IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

Design and Implementation of a Web-Based Car Sharing System Using React, Python, and MySQL

Harshkumar R. Sharma, Tinkesh Y. Mondhe, Bhagyashree Kumbhare, Yamini Kanekar
Students, MCA, Smt. Radhikatai Pandav College of Engineering, Nagpur, India.
HOD, MCA, Smt. Radhikatai Pandav College of Engineering, Nagpur, India.

Abstract: *The Car Sharing System enables users to book vehicles remotely with ease and flexibility. By registering their personal details, users can create an account and access available vehicles through a fully integrated web-based platform. The system streamlines traditional manual booking processes, offering a user-friendly interface that allows individuals to select vehicles based on preferences such as type, location, and availability. This project aims to provide a seamless, automated solution for urban commuters and travelers to schedule rides efficiently from any location. The application ensures a convenient, scalable, and sustainable approach to personal mobility. The system is structured into three primary modules, detailed further in the introduction.*

Keywords: Car sharing, urban mobility, Django, React, transportation system, sustainability, ride-sharing.

I. INTRODUCTION

In the current urban ecosystem, the rise in population and vehicles has led to critical issues such as traffic congestion, air pollution, and underutilized private cars. Car sharing emerges as a sustainable alternative that promotes efficient vehicle use. This paper explores the design and implementation of a Car Sharing System that enables users to list, search, and book cars on a shared platform. The project serves both individual users and administrators, offering features like booking history, car availability, and user management. The Car Sharing System is designed to provide users with a convenient platform for booking and accessing vehicles without the need for ownership. Unlike traditional car rental models, this system promotes shared usage of vehicles to enhance mobility, reduce environmental impact, and lower transportation costs. It consists of three primary phases that govern its functionality:

- **Vehicle Pooling and Network Integration:** In the initial phase, the system categorizes vehicles into shared pools based on location. These pools allow registered users to access a collective fleet, promoting optimal vehicle utilization across nearby zones.
- **Fleet Management and Distribution Planning:** The second phase involves determining the vehicle types and quantities to be allocated to each pool. It includes long-term planning such as coordinating with car owners, adjusting supply based on user demand, and redistributing vehicles across different regions when required.
- **Daily Operations and Scheduling:** The final phase handles day-to-day vehicle availability, booking confirmations, and ensuring timely returns. The system dynamically manages reservations, making real-time updates to the fleet's availability within each pool.

A. Need for a Car Sharing System: In today's urban environment, owning a car is not always practical or economical. The rise of car sharing services offers a flexible alternative to ownership, allowing users to rent vehicles for short durations as needed. It eliminates expenses related to maintenance, insurance, and parking while supporting sustainable travel by reducing the number of vehicles on the road.

B. Objective of the System: The core aim of this system is to digitize and simplify the car sharing process, eliminating the hassle of manual bookings or reliance on traditional rental offices. It offers a responsive, user-centric interface that allows individuals to register, search for nearby vehicles, and book a ride within minutes. The system also supports service validation through feedback mechanisms and adheres to structured documentation such as Software Requirement Specifications (SRS) and Design Descriptions.

DOI: 10.48175/IJARSCT-26919

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

C. Methodology and Development Approach: The backend of the system is built using Django (Python framework), while the frontend utilizes React for a modern, interactive interface. Data is stored in a MySQL database. The system follows modular development practices, separating concerns across user management, booking services, and admin controls. Emphasis is placed on building scalable APIs for seamless communication between frontend and backend services.

D. Project Framework: Project follows a layered architecture, with clearly defined responsibilities across presentation, logic, and data layers. This framework ensures maintainability, easier debugging, and scalability. The design allows for future upgrades such as mobile app integration or AI-driven vehicle suggestions.

E. Data Handling and Security: Accurate and secure data management is critical to the success of the system. User profiles, booking history, vehicle details, and feedback are securely stored in the database. Role-based access ensures that only authorized personnel, particularly administrators, can manage sensitive data and perform system-level operations, thereby safeguarding user privacy and platform integrity.

