Parking Slot Prediction and Face Recognition based Parked Vehicle Theft Prevention in Smart Parking System using Deep Learning

Abstract

More than a million cars are on the roadways of a contemporary major city, but more parking spots are needed to accommodate them. Locating vacant parking places in most contemporary cities might take time, especially during busy periods like festival seasons. In the traditional parking system, drivers face considerable losses in terms of money, productivity and time which is wasted in search of parking spots in densely populated areas. Hence, it can be said that the traditional parking systems are not capable of providing a smooth parking experience to the drivers along with reducing the parking search traffic on the roads. This highlights the rationale of adopting advanced technologies to make the urban transport system modern and ease the problem faced by the drivers. This project proposes a Smart Parking System utilizing Edge Computing and Deep Learning algorithms to seamlessly link multiple parking stations into a unified network, establishing a shared parking system. To address security concerns in highly restricted areas such as residential zones, military bases, and government buildings, the system functions as a centralized automatic vehicle identifier for owner verification. Deep Learning algorithms, such as Convolutional Neural Networks used to recognize the driver/owner face of a vehicle during the departure phase, fortifying security measures and thwarting potential vehicle theft. By implementing facial recognition at both entry and exit points, the system ensures that only authorized individuals gain access to their associated vehicles. Transparent communication of access privileges is facilitated through a user interface at the exit gate, allowing drivers to ascertain whether they are granted or denied entry to a specific parking station. The proposed system ensures real-time decision-making, reducing the time spent searching for parking and contributing to the overall efficiency and security of urban parking environments.

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1. Overview

Parking is the act of stopping and disengaging a vehicle and leaving it unoccupied. Parking on one or both sides of a road is often permitted, though sometimes with restrictions. Some buildings have parking facilities for use of the buildings' users. Car parking is essential to car-based travel. Cars are typically stationary around per cent of the time. The availability and price of car parking supports and subsidize car dependency.

A parking space, parking place or parking spot is a location that is designated for parking, either paved or unpaved. It can be in a parking garage, in a parking lot or on a city street. The space may be delineated by road surface markings. Parking facilities can be divided into public parking and private parking.

- Public parking is managed by local government authorities and available for all members of the public to drive to and park in.
- Private parking is owned by a private entity. It may be available for use by the public or restricted to customers, employees or residents.

Such facilities may be on-street parking, located on the street, or off-street parking, located in a parking lot or parking garage.

1.1.1. Types of Parking

Parking comes in various forms, each tailored to specific needs and space requirements. Understanding the different types of parking can help drivers choose the most suitable option for their situation. Here are the main types of parking:

- **Angle Parking**

In this type of parking, cars are parked at an angle. In most cases, the cars face one direction. It is easy to park in and move out of an angular parking setting, provided everyone follows the rules. Since it is easy to simply accelerate and zoom ahead from angle parking, you need to be alert while accelerating. Give the right signals and be on the lookout for signals from fellow drivers.

- **Perpendicular Parking**

This type of parking is common in parking lots, where people park their cars for a longer duration. Such type of parking is like angle parking, but the angle here is perpendicular to the curb ahead. Cars will be parked in a 90-degree angle. You need to ensure that the tires of your car are pointing straight ahead and the car is positioned at the centre of the allocated parking spot in a perpendicular parking area.

- **Parallel Parking**

This type of parking is usually seen on the roads, where cars are parked parallel to the road. Parallel Parking requires a certain amount of skill as it usually needs the driver to park in between two cars – one ahead and one behind. Entering and exiting Parallel Parking needs focus on the surroundings and control on your driving.

- **Illegal Parking**

You need to park your vehicle only in designated areas. Parking your vehicle in spots where parking is prohibited will lead to monetary penalties. Parking cars in No Parking Zones and areas is an example of Illegal Parking.

- **Lot Parking**

If you are parking your car in a parking lot, you need to follow the rules and regulations prescribed by that area. They might have different types of parking in different areas for efficient usage of space.

- **Bay Parking**

Bay parking often involves reversing your car in an allocated area. There will be cars around you or space for cars around you. Therefore, you need to be considerate of them and park accordingly.

- **Parking Between Two Vehicles**

Irrespective of the type of parking, you need to be alert and attentive while parking between two vehicles. One of the most common issues faced when a car is parked between two vehicles is – a dent on the adjoining car’s door while opening your door or a scratch leading to loss of paint.

1.1.2. Ill Effects of Parking

Parking has some ill-effects like congestion, accidents, pollution, obstruction to fire-fighting operations etc.

- **Congestion**: Parking takes considerable street space leading to the lowering of the road capacity. Hence, speed will be reduced, journey time and delay will also subsequently increase. The operational cost of the vehicle increases leading to great economical loss to the community.
- **Accidents**: Careless manoeuvring of parking and unparking leads to accidents which are referred to as parking accidents. Common type of parking accidents occurs while driving out a car from the parking area, careless opening of the doors of parked cars, and while bringing in the vehicle to the parking lot for parking.
- **Environmental pollution**: They also cause pollution to the environment because stopping and starting of vehicles while parking and unparking results in noise and fumes. They also affect the aesthetic beauty of the buildings because cars parked at every available space creates a feeling that building rises from a plinth of cars.
- **Obstruction to firefighting operations**: Parked vehicles may obstruct the movement of firefighting vehicles. Sometimes they block access to hydrants and access to buildings.

.

.CHAPTER 2
PROJECT DESCRIPTION

2.1. AIM AND OBJECTIVE

The aim of the project is to design, develop, and implement a Smart Parking System that addresses the existing challenges in urban parking systems. The primary goal is to create an innovative and efficient solution that enhances the overall parking experience for users while optimizing space utilization, improving security measures, and contributing to a more sustainable and streamlined urban mobility.

Objectives

- To provide real-time parking updates for user convenience.
- To optimize space allocation, reducing congestion.
- To enhance security with facial recognition and vehicle identification.
- To develop a user-friendly interface for seamless interaction.
- To establish a unified network for cohesive parking management.
- To ensure transparent access control for a smooth entry-exit process.
- To streamline tariff management for cost-efficient parking.
- To integrate vehicle theft prevention measures using facial recognition.
- To implement a centralized monitoring dashboard for security.
- To ensure technological adaptability and seamless integration.
- To contribute to urban efficiency by reducing congestion and emissions.
- To provide accessible parking options for individuals with disabilities.
- To utilize data-driven insights for informed decision-making.
- To minimize environmental impact by reducing fuel consumption.
- To foster collaboration among stakeholders for system sustainability.

2.2. FEATURE OF THE SYSTEM

The Feature of the System design and implementation of a comprehensive system that addresses the challenges associated with parking in densely populated urban environments. The Smart Parking Web App will offer real-time updates on parking space availability, optimize the allocation and utilization of parking spaces, and enhance security through advanced features such as facial recognition and automatic vehicle identification. The user-centric interface will cater to the needs of drivers, parking space providers, and administrators, ensuring a seamless and intuitive experience. The project will establish a unified network of parking stations, streamlining parking processes and contributing to a cohesive parking ecosystem. Transparent access control, cost-efficient tariff management, and proactive vehicle theft prevention measures will be integrated to enhance overall system functionality. The development includes a centralized monitoring dashboard for efficient security oversight, ensuring a quick response to potential security issues. The Smart Parking Web App's technological adaptability and focus on urban efficiency aim to reduce congestion, minimize environmental impact, and contribute to a more sustainable and streamlined urban mobility experience.

2.3. PROJECT DESCRIPTION

The Smart Parking project represents a state-of-the-art solution addressing the complexities of urban parking management. This comprehensive system is built on a robust technology stack, incorporating Python for backend logic, Flask as the web framework, MySQL for database functionality, and Bootstrap for an intuitive frontend design. The Smart Parking Web App seamlessly integrates various modules to optimize parking space utilization, enhance user experience, and prioritize security. Users can effortlessly register, log in, and access real-time data on parking space availability. The system facilitates smooth reservations, integrates with navigation services, and employs transparent tariff structures with secure payment processing. The admin dashboards empower administrators with efficient monitoring tools, ensuring smooth system operation. The End User Dashboard, comprising User/Driver, Parking Space Provider Admin, and Web Admin modules, offers an intuitive, efficient, and secure experience for all stakeholders. Functionalities like User Management, Parking Slot Management, Tariff Management, and Reservation Modules contribute to seamless interaction, efficient allocation of spaces, and flexible booking options. The Vehicle Theft Prevention module employs facial recognition and deep learning algorithms for heightened security. The Payment Module ensures secure and convenient transactions, and the Notification module keeps users and administrators informed of crucial updates in real-time. The Smart Parking project strives to revolutionize urban parking, introducing a more efficient, secure, and user-centric parking environment.

..

CHAPTER 3

SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

The traditional system of parking slot prediction relies on conventional methods and technologies to estimate the availability of parking spaces.

- **Experience-Based Predictions**

Vehicle owners/drivers rely on their past experiences and knowledge of the parking facility to predict parking space availability. This could involve knowing peak hours, busy days, or areas with higher chances of finding vacant spots.

- **Observational Assessments**

Upon arrival at the parking facility, drivers make observational assessments by visually scanning the parking lot for available spaces. This method heavily relies on the driver's ability to gauge available spots based on their observations.

- **Static Signage**

Parking facilities may use static signs or boards at entrances to communicate general information about parking availability. However, these signs are typically not dynamic and may not reflect real-time changes in occupancy.

- **Communication with Attendants**

Drivers may communicate with parking attendants or staff to inquire about parking availability. Attendants may provide guidance based on their visual assessment of the parking lot.

- **Trial and Error**

In the absence of advanced prediction systems, drivers may resort to a trial-and-error approach, entering different sections of the parking facility until they find an available spot.

- **IoT Sensors**

Many systems deploy Internet of Things (IoT) sensors installed in parking spaces. These sensors detect the presence or absence of vehicles and transmit this data in real time.

- **Camera-based Systems**

Cameras with computer vision capabilities may be employed to monitor parking spaces. These systems can recognize license plates, detect vehicle presence, and contribute to real-time occupancy information.

3.1.1. Disadvantage

- Manual Observations lead to limited real-time visibility.
- Manual counting results in imprecise space availability estimates.
- Heavy reliance on attendants, which can be costly and inefficient.
- Lack of dynamic updates frustrates users in finding parking spaces.
- Minimal means for drivers to interact with the system.
- Initial high costs may pose a barrier to implementation.
- Technical glitches can disrupt parking infrastructure.
- Use of cameras or sensors raises user privacy concerns.
- Advanced technologies require regular and complex maintenance.
- Excludes non-digital users, creating an accessibility gap.
- Integrating new technologies into existing infrastructure can be difficult.

3.2. PROPOSED SYSTEM

The proposed system offering a seamless solution to the challenges associated with traditional parking systems. Embracing advanced technologies, the system aims to provide users with real-time information, efficient security measures, and a user-friendly interface. The proposed system is developed using a robust technological stack, including Python for programming, Flask for web development, MySQL for database management, and Bootstrap for responsive and user-friendly design.

- **Unified Parking Network**

The proposed system establishes a unified network of parking stations, creating a shared parking ecosystem. This interconnected network enhances overall efficiency and ensures a more cohesive parking experience for users.

- **User-Centric Interface**

The Smart Parking Web App prioritizes a user-centric interface, ensuring a positive and intuitive experience for drivers, parking space providers, and administrators. This focus on user-friendliness enhances overall usability.

- **Real-time Decision Support**

With a focus on real-time decision-making, the proposed system significantly reduces the time users spend searching for parking spaces. This dynamic approach enhances the overall efficiency and responsiveness of urban parking environments.

- **Advanced Security with Deep Learning**

The system incorporates state-of-the-art Deep Learning algorithms, specifically Convolutional Neural Networks (CNNs), to fortify security measures. Facial recognition technology is employed at entry and exit points, ensuring that only authorized individuals gain access to their associated vehicles.

- **Vehicle Theft Prevention**

Integrating a Vehicle Theft Prevention module, the system captures and recognizes faces during entry and exit. This proactive measure adds an additional layer of security, preventing potential vehicle theft.

- **Security Measures for Restricted Areas**

To address security concerns in highly restricted areas, such as military bases and government buildings, the system employs additional security measures, including advanced facial recognition and owner verification.

3.2.1. Advantage

- Real-time parking updates for users' convenience.
- Streamlined and seamless reservation process.
- Efficient allocation of parking spaces, reducing congestion.
- Advanced security with facial recognition technology.
- Improved traffic flow and urban mobility.
- Proactive vehicle theft prevention measures.
- Centralized monitoring for quick response to security alerts.
- User-friendly interfaces for drivers and administrators.
- Data-driven decisions through analytics insights.

