RESEARCH ARTICLE OPEN ACCESS

Design and Development of a Web-Based Vehicle Parking Management System Using PHP and MySQL

Mr .R.Ramakrishnan[1], A.Shereen Begam[2]

[1]Associate Professor and Head of Department of computer Applications, Sri Manakula Vinayagar Engineering College(Autonomous), Puducherry 605008, India

ramakrishnanmca@smvec.ac.in

[2]Post Graduate student, Department of computer Applications, Sri Manakula Vinayagar Engineering College (Autonomous),Puducherry 605008, India

shereenanvershereen@gmail.com

Abstract:

The proliferation of motor vehicles in urban environments has rendered manual parking management increasingly inadequate, resulting in revenue leakage, operational inefficiency, and poor audit traceability. This paper presents the design, development, and evaluation of a web-based Vehicle Parking Management System (VPMS) built on PHP and MySQL. The system automates the complete operational lifecycle of a parking facility: vehicle entry registration with auto-generated unique parking identifiers, real-time tracking of parked and departed vehicles, parking charge computation and assignment, date-range report generation, and income analytics. The application is structured around a three-tier architecture — an HTML5/Bootstrap 3 presentation layer, a PHP 7.x business-logic layer, and a MySQL 5.6 relational data layer — and employs jQuery DataTables for interactive record browsing and Chart.js for dashboard visualisation. The database schema comprises four normalised tables (admin, settings, vcategory, vehicle_info) supporting multi-category vehicle types (Two Wheeler, Four Wheeler, Three Wheeler). Functional testing across 24 test cases confirmed complete coverage of all modules. The paper further analyses current security limitations and proposes a roadmap of future enhancements including prepared-statement-based SQL injection mitigation, bcrypt password hashing, multi-user role management, and hardware (RFID/barrier) integration. Results demonstrate that VPMS substantially reduces manual processing time and provides management-level visibility into parking revenue and occupancy.

Keywords—*Vehicle Parking System; PHP; MySQL; Web Application; Bootstrap; Session Management; Chart.js; CRUD; DataTables; Three-Tier Architecture.*

I.INTRODUCTION

The global number of registered motor vehicles is projected to exceed two billion by 2035 [1]. Urban parking infrastructure, already strained in most metropolitan areas, faces growing pressure to operate with greater efficiency and transparency. Conventional paper-based systems — still widely deployed in smaller commercial complexes, hospitals, and educational institutions — are characterised by manual ticket issuance, error-prone charge calculation, absence of real-time occupancy data, and negligible audit capability.

Computerised parking management systems have been studied extensively in the literature. Existing commercial solutions tend to be expensive, require proprietary hardware, and lack the customisability needed by smaller operators. Open-source and academically proposed systems, by contrast, are often narrowly scoped, addressing only one aspect of parking operations (e.g., slot allocation [2] or payment [3]) rather than the end-to-end workflow.

This paper addresses this gap by presenting VPMS — a fully integrated, open-source, web-based parking management system developed entirely with freely available technologies: PHP, MySQL, Bootstrap 3, and JavaScript libraries. VPMS covers the complete parking transaction lifecycle from entry to exit, billing, and reporting, in a single cohesive application deployable on any standard LAMP/WAMP stack or Docker environment.

The primary contributions of this work are:

- A modular, three-tier web architecture for parking management, documented in sufficient detail to serve as a replication baseline.
- A normalised four-table MySQL schema that captures all data necessary for operational, revenue, and audit reporting.
- A functional evaluation using 24 defined test cases covering authentication, vehicle lifecycle, reporting, and security.
- A critical analysis of the current implementation's security posture and a structured enhancement roadmap.

The remainder of the paper is organised as follows: Section II reviews related work; Section III describes the system architecture and design; Section IV details the database schema; Section V describes module implementation; Section VI presents results and evaluation; Section VII discusses limitations and future work; Section VIII concludes.

II. RELATED WORK

Parking management as a research domain spans hardware-centric and software-centric approaches. IoT-based systems such as those proposed by Mainetti et al.

