, several key pain points persist, making the user experience less than ideal. Here's a deeper dive into each of the challenges mentioned and their implications:

1. Limited Booking Flexibility

- Current Problem: Rental platforms often offer set rental periods (hourly, daily, or monthly), with little flexibility to extend or modify bookings. For instance, users who need a vehicle for an unscheduled extra hour might face additional charges or be unable to extend the rental at all.
- User Impact: This lack of flexibility can cause frustration, especially for users who need last-minute changes or non-standard rental durations. Long-term rentals may also not align with the user's actual needs, as they are often locked into rigid packages.
- Potential Solution: A dynamic booking system that allows users to extend, shorten, or adjust rentals as required could significantly enhance the user experience. Additionally, a more personalized system could help tailor rental periods based on usage patterns.

2. Lack of Real-Time Vehicle Tracking

- Current Problem: Many platforms either lack live GPS tracking or provide inaccurate/slow updates, making it difficult for users to locate or track their rented vehicle.
- User Impact: Without real-time tracking, users may waste time searching for available bikes or scooters, especially in urban areas. In some cases, GPS errors may lead to safety concerns, such as not being able to find the vehicle in time or in dangerous areas.

- Potential Solution: Real-time GPS tracking that provides accurate locations and optimizes the search process is essential. Enhanced tracking can also improve safety by allowing users to see their vehicle's status, including battery level, maintenance status, and location in case of theft.

3. Poor User Interface & Experience

- Current Problem: Many rental platforms feature outdated apps or websites that are not user-friendly, leading to slow interactions, confusing designs, and high bounce rates. Complex signup processes and poorly designed booking flows deter users from engaging.
- User Impact: Users, especially first-time renters, may find it challenging to navigate the platform, slowing down their process and leading to dissatisfaction. A complicated booking experience can lead to abandonment and lower user retention.
- Potential Solution: Intuitive, easy-to-navigate, and visually appealing user interfaces (UI) would drastically improve the user experience. Streamlined features like quick logins, saved payment information, easy vehicle selection, and a simplified checkout process could enhance customer satisfaction.

4. Inadequate Package Customization

- Current Problem: Most platforms offer a few set pricing plans, and there are limited opportunities for users to customize their rental packages based on their specific needs (e.g., special discounts for students, corporate users, or specific usage days).
- User Impact: Users may feel restricted in their choices, as they may not be able to select the most cost-effective or suitable option. This lack of flexibility means some users could end up overpaying or not utilizing the service optimally.
- Potential Solution: Allowing users to personalize packages based on their requirements could address this gap. Options for different types of users (students, corporate employees, tourists) could be offered, along with discounts, memberships, or flexible plans that cater to their needs.

5. Low Vehicle Availability and Maintenance

- Current Problem: Users often face difficulties finding available vehicles during high-demand hours or in specific locations. Additionally, the lack of transparency around vehicle maintenance means users are often unsure about the quality and reliability of the vehicle they are renting.
- User Impact: Poor availability and vehicles that are in bad condition can severely affect a user's experience, potentially leading to delays or unsafe rides. Without proper maintenance, the vehicle may break down during use, causing frustration and inconvenience.
- Potential Solution: Better vehicle management and predictive maintenance could improve both availability and the overall user experience. Integrating AI to predict vehicle usage patterns and ensuring timely maintenance or repairs would help ensure that vehicles are ready for rental. More frequent checks and transparency regarding vehicle condition (via app notifications) would also help.

6. Limited Payment and Wallet Integration

- Current Problem: Many platforms do not support a wide variety of payment methods, limiting the options for users who prefer specific methods (such as certain digital wallets, credit cards, or installment payments). Some users may find themselves restricted to the payment methods available on the platform.
- User Impact: Inability to use preferred payment options can frustrate users, especially if the platform does not support common mobile wallets or payment plans (e.g., EMIs). This could also lead to transaction failures or payment delays.

- Potential Solution: Integrating more payment gateways, digital wallets, and flexible payment options, such as installment or subscription-based payments, would provide greater convenience for users. This would cater to a wider range of payment preferences, improving satisfaction and trust in the platform.

7. Inconsistent Customer Support

- Current Problem: Many bike/scooter rental services struggle to offer timely and effective customer support. Issues like vehicle breakdowns, app glitches, or late returns may go unresolved for extended periods, leading to negative experiences.
- User Impact: In case of a breakdown, a user's rental might get cut short, causing inconvenience. Slow or poor customer support can lead to frustration and mistrust, driving users away.
- Potential Solution: Offering 24/7 customer support with multiple contact options (live chat, phone, email) would help address issues quickly. Implementing an in-app support system or an emergency button in case of vehicle breakdowns or emergencies could increase user confidence. Additionally, AI-powered chatbots could handle minor issues and provide instant solutions to common problems.

Future Trends and Solutions in the Market:

- AI Integration: Using AI for predictive maintenance, demand forecasting, and real-time optimization of vehicle availability can significantly improve the overall service quality.
- Sustainability: With the growing focus on sustainability, introducing electric vehicles (EVs) or encouraging eco-friendly options for rentals will likely resonate with environmentally conscious consumers.
- Dynamic Pricing: Implementing dynamic pricing models based on demand, time, and vehicle type could make the service more flexible and affordable, appealing to a broader customer base.

Need for System

Okbikes aims to offer a comprehensive and user-friendly two-wheeler rental solution, addressing the gaps in existing services while promoting flexibility, sustainability, and a seamless experience. Below is a more detailed look at how Okbikes stands out in the competitive bike and scooter rental market, enhancing the rental experience with innovative features and solutions:

Key Features of Okbikes

1. Flexible Rental Packages

Okbikes offers multiple plans designed to cater to a wide range of users, from tourists needing short-term rentals to professionals looking for longer-term solutions. Here's how each feature is optimized for user convenience:

- Custom Packages:

  Okbikes tailors its offerings based on the user's unique needs. For instance, students can access exclusive discounts, while corporate clients can get customized rental packages for employees. Weekend plans target those who need vehicles for short bursts of time, offering affordability and convenience.

- Subscription Models:

  For regular users, Okbikes introduces subscription models, where users can pay a fixed monthly or yearly fee for access to a certain number of rentals. These models can include discounted rates or the ability to rent vehicles for longer periods at a reduced cost, perfect for frequent commuters or those needing transport for a longer time frame.

- Dynamic Pricing:

  Using real-time demand analytics, Okbikes adjusts pricing based on factors such as time of day, location, demand surges, and local events. This flexible pricing system ensures that users pay a fair price for the rental, especially during peak times or when there are fewer vehicles available.

- Special Offers:

  Seasonal offers, promotions, and limited-time discounts ensure that Okbikes remains competitive, while providing users with value-added incentives.

2. User-Friendly Booking and Cancellation Process

The ease of booking and managing rentals is a key factor in user satisfaction, and Okbikes focuses on making this process as smooth as possible:

- -Intuitive Interfaces:

  Both the mobile app and website are designed with minimalistic yet highly functional interfaces, allowing users to quickly browse through available vehicles, select their desired rental plan, and confirm bookings in just a few taps or clicks.

- Instant Booking Confirmations:

  Once a user selects a vehicle, Okbikes ensures that the booking is instantly confirmed, and the system sends notifications or reminders about the rental status. If necessary, users can be given smart suggestions for nearby vehicles or alternate plans if their first choice is unavailable.

- Flexible Ride Extensions and Cancellations:

  Okbikes understands that plans may change unexpectedly, so it allows users to extend rentals on the fly or cancel bookings with minimal hassle, including the ability to cancel or adjust booking times without incurring high fees.

- Google Maps Integration:

  Integration with Google Maps helps users locate rental stations or available vehicles more easily, ensuring they never have to waste time searching for pick-up locations. This integration also provides real-time navigation to the nearest vehicle, improving convenience.

3. Smart Vehicle Management

Okbikes leverages advanced technology to ensure that vehicles are in optimal condition and provide a safe, smooth ride:

- Telematics Integration:
  By integrating vehicle telematics, Okbikes monitors key vehicle parameters like battery health, fuel levels, and overall performance. This ensures timely maintenance, improving the lifespan of the vehicles and reducing the chances of breakdowns during the rental period.

- Regular Maintenance Reminders:
  The system automatically generates notifications for routine maintenance, repairs, or servicing needs, ensuring that all vehicles are safe and reliable before each rental.

- Quick Issue Resolution:
  Should a technical issue arise with a vehicle, Okbikes can quickly identify the problem and schedule immediate repairs or replacements. This minimizes downtime and ensures that customers always have access to fully functional vehicles.

4. Customer Support and Feedback Loop

Excellent customer support is a cornerstone of Okbikes, ensuring users feel heard and supported throughout their journey:

- 24/7 Support:
  Okbikes provides round-the-clock customer support through multiple channels—live chat, phone support, and email. Whether users have technical issues with the app or face difficulties with the vehicle during the ride, help is always available.

- Live Ride Assistance:
  In case of issues during the ride (e.g., vehicle malfunctions, navigation concerns), users can instantly reach out to Okbikes' live support team, ensuring immediate assistance and troubleshooting. This could include vehicle replacement or providing a solution remotely.

- User Feedback Collection:

  After each rental, Okbikes actively encourages users to provide feedback. This allows the company to continuously improve the app, vehicle fleet, and overall experience. User reviews and ratings also help future customers make more informed decisions.

- Instant Issue Reporting:

  Okbikes integrates an in-app feature that enables users to report issues instantly, whether it's related to vehicle condition, ride experiences, or app performance. These reports are automatically escalated to the appropriate support teams for swift resolution.

5. Eco-Friendly Options

Okbikes is committed to contributing to a cleaner environment by promoting sustainable transportation alternatives:

- Electric Scooters:

  Okbikes offers electric scooters as an alternative to conventional gasoline-powered bikes, giving users a greener mode of transportation. These electric vehicles are ideal for short-distance city commutes and significantly reduce emissions.

- Green Incentives:

  To encourage eco-friendly choices, Okbikes provides users with incentives such as discounts, loyalty points, or even free rentals after a certain number of electric scooter bookings. These rewards promote sustainable habits among riders.

- Sustainability Initiatives:

  The platform partners with local governments and sustainability organizations to promote clean transportation and encourage users to opt for greener options. Additionally, Okbikes is exploring the use of solar-powered charging stations for electric vehicles to further reduce the carbon footprint.

Additional Enhancements

1. Loyalty Program

Okbikes introduces a rewards-based loyalty program, where frequent riders can earn points for every ride, which can later be redeemed for discounts, free rentals, or exclusive offers. This program helps retain customers by offering long-term benefits.

2. Advanced Vehicle Selection

Users can filter available vehicles based on specific preferences, such as bike type (scooter, motorcycle), vehicle color, battery capacity, and even vehicle performance (e.g., high-speed options for commuters). This level of personalization ensures users get exactly what they need.

3. Safety Features

- Okbikes takes user safety seriously, offering built-in features like:
- Helmet Rentals: Ensure that helmets are available for users, either included with the rental or available for a small additional charge.
- Insurance Coverage: Okbikes provides users with insurance options to ensure they’re covered in the event of accidents, theft, or vehicle damage during the rental period.
- Safety Tips & Guidelines: The app educates users about safe riding practices, traffic rules, and the specific operation of rental vehicles.
- rental services, ready to redefine urban transportation.

1.4 Scope of System

The Okbikes Rental System is designed to provide an efficient, scalable, and user-friendly platform for renting two-wheelers. It focuses on improving the rental experience for both users and administrators through modern digital features and secure processes. The scope covers end-to-end rental lifecycle management—from user registration to ride tracking and administration.

Key Functionalities within Scope

1.User Registration and Authentication

- Users can register using mobile number.
- Secure login system with OTP verification and password recovery.
- Role-based access to distinguish between users, delivery staff, and admins.

2.Booking and Cancellation of Vehicles

- Users can browse and filter available vehicles by type (bike/scooter), model, and location.
- Real-time availability check with booking confirmation.
- Easy cancellation before start time, with automated refund processing based on policy.

3.Selection of Rental Packages

Multiple predefined packages such as:

- 1 Day
- 1 Week
- 15 Days
- 1 Month
- Custom duration booking options based on availability.
- Dynamic pricing depending on vehicle type, duration, and demand.

4.Application of Coupon Codes

- Users can enter promo/coupon codes at checkout to receive discounts.
- System validates code against usage rules (validity period, user eligibility, usage limit).
- Admin can create, manage, and deactivate promo codes from the dashboard.

5.Real-Time Vehicle Tracking

- Integration with GPS/IoT devices for live tracking of rented vehicles.
- Users can view current location, route history, and estimated time of return.
- Alerts and notifications for speed violations, unauthorized zone entry, or battery/fuel status.
- Admin access to tracking data for safety and monitoring purposes.

6.Admin Panel

- A comprehensive backend interface for system administrators to manage:
- Vehicle Inventory: Add/edit/delete vehicles, assign maintenance schedules.
- Bookings: View, modify, or cancel bookings manually.
- User Management: View user profiles, block/unblock accounts, handle disputes.
- Reports and Analytics: View real-time stats on bookings, earnings, usage trends.
- Coupon Management: Create and manage discount offers and referral codes.
- Feedback and Issue Resolution: Monitor user ratings and support tickets.

7.Additional Functionalities (Optional Extensions)

- In-app Wallet Integration for faster transactions.
- Notifications and Alerts for booking reminders, promotions, and ride updates.
- Ride History and Invoice Management.
- Integration with third-party insurance and roadside assistance providers.

1.5 Operating Environment: Hardware and Software

Hardware Requirement

Required field	**Expected response from students with options**
Processor (CPU):	Intel Core i3/i5 or equivalent
Operating System:	Microsoft Windows 10 Professional x64
Memory:	4 GB RAM & Above
Storage:	512 GB HDD or 256 GB HDD
Monitor/Display:	18" LCD monitor, resolution of 1600 x 900 or better.

Software Requirement

Required field	**Expected response from students with options**
Operating System:	Windows 7 and above
User Interface:	Tailwind CSS
Browser:	Google Chrome, Mozilla Firefox, Internet Explorer
Database:	MySQL
Front End:	Vite + ReactJS, JSX
Editor:	Visual Studio Code, IntelliJ IDEA, MySQL Workbench
Programming Languages/Technology:	Spring Boot
Script Languages:	JavaScript

1.6 Comprehensive Description of Technology Used

The Okbikes platform employs a carefully selected technology stack designed to ensure optimal performance, robust security, seamless scalability, and exceptional user experience across web and mobile interfaces. This document provides a detailed exploration of each technology component and its strategic role in powering the Okbikes ecosystem.

1.6.1 Operating Systems & Infrastructure

Windows

Windows serves as both the primary development environment and server platform for Okbikes, providing a versatile foundation for the system.

Strategic Advantages:

- Development Environment Compatibility: Supports an extensive range of development tools, IDEs, and software required for effective Okbikes development
- Enterprise-Grade Hosting: Windows Server editions deliver reliable, high-performance hosting capabilities with enterprise-level security features
- Integration Capabilities: Seamlessly integrates with cloud services, containerization platforms, and virtualization technologies
- Familiar Administration: Provides intuitive administrative interfaces for system maintenance and monitoring
- Robust Security Framework: Includes advanced security features such as Windows Defender, BitLocker encryption, and regular security updates
- Active Directory Integration: Enables centralized identity management and access control for development and administrative staff

Cloud Infrastructure

Okbikes leverages cloud services for enhanced reliability, scalability, and global reach.

Key Components:

- Microsoft Azure: Used for hosting backend services with automatic scaling capabilities to handle demand fluctuations
- Content Delivery Network (CDN): Distributes static content globally to minimize latency for users across different regions
- Load Balancing: Implements intelligent traffic distribution to ensure optimal performance and resource utilization

- Backup and Disaster Recovery: Maintains redundant systems with automated backup procedures to prevent data loss

1.6.2 Database Systems

MySQL Database

MySQL serves as the primary relational database management system, handling structured data essential to Okbikes' operations.

Strategic Advantages:

- ACID Compliance: Ensures transaction reliability and data integrity for critical operations like bookings and payments
- Robust Data Relationships: Enables complex relationships between entities (users, vehicles, bookings, payments)
- Query Optimization: Provides advanced indexing and query optimization capabilities for efficient data retrieval
- Stored Procedures: Supports encapsulation of business logic directly within the database for improved performance
- Replication Support: Enables database replication for improved read performance and failover capabilities
- Partitioning: Allows horizontal partitioning of large tables to maintain performance with growing data volumes
- Security Features: Provides comprehensive access control mechanisms and encryption capabilities

Database Schema Components:

- User management and authentication
- Vehicle inventory and availability tracking
- Booking and reservation systems
- Payment processing and transaction history
- Ratings and reviews
- Maintenance schedules and vehicle status
- Geolocation data for vehicle tracking

Redis Cache

Implements Redis as an in-memory data structure store for caching frequently accessed data.

Strategic Advantages:

- Session Management: Handles user session data with high-speed access
- Rate Limiting: Manages API request rates to prevent abuse
- Real-Time Data Caching: Reduces database load by caching frequently accessed information
- Pub/Sub Capabilities: Supports real-time features like vehicle location updates

1.6.3 Development Tools & Environments

MySQL Workbench

Advanced visual database design and management tool that streamlines MySQL database development.

Key Features Utilized:

- Visual Database Modeling: Creates and maintains comprehensive ERD (Entity Relationship Diagrams) for the Okbikes database
- Forward & Reverse Engineering: Synchronizes database schemas with visual models for documentation accuracy
- SQL Development: Provides advanced SQL editing capabilities with syntax highlighting and code completion
- Performance Tuning: Includes query profiling and execution plan visualization for optimization
- Data Migration: Facilitates data import/export and migration between development and production environments
- Database Administration: Offers user management, server monitoring, and configuration tools

Visual Studio Code

Primary code editor offering flexibility and extensive plugin support for diverse development needs.

Key Features Utilized:

- Integrated Terminal: Allows direct execution of commands without leaving the editor
- Live Share: Enables collaborative coding for team development sessions
- Git Integration: Provides visual diff tools and source control features
- Language Support: Offers comprehensive support for JavaScript, TypeScript, HTML, CSS, and more
- Debugging Tools: Includes breakpoints, call stacks, and variable inspection
- Extension Ecosystem: Integrates specialized tools for React, Node.js, Docker, etc.
- Customizable Workspace: Supports project-specific configurations for consistent team environments

IntelliJ IDEA

Specialized IDE for Java development with advanced code analysis capabilities.

Key Features Utilized:

- Intelligent Code Completion: Provides context-aware suggestions and automatic imports
- Advanced Refactoring: Offers safe, automated code restructuring capabilities
- Code Inspection: Detects potential issues and suggests improvements
- Build Tools Integration: Seamlessly works with Maven and Gradle for dependency management
- Spring Framework Support: Provides specialized tooling for Spring-based microservices
- Database Tools: Includes integrated database access and SQL editing capabilities
- Test Runners: Incorporates JUnit and TestNG for comprehensive testing

Docker Desktop

Containerization platform for consistent deployment across development and production environments.

Key Features Utilized:

- Container Management: Provides visual interface for managing containers and images
- Docker Compose Support: Enables multi-container application definition and management
- Volume Management: Facilitates data persistence for stateful applications
- Network Configuration: Manages isolated networks for service communication
- Resource Monitoring: Tracks container resource usage for optimization

1.6.4 Technologies Used

Frontend

ReactJS is a powerful JavaScript library used for building dynamic and responsive user interfaces, particularly suitable for single-page applications (SPAs). In the Okbikes rental system, ReactJS plays a central role in managing the frontend, ensuring a smooth and interactive user experience. Key features of ReactJS include:

- **Component-Based Architecture**: React enables the development of modular, reusable UI components that manage their own state, improving maintainability and scalability.
- **Virtual DOM**: Enhances performance by efficiently updating and rendering only the components that change, reducing direct interactions with the actual DOM.
- **JSX**: A syntax extension for JavaScript that allows HTML-like code to be written directly in JavaScript, improving code readability and developer productivity.

To enhance the development experience and performance, the **Vite** build tool is integrated with ReactJS. Vite provides a modern, fast development environment with near-instant server startup and lightning-fast hot module replacement (HMR). Benefits of using Vite include:

- **Fast Development Server**: Vite uses native ES modules to serve files directly, making development lightning fast.
- **Hot Module Replacement (HMR)**: Changes in code reflect instantly in the browser without a full reload, allowing faster feedback during development.
- **Optimized Builds**: Vite bundles the project using Rollup during the production build, resulting in highly optimized and efficient output.

To style the application, **Tailwind CSS**, a utility-first CSS framework, is used. Tailwind allows for rapid UI development with a consistent and responsive design. It enables developers to build custom interfaces using pre-defined utility classes directly in the markup. Key features of Tailwind CSS include:

- **Utility-First Approach**: Encourages building custom designs using utility classes like flex, p-4, bg-blue-500, avoiding the need for custom CSS files.
- **Responsive Design**: Built-in responsive utilities allow easy creation of mobile-friendly and adaptive layouts.
- **Customization**: The configuration files (tailwind.config.js) offer extensive customization options, allowing themes, breakpoints, and design systems to be tailored as per project needs.

Together, **ReactJS + Vite + Tailwind CSS** create a modern, scalable, and efficient frontend stack that supports the fast, responsive, and user-friendly interface required by the **Okbikes** rental system.

Backend

Spring Boot is an open-source Java-based framework designed to simplify the setup and development of production-ready Spring applications. It builds on the core Spring Framework, offering a convention-over-configuration approach that enables developers to create robust back-end systems with minimal boilerplate code.

Spring Boot is particularly well-suited for microservices, REST APIs, and enterprise-level applications like the **Okbikes rental system**, thanks to its scalability, reliability, and developer-friendly features.

Java with Spring Boot

Enterprise-grade framework for building robust backend services.

Strategic Advantages:

- Dependency Injection: Promotes loose coupling and testable code
- Aspect-Oriented Programming: Enables separation of cross-cutting concerns
- Spring Security: Provides comprehensive authentication and authorization
- Spring Data JPA: Simplifies database operations with repository abstractions
- Microservices Support: Facilitates development of modular, independently deployable services
- RESTful API Design: Enables standardized communication between frontend and backend
- Transaction Management: Ensures data consistency across operations

Key Features of Spring Boot:

- **Embedded Servers**:
  Spring Boot allows applications to run as standalone executables with embedded web servers such as **Tomcat**, **Jetty**, or **Undertow**, eliminating the need for deploying WAR files to external application servers. This simplifies deployment and testing across environments.
- **Spring Boot CLI**:
  Offers a command-line tool that can be used to rapidly prototype Spring applications using Groovy.
- **Security Integration**:
  Easily integrates with **Spring Security** for managing authentication and authorization. It supports modern protocols like JWT, OAuth2, and LDAP.

- **Rapid API Development**:

  Supports fast creation of **RESTful APIs**, with annotation-based programming (@RestController, @RequestMapping, etc.), simplifying the development of backend services for applications like Okbikes.

Database

MySQL is a widely-used, open-source **Relational Database Management System (RDBMS)** that enables structured storage, retrieval, and manipulation of data using **Structured Query Language (SQL)**. It is a core component of many web applications, and in the case of the **Okbikes rental system**, it is used to manage essential datasets such as user profiles, booking histories, vehicle inventories, payments, and operational records.

Key Features of MySQL:

- **SQL Support**:
  MySQL fully supports ANSI SQL standards, allowing developers to perform powerful queries for data manipulation and reporting. This includes advanced features like joins, stored procedures, views, triggers, and transactions, enabling efficient operations on complex data relationships.
- **Scalability and Performance**:
  MySQL is built to handle small to very large databases. It offers features like indexing, query optimization, caching, and replication, which help ensure performance remains high even with growing datasets.
- **Cross-Platform Compatibility**:
  MySQL can run on all major operating systems (Linux, Windows, macOS) and integrates well with web frameworks and back-end technologies including Spring Boot, which you're using in Okbikes.
- **Community and Ecosystem**:
  With a large global community, MySQL benefits from rich documentation, community support, and frequent updates. Popular tools like **phpMyAdmin**, **MySQL Workbench**, and **DBeaver** provide intuitive interfaces for managing databases visually.

CHAPTER 2 : PROPOSED SYSTEM

1. PROPOSED SYSTEM

1.1 Study of Similar Systems

1.1.1 Analysis of Existing Rental Systems

1. Market Research: Conduct a thorough market analysis to identify existing bike and scooter rental systems.
2. Feature Comparison: Compare the features offered by competitors, such as rental packages, vehicle tracking, user interface, and customer support.
3. User Feedback: Gather user feedback and reviews to understand the strengths and weaknesses of competing systems.
4. Technological Advancements: Identify the technologies used by competitors and assess their effectiveness.

1.1.2 Key Findings

1. Flexibility: Many existing systems lack flexible rental packages and real-time vehicle tracking.
2. User Experience: There is a need for a more user-friendly interface and seamless booking process.
3. Security: Enhanced security measures for user data and payment transactions are essential.
4. Customer Support: Efficient customer support is crucial for resolving user issues and maintaining customer satisfaction.

1.2 Feasibility Study

1.2.1 Technical Feasibility

1. Technology Stack: The proposed system will use a robust technology stack, including ReactJS and Tailwind CSS for the frontend, Spring Boot for the backend, and MySQL for the database.
2. Development Tools: Utilize MySQL Workbench, Visual Studio Code, and IntelliJ IDEA for development and management.
3. Integration: Ensure seamless integration of frontend and backend components, as well as real-time vehicle tracking.
4. Scalability: The system will be designed to handle a large number of users and transactions, ensuring scalability and performance.

1.2.2 Economic Feasibility

1. Cost Analysis: Conduct a detailed cost analysis, including development costs, server hosting, maintenance, and marketing expenses.
2. Revenue Projection: Estimate potential revenue from rental packages, coupon sales, and partnerships.
3. Break-Even Analysis: Determine the break-even point to assess the financial viability of the project.
4. Funding Options: Explore funding options such as venture capital, angel investors, and crowdfunding.

1.2.3 Operational Feasibility

1. Resource Availability: Ensure the availability of skilled developers, designers, and project managers.
2. Project Timeline: Develop a realistic project timeline, including milestones for development, testing, and deployment.
3. Risk Management: Identify potential risks and develop mitigation strategies to ensure smooth operation.
4. User Training: Provide training and support for users and administrators to ensure they can effectively use the system.

1.3 Objectives of Proposed System

1.3.1 Enhanced User Experience

1. Seamless Booking Process: Provide a user-friendly interface for booking and canceling vehicles.
2. Real-time Tracking: Offer real-time vehicle tracking to ensure the safety and security of the rented vehicles.
3. Flexible Rental Packages: Provide a variety of rental packages to cater to different user needs.

1.3.2 Improved Operational Efficiency

1. Admin Panel: Develop a comprehensive admin panel for managing vehicles, stores, bookings, and coupons.
2. Automated Processes: Automate booking confirmations, cancellations, and payment processing to reduce manual effort.
3. Data Analytics: Implement data analytics to track user behavior, booking trends, and system performance.

1.3.3 Security and Reliability

1. Data Protection: Ensure robust security measures to protect user data and payment transactions.
2. System Reliability: Design the system to be reliable and resilient, with minimal downtime and quick recovery from failures.
3. Regulatory Compliance: Ensure compliance with relevant regulations and standards for data protection and privacy.

1.3.4 Scalability and Growth

1. Scalable Architecture: Design the system to handle increasing numbers of users and transactions.
2. Expansion Plans: Plan for future expansion, including adding new features, vehicles, and rental packages.
3. Partnerships: Explore partnerships with local businesses, tourism agencies, and other stakeholders to expand the user base.

1.3.5 Customer Satisfaction

1. User Support: Provide efficient customer support to resolve user issues and enhance satisfaction.
2. Feedback Mechanism: Implement a feedback mechanism to gather user suggestions and improve the system.
3. Loyalty Programs: Develop loyalty programs and promotional offers to retain users and encourage repeat bookings.

1.4 Users of System

USER ROLES AND RESPONSIBILITIES

1. Administrators

An **admin** is a user with the highest level of access and control on a website or system. They are responsible for managing content, users, settings, and security. Admins can make changes that affect the entire website, including updating features, moderating activity, and maintaining the system.

1.1 System Management

1. System Configuration: Configure system settings, user roles, and permissions to ensure optimal performance and security.
2. Data Security: Implement and monitor security protocols to protect user data and system integrity.
3. Performance Monitoring: Analyze system performance, and optimize system resources.

1.2 User Management

1. User Accounts: Manage user accounts, verify users
2. Access Control: Ensure that users have appropriate access levels based on their roles and responsibilities.
3. Support: Provide technical support to users, including troubleshooting issues and answering queries.

1.3 Reporting and Analytics

1. Generate Reports: Create and analyze reports on system usage, user activity, booking trends, and financial transactions.
2. Feedback Analysis: Gather and analyze user feedback to identify areas for improvement.

1.4 System Integrity

1. Maintenance: Perform regular maintenance tasks to ensure the system's functionality and reliability.
2. Updates: Implement system updates and patches to enhance features and security.
3. Backup: Ensure regular backups of system data to prevent data loss.

2. Users

2.1 Travel Planning

1. Search and Book: Search for and book various travel services, including flights, hotels, transportation, tour packages, activities, and travel insurance.
2. Customize Itineraries: Customize their travel itineraries based on their preferences and needs.

2.2 Interaction and Support

1. Customer Support: Interact with customer support services for inquiries, assistance, or changes to their bookings.
2. Feedback: Provide feedback on their travel experiences and system usage.
3. Reviews: Write reviews and recommendations for travel services to help other customers.

2.3 Account Management

1. User Profile: Create and manage their user profiles, including personal information and travel preferences.
2. Booking History: View their booking history and upcoming travel plans.
3. Settings: Adjust account settings and preferences.

3. Guest Users

Guest users refer to individuals who interact with system without creating an account or logging in. They are temporary users who have limited access to certain features and functionalities of the system. Their key activities include:

3.1 Limited Access

1. Browse Services: Browse available packages/tariff plan.
2. View Information: View detailed information about packages, such as descriptions, prices, and availability.
3. Read Reviews: Read reviews and recommendations from other customers.

Chapter 3 : Analysis & Design

3.1 SYSTEM REQUIREMENTS

Functional Requirements

1. User Management

User Registration: Users should be able to create an account by providing necessary details such as name, email, phone number, and password.

User Login: Users should be able to log in using their email and password.

User Profile Management: Users should be able to view their profile information.

2. Booking Management

Vehicle Booking: Users should be able to book a vehicle by selecting the type of vehicle (bike or scooter), rental duration, and pick-up location.

Booking Confirmation: Users should receive a confirmation email or SMS with booking details.

Booking Cancellation: Users should be able to cancel their booking, subject to cancellation policies.

Coupon Application: Users should be able to apply coupon codes during the booking process to avail discounts.

3. Admin Management

Vehicle Management: Admins should be able to add, edit, and delete vehicles in the inventory.

Store Management: Admins should be able to add, edit, and delete store locations.

Package Management: Admins should be able to add, edit, and delete rental packages.

Coupon Management: Admins should be able to add, edit, and delete coupon codes.

Booking Management: Admins should be able to view and manage booking details, including user information and rental periods.

4. Reporting and Analytics

Usage Reports: Admins should be able to generate reports on system usage, user activity, and booking trends.

Performance Metrics: Admins should be able to monitor key performance indicators (KPIs) to assess the system's effectiveness and efficiency.

Feedback Analysis: Admins should be able to gather and analyze user feedback to identify areas for improvement.

5. Customer Support

Support Tickets: Users should be able to create support tickets for inquiries, assistance, or changes to their bookings.

Support Responses: Admins should be able to respond to support tickets and resolve user issues.

FAQ Section: The system should have a FAQ section to address common user queries.

Non-Functional Requirements

1. Performance

Response Time: The system should have a response time of less than 3 seconds for all user interactions.

Scalability: The system should be able to handle a large number of concurrent users and transactions without performance degradation.

Load Handling: The system should be able to handle peak loads during high-demand periods.

