International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

Uber Car Rental Booking Application Using Ror

Maddi keerthana[1], Rama Krishna Peddarapu[2]
[1]PG Student, Computer Science and Engineering VNRVJIET, Hyderabad, Telanagana, India Email:
[2]Assistant Professor, Computer Science and Engineering VNRVJIET, Hyderabad, Telanagana, India
Emails: *kasarlakeerthi0527@gmail.com[1] ,peddarapuramakrishna@gmail.com[2]*

Abstract
This research presents the development of a com- prehensive, web-based car rental booking application engineered using the Ruby on Rails (RoR) framework. The system addresses traditional inefficiencies in vehicle leasing by introducing an online platform that eliminates the need for manual, face-to-face negotiations between car owners and customers. By leveraging the Model-View-Controller (MVC) architectural pattern, the application achieves a clean separation of concerns, isolating complex business logic from the user interface to ensure high maintainability and security.The platform features a multi- tiered authentication system comprising Customers, Admins, and SuperAdmins, each with tailored privileges for managing fleet logistics, tracking insurance issues, and processing rental payments. Key technical implementations include a dynamic car searching and reservation engine, an automated waiting list man- agement system, and an integrated billing module that supports credit card transactions. To optimize resource allocation, the research incorporates a dispatch optimization logic inspired by the DISCO architecture, which utilizes Google S2 library cells for precise location tracking and demand matching.The backend is supported by an RDBMS (PostgreSQL) designed for horizontal scalability and high availability to handle frequent GPS and transaction updates. Experimental results and system testing indicate that the application significantly reduces operational delays, provides real-time transaction reporting, and offers a scalable solution for small-to-medium enterprises to digitalize their transportation services effectively.

Keywords: *Car Rental System, Ruby on Rails, MVC Architecture, Dispatch Optimization, Fleet Management, Web Application.*

1. Introduction

Uber is a technology-driven ridesharing platform that uti- lizes independent contractors as drivers rather than main- taining ownership of a centralized vehicle fleet. Operating within the sharing economy paradigm, the company provides a digital marketplace that connects passengers with drivers who typically use privately owned vehicles for transportation services [1], [2]. Through its primary mobile application, Uber facilitates real-time ride-hailing and route matching between riders and available drivers. To expand its transportation ecosystem, Uber also collab- orates with third-party vehicle leasing and rental providers, enabling drivers to access vehicles through rental partnerships and fleet management programs. Additionally, dedicated fleet management solutions such as Uber Fleet assist organizations and partners in coordinating groups of drivers and rental assets. A defining operational feature of Uber is its dynamic pricing mechanism, commonly referred to as surge pricing, wherein fares are adjusted in real time according to fluctuations in rider demand and driver supply. During peak demand periods, the platform increases fare prices to incentivize greater driver participation and maintain service availability [3], [4]. This adaptive pricing model enables the platform to balance supply-demand equilibrium efficiently while reducing passenger wait times. In recent years, Uber has expanded beyond conventional ride-hailing services by introducing hourly rental options and integrating complementary mobility services such as food delivery, courier logistics, freight transportation, and micro- mobility rentals through partnerships with providers such as Lime. Headquartered in San Francisco, Uber maintains operational presence in over 900 cities worldwide, demon- strating the scalability and global adoption of platform-based transportation systems [4].

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

2. Existing Work and Literature Review

The contemporary transportation landscape is increasingly shaped by the sharing economy, wherein digital platforms connect underutilized transportation resources with consumer demand through technology-enabled marketplaces [1], [2]. Leading platforms such as Uber operate as ridesharing systems that primarily rely on independent contractors as drivers instead of maintaining centralized vehicle ownership [3], [4].

2.1.Current Ridesharing Frameworks

Modern ridesharing platforms depend heavily on mobile applications to facilitate real-time matching between pas- sengers and drivers, who typically operate privately owned vehicles [1], [2]. To support users and drivers lacking vehicle ownership, many platforms collaborate with third-party leasing and rental providers to offer flexible vehicle access solutions [5], [6]. Furthermore, fleet management solutions such as Uber Fleet have been introduced to coordinate multiple drivers and optimize operational logistics across distributed transportation networks. A defining characteristic of these systems is the adoption of dynamic pricing algorithms. Such mechanisms continuously adjust fare prices based on real-time demand and supply condi- tions, often increasing prices during peak periods to incentivize driver participation and maintain service availability [3], [4].