[4] and Pham et al. [5] use ultrasonic sensors and RFID to detect slot occupancy and automate barrier control. While highly automated, such systems require significant capital investment in embedded hardware and are impractical for cost-constrained operators.

Software-only approaches are more relevant to the present work. Idris et al. [6] proposed a database-driven parking information system using Microsoft Access and Visual Basic, demonstrating that relational storage is sufficient for transaction recording without hardware integration. However, that system was a desktop application, limiting multi-client deployment.

Web-based implementations have been proposed by several authors. Khanna and Anand [7] developed a smart parking system using Python/Django with Google Maps integration, focused primarily on slot reservation rather than operational billing. Abubakar et al. [8] implemented a PHP/MySQL parking system but limited their scope to entry/exit logging without revenue analytics or report generation.

Mobile-centric approaches [9][10] address the consumer-facing reservation problem but are orthogonal to the operator-side management problem addressed by VPMS. A survey by Ayala et al. [11] identified that most parking management research focuses on smart-city sensor networks, with comparatively little attention to the administrative software layer used by parking operators.

The present work differs from the above in its focus on completeness of the operator workflow: entry, exit, multi-category billing, daily/cumulative revenue analytics, date-range reporting, vehicle search, and category management — all within a single lightweight PHP/MySQL application deployable without any hardware dependency.

III. SYSTEM ARCHITECTURE AND DESIGN

A. Three-Tier Architecture

VPMS follows the classical three-tier web application architecture (Fig. 1), separating concerns across presentation, business logic, and data tiers.

Tier	Components	Responsibility
Presentation	HTML5, Bootstrap 3, Chart.js v2.2.2, jQuery 1.11.1, Font Awesome 4, DataTables	UI rendering, responsive layout, client-side charts and interactive tables
Business Logic	PHP 5.6/7.x — 22 server-side scripts	Form processing, session management, query execution, data validation, redirect control
Data	MySQL 5.6, InnoDB engine, vehicle-parking-db	Persistent storage, referential integrity, ACID-compliant transactions

Table I. Three-Tier Architecture Component Mapping

B. Technology Stack

The full technology stack is given in Table II. All components are open-source or freely available. The application can be deployed on XAMPP, WAMP, or a Docker container (PHP 7.4 + Apache + MySQL 5.6), making it accessible in resource-constrained academic and commercial environments.

Layer	Technology	Version
Frontend Framework	Bootstrap	3.x
Icon Library	Font Awesome	4.x
JavaScript Library	jQuery	1.11.1
Chart Library	Chart.js	2.2.2
Data Table Plugin	jQuery DataTables	1.10.x
Custom Font	Google Fonts — Montserrat	300–700
CSS Preprocessor	Sass/SCSS	Compiled to styles.css

Layer	Technology	Version
Backend Language	PHP	5.6 / 7.x (mysqli)
Database	MySQL	5.6 (InnoDB, latin1/utf8mb4)
DB Admin	phpMyAdmin	4.6.5.2
Web Server	Apache HTTP Server	2.4.x
Containerisation	Docker / Docker Compose	Optional

Table II. Technology Stack

C. Application Modules

The application comprises 22 PHP scripts grouped into 12 functional modules: (1) Authentication, (2) Dashboard, (3) Vehicle Entry, (4) IN Vehicles, (5) Vehicle Checkout, (6) OUT Vehicles, (7) Reports, (8) Income Analytics, (9) Vehicle Category Management, (10) Vehicle Search, (11) Receipt Printing, and (12) Administration (Profile / Settings / Password). All modules share four common include files: dbconn.php (database connection), navigation.php (top navbar), sidebar.php (left navigation with inline search), and footer.php.

D. Navigation and Session Flow

The entry point index.php authenticates the administrator and sets the session variable $_SESSION['vpmsaid']. Every protected page verifies this variable at load time; an absent or empty session immediately triggers a redirect to logout.php. The sidebar (sidebar.php) uses a PHP $page variable to highlight the active menu item dynamically. Vehicle category data in the entry form is loaded at runtime from the vcategory table, ensuring consistency between category management and entry operations.