II. LITERATURE REVIEW

Car sharing and rental platforms have evolved significantly over the past decade, driven by advancements in mobile and web technologies. Several existing systems such as Zoomcar, Ola Drive, and Uber Rent offer similar services but come with limitations, including high operational costs, complex booking procedures, and restricted customization for small organizations or academic use. Zoomcar provides a robust app-based rental system but primarily targets urban consumers with a well-established fleet. Ola Drive integrates vehicle rental with Ola's existing ride-sharing infrastructure but is limited in terms of flexible system customization or open integration for external use cases. Other platforms like Turo (in the U.S.) and Getaround offer peer-to-peer rentals, emphasizing convenience over control and security. Unlike these large-scale commercial solutions, the proposed Car Sharing System in this paper is a customizable, open-source-friendly platform built using modern web frameworks—React, Django, and MySQL. It is tailored to meet academic and organizational needs where resources, infrastructure, and customization play a key role. While most existing systems are closed-source and license-based, this project offers a developer-friendly structure with modular components, enabling rapid development and deployment in varied environments. Previous studies and papers in vehicle sharing and rental systems have often focused on the business model or logistics management; however, this paper emphasizes software design, user interface simplicity, backend data handling, and security testing. This research fills the gap by providing a full-stack implementation, which is both technically rich and academically applicable.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

The Car Sharing System follows the **Model-View-Controller (MVC)** architectural pattern to ensure separation of concerns, modularity, and ease of maintenance. This architecture splits the system into three core layers: the **Model** (data and database interaction), the **View** (user interface), and the **Controller** (business logic and routing).

Data Flow Diagram: The Data Flow Diagram shown below illustrates the general structure of the system. It demonstrates how and what sorts of services the customer chooses, as well as the amount of admin engagement.

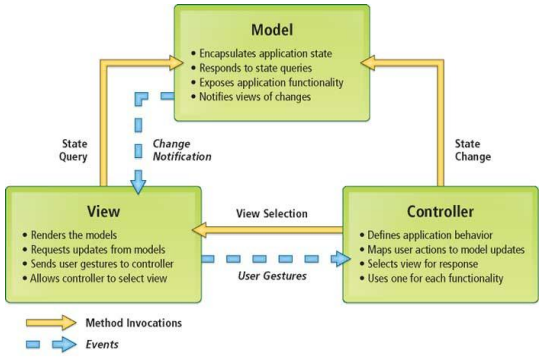

Fig 3.1 Data Flow Diagram

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

Sequence Diagram: A sequence diagram is comparable to an interaction diagram because it explains how and in what order a faction of items interact. A sequence diagram focuses on lifelines or processes and objects that exist concurrently, and the messages transferred between them to complete a function before the lifeline terminates.

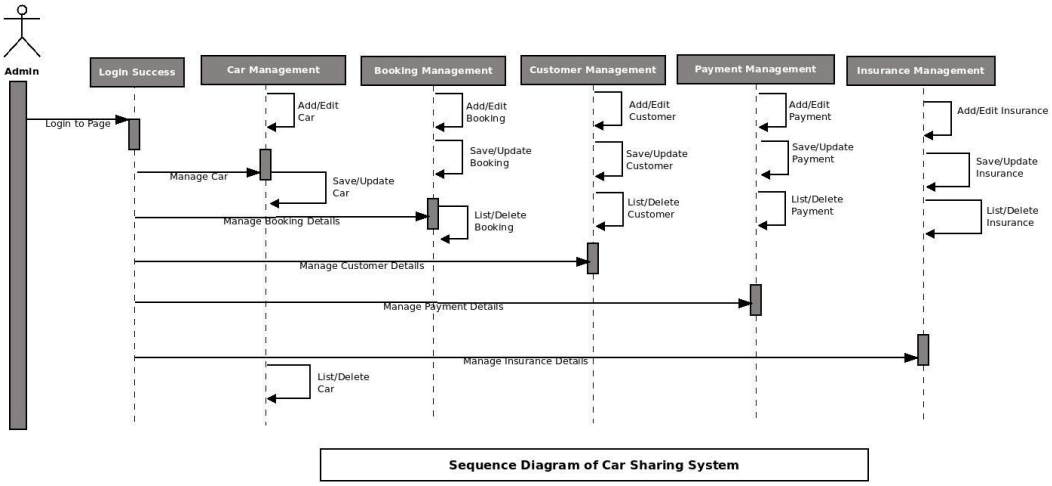

Fig 3.2: Sequence Diagram

ER/EER Diagram: The ER diagram depicts all of the relationships between entity sets in the database. It demonstrates the database's logical structure.

