Online Parking System

THESIS OF
MINI PROJECT

BACHELOR OF COMPUTER APPLICATIONS

SUBMITTED BY
SHUBHAM MISHRA
Batch Year -2022-25
Enrollment No.-M2246020

Date of Submission-26/11/2024

PROJECT GUIDE -Mr. Gaurav Srivastava

C.M.P Degree College
(Constituent of University of Allahabad,Prayagraj)

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to my department “Department of Computer Application” and my college “C.M.P Degree college” for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I would like to express my deepest gratitude to all those who have supported and contributed to the successful completion of the Online Parking System Website project.

First and foremost, I extend my heartfelt thanks to my mentor “Gaurav sir” for their invaluable guidance, encouragement, and insightful feedback throughout the development of this project. Their expertise and support have been instrumental in shaping the direction and success of this endeavour.

Finally, I would like to acknowledge the support of my Family and Friends, whose unwavering encouragement and understanding have been a source of motivation throughout this journey.

Thank you all for your contributions and support.

-

DECLARATION

I Shubham Mishra, a student of Bachelor of Computer Application, hereby declare that the thesis titled "Online Parking System" submitted to my department “Department of Computer Application” is a genuine work done by me under the guidance of “Gaurav Srivastava”.

I certify that all the information and data presented in this synopsis are true and original to the best of my knowledge. Any external sources used have been duly acknowledged.

I affirm that the content and functionalities of this website are developed with the intention of enhancing learning and promoting a deeper understanding of codes, and I take full responsibility for the integrity and accuracy of the information provided.

Table of Contents

Preface

The rapid urbanization and increasing number of vehicles have led to severe parking problems in cities worldwide. Traditional parking systems often result in wasted time, traffic congestion, and environmental pollution. To address these challenges, an innovative solution is needed to streamline parking management and enhance the overall parking experience.

This online parking system aims to provide a comprehensive solution to the parking woes of urban dwellers. By leveraging advanced technologies such as the Internet of Things (IoT) and mobile applications, this system offers real-time information on available parking spaces, automated payment systems, and intelligent parking guidance.

This document outlines the technical details, implementation strategies, and potential benefits of this online parking system. It is intended to serve as a valuable resource for developers, policymakers, and urban planners interested in improving parking infrastructure and enhancing urban mobility.

We believe that this system has the potential to significantly improve the quality of life in urban areas by reducing traffic congestion, saving time, and promoting sustainable transportation.

INTRODUCTION

The Online Parking System is a modern solution designed to streamline the process of managing parking spaces in urban areas, airports, malls, offices, and other crowded locations. With the increasing number of vehicles on the road and limited parking availability, traditional parking management methods often lead to inefficiencies, long waiting times, and user frustration. This system leverages technology to address these challenges, offering a seamless and user-friendly experience for both parking administrators and vehicle owners.

At its core, an Online Parking System allows users to reserve parking slots in advance through a digital platform such as a website or mobile application. It provides real-time information about available spaces, pricing, and location, enabling users to make informed decisions before they arrive at their destination. This not only reduces the hassle of finding parking but also minimizes traffic congestion caused by vehicles searching for spaces.

From an administrative perspective, the system offers tools to efficiently manage parking operations providing accurate monitoring of space occupancy and ensuring efficient utilization of resources.

The Online Parking System also contributes to environmental sustainability by reducing fuel consumption and emissions associated with prolonged vehicle idling. Additionally, it improves safety by eliminating the need for physical cash transactions and reducing the risks of overcrowding in parking areas.

In a world increasingly reliant on technology, this system represents a significant step forward in smart city initiatives. It offers convenience, transparency, and efficiency, aligning with the needs of modern urban living. Whether for individual drivers or large-scale parking facilities, the Online Parking System is an innovative approach that simplifies the parking experience while promoting better resource management and environmental care.

By integrating user-focused features with cutting-edge technology, the Online Parking System addresses the challenges of traditional parking management, paving the way for a smarter, more efficient future.

OBJECTIVES

The online parking system aims to achieve the following objectives:

- **Enhance Parking Efficiency**: Minimize the time and effort required for drivers to find available parking spaces.

- **Reduce Traffic Congestion**: Decrease the number of vehicles circulating in search of parking, leading to improved traffic flow.

- **Improve Air Quality**: Reduce vehicular emissions associated with excessive searching for parking.

- **Provide Convenience**: Offer a user-friendly platform for parking reservations and payments.

- **Optimize Parking Infrastructure**: Gather data on parking usage patterns to inform better urban planning and resource allocation.

- **Promote Sustainable Transportation**: Encourage the use of public transportation or carpooling by providing efficient parking options.

BENEFITS

1.Convenience and Accessibility

- An online parking system allows users to easily find and book parking spaces from their computers or mobile devices. This eliminates the hassle of driving around looking for a spot, saving both time and stress.

2. Real-Time Availability

- Users can view real-time availability of parking spots, ensuring they only book spaces that are currently available. This reduces frustration and optimizes the parking experience.

3.Advanced Reservation

- The ability to book parking spots in advance helps users plan their visits more effectively, especially in busy areas or during peak times. This feature is particularly beneficial for events, airports, and popular destinations.

4.Enhanced Security

- The Online parking systems often include features like secure payment gateways and user authentication, providing a safer transaction experience. Additionally, many systems offer real-time monitoring and security features for the parking facility itself.

