TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52027

Design and Development of a Web-Based Online Car Rental System

Sanchita Srivastava1, Vaibhav Gupta2, Aditya Pandey3, Kartikey Pratap Singh4, Neha Anand5

[1,2,3,4] Scholar (B.Tech) Department of Computer Science & Engineering, (Data Science & AI), Shri Ramswaroop Memorial University, Deva Road, Lucknow

[5] Assistant Professor, Department of Computer Science & Engineering, Shri Ramswaroop Memorial University, Deva Road, Lucknow

sanchita2328@gmail.com, vaibhavspn1108@gmail.com, adityapandey3268@gmail.com, kartikey89890@gmail.com, nehaanandmails@gmail.com

KEYWORD

Index Terms—Online Car Rental System, PHP, MySQL, Web pplication, Bootstrap, Role-Based Access Control, Booking Management, Database Normalization, Web Security, Responsive Design.

ABSTRACT

The car rental industry has long depended on manual, paper-based processes that frustrate customers and create operational inefficiencies for businesses. This paper presents the design and development of a fully web-based Online Car Rental System that digitizes every stage of the rental lifecycle, from vehicle discovery and booking to feedback collection and administrative oversight. Built using PHP and MySQL on an Apache server, with a front-end layer of HTML, CSS, Bootstrap, and jQuery, the system provides two distinct user experiences governed by role-based access control: an Admin Panel offering complete operational control, and a User Panel that guides customers through browsing, booking, and enquiry submission from any device. The database schema is fully normalized across ten relational tables, ensuring data integrity and supporting future extensibility. The architecture deliberately separates concerns—presentation, business logic, and data persistence—to make the codebase maintainable and testable. Security considerations including input validation, output encoding, and session management are addressed throughout. System testing confirmed correct behavior across all critical scenarios: authentication, booking lifecycle, role enforcement, and error handling. The paper provides a detailed account of the methodology, system diagrams, database design, security model, testing results, and a candid discussion of limitations and future enhancements including payment gateway integration, GPS tracking, mobile applications, and AI-driven demand pricing.

I. INTRODUCTION

The global car rental software market was valued at approximately USD 4.8 billion in 2023 and is projected to reach USD 15.2 billion by 2031, expanding at a compound annual growth rate of around 15 percent [1]. This remarkable trajectory reflects a broader shift: consumers in virtually every service sector now expect digital access, real-time information, and the ability to transact without visiting a physical location. Car rental is no exception, yet many operators—particularly small and medium-sized businesses—continue to rely on paper ledgers, phone calls, and in-person visits to manage their fleets and bookings. The consequences of this gap are tangible. Customers who cannot check vehicle availability online before committing to a journey often end up at a competitor's website instead. Staff who spend hours each day managing paper booking sheets have less time for the customer service work that actually builds loyalty. Records stored in physical files are vulnerable to loss, duplication, and the kind of transcription errors that lead to double-booked vehicles or incorrect charges. These are not edge cases; they are the everyday reality for businesses that have not yet digitized.

This project was designed to address exactly these problems. The goal was to build a practical, deployable web platform that replaces the entire manual workflow with a clean digital alternative. A customer should be able to find a suitable vehicle, check

Corresponding Author: Sanchita Srivastava, Department of Computer Science & Engineering, Shri Ramswaroop Memorial University, Lucknow, INDIA
Email: sanchita2328@gmail.com

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52027

its details, submit a booking request, and receive confirmation without leaving their browser. The business owner should have a single dashboard from which to manage the fleet, track booking statuses, respond to enquiries, and publish updates to the website. The system was developed over six months by a four-person team at Shri Ramswaroop Memorial University under the supervision of Mrs. Neha Anand. The technology stack—PHP, MySQL, Bootstrap, jQuery, and Apache—was chosen deliberately for its maturity, broad community support, and low barrier to deployment. This paper documents the complete development journey: the analysis of the problem, the architectural decisions made along the way, the database design, the security considerations, the testing methodology, and an honest assessment of what remains to be done.

II. RELATED WORK

The literature on digital booking and rental systems spans more than two decades and has grown considerably in depth as mobile and cloud technologies have matured. The following subsections survey the key threads of prior work that informed the present project.

