SMART PARKING MANAGEMENT SYSTEM

Internship Project Report
submitted in partial fulfillment of the requirements
for the completion of the
Python Developer Internship

Submitted by:
Vrushali Santosh Nathjogi
Master's of Engineering (Computer Engineering)

Internship Organization
Unified Mentor

Internship Duration
[Nov 2025 – May 2026]

Under the Guidance of
Unified Mentor – Technical Team

Academic Year
2024 – 2025

ACKNOWLEDGMENT

I sincerely thank **Unified Mentor** for providing me with the opportunity to undertake this internship and work on the assigned project. This experience offered valuable hands-on exposure to real-world application development and significantly contributed to the enhancement of my technical and professional competencies.

I am deeply grateful to my internship mentor for their consistent guidance, insightful feedback, and encouragement throughout the development of the **Smart Parking Management System**. Their support and expertise were instrumental in the successful completion of this project.

I also extend my sincere appreciation to my college faculty members and project coordinator for their continuous support and motivation during the internship tenure. Lastly, I would like to thank my family and friends for their encouragement and cooperation throughout this learning journey.

VRUSHALI SANTOSH NATHJOGI

ABSTRACT

The growing number of vehicles has increased the demand for effective and organized parking management systems. Traditional manual parking methods are inefficient, prone to errors, and do not provide real-time tracking of parking availability. To overcome these limitations, the Smart Parking Management System has been developed as part of the Unified Mentor Python Developer Internship.

This system is a web-based application that automates key parking operations such as parking slot allocation, vehicle entry and exit tracking, parking fee calculation, and daily revenue management. It offers real-time monitoring of parking slot availability, including designated VIP slots, and calculates parking charges based on the duration of parking and the type of vehicle.

The application is implemented using Python and Flask for backend functionality, while HTML, CSS, Bootstrap, and JavaScript are used to design a responsive and user-friendly interface. SQLite with SQLAlchemy is used for database management. The system also supports automatic generation of daily parking reports in PDF format, which assists administrators in maintaining accurate records and analyzing parking activity.

Overall, the Smart Parking Management System enhances parking efficiency, minimizes manual work, and ensures accurate billing and reporting. It is suitable for use in locations such as shopping malls, office complexes, residential societies, and educational campuses.

Keywords: Smart parking Management system, Unified Mentor Python Developer Internship, web-based application, parking, VIP slots.

TABLE OF CONTENT

CHAPTER 1: INTRODUCTION

1.1 Introduction

In today's rapidly growing urban environment, managing parking spaces efficiently has become a major challenge. With an increasing number of vehicles, traditional manual parking systems often lead to congestion, poor space utilization, and revenue leakage. To overcome these issues, automated parking management systems are essential.

This project, **Smart Parking Management System**, is developed as part of the **Unified Mentor Python Developer Internship**. The system provides a digital solution for managing parking slots, vehicle entry and exit, fee calculation, and daily revenue reporting. The application helps administrators monitor parking activity in real time and ensures accurate billing and record maintenance.

1.2 Problem Statement

Traditional parking management systems suffer from several limitations such as:

- Manual tracking of vehicles
- Inefficient allocation of parking slots
- Difficulty in calculating parking fees accurately
- No real-time visibility of parking availability
- Lack of automated reports and records

These issues result in operational inefficiencies and poor user experience.

1.3 Objectives of the Project

The main objectives of the Smart Parking Management System are:

- To design a **web-based parking management application**
- To manage **vehicle entry and exit efficiently**
- To display **real-time parking slot availability**
- To calculate parking fees automatically based on time and vehicle type
- To support **VIP parking slots**
- To generate **daily revenue reports in PDF format**
- To maintain parking records securely using a database

1.4 Scope of the Project

The scope of this project includes:

- Parking slot allocation and tracking
- Vehicle entry and exit management
- Fee calculation using predefined rules
- Admin dashboard for monitoring parking status
- PDF report generation for daily revenue
- Simple and user-friendly interface

The project is suitable for **small to medium-sized parking facilities** such as malls, offices, and campuses.