5.Reduced Stress

By eliminating the hassle of finding parking, users can reduce stress and improve their overall experience.

HARDWARE AND SOFTWARE REQUIREMENTS

This proposed software runs effectively on a computing system that has the minimum requirements. Undertaking all the equipment necessities are not satisfied but rather exist in their systems administration between the user's machines already. So, the main need is to introduce appropriate equipment for the product. The requirements are split into two categories, namely:

SOFTWARE REQUIREMENTS

The basic software requirements to run the program are:

- Windows 7
- Visual Studio Code
- HTML, CSS, JavaScript, Bootstrap
- Browser example Mozilla Firefox, Safari, Chrome, etc.

HARDWARE REQUIREMENTS

The basic hardware required to run the program are:

- Hard disk of 5 GB
- System memory (RAM) of 512 MB
- I3 processor-based computer or higher.

METHODOLOGY

1. System Requirements Analysis:

Functional Requirements:

1. User registration and login
2. Vehicle registration
3. Parking slot booking and cancellation
4. Real-time parking slot availability
5. Parking duration tracking and billing
6. Admin panel for managing parking lots, users,etc

Non-Functional Requirements:

1. Security and privacy
2. Scalability
3. Performance
4. User-friendly interface
5. Reliability

2. System Design:

Database Design:

1. User table: Store user information (name, email, phone number, password)
2. Vehicle table: Store vehicle information (registration number, vehicle type)
3. Parking lot table: Store parking lot information (location, capacity, pricing)
4. Booking table: Store booking information (user ID, vehicle

ID, parking lot ID, start time, end time,)

System Architecture:

Frontend:

1. User interface for booking, payment, and account management
2. Responsive design for mobile and desktop

Secure authentication and authorization

Backend:

1. Server-side logic for handling user requests, database interactions, and payment processing
2. API endpoints for communication between frontend and backend
3. Real-time updates for parking slot availability

Database:

1.Store user data, vehicle information, parking lot details, and booking history

2. Efficient data retrieval and storage

3. Development:

Technology Stack:

Frontend: HTML ,CSS ,JAVASCRIPT

Backend: Node.js , Expressjs , MongoDB.

Database: MongoDB

Cloud Platform: AWS, Azure, or Google Cloud Platform

Implementation:

1. Develop user interface components (login, registration, booking, payment)
2. Implement backend logic for user authentication, booking management, payment processing, and real-time updates
3. Integrate payment gateway (e.g., Stripe, PayPal)

Deploy the application to a cloud platform or server

5. Maintenance and Support:

1.Monitoring: Monitor system performance, identify and fix bugs

2.Updates: Update the system with new features and security patches

3.Support: Provide technical support to users

Additional Considerations:

1.Security: Implement robust security measures to protect user data and prevent unauthorized access.

2.Scalability: Design the system to handle increasing user traffic and data volume.

3.User Experience: Create a user-friendly interface with intuitive navigation and clear instructions.

4.Real-time Updates: Use technology like WebSocket or Server- Sent Events to provide real-time updates on parking slot availability.

Testing Strategy

A robust testing strategy is essential to ensure the quality and reliability of an online parking system. Unit and integration testing are two fundamental types of testing that play crucial roles in this process.

Unit Testing

1. Purpose: To test individual units of code (functions, classes, modules) in isolation.

2. Focus: Verifying that each unit functions correctly and produces the expected output for given inputs.

Techniques:

1.White-box testing: Testing based on the internal structure and logic of the code.

2. Black-box testing: Testing based on the input and output behavior of the unit, without considering its internal implementation.

Integration Testing

1.Purpose: To test the interaction between different units or components of the system.

2.Focus: Ensuring that components work together seamlessly and communicate effectively.

Techniques:

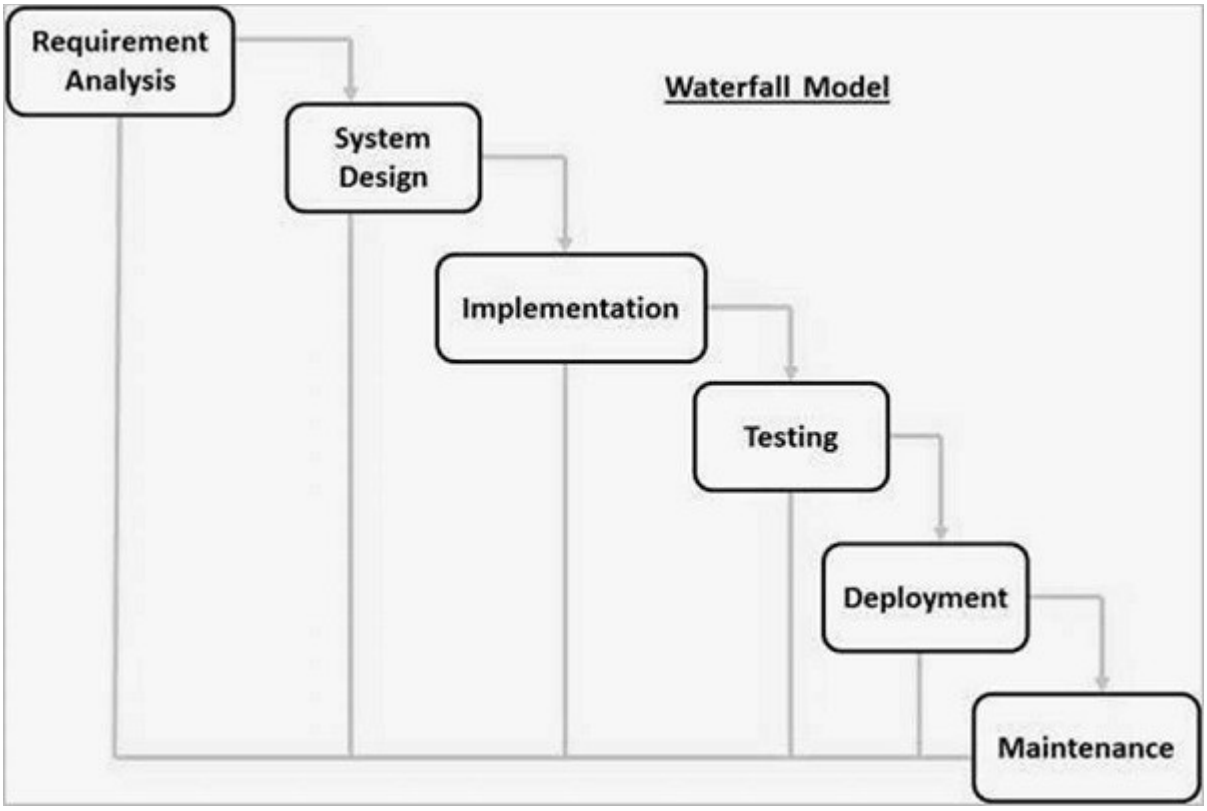
1. Top-down integration: Testing starts with the top-level modules and gradually moves down to lower-level modules.

2. Bottom-up integration: Testing starts with the lowest-level modules and gradually moves up to higher-level modules.

3.Big-bang integration: Testing all modules together at once.

Software Process Model

The Waterfall Model is a linear and sequential approach to software development, where each phase must be completed before the next one begins. Here's how it can be applied to an Online Parking System Project:



1. Requirements Analysis

In this phase, all requirements of the online parking system are gathered and documented in detail. This includes:

Functional requirements: User registration and login.
Parking slot availability checking.
Slot booking and cancellation.
Admin panel for managing parking slots.

Non-functional requirements: System performance.
Scalability for increasing users.
Data security (e.g., user data).

Constraints: Operating systems, browser support, etc.

Deliverable: Software Requirement Specification (SRS) document.

2. **System Design**

Here, the architecture of the system is planned, including:

High-Level Design (HLD): Design of modules: User, Admin, Parking Slot.
Low-Level Design (LLD):
Database schema for user profiles, booking records, etc.

3. **Implementation**

In this phase, coding begins based on the design specifications:

Frontend:
Create user interface components for registration, login, booking, etc.
Backend:
Develop APIs for booking, and admin management.

Database:
Implement the database structure.
Deliverable: Fully developed system.

4. Deployment

The tested system is deployed in the production environment:

Hosting the system on a server or cloud platform.
Setting up the domain for online access.
Configuring SSL for secure communication.
Deliverable: Fully deployed and accessible online parking system.

5. Maintenance
Post-deployment, the system is monitored and updated:

Fixing bugs reported by users.
Adding new features (e.g., loyalty programs).
Ensuring compatibility with new technologies (browsers, devices).
Deliverable: Regular updates and maintenance logs.

Advantages of the Waterfall Model

Clear Structure: The Waterfall model is straightforward and easy to understand, with well-defined phases.
Easy Management: Project management is simplified due to the linear nature of the model.
Well-suited for Smaller Projects: For small projects with well-defined requirements, the Waterfall model can be efficient.
Systematic Approach: It provides a structured approach to software development, ensuring that each phase is completed before moving to the next.
Clear Milestones: Each phase has specific deliverables and deadlines, making it easier to track progress.

Disadvantages of the Waterfall Model

Rigid Approach: Once a phase is completed, it's difficult to go back and make significant changes, limiting flexibility.
Risk of Errors: Errors may not be detected until later stages, leading to costly fixes.
Limited Customer Involvement: Customer feedback is often limited to the initial requirements gathering phase.
Longer Development Time: The sequential nature of the model can lead to longer development cycles.
Less Adaptable to Change: The Waterfall model struggles to accommodate changing requirements or unforeseen challenges.

LANGUAGES USED FOR FRONT-END

HTML*: -*

HTML, or Hypertext Markup Language, is the standard language used to create and design web pages and web applications. It provides the basic structure for web content by defining elements such as headings, paragraphs, links, images, and other types of media. HTML is a cornerstone technology of the World Wide Web, alongside CSS (Cascading Style Sheets) and JavaScript.

USES OF HTML: -

- **Creating Web Page Structure:** HTML provides the fundamental structure of a web page, defining elements such as headings, paragraphs, links, images, tables, and forms. It lays out the content and its organization.

- **Embedding Media:** HTML allows for the inclusion of various media types, including images, videos, and audio. Tags like <img>, <video>, and <audio> are used to incorporate these elements.

- **SEO (Search Engine Optimization):** HTML tags such as <title>, <meta>, and semantic tags (e.g., <header>, <footer>, <article>) help in improving a website's search engine ranking by providing meaningful content structure.