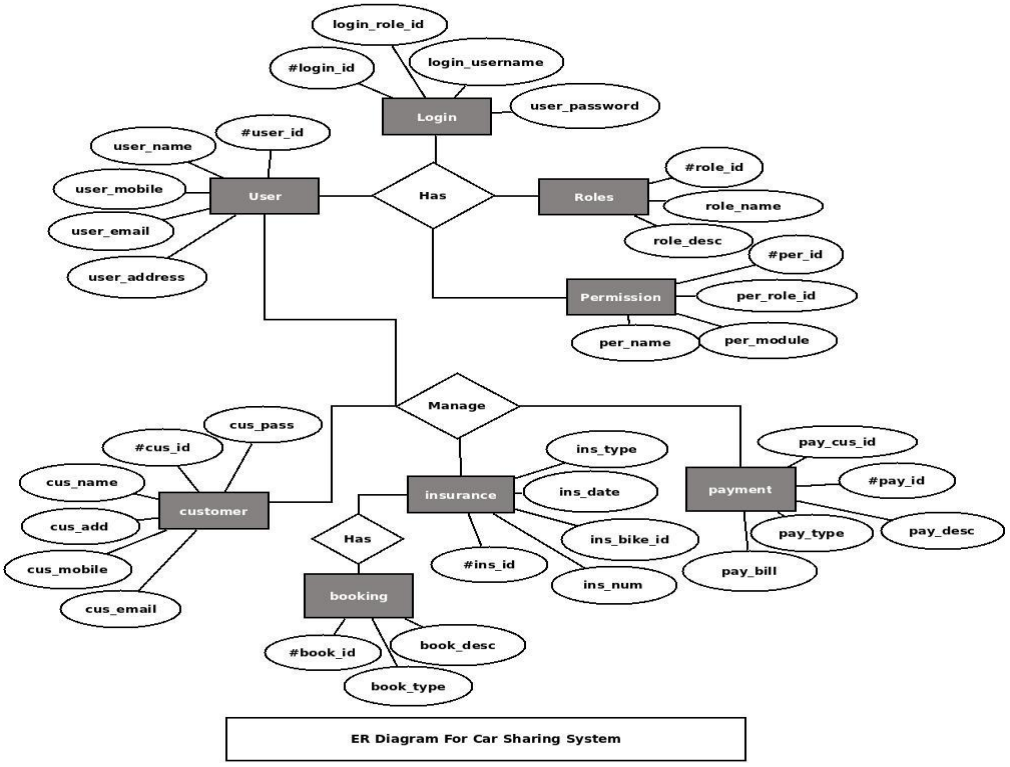

Fig 3.3: ER Diagram

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

Frontend: React.js – chosen for its component-based architecture, reactivity, and user-friendly interface design.