2.2.Technical Methodologies in Demand Prediction

Recent research has focused on improving ridesharing ef- ficiency through predictive analytics and intelligent demand forecasting models. Machine learning techniques, including Markov models and Long Short-Term Memory (LSTM) neural networks, have demonstrated strong performance in predicting transportation demand by learning temporal mobility patterns and long-range dependencies in trip data [2], [4]. These approaches enable service providers to proactively allocate vehicles and optimize dispatch strategies in high-demand regions. Additionally, large-scale distributed data processing frame- works such as Apache Spark and cloud-based analytics pipelines have been employed to process streaming mobility data and identify demand hotspots in real time, thereby im- proving geographic demand prediction and route optimization [4].

2.3.Limitations of Traditional Rental Models

Although recent advancements in ridesharing and digital rental platforms have improved transportation accessibility, many conventional car rental systems still suffer from opera- tional inefficiencies that limit service quality and scalability.

- Manual Negotiation: In many traditional rental environ- ments, customers must contact vehicle owners or rental agents directly to confirm availability and finalize book- ing arrangements. This manual coordination introduces delays and increases operational overhead [7], [8].
- Decentralized Management: Many rental businesses operate without an integrated digital platform, leading to fragmented management of fleet records, insurance documentation, employee coordination, and payment tracking [9], [10].
- Information Asymmetry: Vehicle availability and rental status are often updated manually or inconsistently, creat- ing discrepancies between actual and displayed inventory information and reducing operational efficiency [6], [11].

To address these problems, the proposed system adopts the Ruby on Rails (RoR) system to automate reservation workflows, consolidate administrative operations, and establish a scalable digital infrastructure for modern vehicle rental management [12].

3. Proposed Work

Our system proposes the design and development of a centralized vehicle rental management platform intended to streamline the interaction between customers and rental man- agers while improving the overall efficiency of booking and fleet operations. The motivation behind the proposed system stems from the limitations observed in existing rental practices, where customers frequently encounter delays, fragmented communication, and limited transparency during the booking process [6], [10]. By replacing these manual procedures with an automated web-based platform, the proposed solution seeks to simplify rental operations and provide users with faster and more convenient access to vehicle booking

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

services [7], [12].

3.1.System Objectives and Administrative Control

A key objective of the proposed platform is to provide ad- ministrators with centralized control over the complete vehicle rental lifecycle through an integrated management dashboard. This allows operational activities to be coordinated from a single interface, thereby reducing administrative overhead and improving workflow visibility [9], [10]. The principal administrative capabilities of the system include:

- Fleet Management: Administrators can add new vehi- cles, update existing fleet information, remove unavail- able vehicles, and continuously monitor inventory status through the platform [9], [13].
- Dynamic Pricing Control: Rental prices can be adjusted according to operational and temporal factors such as de- mand trends, rental period, vehicle category, or seasonal requirements [3], [4].
- Operational Oversight: The platform consolidates staff coordination, rental agreements, maintenance tracking, and insurance management within a unified administra- tive environment [8], [10].
- Transaction Reporting: Real-time access to financial records and rental transaction summaries supports opera- tional monitoring and business decision-making [5], [6].

3.2.Architectural Advancements

From a technical perspective, the system is implemented using the Ruby on Rails (RoR) framework following the Model-View-Controller (MVC) architectural pattern. This architectural choice was made to support modular develop- ment, maintain clean separation between application layers, and facilitate long-term maintainability as the platform evolves [10], [12].The proposed architecture introduces several practical ad- vantages:

- Responsive User Interface: The client-facing interface is built using HTML, CSS, and JavaScript to ensure responsive and accessible interaction across desktop and mobile devices [8], [14].
- Efficient Resource Allocation: Vehicle assignment and availability matching mechanisms are incorporated to improve fleet utilization and reduce idle inventory by aligning available vehicles with customer demand in real time [2], [4].
- Reliability and Service Continuity: Backup allocation and replacement workflows are integrated to handle main- tenance issues or unexpected vehicle failures without disrupting ongoing reservations [5], [11].

By combining these administrative and architectural com- ponents within a single digital platform, the proposed system aims to simplify rental management operations, improve ser- vice reliability, and provide a scalable technological founda- tion for future expansion of vehicle rental services [6], [10].

4. Design And Methodology

The proposed Car Rental Booking Application is designed around a modular system architecture intended to support scalability, operational reliability, and efficient low-latency processing of booking and fleet data. The overall methodology combines modern web development practices with intelligent allocation mechanisms to better align vehicle availability with customer demand and improve the responsiveness of rental operations [10], [12].

4.1.System Architecture and Framework

As shown in Figure 1, the platform is structured using a modular service-oriented design to facilitate maintainability and future scalability. Rather than relying on a tightly coupled monolithic implementation, the system is developed using the Ruby on Rails (RoR) framework, which offers a structured environment for rapid backend development and streamlined deployment [10], [12]. Ruby on Rails natively follows the Model-View-Controller (MVC) architectural pattern, separating the application into three distinct functional layers:

- Model: Handles data relationships, validation logic, and interaction with the relational database.
- View: Responsible for rendering user-facing interfaces and presenting processed data dynamically through tem- plates.
- Controller: Processes incoming requests,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

coordinates ap- plication logic, and manages communication between the Model and View layers.