2. Security

Data Encryption: All user data and transactions should be encrypted to ensure data security.

Access Control: The system should implement role-based access control to ensure that users have appropriate access levels.

Authentication: The system should use secure authentication mechanisms, such as multi-factor authentication (MFA), to protect user accounts.

3. Usability

User Interface: The system should have an intuitive and user-friendly interface for both users and admins.

Accessibility: The system should be accessible to users with disabilities, following accessibility standards such as WCAG.

Responsiveness: The system should be responsive and compatible with various devices, including smartphones, tablets, and computers.

4. Reliability

Availability: The system should have an uptime of at least 99.9% to ensure continuous availability.

Fault Tolerance: The system should be designed to handle failures gracefully and recover quickly.

Backup and Recovery: The system should have regular backup and recovery procedures to prevent data loss.

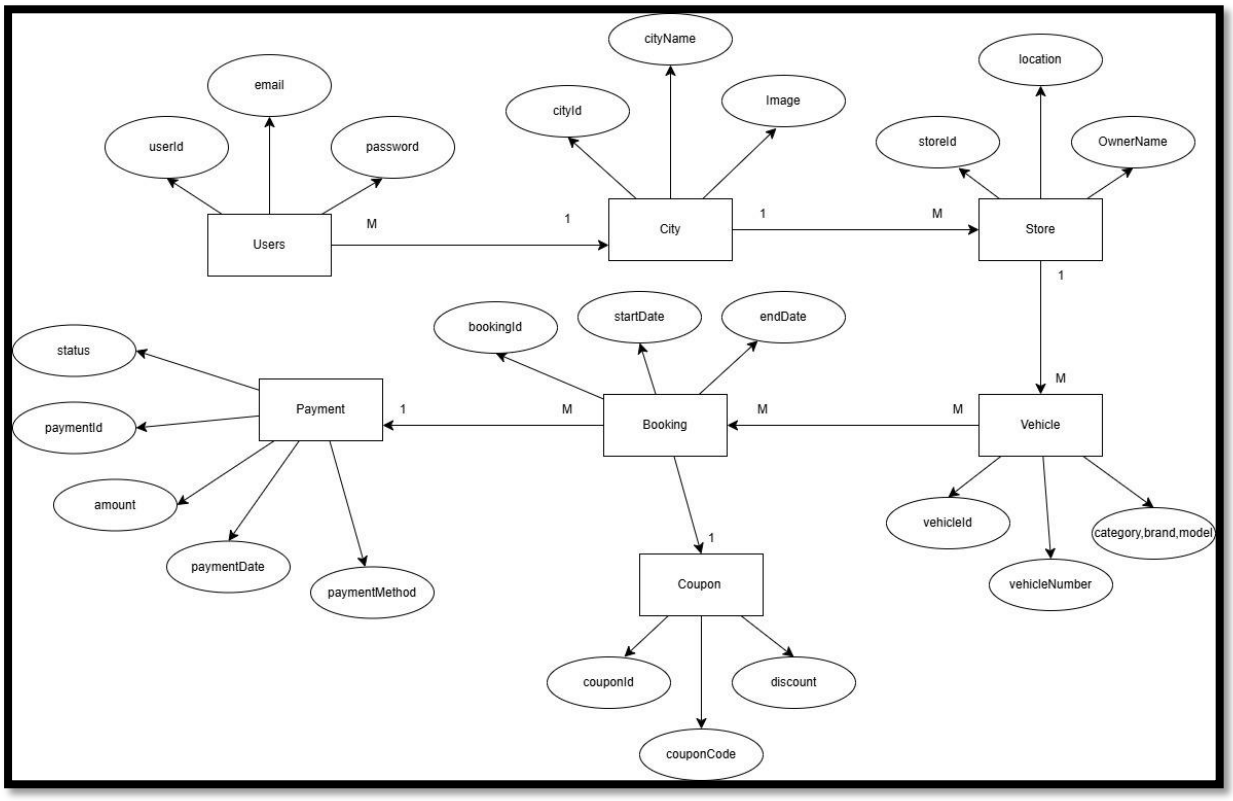
5. Maintainability

Code Quality: The system should follow best practices for code quality, including code reviews and testing.

Documentation: The system should have comprehensive documentation for developers, users, and admins.

Updates: The system should be designed to allow for easy updates and maintenance.

3.2 Entity Relationship Diagram (ERD)



3.3 Table Structure

Table Name: Admin

Table Description: This Table stores admin details.

Primary Key: admin_Id

Field Name	Data Type	Size	Constraint	Description
admin_Id	int	11	Primary Key	Admin id no
name	varchar	255	Not null	Admin Name
email	varchar	255	Not null	Admin Email
password	varchar	255	Not null	Admin Password
phone_number	varchar	255	Not null	Admin contact Number
username	varchar	255	Not null	Admin Details
verification	enum		Not null	Admin authentication

Table Name: Booking

Table Description: This Table stores Booking details.

Primary Key: Booking_Id

Field Name	Data Type	Size	Constraint	Description
Booking _Id	int	11	Primary Key	Booking id no
user_id	bigint	-	Foreign Key	User ID
vehicle_id	bigint	-	Foreign Key	Vehicle ID
vehicle_package_id	bigint	-	Foreign Key	Vehicle Package ID
status	enum	-	Not null	Booking Status
delivery_location	varchar	255	Not null	Delivery Locaion Details
late_fee_charges	double	-	Not null	Late Fee Details
Additional_charges	double	-	Not null	Additional Charges Details
challan	double	-	Not null	Additional Charges Details

Table Name: Booking Payments

Table Description: This Table stores booking payments details.

Primary Key: -

Field Name	Data Type	Size	Constraint	Description
payments_id	bigint	-	Primary Key	Payment id no
booking_id	bigint	-	Foreign Key	Booking Id

Table Name: City

Table Description: This Table stores city's details.

Primary Key: city_id

Field Name	Data Type	Size	Constraint	Description
city_id	bigint	-	Primary Key	City id no
image	longtext	-	Not null	City Image
name	varchar	255	Not null	City Name
State	varchar	255	Not null	State Name

Table Name: Coupon

Table Description: This Table stores coupon's details.

Primary Key: coupon_id

Field Name	Data Type	Size	Constraint	Description
coupon_id	bigint	-	Primary Key	Coupon id no
coupon_code	varchar	255	Not null	Coupon code
coupon_name	varchar	255	Not null	Coupon name
coupon_type	Enum	-	Not null	Coupon type
discount_value	double	-	Not null	Discount Value
is_active	Bit	1	Not null	Is Coupon Active
remaining_coupon	int	10	Not null	Coupon count
start_date	date	-	Not null	Starting Date
end_date	date	-	Not null	Ending Date

Table Name: Coupon Usage

Table Description: This Table stores Coupon Usage details.

Primary Key: coupon_usage_id

Field Name	Data Type	Size	Constraint	Description
coupon_usage_id	bigint	-	Primary Key	Coupon Usage id no
coupon_id	bigint	-	Foreign Key	Coupon id no
user_id	bigint	-	Foreign Key	User Id no

Table Description: This Table stores OTP details.

Primary Key: otp _id

Field Name	Data Type	Size	Constraint	Description
otp _id	bigint	-	Primary Key	OTP id no
otp	varchar	255	Not null	OTP
phone_number	varchar	255	Foreign Key	User Phone no
expiry_time	date	-	Not null	Expiry time of otp

Table Name: Payment

Table Description: This Table stores payment details.

Primary Key: payment_id

Field Name	Data Type	Size	Constraint	Description
payment_id	bigint	-	Primary Key	Payment id no
amount	double	-	Not null	Total Amount
payment_date	date	-	Not null	Payment Date
status	enum	-	Not null	Payment Status
transaction_id	varchar	255	Not null	Payment’s Transaction Id
booking_id	varchar	255	Foreign Key	Booking ID
payment_method	enum	-	Not null	Payment Mode

Table Name: Store

Table Description: This Table stores store details.

Primary Key: store_id

Field Name	Data Type	Size	Constraint	Description
store_id	bigint	-	Primary Key	Store id no
email	varchar	255	Not null	Store's Email
image	longtext	-	Not null	Store's Image
store_name	varchar	255	Not null	Store's Name
store_address	varchar	255	Not null	Store's address
store_location_url	varchar	255	Not null	Store's Location
mobile_number	varchar	255	Not null	Store's Conteact no
store_owner	varchar	255	Not null	Store Owner Name
store_status	enum	-	Not null	Store's Status
city_id	Bigint	-	Foreign Key	City Id no

Table Name: Token

Table Description: This Table stores Token's details.

Primary Key: token _id

Field Name	Data Type	Size	Constraint	Description
token _id	bigint	-	Primary Key	Token id no
token	varchar	255	Not null	Token
user_id	bigint	-	Foreign Key	User Id no

Table Name: User

Table Description: This Table stores User's details.

Primary Key: user_id

Field Name	Data Type	Size	Constraint	Description
user_id	bigint	-	Primary Key	User id no
name	varchar	255	Not null	User's Name
email	varchar	255	Not null	User's Email
password	varchar	255	Not null	User's Password
phone_number	varchar	255	Not null	User's contact Number
role	enum	-	Not null	User's Role
aadhar_front_side	longtext	-	Not null	Aadhar card's Front Details
aadhar_back_side	longtext	-	Not null	Aadhar card's Back Details
driving_license	longtext	-	Not null	Driving License
city_id	bigint	-	Foreign Key	City Id no
aadhar_front_status	enum	-	Not null	Aadhar card's Front Status
aadhar_back_status	enum	-	Not null	Aadhar card's Back Status
driving_license_status	enum	-	Not null	Driving License Status

Table Name: Vehicle

Table Description: This Table stores vehicle details.

Primary Key: vehicle_id

Field Name	Data Type	Size	Constraint	Description
vehicle_id	bigint	-	Primary Key	Vehicle id no
chassis_number	varchar	255	Not null	Chassis Number
engine_number	varchar	255	Not null	Engine Number
fuel_type	enum	-	Not null	Vehicle's Fuel Type
image	longtext	-	Not null	Vehicle's Image
vehicle_registration_number	varchar	255	Not null	Vehicle's Registration Number

registration_year	varchar	255	Not null	Vehicle's Registration Year
store_id	bigint	-	Foreign Key	Store ID
vehicle_brand_id	bigint	-	Foreign Key	Vehicle's Brand ID
vehicle_category_id	bigint	-	Foreign Key	Vehicle's Category ID
vehicle_model_id	bigint	-	Foreign Key	Vehicle's Model ID
vehicle_status	enum	-	Not null	Vehicle's Status

Table Name: Vehicle Brand

Table Description: This Table stores vehicle brand details.

Primary Key: vehicle_brand _id

Field Name	Data Type	Size	Constraint	Description
vehicle_brand_id	bigint	-	Primary Key	Vehicle's Brand ID
brand_name	varchar	255	Not null	Vehicle's Brand Name
image	longtext	-	Not null	Brand Image

Table Name: Vehicle Category

Table Description: This Table stores vehicle categories details.

Primary Key: vehicle_category _id

Field Name	Data Type	Size	Constraint	Description
vehicle_category_id	bigint	-	Primary Key	Vehicle's Category ID
category_name	varchar	255	Not null	Vehicle's Category Name

Table Name: Vehicle Model

Table Description: This Table stores vehicle models details.

Primary Key: vehicle_category _id

Field Name	Data Type	Size	Constraint	Description
vehicle_model_id	bigint	-	Primary Key	Vehicle's Model ID
model_name	varchar	255	Not null	Vehicle's Model Name
vehicle_brand_id	bigint	-	Foreign Key	Vehicle's Brand ID

Table Name: Vehicle Package

Table Description: This Table stores vehicle package details.

Primary Key: vehicle_package _id

Field Name	Data Type	Size	Constraint	Description
vehicle_package _id	bigint	-	Primary Key	Vehicle's Package ID
days	int	50	Not null	Package Days
deposit	double	-	Not null	Vehicle's Deposit
hours	int	100	Not null	Package Hour's
price	double	-	Not null	Package Price
vehicle_category_id	bigint	-	Foreign Key	Vehicle's Category ID

Data Dictionary

Sr. No.	Field Name	Data Type	Size	Constraint	Description	Table Name
1	admin_Id	int	11	Primary Key	Admin id no	Admin
2	name	varchar	255	Not null	Admin Name	Admin
3	email	varchar	255	Not null	Admin Email	Admin
4	password	varchar	255	Not null	Admin Password	Admin
5	phone_number	varchar	255	Not null	Admin contact Number	Admin
6	username	varchar	255	Not null	Admin Details	Admin
7	verification	enum		Not null	Admin authentication	Admin
8	booking _Id	int	11	Primary Key	Booking id no	Booking
9	user_id	bigint	-	Foreign Key	User ID	Booking
10	vehicle_id	bigint	-	Foreign Key	Vehicle ID	Booking
11	vehicle_package_id	bigint	-	Foreign Key	Vehicle Package ID	Booking
12	status	enum	-	Not null	Booking Status	Booking
13	delivery_location	varchar	255	Not null	Delivery Locaion Details	Booking
14	late_fee_charges	double	-	Not null	Late Fee Details	Booking
15	Additional_charges	double	-	Not null	Additional Charges Details	Booking

16	challan	double	-	Not null	Additional Charges Details	Booking
17	payments_id	bigint	-	Primary Key	Payment id no	Booking Payments

18	booking_id	bigint	-	Foreign Key	Booking Id	Booking Payments
19	city_id	bigint	-	Primary Key	City id no	City
20	image	longtext	-	Not null	City Image	City
21	name	varchar	255	Not null	City Name	City
22	State	varchar	255	Not null	State Name	City
23	coupon_id	bigint	-	Primary Key	Coupon id no	Coupon
24	coupon_code	varchar	255	Not null	Coupon code	Coupon
25	coupon_name	varchar	255	Not null	Coupon name	Coupon
26	coupon_type	Enum	-	Not null	Coupon type	Coupon
27	discount_value	double	-	Not null	Discount Value	Coupon

28	is_active	Bit	1	Not null	Is Coupon Active	Coupon
29	remaining_coupon	int	10	Not null	Coupon count	Coupon
30	start_date	date	-	Not null	Starting Date	Coupon
31	end_date	date	-	Not null	Ending Date	Coupon
32	coupon_usage_id	bigint	-	Primary Key	Coupon Usage id no	Coupon Usage
33	coupon_id	bigint	-	Foreign Key	Coupon id no	Coupon Usage
34	user_id	bigint	-	Foreign Key	User Id no	Coupon Usage

35	otp _id	bigint	-	Primary Key	OTP id no	Otp
36	otp	Varchar	255	Not null	OTP	Otp
37	phone_number	Varchar	255	Foreign Key	User Phone no	Otp
38	expiry_time	Date	-	Not null	Expiry time of otp	Otp

39	payment_id	bigint	-	Primary Key	Payment id no	Payment
40	amount	double	-	Not null	Total Amount	Payment
41	payment_date	date	-	Not null	Payment Date	Payment
42	status	enum	-	Not null	Payment Status	Payment
43	transaction_id	varchar	255	Not null	Payment's Transaction Id	Payment
44	booking_id	varchar	255	Foreign Key	Booking ID	Payment
45	payment_method	enum	-	Not null	Payment Mode	Payment
46	store_id	bigint	-	Primary Key	Store id no	Store
47	email	varchar	255	Not null	Store's Email	Store
48	image	longtext	-	Not null	Store's Image	Store
49	store_name	varchar	255	Not null	Store's Name	Store
50	store_address	varchar	255	Not null	Store's address	Store
51	store_location_url	varchar	255	Not null	Store's Location	Store
52	mobile_number	varchar	255	Not null	Store's Conteact no	Store
53	store_owner	varchar	255	Not null	Store Owner Name	Store

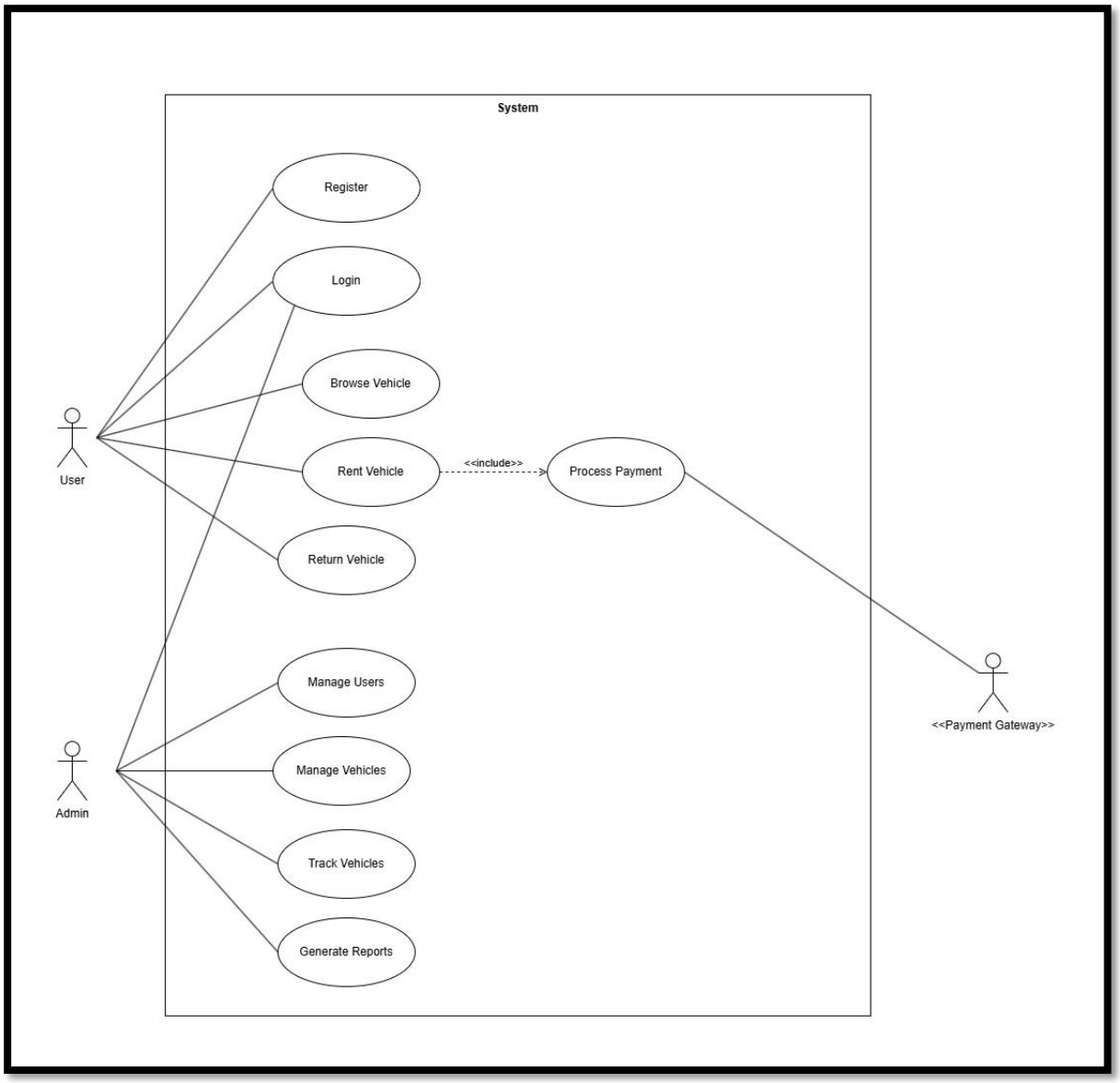
54	store_status	enum	-	Not null	Store's Status	Store
55	city_id	bigint	-	Foreign Key	City Id no	Store
56	token _id	bigint	-	Primary Key	Token id no	Token
57	token	varchar	255	Not null	Token	Token
58	user_id	bigint	-	Foreign Key	User Id no	Token
59	user_id	bigint	-	Primary Key	User id no	User
60	name	varchar	255	Not null	User's Name	User
61	email	varchar	255	Not null	User's Email	User
62	password	varchar	255	Not null	User's Password	User
63	phone_number	varchar	255	Not null	User's contact Number	User
64	role	Enum	-	Not null	User's Role	User
65	aadhar_front_side	longtext	-	Not null	Aadhar card's Front Details	User
66	aadhar_back_side	longtext	-	Not null	Aadhar card's Back Details	User
67	driving_license	longtext	-	Not null	Driving License	User
68	city_id	bigint	-	Foreign Key	City Id no	User
69	aadhar_front_status	enum	-	Not null	Aadhar card's Front Status	User
70	aadhar_back_status	enum	-	Not null	Aadhar card's Back Status	User

71	driving_license_status	enum	-	Not null	Driving License Status	User
72	vehicle_id	bigint	-	Primary Key	Vehicle id no	Vehicle
73	chassis_number	varchar	255	Not null	Chassis Number	Vehicle
74	engine_number	varchar	255	Not null	Engine Number	Vehicle
75	fuel_type	enum	-	Not null	Vehicle‘s Fuel Type	Vehicle
76	image	longtext	-	Not null	Vehicle’s Image	Vehicle
77	vehicle_registration_number	varchar	255	Not null	Vehicle’s Registration Number	Vehicle
78	registration_year	varchar	255	Not null	Vehicle’s Registration Year	Vehicle
79	store_id	bigint	-	Foreign Key	Store ID	Vehicle
80	vehicle_brand_id	bigint	-	Foreign Key	Vehicle’s Brand ID	Vehicle
81	vehicle_category_id	bigint	-	Foreign Key	Vehicle’s Category ID	Vehicle
82	vehicle_model_id	bigint	-	Foreign Key	Vehicle’s Model ID	Vehicle
83	vehicle_status	enum	-	Not null	Vehicle’s Status	Vehicle
84	vehicle_brand_id	bigint	-	Primary Key	Vehicle’s Brand ID	Vehicle Brand
85	brand_name	varchar	255	Not null	Vehicle’s Brand Name	Vehicle Brand
86	image	longtext	-	Not null	Brand Image	Vehicle Brand
87	vehicle_category_id	bigint	-	Primary Key	Vehicle’s Category ID	Vehicle Category
88	category_name	varchar	255	Not null	Vehicle’s Category Name	Vehicle Category
89	vehicle_model_id	bigint	-	Primary	Vehicle’s Model ID	Vehicle Model

				Key		
90	model_name	varchar	255	Not null	Vehicle's Model Name	Vehicle Model
91	vehicle_brand_id	bigint	-	Foreign Key	Vehicle's Brand ID	Vehicle Model
92	vehicle_package _id	bigint	-	Primary Key	Vehicle's Package ID	Vehicle Package
93	days	int	50	Not null	Package Days	Vehicle Package
94	deposit	double	-	Not null	Vehicle's Deposit	Vehicle Package
95	hours	int	100	Not null	Package Hour's	Vehicle Package
96	price	double	-	Not null	Package Price	Vehicle Package
97	vehicle_category_id	bigint	-	Foreign Key	Vehicle's Category ID	Vehicle Package

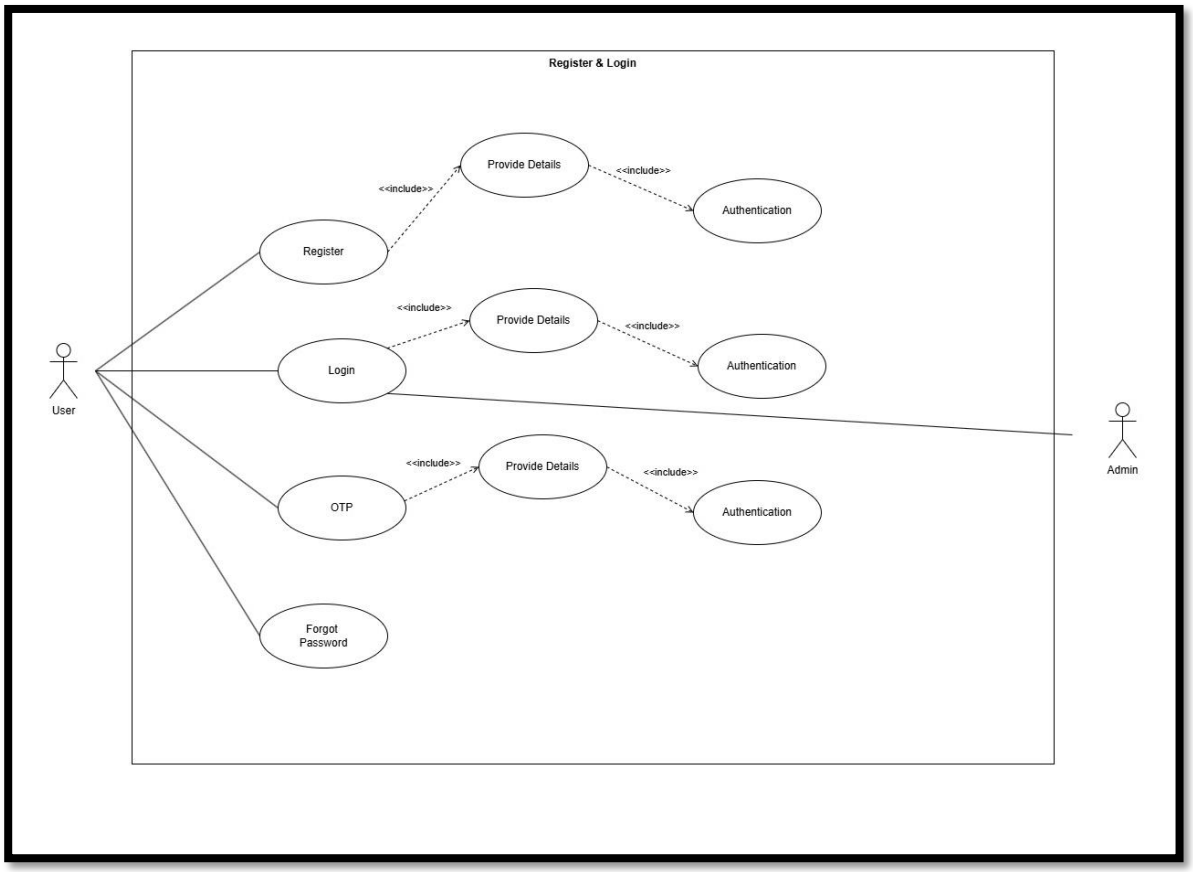
3.4 Use case Diagrams:

1. System



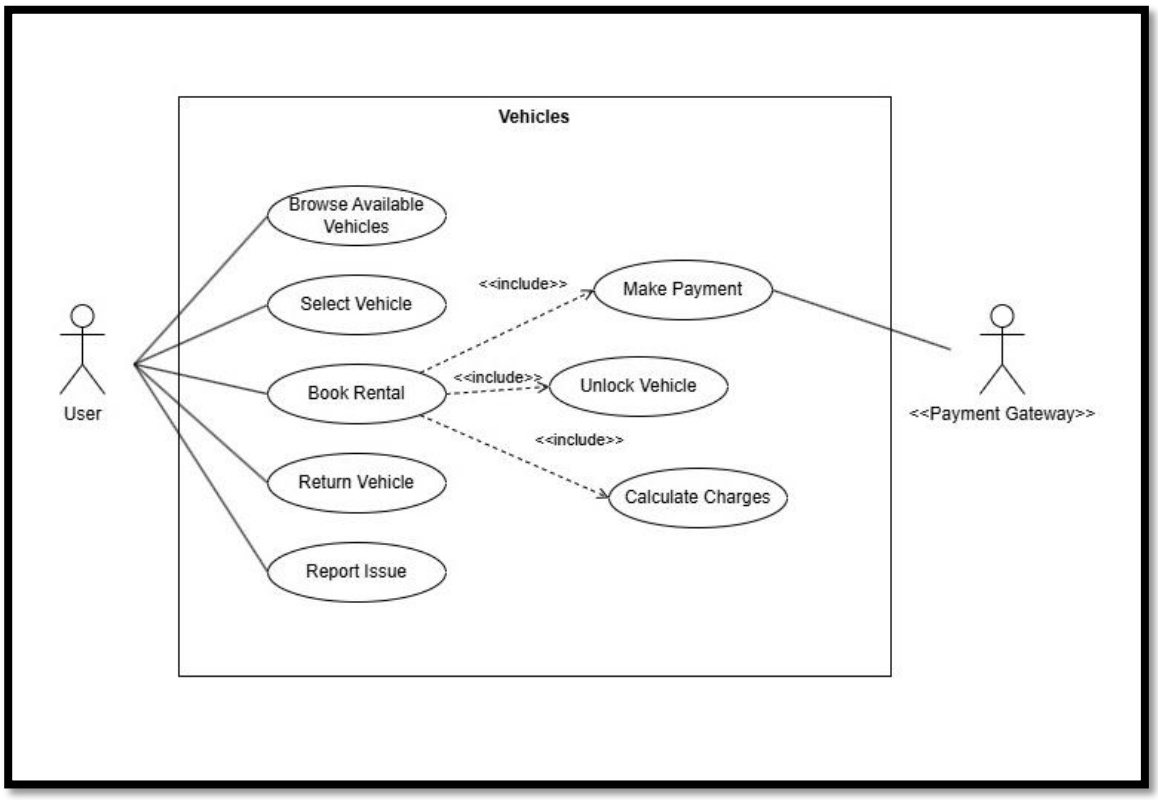
Use case Name	System
Actor	Admin, Customer
Detail Description	This use case diagram describe the registration and login process of Admin & Customer
Pre-condition	User must have a valid email id & Password for the registration
Main Flow	After the registration the both users are have their user id & password. With the help of user id and password they can login.
Post condition	All the users are get the usernames and their Password. After login the admin can manage works.