CHAPTER 4

SYSTEM SPECIFICATION

HARDWARE REQUIREMENTS:

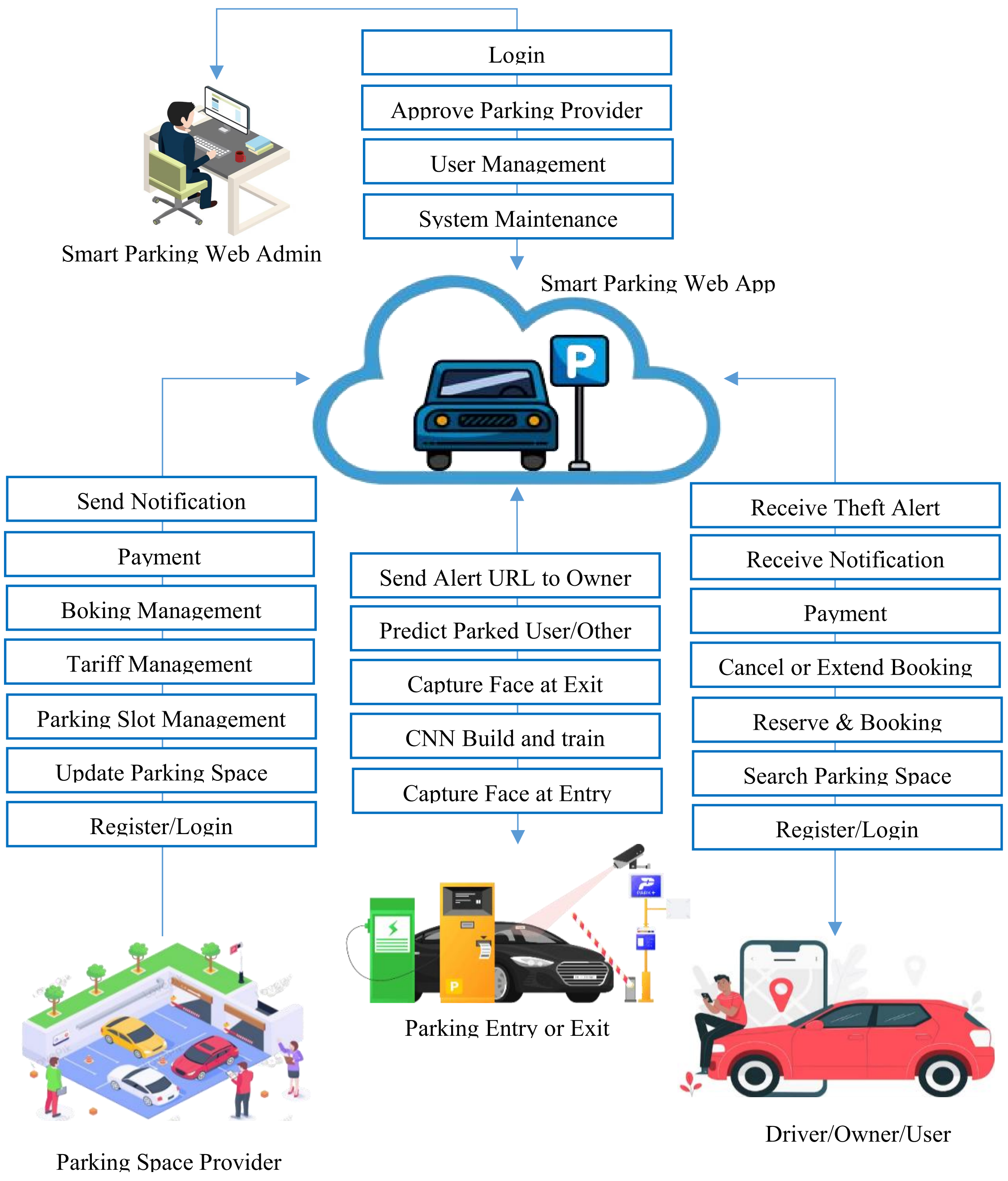
• Processors	: Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 16 GB of DRAM
• Disk space	: 320 GB
• Operating systems	: Windows® 10, macOS*, and Linux*

SOFTWARE REQUIREMENTS:

• Front End	: Python 3.7.4(64-bit) or (32-bit)
• IDE	: Flask 1.1.
• Back End	: MySQL 5.
• Server	: Wampserver 2i
• Blockchain	: JSON