This separation of concerns improves modularity and sim- plifies long-term maintenance and feature extension of the application [8], [14].

4.2.Data Modeling and Entity Relationships

The underlying data model of the platform is defined through relationships among users, rental managers, reser- vations, and vehicle assets. Figure 2 illustrates the Entity Relationship (ER) model used to represent these operational dependencies. To support hierarchical access control, the platform imple- ments role-based authorization in which elevated user roles such as Admin and SuperAdmin inherit standard user at- tributes while receiving additional privileges for managing vehicles, reservations, and account records [9], [10].

4.3.Mapping, Routing, and Dispatch Optimization

Efficient allocation of rental vehicles requires structured modeling of geographic regions and service zones to support optimized dispatch and routing decisions. The proposed sys- tem adopts a hierarchical region classification strategy inspired by modern ridesharing platforms to improve routing efficiency and demand prediction [3], [4].

- Grade A: High-density urban regions and major trans- portation corridors.
- Grade B: Suburban and semi-urban service zones with moderate demand.
- Grade AB: Combined service regions representing over- lapping Grade A and B territories.
- Grade C: Highway and connector routes between service zones.

Dispatch Optimization (DISCO): Vehicle-to-customer assignment is handled through a dispatch optimization strat- egy inspired by Uber's DISCO dispatch framework, which matches available vehicles to customer requests using real- time location data and demand analytics [2], [4]. To improve spatial indexing efficiency and overcome lim- itations of raw latitude-longitude comparisons, the system employs Google S2 Geometry Library-based spatial de- composition. To improve spatial lookup efficiency and avoid the limitations of direct latitude–longitude comparisons, the proposed system employs the Google S2 Geometry Library for spatial decomposition. This approach divides the service region into uniquely indexed geographic cells, allowing the platform to organize and query location data more efficiently during dispatch operations. As illustrated in Figure 3, the dispatch engine segments the map into fine-grained spatial cells, each represented by a unique identifier. These identifiers function as spatial keys that accelerate location indexing and enable faster searching, matching, and updating of available vehicles during dispatch execution.

4.4.Database Requirements and Scalability

The backend infrastructure is supported by PostgreSQL, a relational database management system selected for its reliability in transactional processing and structured data per- sistence capabilities [10], [15]. To support future growth and increasing operational de- mand, the database layer is designed with the following scalability objectives:

- Horizontal Scalability: Supporting incremental capacity expansion through distributed deployment as user traffic and data volume increase.
- High Write Throughput: Accommodating frequent up- dates to reservation, fleet, and location records with minimal processing delay.
- Fault Tolerance: Maintaining service availability during backup operations, maintenance procedures, and storage expansion events.

5. Technologies And Frameworks

A set of modern web development technologies was chosen for the implementation of the platform based on development efficiency, scalability, and ease of maintenance. The imple- mentation incorporates the following technologies and frame- works, each selected based on its suitability for scalable web application development and maintainable system architecture:

- Ruby on Rails (RoR): Ruby on Rails serves as the primary backend development framework for the

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

applica- tion. Built on the Ruby programming language, it follows a convention-over-configuration approach that simplifies the development of database-driven web applications and RESTful services while supporting rapid prototyping and structured code organization [12], [16].

- Front-End Technologies: The user-facing interface is developed using HTML, CSS, and JavaScript, enabling as shown in Figure 1 System Architecture of the Proposed Car Rental Booking Platform, Figure 2 Entity Relationship Diagram and Operational Workflow of the Proposed System.