CSS*: -*

CSS (Cascading Style Sheets) is a style sheet language used to control the presentation and layout of HTML elements on a web page. It separates the content of a web page (written in HTML) from its visual and aural layout, allowing developers to apply styles to multiple pages simultaneously and maintain a consistent look and feel across a website.

Uses of CSS

- **Styling Web Pages**: CSS is used to apply colors, fonts, spacing, and layout designs, enhancing the visual appeal of web pages.
- **Responsive Design**: Enables websites to adapt to different screen sizes and devices using media queries.
- **Animations and Effects**: Adds animations, transitions, and visual effects to elements for interactive user experiences.
- **Customizing UI Components**: Styles buttons, forms, menus, and other interface elements.
- **Separation of Content and Design**: Keeps HTML structure clean by handling design elements separately in a CSS file.

JavaScript (JS): -

JavaScript is a high-level, versatile programming language primarily used for creating dynamic and interactive content on websites. It is an essential technology of web development, alongside HTML and CSS. While HTML provides the structure and CSS the style, JavaScript adds behaviour and functionality, enabling web pages to respond to user interactions and provide a more engaging user experience.

Uses of JavaScript

- **Interactivity**: Adds dynamic behavior, such as toggling menus, form validation, and modals, making web pages interactive.
- **DOM Manipulation**: Modifies HTML and CSS dynamically, such as updating content without refreshing the page.
- **Event Handling**: Responds to user actions like clicks, scrolls, and key presses.
- **Asynchronous Operations**: Manages tasks like data fetching from servers using APIs or AJAX without reloading the page.
- **Game Development**: Creates simple web-based games or interactive visualizations by combining logic and graphics.

“By combining CSS, JavaScript, and React, you can create rich, interactive web applications with well-organized, maintainable code.”

LANGUAGES USED FOR BACK-END

Express.js: -

Express.js, commonly known as Express, is a fast, minimalist web application framework for Node.js. It is designed to simplify the development of server-side applications and APIs by providing a robust set of features for building web and mobile applications. Express.js is one of the most popular frameworks for Node.js and is widely used in web development due to its simplicity, flexibility, and extensive ecosystem.

MongoDB*:* -

MongoDB is a popular open-source NoSQL database designed for storing, retrieving, and managing large volumes of unstructured or semi-structured data. Unlike traditional relational databases, which use structured tables with predefined schemas, MongoDB uses a flexible, document-oriented approach to data storage. It is known for its scalability, high performance, and ease of use, making it suitable for modern applications that require dynamic and diverse data handling.

“By combining **HTML** for structure, **CSS** for styling, **JavaScript** for interactivity, **React.js** for dynamic components, and **Express.js** with **MongoDB** for backend and data management, we can build a powerful and engaging website.”

Tool Used

Visual Studio is an Integrated Development Environment(IDE) developed by Microsoft to develop Desktop applications, GUI(Graphical User Interface), console, web applications, mobile applications, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write code in C#, C++, VB(Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

Advantages Of Visual Studio:

• A full-featured programming platform for several operating systems, the web, and the cloud, Visual Studio IDE is available. Users can easily browse the UI so they can write their code quickly and precisely.

• To help developers quickly identify potential errors in the code, Visual Studio offers a robust debugging tool.

• Developers can host their application on the server with confidence because they have eliminated anything that could lead to performance issues.