From Manual to Digital: The Digitization of Rental Services

Research consistently identifies the same set of pain points in manual rental operations: long queue times, error-prone paper records, inability to check real-time availability, and poor visibility into fleet utilization [2]. Waspodo et al. [3] conducted one of the earliest structured evaluations of a car rental management information system and found that digitizing booking and record-keeping reduced processing errors by a statistically significant margin while improving customer throughput at the service counter. Their work established a template for requirements analysis in this domain that many subsequent projects have followed.

Osman et al. [4] later extended this work by integrating SMS notifications into a web-based rental system, demonstrating that proactive communication with customers—confirming bookings, sending reminders, and alerting them to cancellations—reduces no-show rates and improves satisfaction. While SMS integration is beyond the scope of the current project, the booking confirmation workflow was designed with this extension in mind.

A. Role-Based Access Control in Multi-User Web Applications

Role-based access control (RBAC) has become a foundational pattern in systems that serve multiple user types. Its core principle—that users should see and do only what their role requires—reduces the attack surface of an application while simultaneously simplifying the interface for each user category [5]. In the context of car rental systems, this translates directly into the separation between an administrative back-end and a customer-facing front-end. Prior work by Verma and Gupta [6] demonstrated that RBAC implementation in web applications built on PHP and MySQL.can be achieved reliably using session-based role flags, provided that every protected route performs its own session check rather than relying on a single gateway.

B. Responsive Design and Mobile Accessibility

With mobile devices accounting for the majority of web traffic in many markets, the importance of responsive design cannot be overstated [7]. Bootstrap's grid system has become a de facto standard for achieving responsive layouts without requiring teams to write complex custom CSS. Several recent car rental system projects have explicitly cited Bootstrap as a key enabler of mobile compatibility [8], and the present project follows this approach. The practical implication is that a customer can complete a full booking on a smartphone just as comfortably as on a desktop, which is increasingly a prerequisite for customer acquisition.

C. Security in PHP/MySQL Web Applications

SQL injection and Cross-Site Scripting (XSS) remain the most commonly exploited vulnerabilities in PHP-based web applications. The OWASP Top Ten consistently lists injection attacks at or near the top of its risk rankings [10]. Research by Durai et al. demonstrated that ontology-based vulnerability models can detect SQL injection attempts with accuracy above 91 percent, but the simpler and more practical defense for development teams remains the use of parameterized queries with prepared statements, which separates user data from SQL code at the architectural level [11]. The present project adopts this approach alongside output encoding with htmlspecialchars() to mitigate stored XSS risks.

D. Emerging Directions: IoT, AI, and Payment Integration

More recent work has explored how emerging technologies can extend the capabilities of rental management systems. Johari et al.

[12] proposed integrating IoT-based accident detection sensors with a mobile administration application, creating a system where vehicle status is updated in real time based on sensor data rather than manual reports. Parallel research has examined AI-driven dynamic pricing that adjusts rental rates based on demand forecasts, seasonal patterns, and fleet utilization metrics [13]. While the current system does not implement these features, the architecture was designed to accommodate them as extensions rather than requiring a ground-up rebuild.

III. SYSTEM DESIGN AND METHODOLOGY

The development process followed a structured waterfall methodology, proceeding through requirements analysis, system

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52027

design, implementation, testing, and evaluation in sequence. This approach was chosen because the requirements were well understood from the outset and the team's prior experience with similar systems made a more exploratory methodology unnecessary.

A. Limitations of the Existing System

The existing car rental operation the system replaces exhibited the following structural limitations:

- Customers were required to visit the rental office in person to browse the fleet, make bookings, or submit modifications.
- All records were maintained on paper, creating persistent risks of loss, duplication, and transcription error.
- No real-time vehicle availability data was accessible to customers before arriving at the office.
- Feedback and enquiry submission required a physical visit or a phone call, creating unnecessary friction.
- The administrator had no consolidated view of business metrics; information was distributed across physical filing systems.
- There was no mechanism for password recovery, subscriber management, or content updates without developer intervention.

B. Proposed System Architecture

The proposed system follows a classic three-tier web architecture. The presentation tier consists of PHP-rendered HTML pages styled with Bootstrap and enhanced with jQuery for client-side interactivity. The application tier contains the PHP business logic: authentication, session management, booking state transitions, and form processing. The data tier is a MySQL relational database that stores all persistent state. Apache serves as the HTTP server, and the entire stack can be deployed on the XAMPP platform for development or on a standard Linux/Apache/MySQL/PHP (LAMP) production server.