IV. DATABASE DESIGN

A. Schema Overview

The database vehicle-parking-db contains four tables: admin, settings, vcategory, and vehicle_info. All tables use the InnoDB storage engine with AUTO_INCREMENT primary keys. Table III provides the schema for the central transactional table vehicle_info, which records every parking event.

Column	Type	Constraint / Default	Description
ID	INT(10)	PK, AUTO_INCREMENT	Unique parking record identifier
ParkingNumber	VARCHAR(120)	NULL	5-digit random token (mt_rand); displayed as CA-XXXXX
VehicleCategory	VARCHAR(120)	NOT NULL	Category name from vcategory (e.g., Four Wheeler)
VehicleCompanyname	VARCHAR(120)	NULL	Manufacturer/brand (e.g., Tesla, KTM, Hyundai)
RegistrationNumber	VARCHAR(120)	NULL	Vehicle registration plate (e.g., GGZ-1155)
OwnerName	VARCHAR(120)	NULL	Full name of vehicle owner
OwnerContactNumber	BIGINT(10)	NULL	10-digit contact number
InTime	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Entry time — auto-set on INSERT
OutTime	TIMESTAMP	NULL, ON UPDATE CURRENT_TIMESTAMP	Exit time — auto-set on UPDATE
ParkingCharge	VARCHAR(120)	NOT NULL	Charge collected; empty string until checkout

Column	Type	Constraint / Default	Description
Remark	MEDIUMTEXT	NOT NULL	Operator notes (e.g., 'Vehicle Out', 'NA')
Status	VARCHAR(5)	NOT NULL	'' = parked; 'Out' = departed

Table III. vehicle_info Schema (Core Transactional Table)

B. State Machine via Status Field

The Status column functions as a two-state machine. An empty string ('') indicates the vehicle is currently parked; the value 'Out' indicates the vehicle has departed. This design allows all IN-vehicle and OUT-vehicle queries to be expressed as simple equality predicates (WHERE Status='' and WHERE Status='Out'), minimising query complexity and enabling efficient indexing.

C. Automatic Timestamp Management

InTime uses DEFAULT CURRENT_TIMESTAMP to record the entry moment without requiring explicit PHP input. OutTime uses ON UPDATE CURRENT_TIMESTAMP, which causes MySQL to automatically populate the exit timestamp when the record is updated during the checkout process. This eliminates the possibility of an administrator forgetting to record the exit time.

D. Supporting Tables

The admin table stores administrator credentials (MD5-hashed password, security code for password recovery, email, mobile). The settings table stores organisation metadata (name, email, website, address) used in receipts and footers. The vcategory table provides a dynamic, manageable list of vehicle categories (seeded with Three Wheeler, Two Wheeler, Four Wheeler) that populates the vehicle entry form dropdown, decoupling category labels from hard-coded application logic.

V. MODULE IMPLEMENTATION

A. Authentication and Session Management

Login is handled by index.php. The submitted password is hashed with PHP's md5() function and compared against the stored hash using a parameterised WHERE clause. On success, $_SESSION['vpmsaid'] is set to the administrator's database ID, and the browser is redirected to dashboard.php. All 22 protected pages open with a session guard: if (strlen($_SESSION['vpmsaid']==0)) { header('location:logout.php'); }, ensuring that direct URL access without a valid session is impossible.

Password recovery uses a dual-factor model: the administrator must supply both their registered email address and a numeric Security_Code stored in the admin table (default value: 1100). On successful verification, session variables $_SESSION['secode'] and $_SESSION['email'] are set and the user is forwarded to a password reset form.

B. Vehicle Entry and Parking Number Generation

manage-vehicles.php accepts five fields: Registration Number, Company Name, Vehicle Category (dropdown from vcategory), Owner Name, and Owner Contact Number. On POST submission, PHP generates a unique 5-digit parking token using mt_rand(10000, 99999). This token, prefixed with 'CA-' for display, is stored in the ParkingNumber column. The INSERT statement omits InTime (relying on MySQL DEFAULT), Status, ParkingCharge, and Remark (defaulting to empty strings). On success a JavaScript alert confirms the entry and the page redirects to the dashboard.

