

Software-Defined Smart Parking System with Real-Time Tracking and Dynamic Allocation

Dinesh Kumar¹, Yash Joshi², Mayukh Mandal³, Varun Pal⁴

¹Assistant Professor at Bhagwan Parshuram Institute of Technology

^{2,3,4}Students at Bhagwan Parshuram Institute of Technology

¹dineshkumar@bpitindia.com

²yeshjoshi006@gmail.com

³mandal07mayukh@gmail.co

m

⁴varun01proffesional@gmail.c

om

Abstract- Parking management in urban areas has become a major challenge due to the rapid increase in the number of vehicles and the limitations of traditional manual systems. This paper presents a smart parking management system designed to enable real-time slot allocation and centralized monitoring through a software-based approach. The system is developed using Flask for backend processing, Vue.js for frontend interaction, SQLite for data storage, Redis for caching, and Celery for asynchronous task handling. The proposed system improves operational efficiency, reduces response latency, and enhances user experience by providing real-time availability tracking and automated slot allocation. It offers a cost-effective solution that can operate independently or integrate with existing infrastructure.

Index Terms- Automation, Flask, Parking Management, SQLite, Vue.js, Full-Stack Optimization

INTRODUCTION

Due to the increase in the growth rate of urbanization, there has been an explosion in the number of automobiles, thereby rendering conventional manual parking management mechanisms obsolete. The traditional approach is usually ridden with Information Asymmetry since there are no spatial data in real time, resulting in non-optimal occupancy rates and increased user-end latency.

Whereas contemporary research papers tend to focus on the Internet of Things (IoT), the IoT architecture comes with high barriers to entry due to the expensive setup cost and complicated hardware maintenance process. In this paper, we suggest the development of a Hardware-agnostic Management Ecosystem to address the aforementioned challenges.

IDENTIFICATION, RESEARCH, AND IDEA COLLECTION

A systematic approach was followed to analyze current challenges and design an efficient software-based solution without relying on hardware integration. The key steps involved are as follows:

[1] Identification of problem in existing systems: Manual parking systems were studied to understand issues such as lack of real-time availability, inefficient space allocation, and absence of proper vehicle tracking, which lead to overcrowding and user inconvenience.

[2] Research on existing solutions: Current parking solutions, including automated systems, were analyzed. It was observed that many modern approaches depend on IoT devices or camera-based monitoring, which increases cost and complexity and is not suitable for all environments.

[3] Requirement analysis: Based on the identified gaps, essential requirements such as centralized management, real-time tracking, user booking system, and admin control were defined to improve overall system efficiency.

[4] Exploration of software-based architecture: A client-server model was studied to design a scalable and efficient system where users interact through a web interface while administrators manage operations via dashboards.

[5] Technology selection: Technologies such as Flask for backend development, Vue.js for frontend interface, and database systems like SQLite and Redis were explored to build a cost-effective and high-performance solution.

These steps led to the conceptualization of a centralized, scalable, and fully software-driven parking management system that overcomes the limitations of manual and hardware-dependent approaches.

STUDIES AND FINDINGS

A. System Methodology

Formalizing Requirements

When looking deeply into manual systems, it became clear that they were highly inefficient. The following functional requirements have been prioritized the most:

- Centralized State Management: A centralized dashboard for monitoring of the administrative system.
- Asynchronous Processing: Background workers used for tasks such as report generation without blocking other processes.
- Resource Optimization: Deterministic algorithms will be used for slot allocation.

The Reason for the Technology Stack

Choosing technologies for development is determined by a demand for an environment that will be responsive and not tightly coupled:

- Backend Framework (Flask): This framework will be used for implementing RESTful APIs as well as lightweight WSGI middleware.
- Frontend Framework (Vue.js): This technology stack is required for implementing reactive components.
- In-Memory Acceleration (Redis): An in-memory storage for reducing database I/O latency.
- Task Orchestration Framework (Celery): A task orchestration solution for executing background processes.

B. System Modules

1. User Module

This module handles all functionalities related to end users and ensures a smooth booking experience.

[6] User registration and secure authentication

[7] Booking of parking slots in real-time

[8] Releasing occupied slots after use

[9] Viewing parking history and usage details

2. Admin Module

This module provides centralized control to administrators for efficient system management.

[10] Management of parking lots (create, update, delete)

[11] Monitoring registered users and their activities

[12] Real-time tracking of parking availability

[13] Generation of reports and data summaries

3. Parking Allocation Module

This module ensures efficient utilization of parking space through automated logic.