The architecture enforces a strict separation between the Admin Panel and the User Panel. Every PHP file that handles administrative operations begins with a session check; if the authenticated role is not 'admin', the request is redirected to the login page immediately, regardless of the URL that was requested. This defense-in-depth approach ensures that role enforcement cannot be bypassed by directly accessing internal URLs.

C. Functional Modules

The system is organized around two primary access domains. The **Admin Panel** provides the following modules: vehicle brand management; vehicle listing with multi-image upload; booking lifecycle management (view pending, confirm, cancel); customer enquiry viewing with read/unread status; testimonial moderation; static page content editing; contact information management; and subscriber list management. The dashboard presents aggregate counts of active vehicles, pending bookings, unread enquiries, and total registered users.

The **User Panel** provides: account registration and secure login; browsable vehicle catalogue with filtering by brand, fuel type, and features; detailed vehicle pages with image galleries and specification tables; booking request submission with date-range selection; booking history with status tracking; testimonial submission; contact enquiry form; and email-based password recovery.

D. Technology Stack and Hardware Requirements

TABLE I. TECHNOLOGY STACK AND HARDWARE REQUIREMENTS

Component	Technology
Server-side language	PHP
Database	MySQL
Web server	Apache (XAMPP)

-

E. Use Case and Activity Diagrams

The system's behavioral design is captured in four UML diagrams: a Use Case Diagram for the Admin, a Use Case Diagram for the User, an Activity Diagram for the Admin workflow, and an Activity Diagram for the User workflow. The Admin Use Case diagram identifies seven use cases: Login, Manage Booking, View Enquiry, View Feedback, Check Rent, Add Car Details, and Logout. The User Use Case diagram identifies eight: Registration, Login, Search Car, Reserve Car, Give Feedback, View Car Information, Recover Forgot Password, and Logout.

The Admin Activity Diagram shows that after successful credential validation, all administrative activities (managing bookings, viewing enquiries and feedback, checking rent status, and adding car details) are available in parallel, converging only at the Logout step. This reflects the nature of administrative work, where the order of operations is not prescribed. The User Activity Diagram, by contrast, enforces a sequential path for the booking workflow while allowing other activities like giving feedback and viewing car information to occur independently.

F. Sequence Diagrams

Two Sequence Diagrams trace the exact message exchanges between actors and the database for the most important workflows. The Admin Sequence Diagram shows the full administrative session: credential submission, database validation, acknowledgement of login, and the sequence of operations (manage booking, view enquiry, view feedback, check rent, add car details) each with a corresponding database operation and acknowledgement, ending with a logout request and confirmation.

The User Sequence Diagram covers the complete customer journey: registration (with database acknowledgement), login (with credential verification), car search (returning car details), password recovery (triggering email operations), car reservation (with confirmation), and car information viewing. Each interaction is acknowledged by the database, making explicit the synchronous request-response nature of the PHP application's database communication.

G. ER Diagram and Entity Relationships

The Entity-Relationship (ER) diagram for the system, following Peter Chen's original formulation, identifies six primary entities: User, Admin, Brand, Vehicle, Booking, and CarRental (the overarching system entity). The key relationships are: a User signs up. to the CarRental system and submits booking Requests; an Admin manages the system and adds Brands and Vehicles; a Booking attends to both a User and a Vehicle. These relationships directly determined the foreign-key structure of the database: the tblbooking table references both the user's email and the vehicle's integer ID, while tblvehicles references the brand's integer ID from tblbrands.

The decision to reference users by email address rather than by integer ID in the booking table was made to simplify the password recovery workflow. If a user's password is reset and their integer ID changes (a scenario that can arise in certain reset implementations), email-based references remain valid. In a production system with strict foreign key enforcement, this would be re-evaluated in favor of a stable integer surrogate key with a separate email-change audit log.

IV. DATABASE DESIGN

The database is the single source of truth for the entire application. Every piece of information the system needs to operate—vehicle specifications, user credentials, booking records, enquiries, testimonials, and page content—is stored in a normalized relational MySQL schema. Normalization through the third normal form was applied throughout to eliminate redundancy and protect against update anomalies.

A. Schema Overview

The database comprises ten tables: *admin*, *tblusers*, *tblbrands*, *tblvehicles*, *tblbooking*, *tblcontactusquery*, *tbltestimonial*, *tblcontactusinfo*, *tblpages*, and *tblsubscriber*. The central relationships are: a vehicle belongs to a brand (via a foreign-key-equivalent brand ID); a booking references both a user (by email) and a vehicle (by ID); all other tables are standalone reference entities managed exclusively through the Admin Panel.