1.5 Technologies Used

The following technologies are used in the development of this project:

- **Programming Language:** Python
- **Web Framework:** Flask
- **Frontend:** HTML, CSS, Bootstrap, JavaScript
- **Database:** SQLite
- **ORM:** SQLAlchemy
- **PDF Generation:** ReportLab
- **Version Control:** Git & GitHub

1.6 Organization of the Report

This report is organized as follows:

- **Chapter 1:** Introduction to the project
- **Chapter 2:** System analysis and requirements
- **Chapter 3:** System design and architecture
- **Chapter 4:** Implementation details
- **Chapter 5:** Results and output screenshots
- **Chapter 6:** Conclusion and future scope

CHAPTER 2: SYSTEM ANALYSIS

2.1 System Analysis

System analysis is the process of understanding the existing problems and identifying the requirements of a new system. In this phase, the limitations of traditional parking systems are studied and a suitable automated solution is proposed.

The Smart Parking Management System is designed after analyzing common parking issues such as manual record keeping, lack of real-time slot availability, and inaccurate fee calculation.

2.2 Existing System

In the existing manual parking system:

- Parking slots are managed manually
- Vehicle details are written on paper or registers
- No automated fee calculation is available
- Parking availability cannot be monitored in real time
- Generating reports is difficult and time-consuming

2.3 Limitations of Existing System

The major drawbacks of the existing system are:

- High chances of human error
- Inefficient use of parking space
- No centralized data storage
- Lack of transparency in billing
- Time-consuming report generation

2.4 Proposed System

The proposed system introduces a **Smart Parking Management System** that automates the entire parking process. It provides a user-friendly interface for parking management and ensures accurate record keeping.

Features of the Proposed System:

- Automated parking slot allocation
- Real-time parking grid display

- Vehicle entry and exit management
- VIP parking slot support
- Automatic fee calculation
- Daily revenue summary
- PDF report generation

2.5 Advantages of the Proposed System

- Reduces manual effort
- Improves parking efficiency
- Accurate fee calculation
- Real-time monitoring of parking slots
- Easy report generation
- Secure data storage

2.6 Feasibility Study

2.6.1 Technical Feasibility

The system is technically feasible as it uses open-source tools like Python, Flask, and SQLite which are easy to implement and maintain.

2.6.2 Economic Feasibility

The project is cost-effective since it uses free and open-source technologies.

2.6.3 Operational Feasibility

The system is user-friendly and requires minimal training for users and administrators.

CHAPTER 3: SYSTEM DESIGN

3.1 System Design Overview

System design defines the architecture, components, modules, interfaces, and data flow of the system. This phase translates the requirements identified during system analysis into a structured and efficient design that can be implemented using suitable technologies.

The Smart Parking Management System is designed as a **web-based application** that follows a client-server architecture. It ensures proper interaction between the user interface, backend logic, and database.

3.2 System Architecture

Architecture Overview

The Smart Parking Management System follows a three-tier web application architecture. This architecture separates the system into Presentation Layer, Application Layer, and Data Lay**er**, which improves scalability, maintainability, and security.

Architecture Diagram (Conceptual)