CHAPTER 5
SYSTEM DESIGN

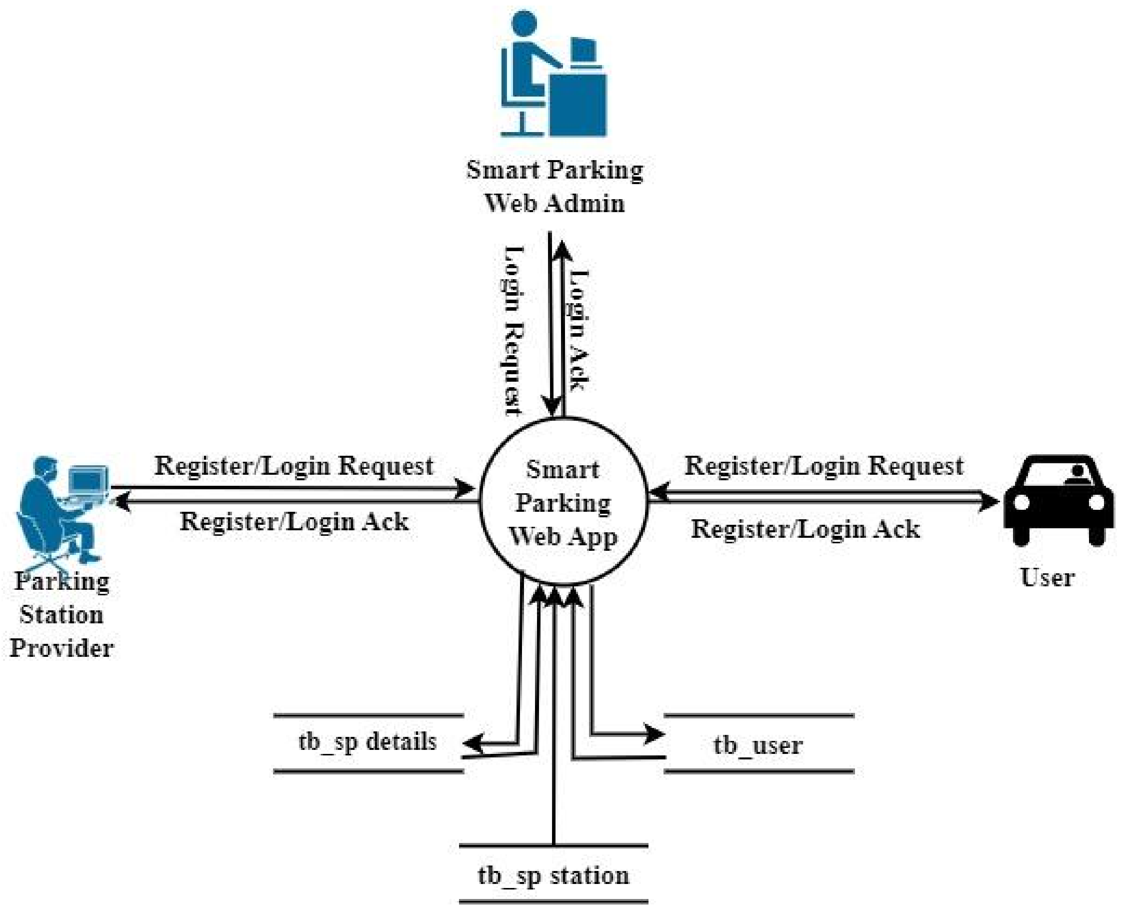
5.1 CONTEXT LEVEL DIAGRAM



5.1. Fig 1 : Architecture Diagram

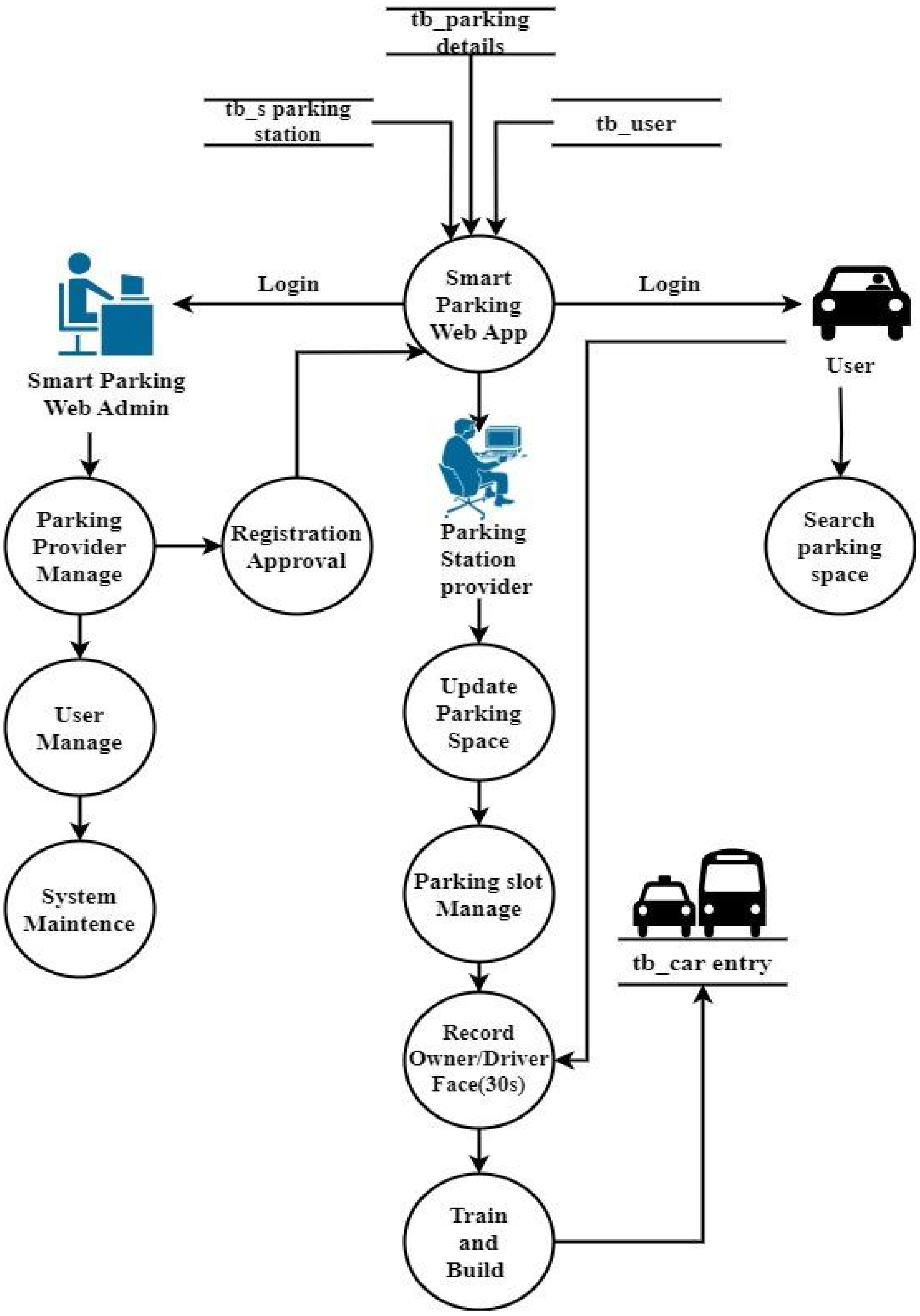
3.2. DATA FLOW DIAGRAM

Level 0



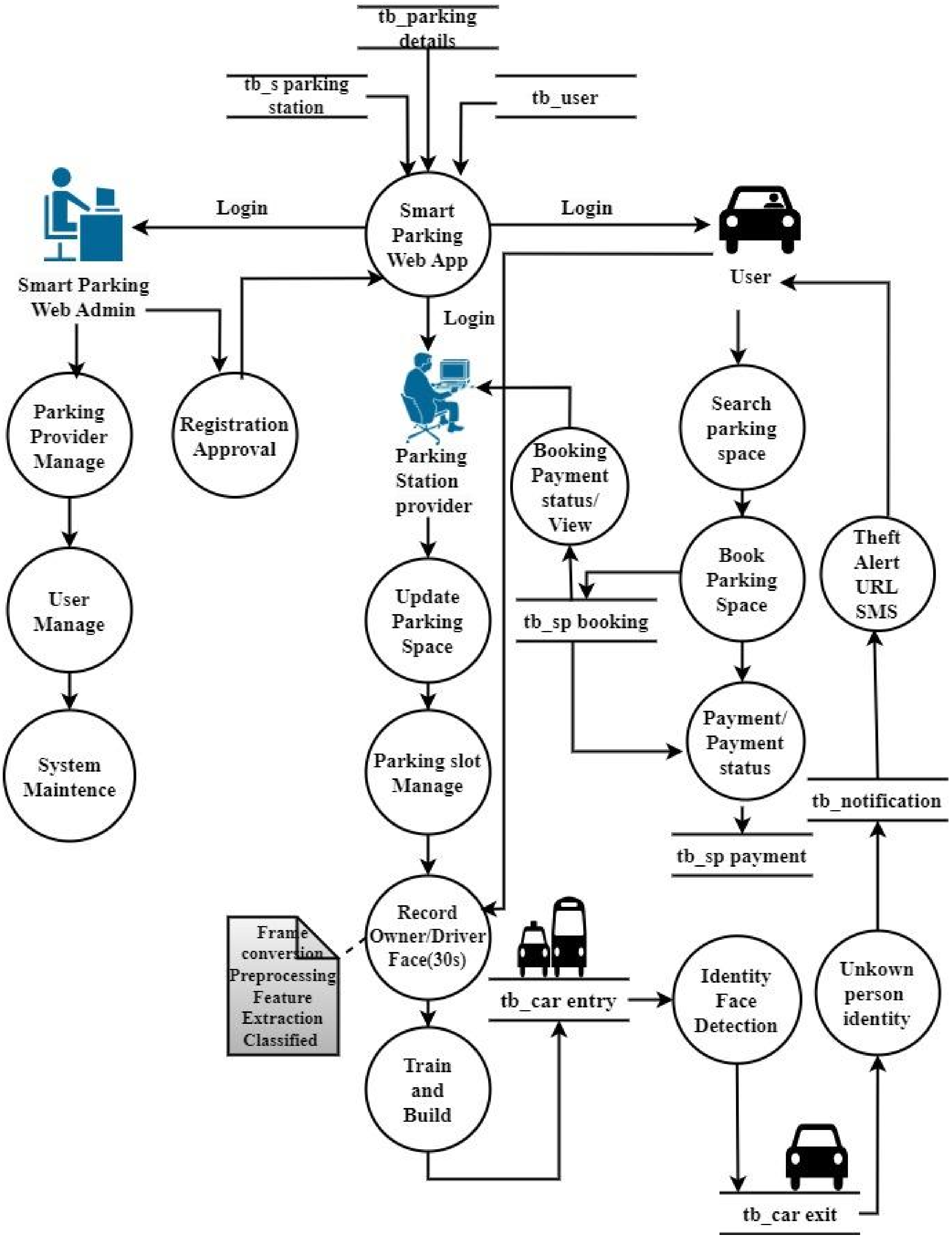
5.2. Fig 2 : Data Flow Diagram Level 0

Level 1



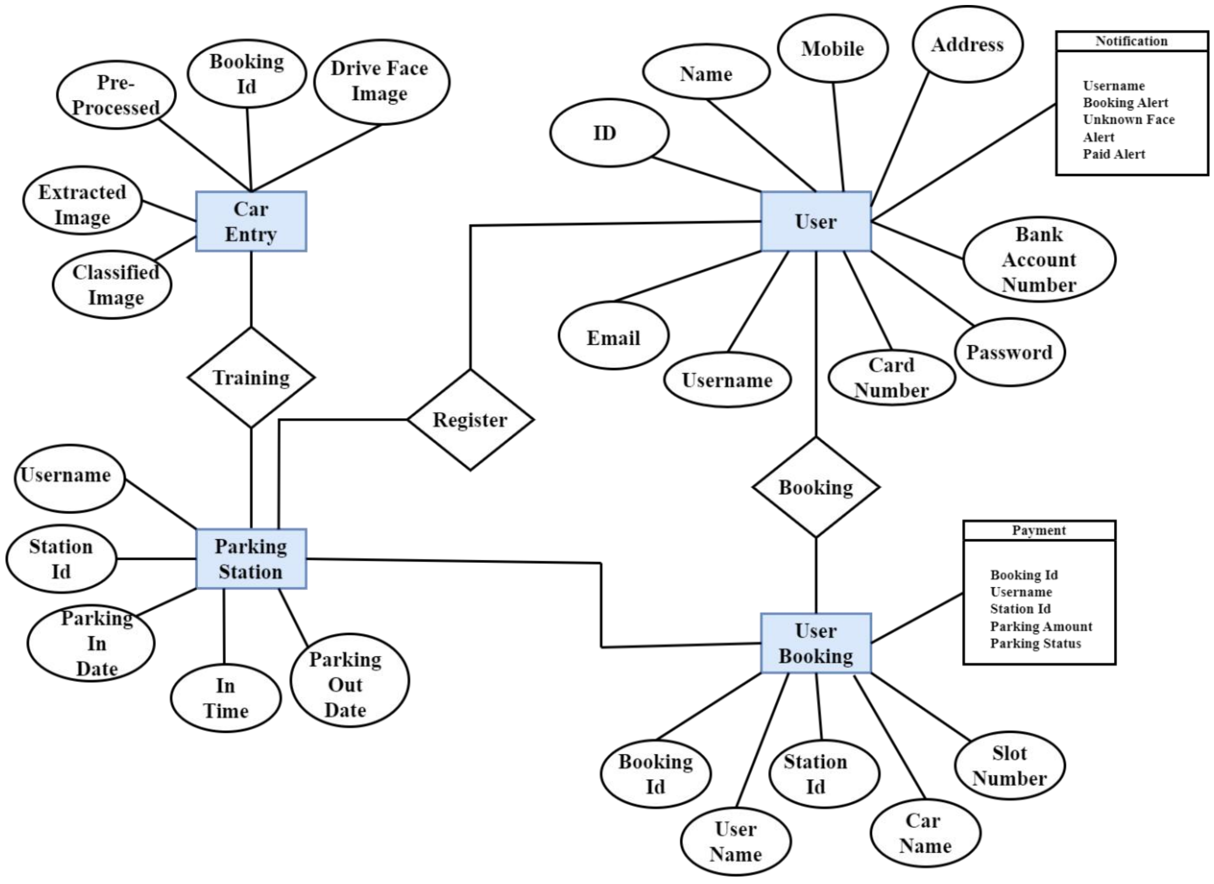
5.3. Fig 3 : Data Flow Diagram Level 1

Level 2



5.4. Fig 4 : Data Flow Diagram Level 2

3.3. ER DIAGRAM



5.5. Fig 5 : ER Diagram

3.4. DATABASE DESIGN

Table name: Parking Station					
S.no	**Field**	**Data type**	**Field size**	**Constraint**	**Description**
1	Id	Int	11	Null	Parking Id
2	Station Name	Varchar	20	Null	Station Name
3	Number of Slot	Int	11	Null	Number of slot
4	Area	Varchar	30	Null	Parking Area
5	City	Varchar	30	Null	City
6	Latitude	Varchar	20	Null	Latitude
7	Longitude	Varchar	20	Null	Longitude
8	Station Id	Varchar	20	Primary key	Station Id
9	Password	Varchar	20	Null	Password
10	Register date	Timestamp	Timestamp	Null	Register date

Table name: User Register					
S.no	**Field**	**Data type**	**Field size**	**Constraint**	**Description**
1	Id	Int	11	Null	User id
2	Name	Varchar	20	Null	User Name
3	Address	Varchar	40	Null	User Address
4	Mobile Number	Bigint	20	Null	User Mobile Number
5	Email	Varchar	30	Null	User Email
6	Bank Account Number	Varchar	30	Null	User Bank Account Number
7	Card Number	Varchar	20	Null	User Card Number

8	Bank Name	Varchar	20	Null	Bank Name
9	User Name	Varchar	20	Primary key	User Name
10	Password	Varchar	20	Null	User Password
11	Register date	Timestamp	Timestamp	Null	Register date

Table name: Booking					
S.no	**Field**	**Data type**	**Field size**	**Constraint**	**Description**
1	Id	Int	11	Null	Id
2	Booking Id	Int	11	Primary key	Booking Id
3	User Name	Varchar	20	Null	User Name
4	Station Id	Varchar	30	Null	Station Id
5	Car Number	Varchar	20	Null	Car Number
6	Slot Number	Int	11	Null	Slot Number
7	Booking Date	Timestamp	Timestamp	Null	Booking date

Table name: Car Entry					
S.no	**Field**	**Data type**	**Field size**	**Constraint**	**Description**
1	Id	Int	11	Null	Id
2	Booking Id	Varchar	20	Foreign key	Booking Id
3	Driver Face Image	Varchar	30	Null	Driver Face Image
4	Pre-processed image	Varchar	30	Null	Pre-processed image
5	Extracted Feature	Varchar	30	Null	Extracted Feature
6	Classified image	Varchar	30	Null	Classified image

Table name: Parking Details					
S.no	**Field**	**Data type**	**Field size**	**Constraint**	**Description**
1	Booking Id	Int	11	Primary key	Booking Id
2	User Name	Varchar	20	Foreign key	User Name
3	Station Id	Varchar	20	Null	Station Id
4	Parking In date	Varchar	15	Null	Parking In date
5	In Time	Varchar	15	Null	In Time
6	Parking Out date	Varchar	15	Null	Parking Out date
7	Out Time	Varchar	15	Null	Out Time

Table name: Car Exit					
S.no	**Field**	**Data type**	**Field size**	**Constraint**	**Description**
1	Id	Int	11	Null	Id
2	Booking Id	Varchar	20	Foreign key	Booking Id
3	Test Image	Varchar	30	Null	Test Image
4	Verify Status	Int	11	Null	Verify Status
5	Date Time	Timestamp	Timestamp	Null	Date Time

Table name: Payment					
S.no	**Field**	**Data type**	**Field size**	**Constraint**	**Description**
1	Id	Int	11	Null	Id
2	Booking Id	Varchar	20	Primary key	Booking Id
3	User Name	Varchar	40	Null	User Name
4	Station Id	Bigint	20	Null	Station Id

5	Parking Amount	Varchar	40	Null	Parking Amount
6	Payment status	Varchar	20	Null	Payment status

Table name: Notification					
S.no	**Field**	**Data type**	**Field size**	**Constraint**	**Description**
1	Id	Int	11	Null	Id
2	User Name	Varchar	20	Foreign key	User Name
3	Booking Alert	Varchar	20	Null	Booking alert
4	Unknown face alert	Varchar	20	Null	Unknown face alert
5	Paid Alert	Varchar	20	Null	Paid Alert
6	Date Time	Timestamp	Timestamp	Null	Date Time

CHAPTER 6
SOFTWARE DESCRIPTION

6.3.1. PYTHON 3.7.4

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python is huge collection of standard library which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks

- Multimedia
- Scientific computing
- Text processing and many more.

6.3.2. Pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

Pandas is mainly used for data analysis and associated manipulation of tabular data in Data frames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features. The development of pandas introduced into Python many comparable features of working with Data frames that were established in the R programming language. The panda's library is built upon another library NumPy, which is oriented to efficiently working with arrays instead of the features of working on Data frames.

6.3.3. NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

6.3.4. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

6.3.5. Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

seaborn

Seaborn offers the following functionalities:

- Dataset oriented API to determine the relationship between variables.
- Automatic estimation and plotting of linear regression plots.
- It supports high-level abstractions for multi-plot grids.
- Visualizing univariate and bivariate distribution.

6.3.6. Scikit Learn

scikit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.

scikits learn
machine learning in Python

Scikit-learn (formerly scikits. learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

6.3.7. MYSQL

MySQL tutorial provides basic and advanced concepts of MySQL. Our MySQL tutorial is designed for beginners and professionals. MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

6.3.8. WAMPSERVER

WampServer is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows you to manage easily your database.

WAMPServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world. The software is free to use and doesn't require a payment or subscription.

6.3.9. BOOTSTRAP 4

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.

It solves many problems which we had once, one of which is the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). **Easy to use**: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops

Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework

Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)

3.3.10. FLASK

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Using an IDE

As good as dedicated program editors can be for your programming productivity, their utility pales into insignificance when compared to Integrated Developing Environments (IDEs), which offer many additional features such as in-editor debugging and program testing, as well as function descriptions and much more.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have formed a validation support. Instead, Flask supports the extensions to add such functionality to the application. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let's take a closer look into Flask, so-called "micro" framework for Python. Flask was designed to be easy to use and extend. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you are free to plug in any extensions you think you need. Also you are free to build your own modules. Flask is great for all kinds of projects. It's especially good for prototyping. Flask is part of the categories of the micro-framework. Micro-framework is normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that sometime you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

CHAPTER 7
SYSTEM IMPLEMENTATION

IMPLEMENTATION PROCEDURES

The implementation of the Smart Parking System involves a meticulous process to seamlessly integrate cutting-edge technologies and user-friendly interfaces. Leveraging a robust technology stack with Python for backend logic, Flask as the web framework, and MySQL for the database, the system is designed to revolutionize urban parking environments. The implementation encompasses several key modules.

The Smart Parking Web App, the central component, allows users to register, log in, and access real-time information on parking space availability. It facilitates easy reservations, integrates with navigation services, and manages transparent tariff structures with secure payment processing. The system also provides an intuitive admin dashboard for efficient monitoring and control.