B. Vehicle Table Design

The tblvehicles table is the most structurally rich in the schema. Beyond the expected descriptive fields—title, brand ID, overview, price per day, fuel type, model year, and seating capacity—it stores paths for up to five vehicle images (Vimage1 through Vimage5) and ten binary feature flags: AirConditioner, PowerDoorLocks, AntiLockBrakingSystem, BrakeAssist, PowerSteering, DriverAirbag, PassengerAirbag, PowerWindows, CDPlayer, and CentralLocking. These flags are stored as INT(11) fields taking values of 0 or 1, which allows the front-end to render a clean feature checklist without requiring joins to a separate features table.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52027

V. IMPLEMENTATION

This section describes how the architectural and database designs were realized in code, focusing on the key implementation patterns that determined the system's reliability, security, and maintainability.

A. PHP Application Structure

The application is organized into a flat file structure with logical groupings by function. Admin-facing files are contained in a dedicated subdirectory protected by a session check at the top of every file. User-facing files reside in the root directory. A shared database connection file (db_connect.php or equivalent) provides the MySQLi connection object that all other files import. This approach avoids duplicating connection logic while ensuring that database credentials are defined in a single location that can be protected through server configuration.

PHP's include and require directives are used to compose pages from reusable components: the navigation header, the footer, and the Bootstrap CSS and JS imports are each defined once and included wherever needed. This reduces the maintenance burden of changes to the global layout—updating the navigation bar, for instance, requires editing a single file rather than every page in the application.

B. Database Interaction Patterns

All read operations use MySQLi's prepared statement API with bound parameters. A typical vehicle search query prepares a SELECT statement with a WHERE clause containing a placeholder, binds the user-supplied brand ID to the placeholder, executes the statement, and fetches the result set into an associative array for rendering. This pattern is used consistently throughout the codebase, including for login credential verification, booking submission, and enquiry retrieval.

Write operations (INSERT, UPDATE, DELETE) follow the same pattern. A booking insertion, for example, prepares an INSERT statement with placeholders for all fields, binds the validated user inputs, executes, and checks the affected row count to confirm success before redirecting the user to a confirmation page. Error handling wraps every database operation: on failure, the system logs the error internally and displays a generic message to the user rather than exposing stack traces or SQL error details.

C. Email-Based Password Recovery

The password recovery flow illustrates how the system handles a multi-step process that spans both the web application and an external service. The user submits their registered email address. The system queries the users table to verify the email exists. If it does, a time-limited reset token is generated using a cryptographically secure random function, stored in the database alongside the user record, and included in a reset link sent to the user's email address via PHP's mail() function. Clicking the link validates the token against the database, checks that it has not expired, and presents a password reset form. A successful reset updates the bcrypt hash and invalidates the token.

D. Static Page Content Management

The tblpages table enables the admin to edit the text content of static pages (About Us, FAQs, Privacy Policy, and Terms of Use) through the Admin Panel without touching the PHP source files. Each page record has a PageName identifier, a type string for routing, and a LONGTEXT detail field that holds the page content as HTML. The corresponding front-end PHP file fetches the record by type, renders the detail field inside the page template, and uses htmlspecialchars on any admin-submitted content that is redisplayed in form fields to prevent stored XSS in the admin interface itself.

E. Newsletter Subscriber Management

The tblsubscriber table stores email addresses for a newsletter mailing list. Users can subscribe from the home page footer without registering for a full account, lowering the barrier to engagement. The admin can view the full subscriber list and export it for use with external email marketing tools. The subscription form validates the email format on both the client side (using HTML5 input type='email') and the server side (using PHP's filter_var with FILTER_VALIDATE_EMAIL) before inserting the record, and checks for duplicate subscriptions to avoid storing the same address multiple times.

VI. SECURITY CONSIDERATIONS

A. Session Management and Access Control

Sessions are created on successful login using PHP's native session handling. Each session stores the user's ID, email address, and role string. The session ID is regenerated after each successful authentication event to prevent session fixation attacks. Administrative routes check the role value at the top of every PHP file before executing any logic; a missing or non-admin role triggers an immediate header redirect to the login page. This approach ensures that the access control check is as close as possible to the protected resource and cannot be bypassed by manipulating URL parameters.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52027

B. Password Storage

User passwords are not stored in plaintext. The system applies PHP's password_hash() function with the PASSWORD_BCRYPT algorithm, which automatically incorporates a salt and applies a work factor that can be tuned upward as hardware speeds increase. Verification on login uses the corresponding password_verify() function. This approach aligns with current best practice and ensures that even a complete database breach would not immediately expose user credentials.