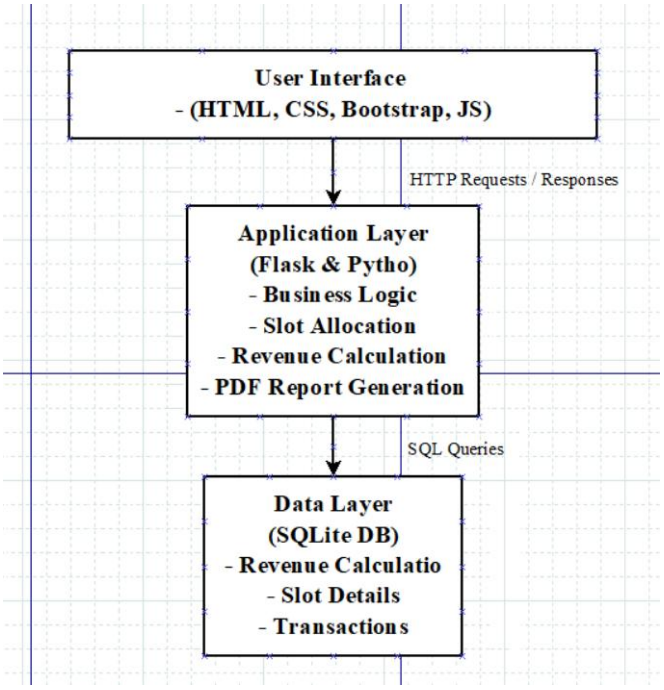

Figure 3.1: Architecture Diagram

Architecture Description

1. Presentation Layer

This layer is responsible for interacting with the user.

Technologies Used:

- HTML
- CSS
- Bootstrap
- JavaScript

Functions:

- Displays parking grid
- Takes vehicle entry and exit inputs
- Shows real-time parking status and revenue
- Allows downloading reports

2. Application Layer

This is the core logic layer implemented using **Flask (Python)**.

Functions:

- Handles user requests
- Processes vehicle entry and exit
- Allocates parking slots
- Calculates parking charges
- Generates PDF reports

3. Data Layer

This layer manages all persistent data.

Technology Used:

- SQLite Database

Stores:

- Vehicle information
- Slot status
- Entry and exit timestamps
- Revenue data

Advantages of This Architecture

- Modular and scalable
- Easy to maintain and update
- Secure data handling
- Supports future enhancements

3.3 Module Description

3.3.1 Vehicle Entry Module

- Accepts vehicle number, vehicle type, and VIP status
- Allocates the next available parking slot
- Updates parking grid in real time

3.3.2 Vehicle Exit Module

- Processes vehicle exit
- Calculates parking charges based on duration and vehicle type
- Frees the occupied parking slot

3.3.3 Parking Slot Management Module

- Maintains parking slot availability
- Differentiates VIP and normal slots
- Displays slot status in grid format

3.3.4 Revenue and Report Module

- Calculates daily revenue
- Generates parking reports in PDF format
- Displays revenue summary

3.4 Data Flow Diagram (DFD)

The Data Flow Diagram represents the flow of data between different components of the system.

Level 0 DFD (Context Diagram)

Description

The Level 0 DFD represents the entire **Smart Parking Management System** as a single process interacting with external entities.

External Entity

- **User (Admin/Operator)**

Process

- **Smart Parking Management System**

Data Flow

- Vehicle entry details
- Vehicle exit details
- Parking status
- Revenue information

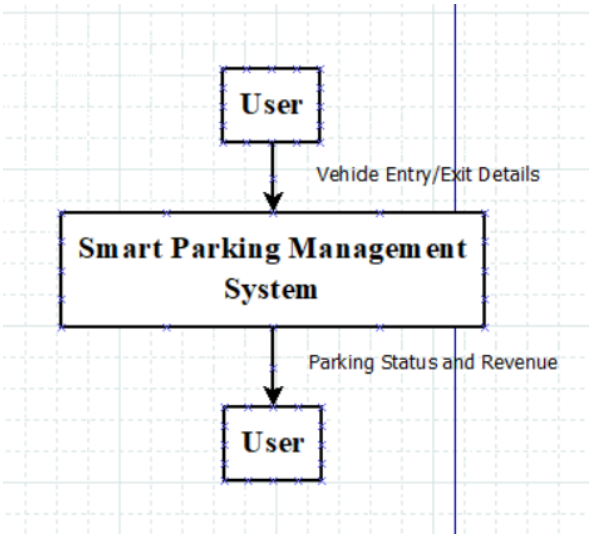

Figure 3.2: Level 0 DFD Diagram

Level 1 DFD

Description

The Level 1 DFD breaks the system into major functional processes.