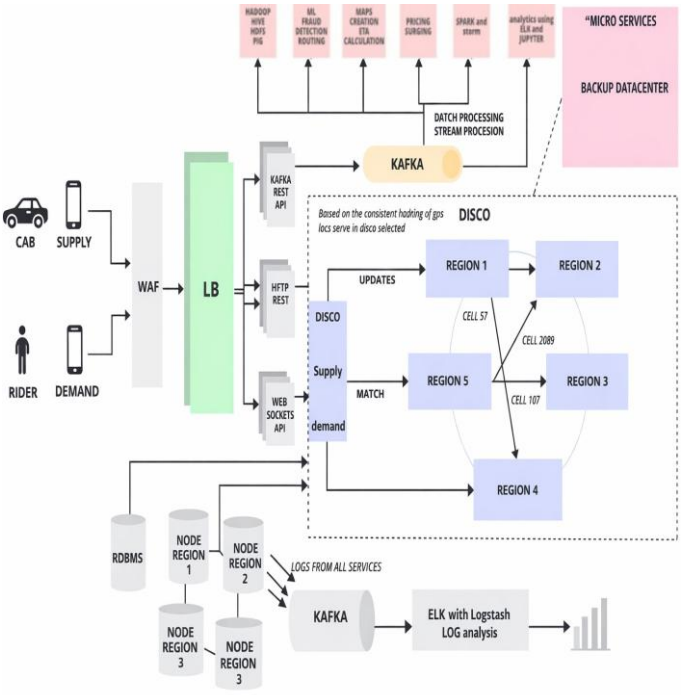

Figure 1 System Architecture of the Proposed Car Rental Booking Platform

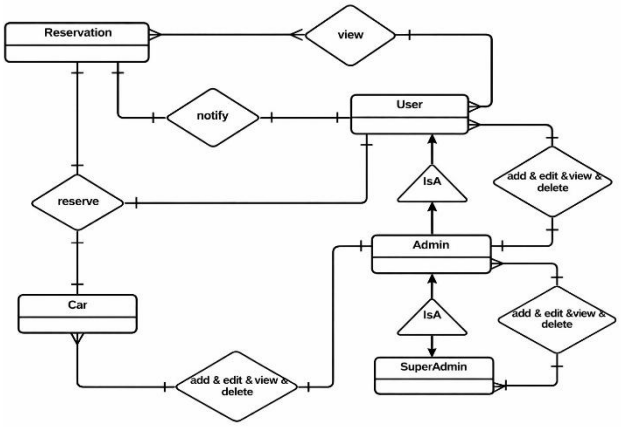

Figure 2 Entity Relationship Diagram and Operational Workflow of the Proposed System

responsive layouts and interactive functionality across multiple device types and screen resolutions [8], [14].Database Management: PostgreSQL is used as the re- lational database management system for storing and re- trieving reservation, vehicle, and user-related information. Its support for ACID-compliant transactions, concurrency control, and scalable data operations makes it suitable for high-volume transactional applications [15], [17].As shown in Figure 3 Dispatch Optimization Workflow for Vehicle Assignment

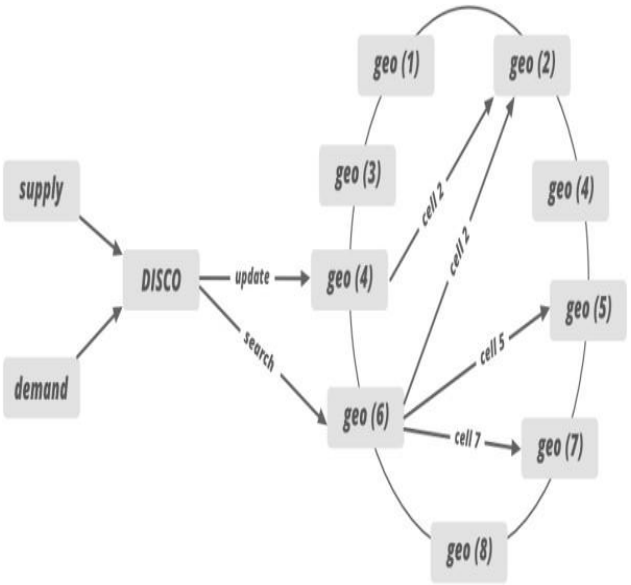

Figure 3 Dispatch Optimization Workflow for Vehicle Assignment

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

- Data Transfer Standards: Communication between ap- plication components follows established web protocols and structured data exchange practices to maintain inter- operability and support efficient client–server interactions in distributed deployment environments [18].

5.1.Architectural Pattern: Model-View-Controller (MVC)

The proposed application follows the Model-View-Controller (MVC) architectural pattern, which decomposes the software into separate layers for business logic, presenta- tion logic, and request processing. This separation of concerns significantly improves modularity, maintainability, testability, and extensibility of the application architecture [16], [19]. By isolating core application logic from user interface components, the MVC paradigm enables independent devel- opment, modification, and scaling of system modules without affecting overall application functionality.

6. Implementation And Module Analysis

The implementation of the proposed platform is carried out using the Ruby on Rails (RoR) framework, which offers a convention-driven environment well suited for rapid web application development. By adopting the Model-View-Controller (MVC) architectural pattern, the system maintains clear separation between business logic, data handling, and presentation components, making the application easier to maintain, extend, and scale over time [16], [19].