C. Vehicle Exit Processing

When an administrator selects 'Take Action' on an IN-vehicle record, update-incomingdetail.php loads the vehicle's full details in read-only fields alongside three editable inputs: Total Charge (numeric), Status (dropdown: 'Out' only), and Remarks (text). On submission, the UPDATE statement sets ParkingCharge, Status='Out', and Remark for the given ID. MySQL's ON UPDATE CURRENT_TIMESTAMP on the OutTime column automatically records the precise exit timestamp without any PHP logic. The vehicle is immediately visible in the OUT Vehicles list.

D. Dashboard Analytics

The dashboard (dashboard.php) presents four real-time KPI widgets and two Chart.js pie charts. KPI values are fetched by six modular PHP include files in the /counters/ directory, each executing a single aggregate query (COUNT or SUM). This modularity allows each counter to be reused independently on other pages (e.g., income-count.php is reused on total-income.php).

The two pie charts use Chart.js v2.2.2 with data injected directly from PHP echo statements into JavaScript data arrays. Chart 1 visualises the IN/OUT vehicle ratio (blue: #30a5ff, teal: #33cccc). Chart 2 visualises the vehicle category distribution (Two Wheeler: red #f55d42, Four Wheeler: amber #f5c542, Three Wheeler: grey #6b6b6b). Both charts are rendered on HTML5 canvas elements.

E. Report Generation and Income Analytics

Date-range reports are generated by POSTing FROM and TO date values to generate-reports.php, which executes: SELECT * FROM vehicle_info WHERE date(InTime) BETWEEN '{fdate}' AND '{tdate}'. Results are presented in a DataTables-enhanced table supporting client-side sort, search, and pagination. Income analytics (total-income.php) executes three SUM(ParkingCharge) queries: all-time total, today (OutTime >= CURDATE() AND OutTime < CURDATE() + INTERVAL 1 DAY), and yesterday (date(OutTime) = CURDATE()-1).

F. Receipt Printing

print-receipt.php generates a formatted receipt table for any vehicle record fetched by its ID. A JavaScript CallPrint() function captures the receipt div's innerHTML, opens a new browser window, writes the content, and invokes window.print(). This browser-native approach requires no server-side PDF library, keeping the implementation lightweight and dependency-free.

G. Search Functionality

The sidebar's vehicle search form submits to search-results.php using a LIKE query with prefix matching: SELECT * FROM vehicle_info WHERE RegistrationNumber LIKE '{sdata}%'. This enables partial registration number lookups. If the matched vehicle is still parked (empty Status), a 'Take Action' button is rendered directly in the results row, enabling one-click navigation to the checkout form.

VI. RESULTS AND EVALUATION

A. Functional Test Results

The system was evaluated against 24 functional test cases covering all modules. Table IV summarises key test outcomes. All 24 test cases passed under black-box testing on a local XAMPP environment with the seeded database (20 vehicle records across Three Wheeler, Two Wheeler, and Four Wheeler categories; 13 departed vehicles, 7 currently parked).

TC	Module	Scenario	Result
TC-01	Authentication	Valid credentials — admin / Password@123	Pass
TC-02	Authentication	Invalid credentials	Pass — error alert shown
TC-03	Session Guard	Direct URL access without login	Pass — redirect to logout.php
TC-04	Vehicle Entry	Submit complete entry form	Pass — record inserted; dashboard redirect
TC-05	Vehicle Entry	Category dropdown population	Pass — vcategory data rendered
TC-06	Checkout	Assign charge and mark Out	Pass — Status, ParkingCharge, OutTime updated
TC-07	Receipt	Print button on OUT Vehicles	Pass — browser print dialog opened
TC-08	Reports	Date range 2021-03-01 to 2021-03-31	Pass — 9 matching records displayed
TC-09	Income	Total / Today / Yesterday panels	Pass — correct SUM values rendered
TC-10	Search	Prefix 'GGZ' search	Pass — vehicle GGZ-1155 returned
TC-11	Search	Non-existent prefix	Pass — 'No Results Found!' alert shown
TC-12	Category CRUD	Add / Edit / Delete category	Pass — all operations reflected in DB
TC-13	Dashboard Charts	Pie chart data accuracy	Pass — IN/OUT and category ratios correct
TC-14	Password Recovery	Correct email + security code 1100	Pass — redirect to reset form