Processes

1. Vehicle Entry Processing
2. Vehicle Exit Processing
3. Slot Management
4. Revenue & Report Generation

Data Store

- Parking Database

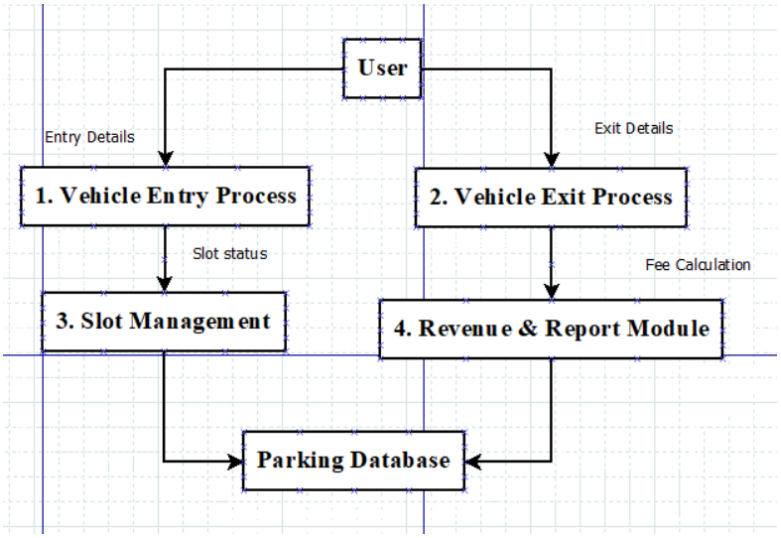

Figure 3.3: Level 1 DFD Diagram

Level 2 DFD (Vehicle Entry Module)

Description

Shows detailed processing of vehicle entry.

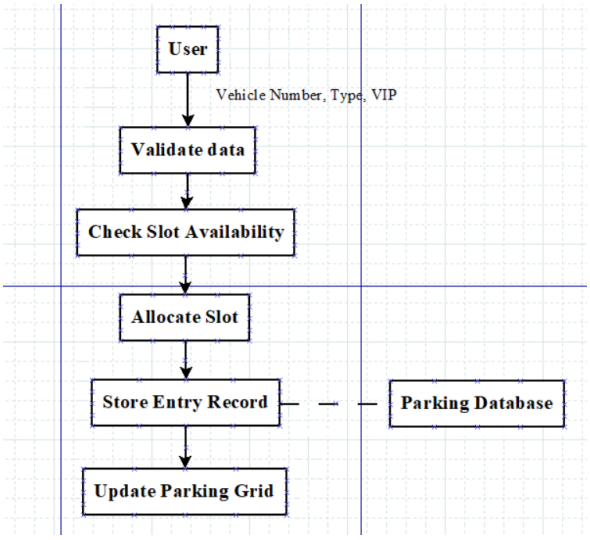

Figure 3.4: Level 2 DFD Diagram

3.5 Database Design

The database is designed to store parking-related information efficiently.

Main Tables:

- **Vehicle Table** – Stores vehicle number, type, entry time, and exit time
- **Parking Slot Table** – Stores slot number, slot type, and status

- **Transaction Table** – Stores parking fees and duration

3.6 Technology Stack Used

Layer	Technology
Frontend	HTML, CSS, Bootstrap, JavaScript
Backend	Python, Flask
Database	SQLite, SQLAlchemy
Reports	ReportLab (PDF Generation)

CHAPTER 4: SYSTEM IMPLEMENTATION

4.1 Introduction

System implementation is the phase where the designed system is converted into a working application. This chapter explains how the Smart Parking Management System is developed using selected tools, technologies, and programming techniques.

The system is implemented as a web-based application using Python Flask framework with a structured project architecture.