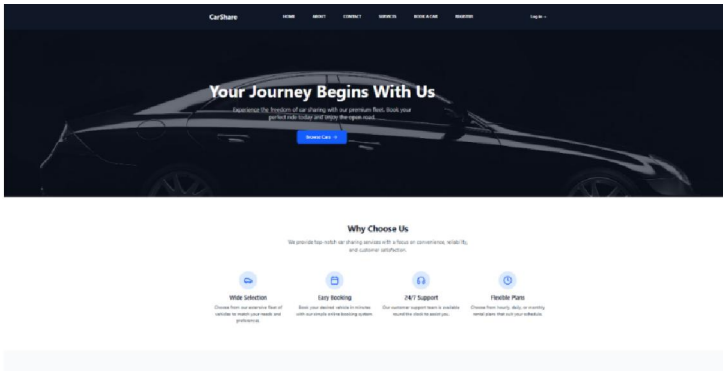

Fig 3.4 Home

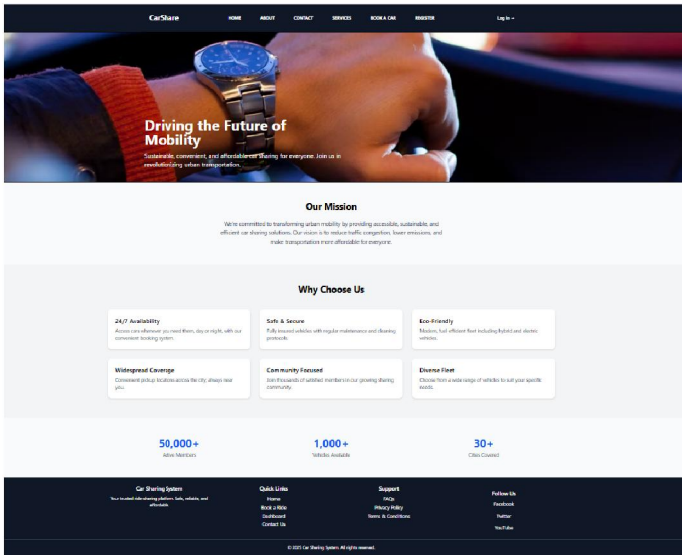

Fig 3.5 About

Backend: Django (Python) – provides robust security, ORM-based database access, and clean project structuring.

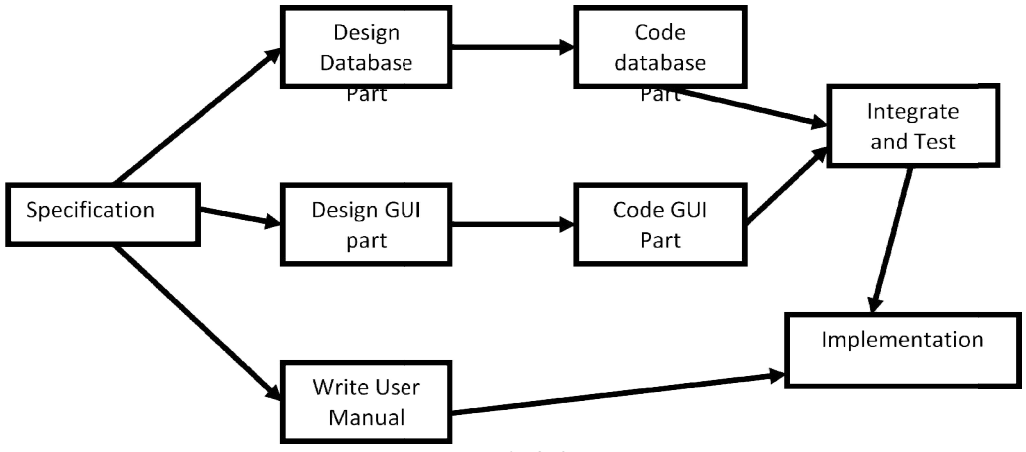

Fig 3.6 Backend

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

Database: MySQL – a reliable, open-source relational database used to store user data, car listings, bookings, payments, and history.