Table IV. Selected Functional Test Case Results (14 of 24 shown)

B. Performance Observations

All pages loaded within 1.5 seconds on a local XAMPP installation (Intel Core i5, 8 GB RAM, MySQL 5.6, Apache 2.4). Dashboard page load — the heaviest page due to six counter queries and two chart data queries — completed in under 800 ms. DataTables pagination

eliminates the performance impact of large vehicle_info record sets by performing all sorting and filtering client-side after the initial full-table fetch.

C. Dataset

The seeded database contains 20 vehicle records spanning real-world brands (Hyundai, Tesla, KTM, Yamaha, Volkswagen, Ford, Suzuki, Honda, Kawasaki, Renault, Chevrolet, MG, Aprilia, Piaggio) across all three vehicle categories. Thirteen records have Status='Out' with recorded ParkingCharge values (range: $2–$34) and OutTime; seven records remain parked (Status=''). This dataset provides a realistic representation of a mixed parking facility state.

VII. LIMITATIONS AND FUTURE WORK

A. Security Limitations

The most significant limitation of the current implementation is the use of direct string interpolation in SQL queries, which exposes the system to SQL injection attacks. All database interactions should be migrated to MySQLi Prepared Statements with parameter binding before any production deployment. Additionally, MD5 password hashing is cryptographically weak; PHP's password_hash() function with the PASSWORD_BCRYPT algorithm should replace it. The current single-factor login should be augmented with TOTP-based two-factor authentication for production use.

B. Functional Limitations

The system supports only a single administrator account with no role differentiation. Multi-user deployment (e.g., multiple booth operators supervised by a manager) requires role-based access control (RBAC). Physical parking slot assignment is not supported; ParkingNumber is a random transaction identifier, not a physical slot locator. No automated notifications (SMS/email) are sent to vehicle owners on entry or exit. The report module generates a vehicle listing but does not produce a downloadable PDF; a server-side PDF library (TCPDF or mPDF) integration is required for print-ready reporting.

C. Future Enhancements

A structured roadmap of enhancements, in order of priority, is proposed:

- Security hardening: Prepared statements, bcrypt hashing, 2FA.
- Multi-user RBAC: Admin, Operator, and Viewer roles with granular permissions.
- Physical slot management: Visual slot map with real-time occupancy indicators.
- Payment gateway integration: Razorpay/Stripe for digital fee collection.
- PDF report export: Server-side PDF generation via mPDF.
- SMS/email notifications: Twilio/SendGrid integration for owner alerts.
- Monthly/annual revenue charts: Chart.js bar and line chart extensions.
- RFID/barrier hardware integration: Serial/USB RFID reader support via a middleware service.
- REST API layer: Enable mobile app clients and third-party integrations.
- Progressive Web App (PWA): Offline capability and mobile-first booth interface.

VIII. CONCLUSION

This paper has presented the design, implementation, and evaluation of VPMS, a web-based vehicle parking management system built on PHP, MySQL, Bootstrap 3, and JavaScript. The system addresses the operational shortcomings of manual parking management by automating the full transaction lifecycle — vehicle entry, exit, billing, receipt printing, date-range reporting, and income analytics — within a lightweight, hardware-independent, open-source application.

The three-tier architecture, four-table normalised database schema, and modular PHP codebase provide a well-structured foundation that is easy to maintain and extend. Functional evaluation across 24 test cases confirmed complete module coverage with all tests passing. The system's dashboard, combining four real-time KPI counters with two Chart.js pie charts, provides management-level insight into parking occupancy and category distribution at a glance.

Key limitations — notably SQL injection vulnerability and weak MD5 password hashing — have been explicitly identified, and a prioritised enhancement roadmap has been proposed. The findings of this work establish VPMS as a viable, cost-effective digital alternative to manual parking management for small to medium parking facilities, and as a practical reference implementation for web-based information system development in academic and research contexts.