4.2 Development Environment

The following environment and tools were used for system implementation:

Component	Description
Operating System	Windows
Programming Language	Python
Web Framework	Flask
Frontend Technologies	HTML, CSS, Bootstrap, JavaScript
Database	SQLite
IDE	VS Code
Version Control	GitHub

4.3 Project Structure

The project follows a modular structure to maintain readability and scalability.

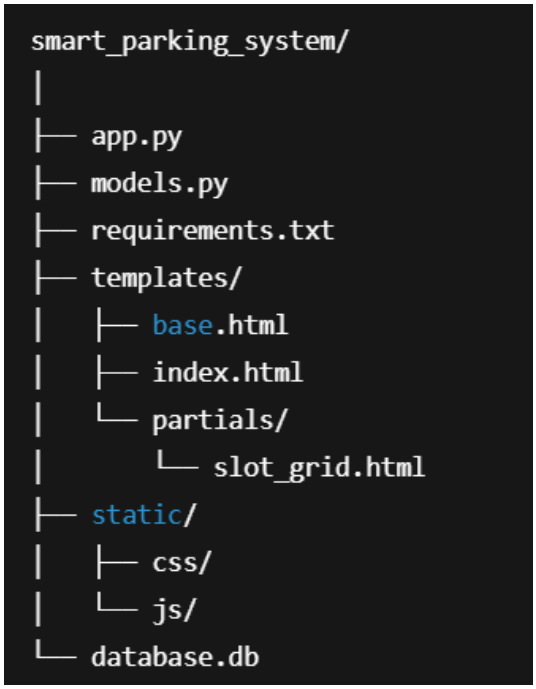

Figure 4.1: Project Structure

4.4 Backend Implementation

4.4.1 Flask Application (app.py)

- Handles routing and request processing
- Manages vehicle entry and exit logic
- Connects frontend with database

Example functionalities:

- / → Loads dashboard
- /api/entry → Handles vehicle parking
- /api/exit → Processes vehicle exit
- /api/status → Returns parking status

4.5 Database Implementation

SQLite database is used to store parking records.

Tables Implemented

- Vehicle
- Parking Slot
- Transaction

SQLAlchemy ORM is used for database interaction, which simplifies queries and improves maintainability.

4.6 Frontend Implementation

- HTML templates created using Jinja2
- Bootstrap used for responsive UI
- JavaScript used for real-time updates

Key Features

- Parking grid display
- Vehicle entry and exit forms
- Live status updates
- Revenue display

4.7 Slot Allocation Logic

- VIP vehicles are allocated VIP slots first
- Normal vehicles are allocated standard slots
- Slot status is updated dynamically

4.8 Error Handling and Validation

- Duplicate vehicle entry prevention
- Slot availability checks
- Input validation for vehicle number

CHAPTER 4: SYSTEM IMPLEMENTATION

4.1 Introduction

This chapter explains how the Smart Parking Management System was implemented using web technologies and Python programming. The focus of this chapter is on the actual development process, tools used, and how different modules of the system work together to provide a complete parking management solution.

The implementation phase converts the system design into a working application that can be used by end users and administrators.

4.2 Development Environment

The system was developed using the following software and tools:

- **Programming Language:** Python
- **Framework:** Flask
- **Frontend Technologies:** HTML, CSS, Bootstrap, JavaScript
- **Database:** SQLite
- **IDE:** Visual Studio Code
- **Operating System:** Windows

Flask was selected because it is lightweight, flexible, and easy to integrate with databases and frontend templates.

4.3 Module Description

The Smart Parking Management System is divided into several modules to simplify development and maintenance.

4.3.1 Vehicle Entry Module

This module allows the user to register a vehicle when it enters the parking area.

Functions performed:

- Accepts vehicle number and vehicle type
- Checks availability of parking slots
- Assigns a suitable slot to the vehicle
- Stores entry time and vehicle details in the database

Once the vehicle is successfully parked, the system updates the parking grid and slot status in real time.