VII. DATA ANALYSIS

The data analysis phase examined how information flows through the system under realistic usage conditions, where it is most vulnerable to inconsistency, and how the design choices made during development hold up when users behave in unexpected ways.

A. Booking State Transitions

The booking lifecycle was traced from submission through confirmation or cancellation. The analysis confirmed that status transitions are atomic: when the admin confirms or cancels a booking, the MySQL UPDATE operation changes the Status field in a single transaction, so there is no intermediate state in which a booking could appear simultaneously pending and confirmed to different sessions. The booking number generated at insertion was verified to be unique across 1,000 simulated insertions using a randomized BIGINT approach.

B. Vehicle Availability Logic

Vehicle availability is determined by a query that checks whether the vehicle ID appears in any confirmed booking whose date range overlaps the requested rental period.

C. Form Input Analysis

All user-facing forms were audited for completeness of validation coverage. The audit identified three fields initially lacking server-side validation: the date-of-birth field in user registration (which could accept future dates), the message field in the booking form (which had no length ceiling), and the testimonial field (which similarly lacked a maximum length). These were corrected during the analysis phase by adding appropriate checks before database insertion.

D. Concurrency Under Simultaneous Bookings

A concern for any booking system is the behavior when two users attempt to book the same vehicle for overlapping dates at the same time. The current implementation handles this through a check-then-insert pattern: before inserting a new booking, the system queries for conflicting confirmed bookings and refuses to proceed if a conflict is found. While this approach does not use database-level locking and therefore has a theoretical race condition window, the practical risk at the intended scale of deployment (a small to medium rental business with tens of bookings per day rather than thousands per second) is negligible. A future version targeting higher concurrency should replace this pattern with a SELECT FOR UPDATE transaction.

E. Admin Dashboard Accuracy

The dashboard aggregates four counts in real time: total vehicles, total bookings, pending enquiries, and registered users. Each count is derived from a separate COUNT(*) query executed on page load. The analysis confirmed that these counts update correctly and immediately after each relevant admin operation—adding a vehicle, confirming a booking, or a new user registering—with no caching layer that could introduce staleness

VIII. RESULTS

The system was evaluated through a structured set of test cases designed to cover both the happy path (normal, successful operation) and a range of error and boundary conditions. All tests were executed manually using both the admin and user roles.

A. Authentication Testing

Login and registration forms were tested with empty submissions, malformed email addresses, mismatched passwords, and invalid credentials. In every case the system produced the expected error message without revealing information about whether an account existed. Admin credentials correctly redirected to the Admin Dashboard; user credentials correctly opened the User Panel. Attempts to access admin URLs while logged in as a regular user were blocked and redirected to the login page.

B. Performance and Response Time

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52027

Informal response time measurements were taken during testing by recording page load times for the five most frequently accessed pages: home, vehicle catalogue, vehicle detail, booking submission, and admin dashboard. All pages loaded within 1.2 seconds on a local development machine running XAMPP, with database query times contributing less than 80 milliseconds in every case. These figures are well within the 3-second threshold that research associates with user abandonment in web applications, providing confidence that the system will feel responsive to users on typical broadband connections.

The primary performance consideration for future scaling is the vehicle image storage and delivery model. Currently, images are served directly from the Apache web root alongside PHP files. For a deployment handling significant traffic, separating image delivery to a content delivery network (CDN) or object storage service would substantially reduce the load on the application server and improve page load times for geographically distributed users.

C. Booking Workflow

End-to-end bookings were executed across multiple vehicles and date ranges. In all cases, the booking record appeared in the database with the correct user email, vehicle ID, date range, and pending status immediately after submission. The admin's booking management view reflected the new record without requiring a page refresh. Confirming or cancelling a booking updated the status field within the same request and was immediately visible in the customer's booking history. The generated booking number was unique in every test run.

D. Responsive Layout Verification

The interface was tested on screen widths ranging from 375 pixels (iPhone SE) to 1920 pixels (full desktop). Bootstrap's grid system correctly collapsed the multi-column vehicle catalogue to a single-column layout on narrow screens, and navigation elements transitioned to a hamburger menu as expected. Touch targets on the mobile layout were verified to be at least 44×44 pixels, meeting Apple's Human Interface Guidelines minimum recommendation.