The End User Dashboard includes the User/Driver Dashboard, enabling account creation, login, profile management, parking space search, reservation, and payment. The Parking Space Provider Admin Dashboard empowers administrators to manage parking slots, tariffs, booking confirmations, and user notifications. The Web Admin Dashboard facilitates overall system control, including user management, registration approvals, and system maintenance.

User Management ensures seamless interaction and security. Users can register, log in, and update their profiles. Administrators efficiently manage user accounts, control access permissions, and monitor activities through an audit trail. Notifications keep users informed, and analytics provide valuable insights.

Parking Slot Management oversees the allocation, monitoring, and maintenance of parking spaces. It includes adding, editing, and removing parking slots, real-time slot availability, and location mapping. Tariff Management defines and manages pricing structures, supporting dynamic pricing strategies, special tariffs, and integration with payment gateways.

Parking Slot Finder helps users locate and reserve available parking spaces efficiently. It includes search by location and time, real-time availability updates, filtering options, map integration, and reservation preview. The *Parking Slot Visualizer* provides a visual representation of available parking spaces with color-coded indicators and an interactive map interface.

The Reservation Modules streamline the process of reserving parking spaces, allowing users to visually select slots, specify arrival times, view tariff information, and receive immediate confirmation. The Vehicle Theft Prevention module utilizes facial recognition for added security during entry and exit, with immediate alerts for potential unauthorized access.

The Payment Module ensures secure and convenient transactions, supporting various payment methods. It seamlessly integrates with the Reservation and Booking system, aligning payment details with specific reservations.

The Notification Module keeps users, providers, and administrators informed about key events and updates in real-time. It allows users to customize notification preferences, and security alerts are triggered for potential unauthorized access.

In summary, the Smart Parking System implementation is a comprehensive endeavor that combines technological innovation with user-centric design. By addressing user needs, ensuring security, and optimizing operational efficiency, the system aims to transform the parking experience in urban environments.

SYSTEM MAINTANCE

Maintaining a Smart Parking System is crucial to ensure its continuous and efficient operation. Here are key aspects of maintenance for a Smart Parking System:

1. **Regular System Monitoring**
    - Implement a monitoring system that tracks the overall health of the Smart Parking System in real-time.
    - Monitor server performance, database operations, and the functioning of critical modules.
2. **Database Maintenance**
    - Regularly perform database maintenance tasks such as indexing, optimization, and backup procedures.
    - Ensure data integrity and resolve any inconsistencies in the database.
3. **Software Updates and Upgrades**
    - Stay current with software updates and patches for the entire system, including the backend logic, web framework, and database.

- Schedule regular system upgrades to leverage new features, improve security, and address any bugs or vulnerabilities.

4. **Hardware Maintenance**
    - Monitor the health of physical hardware components such as servers, sensors, cameras, and networking equipment.
    - Replace or upgrade hardware components as needed to prevent system failures.
5. **Security Audits and Updates**
    - Conduct regular security audits to identify and address potential vulnerabilities.
    - Update security protocols and implement the latest technologies to safeguard against cyber threats.
6. **User Support and Training**
    - Provide ongoing user support to address any issues or queries from end-users.
    - Conduct training sessions for both administrators and end-users to ensure they are familiar with system features and functionalities.
7. **Incident Response and Troubleshooting**
    - Develop and regularly update an incident response plan to address system failures, security breaches, or other emergencies.
    - Establish a troubleshooting process to quickly identify and resolve any issues reported by users or detected through system monitoring.
8. **Data Backups and Recovery**
    - Implement regular data backup procedures to prevent data loss in the event of system failures.
    - Test data recovery processes to ensure the system can be quickly restored to normal operation.
9. **Compliance Checks**
    - Regularly review and ensure compliance with relevant data protection regulations, industry standards, and legal requirements.
    - Update system features to align with changing compliance standards.
10. **Performance Optimization**
    - Analyze system performance regularly and optimize resource utilization for better efficiency.

- Identify and address any bottlenecks or performance issues that may impact user experience.

11. **Notification System Maintenance**
    - Ensure the proper functioning of the notification module to keep users and administrators informed of critical events.
    - Test and verify the effectiveness of real-time alerts and notifications.
12. **Documentation Update**
    - Keep system documentation up-to-date, including manuals, standard operating procedures, and troubleshooting guides.
    - Document any changes or updates made to the system architecture or configurations.

Smart Parking System maintenance is an ongoing process that requires a proactive approach to prevent issues, address emerging challenges, and enhance overall system performance. Regular assessments and updates contribute to the system's reliability, security, and user satisfaction.

:

SOURCECODE

Packages

```
from flask import Flask, render_template, Response, redirect, request, session, abort,
url_for
from camera import VideoCamera
import os
import base64
import mysql.connector
import hashlib
import datetime
from datetime import date
import cv2
import numpy as np
import time
from random import randint
import shutil
import imagehash
import PIL.Image
from PIL import Image
from PIL import ImageTk
import urllib.request
import webbrowser
```

Database Connection

```
mydb = mysql.connector.connect(
host="localhost",
user="root",
password="",
charset="utf8",
database="smart_parking_face"
```

Login

```
def login():
msg=""
if request.method=='POST':
uname=request.form['uname']
pwd=request.form['pass']
cursor = mydb.cursor()
cursor.execute('SELECT * FROM ev_register WHERE uname = %s AND pass = %s',
(uname, pwd))
account = cursor.fetchone()
if account:
session['username'] = uname
cursor.execute('SELECT * FROM ev_register WHERE uname = %s', (uname, ))
dd = cursor.fetchone()
ff=open("name.txt","w")
ff.write(dd[1])
ff.close()
return redirect(url_for('userhome'))
else:
msg = 'Incorrect username/password!'
```

User Registration

```
def register():
```

```
msg=""
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM ev_register")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
if request.method=='POST':
address=request.form['address']
name=request.form['name']
mobile=request.form['mobile']
email=request.form['email']
account=request.form['account']
card=request.form['card']
bank=request.form['bank']
uname=request.form['uname']
pass1=request.form['pass']
cursor = mydb.cursor()
sql = "INSERT INTO
ev_register(id,name,address,mobile,email,account,card,bank,amount,uname,pass)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid,name,address,mobile,email,account,card,bank,'10000',uname,pass1)
cursor.execute(sql, val)
mydb.commit()
print(cursor.rowcount, "Registered Success")
msg="sucess"
return redirect(url_for('login'))
```

Booking for Parking

```
if request.method=='POST':
carno=request.form['carno']
reserve=request.form['reserve']
sid=request.form['sid']
slot=request.form['slot']
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM ev_booking")
```

```
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
t = time.localtime()
rtime = time.strftime("%H:%M:%S", t)
today= date.today()
rdate= today.strftime("%d-%m-%Y")
rn=randint(1, 10)
cimage="c"+str(rn)+".jpg"
cursor = mydb.cursor()
sql = "INSERT INTO
ev_booking(id,uname,station,carno,reserve,slot,cimage,rtime,rdate,status) VALUES
(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid,uname,sid,carno,reserve,slot,cimage,rtime,rdate,'1')
cursor.execute(sql, val)
mydb.commit()
vid=str(maxid)
print(cursor.rowcount, "Booked Success")
```

Training for Face Verification

```
##Preprocess
path="static/frame/"+rs[2]
path2="static/process1/"+rs[2]
mm2 = PIL.Image.open(path).convert('L')
rz = mm2.resize((200,200), PIL.Image.ANTIALIAS)
rz.save(path2)
'''img = cv2.imread(path2)
dst = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 15)
path3="static/process2/"+rs[2]
cv2.imwrite(path3, dst)'''
#noice
img = cv2.imread('static/process1/'+rs[2])
dst = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 15)
fname2='ns_'+rs[2]
cv2.imwrite("static/process1/"+fname2, dst)
```

```
##bin
image = cv2.imread('static/process1/'+rs[2])
original = image.copy()
kmeans = kmeans_color_quantization(image, clusters=4)
# Convert to grayscale, Gaussian blur, adaptive threshold
gray = cv2.cvtColor(kmeans, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (3,3), 0)
thresh = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY_INV,21,2)
# Draw largest enclosing circle onto a mask
mask = np.zeros(original.shape[:2], dtype=np.uint8)
cnts = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
for c in cnts:
((x, y), r) = cv2.minEnclosingCircle(c)
cv2.circle(image, (int(x), int(y)), int(r), (36, 255, 12), 2)
cv2.circle(mask, (int(x), int(y)), int(r), 255, -1)
break
# Bitwise-and for result
result = cv2.bitwise_and(original, original, mask=mask)
result[mask==0] = (0,0,0)
cv2.imwrite("static/process1/bin_"+rs[2], thresh)
#RPN - Segment
img = cv2.imread('static/process1/'+rs[2])
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, thresh =
cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)
# sure background area
sure_bg = cv2.dilate(opening,kernel,iterations=3)
# Finding sure foreground area
```

```
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
# Finding unknown region
sure_fg = np.uint8(sure_fg)
segment = cv2.subtract(sure_bg,sure_fg)
img = Image.fromarray(img)
segment = Image.fromarray(segment)
path3="static/process2/fg_"+rs[2]
segment.save(path3)
img = cv2.imread('static/process2/fg_'+rs[2])
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, thresh =
cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)
sure_bg = cv2.dilate(opening,kernel,iterations=3)
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
# Finding unknown region
sure_fg = np.uint8(sure_fg)
segment = cv2.subtract(sure_bg,sure_fg)
img = Image.fromarray(img)
segment = Image.fromarray(segment)
path3="static/process2/fg_"+rs[2]
segment.save(path3)
image = cv2.imread(path2)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
edged = cv2.Canny(gray, 50, 100)
image = Image.fromarray(image)
edged = Image.fromarray(edged)
path4="static/process3/"+rs[2]
edged.save(path4)
def kmeans_color_quantization(image, clusters=8, rounds=1):
h, w = image.shape[:2]
```

```
samples = np.zeros([h*w,3], dtype=np.float32)
count = 0
for x in range(h):
for y in range(w):
samples[count] = image[x][y]
count += 1
compactness, labels, centers = cv2.kmeans(samples,
clusters,
None,
(cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10000,
0.0001),
rounds,
cv2.KMEANS_RANDOM_CENTERS)
centers = np.uint8(centers)
res = centers[labels.flatten()]
return res.reshape((image.shape))
```

Face Verification

```
def get_frame(self):
success, image = self.video.read()
#self.out.write(image)
cv2.imwrite("getimg.jpg", image)
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Read the frame
#_, img = cap.read()
# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Detect the faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
#-Local Binary Patterns  (LBP)
id = 0
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
```

```
font = cv2.FONT_HERSHEY_SIMPLEX
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(
gray,
scaleFactor = 1.2,
minNeighbors = 5,
minSize = (int(self.minW), int(self.minH)),
# Draw the rectangle around each face
j = 1
        <script src="web_home/js/main.js"></script> <!-- Resource jQuery -->

        <!-- //banner js -->

        <!-- flexSlider --><!-- for testimonials -->

        <script defer src="web_home/js/jquery.flexslider.js"></script>

        <script type="text/javascript">

                $(window).load(function(){

                  $('.flexslider').flexslider({

                        animation: "slide",

                        start: function(slider){

                          $('body').removeClass('loading');

                        }

                  });

                });

        </script>

        <!-- //flexSlider --><!-- for testimonials -->

        <!-- start-smoth-scrolling -->

        <script src="web_home/js/SmoothScroll.min.js"></script>

        <script type="text/javascript" src="web_home/js/move-top.js"></script>
```

```
<script type="text/javascript" src="web_home/js/easing.js"></script>
<script type="text/javascript">
        jQuery(document).ready(function($) {
                $(".scroll").click(function(event){
                        event.preventDefault();
        $('html,body').animate({scrollTop:$(this.hash).offset().top},1000);
                });
        });
</script>
<!-- here stars scrolling icon -->
<script type="text/javascript">
        $(document).ready(function() {
                /*
                        var defaults = {
                        containerID: 'toTop', // fading element id
                        containerHoverID: 'toTopHover', // fading element
hover id
                        scrollSpeed: 1200,
                        easingType: 'linear'
                        };
                */

                $().UItoTop({ easingType: 'easeOutQuart' });

                });
```

```
            </script>

            <!-- //here ends scrolling icon -->

            <!-- start-smoth-scrolling -->

    <!-- //js-scripts -->

    </body>

    </html>
```

CHAPTER 8
SYSTEM TESTING

SYSTEM TESTING

System testing of the proposed system involves evaluating its functionality, usability, security, and performance to ensure that it meets the requirements and expectations of users, administrators, and other stakeholders. Here's a systematic approach to conducting system testing:

- **Functionality Testing**

Test each module of the web app to ensure that it performs as expected according to its specifications. Verify that users can register, log in, search for parking spaces, make reservations, update their profiles, and perform other essential tasks smoothly. Validate the functionality of administrative dashboards for managing parking spaces, tariffs, users, reservations, and notifications. Test the accuracy and reliability of real-time information updates, reservation management, and facial recognition-based security measures.