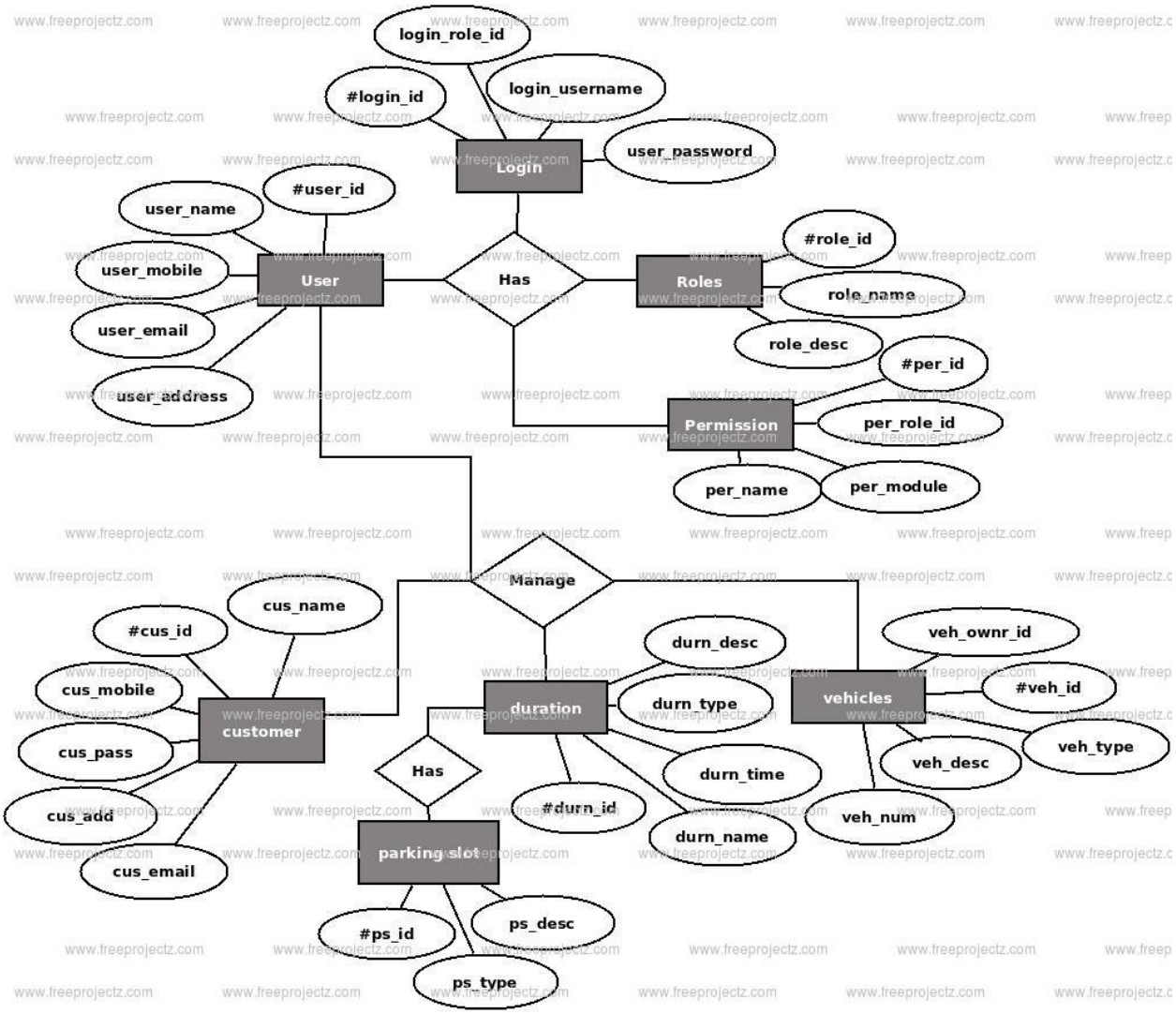
• No matter what programming language developers are using, users of Visual Studio can get live coding support. For faster development, the Platform offers an autocomplete option. The built-in intelligent system offers descriptions and tips for APIs.

• Through Visual Studio IDE you can easily collab with your teammates in a same project. This IDE helps the developers to share, push and pull their code with their teammates.

• Every user of Visual Studio has the ability to customize it. They have the option to add features based on their needs. For example, they can download add-ons and install extensions in their IDE. Even programmers can submit their own extensions.

ER DIAGRAM

-



ER Diagram For Parking System

Coding

Bike.js

```
class Entry {
    constructor(owner, bike, licensePlate, entryDate, exitDate) {
        this.owner = owner;
        this.bike = bike;
        this.licensePlate = licensePlate;
        this.entryDate = entryDate;
        this.exitDate = exitDate;
    }
}

class UI {
    static displayEntries() {
        const entries = Store.getEntries();
        entries.forEach((entry) => UI.addEntryToTable(entry));
    }

    static addEntryToTable(entry) {
        const tableBody = document.querySelector('#tableBody');
        const row = document.createElement('tr');
        row.innerHTML = `
            <td>${entry.owner}</td>
            <td>${entry.bike}</td>
            <td>${entry.licensePlate}</td>
            <td>${entry.entryDate}</td>
```

```
        <td>${entry.exitDate}</td>
        <td><button class="btn btn-danger delete">X</button></td>
    `;
    tableBody.appendChild(row);
}

static clearInput() {
    const inputs = document.querySelectorAll('.form-control');
    inputs.forEach((input) => input.value = "");
}

static deleteEntry(target) {
    if (target.classList.contains('delete')) {
        target.parentElement.parentElement.remove();
    }
}

static showAlert(message, className) {
    const div = document.createElement('div');
    div.className = alert alert-${className} w-50 mx-auto;
    div.appendChild(document.createTextNode(message));
    const formContainer = document.querySelector('.form-container');
    const form = document.querySelector('#entryForm');
    formContainer.insertBefore(div, form);
    setTimeout(() => document.querySelector('.alert').remove(), 3000);
}

static validateInputs() {
    const owner = document.querySelector('#owner').value;
    const bike = document.querySelector('#car').value;
```

```
        const licensePlate = document.querySelector('#licensePlate').value;
        const entryDate = document.querySelector('#entryDate').value;
        const exitDate = document.querySelector('#exitDate').value;
        const licensePlateRegex = /^(?:[A-Z]{2}-\d{2}-\d{2})|(?:\d{2}-[A-Z]{2}-
\d{2})|(?:\d{2}-\d{2}-[A-Z]{2})$/;


        if (owner === '' || bike === '' || licensePlate === '' || entryDate === '' || exitDate
=== '') {
            UI.showAlert('All fields must be filled!', 'danger');
            return false;
        }
        if (exitDate < entryDate) {
            UI.showAlert('Exit Time cannot be before Entry Time', 'danger');
            return false;
        }
        if (!licensePlateRegex.test(licensePlate)) {
            UI.showAlert('License Plate must be in the format NN-NN-LL, NN-LL-
NN, LL-NN-NN', 'danger');
            return false;
        }
        return true;
    }
}

class Store {
    static getEntries() {
        let entries;
        if (localStorage.getItem('entries') === null) {
            entries = [];
        } else {
            entries = JSON.parse(localStorage.getItem('entries'));
```

```
        }
        return entries;
    }

    static addEntries(entry) {
        const entries = Store.getEntries();
        entries.push(entry);
        localStorage.setItem('entries', JSON.stringify(entries));
    }

    static removeEntries(licensePlate) {
        const entries = Store.getEntries();
        entries.forEach((entry, index) => {
            if (entry.licensePlate === licensePlate) {
                entries.splice(index, 1);
            }
        });
        localStorage.setItem('entries', JSON.stringify(entries));
    }
}

document.addEventListener('DOMContentLoaded', UI.displayEntries);

document.querySelector('#entryForm').addEventListener('submit', (e) => {
    e.preventDefault();


    const owner = document.querySelector('#owner').value;
    const bike = document.querySelector('#car').value;
    const licensePlate = document.querySelector('#licensePlate').value;
```

```
    const entryDate = document.querySelector('#entryDate').value;
    const exitDate = document.querySelector('#exitDate').value;


    if (!UI.validateInputs()) {
        return;
    }

    const entry = new Entry(owner, bike, licensePlate, entryDate, exitDate);

    UI.addEntryToTable(entry);
    Store.addEntries(entry);

    UI.clearInput();

    UI.showAlert('Bike successfully added to the parking lot', 'success');
});

document.querySelector('#tableBody').addEventListener('click', (e) => {

    UI.deleteEntry(e.target);

    const licensePlate =
e.target.parentElement.previousElementSibling.previousElementSibling.previousE
lementSibling.textContent;
    Store.removeEntries(licensePlate);

    UI.showAlert('Bike successfully removed from the parking lot list', 'success');
});
```

```
document.querySelector('#searchInput').addEventListener('keyup', function
searchTable() {
    const searchValue =
document.querySelector('#searchInput').value.toUpperCase();
    const tableLines = document.querySelector('#tableBody').querySelectorAll('tr');

    tableLines.forEach((line) => {
        let count = 0;
        const lineValues = line.querySelectorAll('td');

        lineValues.forEach((cell, index) => {
            if (index < lineValues.length - 1 &&
cell.innerHTML.toUpperCase().startsWith(searchValue)) {
                count++;
            }
        });

        line.style.display = count > 0 ? '' : 'none';
    });
});
```