2. Register & Login



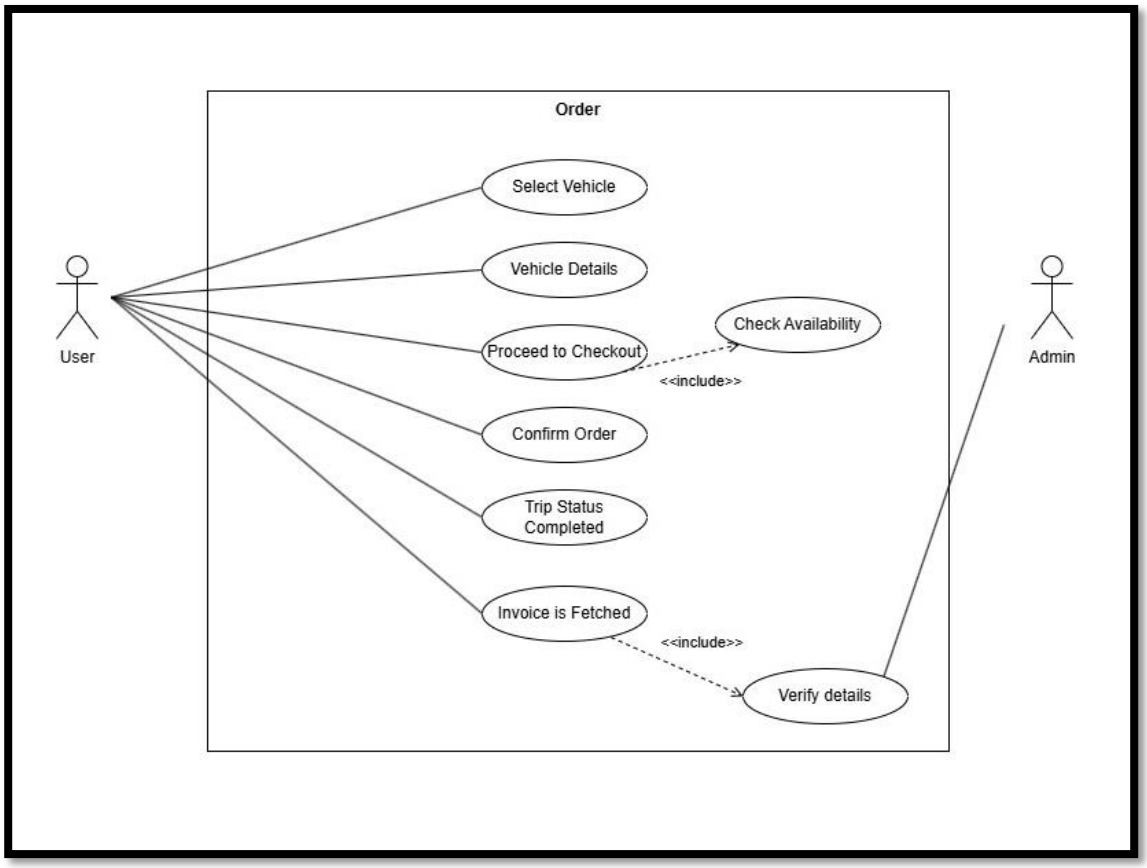
Use Case Name	Registration and login
Actor	Admin and User
Pre-Condition	Admin have own system and their login account
Description	Admin logins and access the system
Basic Flow	• User need to register first. • User needs to login. • User have profile , order , checkout details page
Post Condition	Admin and user have their own logins

3. Vehicles



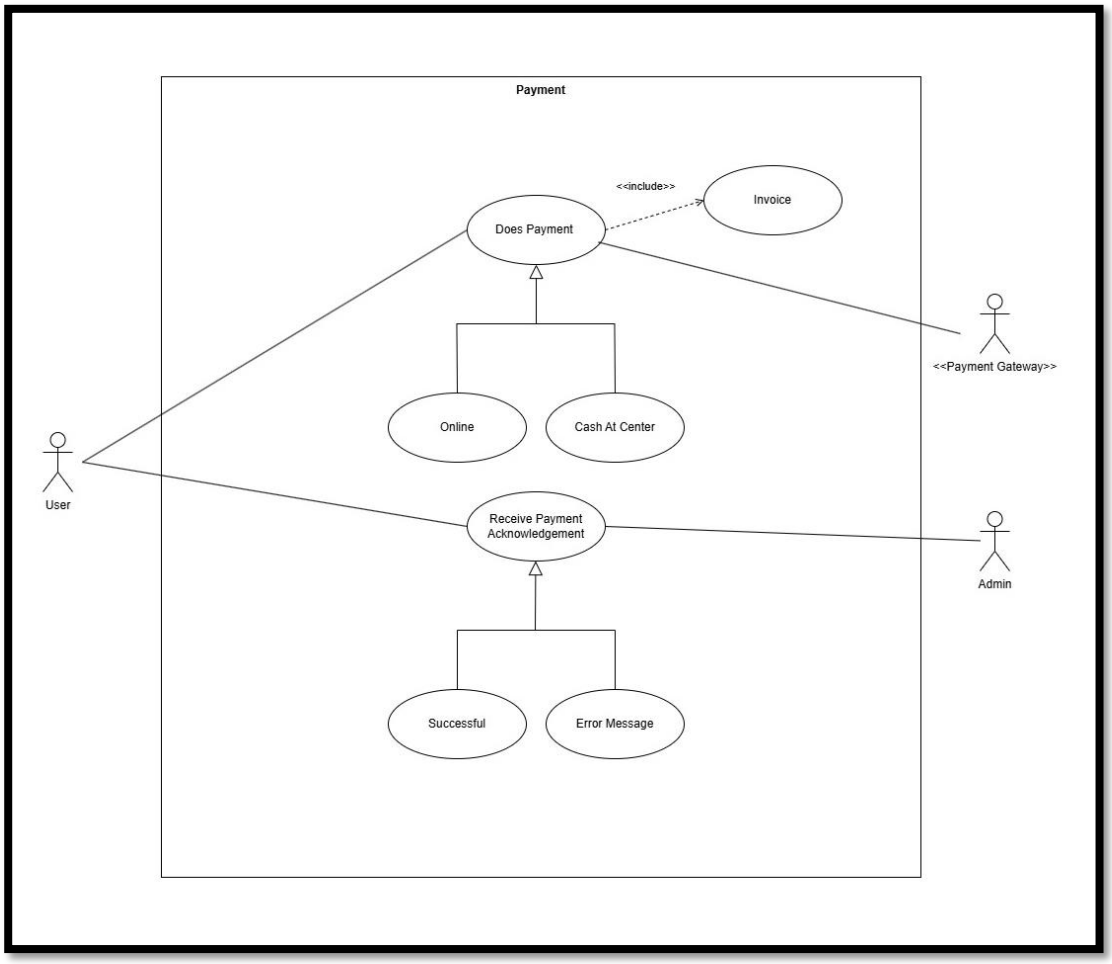
Use Case Name	Vehicles
Actor	Admin and User
Pre-Condition	users have their own login account
Description	Users can access vehicles
Basic Flow	• User needs to login • User can view vehicles. • Admin can manage vehicles.
Post Condition	Vehicles are fetching properly.

4. Order



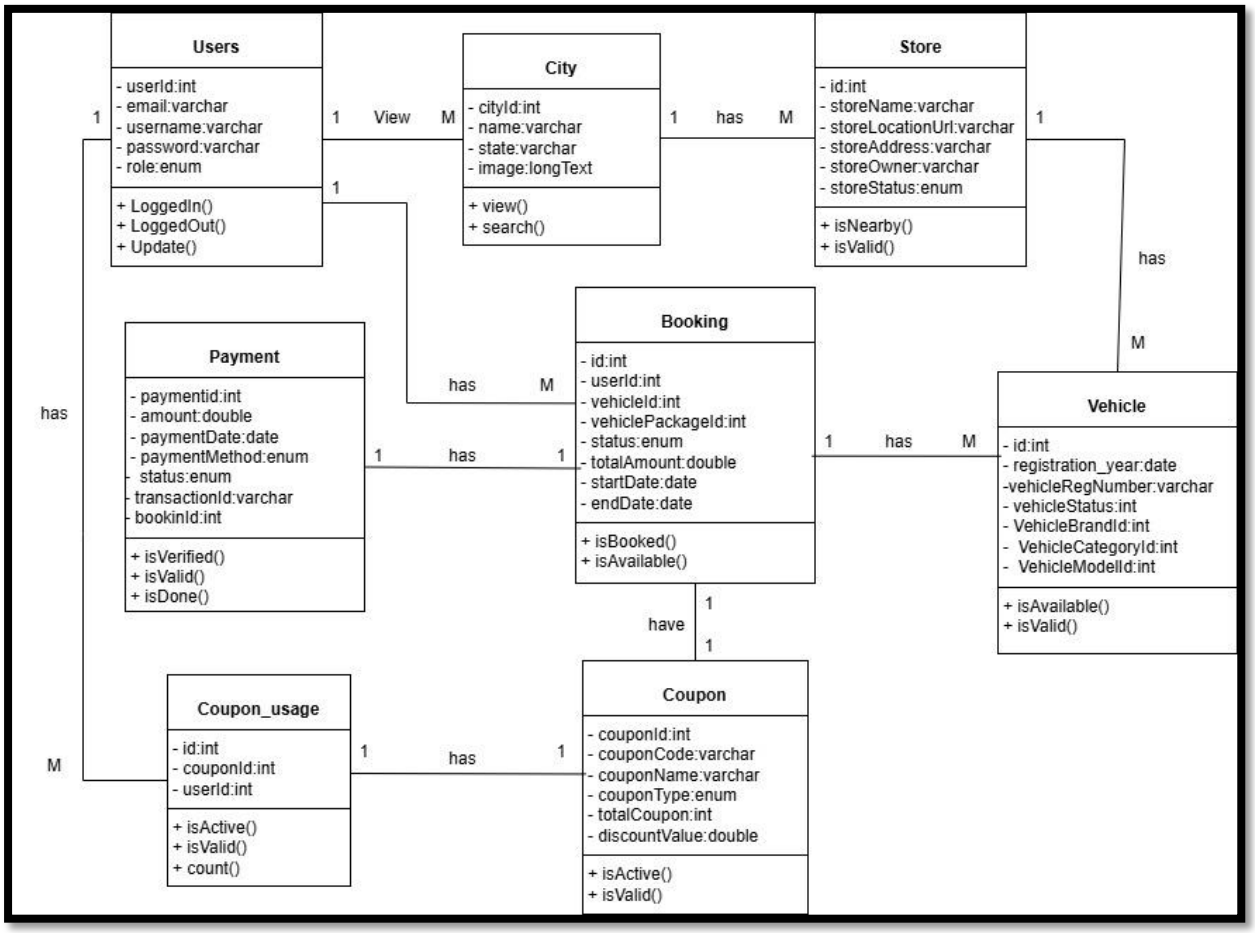
Use Case Name	Order
Actor	Admin and User
Pre-Condition	users have rights to order
Description	Users can access orders
Basic Flow	• User needs to login • User can order products • Admin can manage Products • Users receive bill once order is placed
Post Condition	Orders placed successfully.

5. Payment



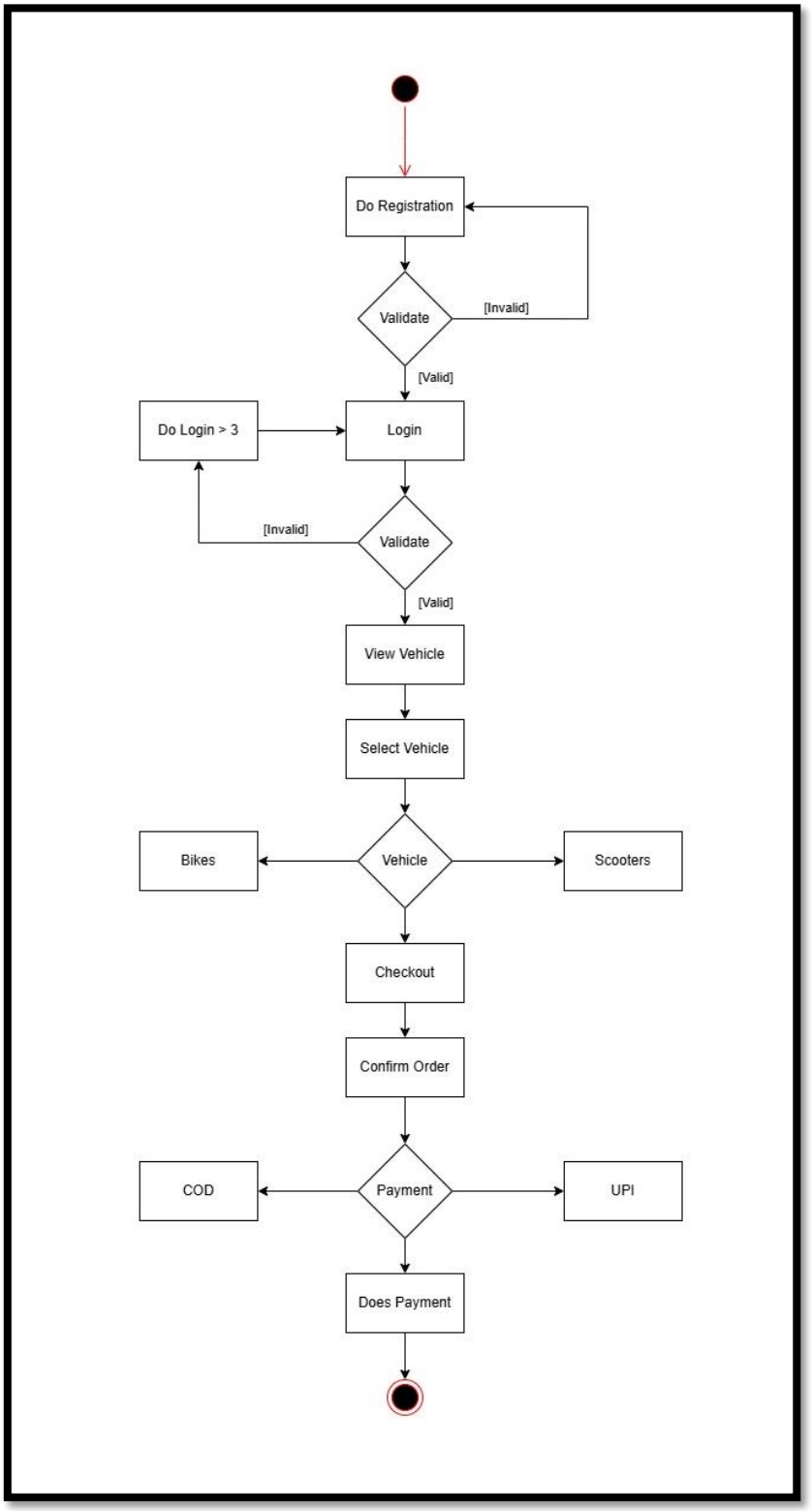
Use Case Name	Payment
Actor	Admin and User
Pre-Condition	users can pay once order is confirmed
Description	Users can order products
Basic Flow	• User needs to login • User can order products • Users receive bill once order is placed • Admin receives payment details.
Post Condition	Payment done successfully.

3.5 Class Diagram

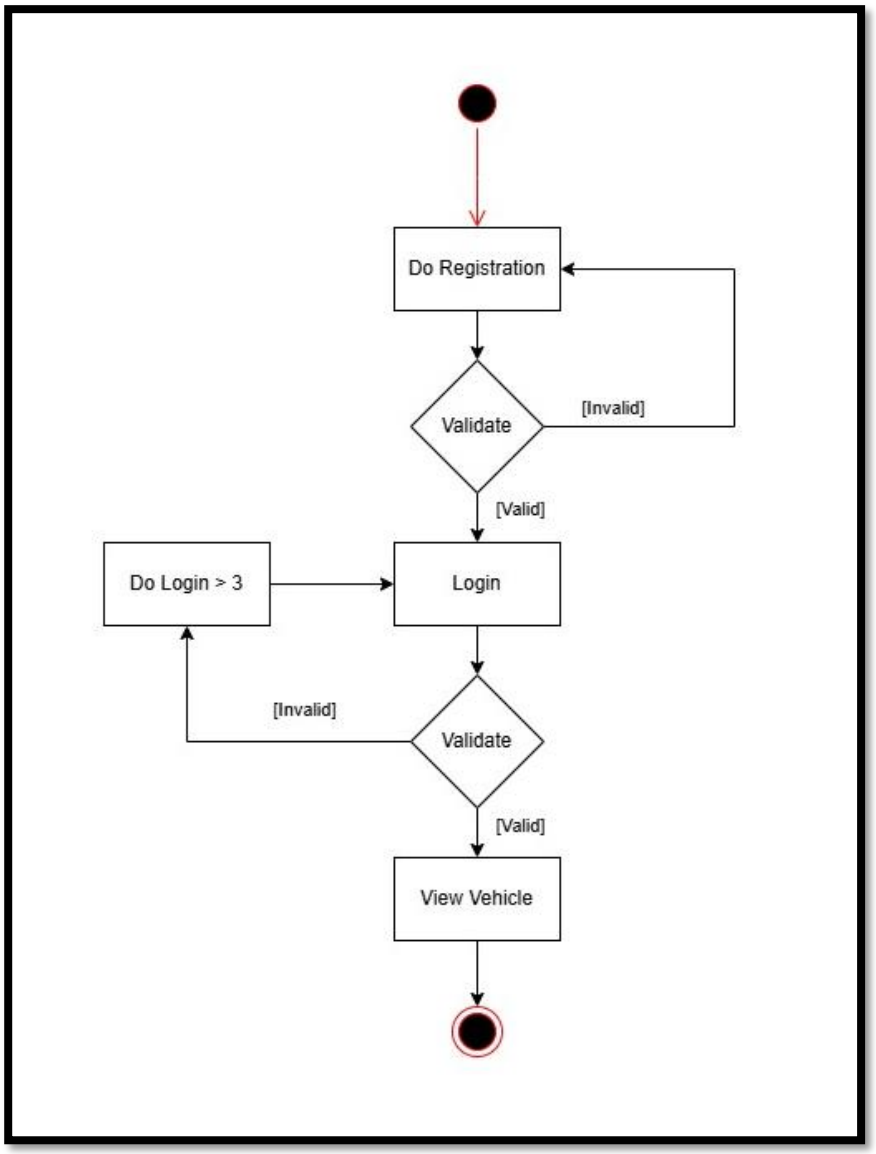


3.6 Activity Diagrams

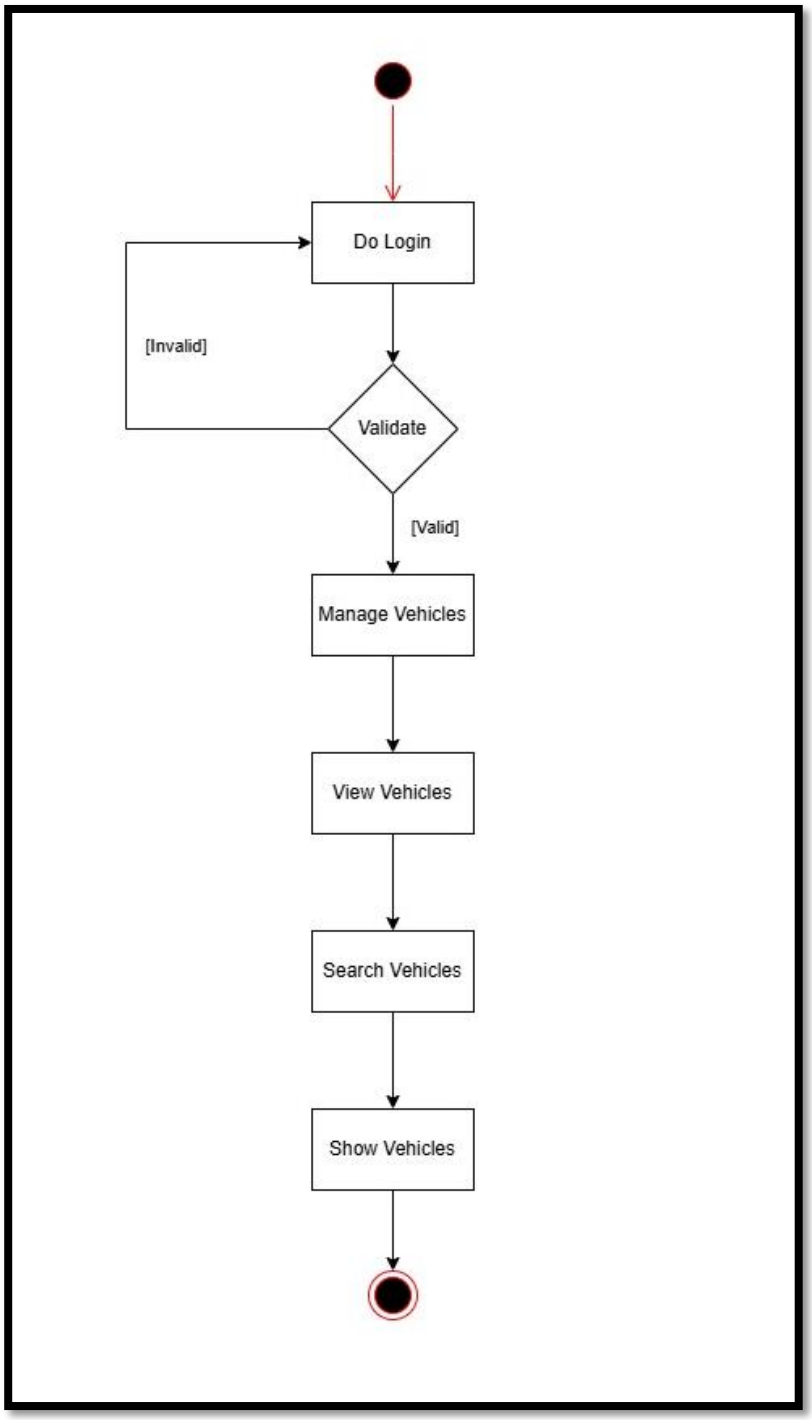
1. System



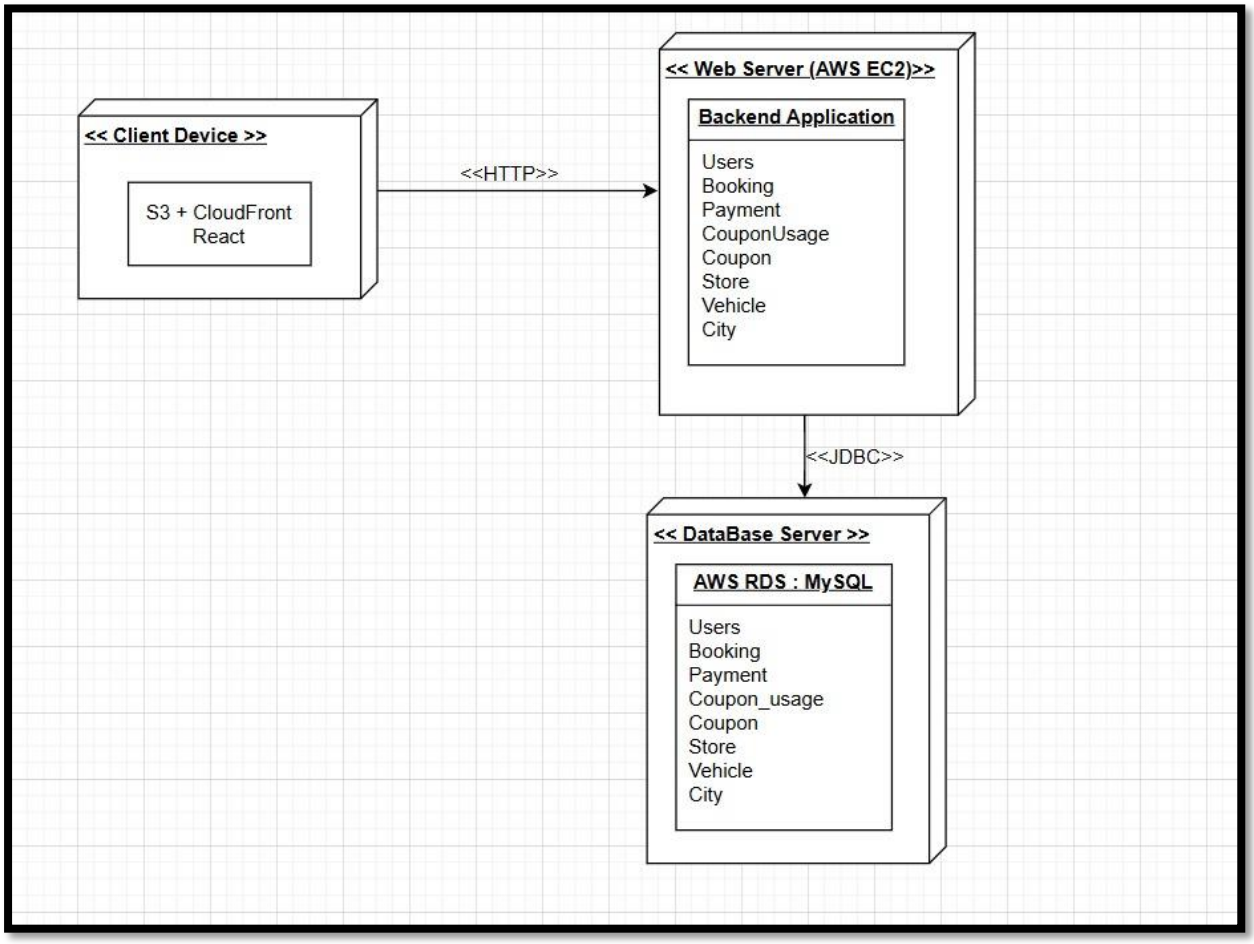
2. Register & Login



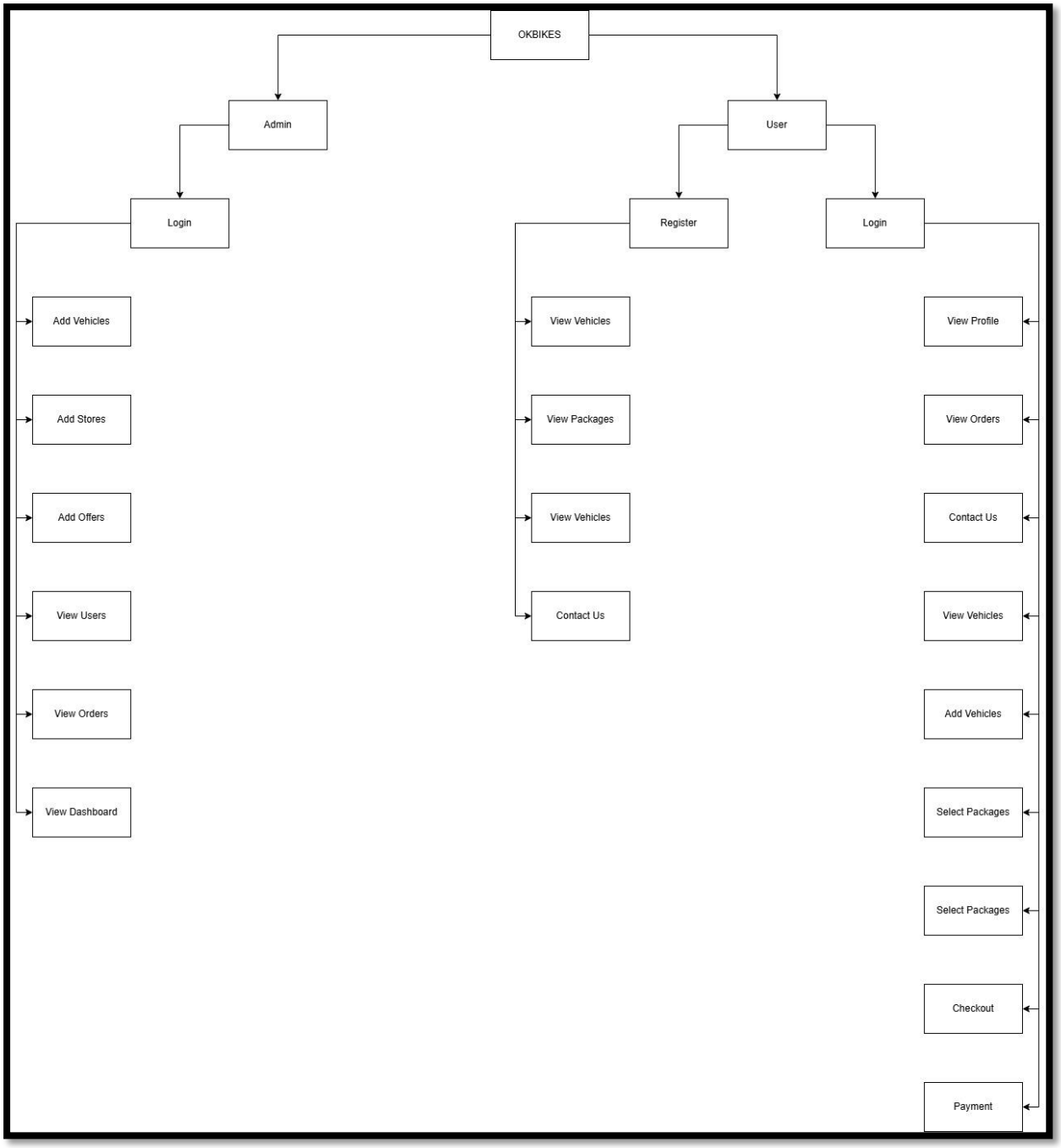
3. Vehicles


Do Login
[Invalid]
Validate
[Valid]
Manage Vehicles
View Vehicles
Search Vehicles
Show Vehicles