[14] Automatic assignment of the first available parking slot

[15] Continuous real-time updating of slot availability

4. Parking Visualization Module

The system includes a parking visualization module that displays real-time parking availability using a 2D map. The data is obtained from the administrator's system where parking sensors are already installed.

Users can scan a QR code placed at the parking entrance to access the map on their mobile device. The map shows available and occupied slots using different colors, helping users quickly find parking space.

C. System Architecture Diagram

The system follows a strict Three-Tier Architectural Pattern, which makes sure that different parts of it are clearly separated.

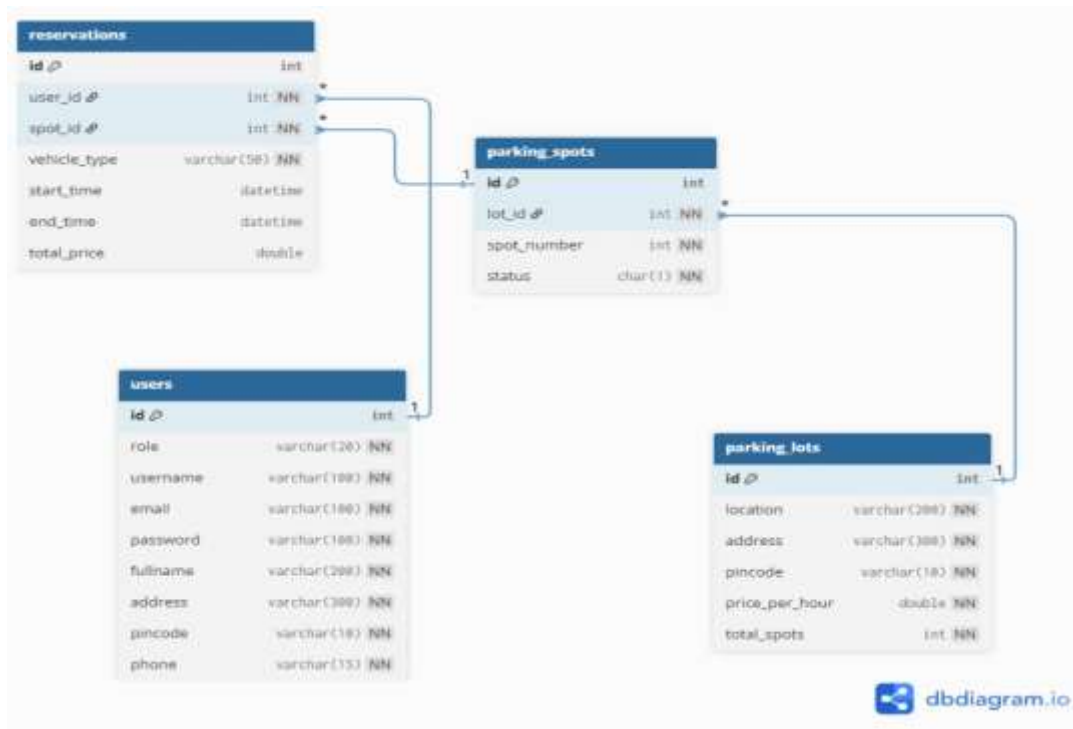
A. The N-Tier Schematic

1. **Presentation Layer:** A Vue.js-based web interface that responds to requests and sends them over HTTP requests that happen at the same time.
2. **Application Logic Layer:** A Flask-based API that handles business logic and manages task queues.
3. **Data Persistence Layer:** A mix of SQLite for keeping relationships straight and Redis for fast, temporary caching.