- **Usability Testing**

Evaluate the user interface design for intuitiveness, consistency, and accessibility across different devices and screen sizes. Conduct user walkthroughs to assess the ease of performing common tasks such as searching for parking spaces, making reservations, and updating profiles. Gather feedback from representative users to identify any usability issues or areas for improvement in navigation, layout, and interaction flow.

- **Security Testing**

Assess the effectiveness of security measures such as encryption, authentication, authorization, and data protection mechanisms. Verify the integrity and confidentiality of user data, payment transactions, and sensitive information stored in the system.

Perform penetration testing to identify vulnerabilities and potential attack vectors, and implement countermeasures to mitigate security risks.

- **Performance Testing**

Measure the responsiveness and scalability of the web app under various load conditions, including peak usage periods. Conduct stress testing to determine the system's stability and resilience under high traffic volumes or resource constraints. Monitor system resources such as CPU usage, memory consumption, and network bandwidth to identify any performance bottlenecks or optimization opportunities.

- **Compatibility Testing**

Validate the compatibility of the web app with different web browsers, operating systems, and devices commonly used by users. Test the responsiveness and functionality of the app across a range of devices, including desktops, laptops, tablets, and smartphones.

- **Integration Testing**

Verify the seamless integration and interoperability of different modules within the web app ecosystem. Test data exchange and communication between modules such as user management, parking slot management, tariff management, payment processing, and notification delivery.

- **Regression Testing**

Ensure that recent changes or updates to the web app have not introduced any new defects or regressions. Re-run previously conducted tests to validate the continued functionality and stability of the system after modifications or enhancements.
By systematically conducting these tests, the project can be evaluated and validated to ensure its reliability, security, and usability in real-world urban parking environments. Any issues or deficiencies identified during testing can be addressed promptly, leading to a more robust and dependable solution for users and administrators alike.

Testing Methodologies

Functional Testing

- **Unit Testing:** Test individual modules of the web app, such as user registration, login, reservation management, and notification systems, to ensure they perform as expected.
- **Integration Testing:** Verify the interaction and data exchange between different modules to ensure seamless integration and interoperability.
- **System Testing:** Conduct end-to-end testing of the entire system to validate its functionality and behavior as a whole, including user interactions, database operations, and external integrations.

Non-Functional Testing

- **Usability Testing:** Evaluate the user interface design and interaction flow to ensure it is intuitive, consistent, and user-friendly.
- **Performance Testing:** Assess the responsiveness, scalability, and stability of the web app under various load conditions to ensure it can handle expected traffic volumes.
- **Security Testing:** Test the security features and mechanisms of the system to identify vulnerabilities, ensure data protection, and prevent unauthorized access.
- **Compatibility Testing:** Verify the compatibility of the web app with different browsers, devices, and operating systems to ensure a consistent user experience across platforms.

Automation Testing

- **Regression Testing:** Automate repetitive test cases to ensure that recent changes or updates to the web app do not introduce new defects or regressions.
- **Functional Testing:** Automate test scenarios for critical functionalities to expedite testing and improve test coverage.
- **Load Testing:** Use automated tools to simulate heavy user traffic and assess the performance of the system under stress conditions.

Manual Testing

- **Exploratory Testing:** Allow testers to explore the web app freely to uncover unexpected issues or usability concerns.
- **Ad Hoc Testing:** Perform unplanned testing to identify defects or inconsistencies that may not be covered by formal test cases.
- **User Acceptance Testing (UAT):** Engage end-users to manually test the web app and provide feedback on its functionality, usability, and overall satisfaction.

Black Box Testing

Test the functionality of the web app without knowledge of its internal structure or code implementation to simulate real-world user interactions and scenarios. Focus on validating inputs, outputs, and system behavior based on specifications and requirements.

White Box Testing

Inspect the internal structure and logic of the web app to ensure that all code paths are tested and potential errors are identified. Verify the correctness of algorithms, data structures, and error-handling mechanisms implemented in the system.

By employing a combination of these testing methodologies, the Smart Parking Web App can undergo thorough evaluation and validation, resulting in a high-quality, reliable, and user-friendly solution for urban parking management.

Test Cases and Expected Results

User Registration

Input: New user provides name, contact details, and preferred payment methods.

Expected Result: User account is successfully created in the system.

Actual Result: User account is created with provided details.

Status: Pass

Login Authentication

Input: User provides correct login credentials.

Expected Result: User is successfully authenticated and gains access to the dashboard.

Actual Result: User is authenticated and gains access to the dashboard.

Status: Pass

Search for Parking Space

Input: User specifies location and time preferences for parking.

Expected Result: System displays available parking spaces matching the criteria.

Actual Result: System displays relevant parking spaces based on the input.

Status: Pass

Reservation of Parking Space

Input: User selects a parking space and specifies time and duration for reservation.

Expected Result: Selected parking space is reserved for the specified time.

Actual Result: Parking space is successfully reserved as per user's request.

Status: Pass

Cancellation of Booking:

Input: User cancels a previously made parking reservation.

Expected Result: Reserved parking space becomes available again, and any charges are refunded.

Actual Result: Reserved parking space is freed up, and cancellation is processed.

Status: Pass

Facial Recognition during Entry

Input: User's face is captured during vehicle entry.

Expected Result: System accurately identifies the user's face and grants access.

Actual Result: Facial recognition successfully identifies the user, allowing entry.

Status: Pass

Facial Recognition during Exit

Input: User's face is captured during vehicle exit.

Expected Result: System verifies user's identity and allows exit if matched with entry.

Actual Result: User's identity is verified, and exit is permitted upon successful match.

Status: Pass

Real-time Notifications

Input: Various triggers such as reservation confirmations, payment status changes, and parking slot availability updates.

Expected Result: Users, administrators, and parking space providers receive timely notifications.

Actual Result: Notifications are promptly delivered to relevant stakeholders based on triggers.

Status: Pass

Payment Processing:

Input: User initiates payment for parking reservation.

Expected Result: Payment is securely processed, and user receives confirmation.

Actual Result: Payment is processed securely, and user receives confirmation of successful transaction.

Status: Pass

System Maintenance

Input: Routine maintenance tasks and updates are scheduled.

Expected Result: System undergoes maintenance without affecting user experience.

Actual Result: Maintenance tasks are executed smoothly, and users are unaffected.

Status: Pass

Test Report

The System has undergone rigorous testing to ensure its functionality, reliability, and security. This report summarizes the testing process, results, and conclusions.

Test Objective

The objective of testing the Smart Parking Web App is to verify its modules and features, including user registration, login authentication, reservation management, payment processing, and security measures such as facial recognition.

Test Scope

The testing scope encompasses all modules and functionalities of the Smart Parking Web App, including user interfaces, administrative dashboards, backend logic, database operations, and third-party service integrations.

Test Environment

The testing environment includes:

- Backend Logic: Python
- Web Framework: Flask
- Database: MySQL
- Frontend Design: Bootstrap
- Technologies: Facial recognition libraries, Payment gateways

Test Result

The testing results are as follows:

New User Registration: New users successfully created accounts with provided details.

User Authentication: Users were authenticated and gained access to the dashboard with correct login credentials.

Parking Space Search: The system displayed available parking spaces matching user-specified location and time preferences accurately.

Reservation Management: Users successfully reserved parking spaces for specified times, and cancellations were processed without issues.

Facial Recognition: Facial recognition accurately identified users during vehicle entry and exit, granting or denying access accordingly.

Notification System: Users, administrators, and parking space providers received timely notifications based on various triggers.

Payment Processing: Payment for parking reservations was securely processed, and users received confirmation of successful transactions.

Maintenance: Routine maintenance tasks and updates were executed smoothly without affecting user experience.

Test Conclusion

The Smart Parking Web App has successfully passed all test cases, demonstrating its functionality, reliability, and security. With its user-friendly interfaces, efficient reservation management, and robust security measures, the app presents a promising solution to urban parking challenges. Further refinement and continuous testing will ensure its ongoing effectiveness and reliability in real-world scenarios.

QUALITY ASSURANCE

Quality Assurance (QA) is a systematic process or set of activities implemented within a project or organization to ensure that the products or services delivered meet predefined quality standards and expectations. QA is an integral part of the software development life cycle (SDLC) and is applicable to various industries beyond software, including manufacturing, healthcare, and services.

4.2.1. Generic Risks:

- Technical Challenges: Implementing advanced technologies like facial recognition and real-time data processing may pose technical challenges, including system integration issues, compatibility concerns, and potential software bugs.
- Data Accuracy and Integrity*:* Ensuring the accuracy and integrity of parking-related data is critical. Inaccurate information on space availability or payment processing errors can lead to user dissatisfaction and operational disruptions.
- User Adoption and Training*:* The success of the Smart Parking system relies on user adoption. Risks related to a lack of user awareness, training, or resistance to technology adoption could impact the project's overall effectiveness.

Security Technologies and Policies

- Facial Recognition*:* The Vehicle Theft Prevention module utilizes facial recognition technology. Risks associated with false positives/negatives, privacy concerns, and potential system vulnerabilities need thorough consideration and robust security measures.

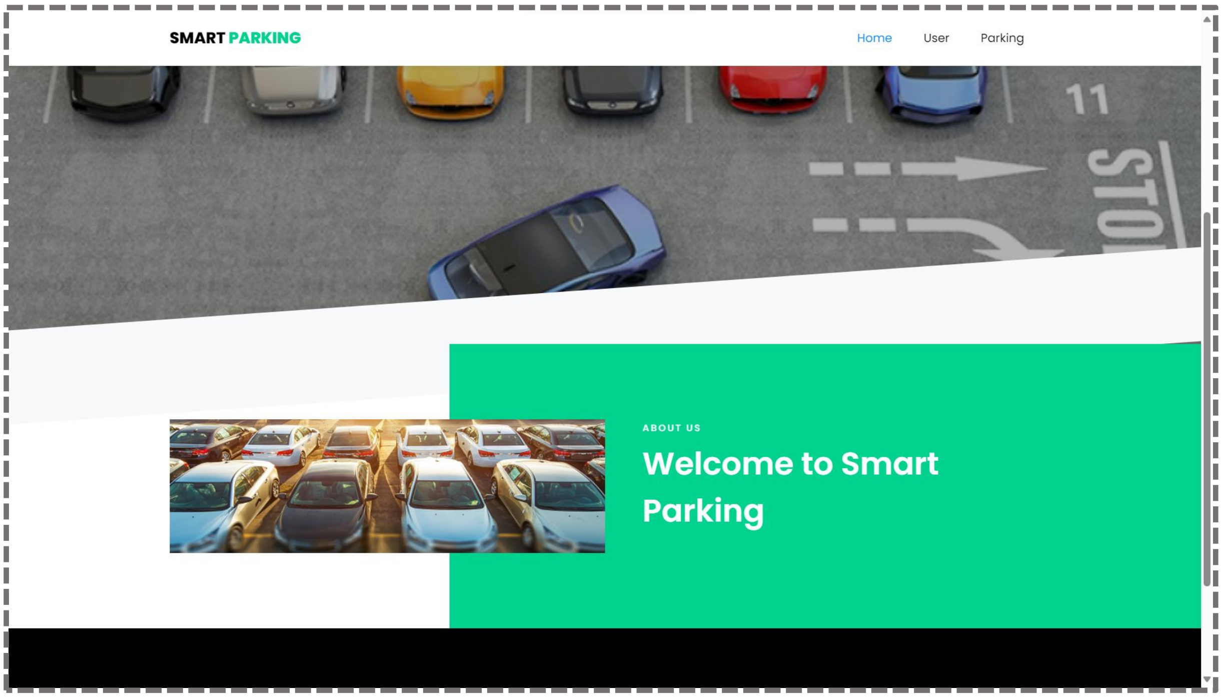
- Secure Payment Gateway*:* The Payment Module relies on a secure payment gateway. Risks include payment fraud, data breaches, and transaction security. The implementation of encryption and compliance with industry standards are crucial to mitigate these risks.
- Data Encryption*:* Ensuring end-to-end encryption for sensitive user and payment data is essential. Risks associated with data interception during transmission should be mitigated through robust encryption protocols.