Cycle.js

```
class Entry {
    constructor(owner, tokenNumber, entryDate, exitDate) {
        this.owner = owner;
        this.tokenNumber = tokenNumber;
        this.entryDate = entryDate;
        this.exitDate = exitDate;
    }
}

class Store {
    static getEntries() {
        let entries;
        if (localStorage.getItem('entries') === null) {
            entries = [];
        } else {
            entries = JSON.parse(localStorage.getItem('entries'));
        }
        return entries;
    }

    static addEntries(entry) {
        const entries = Store.getEntries();
        entries.push(entry);
        localStorage.setItem('entries', JSON.stringify(entries));
    }

    static removeEntries(tokenNumber) {
```

```
        const entries = Store.getEntries();
        const updatedEntries = entries.filter((entry) => entry.tokenNumber !==
tokenNumber);
        localStorage.setItem('entries', JSON.stringify(updatedEntries));
    }
}

class UI {
    static displayEntries() {
        const entries = Store.getEntries();
        entries.forEach((entry) => UI.addEntryToTable(entry));
    }

    static addEntryToTable(entry) {
        const tableBody = document.querySelector('#tableBody');
        const row = document.createElement('tr');
        row.innerHTML = `
            <td>${entry.owner}</td>
            <td>${entry.tokenNumber}</td>
            <td>${entry.entryDate}</td>
            <td>${entry.exitDate}</td>
            <td><button class="btn btn-danger delete">Delete</button></td>
        `;
        tableBody.appendChild(row);
    }

    static clearInput() {
        const inputs = document.querySelectorAll('.form-control');
        inputs.forEach((input) => input.value = "");
    }
```

```
static deleteEntry(target) {
    if (target.classList.contains('delete')) {
        target.parentElement.parentElement.remove();
    }
}

static showAlert(message, className) {
    const currentAlert = document.querySelector('.alert');
    if (currentAlert) {
        currentAlert.remove();
    }


    const div = document.createElement('div');
    div.className = alert alert-${className} text-center;
    div.appendChild(document.createTextNode(message));


    const formContainer = document.querySelector('.form-container');
    const form = document.querySelector('#entryForm');
    formContainer.insertBefore(div, form);


    setTimeout(() => {
        if (document.querySelector('.alert')) {
            document.querySelector('.alert').remove();
        }
    }, 3000);
}
```

```
    static validateInputs() {
        const owner = document.querySelector('#owner').value;
        const tokenNumber = document.querySelector('#licensePlate').value;
        const entryDate = document.querySelector('#entryDate').value;
        const exitDate = document.querySelector('#exitDate').value;

        if (owner === '' || tokenNumber === '' || entryDate === '' || exitDate === '') {
            UI.showAlert('All fields must be filled!', 'danger');
            return false;
        }
        if (exitDate < entryDate) {
            UI.showAlert('Exit Time cannot be before Entry Time', 'danger');
            return false;
        }
        return true;
    }
}

document.addEventListener('DOMContentLoaded', UI.displayEntries);


document.querySelector('#entryForm').addEventListener('submit', (e) => {
    e.preventDefault();

    const owner = document.querySelector('#owner').value;
    const tokenNumber = document.querySelector('#licensePlate').value;
    const entryDate = document.querySelector('#entryDate').value;
    const exitDate = document.querySelector('#exitDate').value;
```

```
    if (!UI.validateInputs()) {
        return;
    }

    const entry = new Entry(owner, tokenNumber, entryDate, exitDate);

    UI.addEntryToTable(entry);
    Store.addEntries(entry);
    UI.clearInput();

    UI.showAlert('Cycle successfully added to the parking lot', 'success');
});



document.querySelector('#tableBody').addEventListener('click', (e) => {
    UI.deleteEntry(e.target);

    const tokenNumber =
e.target.parentElement.previousElementSibling.previousElementSibling.previousE
lementSibling.textContent;
    Store.removeEntries(tokenNumber);

    UI.showAlert('Cycle successfully removed from the parking lot list', 'success');
});



document.querySelector('#searchInput').addEventListener('keyup', function
searchTable() {
    const searchValue =
document.querySelector('#searchInput').value.toUpperCase();
    const tableLines = document.querySelector('#tableBody').querySelectorAll('tr');
```

```
    tableLines.forEach((line) => {
        let count = 0;
        const lineValues = line.querySelectorAll('td');

        lineValues.forEach((cell, index) => {
            if (index < lineValues.length - 1 &&
cell.innerHTML.toUpperCase().startsWith(searchValue)) {
                count++;
            }
        });

        line.style.display = count > 0 ? '' : 'none';
    });
});
```