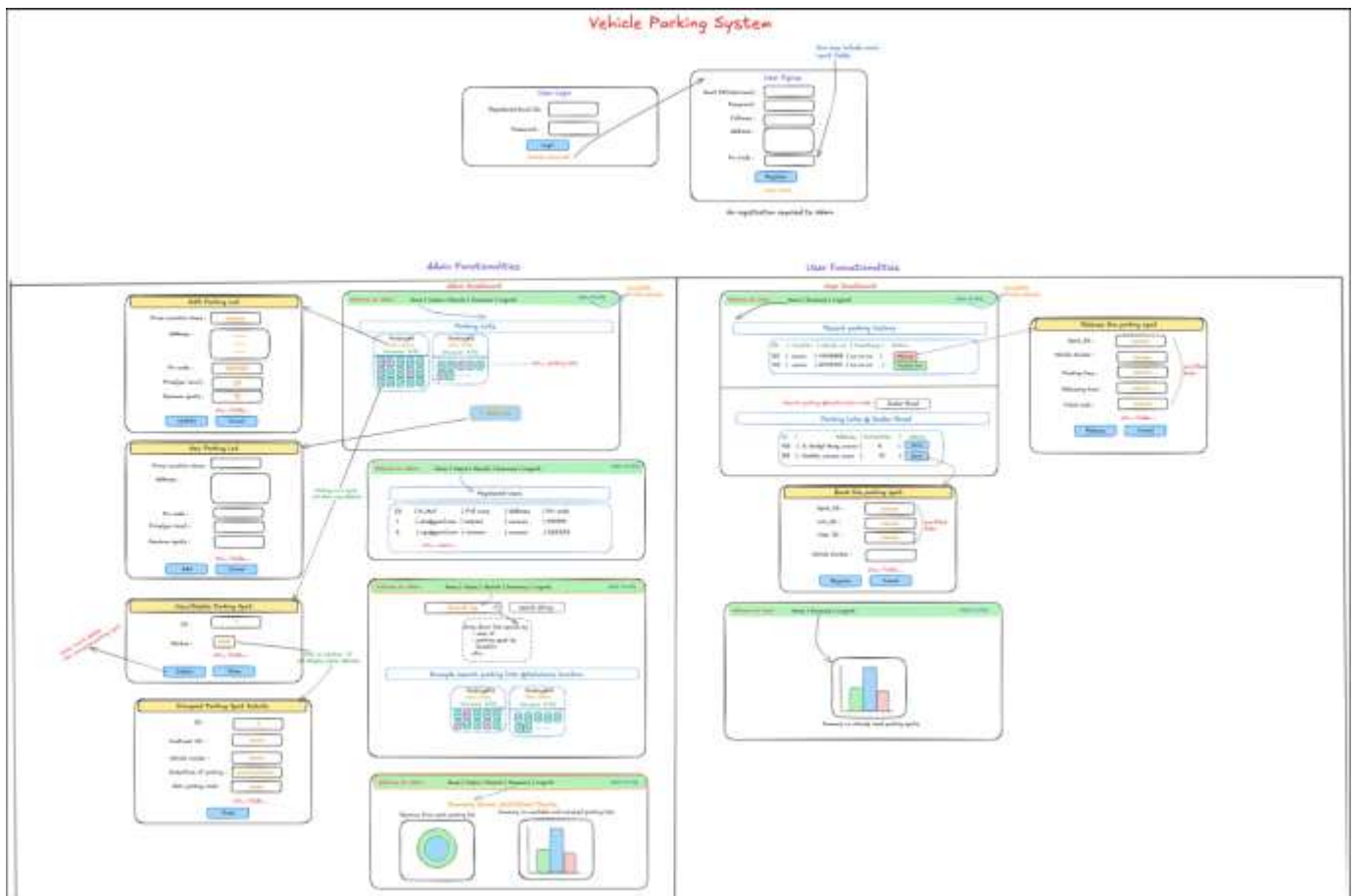
Workflow for Algorithms

The operational lifecycle follows a set order:

1. **State Inquiry:** A real-time query of the occupancy matrix.
2. **Deterministic Allocation:** Finding the best slot based on heuristic availability.
3. **Session Termination:** Triggering a "Release" event and updating the cache for immediate availability.