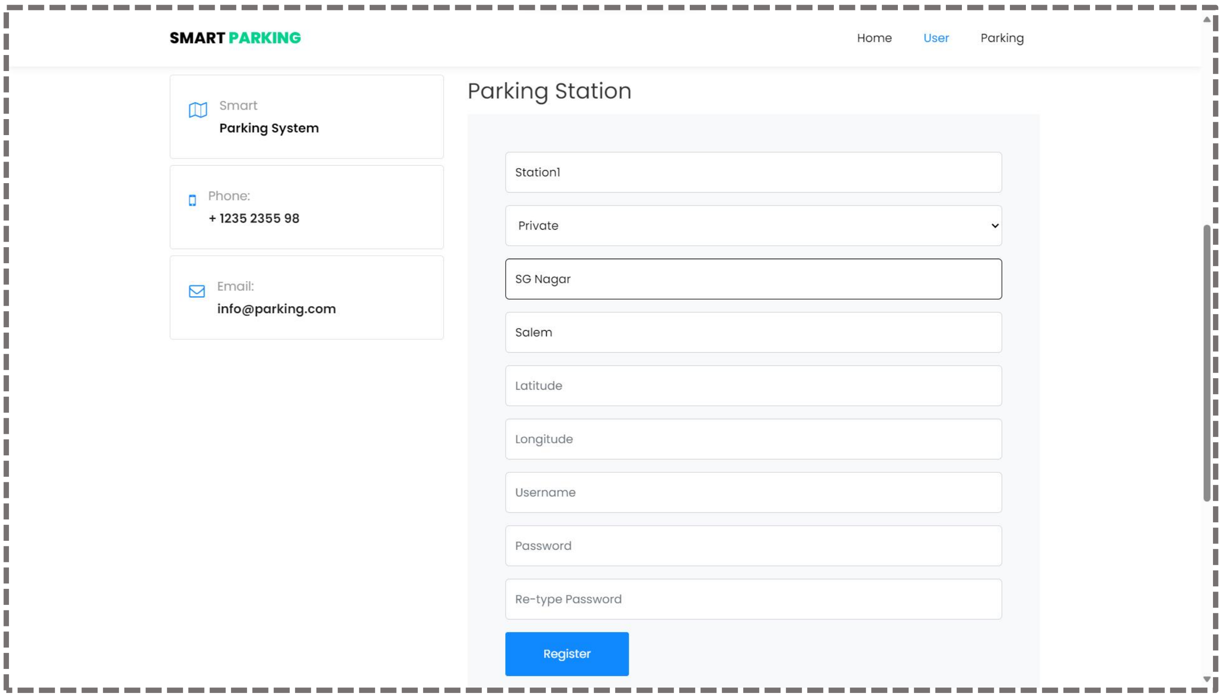
Security Policies:

- Access Control Policies*:* Strict access control policies must be in place to prevent unauthorized access to sensitive data. Role-based access ensures that users and administrators have the appropriate level of access based on their responsibilities.
- Data Privacy and Compliance*:* Compliance with data protection regulations is imperative. Implementing policies that prioritize user privacy, secure data storage, and adherence to legal requirements are essential components of the security framework.
- Incident Response Plan: A comprehensive incident response plan should be established to address security breaches or system failures promptly. This includes protocols for notifying users, investigating incidents, and implementing corrective actions.
- Regular Security Audits: Frequent security audits and vulnerability assessments help identify and address potential security gaps. Regular updates to security policies and protocols ensure the system remains resilient against evolving threats.

In summary, ensuring the quality and security of the Smart Parking project involves addressing generic risks through robust technical solutions and the implementation of comprehensive security technologies and policies. Regular monitoring, updates, and adherence to industry best practices are crucial for the success and sustainability of the Smart Parking system.

CHAPTER 9
SCREENSHOTS





Smart
Parking System

Phone:
+ 1235 2355 98

Email:
info@parking.com

Parking Station

Station1

Private

SG Nagar

Salem

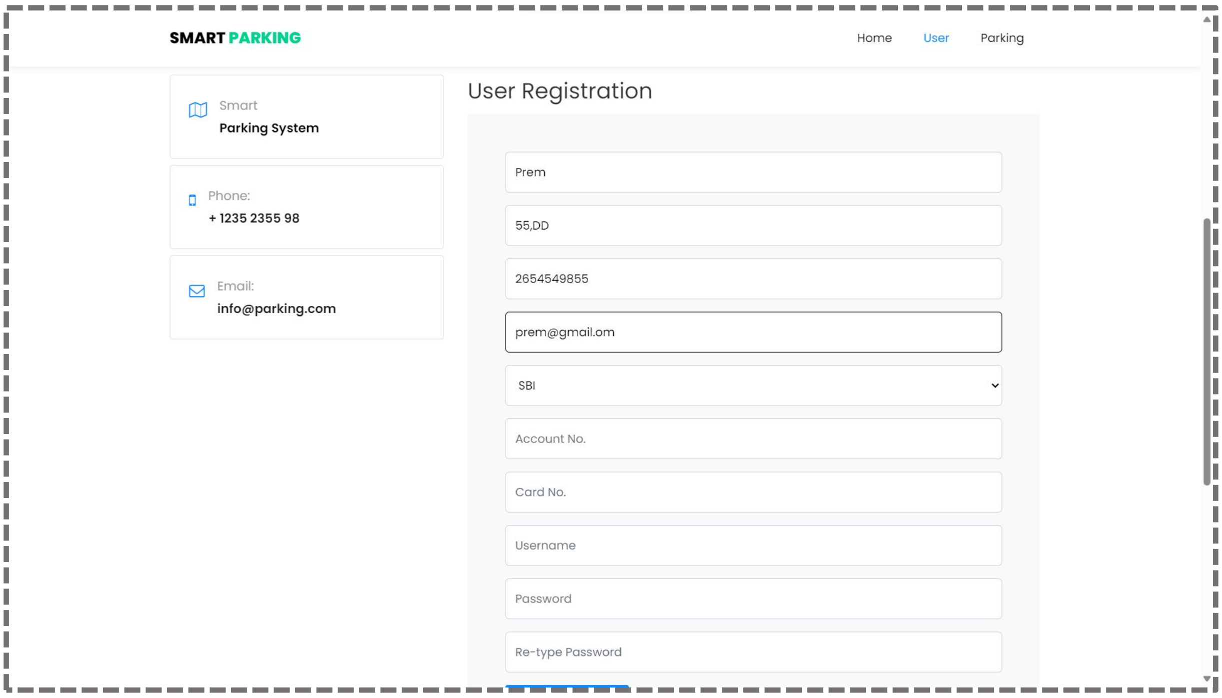
Latitude

Longitude

Username

Password

Re-type Password


SMART PARKING

Home User Parking

Smart
Parking System

Phone:
+ 1235 2355 98

Email:
info@parking.com

User Registration

Prem

55,DD

2654549855

prem@gmail.om

SBI

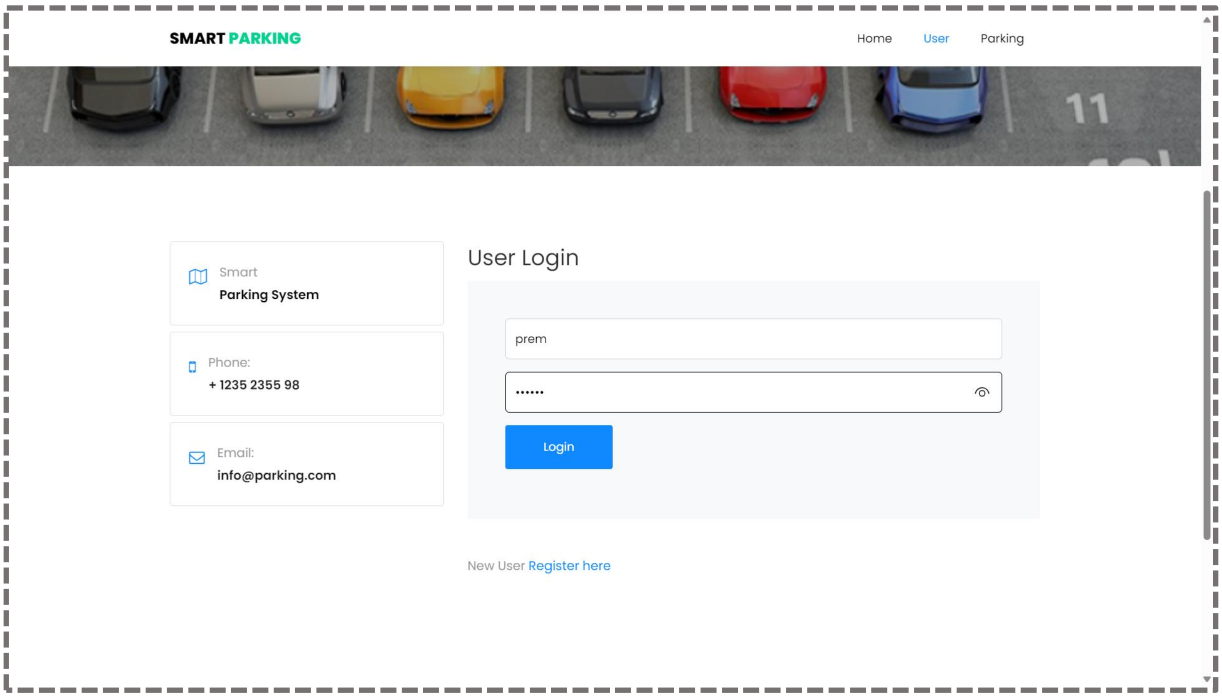
Account No.

Card No.