6.1.Development Environment and Tools

Development was performed primarily using Sublime Text, selected for its lightweight design and efficient support for web development workflows. The editor was chosen based on several practical advantages:

- Efficiency: Provides a responsive coding environment capable of handling medium to large project structures.
- Extensibility: Supports numerous plugins for debugging, syntax highlighting, linting, and formatting.
- Cross-platform Compatibility: Runs natively on Win- dows, Linux, and macOS.
- Version Control Support: Integrates effectively with distributed version control systems used in collaborative development.

6.2.System Modules

To organize the platform's functionality, the implementation is divided into multiple modules aligned with operational workflows and user responsibilities.

- Cars and Fleet Management Module: This module han- dles vehicle inventory management, fleet visibility, and search functionality.
- Search and Discovery: Users can locate available vehi- cles using search filters such as vehicle category, booking duration, and availability status.
- Vehicle Information: Each listing presents detailed metadata including manufacturer, model, registration number, and current rental status.
- Fleet Oversight: Authorized management users can add, modify, and remove vehicle records while monitoring overall fleet utilization [9], [13],As shown in Figure 4 Vehicle Search and Fleet Browsing Interface, Figure 5 Fleet Management Interface for Authorized Users.

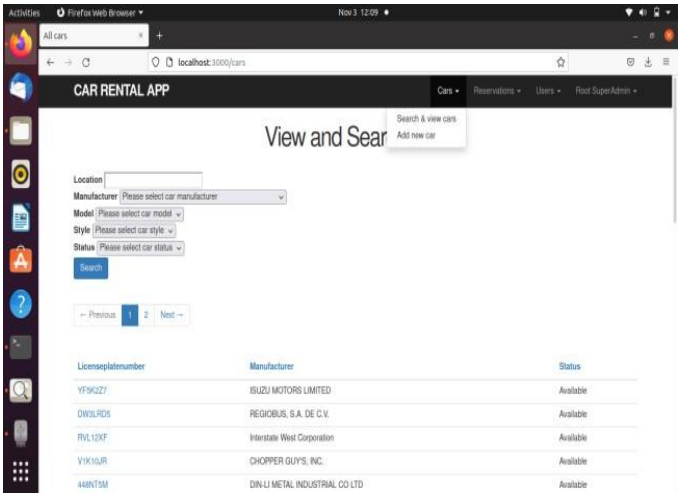

Figure 4 Vehicle Search and Fleet Browsing Interface

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

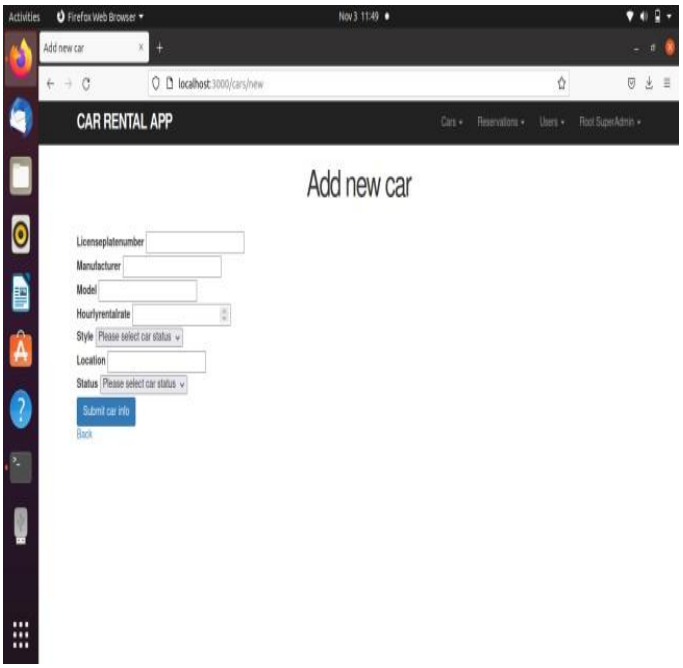

Figure 5 Fleet Management Interface for Authorized Users

6.3.Reservation and Waiting List Module:

Booking and reservation operations are coordinated through an automated reservation engine integrated with waiting list support.

- Booking Workflow: Customers reserve vehicles through a structured digital booking process.
- Waiting List Management: Prioritized waiting queues are maintained for vehicles that are temporarily unavail- able or overbooked.
- Reporting: Rental managers can review reservation records, transaction summaries, and utilization statistics in real time [6], [10].

6.4.User and Authentication Module:

User access is gov- erned through a secure role-based authentication and autho- rization mechanism.

- Account Management: Users register and maintain per- sonal profiles before accessing booking features.
- Role-Based Access Control: Separate permission levels are enforced for Customers, Admins, and SuperAdmins.
- Financial Transactions: Payment workflows are inte- grated into the reservation process to support secure transaction handling [7], [8].

6.5.Root SuperAdmin Functionality:

The SuperAdmin role represents the highest level of authority within the system,As shown in Figure 6 User Management Dashboard.