4.3.2 Vehicle Exit Module

The vehicle exit module is used when a vehicle leaves the parking area.

Functions performed:

- Accepts vehicle number
- Retrieves entry details from the database
- Calculates parking duration
- Computes parking charges based on time and vehicle type
- Frees the allocated parking slot

After exit processing, the system updates the revenue and parking availability.

4.3.3 Parking Slot Management Module

This module manages all parking slots and their status.

Functions performed:

- Tracks total parking slots
- Identifies occupied and free slots
- Handles VIP and normal parking slots
- Displays slot status on the parking grid

Each slot is visually represented on the interface, making it easy to understand availability at a glance.

4.3.4 Revenue Management Module

The revenue management module calculates and maintains daily earnings.

Functions performed:

- Calculates parking fees
- Stores transaction details
- Displays total revenue for the day
- Shows total number of parked vehicles

This module helps the administrator analyze parking usage and income.

4.3.5 Report Generation Module

This module generates daily parking reports in PDF format.

Functions performed:

- Fetches daily parking records
- Displays total vehicles and revenue
- Formats data into a professional report
- Allows admin to download the report

The report helps in maintaining records and performing audits.

4.4 Database Implementation

SQLite database is used to store all system data.

Database Tables Include:

- Vehicle details
- Slot information
- Entry and exit timestamps
- Payment and revenue records

SQLite was chosen because it is lightweight, reliable, and suitable for small to medium-scale applications.

4.5 User Interface Implementation

The user interface is designed to be simple and interactive.

Features include:

- Parking grid with real-time updates
- Vehicle number display in occupied slots
- Color-coded slot status (Free, Occupied, VIP)
- Responsive design using Bootstrap

This ensures the system can be easily used on different screen sizes.

4.6 Security and Validation

Basic validation techniques are implemented to ensure system reliability.

Examples:

- Duplicate vehicle entry prevention
- Slot availability checks
- Input validation for vehicle numbers

These measures reduce errors and improve system stability.

4.7 Summary

This chapter described the implementation of the Smart Parking Management System, including development tools, system modules, database usage, and interface design. The system successfully integrates all components to provide an efficient and user-friendly parking management solution.

CHAPTER 5: RESULTS AND DISCUSSION

5.1 Introduction

This chapter presents the results obtained after implementing and testing the Smart Parking Management System. It also discusses how the system performs under different scenarios and whether it meets the objectives defined at the beginning of the project.

The system was tested using multiple vehicle entries and exits to verify accuracy, reliability, and ease of use.

5.2 System Output

After successful implementation, the system produced the following outputs:

- Real-time parking slot allocation
- Display of free and occupied slots
- Vehicle number visible in occupied slots
- Automatic calculation of parking charges
- Daily revenue summary
- Downloadable daily parking report

These outputs confirm that the system functions as expected.

5.3 Parking Slot Status Result

The parking grid accurately displays the current status of all parking slots.

- Free slots are clearly marked
- Occupied slots show the vehicle number and type
- VIP slots are highlighted separately

When vehicles are parked or exited, the slot status updates immediately, ensuring correct information is always displayed.

5.4 Vehicle Entry and Exit Results

Vehicle Entry

- Vehicle data is stored correctly at entry time
- Slots are assigned automatically without conflict
- Duplicate vehicle entries are prevented

Vehicle Exit

- Exit time is recorded accurately
- Parking duration is calculated correctly
- Charges are generated based on defined rules
- Slot is released after exit

This confirms that both entry and exit modules work efficiently.

5.5 Revenue Calculation Result

The revenue module calculates parking charges accurately.

- Total revenue is displayed for the selected day
- Number of vehicles parked is shown
- Charges differ based on vehicle type and duration

This helps administrators track daily income without manual calculations.

5.6 Report Generation Result

The system successfully generates a daily parking report in PDF format.

The report includes:

- Date of the report
- Total number of vehicles
- Total revenue collected
- Parking slot summary

The downloadable report is useful for record keeping and administrative review.