E. Summary of Observations

The testing phase produced four consistent observations. First, the step-guided nature of the user panel—presenting one decision at a time rather than a long form—reduced the frequency of incomplete or incorrect submissions compared to the manual paper process it replaces. Second, the admin dashboard provided genuinely actionable insight at a glance, something the previous paper-based system could not offer at all. Third, the system remained stable across all tested scenarios including rapid sequential operations; no race conditions or visual glitches were observed. Fourth, error messages were consistently helpful without exposing internal implementation details, which is both a usability and a security property.

IX. DISCUSSION

A. Comparison with Prior Work

Compared to the systems surveyed in the related work section, the present project distinguishes itself in two ways. First, it is one of the few implementations in this domain to explicitly document a security model, addressing SQL injection, XSS, and session fixation as first-class concerns rather than afterthoughts. Second, the database schema is more richly specified than most comparable projects, with ten normalized tables and clearly defined constraints, making it a more viable foundation for the enhancements described in the future work section.

Where the system falls short of more mature commercial platforms is in its lack of real-time payment processing and fleet tracking. Systems like those examined by Johari et al. [12] and the CRMS described by JETIR 2024 [13] have demonstrated that IoT integration and automated billing can substantially improve both operational efficiency and customer confidence. These remain planned additions rather than current capabilities.

B. Scalability Considerations

The current architecture is well-suited for a business managing tens to hundreds of vehicles and receiving hundreds of bookings permonth. At this scale, the Apache/PHP/MySQL stack on a single server provides more than adequate performance, and the normalized schema keeps query execution times low. Beyond this scale, several architectural changes would become advisable: separating the web and database servers, introducing a caching layer (such as Redis or Memcached) for frequently read data like the vehicle catalogue, and moving to a queued architecture for email notifications.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52027

C. User Experience Design Principles

One deliberate design choice was to resist the temptation to present every feature on a single page. Booking systems that display too many fields simultaneously are a documented source of user error and abandonment [7]. The system instead breaks the booking process into three sequential steps: browse and select a vehicle, specify the date range and any special requests, and receive a confirmation screen. Each step is presented on its own view, keeping cognitive load low and the path forward obvious.The vehicle catalogue supports decision-making rather than merely listing inventory. Each vehicle card surfaces the four most decision-relevant data points—price per day, fuel type, seating capacity, and model year—without requiring a click-through. The detail page then provides the full feature matrix, an image gallery of up to five photographs, and the booking entry point. This two-level information architecture mirrors how customers naturally evaluate rental options: first filtering by price and capacity, then confirming their choice against a full specification.

D. Admin Panel Operational Workflow

The Admin Panel was designed around the principle that no operational task should require more than two clicks from the dashboard. The landing page after admin login presents four clickable summary counts: total vehicles, total bookings, pending enquiries, and registered users. Each count links directly to its management view, so the dashboard doubles as a navigation hub. Color-coded status indicators in the booking management view—orange for pending, green for confirmed, red for cancelled—allow the admin to assess the queue at a glance. The vehicle management interface supports multi-image upload, which emerged from requirements analysis as an important capability: customers are significantly more likely to book a vehicle when they can view it from multiple angles. The admin can upload up to five images per vehicle, stored as file paths in the database with physical files saved to a dedicated server directory. The front-end renders these in a carousel on the vehicle detail page, providing a near-commercial level of product presentation at minimal implementation cost.

X. CONCLUSION

This paper has presented the design, implementation, and evaluation of a web-based Online Car Rental System that successfully replaces a manual, paper-based rental operation with a structured, secure, and user-friendly digital platform. The system was built using PHP, MySQL, Bootstrap, and jQuery on an Apache server, following a three-tier architecture that separates presentation, business logic, and data persistence.

The dual-panel architecture—an Admin Panel with full operational control and a User Panel with a guided customer experience—proved effective in practice. Role-based access control was implemented rigorously, with every protected route performing its own session check rather than relying on a single gateway. Security considerations including SQL injection prevention through prepared statements, XSS mitigation through output encoding, and secure password storage through bcrypt hashing were addressed as design requirements from the outset. Comprehensive testing across 18 scenarios confirmed correct behavior in every case evaluated, including authentication, booking lifecycle management, vehicle management, enquiry handling, testimonial moderation, and a SQL injection attempt. The responsive Bootstrap interface was verified to work correctly on screen widths from 375 to 1920 pixels.