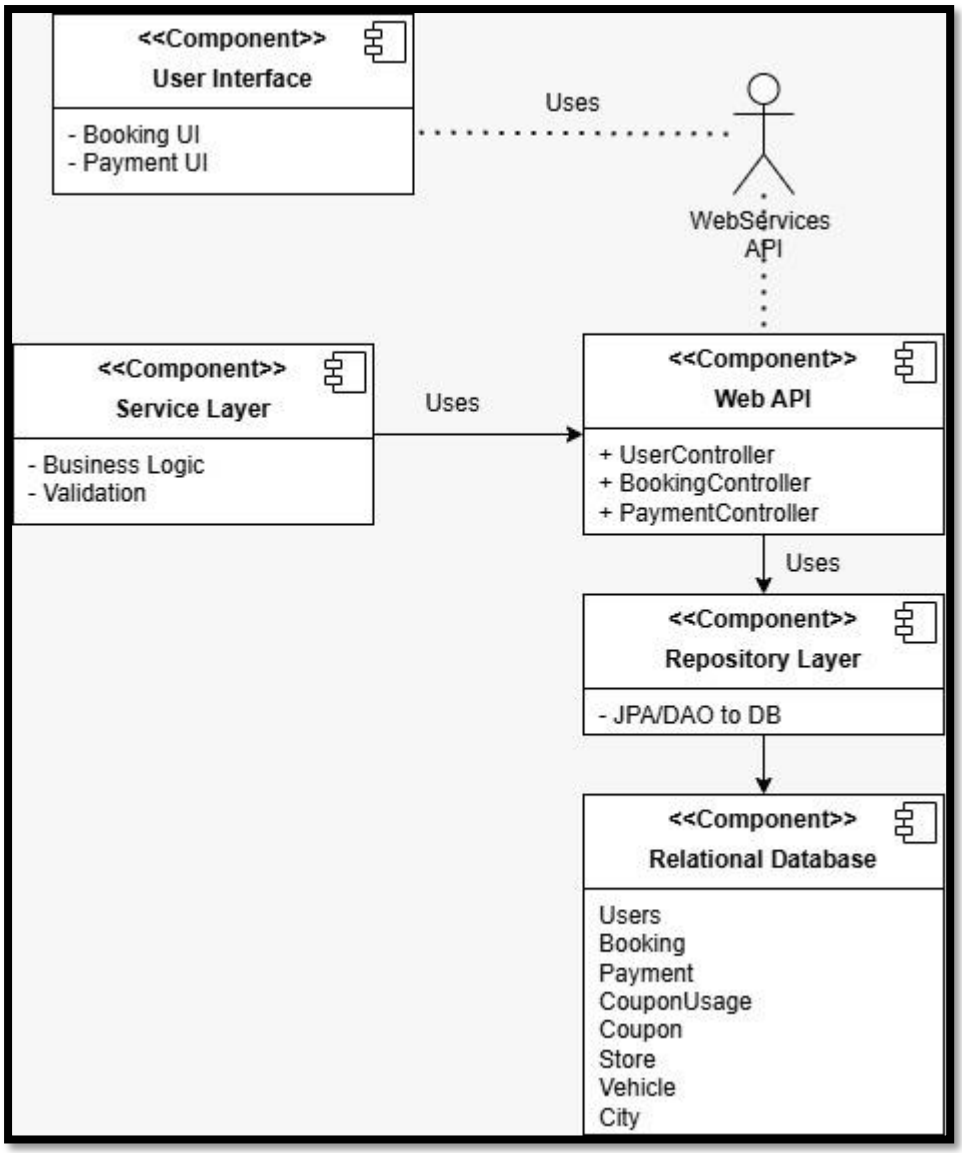
3.7 Deployment Diagram



3.8 Module Hierarchy Diagram

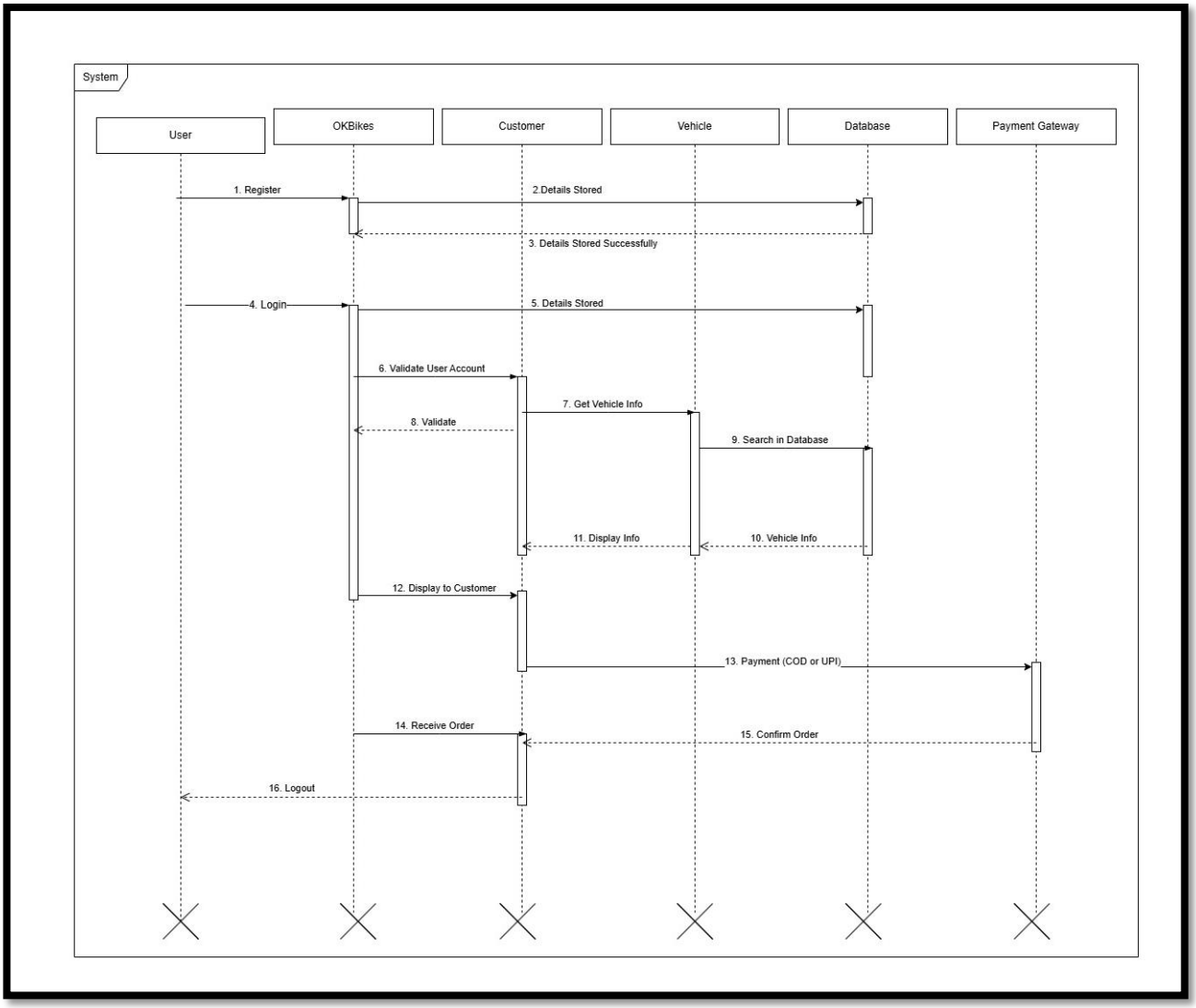


3.9 Component Diagram

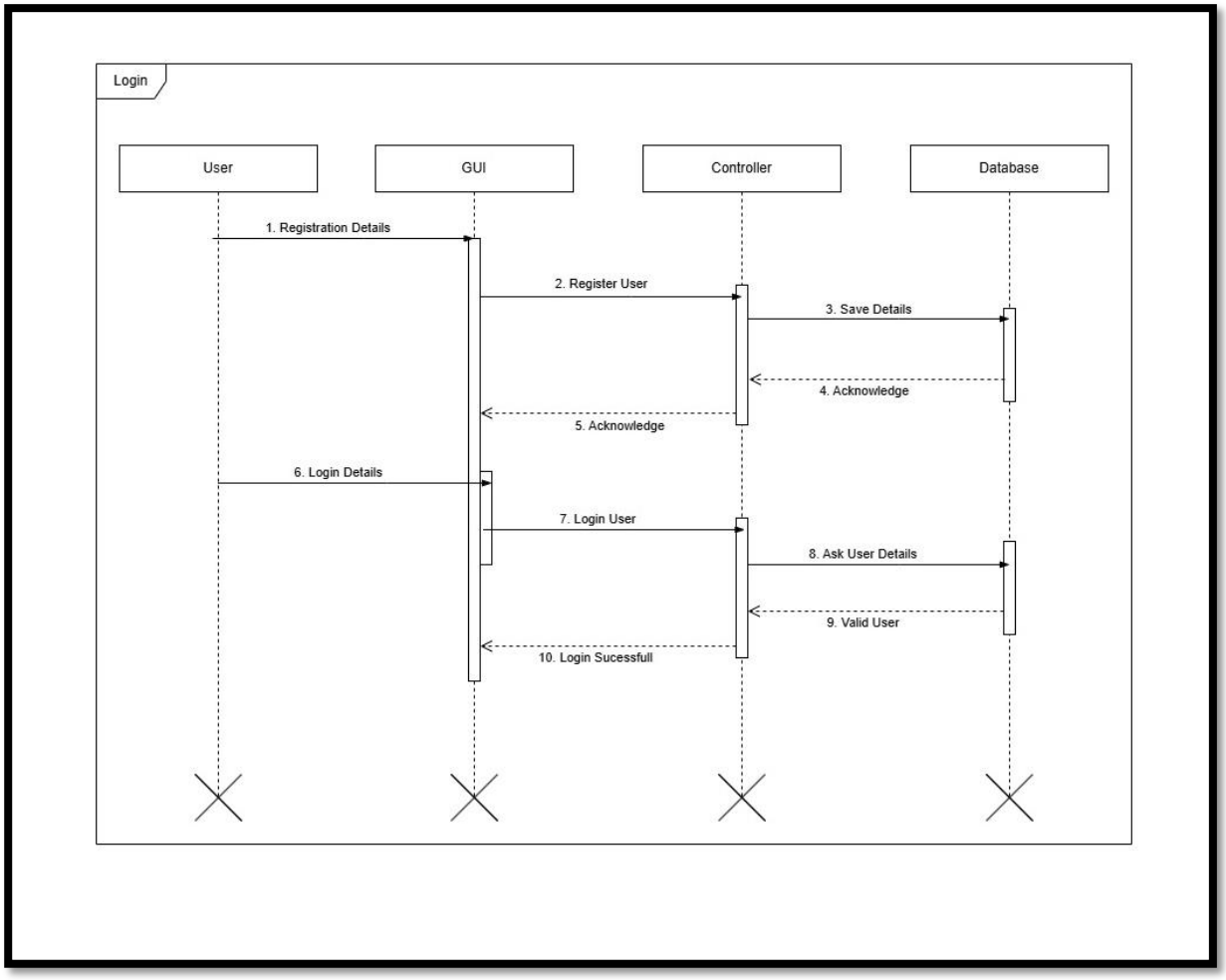


3.10 Sequence Diagram

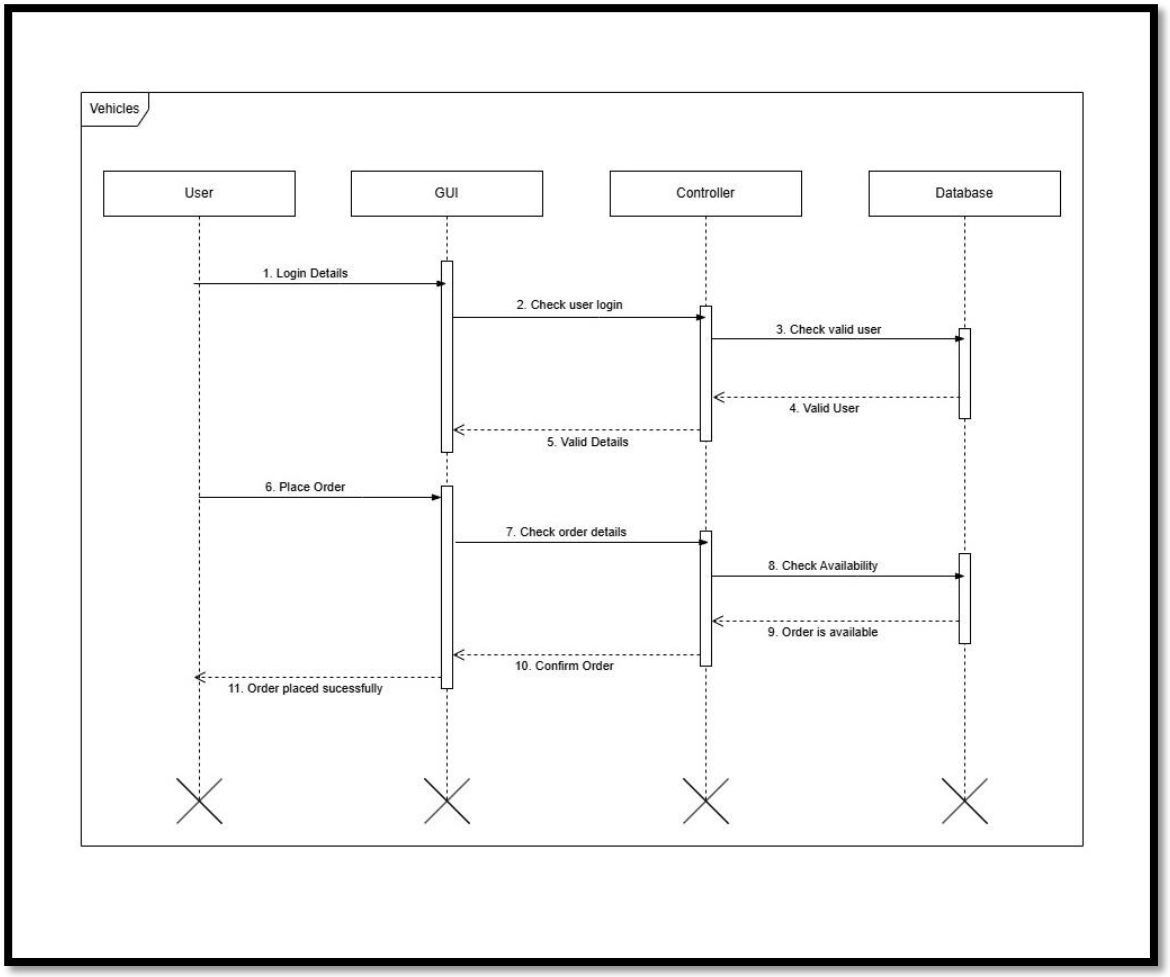
1. System



2. Register & Login

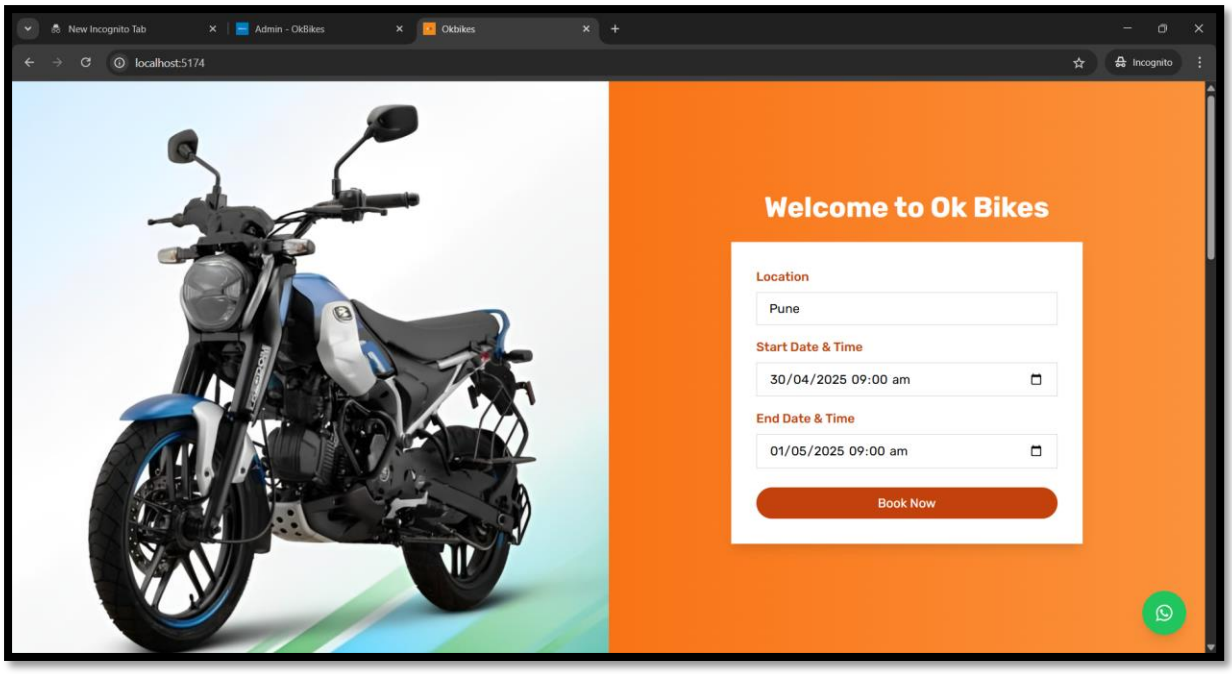


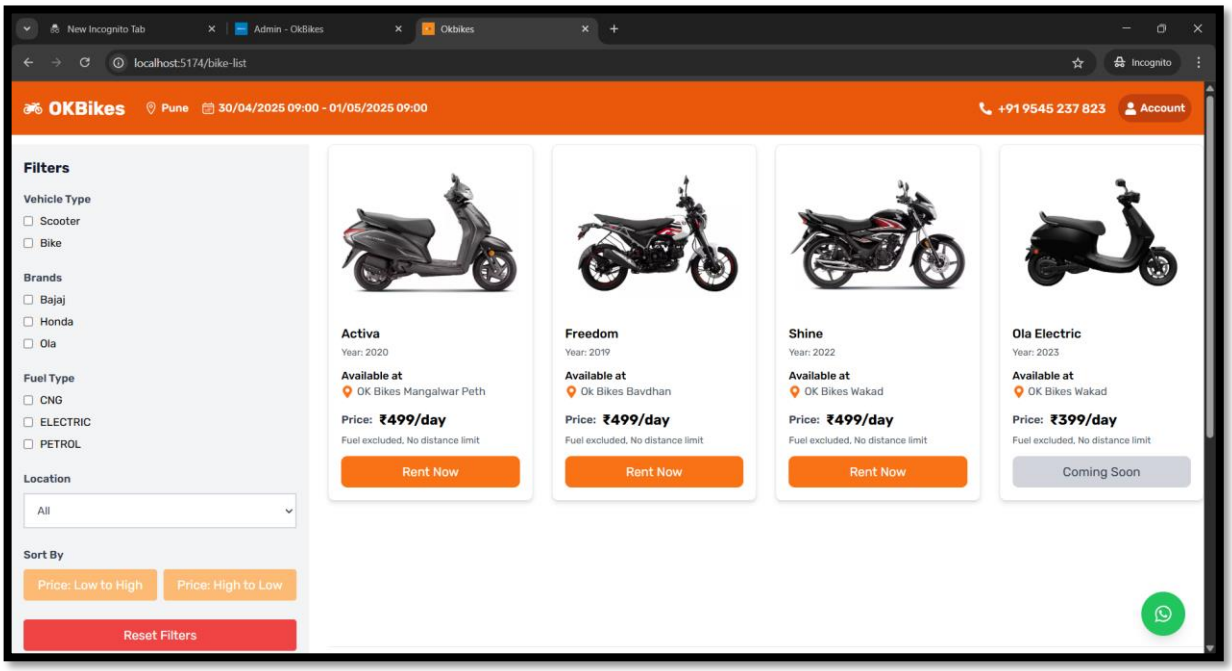
3. Vehicle

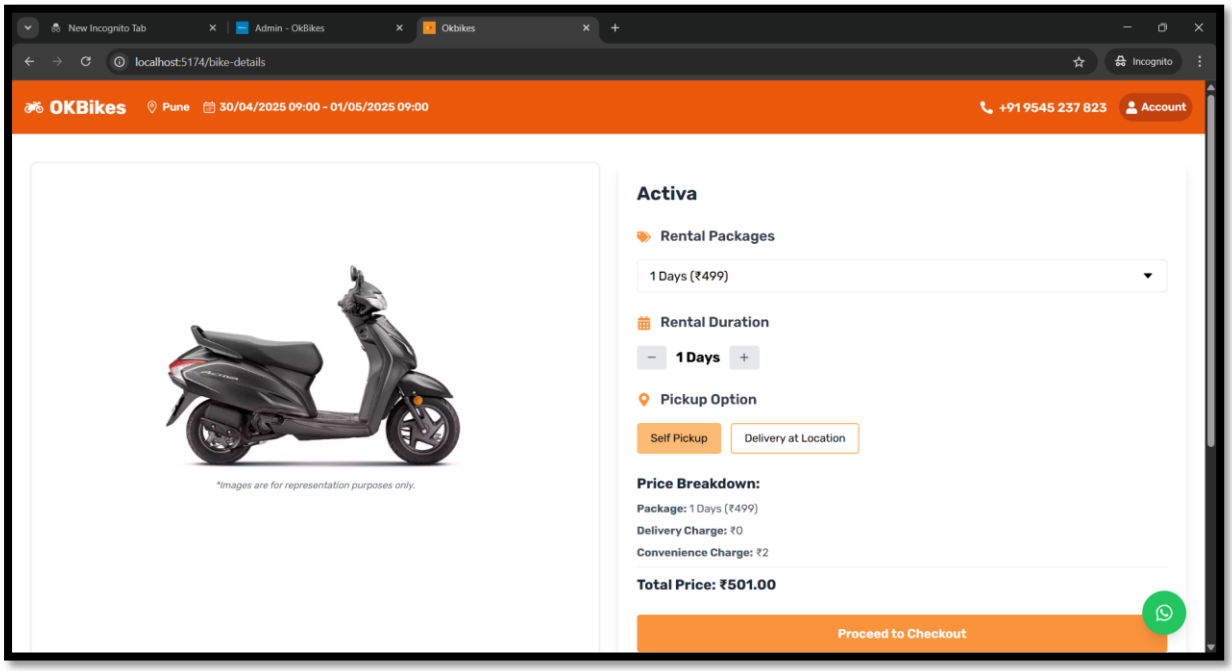

Vehicles
User
GUI
Controller
Database
1. Login Details
2. Check user login
3. Check valid user
4. Valid User
5. Valid Details
6. Place Order
7. Check order details
8. Check Availability
9. Order is available
10. Confirm Order
11. Order placed sucessfully

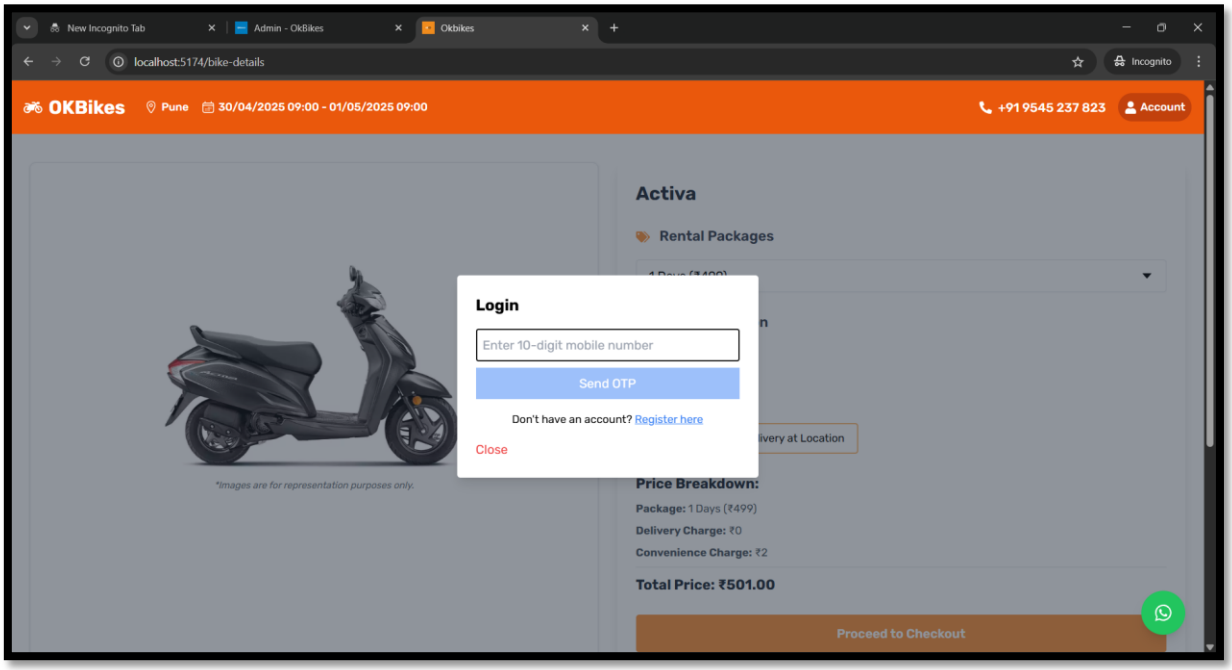
3.10 Sample Input and Output Screens

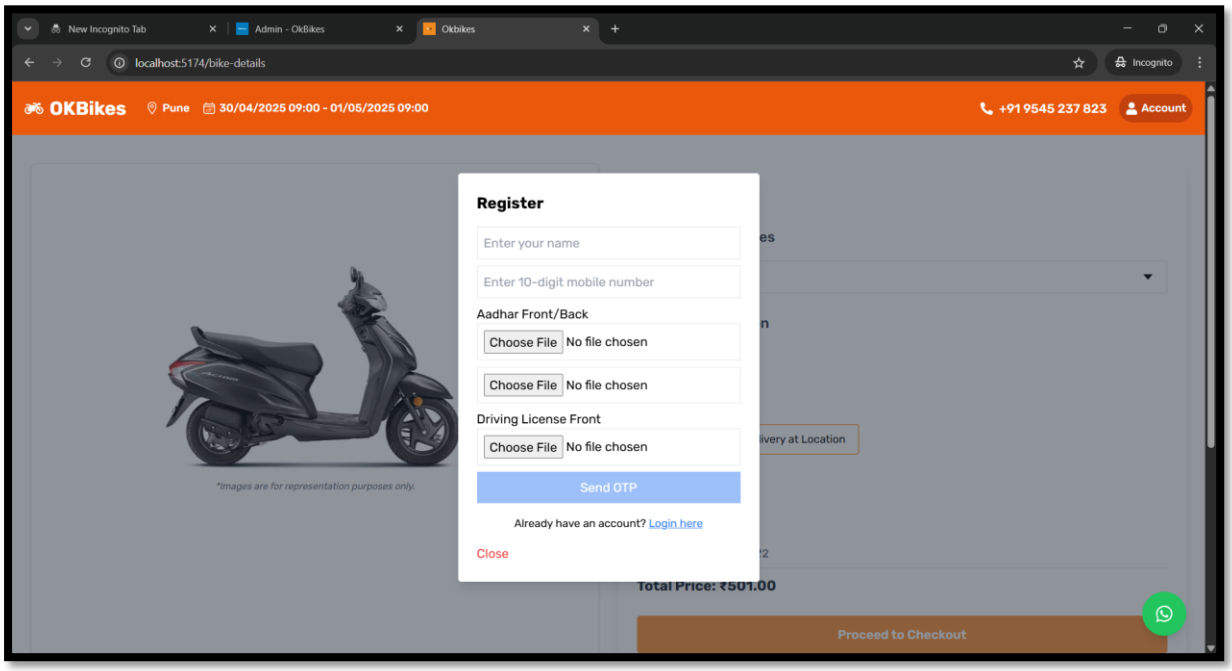
User Side

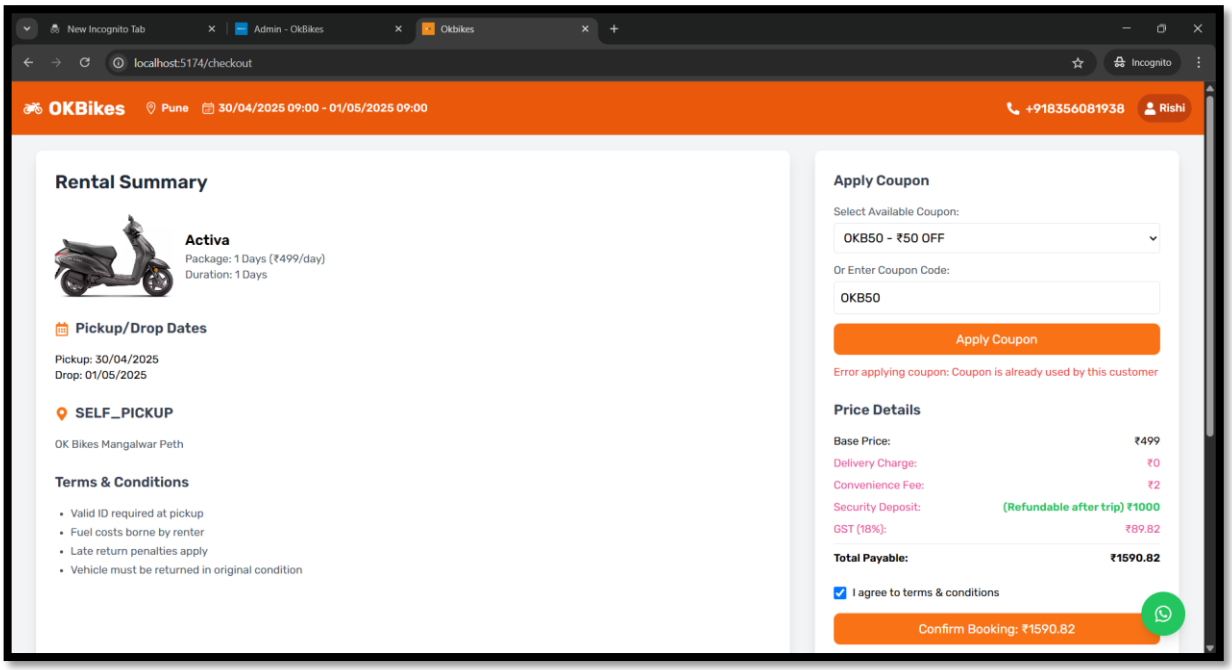


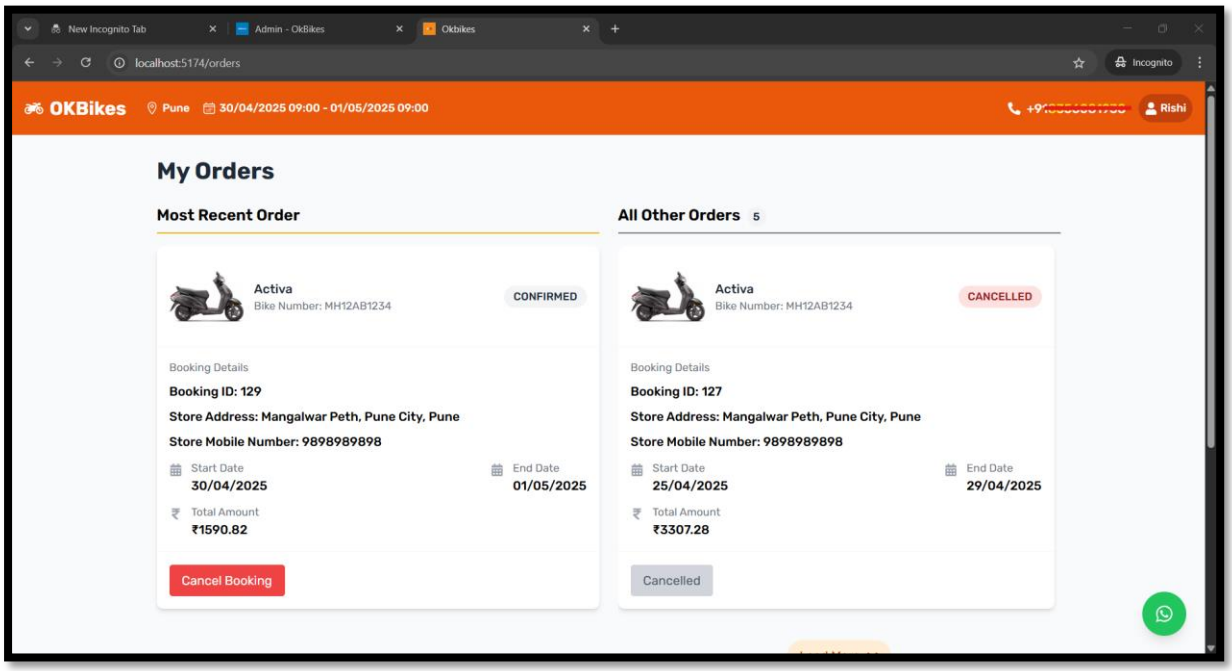


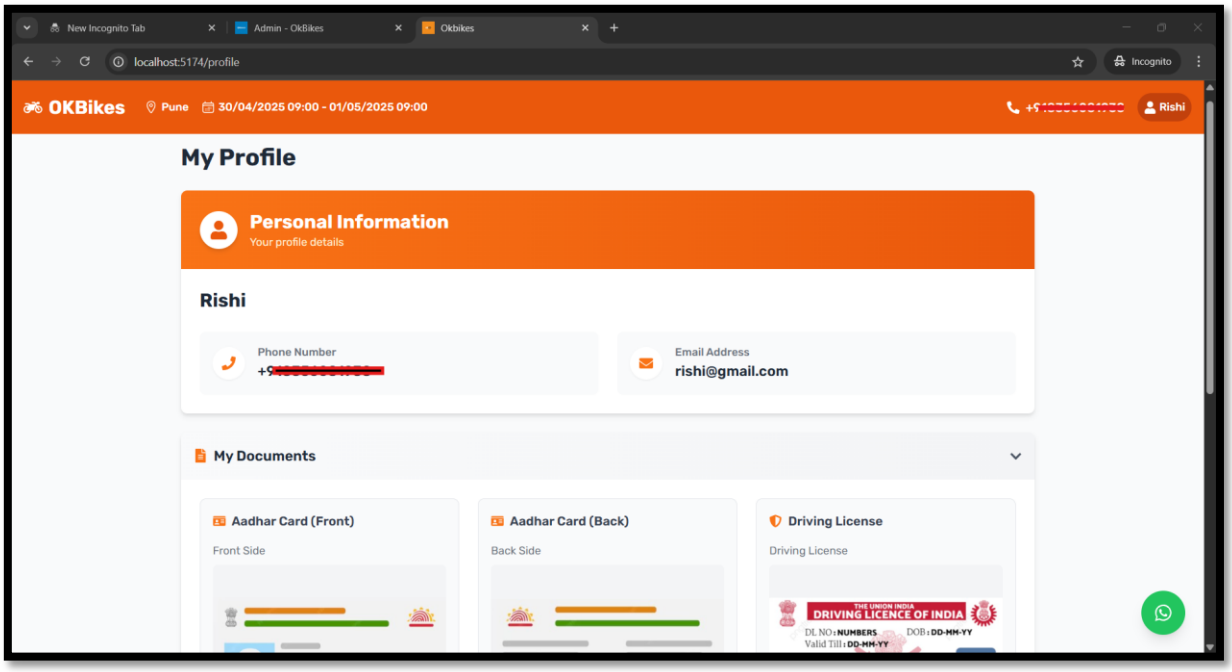

New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5174/bike-details
Incognito
OKBikes
Pune
30/04/2025 09:00 - 01/05/2025 09:00
+91 9545 237 823
Account
*Images are for representation purposes only.
Activa
Rental Packages
1 Days (₹499)
Rental Duration
1 Days
Pickup Option
Self Pickup
Delivery at Location
Price Breakdown:
Package: 1 Days (₹499)
Delivery Charge: ₹0
Convenience Charge: ₹2
Total Price: ₹501.00
Proceed to Checkout