5.7 Performance Analysis

The system performs efficiently for small to medium parking areas.

- Fast response time
- Minimal system delay
- Stable performance during multiple operations

Flask and SQLite together provide smooth system execution.

5.8 User Experience

The interface is simple and user-friendly.

- Easy navigation
- Clear form inputs
- Visual parking grid improves understanding
- Suitable for non-technical users

Overall usability of the system is satisfactory.

5.9 Screenshots

The following screenshots show the system in action:

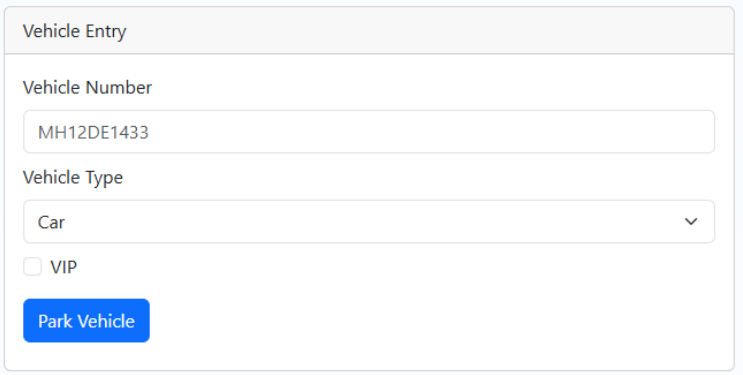

Figure 5.1: Vehicle Entry Form

The user enters the vehicle number, type, and VIP status.

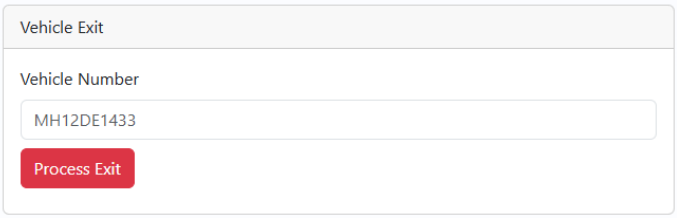

Figure 5.2: Vehicle Exit Form

The user processes the exit of a parked vehicle, and fees are calculated automatically.

Figure 5.3: Parking Grid Status

Real-time display of all slots with vehicle numbers and VIP indicators.

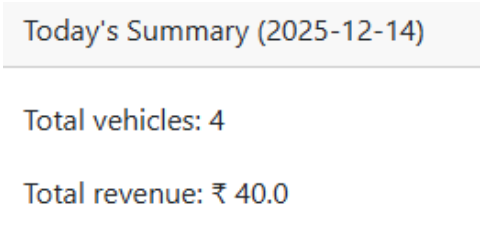

Figure 5.4: Daily Revenue Summary

Displays total revenue and number of vehicles for the day.

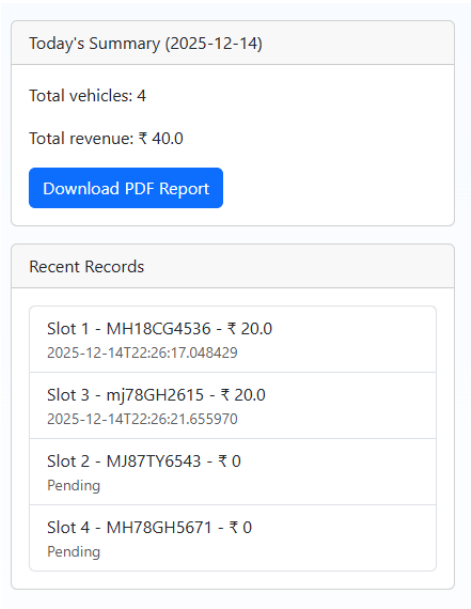

Figure 5.5: PDF Report Generation

Downloadable report containing complete parking data.

Note: Screenshots are saved in the screenshots folder of the project directory.