Username

Password

Re-type Password


SMART PARKING

Home User Parking

Smart
Parking System

Phone:
+ 1235 2355 98

Email:
info@parking.com

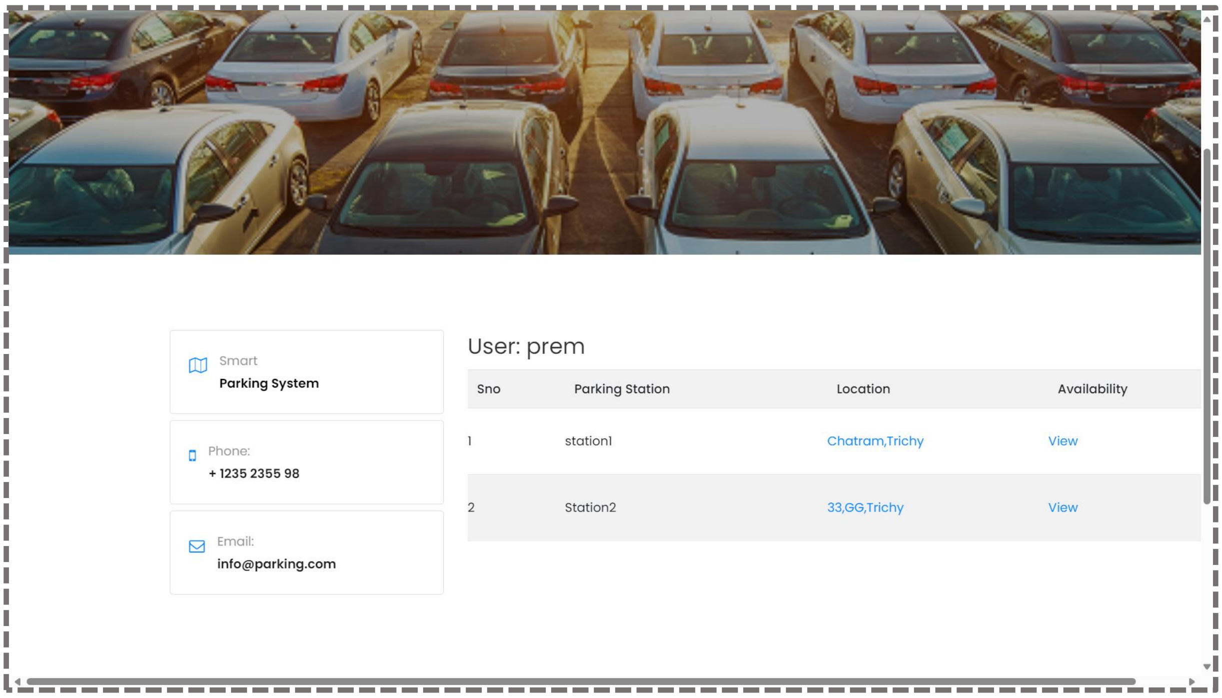
User Login

prem

••••••

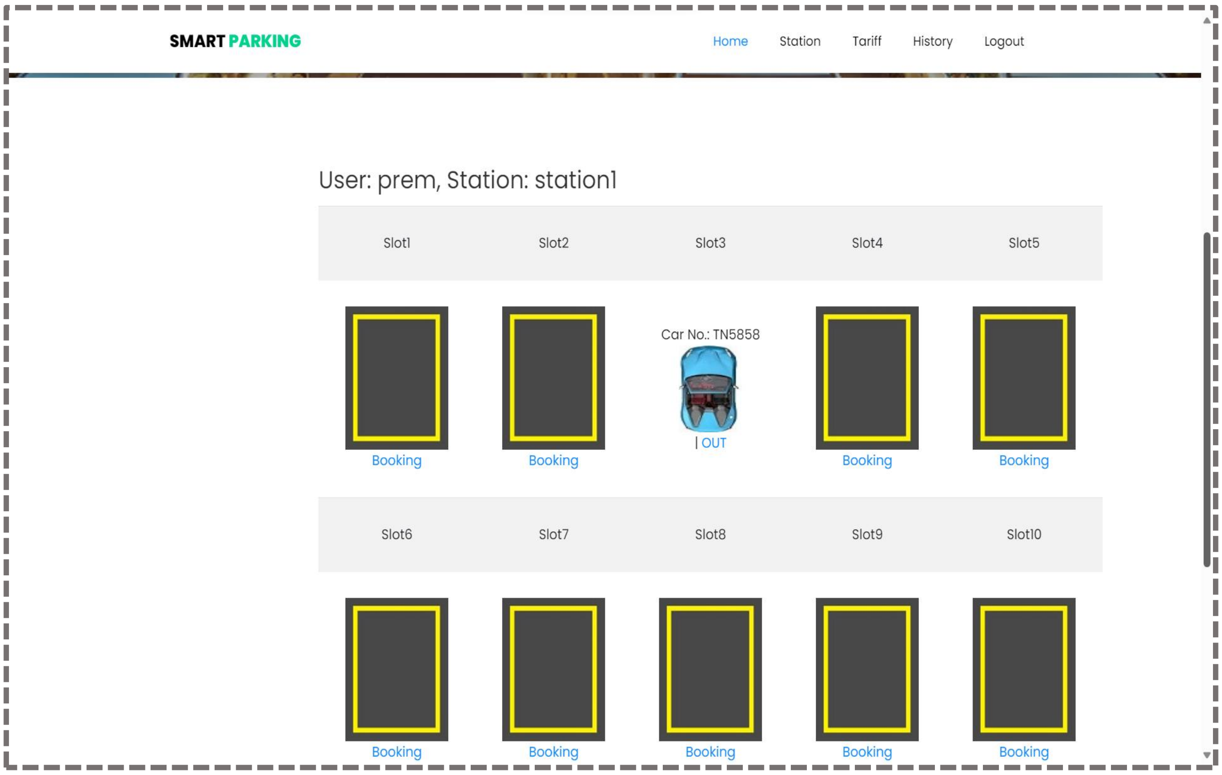
Login

New User Register here



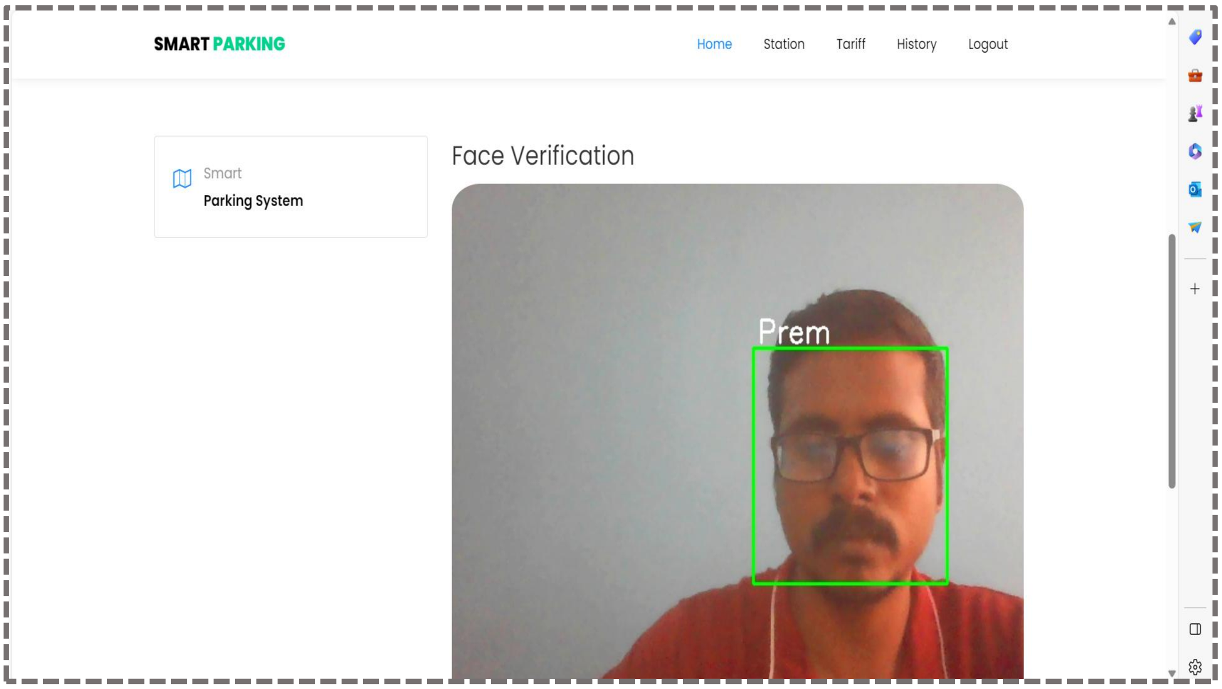
User: prem

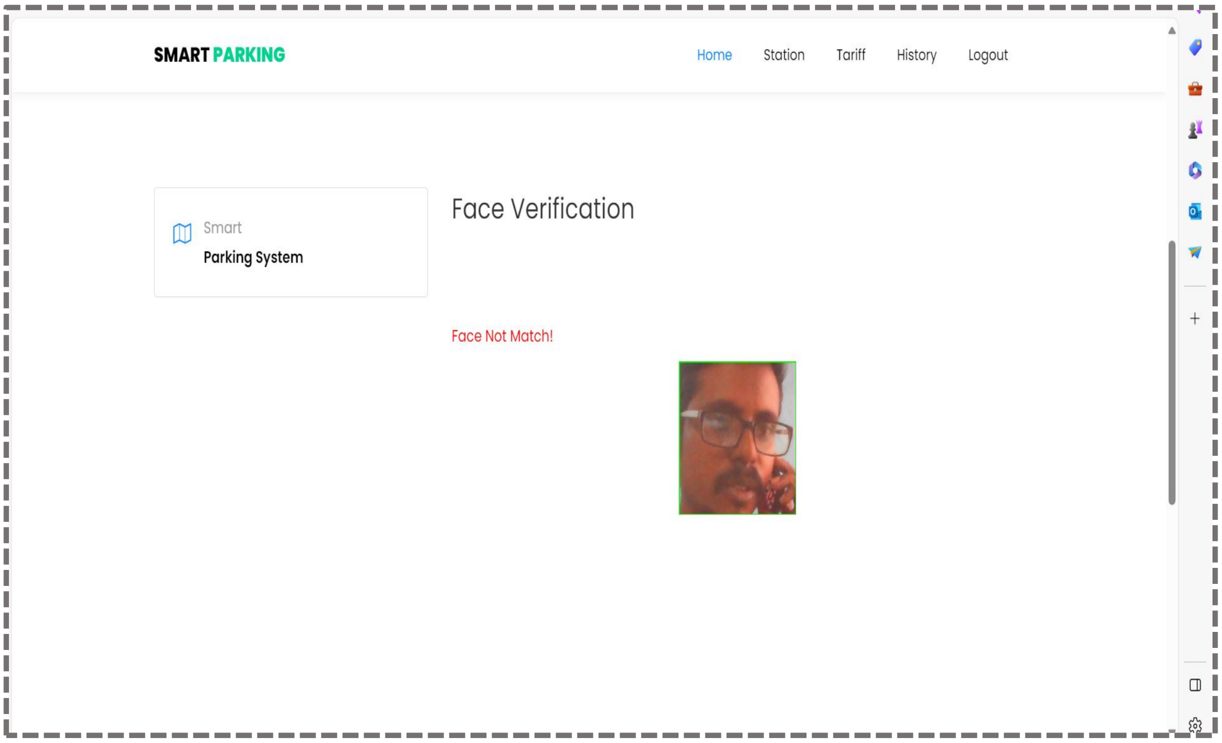
Sno	Parking Station	Location	Availability
1	station1	Chatram,Trichy	View
2	Station2	33,GG,Trichy	View



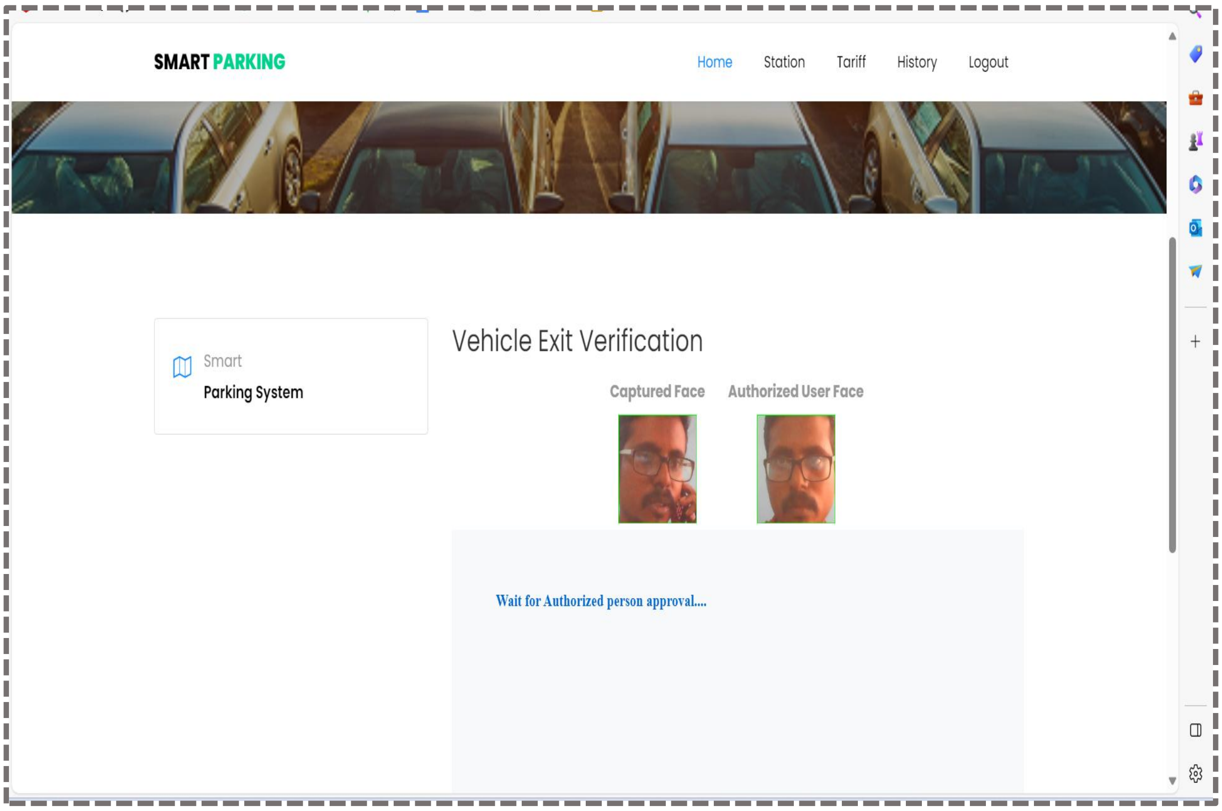
User: prem, Station: station1

Slot1	Slot2	Slot3	Slot4	Slot5
Booking	Booking	Car No.: TN5858 | OUT	Booking	Booking
Slot6	**Slot7**	**Slot8**	**Slot9**	**Slot10**
Booking	Booking	Booking	Booking	Booking





Face Not Match!