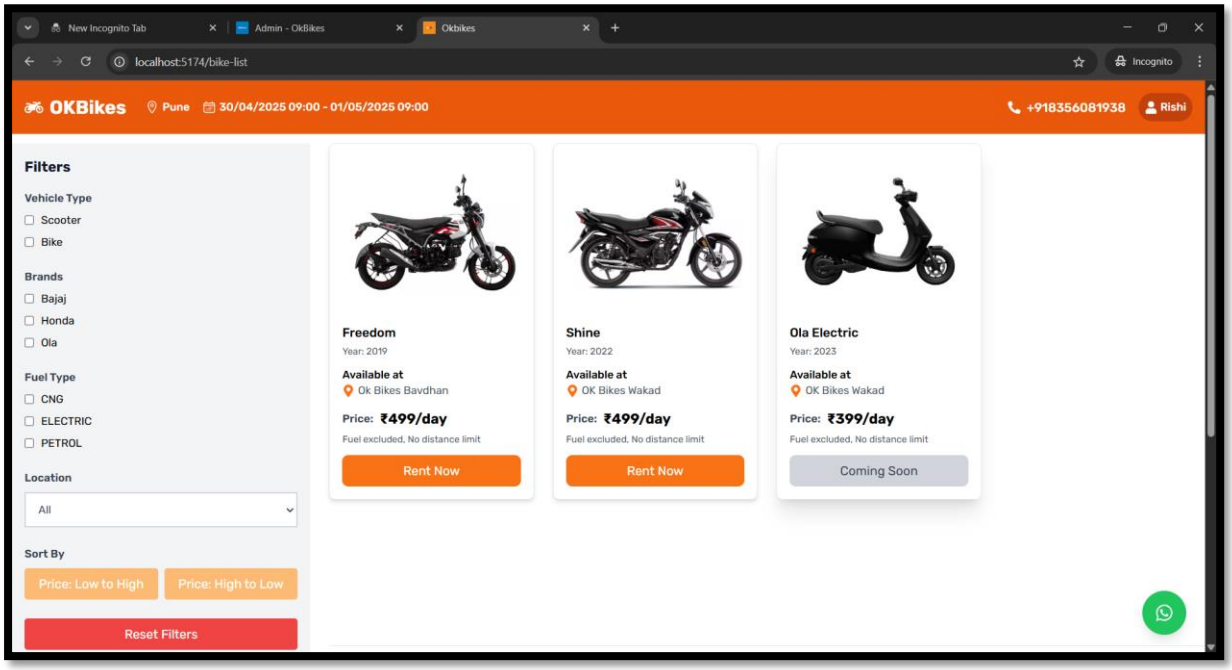

New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5174/bike-details
Incognito
OKBikes
Pune
30/04/2025 09:00 - 01/05/2025 09:00
+91 9545 237 823
Account
*Images are for representation purposes only.
Login
Enter 10-digit mobile number
Send OTP
Don't have an account? Register here
Close
Activa
Rental Packages
Price Breakdown:
Package: 1 Days (₹499)
Delivery Charge: ₹0
Convenience Charge: ₹2
Total Price: ₹501.00
Proceed to Checkout


New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5174/bike-details
Incognito
OKBikes
Pune
30/04/2025 09:00 - 01/05/2025 09:00
+91 9545 237 823
Account
*Images are for representation purposes only.
Register
Enter your name
Enter 10-digit mobile number
Aadhar Front/Back
Choose File
No file chosen
Choose File
No file chosen
Driving License Front
Choose File
No file chosen
Send OTP
Already have an account? Login here
Close
Total Price: ₹501.00
Proceed to Checkout

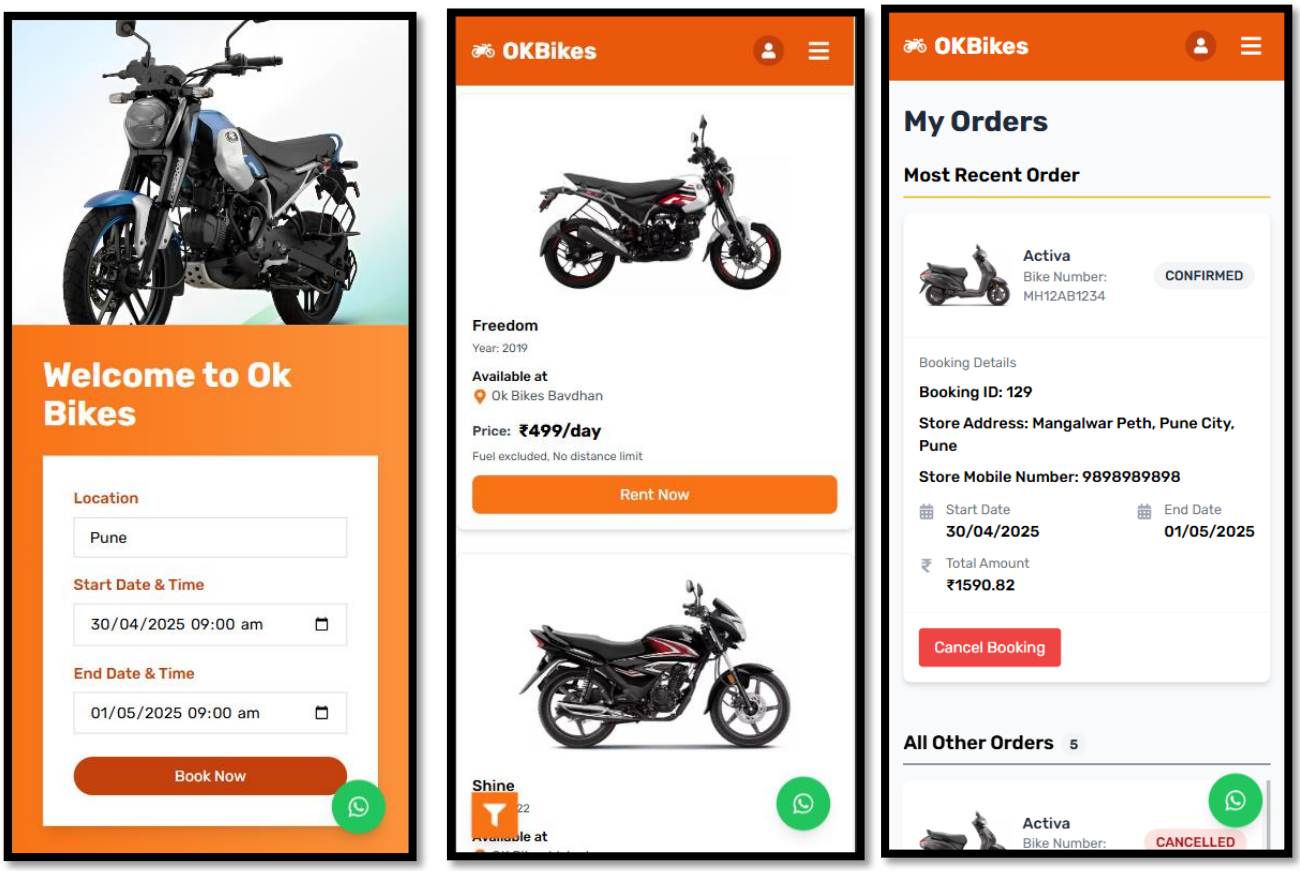

New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5174/checkout
Incognito
OKBikes
Pune
30/04/2025 09:00 - 01/05/2025 09:00
+918356081938
Rishi
Rental Summary
Activa
Package: 1 Days (₹499/day)
Duration: 1 Days
Pickup/Drop Dates
Pickup: 30/04/2025
Drop: 01/05/2025
SELF_PICKUP
OK Bikes Mangalwar Peth
Terms & Conditions
Valid ID required at pickup
Fuel costs borne by renter
Late return penalties apply
Vehicle must be returned in original condition
Apply Coupon
Select Available Coupon:
OKB50 - ₹50 OFF
Or Enter Coupon Code:
OKB50
Apply Coupon
Error applying coupon: Coupon is already used by this customer
Price Details
Base Price: ₹499
Delivery Charge: ₹0
Convenience Fee: ₹2
Security Deposit: (Refundable after trip) ₹1000
GST (18%): ₹89.82
Total Payable: ₹1590.82
I agree to terms & conditions
Confirm Booking: ₹1590.82


New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5174/orders
Incognito
OKBikes
Pune
30/04/2025 09:00 - 01/05/2025 09:00
Rishi
My Orders
Most Recent Order
All Other Orders 5
Activa
Bike Number: MH12AB1234
CONFIRMED
Booking Details
Booking ID: 129
Store Address: Mangalwar Peth, Pune City, Pune
Store Mobile Number: 9898989898
Start Date
30/04/2025
End Date
01/05/2025
Total Amount
₹1590.82
Cancel Booking
Activa
Bike Number: MH12AB1234
CANCELLED
Booking Details
Booking ID: 127
Store Address: Mangalwar Peth, Pune City, Pune
Store Mobile Number: 9898989898
Start Date
25/04/2025
End Date
29/04/2025
Total Amount
₹3307.28
Cancelled

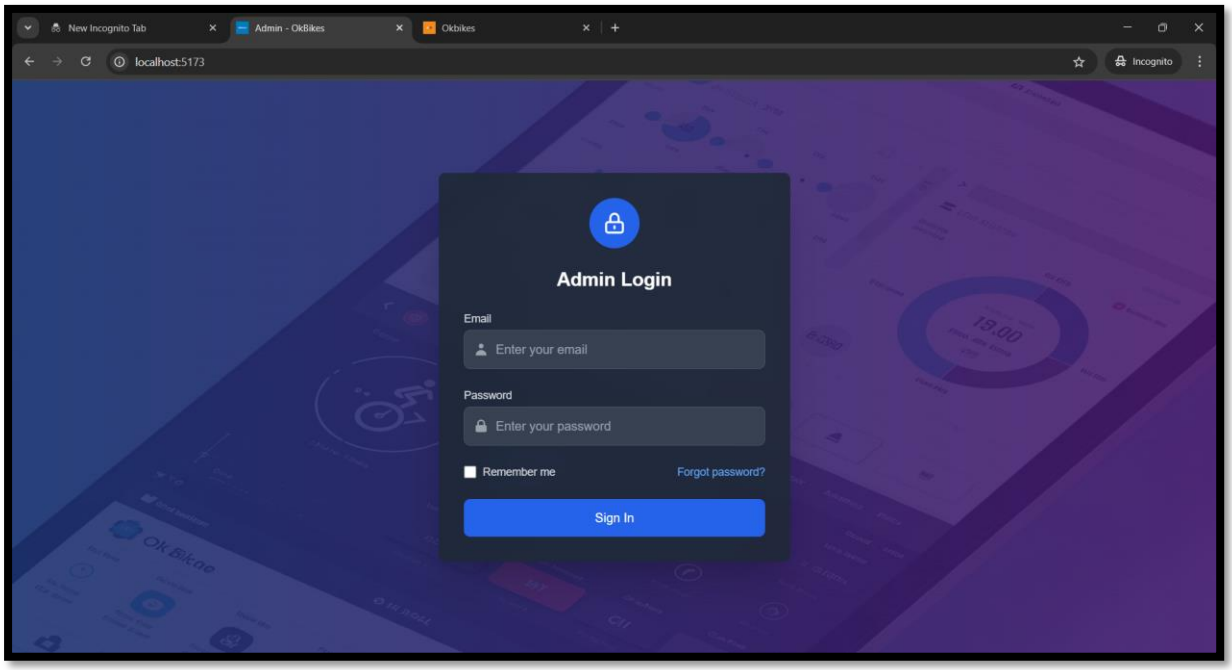

New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5174/profile
Incognito
OKBikes
Pune
30/04/2025 09:00 - 01/05/2025 09:00
Rishi
My Profile
Personal Information
Your profile details
Rishi
Phone Number
Email Address
rishi@gmail.com
My Documents
Aadhar Card (Front)
Front Side
Aadhar Card (Back)
Back Side
Driving License
Driving License
DRIVING LICENCE OF INDIA

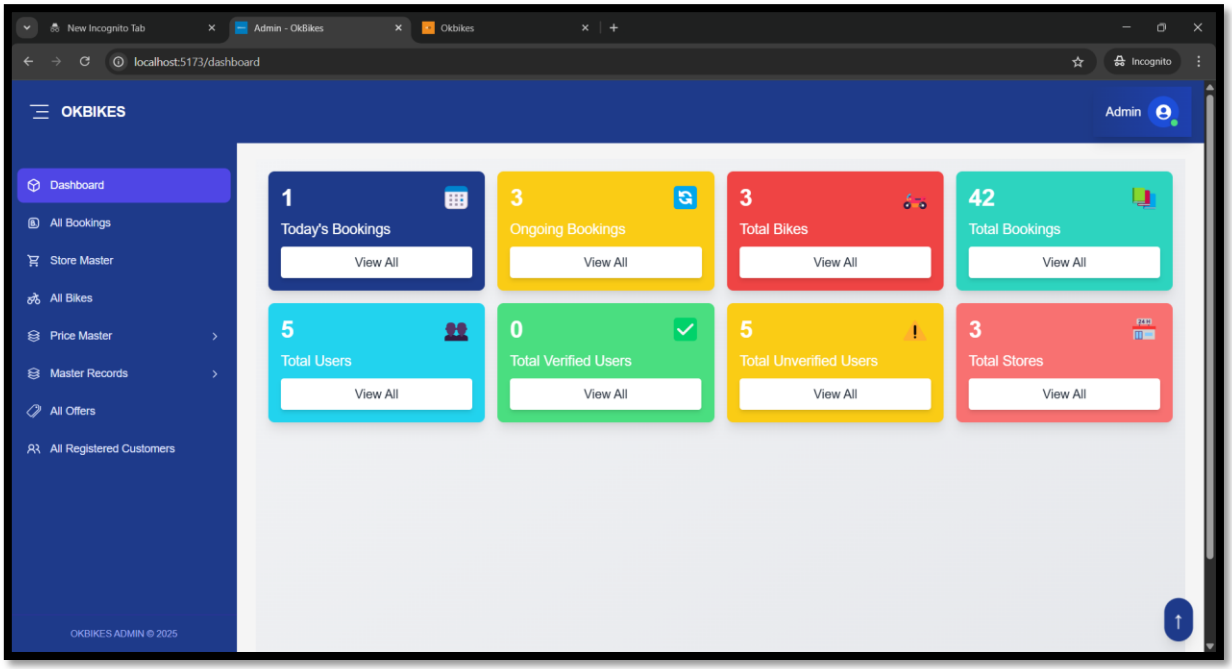


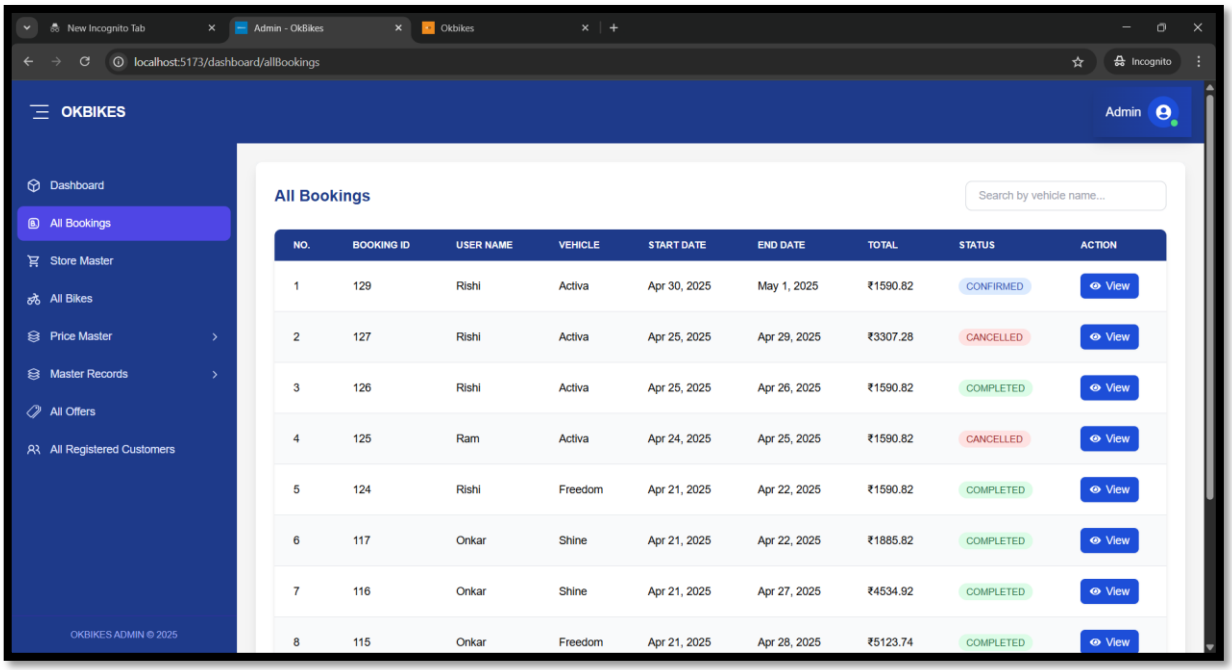
It is mobile responsive also

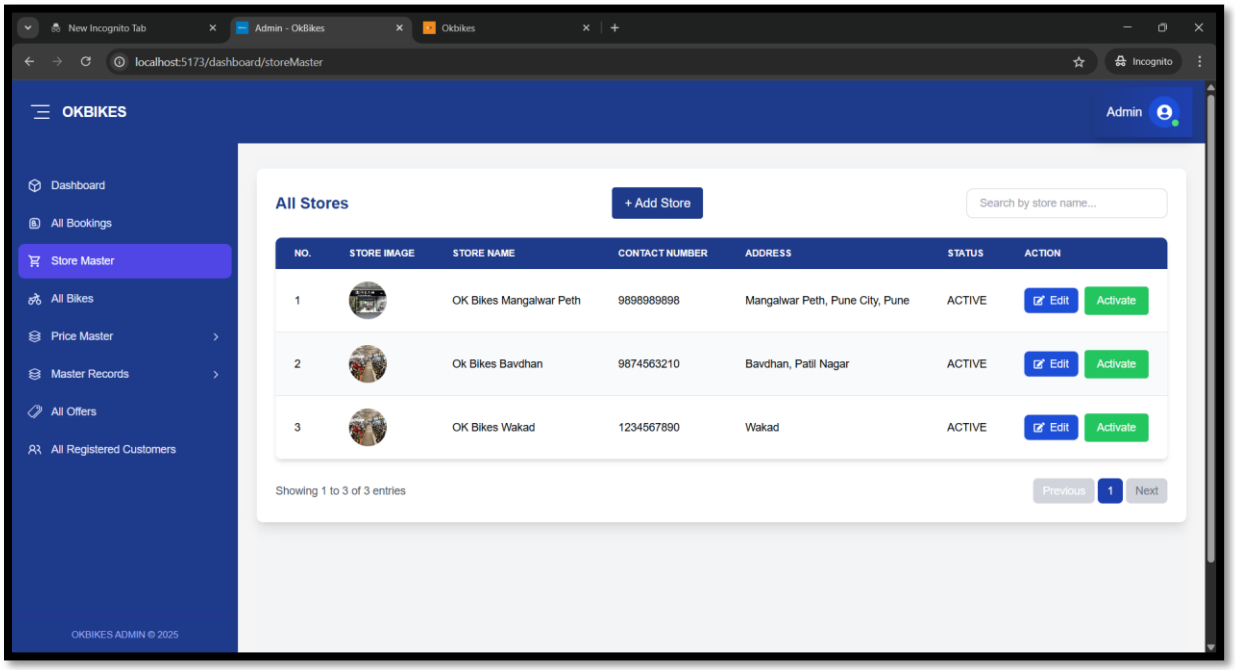


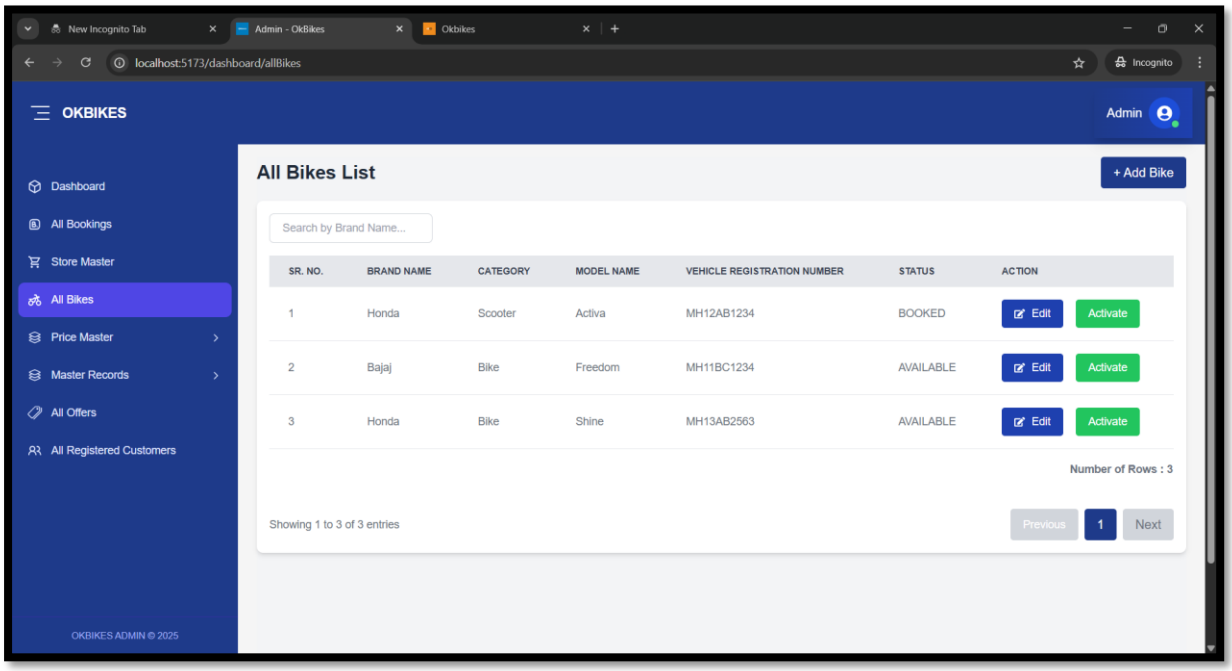
Admin Side

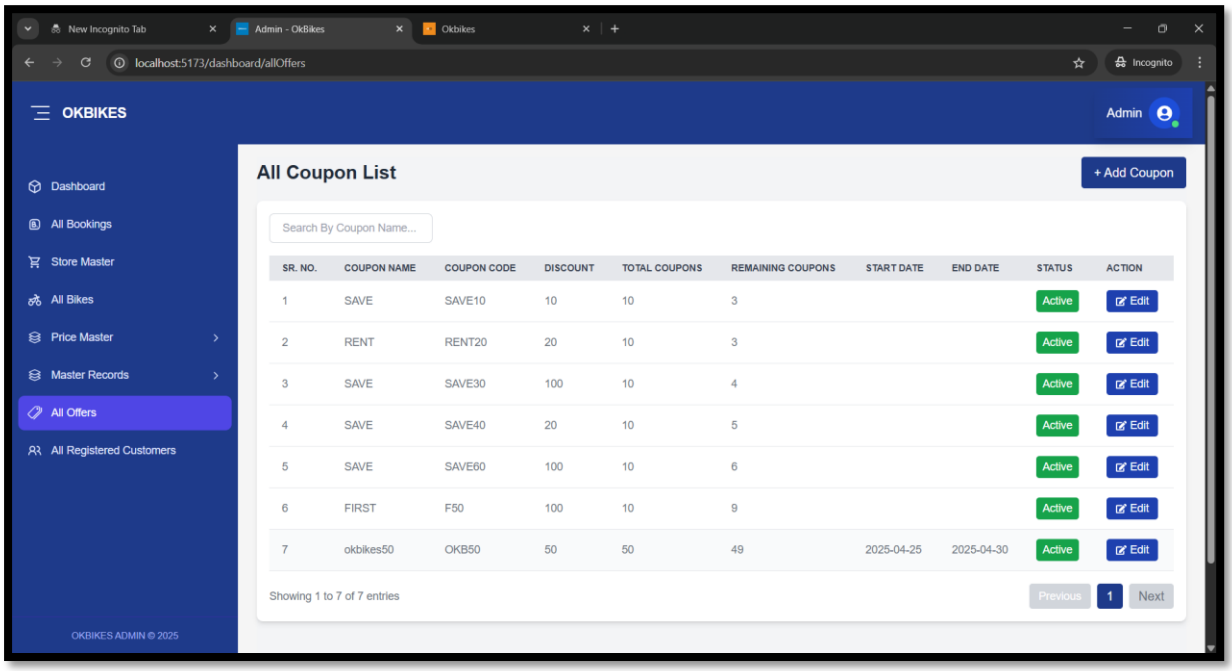

New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5173
Incognito
Admin Login
Email
Enter your email
Password
Enter your password
Remember me
Forgot password?
Sign In

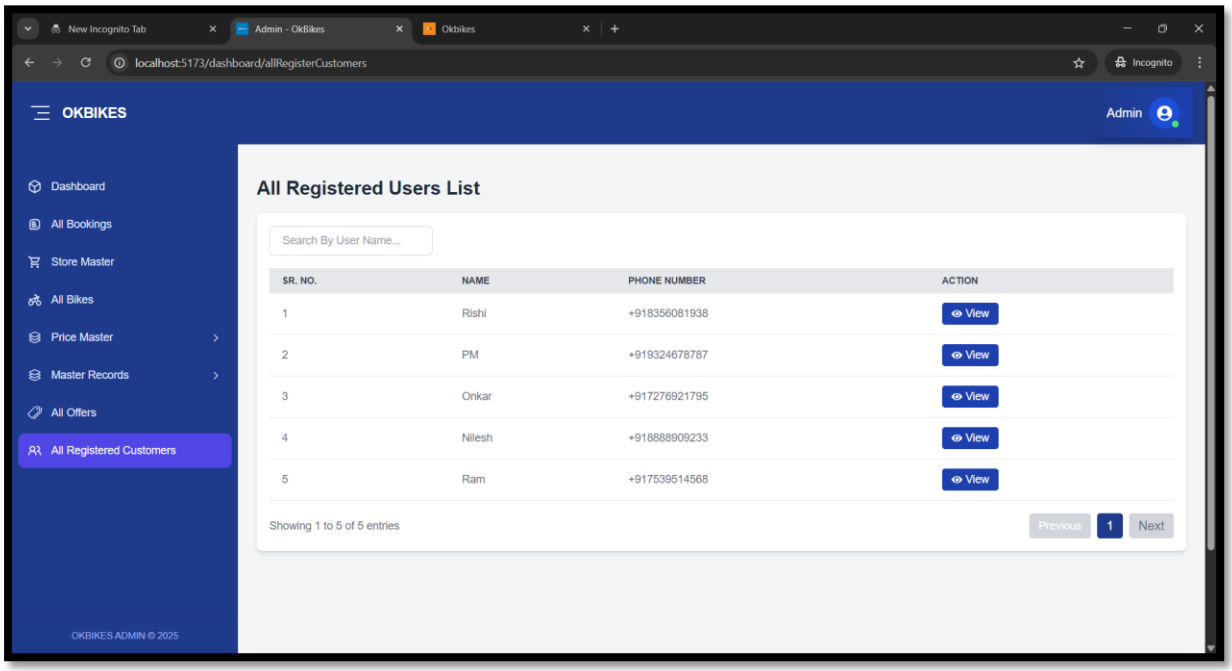

New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5173/dashboard
Incognito
OKBIKES
Admin
Dashboard
All Bookings
Store Master
All Bikes
Price Master
Master Records
All Offers
All Registered Customers
1
Today's Bookings
View All
3
Ongoing Bookings
View All
3
Total Bikes
View All
42
Total Bookings
View All
5
Total Users
View All
0
Total Verified Users
View All
5
Total Unverified Users
View All
3
Total Stores
View All
OKBIKES ADMIN © 2025

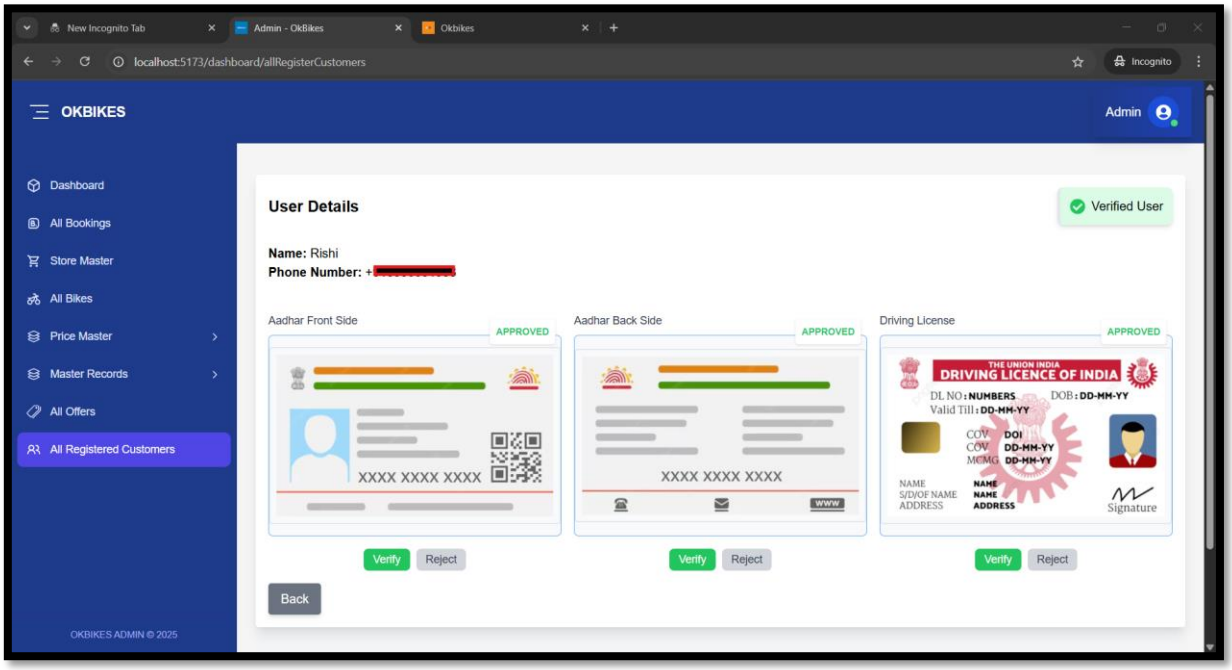

New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5173/dashboard/allBookings
Incognito
OKBIKES
Admin
Dashboard
All Bookings
Store Master
All Bikes
Price Master
Master Records
All Offers
All Registered Customers
OKBIKES ADMIN © 2025
All Bookings
Search by vehicle name...
NO. BOOKING ID USER NAME VEHICLE START DATE END DATE TOTAL STATUS ACTION
1 129 Rishi Activa Apr 30, 2025 May 1, 2025 ₹1590.82 CONFIRMED View
2 127 Rishi Activa Apr 25, 2025 Apr 29, 2025 ₹3307.28 CANCELLED View
3 126 Rishi Activa Apr 25, 2025 Apr 26, 2025 ₹1590.82 COMPLETED View
4 125 Ram Activa Apr 24, 2025 Apr 25, 2025 ₹1590.82 CANCELLED View
5 124 Rishi Freedom Apr 21, 2025 Apr 22, 2025 ₹1590.82 COMPLETED View
6 117 Onkar Shine Apr 21, 2025 Apr 22, 2025 ₹1885.82 COMPLETED View
7 116 Onkar Shine Apr 21, 2025 Apr 27, 2025 ₹4534.92 COMPLETED View
8 115 Onkar Freedom Apr 21, 2025 Apr 28, 2025 ₹5123.74 COMPLETED View


New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5173/dashboard/storeMaster
Incognito
OKBIKES
Admin
Dashboard
All Bookings
Store Master
All Bikes
Price Master
Master Records
All Offers
All Registered Customers
OKBIKES ADMIN © 2025
All Stores
+ Add Store
Search by store name...
NO. STORE IMAGE STORE NAME CONTACT NUMBER ADDRESS STATUS ACTION
1 OK Bikes Mangalwar Peth 9898989898 Mangalwar Peth, Pune City, Pune ACTIVE Edit Activate
2 Ok Bikes Bavdhan 9874563210 Bavdhan, Patil Nagar ACTIVE Edit Activate
3 OK Bikes Wakad 1234567890 Wakad ACTIVE Edit Activate
Showing 1 to 3 of 3 entries
Previous 1 Next