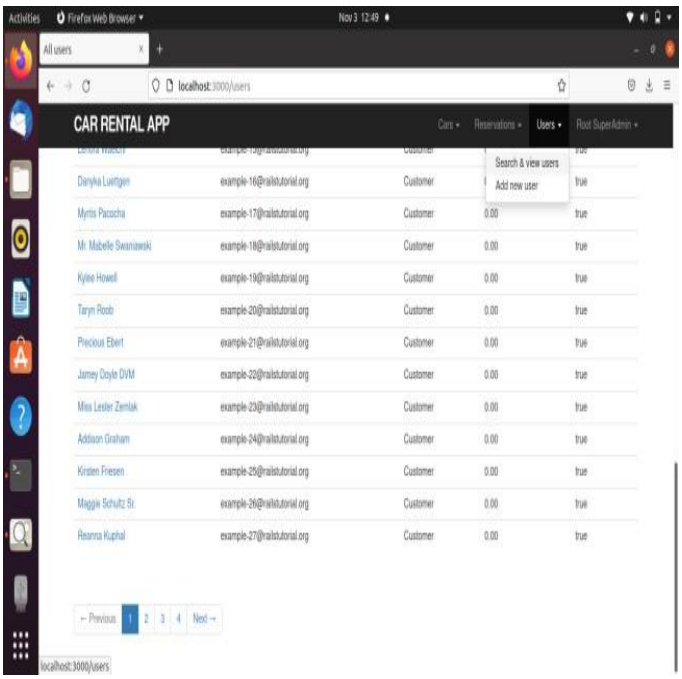

Figure 6 User Management Dashboard

Beyond standard management privileges, this role is respon- sible for configuring dynamic pricing strategies, approving replacement vehicles during maintenance events, and supervis- ing financial settlements with vehicle owners and associated stakeholders [3], [5], [11].

7. Software Logic and Code Architecture

The internal software structure of the Car Rental Book- ing Application is organized according to the Model-View- Controller (MVC) architectural pattern. This design approach separates the system into distinct layers for data handling, presentation, and request processing, which helps improve code maintainability, modularity, and overall software testa- bility [16], [19].

7.1.Model Component

The Model layer manages the application's core data logic, validation constraints, and business rules. It functions as the bridge between the database layer and the higher application components while encapsulating domain-specific operations related to

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

vehicles, users, and reservations [16],As shown in Figure 7 Implementation of Model Layer Showing Core Active Record Definitions

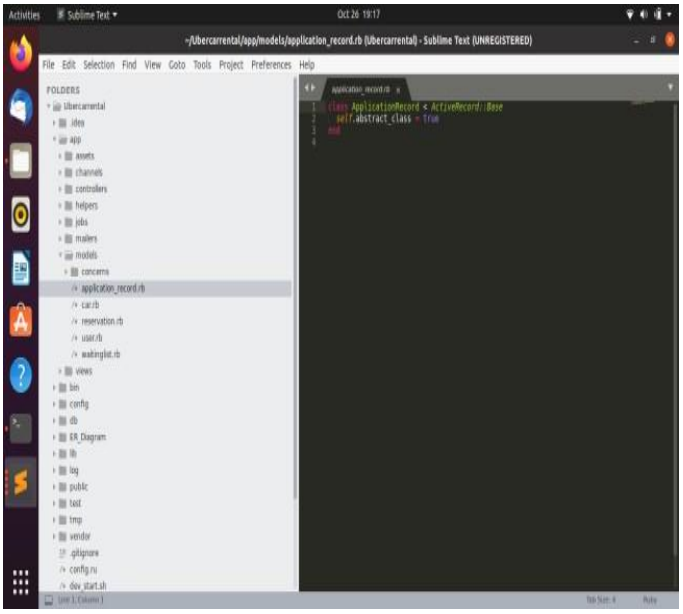

Figure 7 Implementation of Model Layer Showing Core Active Record Definitions

As shown in Figure 7, the main model definitions used in the implementation include:

- application record.rb: Base abstract class inherited by all models, enabling interaction with the database through Active Record ORM.
- car.rb: Defines vehicle-related attributes, validation rules, availability conditions, and fleet status logic.
- reservation.rb: Handles reservation workflows, booking states, and waiting list prioritization.
- user.rb: Stores user profile data and implements authen- tication and role-based authorization logic.