SMART PARKING

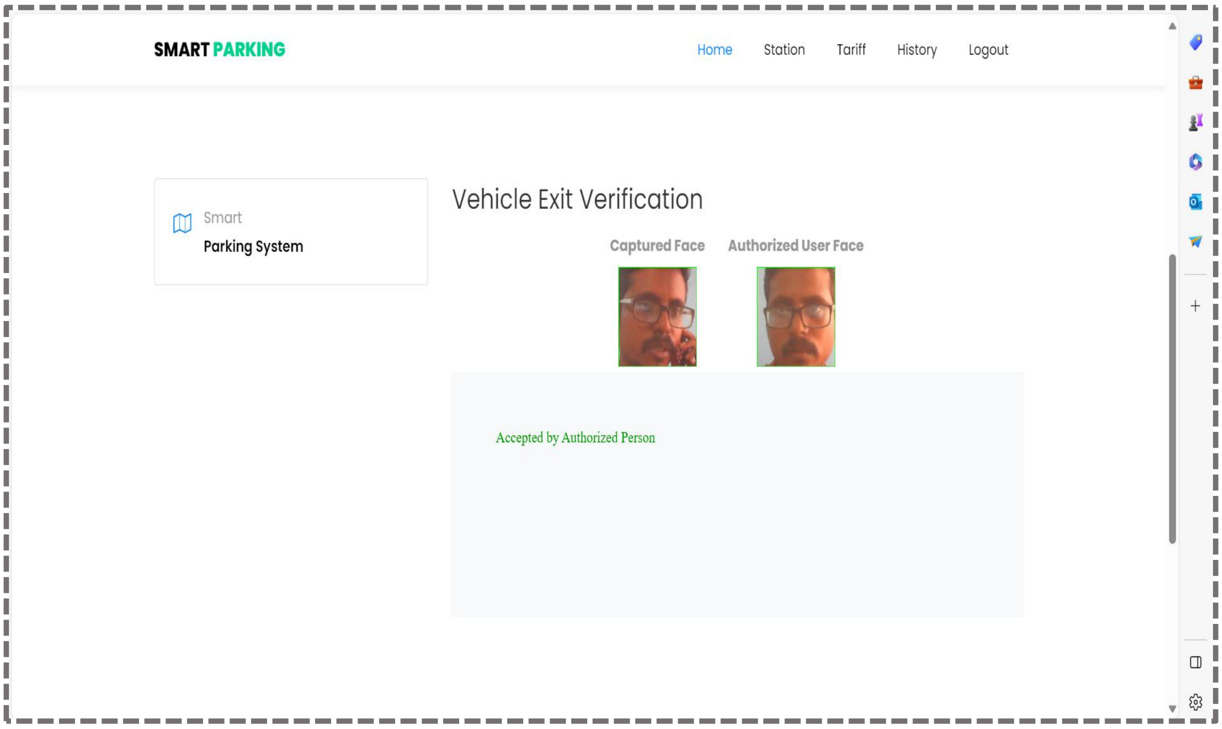
Home Station Tariff History Logout

Smart
Parking System

Vehicle Exit Verification

Captured Face Authorized User Face

Wait for Authorized person approval....



SMART PARKING

Home Station Tariff History Logout

Smart
Parking System

Vehicle Exit Verification

Captured Face Authorized User Face

Accepted by Authorized Person

Online

CHAPTER 10
CONCLUSION

In conclusion, the project stands as a transformative solution poised to revolutionize urban parking management. By leveraging advanced technologies and innovative features, the project addresses the challenges associated with traditional parking systems, providing users with a seamless and secure parking experience. The integration of Edge Computing and Deep Learning algorithms establishes a robust foundation for real-time decision-making and enhances the overall efficiency and security of urban parking environments. The Smart Parking System's ability to link multiple parking stations into a unified network fosters a shared parking ecosystem, promoting optimal space utilization. The Vehicle Theft Prevention module, incorporating facial recognition and deep learning, exemplifies the project's commitment to security. This proactive approach not only safeguards parked vehicles but also empowers owners to intervene swiftly in case of suspicious activities, reinforcing user trust. The End User Dashboard, Parking Space Provider Admin functionality, and Web Admin controls create a well-rounded and user-centric platform. Users benefit from features like easy registration, intuitive parking slot search, and transparent tariff information. Parking space providers gain tools for efficient management, and administrators wield control over the system's overall health and security. Looking ahead, the future scope of the project holds exciting possibilities, from IoT integration and predictive analytics to collaborations with smart city initiatives and advancements in user experience. The project is positioned not only to meet the current demands of urban mobility but also to adapt and thrive in the dynamic landscape of smart cities.

In essence, the project represents a significant step toward creating more sustainable, secure, and user-friendly urban environments. As urbanization continues, the project's contributions to traffic decongestion, environmental sustainability, and enhanced security underscore its potential to shape the future of urban living. The journey doesn't end here; the project is a dynamic solution ready to evolve with the ever-changing needs of modern cities..

CHAPTER 11
FUTURE ENHANCEMENT

- Integration with IoT and Smart City Infrastructure:

- IoT Sensors and Devices: Deploy more IoT sensors across the city to gather real-time data on parking spot availability and traffic conditions. This data can be used to guide drivers to the nearest available parking spot efficiently.

- Smart Traffic Management: Integrate with the city's traffic management systems to optimize traffic flow and reduce congestion caused by vehicles searching for parking.

- Advanced Analytics and AI:
  - Predictive Analytics: Implement predictive analytics to forecast parking spot availability based on historical data and current trends. This will help drivers plan their trips better and reduce search time.
  - Enhanced Deep Learning Models: Continuously train and improve the deep learning models used for facial recognition and vehicle identification to enhance accuracy and speed.

- Mobile Application Enhancements:

  - User-friendly Interface: Improve the user interface of the mobile application to make it more intuitive and user-friendly.
  - Real-time Notifications: Provide real-time notifications about available parking spots, parking duration alerts, and other relevant information.
  - Reservation System: Allow users to reserve parking spots in advance through the mobile app, reducing uncertainty and improving convenience.

- Dynamic Pricing and Payment Systems:

  - Dynamic Pricing: Implement dynamic pricing strategies to manage demand and optimize parking spot utilization. Prices can vary based on location, time of day, and demand.
  - Multiple Payment Options: Integrate multiple payment options, including mobile payments, contactless payments, and digital wallets, to offer more flexibility to users.

- Enhanced Security Features:

  - Multi-factor Authentication: Introduce multi-factor authentication (MFA) for added security. This could include biometric verification (fingerprint, facial recognition) along with traditional methods.
  - Anomaly Detection: Use machine learning algorithms to detect unusual activities or security breaches in real-time and alert the authorities.

- Scalability and Flexibility:

  - Cloud Integration: Migrate data storage and processing to the cloud to enhance scalability and ensure that the system can handle large volumes of data and users.
  - Modular Design: Design the system with a modular architecture to allow easy upgrades and integration of new features without disrupting the existing system.

- Environmental Impact:

  - Green Parking Initiatives: Promote green parking initiatives by providing incentives for electric vehicles (EVs) and integrating EV charging stations within parking facilities.
  - Energy-efficient Operations: Optimize the energy usage of the parking facilities by using energy-efficient lighting and automated systems.

- User and Stakeholder Engagement:
  - Feedback Mechanism: Implement a robust feedback mechanism to gather insights from users and stakeholders. Use this feedback to continuously improve the system.
  - Educational Campaigns: Conduct educational campaigns to inform the public about the benefits of the smart parking system and how to use it effectively.
- Data Privacy and Compliance:
  - Data Encryption: Ensure that all data collected and transmitted is encrypted to protect user privacy.
  - Regulatory Compliance: Stay updated with local and international data protection regulations (e.g., GDPR) and ensure that the system complies with all relevant laws.
- Community and Business Integration:
  - Partnerships with Local Businesses: Establish partnerships with local businesses to offer discounts or incentives for using the smart parking system.
  - Community Programs: Develop community programs to promote the adoption of smart parking solutions and encourage responsible parking habits.
- Remote Monitoring and Maintenance:
  - Remote Diagnostics: Implement remote diagnostics and maintenance capabilities to quickly identify and resolve issues with the parking system.
  - 24/7 Support: Provide 24/7 customer support to assist users with any problems or queries related to the smart parking system.

- Global Expansion and Adaptation:

  - International Adaptation: Adapt the smart parking system to meet the needs of different cities and countries, taking into account local regulations, cultural differences, and urban infrastructure.
  - Scalability for Large Cities: Ensure the system is scalable to handle the demands of larger cities with higher vehicle densities and more complex traffic patterns.

REFERENCES

1. Eco Park: A Low-Cost and Sustainable Approach for Smart Parking Systems Yash M Dalal;D R Kumar Raja;Um Ashwin Kumar 2023 World Conference on Communication & Computing (WCONF) Year: 2023
2. Smart parking in the smart city application Jan Šilar;Jirí Růžička;Zuzana Bělinovà;Martin Langr;Kristýna Hlubučková 2018 Smart City Symposium Prague (SCSP) Year: 2018
3. Towards a Smart Parking Management System for Smart Cities Paul Melnyk;Soufiene Djahel;Farid Nait-Abdesselam 2019 IEEE International Smart Cities Conference (ISC2) Year: 2019
4. An IoT-based Eco-Parking System for Smart Cities Ibrahim Tamam;Shen Wang;Soufiene Djahel 2020 IEEE International Smart Cities Conference (ISC2) Year: 2020
5. An IoT based Smart Outdoor Parking System S GokulKrishna;J Harsheetha;S Akshaya;D Jeyabharathi 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS) Year: 2021
6. Smart parking system with pre & post reservation, billing and traffic app Gayatri N. Hainalkar;Mousami S. Vanjale 2017 International Conference on Intelligent Computing and Control Systems (ICICCS) Year: 2017
7. Smart Parking: Novel Framework of Secure Smart Parking Solution using 5G Technology Aamir Anwar;Ijaz-ul-Haq;Nagham Saeed;Parisa Saadati 2021 IEEE International Smart Cities Conference (ISC2) Year: 2021
8. Smart Parking System (S-Park) – A Novel Application to Provide Real-Time Parking Solution Abhijeet Anand;Abhinav Kumar;A N Mukunda Rao;Anupam Ankesh;Ankur Raj 2020 Third International Conference on Multimedia Processing, Communication & Information Technology (MPCIT) Year: 2020
9. A navigation and reservation based smart parking platform using genetic optimization for smart cities Ilhan Aydin;Mehmet Karakose;Ebru Karakose 2017 5th International Istanbul Smart Grid and Cities Congress and Fair (ICSG) Year: 2017

10. Project of an Intelligent Recommender System for Parking Vehicles in Smart Cities Yuriy Pankiv;Nataliia Kunanets;Olga Artemenko;Nataliia Veretennikova;Ruslan Nebesnyi 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) Year: 2021
11. Smart Parking System for Monitoring Cars and Wrong Parking Faris Alshehri;A. H. M. Almawgani;Ayed Alqahtani;Abdurahman Alqahtani 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS) Year: 2019
12. Low cost smart parking system for smart cities D. Vakula;Yeshwanth Krishna Kolli
13. 2017 International Conference on Intelligent Sustainable Systems (ICISS) Year: 2017
14. ITS for Smart Parking Systems, towards the creation of smart city services using IoT and cloud approaches Luis Felipe Herrera-Quintero;Julián Vega-Alfonso;Diego Bermúdez;Luis Andres Marentes;Klaus Banse 2019 Smart City Symposium Prague (SCSP) Year: 2019
15. PARKIT: An Android-based Real Time Smart Parking System using IoT Shruti Agarwal;Shagun Gupta;Piyush Agarwal;Garima Sharma;Vikas Tripathi 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom) Year: 2023
16. Relocating On-Street Parking to Smart Parking Structure for Optimizing the Commercial Shared Street in Yeonnam-Dong, Seoul Ian Pranita 2023 10th International Conference on ICT for Smart Society (ICISS) Year: 2023

Book References

1. "Python Crash Course" by Eric Matthes - Provides a comprehensive introduction to Python programming, covering fundamental concepts and practical examples.
2. "Learning MySQL" by Robin Nixon - Offers a beginner-friendly guide to MySQL database management, covering database design, querying, and administration.

3. "WampServer: A Step-by-Step Guide" by Dr. Thomas D. Snyder - Provides a detailed walkthrough of installing, configuring, and using WampServer for local web development environments.
4. "Bootstrap 4 Quick Start" by Jacob Lett - Offers a quick and practical guide to getting started with Bootstrap 4, covering its key features and components.
5. "Flask Web Development" by Miguel Grinberg - A comprehensive guide to web development using Flask, covering topics such as routing, templates, forms, databases, and deployment strategies.
6. "Python Crash Course" by Eric Matthes - An essential resource for beginners to Python programming, covering core concepts and practical exercises.
7. "Learning MySQL" by Robin Nixon - A comprehensive guide to MySQL database management, offering insights into database design, querying, and administration.
8. "WampServer: A Step-by-Step Guide" by Dr. Thomas D. Snyder - A detailed manual for installing, configuring, and utilizing WampServer as a local web development environment.
9. "Bootstrap 4 Quick Start" by Jacob Lett - A concise introduction to Bootstrap 4, providing guidance on building responsive and visually appealing web interfaces.
10. "Flask Web Development" by Miguel Grinberg - A definitive guide to web development with Flask, covering routing, templates, forms, databases, and deployment strategies.
11. "Python for Data Analysis" by Wes McKinney - An in-depth exploration of using Python for data analysis tasks, including manipulation, visualization, and statistical analysis.
12. "MySQL Cookbook" by Paul DuBois - A collection of practical solutions and examples for common MySQL tasks, ranging from basic queries to advanced database management techniques.

Web References

1. Python Tutorial for Beginners. Retrieved from w3schools.com
2. Flask Mega-Tutorial by Miguel Grinberg. Retrieved from blog.miguelgrinberg.com

3. MySQL Workbench Documentation. Retrieved from dev.mysql.com
4. WampServer Official Website. Retrieved from wampserver.com
5. Bootstrap Documentation. Retrieved from getbootstrap.com
6. Flask Documentation. Retrieved from flask.palletsprojects.com