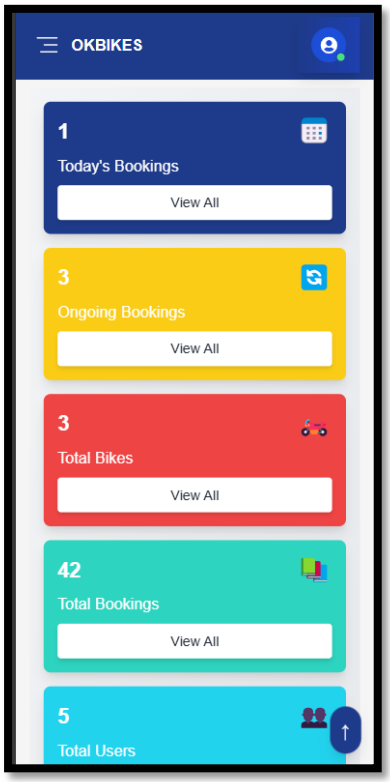

New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5173/dashboard/allBikes
Incognito
OKBIKES
Admin
Dashboard
All Bookings
Store Master
All Bikes
Price Master
Master Records
All Offers
All Registered Customers
All Bikes List
+ Add Bike
Search by Brand Name...
SR. NO.
BRAND NAME
CATEGORY
MODEL NAME
VEHICLE REGISTRATION NUMBER
STATUS
ACTION
1 Honda Scooter Activa MH12AB1234 BOOKED Edit Activate
2 Bajaj Bike Freedom MH11BC1234 AVAILABLE Edit Activate
3 Honda Bike Shine MH13AB2563 AVAILABLE Edit Activate
Number of Rows : 3
Showing 1 to 3 of 3 entries
Previous
1
Next
OKBIKES ADMIN © 2025


New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5173/dashboard/allOffers
Incognito
OKBIKES
Admin
Dashboard
All Bookings
Store Master
All Bikes
Price Master
Master Records
All Offers
All Registered Customers
All Coupon List
+ Add Coupon
Search By Coupon Name...
SR. NO.
COUPON NAME
COUPON CODE
DISCOUNT
TOTAL COUPONS
REMAINING COUPONS
START DATE
END DATE
STATUS
ACTION
1 SAVE SAVE10 10 10 3 Active Edit
2 RENT RENT20 20 10 3 Active Edit
3 SAVE SAVE30 100 10 4 Active Edit
4 SAVE SAVE40 20 10 5 Active Edit
5 SAVE SAVE60 100 10 6 Active Edit
6 FIRST F50 100 10 9 Active Edit
7 okbikes50 OKB50 50 50 49 2025-04-25 2025-04-30 Active Edit
Showing 1 to 7 of 7 entries
Previous
1
Next
OKBIKES ADMIN © 2025


New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5173/dashboard/allRegisterCustomers
Incognito
OKBIKES
Admin
Dashboard
All Bookings
Store Master
All Bikes
Price Master
Master Records
All Offers
All Registered Customers
All Registered Users List
Search By User Name...
SR. NO.
NAME
PHONE NUMBER
ACTION
1
Rishi
+918356081938
View
2
PM
+919324678787
View
3
Onkar
+917276921795
View
4
Nilesh
+918888909233
View
5
Ram
+917539514568
View
Showing 1 to 5 of 5 entries
Previous
1
Next
OKBIKES ADMIN © 2025


New Incognito Tab
Admin - OkBikes
Okbikes
localhost:5173/dashboard/allRegisterCustomers
Incognito
OKBIKES
Admin
Dashboard
All Bookings
Store Master
All Bikes
Price Master
Master Records
All Offers
All Registered Customers
User Details
Verified User
Name: Rishi
Phone Number: +[redacted]
Aadhar Front Side
APPROVED
XXXX XXXX XXXX
Aadhar Back Side
APPROVED
XXXX XXXX XXXX
Driving License
APPROVED
THE UNION INDIA
DRIVING LICENCE OF INDIA
DL NO : NUMBERS
DOB : DD-MM-YY
Valid Till : DD-MM-YY
COV
DOI
COV
DD-MM-YY
MCWG
DD-MM-YY
NAME
S/D/OF NAME
ADDRESS
NAME
NAME
ADDRESS
Signature
Verify
Reject
Verify
Reject
Verify
Reject
Back
OKBIKES ADMIN © 2025

It is mobile responsive also



CHAPTER 4 : CODING

4.1 Algorithms

Algorithm 1: Customer Self-Registration

Covers: New user creates account via web app

Steps:

1. User opens the Okbikes website
2. Clicks on "Register"
3. Enters name, mobile number, and upload documents.
4. User enters OTP for verification
5. Backend verifies OTP and creates user in users table
6. User is redirected to login
7. End

Algorithm 2: Customer Login via OTP

Covers: Existing user logs in using OTP

Steps:

1. User clicks on "Login"
2. Enters registered mobile number
3. System sends OTP
4. User enters OTP
5. Backend verifies OTP
6. If valid, generate session token and redirect to dashboard
7. Else, display error and retry option
8. End

Algorithm 3: Cancel a Booking

Covers: User cancels a confirmed booking

Steps:

1. User goes to "My Bookings"
2. Clicks on "Cancel Booking"
3. System checks if cancellation is within policy time
4. Booking status is updated to “Cancelled” in DB
5. End

Algorithm 4: Booking a Vehicle

Covers: Customer books a bike or scooter

Steps:

1. User logs into their account
2. Clicks on "Book a Ride"
3. Selects vehicle type, rental package, location, and duration
4. System fetches available vehicles matching filters
5. User selects a vehicle and adds coupon code (optional)
6. System calculates total fare and applies any discount
7. User confirms and makes payment
8. Booking is recorded in bookings table
9. End

Algorithm 5: Apply Coupon Code During Booking

Covers: Discount via promo code

Steps:

1. User reaches checkout page
2. Enters promo/coupon code
3. System checks validity (expiry, usage limit, user eligibility)
4. If valid, applies discount and updates total fare
5. If invalid, displays appropriate message
6. Proceeds with updated booking cost
7. End

Algorithm 6: Generate Invoice

Covers: After rental period ends

Steps:

1. Rental period completes or vehicle is returned
2. System calculates total fare, discounts, taxes
3. Invoice is generated as PDF
4. Invoice record is stored in invoices table
5. Option to download from "Ride History"
6. End

Algorithm 7: Admin Adds New Vehicle to Inventory

Covers: Admin onboarding a new bike/scooter

Steps:

1. Admin logs into the Admin Panel
2. Navigates to "Vehicle Inventory" section
3. Clicks "Add Vehicle"
4. Fills in details: vehicle type, model, number, location, image, availability
5. System stores data in vehicles table
6. Vehicle becomes available for booking
7. End

Algorithm 8: Admin Manages Vehicle Maintenance

Covers: Admin schedules and tracks vehicle maintenance

Steps:

1. Admin logs into the Admin Panel
2. Navigate to "Vehicle Maintenance" section
3. Selects a vehicle from the inventory list
4. Clicks "Schedule Maintenance"
5. Admin sets the maintenance date and time
6. Maintenance details (e.g., service type, cost) are added to the system
7. System updates the vehicle's status to "In Maintenance"
8. Once maintenance is completed, the vehicle's status is updated to "Available"
9. Notification is sent to admin and user when vehicle is ready for use
10. End

Algorithm 9: Admin Views Reports and Analytics

Covers: Admin generates system reports for performance tracking

Steps:

1. Admin logs into the Admin Panel
2. Navigates to the "Reports" section
3. Selects the desired report (e.g., booking trends, earnings, vehicle usage)
4. Admin chooses the report time frame (e.g., daily, weekly, monthly)
5. System fetches relevant data from the database
6. Report is displayed in tabular or graphical form
7. Admin can export report to PDF or CSV format
8. End

Algorithm 10: Vehicle Maintenance Reminder

Covers: Admin receives a reminder for vehicle maintenance

Steps:

1. Vehicle's maintenance date is recorded in the system
2. System schedules a reminder based on the maintenance date (e.g., 1 week before)
3. Admin logs into the Admin Panel to view and confirm the maintenance task
4. Maintenance is scheduled with relevant service providers
5. End

Algorithm 11: Vehicle Returns and Final Billing

Covers: Customer returns the vehicle and finalizes payment

Steps:

1. User returns the vehicle to the designated drop-off location
2. System checks the vehicle's return status and condition
3. If vehicle is returned within time, calculate total fare (e.g., additional charges for damages or overtime)
4. System generates final bill and displays on user's screen
5. User proceeds to make the final payment (if additional charges are applicable)
6. Update vehicle status to "Available" for next rental
7. End

4.2 Code Snippets

- **Frontend**

1. User Side

HomePage.jsx

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import {
  FaTimes,
  FaBicycle,
  FaHandshake,
  FaPhone,
  FaCheck,
  FaMapMarkerAlt,
  FaCreditCard,
} from "react-icons/fa";
import { useGlobalState } from "../context/GlobalStateContext";
import axios from "axios";
import './HomePage.css'; // Import the CSS file

const HomePage = () => {
  const navigate = useNavigate();
  const { formData, setFormData } = useGlobalState();
  const [popupOpen, setPopupOpen] = useState(false);
  const [searchTerm, setSearchTerm] = useState("");
  const [errors, setErrors] = useState({});
  const [selectedCityImage, setSelectedCityImage] = useState(
    "/banner-freedom.jpg"
  );
  const [cities, setCities] = useState([]);
  const [availableBikes, setAvailableBikes] = useState([]);
  const [loading, setLoading] = useState(false);
  const [lastFetchError, setLastFetchError] = useState(null);
```

```
// Improved time handling function
 const formatDateForInput = (date) => {
   const year = date.getFullYear();
   const month = String(date.getMonth() + 1).padStart(2, "0");
   const day = String(date.getDate()).padStart(2, "0");
   const hours = String(date.getHours()).padStart(2, "0");
   const minutes = String(date.getMinutes()).padStart(2, "0");
   return `${year}-${month}-${day}T${hours}:${minutes}`;
 };

 const roundToNextHour = (date) => {
   const roundedDate = new Date(date);
   roundedDate.setHours(roundedDate.getHours() + 1, 0, 0, 0);
   return roundedDate;
 };

 useEffect(() => {
   // Scroll to the top of the page when the component mounts
   window.scrollTo(0, 0);

   const fetchCities = async () => {
     try {
       const response = await axios.get(`${import.meta.env.VITE_BASE_URL}/city/all`, {
         headers: {
           "Content-Type": "application/json",
         },
         withCredentials: true,
       });
```

BikeListPage.jsx

```
import React, { useState, useEffect, useRef } from "react";
import { useNavigate, useLocation } from "react-router-dom";
import {
  FaFilter,
  FaTimes,
  FaSpinner,
  FaMapMarkerAlt,
} from "react-icons/fa";
import axios from "axios";
import { useAuth } from "../context/AuthContext"; // Import useAuth for token management

const BikeListPage = () => {
  const navigate = useNavigate();
  const location = useLocation();
  const { token } = useAuth(); // Use the token from AuthContext

  // Define static locations as strings
  const staticLocation1 = "OK Bikes Mangalwar Peth";
  const staticLocation2 = "Ok Bikes Bavdhan";
  const staticLocation3 = "OK Bikes Wakad";

  const { formData } = location.state || {};

  // Static bike details
  const staticBikeDetails = {
    id: 1,
    model: "Ola Electric",
    image: "/ola.jpg",
    perDayRent: 399,
    deposit: 0, // Assuming no deposit mentioned
    registrationYear: 2023,
    storeName: staticLocation3,
    categoryName: "Scooter",
    categoryId: 1,
```

```
  fuelType: "ELECTRIC",
  brand: "Ola",
  vehicleType: "Scooter", // Ensure vehicleType is set
};

const [bikes, setBikes] = useState([]);
const [filteredBikes, setFilteredBikes] = useState([]);
const [selectedFilters, setSelectedFilters] = useState({
  vehicleType: [],
  brands: [],
  fuelType: [],
  location: "",
});
const [sortOrder, setSortOrder] = useState("");
const [showFilters, setShowFilters] = useState(false);
const [loading, setLoading] = useState(true);
const [currentPage, setCurrentPage] = useState(1);
const filterRef = useRef(null);
const bikesPerPage = 8;

// Log the token when the component mounts
useEffect(() => {
  console.log("Token from AuthContext:", token);
}, [token]);

useEffect(() => {
  if (formData) {
    fetchAvailableBikes();
  } else {
    console.error(
      "No form data found. Please return to the home page and make a selection."
    );
    navigate("/");
  }
}, [formData]);
```

BikeDetailPage.jsx

```
import React, { useState, useEffect } from "react";
import { useLocation, useNavigate } from "react-router-dom";
import { FaMapMarkerAlt, FaCalendarAlt, FaTags } from "react-icons/fa";
import { AiOutlinePlus, AiOutlineMinus, AiOutlineCaretDown, AiOutlineCaretUp } from "react-icons/ai";
import LoginPopup from "../components/LoginPopup";
import RegistrationPopup from "../components/RegistrationPopup";
import { motion, AnimatePresence } from "framer-motion";

const BikeDetailsPage = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const bike = location.state || {};
  const [isLoginPopupOpen, setIsLoginPopupOpen] = useState(false);
  const [isRegistrationPopupOpen, setIsRegistrationPopupOpen] = useState(false);
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const serviceCharge = 2;

  const [packages, setPackages] = useState([]);
  const [selectedPackage, setSelectedPackage] = useState(null);
  const [oneDayPackage, setOneDayPackage] = useState(null);
  const [dropdownOpen, setDropdownOpen] = useState(false);
  const [pickupOption, setPickupOption] = useState("SELF_PICKUP");
  const [showAddressPopup, setShowAddressPopup] = useState(false);
  const [addressDetails, setAddressDetails] = useState({
    fullAddress: "",
    pinCode: "",
    nearby: "",
  });
  const [rentalDays, setRentalDays] = useState(1);
  const [showToast, setShowToast] = useState(false);
  const [isAnimating, setIsAnimating] = useState(false);
```

```
useEffect(() => {
    const token = localStorage.getItem("jwtToken");
    console.log("BikeDetailsPage - Token in localStorage:", token);
    if (token) {
      setIsLoggedIn(true);
    }

    if (bike.categoryId) {
      fetchPackages(bike.categoryId);
    }
// Scroll to top when the component mounts
    window.scrollTo(0, 0);
  }, [bike.categoryId]);

  useEffect(() => {
    // Automatically select the package based on rental days
    if (packages.length > 0) {
      const packageForDays = packages.find(pkg => pkg.days === rentalDays);
      if (packageForDays) {
        setSelectedPackage(packageForDays);
      } else {
        setSelectedPackage(packages[0]);
      }
    }
  }, [rentalDays, packages]);
```

```
const fetchPackages = async (categoryId) => {
   try {
                                        const response = await
fetch(`${import.meta.env.VITE_BASE_URL}/package/list/${categoryId}`);
     const data = await response.json();

     // Filter active packages
     const activePackages = data.filter(pkg => pkg.active);
     setPackages(activePackages);
```

2. Admin Side

Home.jsx

```
import React, { useEffect, useState } from "react";
import { useSpring, animated, useTrail, useChain, useSpringRef } from 'react-spring';
import { useTable } from 'react-table';
import apiClient from "../api/apiConfig";

const Home = () => {
  const [users, setUsers] = useState([]);
  const [bookings, setBookings] = useState([]);
  const [stores, setStores] = useState([]);
  const [bikes, setBikes] = useState([]);
  const [loading, setLoading] = useState(true);
  const [currentPage, setCurrentPage] = useState(0);
  const [totalPages, setTotalPages] = useState(1);
  const [showUsers, setShowUsers] = useState(false);
  const [showBookings, setShowBookings] = useState(false);
  const [showStores, setShowStores] = useState(false);
  const [showBikes, setShowBikes] = useState(false);
  const [showTodaysBookings, setShowTodaysBookings] = useState(false);
  const [todaysBookings, setTodaysBookings] = useState([]);
  const [verifiedUsers, setVerifiedUsers] = useState([]);
  const [unverifiedUsers, setUnverifiedUsers] = useState([]);

  // Spring references for animation chaining
```

```
  const cardsSpringRef = useSpringRef();
  const tableSpringRef = useSpringRef();

  useEffect(() => {
    window.scrollTo({ top: 0, behavior: 'smooth' });

    const fetchUsers = async () => {
      setLoading(true);
      try {
        const response = await apiClient.get("/users/all", {
          params: { page: currentPage, size: 10, sortBy: 'id', sortDirection: 'asc' }
        });
        setUsers(response.data.content);
        setTotalPages(response.data.totalPages);

        // Fetch verified and unverified users
        const verified = response.data.content.filter(user => user.isVerified);
        const unverified = response.data.content.filter(user => !user.isVerified);
        setVerifiedUsers(verified);
        setUnverifiedUsers(unverified);
      } catch (error) {
        console.error("Error fetching users data:", error);
      } finally {
        setLoading(false);
      }
    };
  const fetchBookings = async () => {
      try {
        const response = await apiClient.get("/booking/all");
            const sortedBookings = response.data.sort((a, b) => new Date(b.startDate) - new
Date(a.startDate));
        setBookings(sortedBookings);

        // Fetch usernames for each booking
        const bookingWithUsernames = await Promise.all(
```

```
        sortedBookings.map(async (booking) => {
          const userResponse = await apiClient.get(`/users/${booking.userId}`);
          return { ...booking, userName: userResponse.data.name };
        })
      );
      setBookings(bookingWithUsernames);

      // Filter today's bookings
      const today = new Date().toISOString().split('T')[0];
                    const todaysBookings = sortedBookings.filter(booking =>
booking.startDate.includes(today));
      setTodaysBookings(todaysBookings);
    } catch (error) {
```

AllRegisteredCustomers.jsx

```
import React, { useEffect, useState } from "react";
import { FaEye, FaCheckCircle, FaTimesCircle } from "react-icons/fa";
import apiClient from "../api/apiConfig";
import { motion } from "framer-motion";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

const AllRegisterCustomers = () => {
  const [data, setData] = useState([]);
  const [searchQuery, setSearchQuery] = useState("");
  const [currentPage, setCurrentPage] = useState(1);
  const [loading, setLoading] = useState(true);
  const [itemsPerPage] = useState(6);
  const [totalPages, setTotalPages] = useState(1);
  const [selectedUser, setSelectedUser] = useState(null);
  const [viewMode, setViewMode] = useState(false);
  const [selectedImage, setSelectedImage] = useState(null);
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [documentStatus, setDocumentStatus] = useState({
    aadharFrontSide: 'PENDING',
    aadharBackSide: 'PENDING',
```

```
    drivingLicense: 'PENDING',
  });

  // Compute verification status based on document statuses
  const userVerificationStatus = () => {
    if (Object.values(documentStatus).every(status => status === 'APPROVED')) {
      return { status: "Verified User", color: "green" };
    } else if (Object.values(documentStatus).some(status => status === 'REJECTED')) {
      return { status: "Unverified User", color: "red" };
    } else {
      return { status: "Pending Verification", color: "orange" };
    }
  };

const fetchUsers = async () => {
    setLoading(true);
    try {
      const response = await apiClient.get("/users/all", {
        params: {
          page: currentPage - 1,
          size: itemsPerPage,
        },
      });

      // Remove the first user record
      const filteredData = response.data.content.slice(1);

      setData(filteredData);
      setTotalPages(response.data.totalPages);
    } catch (error) {
      console.error("Error fetching user data:", error);
    } finally {
      setLoading(false);
    }
  };
  fetchUsers();
```

```
}, [currentPage, itemsPerPage]);

const filteredData = data.filter(
  (item) =>
    item.name && item.name.toLowerCase().includes(searchQuery.toLowerCase())
);
```

Backend

UserController.java

```
package com.prasad.bikebookingapplication.controllers;

import com.prasad.bikebookingapplication.constants.CommonConstants;
import com.prasad.bikebookingapplication.dtos.UserResponseDto;
import com.prasad.bikebookingapplication.models.User;
import com.prasad.bikebookingapplication.services.JwtService;
import com.prasad.bikebookingapplication.services.UserService;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/users")
public class UserController {
    private final UserService userService;
    private final JwtService jwtService;

    @Autowired
    public UserController(UserService userService, JwtService jwtService) {
        this.userService = userService;
        this.jwtService = jwtService;
    }

    @GetMapping("/{id}")
    public UserResponseDto getUserById(@PathVariable Long id) {
        return new UserResponseDto(userService.getUserById(id));
    }
```

BookingController.java

```
package com.prasad.bikebookingapplication.controllers;


import com.prasad.bikebookingapplication.dtos.*;
//import com.prasad.bikebookingapplication.dtos.DocumentUploadRequest;
import com.prasad.bikebookingapplication.models.Booking;
import com.prasad.bikebookingapplication.models.DocumentVerification;
import com.prasad.bikebookingapplication.models.Role;
import com.prasad.bikebookingapplication.models.User;
import com.prasad.bikebookingapplication.repositories.BookingRepository;
import com.prasad.bikebookingapplication.repositories.UserRepository;
import com.prasad.bikebookingapplication.services.BookingService;
import com.prasad.bikebookingapplication.services.JwtService;
import com.prasad.bikebookingapplication.services.UserService;
//import com.twilio.base.Resource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.net.MalformedURLException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.Map;
import java.util.Optional;
import java.util.stream.Collectors;

@RestController
```

```
@RequestMapping("/booking")
public class BookingController {
    private final BookingService bookingService;
    private final JwtService jwtService;
    private final UserRepository userRepository;
    private final UserService userService;
    private final BookingRepository bookingRepository;

    @Autowired
    public BookingController(BookingService bookingService, JwtService jwtService,
UserRepository userRepository, UserService userService, BookingRepository
bookingRepository) {
        this.bookingService = bookingService;
        this.jwtService = jwtService;
        this.userRepository = userRepository;//
        this.userService = userService;
        this.bookingRepository = bookingRepository;
    }

@PostMapping("/book")
public ResponseEntity<BookingResponseDto> bookVehicle(@RequestBody
BookingRequestDto request) {
    Booking booking =
bookingService.bookVehicle(bookingService.convertRequestDtoToBooking(request));
    return new ResponseEntity<>(new BookingResponseDto(booking), HttpStatus.OK);
}

@GetMapping("/all")
public ResponseEntity<List<BookingResponseDto>> getAllBookings() {
    List<Booking> bookings = bookingService.getAllBookings();
    List<BookingResponseDto> bookingResponseDtos = bookings.stream()
            .map(BookingResponseDto::new)
            .collect(Collectors.toList());
    return new ResponseEntity<>(bookingResponseDtos, HttpStatus.OK);
}
```

VehicleController.java

```
package com.prasad.bikebookingapplication.controllers;

import com.prasad.bikebookingapplication.constants.CommonConstants;
import com.prasad.bikebookingapplication.dtos.VehicleRequestDto;
import com.prasad.bikebookingapplication.dtos.VehicleResponseDto;
import com.prasad.bikebookingapplication.dtos.VehicleResponseDtoForCustomer;
import com.prasad.bikebookingapplication.models.Vehicle;
import com.prasad.bikebookingapplication.models.VehicleStatus;
import com.prasad.bikebookingapplication.services.VehicleService;
import com.prasad.bikebookingapplication.utils.CommonUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.io.IOException;
import java.time.LocalDateTime;

@RestController
@RequestMapping("/vehicle")
public class VehicleController {
    private final VehicleService vehicleService;

    @Autowired
    public VehicleController(VehicleService vehicleService) {
        this.vehicleService = vehicleService;
    }

    @PostMapping("/add")
    public ResponseEntity<String> addVehicle(@ModelAttribute VehicleRequestDto request)
{
```

```
        try {
            System.out.println("Inside addVehicle");
            System.out.println(request);
            Vehicle vehicle = vehicleService.convertRequestDtoToVehicle(request);
            System.out.println("Converted to vehicle");
            Vehicle savedVehicle = vehicleService.saveVehicle(vehicle);
            return ResponseEntity.status(HttpStatus.CREATED)
                    .body("Vehicle with Number " + savedVehicle.getVehicleRegistrationNumber()
+ " added successfully");
        } catch (IOException e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                    .body("Failed to process vehicle data: " + e.getMessage());
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("An unexpected error occurred: " + e.getMessage());
        }
    }

    @PutMapping("/{id}")
    public ResponseEntity<String> updateVehicle(@PathVariable Long id,
                                    @ModelAttribute VehicleRequestDto request) {
        try {
            Vehicle vehicle = vehicleService.convertRequestDtoToVehicle(request);
            vehicle.setId(id);
            vehicleService.saveVehicle(vehicle);
            return ResponseEntity.status(HttpStatus.OK)
                    .body("Vehicle with id " + id + " updated successfully");
        } catch (IOException e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                    .body("Failed to process vehicle data: " + e.getMessage());
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("An unexpected error occurred: " + e.getMessage());
        }
    }
```

CityController.java

```
package com.prasad.bikebookingapplication.controllers;

import com.prasad.bikebookingapplication.constants.CommonConstants;
import com.prasad.bikebookingapplication.dtos.CityRequestDto;
import com.prasad.bikebookingapplication.dtos.CityResponseDto;
import com.prasad.bikebookingapplication.models.City;
import com.prasad.bikebookingapplication.services.CityService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/city")
public class CityController {
    private final CityService cityService;

    @Autowired
    public CityController(CityService cityService) {
        this.cityService = cityService;
    }

    @PostMapping("/add")
    public ResponseEntity<CityResponseDto> addCity(@RequestBody CityRequestDto
request) {
        City city = cityService.saveCity(request.getName(), request.getState(),
request.getImage());
        CityResponseDto response = new CityResponseDto(city);
        return new ResponseEntity<>(response, HttpStatus.CREATED);
    }

    @PutMapping("/{id}")
    public ResponseEntity<String> updateCity(@PathVariable Long id,
```

```
                                    @RequestBody CityRequestDto request) {
        City city = cityService.convertRequestDtoToCity(request);
        city.setId(id);
        cityService.updateCity(city);
        return ResponseEntity.status(HttpStatus.OK).body("City with id " + id + " updated
successfully");
    }

    @GetMapping("{id}")
    public CityResponseDto getCity(@PathVariable Long id) {
        return new CityResponseDto(cityService.findCityById(id));
    }

    @GetMapping("/all")
    public ResponseEntity<Page<CityResponseDto>> getAllCities(
            @RequestParam(defaultValue = CommonConstants.DEFAULT_PAGE_NUMBER)
Integer page,
            @RequestParam(defaultValue = CommonConstants.DEFAULT_PAGE_SIZE) Integer
size,
            @RequestParam(defaultValue = CommonConstants.DEFAULT_SORT_BY) String
sortBy,
            @RequestParam(defaultValue = CommonConstants.DEFAULT_SORT_DIRECTION)
String sortDirection) {