Car.js

```
class Entry{
    constructor(owner,car,licensePlate,entryDate,exitDate){
        this.owner = owner;
        this.car = car;
        this.licensePlate = licensePlate;
        this.entryDate = entryDate;
        this.exitDate = exitDate;
    }
}

class UI{
    static displayEntries(){

        const entries = Store.getEntries();
        entries.forEach((entry) => UI.addEntryToTable(entry));
    }
    static addEntryToTable(entry){
        const tableBody=document.querySelector('#tableBody');
        const row = document.createElement('tr');
        row.innerHTML = `   <td>${entry.owner}</td>
                            <td>${entry.car}</td>
                            <td>${entry.licensePlate}</td>
                            <td>${entry.entryDate}</td>
                            <td>${entry.exitDate}</td>
                            <td><button class="btn btn-danger delete">X</button></td>
```

```
                `;
        tableBody.appendChild(row);
    }
    static clearInput(){

        const inputs = document.querySelectorAll('.form-control');

        inputs.forEach((input)=>input.value="");
    }
    static deleteEntry(target){
        if(target.classList.contains('delete')){
            target.parentElement.parentElement.remove();
        }
    }
    static showAlert(message,className){
        const div = document.createElement('div');
        div.className=`alert alert-${className} w-50 mx-auto`;
        div.appendChild(document.createTextNode(message));
        const formContainer = document.querySelector('.form-container');
        const form = document.querySelector('#entryForm');
        formContainer.insertBefore(div,form);
        setTimeout(() => document.querySelector('.alert').remove(),3000);
    }
    static validateInputs(){
        const owner = document.querySelector('#owner').value;
        const car = document.querySelector('#car').value;
        const licensePlate = document.querySelector('#licensePlate').value;
        const entryDate = document.querySelector('#entryDate').value;
        const exitDate = document.querySelector('#exitDate').value;
        var licensePlateRegex = /^(?:[A-Z]{2}-\d{2}-\d{2})|(?:\d{2}-[A-Z]{2}-
\d{2})|(?:\d{2}-\d{2}-[A-Z]{2})$/;
```

```
        if(owner === '' || car === '' || licensePlate === '' || entryDate === '' || exitDate
=== ''){
            UI.showAlert('All fields must me filled!','danger');
            return false;
        }
        if(exitDate < entryDate){
            UI.showAlert('Exit Time cannot be lower than Entry Time','danger');
            return false;
        }
        if(!licensePlateRegex.test(licensePlate)){
            UI.showAlert('License Plate must be like NN-NN-LL, NN-LL-NN, LL-
NN-NN','danger');
            return false;
        }
        return true;
    }
}

class Store{
    static getEntries(){
        let entries;
        if(localStorage.getItem('entries') === null){
            entries = [];
        }
        else{
            entries = JSON.parse(localStorage.getItem('entries'));
        }
        return entries;
    }
    static addEntries(entry){
        const entries = Store.getEntries();
```

```
        entries.push(entry);
        localStorage.setItem('entries', JSON.stringify(entries));
    }
    static removeEntries(licensePlate){
        const entries = Store.getEntries();
        entries.forEach((entry,index) => {
            if(entry.licensePlate === licensePlate){
                entries.splice(index, 1);
            }
        });
        localStorage.setItem('entries', JSON.stringify(entries));
    }
}

    document.addEventListener('DOMContentLoaded',UI.displayEntries);

    document.querySelector('#entryForm').addEventListener('submit',(e)=>{
        e.preventDefault();

        const owner = document.querySelector('#owner').value;
        const car = document.querySelector('#car').value;
        const licensePlate = document.querySelector('#licensePlate').value;
        const entryDate = document.querySelector('#entryDate').value;
        const exitDate = document.querySelector('#exitDate').value;
        if(!UI.validateInputs()){
            return;
        }
        const entry = new Entry(owner, car, licensePlate, entryDate, exitDate);
        UI.addEntryToTable(entry);
        Store.addEntries(entry);
```

```
        UI.clearInput();

        UI.showAlert('Car successfully added to the parking lot','success');

    });
    document.querySelector('#tableBody').addEventListener('click',(e)=>{
        UI.deleteEntry(e.target);
        var licensePlate =
e.target.parentElement.previousElementSibling.previousElementSibling.previousE
lementSibling.textContent;
        Store.removeEntries(licensePlate);
        UI.showAlert('Car successfully removed from the parking lot list','success');
    })

    document.querySelector('#searchInput').addEventListener('keyup', function
searchTable(){
        const searchValue =
document.querySelector('#searchInput').value.toUpperCase();
        const tableLine =
(document.querySelector('#tableBody')).querySelectorAll('tr');
        for(let i = 0; i < tableLine.length; i++){
            var count = 0;
            const lineValues = tableLine[i].querySelectorAll('td');
            for(let j = 0; j < lineValues.length - 1; j++){
                if((lineValues[j].innerHTML.toUpperCase()).startsWith(searchValue)){
                    count++;
                }
            }
            if(count > 0){
                tableLine[i].style.display = '';
            }else{
                tableLine[i].style.display = 'none';
```

```
        }
    }
});
```