7.2.View Component

The View layer is responsible for presenting application data to end users through dynamically rendered interfaces. To achieve this, the platform uses Embedded Ruby (ERB) templates that combine HTML with server-side Ruby logic for generating dynamic web pages [16],As shown in Figure 8 Dynamic Vehicle Form Implemented Using Embedded Ruby Template

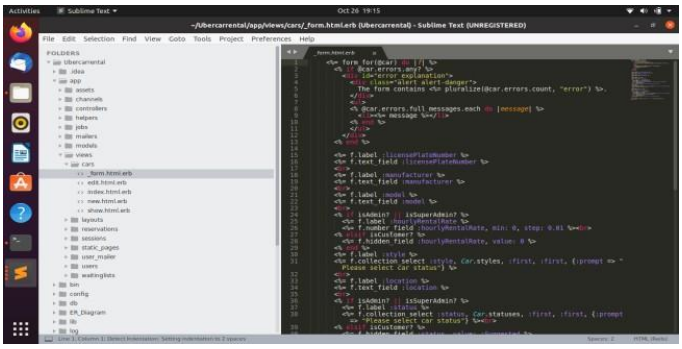

Figure 8 Dynamic Vehicle Form Implemented Using Embedded Ruby Template

Figure 8 presents the implementation of the vehicle input form, which includes conditional rendering logic to differen- tiate user permissions based on account roles. The principal view templates include:

- form.html.erb: Reusable form template for vehicle and reservation data entry.
- edit.html.erb: Supports modification of previously cre- ated records by authorized users.
- index.html.erb: Displays searchable listings of available vehicles and current reservation records.

7.3.Controller Component

The Controller layer coordinates communication between the Model and View components by processing incoming HTTP requests, invoking business logic through models, and selecting appropriate templates for response generation [16], [18].

- **application controller .rb:**Parent controller contain- ing shared authentication, authorization, and exception- handling functionality.
- **cars controller.rb:** Handles CRUD operations associated with vehicle inventory and fleet management.
- **reservations controller.rb:** Manages reservation cre- ation, confirmation, cancellation, and waiting list process- ing.

8. Testing And Experimental Results

Following implementation, the application

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

underwent system-level testing to confirm that it operates according to the defined functional requirements while maintaining acceptable performance and reliability. Software testing involves execut- ing the system under controlled input conditions to verify con- formance with expected behavior and identify implementation defects prior to deployment [20], [21].

8.1.Testing Phases and Methodologies

The evaluation process was divided into two principal testing phases:

- Unit Testing: Individual modules and software com- ponents were tested independently to validate localized business logic and functional correctness.
- System Testing: Integrated testing of the complete plat- form was conducted to verify end-to-end operational behavior and ensure satisfaction of system-level require- ments.

To comprehensively evaluate the application, multiple soft- ware testing strategies were employed, including[20]:

- White Box Testing: Verification of internal logic paths, control flow, and code-level correctness.
- Black Box Testing: Validation of external functionality against expected user-facing behavior without knowledge of internal implementation.
- Grey Box Testing: Combined testing approach utilizing partial knowledge of internal architecture to improve defect detection.

Although testing significantly improves software quality, it is recognized that the absence of detected defects does not guarantee complete system correctness [20].

8.2.Database Performance and Features

The backend of the proposed platform is supported by PostgreSQL, which is employed as the relational database management system due to its reliability in transactional pro- cessing, concurrent access handling, and scalable data storage capabilities [15]. To ensure that the database layer could meet the operational demands of the application, its design was assessed against the following requirements:

- Horizontal Scalability: Allows incremental expansion of database capacity as system usage and stored data increase over time.
- High Throughput: Supports frequent reservation updates and fleet-status modifications with minimal latency.
- Fault Tolerance: Preserves service availability during backup procedures, maintenance operations, and system recovery events[21].

8.3.Evaluated Core System Functionalities:

- Multi-role user management with privilege enforcement.
- Vehicle search and filtering mechanisms for fleet brows- ing.
- Dynamic reservation and waiting list processing work- flows[22].
- Automated notification delivery through email and system-generated alerts,as shown in Table 1 Default Authentication Credentials For System Evaluation

8.4.Experimental Setup and Default Credentials

For controlled system evaluation and testing, the applica- tion includes a predefined seeded dataset generated through db/seeds.rb. The initialized dataset contains:

- 1 SuperAdmin account
- 1 Administrator account
- 100 Default Customer accounts
- 10 Sample Customer Profiles
- 50 Vehicle Records

Table 1 Default Authentication Credentials for System Evaluation

User Role	Email Address	Default Password
Customer	example customer@email.com	123456
Administrator	root admin@email.com	123456
SuperAdmin	root superadmin@email.com	123456

8.5.Operational Workflow Validation

Operational validation demonstrated that the system cor- rectly supports the following role-dependent workflows:

- Authentication Workflow: Users are

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

able to register new accounts and authenticate successfully before accessing booking functionalities.