        Page<CityResponseDto> cities = cityService.findAllCities(page, size, sortBy,
sortDirection);
        return new ResponseEntity<>(cities, HttpStatus.OK);
    }
}
```

CHAPTER 5 - TESTING

5.1 Test Strategy

1. Test Objectives:

- Identify and mitigate defects, errors, and inconsistencies in the application.
- Validate the usability, accessibility, and user experience of the application.
- Verify the security, privacy, and compliance aspects of the application.
- Assess the performance, scalability, and reliability of the application under varying conditions.

2. Test Phases:

- Unit Testing: Individual components and modules of the application are tested in isolation to ensure they function correctly.
- Integration Testing: Integrated components are tested to ensure they interact seamlessly and produce the expected results.
- System Testing: The entire application is tested as a whole to validate end- to-end functionality, usability, and performance.
- Acceptance Testing: The application is tested against user acceptance criteria to ensure it meets user expectations and business requirements.

3. Test Methodologies:

- Manual Testing: Testers manually execute test cases to validate application functionality, usability, and user experience.
- Automated Testing: Automated test scripts are used to perform repetitive and regression testing tasks, improving efficiency and accuracy.
- Exploratory Testing: Testers explore the application dynamically to uncover defects, usability issues, and edge cases not covered by scripted tests.
- Load Testing: The application is subjected to simulated loads to evaluate performance, scalability, and resource utilization under high traffic conditions.

4. Test Coverage:

- Functional Testing: Validate all functional requirements specified for the application, including user registration, appointment booking, service management, and payment processing.
- Non-Functional Testing: Assess non-functional aspects such as usability, security, performance, compatibility, and regulatory compliance.
- Edge Case Testing: Test application behavior in edge cases, including invalid inputs, boundary conditions, and error handling scenarios.
- User Experience Testing: Evaluate the application's usability, accessibility, and user interface design to ensure a positive user experience.
- Security Testing: Conduct security testing to identify vulnerabilities, threats, and risks to the application's data and infrastructure.

5. Test Environment:

- Utilize testing environments that closely resemble the production environment to ensure accurate testing results.
- Use virtualization and containerization technologies to create isolated test environments for testing different components and configurations.

6. Test Tools:

- Test Management Tools: Use tools like Jira, TestRail, or HP ALM for test case management, defect tracking, and reporting.
- Automation Tools: Employ automation tools such as Selenium WebDriver, Appium, or Cypress for automating test scripts and regression testing.
- Performance Testing Tools: Utilize tools like Apache JMeter, LoadRunner, or Gatling for load testing and performance monitoring.

7. Reporting and Documentation:

- Document test plans, test cases, test results, and defects encountered during testing.
- Generate test reports summarizing test coverage, test execution status, and any issues identified during testing.
- Provide stakeholders with regular updates on testing progress, including milestones achieved and risks identified.

8. Test Team Collaboration:

- Foster collaboration and communication among cross-functional teams, including developers, testers, designers, and product owners.
- Conduct regular meetings, reviews, and feedback sessions to ensure alignment on testing objectives, priorities, and outcomes.

9. Continuous Improvement:

- Continuously monitor and evaluate the effectiveness of testing processes, methodologies, and tools.
- Incorporate feedback from testing activities to improve test coverage, efficiency, and effectiveness.
- Identify lessons learned and best practices to inform future testing efforts and enhance overall quality assurance practices.

5.2 Unit Test Plan

1. Objective:

- The objective of unit testing is to validate the functionality, correctness, and reliability of individual units or components of the Caarify application.

2. Scope:

- The unit test plan covers the testing of all functional components, modules, and classes of the Caarify application, including backend APIs, frontend components, and database operations.

3. Testing Approach:

- White-box Testing: Unit tests will be designed based on the internal structure, code logic, and business rules of each component.
- Test-Driven Development (TDD): Follow the TDD approach by writing tests before implementing the code to ensure comprehensive test coverage.

4. Test Environment:

- Utilize development or testing environments that closely resemble the production environment, including necessary dependencies and configurations.

5. Testing Tools:

- Use testing frameworks and libraries compatible with the technology stack of the Caarify application, such as JUnit, Mockito, Jest, or PyTest.

6. Test Cases:

- Develop unit test cases for each functional unit or method, covering both positive and negative scenarios, edge cases, and boundary conditions.
- Test cases should verify input validation, error handling, exception handling, and expected outcomes.

7. Test Coverage:

- Aim for high test coverage to ensure that all critical paths and code branches are tested.
- Target a minimum coverage threshold (e.g., 80%) to maintain code quality and reliability.

8. Test Execution:

- Execute unit tests automatically as part of the continuous integration (CI) pipeline to ensure early detection of defects.
- Run unit tests locally during development and before merging code changes to the main branch.

9. Test Reporting:

- Document test results, including pass/fail status, test coverage metrics, and any defects encountered during testing.
- Generate test reports and share them with the development team for review and analysis.

10. Test Maintenance:

- Update unit tests as needed to accommodate changes in the application code, requirements, or design.
- Refactor and optimize existing unit tests for improved readability, maintainability, and performance.

11. Testing Considerations:

- Concurrency Testing: Test components that handle concurrent requests or operations to ensure thread safety and avoid race conditions.
- Integration Points: Mock or stub external dependencies (e.g., databases, APIs) to isolate units and focus on testing individual functionality.
- Performance Testing: Identify and address performance bottlenecks or inefficiencies at the unit level to optimize application performance.

12. Review and Approval:

- Review the unit test plan with relevant stakeholders, including developers, testers, and project managers, to ensure alignment with project goals and objectives.
- Obtain approval from the project lead or QA manager before proceeding with test execution.

5.3 Acceptance and Test Plan

1. Objective:

- The objective of acceptance testing is to validate that the Caarify Car Service Provider Application meets user requirements, business objectives, and quality standards before deployment.

2. Scope:

- The acceptance test plan covers the testing of end-to-end functionality, usability, and performance of the Caarify application from a user's perspective.

3. Testing Approach:

- Black-box Testing: Acceptance tests will be designed based on the application's external behavior and user interactions, without knowledge of its internal implementation.
- User Scenario Testing: Test real-world user scenarios and workflows to ensure that the application meets user needs and expectations.

4. Test Environment:

- Utilize staging or pre-production environments that closely resemble the production environment, including necessary configurations and datasets.

5. Testing Tools:

- Use testing tools and frameworks suitable for conducting manual acceptance testing, such as TestRail, JIRA, or Microsoft Excel.

6. Test Cases:

- Develop acceptance test cases based on user stories, use cases, and functional requirements specified for the Caarify application.
- Test cases should cover all critical features, workflows, and user interactions of the application.

7. Test Scenarios:

User Registration and Login:

- Verify that users can register for an account and log in successfully using valid credentials.
- Validate the functionality of password recovery and account activation processes.

Appointment Booking:

- Test the process of scheduling service appointments, including selecting services, mechanics, dates, and times.
- Verify that users receive confirmation notifications and appointment reminders.

Service Management:

- Ensure administrators can manage mechanics, services, and user accounts effectively through the admin dashboard.
- Validate the functionality of adding, editing, and deleting mechanics and services.

Payment Processing:

- Test the payment process for booking service appointments, including selecting payment methods, entering payment details, and completing transactions.

- Verify that users receive payment confirmation and receipts for successful transactions.

Emergency Assistance:

- Validate the functionality of emergency location mapping and communication tools for users in critical situations.
- Test the process of sharing location details with emergency responders and service providers.

8. Test Data:

- Prepare test data sets that simulate real-world scenarios, including sample user accounts, service appointments, and administrative configurations.

9. Test Execution:

- Execute acceptance tests manually by following predefined test cases and scenarios.
- Record test results, including pass/fail status, observations, and any defects encountered during testing.

10. Defect Reporting:

- Document and report any defects or issues encountered during acceptance testing using a standardized defect tracking system.
- Include detailed descriptions, screenshots, and steps to reproduce each reported defect.

11. Regression Testing:

- Perform regression testing to ensure that fixes for reported defects do not introduce new issues or regressions.
- Re-run previously executed acceptance tests to validate the stability and reliability of the application.

12. Test Sign-off:

- Obtain sign-off from stakeholders, including product owners, project managers, and QA leads, to confirm the completion of acceptance testing and readiness for deployment

5.4 Test Case:

Test Case Id	Test Case Name	Purpose	Steps	Expected Result	Actual result	Pass /Fail
T1	Register New Account	Registration	1. Open Web app 2. Go to Register 3. Enter valid email, password 4. Submit	User account is created and confirmation is shown	Valid	Pass
T2	Log In with Valid Credentials	Log In with register credentials	1. Go to Login 2. Enter correct email, password 3. Submit	User is logged in and redirected to the home page	Valid	Pass
T3	Log In with Valid Credentials	Log In with register credentials	1. Go to Login 2. Enter correct email, password 3. Submit	User is logged in Failed	Invalid	Fail
T4	Log In with Invalid Credentials	Web App accessible	1. Go to Login 2. Enter incorrect email or password 3. Submit	Error message "Invalid credentials" is displayed	Invalid	Pass
T5	Browse Vehicle by Category	Web App accessible, products loaded	1. Open Web app 2. Navigate to "Cities" 3. Select a City	Vehicles list displays Vehicles within selected type	Valid	Pass

T6	Search for Vehicle by Keyword	Web App accessible, products available	1. Open search bar 2. Enter search term (e.g., "Accessories") 3. Submit	Relevant Vehicle are displayed in search results	Valid	Pass
T7	Search for Vehicle by Keyword	Web App accessible, products available	1. Open search bar 2. Enter search term (e.g., "Items") 3. Submit	Relevant Vehicles are not displayed in search results	Invalid	Fail
T8	View Vehicle Details	Vehicle available	1. Select a Vehicle 2. View Vehicle details page	Vehicle page shows name, price, description, images, and options	Valid	Pass
T9	View Vehicle User and Guest	Vehicle available	1. Open Vehicles page 2. Tap "Proceed to Checkout" button	Vehicle is viewed by User and Guest	Valid	Pass
T10	Add Vehicle to Checkout	Vehicle available	1. Open Vehicles page 2. Tap "Add to Checkout" button	Vehicle is added to checkout with specified quantity	Valid	Pass
T11	Add Vehicle to Checkout	Vehicle Out of Stock	1. Open Vehicles page 2. Tap "Add to Checkout" button	Vehicle is not added to checkout	Invalid	Fail

T12	View and Edit Checkout	Item added to Checkout	1. Open checkout 2. Adjust quantity/remove item 3. Confirm changes	Checkout is updated with specified changes	Valid	Pass
T13	Successful Payment Processing	Item in checkout, valid payment details	1. Open checkout 2. Proceed to checkout 3. Enter valid payment details 4. Complete purchase	Payment is processed successfully, order confirmation page displays order details, and confirmation email is sent	Valid	Pass
T14	Payment Failure Handling	Item in checkout, incorrect payment details	1. Open checkout 2. Proceed to checkout 3. Enter invalid payment details (e.g., expired card) 4. Complete purchase	Error message "Payment failed. Please check your payment details" is displayed, and order is not placed	Invalid	Fail

T15	Retry Payment After Failure	Payment initially failed	1. Attempt purchase with invalid payment 2. Update payment details with valid card 3. Retry checkout	Payment is successful, and order confirmation is shown along with confirmation email received	Invalid	Fail
T16	View Order History	Logged in, Vehicle order placed	1. Open profile 2. Go to "Order History"	Order history displays past orders with details	Valid	Pass

CHAPTER 6 : LIMITATIONS OF PROPOSED SYSTEM

Limitations of Proposed System

1.5 Limitations of the Proposed System

While the Okbikes Rental System aims to deliver a feature-rich and user-centric platform for two-wheeler rentals, there are certain limitations in its current scope and implementation. These limitations may arise from technical, operational, or external constraints and should be considered for future improvements.

1. Dependency on Internet and GPS Connectivity

- Real-time vehicle tracking and booking processes depend heavily on uninterrupted internet and GPS connectivity.
- Users in remote or low-network areas may face challenges in accessing services or tracking vehicles accurately.

2. Limited Geographic Coverage (Initial Phase)

- During the early stages of deployment, Okbikes services may only be available in selected cities or regions.
- Expansion to rural or less urbanized areas may take additional time due to logistical and regulatory challenges.

3. Hardware Dependency for Tracking

- The tracking feature relies on GPS/IoT hardware installed in vehicles.
- Any hardware malfunction or tampering may lead to inaccurate tracking or complete failure of location services.

4. Vehicle Availability During Peak Hours

- Despite dynamic booking and inventory management, there may be times when high demand results in a shortage of available vehicles, especially during festivals, weekends, or holidays.

5. Limited Support for Offline Users

- The system is designed primarily for users with smartphones and access to mobile apps or websites.
- Users without smartphones or digital literacy may not be able to utilize the platform effectively.

6. Maintenance and Downtime

- Like all digital platforms, Okbikes may require occasional system updates, bug fixes, or maintenance which can cause temporary downtime or reduced functionality.

7. Dependency on Third-Party Services

- Features such as payment gateways, SMS/OTP delivery, map services (e.g., Google Maps), and insurance integration rely on third-party APIs and services.
- Any outage or limitation in these services can affect the overall user experience and system functionality.

8. Security Risks and Data Privacy Concerns

- Although the system includes secure login and encrypted transactions, it is still exposed to common cybersecurity threats (e.g., phishing, data breaches, and fraud) if not continuously monitored and updated.
- Adherence to data protection regulations (such as GDPR or regional laws) must be regularly enforced to maintain user trust.

9. Limited AI/ML Integration (Current Phase)

- In the initial version, Okbikes may not include AI-powered features such as predictive maintenance, demand forecasting, or personalized recommendations, though these can be part of future updates.

10. Weather Dependency

- Two-wheeler rentals are significantly affected by adverse weather conditions like heavy rain, snow, or extreme temperatures.
- The system lacks weather-adaptive pricing or reservation policies that account for seasonal variations.

11. Limited Battery Range for Electric Options

- If electric two-wheelers are part of the fleet, their limited battery range may restrict longer journeys.
- Insufficient charging infrastructure in many areas could lead to range anxiety among users.

12. Regulatory Compliance Variations

- Different regions have varying regulations for vehicle rentals, license requirements, and insurance policies.
- Adapting the system to comply with all regional differences simultaneously presents operational challenges.

13. Identity Verification Limitations

- Remote identity verification processes may not be as secure as in-person verification.
- Potential for identity fraud or misuse of the platform if verification systems are circumvented.

14. Limited Integration with Public Transport

- The current system may lack seamless integration with public transportation networks for true multi-modal mobility.
- This limits the "first mile/last mile" transportation solution potential of the service.

15. Helmet and Safety Equipment Management

- Ensuring the availability, cleanliness, and proper sizing of helmets and other safety equipment across all pickup points.
- Challenges in monitoring whether users actually use provided safety equipment.

16. Limited Fleet Diversity

- Initial vehicle offerings may be restricted to certain models or types.
- Special requirements like disability-friendly options, cargo capacity, or child seats may not be available.

17. Dependency on Stable Fuel/Energy Prices

- Rental pricing models can be significantly affected by volatile fuel or electricity prices.
- May lack dynamic pricing systems that can quickly adapt to energy market fluctuations.

18. User Behavior Monitoring Challenges

- Difficulty in monitoring reckless driving, traffic violations, or misuse of vehicles.
- Limited recourse if users violate traffic rules while using rented vehicles.

19. Cross-Platform Compatibility Issues

- The app or website may not function optimally across all device types, operating systems, or browser versions.
- Older devices may experience performance issues or incompatibility with certain features.

20. Language and Accessibility Barriers

- Limited support for multiple languages could restrict usability in diverse regions.
- The interface may lack comprehensive accessibility features for users with disabilities.

CHAPTER 7 : PROPOSED ENHANCEMENTS

Proposed Enhancements

To ensure continuous improvement and long-term success, the Okbikes Rental System can incorporate the following enhancements in future phases. These enhancements are designed to address current limitations, meet evolving user expectations, and leverage technological advancements.

1. Offline Booking Support

- Introduce kiosk-based or call-center booking options for users without smartphones or internet access.
- Enable agents or franchise partners to assist users in booking and returning vehicles manually.

2. Expansion of Service Area

- Gradually scale the platform to cover more cities, towns, and rural areas.
- Partner with local dealerships or franchise operators to expand reach and operational support.

3. Predictive Analytics and AI Integration

- Use AI to forecast demand patterns based on location, weather, time, and past usage.
- Implement predictive maintenance algorithms to detect potential vehicle issues before failure.
- Recommend personalized rental packages based on user behavior and history.

4. Enhanced Security and Fraud Prevention

- Implement biometric authentication (e.g., fingerprint or face recognition) for added user security.
- Use AI-based anomaly detection to prevent fraudulent bookings or misuse of accounts.
- Periodic security audits and compliance with updated data protection regulations (e.g., GDPR, CCPA).

5. Advanced Fleet Management

- Integrate telematics to monitor vehicle health (engine status, tire pressure, fuel level).
- Automate scheduling of maintenance and service tasks based on usage metrics.
- Track battery health and charging needs for electric vehicles.

6. In-App Chatbot and Multilingual Support