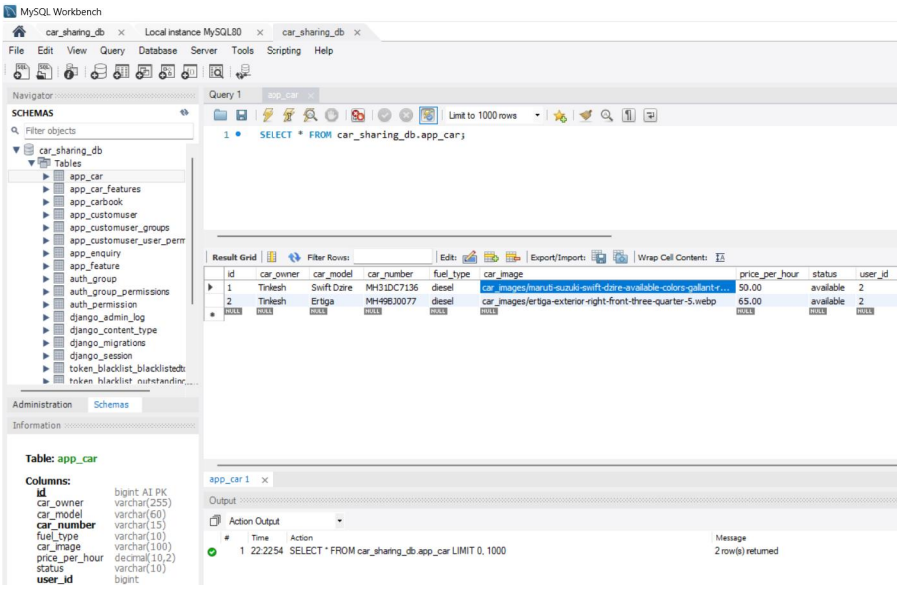

Fig 3.7 Database

Web Server: XAMPP or similar LAMP stacks – for development and deployment environments.

The system uses a normalized MySQL schema with entities like Users, Cars, Bookings, Payments, and Car History. Each table is linked via foreign keys to maintain referential integrity. An **Entity Relationship (ER) Diagram** was used during planning to model these relationships clearly. Data flow within the system follows a multi-tier architecture. Input from users is handled by the **React frontend**, passed to the **Django backend via RESTful routes**, processed, and then persisted in the **MySQL database**. Responses and results are returned and rendered dynamically on the frontend.

Data Flow Diagrams (DFD): Used to represent how data moves between modules such as Car Management, Booking, and Payment.

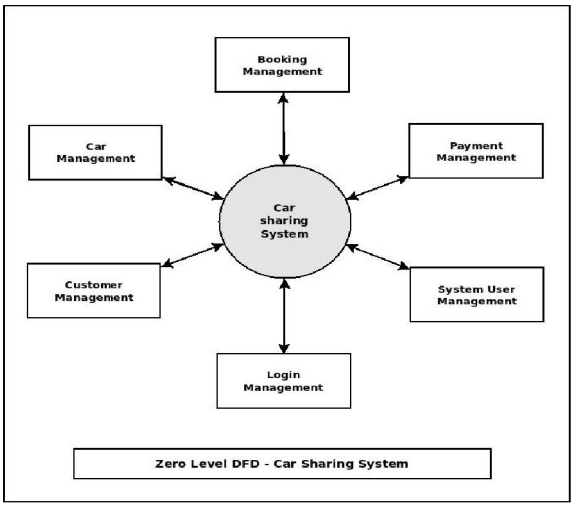

Fig 3.8 Data Flow Diagram (DFD)

Use Case Diagrams: Illustrate how users interact with the system – including Admin, Customer, and Vendor roles.

Class Diagrams: Represent the internal structure of objects and their interactions, helpful in visualizing the object-oriented backend. A waterfall model was adopted, beginning with requirement gathering, followed by design,

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

implementation, testing, deployment, and maintenance. Agile elements such as feedback-based iteration were used during UI refinement and testing

IV. FEATURES AND MODULES

The Car Sharing System is composed of multiple integrated modules, each handling a specific part of the functionality. Together, they provide a smooth and complete experience for both administrators and end users.