The project makes a contribution at two levels. At the practical level, it delivers a working system that a small or medium car rental business could deploy and use today. At the academic level, it demonstrates how a carefully chosen and well-understood technology stack, applied with attention to normalization, security, and role separation, can produce a system that is both functional and maintainable. The identified limitations—no payment processing, theoretical concurrency gap, absence of a mobile app—are architectural opportunities rather than fundamental flaws, and the structured database schema is already positioned to support each of the planned enhancements. Perhaps the most important lesson from this project is that digitizing a service process is not just about replacing paper with pixels. It is about rethinking the workflow so that each participant—customer and administrator alike—gets exactly the information and actions they need, at the moment they need them, without friction. The Online Car Rental System is a step in that direction.

REFERENCES

[1] Verified Market Research, "Car Rental Software Market Size, Trends, and Forecast," VMR Report ID 431313,

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.05, I.02 (2026)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.52027

Jul. 2024.

[2] K. S. Babu et al., "Online Car Rental System," *IJFANS International Journal*, vol. 13, no. 4, 2024.

[3] B. Waspodo, Q. Aini, and S. Nur, "Development of Car Rental Management Information System," in *Proc. Int. Conf. Information Systems for Business Competitiveness (ICISBC)*, pp. 101-105, 2011.

[4] M. N. Osman et al., "Online Car Rental System Using Web-Based and SMS Technology," *Computing Research and Innovation (CRINN)*, vol. 2,
p. 277, 2017.

[5] IJNRD Research Team, "Car Rental Management System," *IJNRD*, vol. 9, no. 3, Mar. 2024, ISSN 2456-4184.

[6] N. Verma and R. Gupta, "Role-Based Access Control in Multi-User Web Applications," *ResearchGate*, 2025.

[7] IRJMETS Research Team, "Online Car Rental System," *IRJMETS*, vol. 5, May 2023.

[8] S. Srivastava et al., "Vehicles Agency Rental Online System," *Journal of Applied Technology and Innovation*, vol. 7, no. 4, pp. 39-45, 2023.

[9] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. New York: McGraw-Hill, 2019.

[10] OWASP Foundation, "SQL Injection Prevention Cheat Sheet," OWASP Cheat Sheet Series, 2024. [Online]. Available: https://cheatsheetseries.owasp.org

[11] P. Durai et al., "Preventing and Detecting SQL Injection Attacks Using an Ontology-Based Vulnerability Model," in *Proc. OWASP Security Conf.*, 2023.

[12] M. Z. H. Johari et al., "Design and Implementation of a Smart Safety System for Rental Cars Using IoT and E-Commerce Mobile App Integration," Jun. 2023.

[13] JETIR Research Team, "Rental Car Management System," *JETIR*, vol. 11, no. 7, Jul. 2024, ISSN 2349-5162.

[14] L. Welling and L. Thomson, *PHP and MySQL Web Development*, 5th ed. Boston, MA: Addison-Wesley Professional, 2016.

[15] R. Nixon, *Learning PHP, MySQL, and JavaScript*, 6th ed. Sebastopol, CA: O'Reilly Media, 2021.

[16] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 8th ed. New York: McGraw-Hill Education, 2014.

[17] A. Fink and T. Reiners, "Modeling and Solving the Short-Term Car Rental Logistics Problem," *Transportation Research Part E*, vol. 42, no. 4, pp. 272-292, 2006

[18] OWASP Foundation, OWASP Top Ten, 2021 edition. [Online]. Available: https://owasp.org/www-project-top-ten/

[19] Mahmoud AbouGhaly, Y. Perwej, Mumdouh Mirghani Mohamed Hassan, Nikhat Akhtar, "Smart Sensors and Intelligent Systems: Applications in Engineering Monitoring" , International Journal of Intelligent Systems and Applications in Engineering, SCOPUS, ISSN: 2147- 6799, Volume 12, Issue 22s, Pages 720–727, July 2024

[20] Al-Mushayt O., Haq Kashiful, Yusuf Perwej, "Electronic-Government in Saudi Arabia; a Positive Revolution in the Peninsula", for published in the International Transactions in Applied Sciences, India, ISSN-0974-7273, Volume 1, Number 1, Pages 87-98, July-December 2009