- Deploy an AI-powered chatbot for 24/7 customer assistance on common queries.
- Include support for multiple languages to improve accessibility for diverse user groups.

7. Green Initiatives and Carbon Tracking

- Encourage users to choose electric vehicles by offering incentives or rewards.
- Introduce carbon tracking to show users the environmental impact of their rides.
- Partner with eco-friendly organizations for CSR initiatives and awareness campaigns.

8. Ride-Sharing and Multi-Vehicle Booking

- Enable users to share rides with friends or coworkers through in-app invite features.
- Allow users to book multiple vehicles in one transaction (useful for group tours or corporate bookings).

9. Integration with Public Transport Systems

- Collaborate with city transport systems to provide last-mile connectivity options.
- Offer bundled passes that combine scooter rentals with bus/metro tickets.

10. Advanced Admin and Analytics Dashboard

- Provide real-time analytics on vehicle usage, revenue trends, peak demand times, and customer feedback.
- Add forecasting tools to help management plan fleet distribution and promotional campaigns.

11. Enhanced Vehicle Customization Options

- Allow users to pre-configure their rental vehicles with add-ons like phone mounts, extra storage, or rain gear.
- Implement "vehicle profiles" where users can save their preferred settings for quick booking.
- Offer premium vehicle modifications for long-term rentals (ergonomic seats, luggage carriers).

12. Blockchain-Based Vehicle History and Authentication

- Implement blockchain technology to create immutable records of vehicle maintenance history.
- Enable transparent tracking of vehicle usage, accidents, and repairs for user confidence.
- Use smart contracts for automated deposit returns and damage assessments.

13. Augmented Reality (AR) Features

- Develop AR navigation that projects directions onto the rider's helmet visor or smartphone.
- Create virtual vehicle tours allowing users to inspect vehicles before booking.
- Implement AR-based troubleshooting guides for common issues riders might encounter.

14. Dynamic Weather-Responsive System

- Integrate real-time weather forecasting to alert users about unsafe riding conditions.
- Implement automatic rebooking options during extreme weather events.
- Offer weather-appropriate gear recommendations or rentals based on forecasts.

15. Advanced Loyalty and Gamification Program

- Create a tiered membership program with increasing benefits for frequent users.
- Implement gamification elements like badges for eco-friendly riding, safe driving, or exploring new areas.
- Establish a points system redeemable for free rides, upgrades, or partner services.

16. Community Features and Social Integration

- Build an in-app community for riders to share routes, experiences, and tips.
- Enable group ride planning and coordination features.
- Implement ride challenges and leaderboards to encourage engagement.

17. Comprehensive Insurance and Safety Enhancements

- Offer tiered insurance packages with varying coverage levels.
- Implement in-app safety training modules and quizzes for new users.
- Provide real-time safety alerts for hazardous road conditions or accident-prone areas.

18. Advanced Payment Flexibility

- Implement subscription-based rental models for regular users.
- Support cryptocurrency payments for greater privacy and reduced transaction fees.
- Offer installment payment options for long-term rentals or premium vehicles.

19. IoT-Enhanced Vehicle Features

- Deploy smart helmets with built-in communication systems and heads-up displays.
- Implement keyless ignition through smartphone proximity or biometric verification.
- Add theft prevention features like remote immobilization and geofence alerts.

20. Accessibility Enhancements

- Develop specialized vehicles for users with physical disabilities.
- Implement voice-command features for hands-free operation of the app while riding.
- Create simplified interfaces for elderly users or those with limited technological proficiency.

CHAPTER 8 : CONCLUSION

Conclusion

The Okbikes Rental System presents a modern, efficient, and scalable solution for two-wheeler rentals, addressing many of the shortcomings found in existing platforms. By focusing on user convenience, flexible rental packages, and secure transaction handling, the system aims to provide a seamless and reliable experience for both users and administrators. While the initial version lays a solid foundation, the system also acknowledges its current limitations and outlines a clear roadmap for future enhancements.

With continued development and integration of advanced technologies such as AI, telematics, and eco-friendly initiatives, Okbikes has the potential to become a leading platform in the urban mobility sector. The system is not only designed to meet the present demands of commuters but also to adapt and grow in response to changing market trends and user expectations. As urban transportation evolves, Okbikes is well-positioned to offer innovative and sustainable mobility solutions for the future.

CHAPTER 9 : BIBLIOGRAPHY

Bibliography

1. Baeldung. (n.d.).Spring Boot Tutorial.
Retrieved April 30, 2025, from https://www.baeldung.com/spring-boot
Used for building RESTful backend services and implementing secure authentication using Spring Boot.

2. MySQL Documentation. (n.d.).MySQL 8.0 Reference Manual. Oracle.
Retrieved April 30, 2025, from https://dev.mysql.com/doc/refman/8.0/en/
Used to design and manage the relational database structure for storing rental and user data.

3. GeeksforGeeks. (n.d.).ReactJS – Introduction.
Retrieved April 30, 2025, from https://www.geeksforgeeks.org/reactjs-introduction/
Served as a reference for learning ReactJS fundamentals and implementing component-based frontend logic.

4. ReactJS.org. (n.d.).React – A JavaScript library for building user interfaces. Meta.
Retrieved April 30, 2025, from https://reactjs.org/
Official documentation used for frontend architecture, hooks, state management, and component lifecycles.

5. Vite. (n.d.).Vite: Next Generation Frontend Tooling.
Retrieved April 30, 2025, from https://vitejs.dev/
Used to accelerate frontend development and manage module bundling in the React project.

6. Tailwind Labs. (n.d.).Tailwind CSS – Rapidly Build Modern Websites.
Retrieved April 30, 2025, from https://tailwindcss.com/
Used to design a responsive, modern UI using utility-first CSS classes.

7. EmailJS. (n.d.).EmailJS – Send emails directly from your client-side JC.
Retrieved April 30, 2025, from https://www.emailjs.com/
Integrated for sending emails like OTPs and contact forms directly from the frontend without a backend mail server.

8. React Native. (n.d.).React Native – Create native apps for Android and iOS
Retrieved April 30, 2025, from https://reactnative.dev/
Considered for future expansion to mobile platforms, enabling cross-platform Android/iOS development.

9. Brave Software, Inc. (n.d.).Brave Browser: Secure, Fast & Private Web Browser
Retrieved April 30, 2025, from https://brave.com/
Used during browser compatibility testing and privacy assurance for the Okbikes web application.

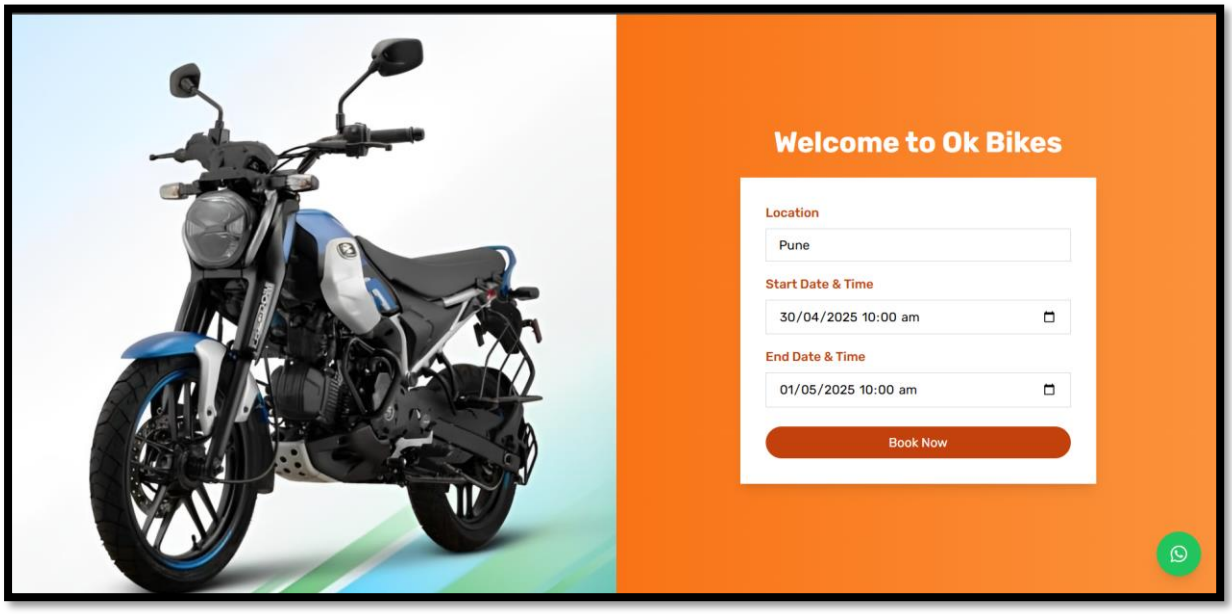
CHAPTER 10 : USER MANUAL

User Manual (All Screens With Proper description)

Input screen name: Home page

Description: Here Customer and admin seeing user interface of website

1. The Home Page serves as the main entry point for both customers and administrators of the bike rental system.
2. It provides an intuitive interface showcasing available bikes, categories, and rental options.
3. A prominent search bar allows users to filter bikes by location, type, and rental duration.
4. Customers can quickly navigate to features like “Book Now”, “Login/Signup”, and “Explore Bikes”.
5. Promotional banners and featured bikes are displayed to attract user attention.
6. An informative section explains the rental process in simple steps: Search → Book → Ride.
7. Testimonials and user reviews help build trust and credibility for the platform.
8. Admin users may see quick links to dashboard metrics, booking alerts, and system updates.
9. The navigation bar and footer offer access to pages like About Us, Contact, and Pricing.
10. The page is designed to be fully responsive, offering a smooth experience across devices.



Input screen name: Bike Listing Home page

Description:

1. The Bike Listing Home Page displays a comprehensive list of all available bikes for rent.

2. Users can view key details such as bike model, brand, price per hour/day, and availability status.

3. Each bike entry includes an image, brief specifications, and a “Book Now” button for easy access.

4. Advanced filtering options are provided to help users refine their search.

5. Filters include vehicle type (e.g., scooter, sports bike), vehicle brand (e.g., Honda, Yamaha), fuel type (petrol/electric), and rental location.

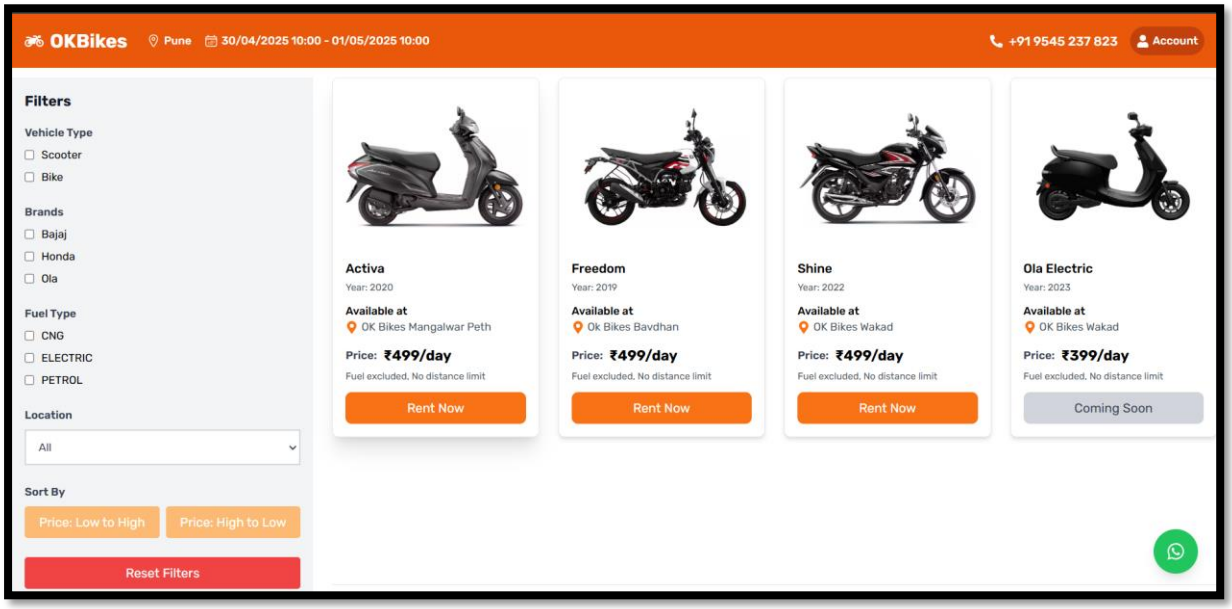
6. The list updates dynamically based on the selected filters for a more personalized experience.

7. Sorting options allow users to organize bikes by price, popularity, or newest arrivals.

8. A search bar is also available for quickly finding a specific bike model or brand.

9. Each listing is clickable and leads to a detailed view with full specifications and booking options.

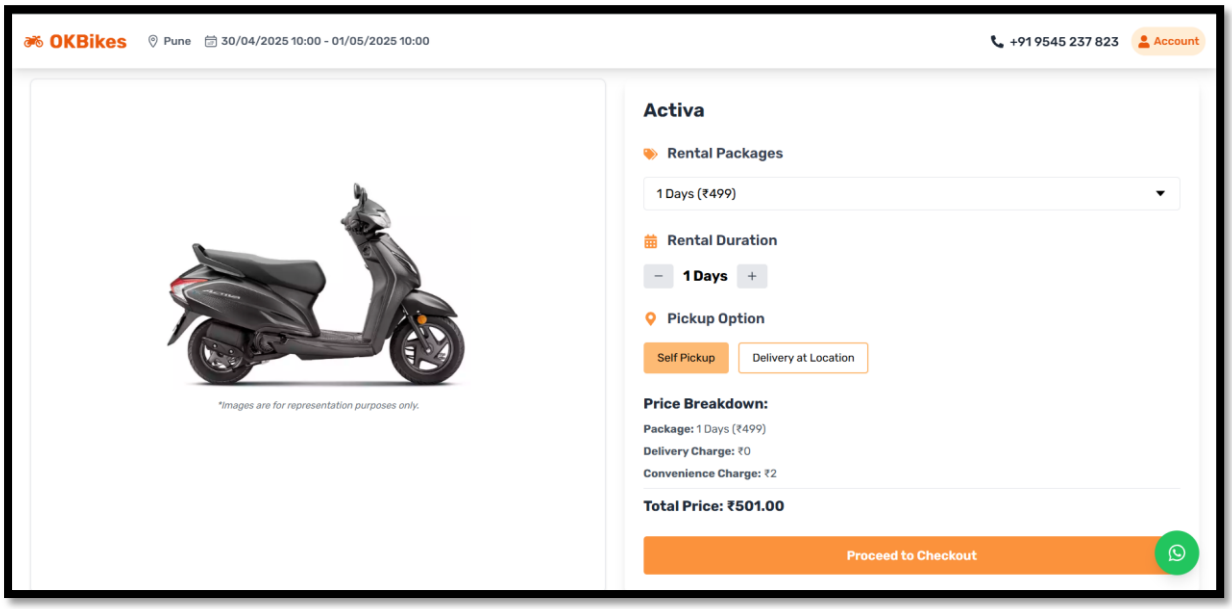
10. The layout is clean and responsive, ensuring a smooth browsing experience on both mobile and desktop devices.



Input screen name: Selected Bike Page .

Description:

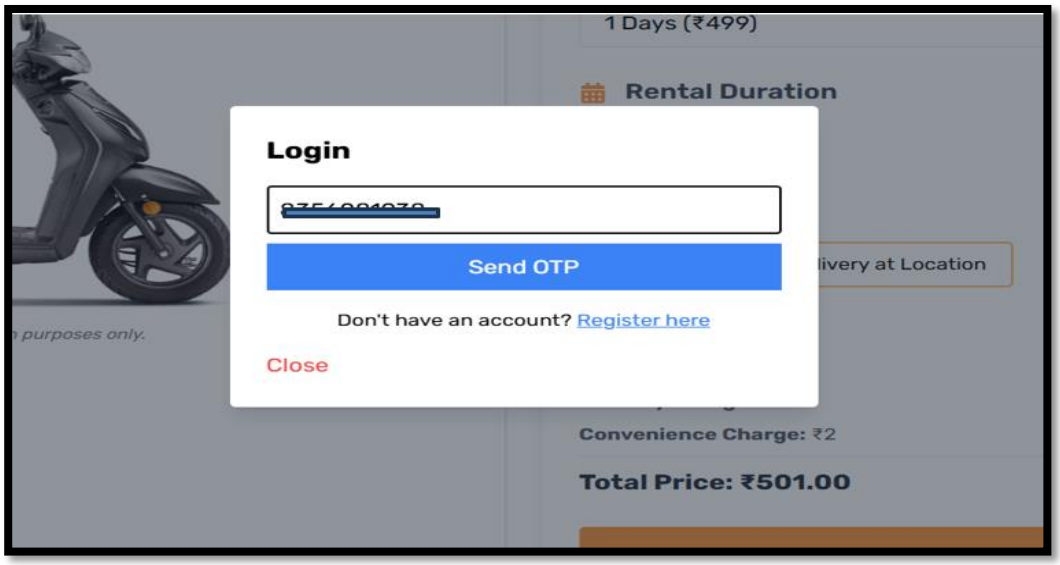
1. The Selected Bike Page displays detailed information about the bike chosen by the customer.
2. It includes specifications such as model name, brand, engine capacity, fuel type, and rental price.
3. High-quality images and customer ratings are shown to help users make informed decisions.
4. Rental package options are clearly listed, including hourly, daily, and weekly plans.
5. Users can select the desired rental duration using an interactive date and time picker.
6. Pickup and return location options are provided for customer convenience.
7. A dynamic pricing section calculates the total bill based on selected package and duration.
8. Additional charges, offers, and discounts (if any) are transparently displayed.
9. Users can proceed to the next step with a "Continue to Booking" or "Book Now" button.
10. The page is designed to be clean, user-friendly, and responsive across all device sizes.

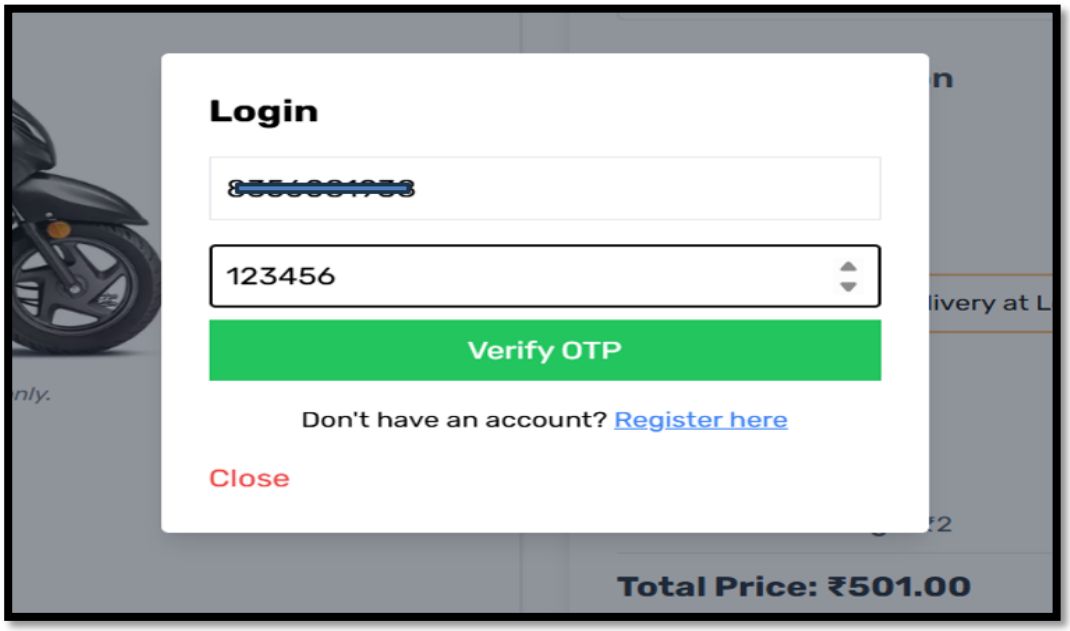


Input screen name: Login Page .

Description:

1. The Login Page prompts users to log in before proceeding to checkout or booking a bike.
2. Users are required to enter their registered mobile number to begin the login process.
3. An OTP (One-Time Password) is sent to the entered number for secure validation.
4. The user must enter the OTP within a time limit to complete the login process.
5. If the number is not registered, the user is redirected to the Sign-Up page.
6. A “Resend OTP” option is available in case the OTP is not received or expires.
7. All inputs are validated in real-time to ensure correct and secure data entry.
8. Upon successful login, the user is redirected back to the checkout or selected bike page.
9. Minimalistic design ensures quick access, with smooth animations for a better experience.
10. The page is responsive and optimized for both mobile and desktop use.

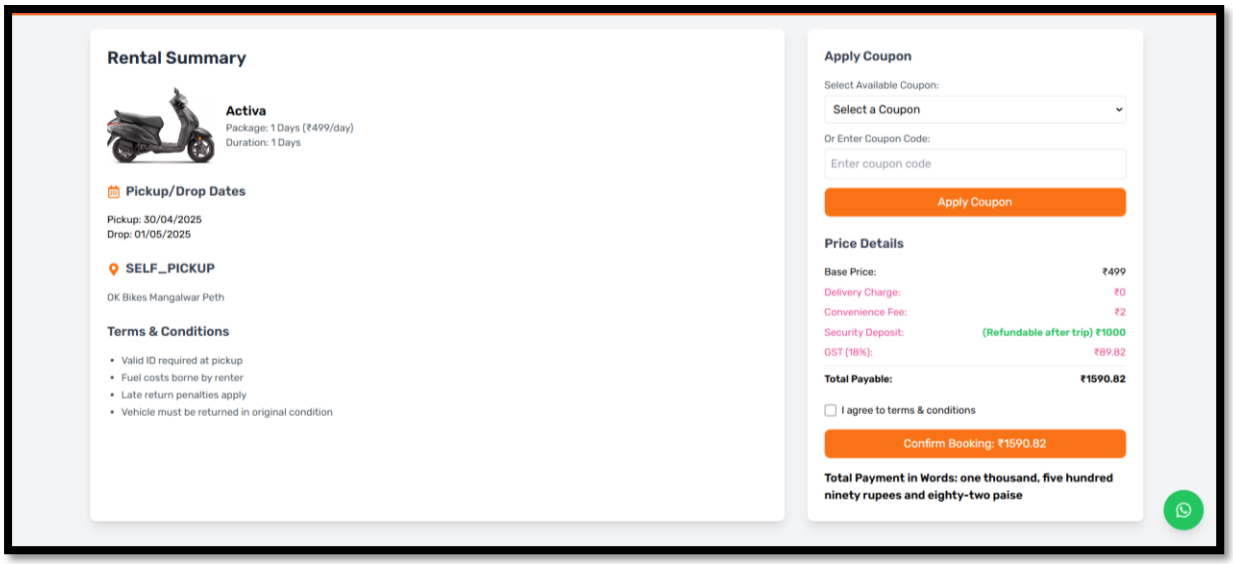




Input screen name: Checkout Page

Description: Description:

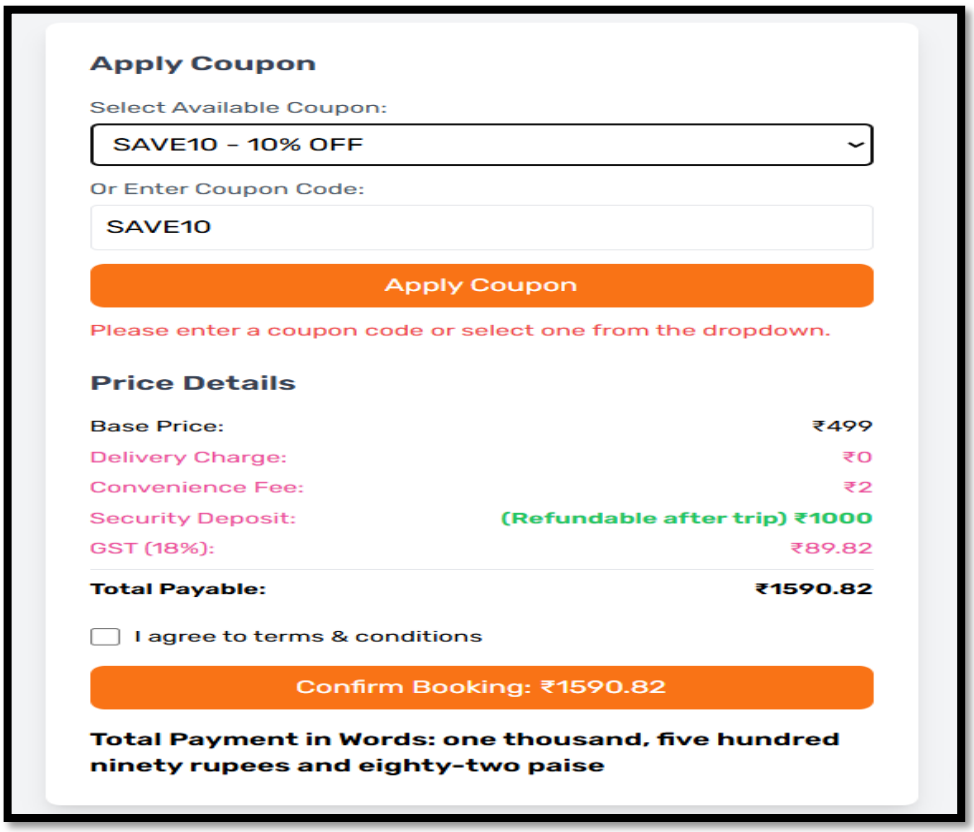
1. The Checkout Page displays a complete summary of the bike rental details selected by the customer.
2. It includes information such as bike name, rental duration, pickup and return locations, and package selected.
3. Users can review pricing details including base fare, taxes, discounts, and the total payable amount.
4. Editable fields allow customers to make last-minute changes to rental time or location before confirming.
5. Contact details like name and phone number are auto-filled from the user profile for convenience.
6. A billing breakdown helps users understand each cost component clearly.
7. Payment method selection is provided, including options like UPI, credit/debit card, or cash on pickup.
8. A checkbox is included to confirm agreement to terms and conditions before proceeding.
9. The "Confirm & Pay" button finalizes the booking and redirects to the payment gateway.
10. The page is secure, responsive, and designed for a smooth final step in the rental process.



Input Screen name: Coupons Page & Bill .

Description:

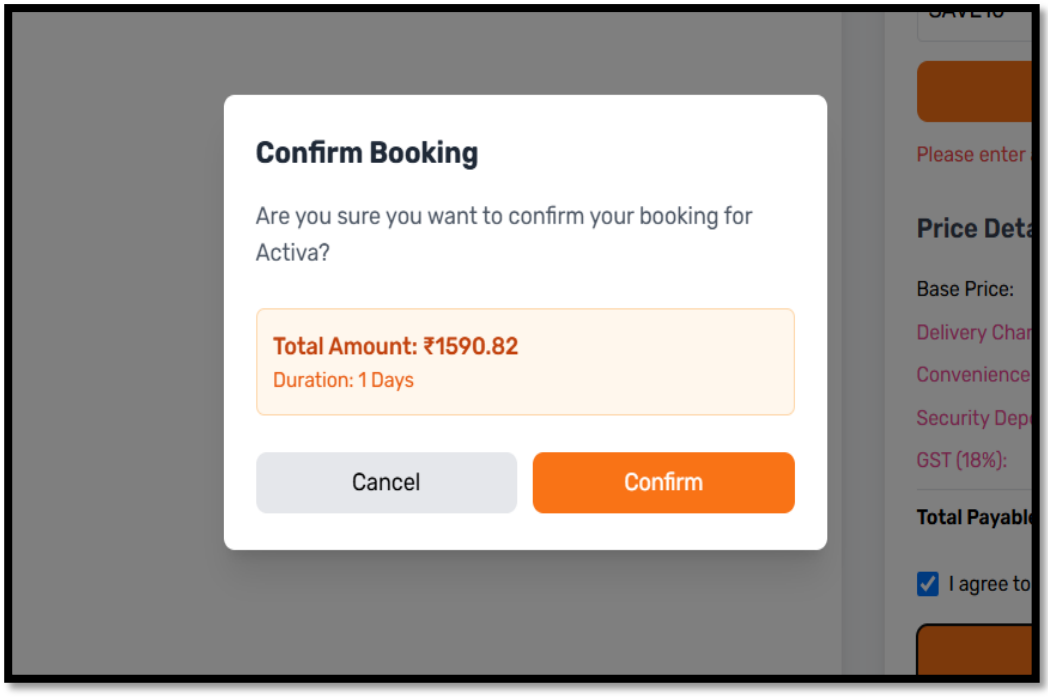
1. The Coupons Page allows customers to apply valid discount coupons based on their selected rental package.
2. Available coupons are listed with terms, expiry dates, and applicable conditions clearly mentioned.
3. Customers can enter a coupon code manually or select from suggested offers.
4. The system instantly validates the coupon and updates the bill if applicable.
5. A success or error message is shown based on coupon eligibility and usage status.
6. The page displays a detailed rental bill including base fare, taxes, discount amount, and final total.
7. Users can review the updated total payable amount before proceeding to payment.
8. All charges are broken down clearly to maintain billing transparency.
9. A “Proceed to Pay” button is provided to take users to the secure payment gateway.
10. The page is responsive and ensures a smooth and error-free checkout experience.



Input Screen name: Confirm Booking

Description:

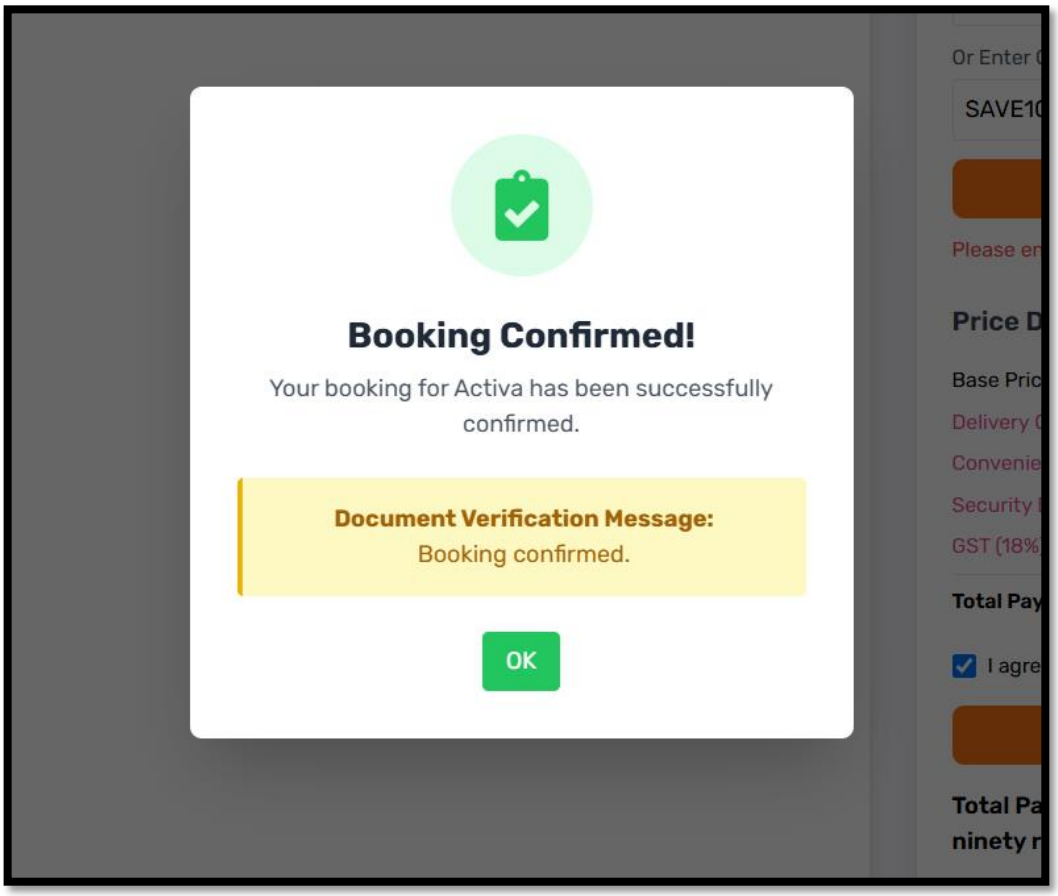
1. The Confirm Booking page acts as the final review step before the user proceeds to payment.
2. It displays all selected rental details including bike name, rental duration, pickup & return locations, and selected package.
3. A summary of the applied coupon (if any) and total billing amount is shown clearly.
4. User contact details and booking time are confirmed for accuracy.
5. A checkbox is provided to agree with the terms & conditions and rental policy.
6. Users can go back to edit rental details if needed before finalizing.
7. A unique booking reference number may be temporarily generated for tracking.
8. Visual confirmation cues like icons or green check marks improve user confidence.
9. A "Confirm & Proceed to Payment" button is highlighted at the bottom.
10. The page ensures all critical information is double-checked to avoid booking errors.



Input Screen name: Confirm Booked

Description:

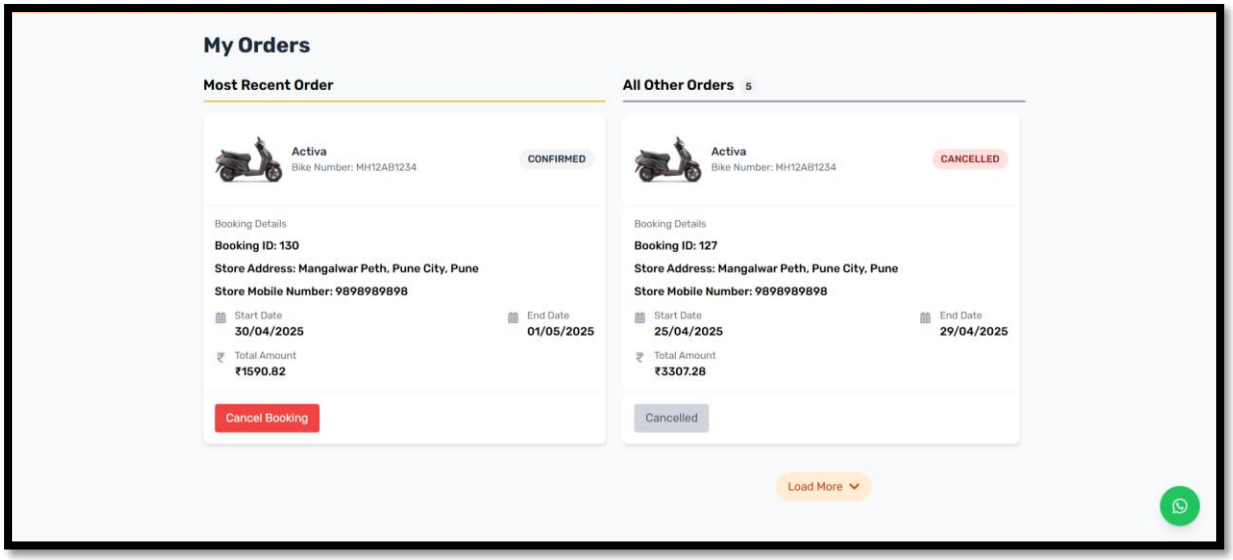
1. The Confirm Booked screen is a popup that appears after successful booking and payment completion.
2. It displays a confirmation message stating that the bike has been successfully booked.
3. A unique Booking ID or reference number is shown for future reference and tracking.
4. Key booking details like bike name, rental dates, pickup location, and total amount paid are summarized.
5. A success icon or animation (like a green check mark) enhances user satisfaction.
6. Options are provided to Download Invoice or View Booking Details.
7. A “Go to Home” or “My Bookings” button allows users to continue browsing or manage bookings.
8. The popup ensures that no accidental clicks bypass the confirmation visibility.
9. All data shown is retrieved in real-time to reflect accurate booking status.
10. The popup is designed to be clean, responsive, and compatible with all device sizes.



Input Screen name: My Orders Page

Description : Displays a list of previous orders with order IDs and dates.

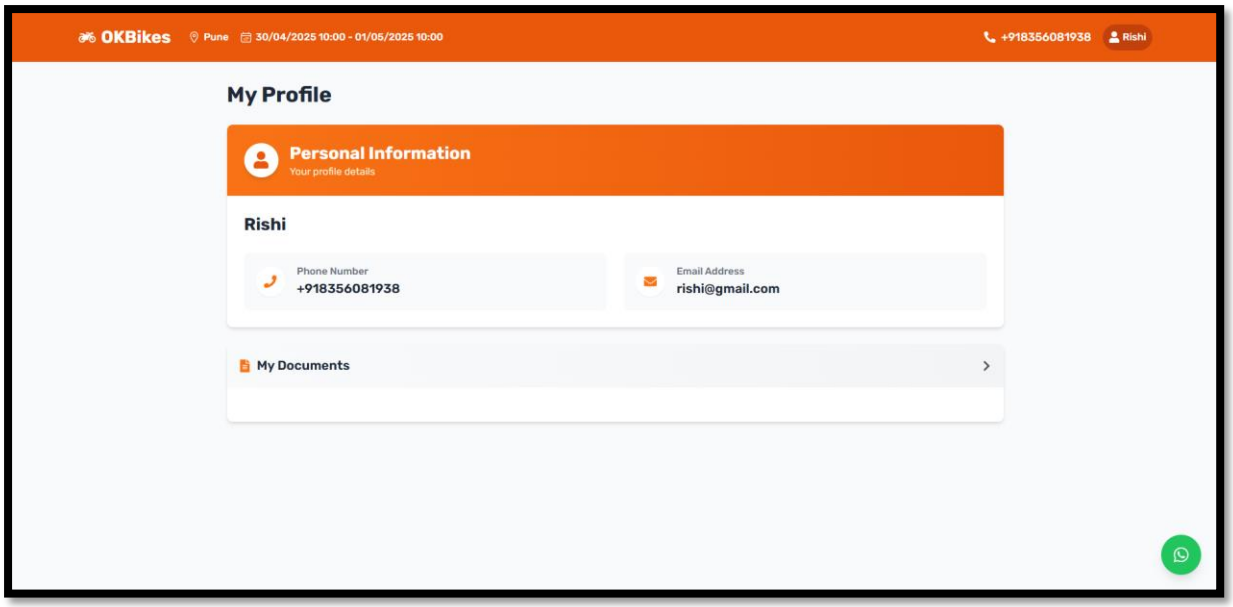
1. Shows detailed information about each order (products, quantities, prices).
2. Allows users to filter orders by date range or order status.
3. Users can search orders by order ID or product name.
4. Displays current status of each order (processing, shipped, delivered, etc.).
5. Includes an option to cancel eligible bookings (if cancellation window is available).
6. Provides a "Track Order" button to track the delivery status.
7. Allows users to view detailed order history with full breakdowns of payments.
8. Displays estimated delivery date for each order.
9. Includes customer support contact details for issues with orders.



Input Screen name: My Profile Page

Description: Displays user's profile picture and allows for easy upload or change.

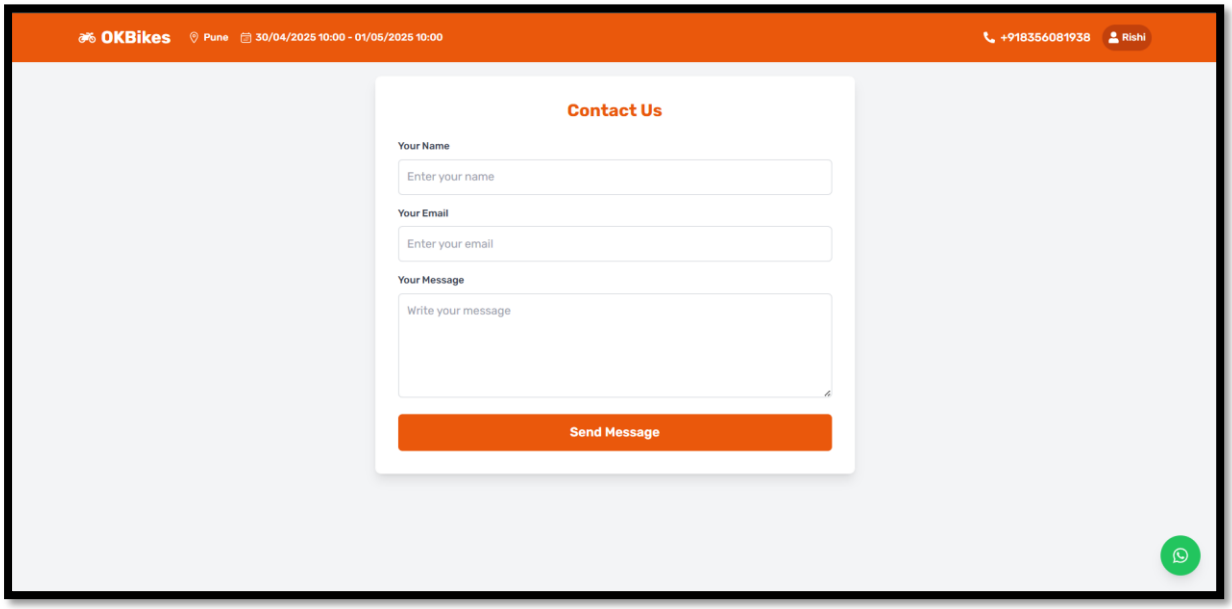
1. Shows basic user information (name, email, phone number).
2. Provides an option to edit personal details (name, contact information, etc.).
3. Allows users to update their password or change security settings.
4. Displays past activity or purchase history for reference.
5. Includes a "Save Changes" button to apply updates to profile.
6. Provides an option to link or unlink social media accounts (if applicable).
7. Shows account settings, including notification preferences.
8. Includes an option to deactivate or delete the account.
9. Displays login history for security purposes.



Input Screen name: Contact US Page

Description:

1. Displays company contact information (phone number, email, address).
2. Includes a contact form for users to submit inquiries or feedback.
3. Form includes fields for name, email, subject, and message.
4. Provides an option to attach files (if necessary for the inquiry).
5. Includes a "Send Message" button to submit the contact form.
6. Displays customer support working hours.
7. Provides a map to the company's physical location (if relevant).
8. Shows frequently asked questions (FAQ) for quick assistance.
9. Includes links to social media platforms for additional contact methods.
10. Displays a confirmation message once the form is submitted successfully.

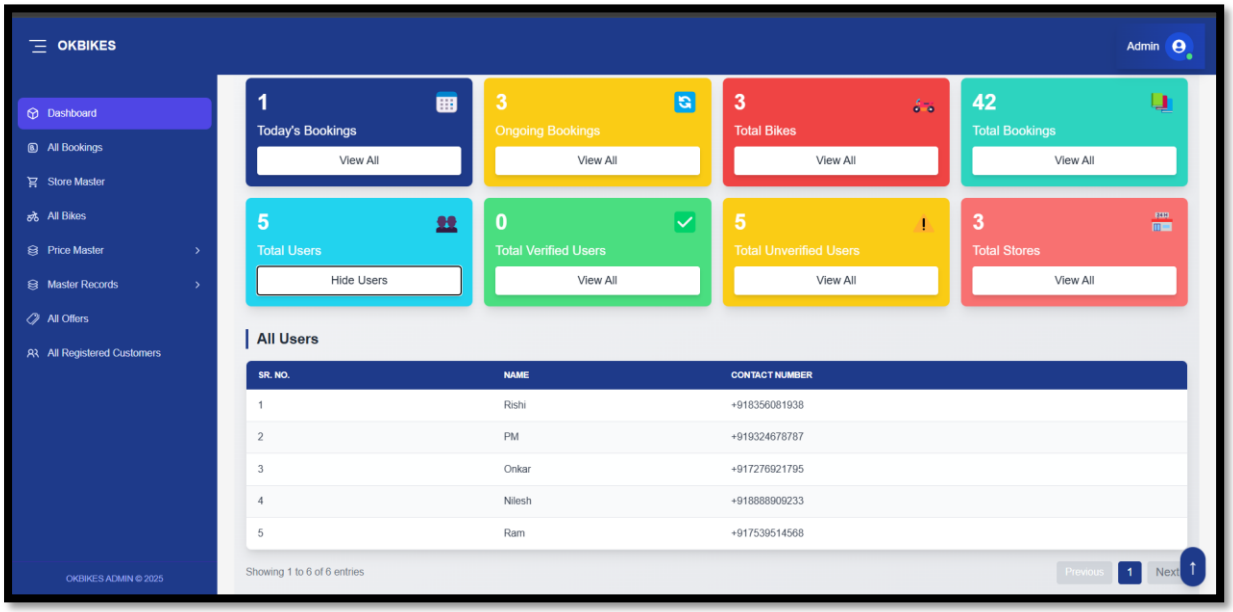


Output Forms with Data

Output screen name: all Users

Description: Displays a list of all registered users with their usernames or full names.

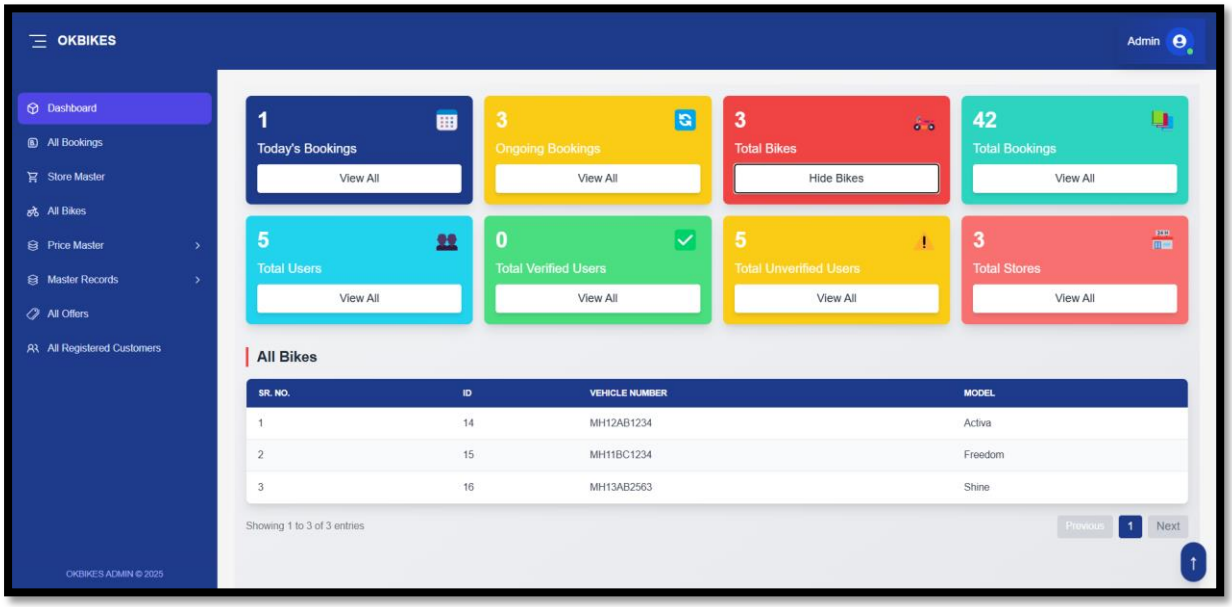
1. Shows key details for each user (email, phone number, registration date).
2. Includes an option to search and filter users by name, email, or registration date.
3. Displays user status (active, suspended, banned, etc.).
4. Provides an option to view detailed profile information for each user.
5. Includes buttons to edit user details or reset passwords (admin privileges only).
6. Shows user activity status or last login time for each user.
7. Allows admin to perform actions like ban or suspend a user account.
8. Displays total number of users in the system.
9. Provides pagination for browsing large lists of users.



Output screen name: all Bikes

Description: Displays a list of all available bikes with their names and model numbers.

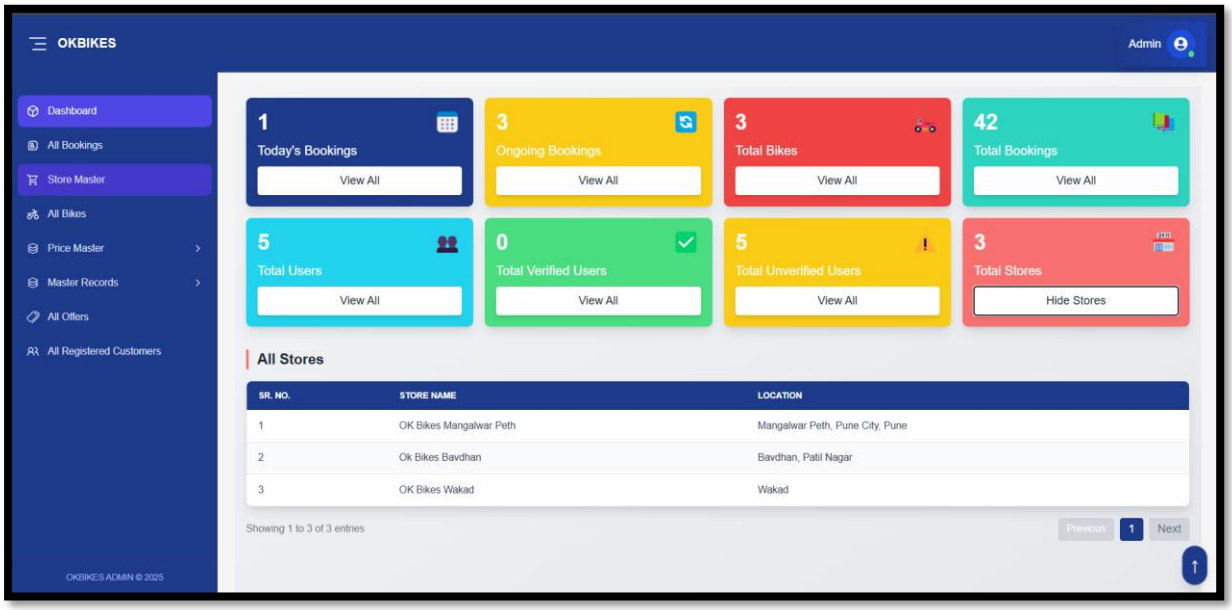
1. Shows key details for each bike (price, color, engine type, availability status).
2. Includes an option to search and filter bikes by name, model, price, or availability.
3. Displays bike status (available, out of stock, under maintenance, etc.).
4. Provides an option to view detailed specifications for each bike.
5. Allows admin to edit bike details (price, specifications, etc.).
6. Includes options to add new bikes or remove outdated ones from the list.
7. Shows total number of bikes in the system.
8. Displays any promotions or discounts applicable to each bike.
9. Provides pagination for browsing large lists of bikes.



Output screen name: all Stores

Description: Displays a list of all stores with their names and locations.

1. Shows key details for each store (contact information, working hours, store manager).
2. Includes an option to search and filter stores by name, location, or status.
3. Displays store status (open, closed, under maintenance, etc.).
4. Provides an option to view detailed store information, including inventory.
5. Allows admin to edit store details (contact info, working hours, etc.).
6. Includes options to add new stores or remove outdated ones from the list.
7. Shows total number of stores in the system.
8. Displays sales or performance data for each store (if applicable).
9. Provides pagination for browsing large lists of stores.



Output screen name: all Bookings

Description: Displays a list of all bookings with booking IDs and customer names.

1. Shows key details for each booking (date, time, status, payment status).
2. Includes an option to search and filter bookings by customer name, booking date, or status.
3. Displays booking status (confirmed, canceled, pending, etc.).
4. Provides an option to view detailed information for each booking (customer details, service booked, etc.).
5. Allows admin to edit booking details (change status, update customer information, etc.).
6. Includes options to cancel or approve bookings.
7. Shows total number of bookings in the system.
8. Displays payment information, including amount paid and pending payments.
9. Provides pagination for browsing large lists of bookings.