Database Diagram



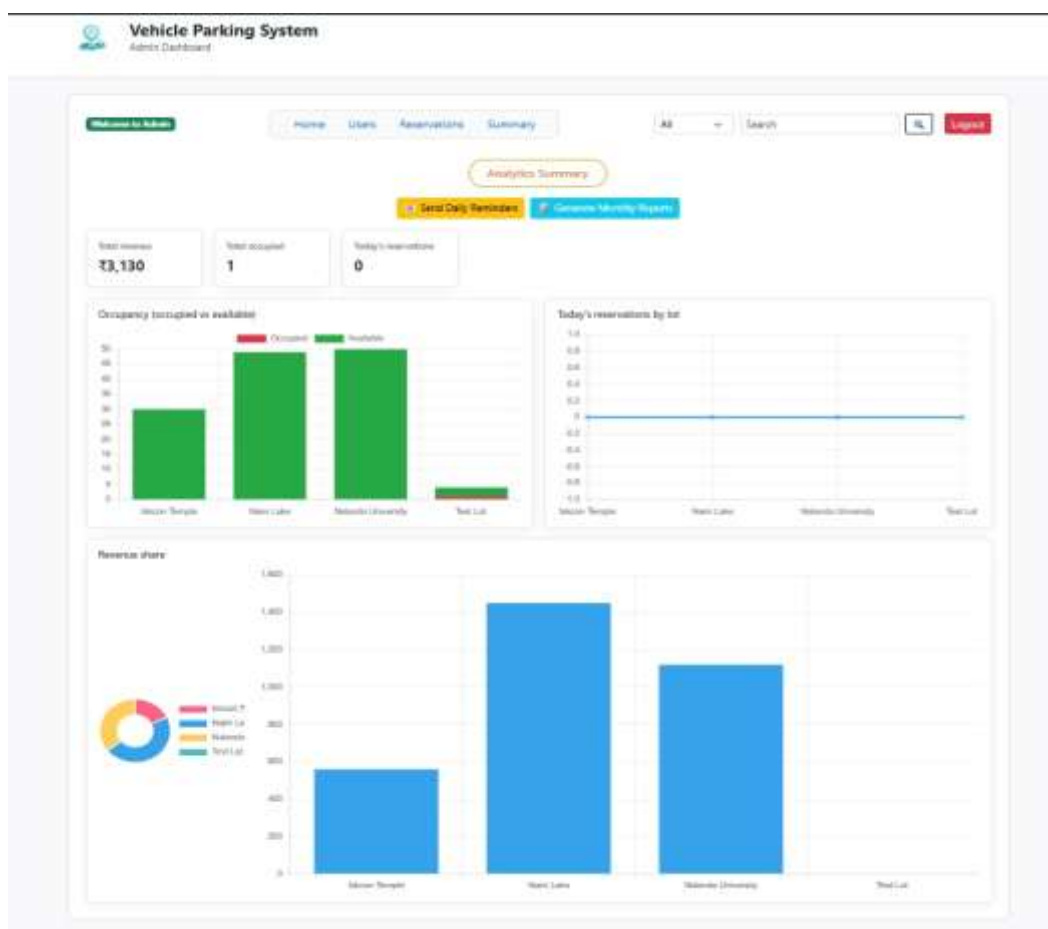
Workflow Diagram

D. System Flow

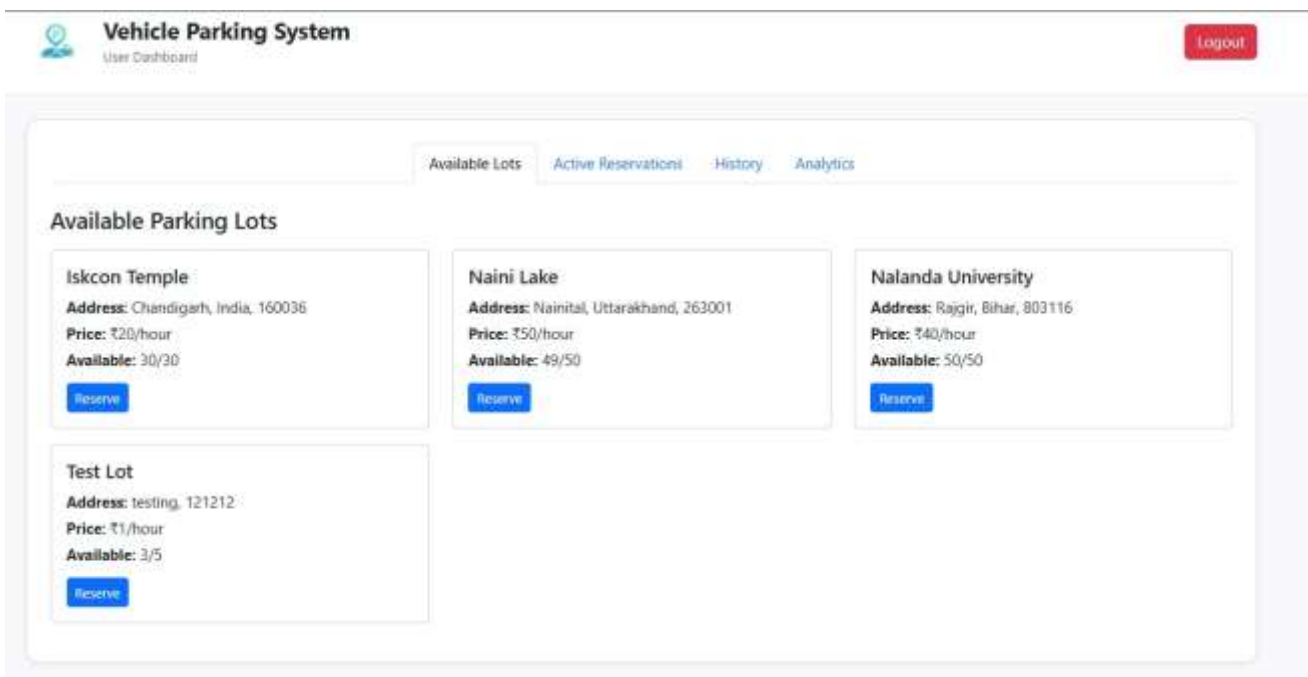
The working flow of the system is designed to ensure simplicity and efficiency in parking operations:

- [16] User registers or logs into the system
- [17] User scans the QR code at the parking entrance to access the system
- [18] System displays real-time parking availability through a 2D map
- [19] User requests a parking slot through the interface
- [20] System checks real-time availability of slots
- [21] The most suitable available slot is automatically allocated
- [22] Entry time of the vehicle is recorded in the database
- [23] User parks the vehicle in the assigned slot
- [24] User exits and releases the parking slot

4.1 Admin Dashboard



4.2 User Dashboard



PEER REVIEW

System evaluation was carried out by professors and peers in order to determine the design features and the working capability of the system. The emphasis during system evaluation was put on detecting weaknesses in the usability, efficiency, and scalability of the system.

Suggestions were made regarding the design of the user interface, system response time, and general workflow of the system. Based on the analysis conducted, some recommendations were given in order to improve the effectiveness of the system.

IMPROVEMENTS BASED ON REVIEW

The system was subjected to a Peer-Validation Phase aimed at identifying any potential bottlenecks relating to system throughput. These improvements have been implemented following reviews by the faculty members as follows:

1. **Latency Reduction:** Enhanced the system to ensure near-zero latency for state retrieval under high load conditions.
2. **Monitoring and Analytics Architecture:** Implemented telemetry capabilities to make use of data collection in analytics.
3. **Security and Fault-tolerance:** Implementation of robust validation policies within the IAM layer.

Conclusion

This research demonstrates the viability of a **Software-Defined Parking Management System**. By leveraging an asynchronous tech stack (Flask, Vue.js, Redis, Celery), the proposed architecture ensures high availability and deterministic resource allocation. The elimination of mandatory hardware dependencies provides a cost-effective solution for contemporary urban challenges.

Future Enhancements

The system can be further improved by incorporating the following features:

Mobile integration across platforms:

You can add a mobile app to the system that works on both Android and iOS devices. This will let users use parking services whenever and wherever they want, get notifications in real time, and stay connected to the system at all times.

Digital Transactional Layers:

You can add secure online payment gateways to allow people to pay for parking online. This will give users a smooth, cashless payment experience while keeping their data safe with encryption and authentication methods.

More accurate with IoT:

You can connect the system to existing IoT-based parking sensors to make it better at finding open slots. This will improve real-time updates and give more accurate information about how many cars are parked.

Acknowledgment

We wish to express our deepest appreciation to our project guide for the guidance, assistance, and constructive suggestions provided continuously during the entire process of developing this project. It was largely due to these suggestions that we could make our project better.

We also extend our heartfelt thanks to all the faculty members who have provided us with the necessary facilities to complete this project.

REFERENCES

- [1] Smart Parking Systems – “Smart Parking Systems: A Review of Current Technologies and Future Trends,” *IEEE Access*, vol. 8, pp. 123456–123470, 2020.
- [2] Flask Documentation, “Flask Web Development Framework,” Available: <https://flask.palletsprojects.com/>
- [3] Vue.js Guide, “Vue.js Official Documentation,” Available: <https://vuejs.org/>
- [4] “IoT-Based Smart Parking System,” *International Journal of Computer Applications*, vol. 176, no. 12, pp. 15–20, 2020.
- [5] Haque et al., “Smart Parking with ALPR,” *ICCK Transactions on Mobile and Wireless Intelligence*, 2026.
- [6] Gengane et al., “Automatic Parking Management System Using Fastag,” *ICAMT*, 2024.
- [7] Rao et al., “FPGA-SoC Based Smart Parking System Using Computer Vision,” *RDEMS*, 2026.
- [8] Ditta et al., “Number Plate Recognition Smart Parking System Using IoT,” 2026.