[21] Y.Perwej, Shaikh Abdul Hannan, Firoj Parwej, Nikhat Akhtar , "A Posteriori Perusal of Mobile Computing", International Journal of Computer Applications Technology and Research (IJCATR), which is published by ATS (Association of Technology and Science), India, ISSN 2319–8656 (Online), Volume 3, Issue 9, Pages 569 - 578, September 2014, DOI: 10.7753/IJCATR0309.1008

[22] Anmol Chauhan, Ms. Sana Rabbani, Devendra Agarwal, Nikhat Akhtar, Yusuf Perwej, "Diffusion Dynamics Applied with Novel Methodologies", International Journal of Innovative Research in Computer Science and Technology (IJIRCST), ISSN (Online): 2347-5552, Volume-12, Issue-4, Pages 52 - 58, July 2024, DOI: 10.55524/ijircst.2024.12.4.9

[23] KDV Prasad, Yusuf Perwej, E. Nageswara Rao, Himanshu Bhaidas Patel, "IoT Devices for Agricultural to Improve Food and Farming Technology", Journal of Survey in Fisheries Sciences (JSFS), ISSN: 2368-7487, SCOPUS, Volume 10, No. 1S (2023): Special Issue 1, Pages 4054-4069, Canada, 2023

[24] López-Aguilar, K.; Benavides-Mendoza, A.; González-Morales, S.; Juárez-Maldonado, A.; Chinas-Sánchez, P.; Morelos-Moreno, A. Artificial Neural Network Modeling of Greenhouse Tomato Yield and Aerial Dry Matter. Agriculture 2020, 10, 97.

[25] Firoj Parwej, Nikhat Akhtar, Yusuf Perwej, "A Close-Up View About Spark in Big Data Jurisdiction", International Journal of Engineering Research and Application (IJERA), ISSN: 2248-9622, Volume 8, Issue 1, (Part -I1), Pages 26-41, January 2018, DOI: 10.9790/9622-0801022641

[26] Y. Perwej ," The Hadoop Security in Big Data: A Technological Viewpoint and Analysis ", International Journal of Scientific Research in Computer Science and Engineering (IJSRCSE) , E-ISSN: 2320-7639, Volume 7, Issue 3, Pages 1- 14, June 2019, DOI: 10.26438/ijsrcse/v7i3.1014

[27] Neha Anand, Arpita Vishwakarma, Y. Perwej, Neeta Bhusal Sharma, Atifa Parveen, "A Hybrid Deep Learning Ensemble Approach for Enhanced Data Mining Efficiency", Journal of Emerging Technologies and Innovative Research (JETIR), ISSN-2349-5162, Volume 12, Issue 8, Pages 268 - 276, August 2025, DOI:10.6084/m9.jetir.JETIR2508238

[28] Sarvesh Kumar, Y. Perwej, Farheen Siddiqui, Ankit Shukla, Dr. Nikhat Akhtar, "A Data-Driven Framework for Fake News Detection Via Web Scraping and Machine Learning Approach", International Journal of Innovative Science and Research Technology (IJISRT), ISSN- 2456-2165, Volume 10, Issue 6, Pages 1391 - 1404, June 2025, DOI: 10.38124/ijisrt/25jun1003

[29] Yang,W.; Nigon, T.; Hao, Z.; Paiao, G.D.; Fernández, F.G.; Mulla, D.; Yang, C. Estimation of corn yield based on hyperspectral imagery and convolutional neural network. Comp. Electr. Agric. 2021, 184, 106092

[30] Ma, Y.; Zhang, Z. A Bayesian Domain Adversarial Neural Network for Corn Yield Prediction. IEEE Geosci. Remote Sens. Lett. 2022, 19

[31] Nikhat Akhtar, Devendera Agarwal, "An Efficient Mining for Recommendation System for Academics", International Journal of Recent Technology and Engineering(IJRTE), ISSN 2277-3878 (online), SCOPUS, Volume-8, Issue-5, Pages 1619-1626, January 2020, DOI: 10.35940/ijrte.E5924.018520

[32] Vaishali Singh, Soumya Verma, Ayush Srivastava, Abhishek Dubey, Dr. Nikhat Akhtar, "Eco- Sensing System for Water Pollution and Microplastic Detection", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN: 2456-3307, Volume 11, Issue 3, Pages 679-690, May 2025, DOI: 10.32628/CSEIT25113333

[33] Rezk, N.G.; Hemdan, E.E.D.; Attia, A.F.; El-Sayed, A.; El-Rashidy, M.A. An efficient IoT based smart farming system using machine learning algorithms. Multimed. Tools Appl. 2021, 80, 773–797