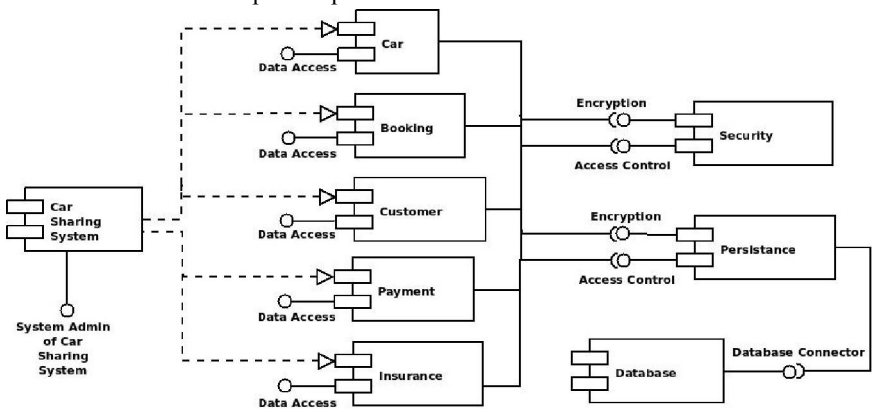

Fig 4.1 component Diagram

1. Car Management Module: **Allows vendors or admins to add, update, delete, and manage car listings. Each entry contains details like car type, company, price, source/destination locations, image, and description.**

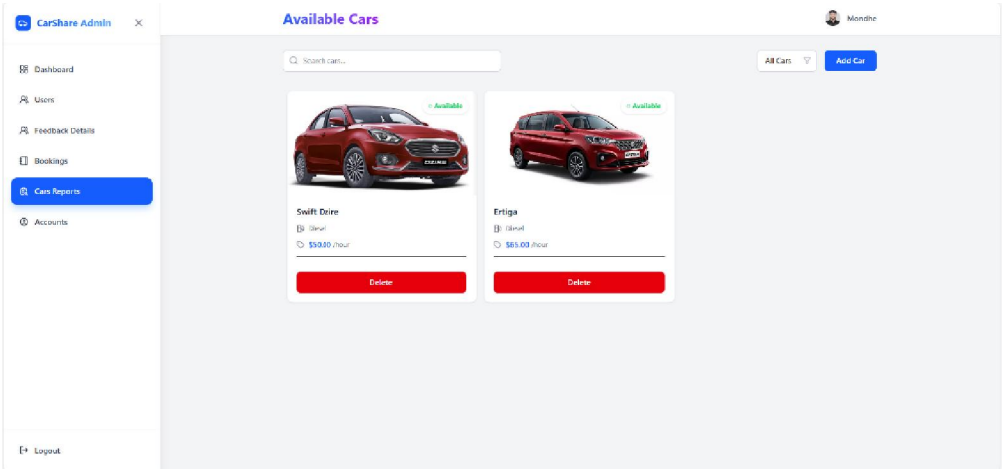

Fig 4.2 Car Management Module

2. Booking Management Module: **Users can view available cars based on selected routes and book them by entering pickup/drop details. Bookings are stored in the system and linked to both the car and user accounts.**

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

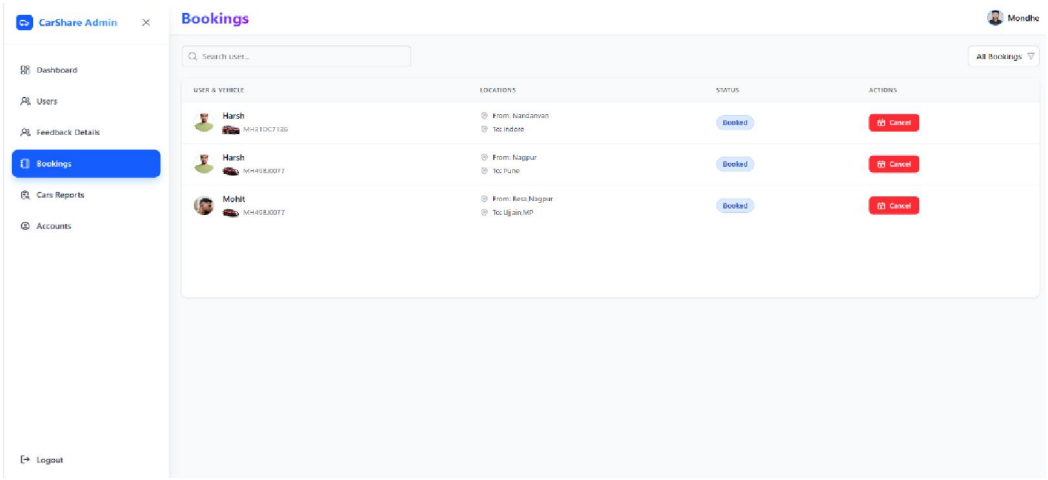

Fig 4.3 Booking Management Module

3. User & Role Management: The system supports role-based access,
Admins can manage the entire platform.
Vendors can manage their car listings.
Customers can search and book cars.
Each user is authenticated and managed securely with session handling and login/logout functions.