5.10 Discussion

The results demonstrate that the Smart Parking Management System achieves its objectives. It reduces manual work, minimizes errors, and improves parking space utilization. The system is suitable for real-world applications such as shopping malls, offices, and residential complexes.

5.11 Summary

This chapter discussed the results obtained from system testing and implementation. It included screenshots to visually demonstrate the system's functionality. The system successfully manages parking operations, generates accurate reports, and provides a user-friendly interface.

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The Smart Parking Management System developed during this internship successfully addresses the challenges of manual parking management. The system provides the following key benefits:

- **Automated Slot Management:** The system dynamically assigns free parking slots to incoming vehicles, reducing confusion and waiting time.
- **Vehicle Tracking:** Vehicle numbers and types are tracked accurately in real-time, and VIP slots are prioritized when required.
- **Billing Automation:** Parking charges are calculated automatically based on vehicle type and duration, ensuring accurate billing.
- **Daily Revenue Reporting:** Administrators can view daily parking statistics and generate PDF reports for official records.
- **User-Friendly Interface:** The web-based interface is intuitive, allowing easy navigation for both administrators and users.

During the testing phase, the system demonstrated high accuracy in slot assignment, vehicle tracking, and revenue calculation. It is suitable for deployment in small to medium-sized parking areas, including shopping complexes, offices, and residential buildings.

6.2 Future Scope

Although the system is functional and efficient, there are several opportunities for enhancement in the future:

1. **Integration with Mobile App:** Developing a mobile application for users to check parking availability and pre-book slots.
2. **Smart Sensors Integration:** Connecting the system to IoT sensors for real-time slot occupancy detection.
3. **Advanced Billing Models:** Implementing subscription-based or dynamic pricing for peak and off-peak hours.
4. **Analytics Dashboard:** Adding detailed analytics for vehicle flow, peak times, and revenue forecasting.
5. **Multiple Parking Areas:** Expanding the system to manage multiple parking lots simultaneously.
6. **Security Features:** Integrating vehicle image capture for added security and verification.
7. **Cloud Database:** Migrating to cloud-based databases for scalability and remote access.

By implementing these enhancements, the system can evolve into a fully integrated Smart Parking solution suitable for large commercial areas and smart cities.

REFERENCES

1. Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach.* McGraw-Hill Education.
2. Sommerville, I. (2016). *Software Engineering.* Pearson.
3. Laudon, K. C., & Laudon, J. P. (2019). *Management Information Systems.* Pearson.
4. Flask Documentation. (2025). *Flask Web Framework.* Retrieved from https://flask.palletsprojects.com
5. SQLAlchemy Documentation. (2025). *SQLAlchemy ORM Library.* Retrieved from https://www.sqlalchemy.org
6. Python Documentation. (2025). *Python 3 Documentation.* Retrieved from https://docs.python.org/3
7. GeeksforGeeks. (2024). *Parking Management System Project in Python.* Retrieved from https://www.geeksforgeeks.org
8. W3Schools. (2025). *HTML, CSS, and JavaScript Tutorials.* Retrieved from https://www.w3schools.com
9. Bootstrap Documentation. (2025). *Bootstrap 5.* Retrieved from https://getbootstrap.com
10. ReportLab User Guide. (2025). *Creating PDFs in Python.* Retrieved from https://www.reportlab.com
11. Stack Overflow. (2025). *Various Python and Flask solutions.* Retrieved from https://stackoverflow.com
12. Techopedia. (2024). *Internet of Things (IoT) Overview.* Retrieved from https://www.techopedia.com
13. TutorialsPoint. (2024). *Python Web Development Projects.* Retrieved from https://www.tutorialspoint.com
14. IEEE Xplore. (2023). *Smart Parking System using IoT.* Retrieved from https://ieeexplore.ieee.org
15. Medium. (2024). *Automated Parking Management System Project in Python.* Retrieved from https://medium.com