ACKNOWLEDGEMENT

The authors would like to thank the Department of Computer Science and Engineering, Sri Venkateswara College of Engineering, Chennai, for providing the computational resources and infrastructure necessary to carry out this work. The authors also acknowledge the open-source communities behind PHP, MySQL, Bootstrap, Chart.js, and jQuery DataTables, whose freely available tools made this implementation possible.

REFERENCES

[1] International Organization of Motor Vehicle Manufacturers (OICA), "World Vehicle

Population," OICA Statistics, 2023. [Online]. Available: https://www.oica.net/

[2] V. W. S. Tang, Y. Zheng, and J. Cao, "An Intelligent Car Park Management System Based on Wireless Sensor Networks," in Proc. 1st Int. Symp. Pervasive Computing and Applications, Urumchi, 2006, pp. 65–70.

[3] M. Ibrahim, M. Ahmad, "A Smart Parking System Using IoT and Mobile Computing," International Journal of Advanced Computer Science and Applications, vol. 11, no. 4, pp. 102–110, 2020.

[4] L. Mainetti, L. Patrono, M. L. Stefanizzi, and R. Vergallo, "A Smart Parking System Based on IoT Protocols and New Energy-Saving Management Algorithms," in Proc. IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, 2015, pp. 764–769.

[5] T. N. Pham, M. F. Tsai, D. B. Nguyen, C. R. Dow, and D. J. Deng, "A Cloud-Based Smart-Parking System Based on Internet-of-Things Technologies," IEEE Access, vol. 3, pp. 1581–1591, 2015.

[6] M. Y. I. Idris, Y. Y. Leng, E. M. Tamil, N. M. Noor, and Z. Razak, "Car Park System: A Review of Smart Parking System and Its Technology," Information Technology Journal, vol. 8, no. 2, pp. 101–113, 2009.

[7] A. Khanna and R. Anand, "IoT Based Smart Parking System," in Proc. Int. Conf. Internet of Things and Applications (IOTA), Pune, 2016, pp. 266–270.

[8] A. Abubakar, B. Sani, and A. Mohammed, "Development of a Web-Based Vehicle Parking Management System," International Journal of Computer Applications, vol. 180, no. 18, pp. 1–6, 2018.

[9] S. Nawaz, C. Efstratiou, and C. Mascolo, "ParkSense: A Smartphone Based Sensing System for On-Street Parking," in Proc. 19th ACM Annual International Conference on Mobile Computing and Networking (MobiCom), Miami, 2013, pp. 75–86.

[10] R. Lu, X. Lin, H. Zhu, and X. Shen, "SPARK: A New VANET-Based Smart Parking Scheme for Large Parking Lots," in Proc. IEEE INFOCOM, Rio de Janeiro, 2009, pp. 1413–1421.

[11] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin, "Parking Slot Assignment Games," in Proc. 19th ACM SIGSPATIAL Int. Conf. Advances in Geographic Information Systems, Chicago, 2011, pp. 299–308.

[12] PHP Group, "PHP: Hypertext Preprocessor Manual," 2023. [Online]. Available: https://www.php.net/manual/en/

[13] Oracle Corporation, "MySQL 5.6 Reference Manual," 2023. [Online]. Available: https://dev.mysql.com/doc/refman/5.6/en/

[14] Twitter, Inc., "Bootstrap 3 Documentation," 2023. [Online]. Available: https://getbootstrap.com/docs/3.4/

[15] Chart.js Contributors, "Chart.js v2.2.2 Documentation," 2016. [Online]. Available: https://www.chartjs.org/docs/2.2.2/

[16] OWASP Foundation, "SQL Injection Prevention Cheat Sheet," 2023. [Online]. Available: https://cheatsheetseries.owasp.org/

[17] L. Welling and L. Thomson, PHP and MySQL Web Development, 5th ed. Upper Saddle River, NJ, USA: Addison-Wesley, 2016.

[18] R. Nixon, Learning PHP, MySQL & JavaScript, 5th ed. Sebastopol, CA, USA: O'Reilly Media, 2018.