4. Payment Module: Handles payment-related records for bookings. Though online payment integration is optional, the system manages transaction logs, receipts, and reports for tracking.

5. Login & Authentication: Users are authenticated using Django's built-in auth system. Unauthorized access is restricted by role-based checks and validation.

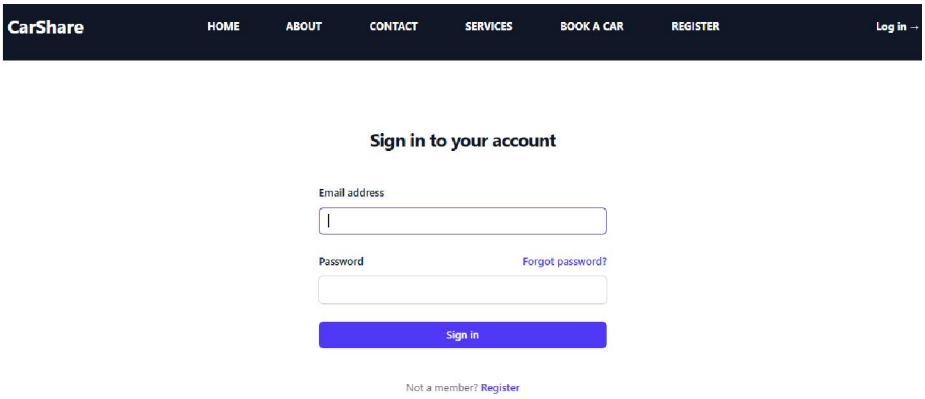

Fig 4.4 Login & Authentication Page

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

8. Car History Module: Tracks previous bookings or usage logs for each vehicle. Useful for maintenance scheduling or record keeping.