- Fleet Management Workflow: Customers can browse available vehicles, while authorized management users maintain fleet records under controlled modification per- missions.
- Reservation Workflow: Customers create and monitor their reservations, while management users over see booking activity across the platform.
- Waiting List Workflow: Prioritized waiting queues are managed by authorized personnel for vehicles that are temporarily unavailable.
- User Administration Workflow: SuperAdmin privileges enforce hierarchical control over user account manage- ment and role assignment.

Conclusion

The developed Car Rental Booking Application demon- strates how the Ruby on Rails (RoR) framework can be effectively applied to modernize conventional vehicle rental operations through scalable web-based service delivery. By digitizing rental workflows that were previously handled through manual and decentralized processes, the platform reduces operational delays, improves service accessibility, and simplifies digital adoption for transportation service providers [10], [12], [16]. The use of the Model-View-Controller (MVC) architec- tural pattern contributed to a modular system structure by sep- arating business logic, data handling, and presentation compo- nents. This architectural organization improves maintainability and extensibility while enabling centralized control over fleet operations, user management, and transaction reporting [16], [19]. In addition, the integration of dispatch optimization con- cepts inspired by the DISCO framework, combined with a scalable PostgreSQL backend, allows the platform to support high-volume reservation processing, real-time fleet updates, and scalable transaction management suitable for dynamic rental environments [15], [17], [22]. Overall, the proposed platform provides a practical and scalable solution for small- and medium-scale rental busi- nesses by improving fleet utilization, operational transparency, and customer interaction, while also offering a technological foundation for future system expansion and intelligent trans- portation service integration.

Acknowledgment

The author Maddi Keerthana, would like to express sincere gratitude to the supervisor, Mr. Rama Krishna Peddarapu, for his valuable guidance, continuous support, and insightful discussions throughout the course of this research work. The authors also acknowledge the Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering & Technology for providing the necessary research facilities and an encouraging academic environment.

References

[1]. N. Singh, V. G. Pandey, and N. Thilaras, “Web-based online car rental system,” IJAST, 2020.
[2]. G. Yatnalkar and H. Malik, “An enhanced ride sharing model based on human characteristics and machine learning recommender system,” 2020.
[3]. Jiang and L. Zhang, “The impact of the transport network companies on the taxi industry: Evidence from beijing’s gps taxi trajectory data,” 2020.
[4]. R. Srinivas and R. S. Krishna, “Uber related data analysis using machine learning,” 2021.
[5]. D. Asha, D. Vyahnavi, Rishika, and Shaurya, “A study on self-driving car rental service in india,” IJCRT, April 2023.
[6]. R. Rawat and S. Patel, “Car rental service,” IJCRT, June 2024.
[7]. P. P. Om, T. Rupesh, and M. Karad, “Car rental management system,”IJRPR, April 2023.
[8]. V. Mohite, P. Murkute, and S. Kakade, “Online car rental system using web technology,” IJRASET, May 2022.
[9]. L. P. R., A. K., and G. R., “Car rental management system,” IJNRD, March 2024.
[10]. J. T. Obgiti and W. Aaron, “Development of a web-based online car rental management system,” SWJ, vol. 19, 2024.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 06 June 2026
Page No: 4499 - 4509
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0590

[11]. K. Waje, Akshay, A. Patil, Ashwin, and P. T., "Car rental system," IJSSR, June 2024.
[12]. A. Singh, A. Shukla, A. Srivastav, and A. Pandey, "Online car rental system," IRJMETSI, May 2023.
[13]. R. Assyifa, S. J. Tao, and Suhendro, "Designing and implementing car rental system," in ICITB, July 2022.
[14]. M. Sumithra, M. Tharun, N. Suriya, and C. S. Vinoth, "Car rental system using php and mysql," IRJET, July 2022.
[15]. PostgreSQL Global Development Group, PostgreSQL Documentation, 2024. [Online]. Available: https://www.postgresql.org/docs/
[16]. M. Hartl, Ruby on Rails Tutorial. Addison-Wesley, 2023.
[17]. A. S. Tanenbaum and M. V. Steen, Distributed Systems. Pearson, 2017.
[18]. R. Fielding, "Architectural styles and the design of network-based software architectures," Doctoral Dissertation, University of California, Irvine, 2000.
[19]. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-Oriented Software Architecture: A System of Patterns. Wiley, 1996.
[20]. R. S. Pressman and B. R. Maxim, Software Engineering: A Practi- tioner's Approach. McGraw-Hill, 2019.
[21]. I. Sommerville, Software Engineering. Pearson, 2016.
[22]. Uber Engineering, "Dispatch optimization at uber with disco," https: //www.uber.com/blog/dispatch optimization/, 2021, accessed: 2026-04-12.