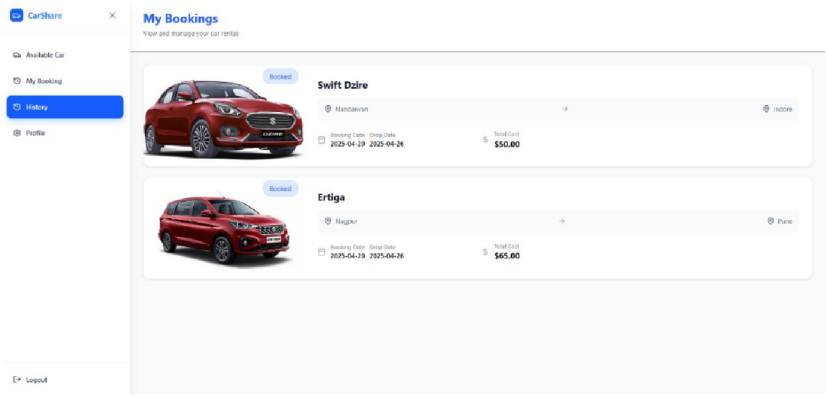

Fig 4.5 Car History Module

9. Dashboard: A central view for admin and vendors showing car stats, bookings, and recent activity.

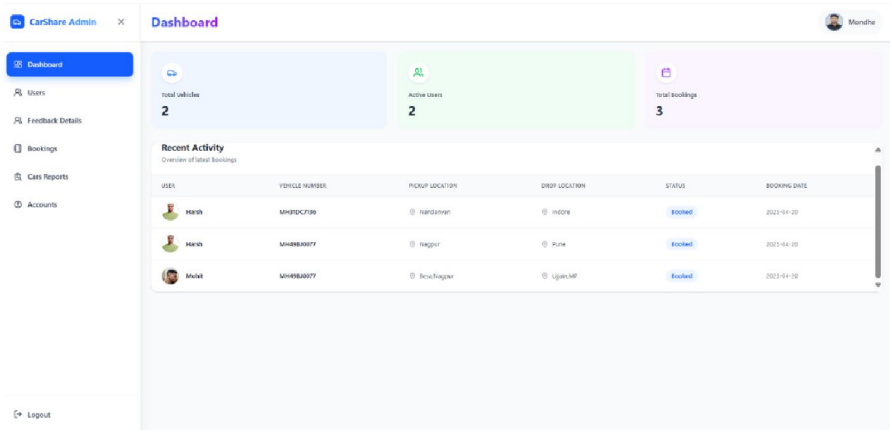

Fig 4.6 Dashboard

V. CONCLUSION

The design and implementation of the Car Sharing System using React, Django, and MySQL provide an efficient, scalable, and user-friendly platform for urban transportation. By leveraging modern web technologies, the system addresses the limitations of traditional car rental methods through automation, real-time booking, and intuitive user interfaces. The modular architecture ensures maintainability, extensibility, and ease of deployment across various environments, making it suitable for academic, organizational, and startup use cases. This system promotes sustainable urban mobility by encouraging vehicle sharing, thereby helping reduce traffic congestion, air pollution, and the underutilization of private cars. Key features such as role-based access control, car history tracking, secure data handling, and dynamic reporting ensure the platform remains robust and reliable. Future enhancements may include mobile application integration, AI-driven car suggestions, real-time GPS tracking, and advanced analytics to further enrich the user experience and operational efficiency. Overall, the project demonstrates a practical application of full-stack web development in solving real-world transportation challenges.

REFERENCES

[1] React.js (Frontend)

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 8, May 2025
Impact Factor: 7.67

Facebook Inc., "React – A JavaScript library for building user interfaces," React Documentation, Meta Platforms, Inc. [Online]. Available: https://reactjs.org/

[2] Django (Backend)

Django Software Foundation, "The Web framework for perfectionists with deadlines," Django Documentation, 2024. [Online]. Available: https://www.djangoproject.com/

[3] MySQL (Database)

Oracle Corporation, "MySQL 8.0 Reference Manual," MySQL Documentation, 2024. [Online]. Available: https://dev.mysql.com/doc/

[4] XAMPP / Apache (Web Server)

Apache Friends, "XAMPP Apache + MariaDB + PHP + Perl," XAMPP Project, 2024. [Online]. Available: https://www.apachefriends.org/

[5] M. Fowler, "Patterns of Enterprise Application Architecture," Addison-Wesley, 2003. [Book] — (Covers MVC, architectural best practices)

[6] S. Bradbury, "Full Stack Development with React and Django," Packt Publishing, 2021. [Book] — (Practical guide combining React frontend with Django backend)

[7] M. Grinberg, "Flask Web Development: Developing Web Applications with Python," O'Reilly Media, 2018. (Flask is a Django alternative; still useful for comparisons.)

[8] M. Beynon, "Building Dynamic Web Applications with ReactJS," International Journal of Web & Semantic Technology (IJWesT), vol. 12, no. 3, pp. 25–34, 2021.

[Online]. Available: https://aircconline.com/ijwest/V12N3/12321ijwest03.pdf

[9] M. Widenius and D. Axmark, "MySQL Reference Manual," MySQL AB, 2022. [Online]. Available: https://dev.mysql.com/doc/

[10] The Apache Software Foundation, "Apache HTTP Server Documentation," 2023. [Online]. Available: https://httpd.apache.org/

[11] GitHub Inc., "GitHub Docs – Version Control Using Git," 2024. [Online]. Available: https://docs.github.com/en

[12] I. Sommerville, "Software Engineering," 10th ed., Pearson, 2015. (Explains SRS, testing models, requirement analysis)

[13] K. Beck et al., "Manifesto for Agile Software Development," 2001. [Online]. Available: https://agilemanifesto.org/

[14] S. McConnell, "Code Complete: A Practical Handbook of Software Construction," Microsoft Press, 2004. (For clean coding and development practices)