home.js

```
function showOptions() {
        const vehicleType = document.getElementById("vehicleType").value;
        const twoWheelerOptions =
document.getElementById("twoWheelerOptions");
        const fourWheelerOptions =
document.getElementById("fourWheelerOptions");

        twoWheelerOptions.classList.add("hidden");
        fourWheelerOptions.classList.add("hidden");

        if (vehicleType === "two-wheeler") {
            twoWheelerOptions.classList.remove("hidden");
            twoWheelerOptions.classList.add("show");
        } else if (vehicleType === "four-wheeler") {
            fourWheelerOptions.classList.remove("hidden");
            fourWheelerOptions.classList.add("show");
        }

        toggleButtons();
    }

    function toggleButtons() {
        const twoWheelerValue =
document.getElementById("twoWheeler").value;
```

```
        const fourWheelerValue =
document.getElementById("fourWheeler").value;



        document.getElementById("cycleButton").style.display = "none";
        document.getElementById("motorcycleButton").style.display = "none";
        document.getElementById("carButton").style.display = "none";

        if (twoWheelerValue === "cycle") {
            document.getElementById("cycleButton").style.display = "block";
        } else if (twoWheelerValue === "motorcycle") {
            document.getElementById("motorcycleButton").style.display = "block";
        } else if (fourWheelerValue === "car") {
            document.getElementById("carButton").style.display = "block";
        }
    }

    const cycleButton=document.getElementById('cycleButton');
    const motorcycleButton=document.getElementById('motorcycleButton');
    const carButton=document.getElementById('carButton')

    cycleButton.addEventListener('click',()=>{
        window.location.href='/cycle'
    })



    motorcycleButton.addEventListener('click',()=>{
        window.location.href='/bike'
    })
```

```
carButton.addEventListener('click',()=>{
    window.location.href='/car'
})
```

index.js

```
const express = require("express");
const path = require("path");
const LogInCollection = require("./mongo");
const app = express();
const port = process.env.PORT || 3000;

app.use(express.json());
app.use(express.urlencoded({ extended: false }));

const templatePath = path.join(__dirname, "../templates");
const publicPath = path.join(__dirname, "../public");

app.use(express.static(publicPath));

app.set("view engine", "hbs");
app.set("views", templatePath);


app.get("/", (req, res) => res.redirect("/login"));
app.get("/signup", (req, res) => res.render("signup"));
```

```
app.get("/login", (req, res) => res.render("login"));


app.post("/signup", async (req, res) => {
    console.log(req.body);
    const { Username, Email, Password, MobileNo } = req.body;

    if (!Username || !Email || !Password || !MobileNo) {
        return res.status(400).render("signup", { error: "All fields are required." });
    }

    try {
        const existingUser = await LogInCollection.findOne({ Username });
        if (existingUser) {
            return res.render("signup", { error: "User already exists. Please log in." });
        }

        await LogInCollection.insertMany([{ Username, Email, Password,
MobileNo }]);
        return res.status(201).render("home", { naming: Username });
    } catch (error) {
        console.error("Error during signup:", error);
        return res.render("signup", { error: "Error occurred during signup." });
    }
});


app.post("/login", async (req, res) => {
    console.log(req.body);
    const { Username, Password } = req.body;
```

```
    if (!Username || !Password) {
        return res.status(400).render("login", { error: "Both fields are required." });
    }

    try {
        const user = await LogInCollection.findOne({ Username });
        if (user && user.Password === Password) {
            return res.status(201).render("home", { naming: Username });
        } else {
            return res.render("login", { error: "Invalid username or password." });
        }
    } catch (error) {
        console.error("Error during login:", error);
        return res.render("login", { error: "Error occurred during login." });
    }
});

app.get("/cycle", (req, res) => res.render("cycle"));
app.get("/bike", (req, res) => res.render("bike"));
app.get("/car", (req, res) => res.render("car"));


app.listen(port, () => console.log(`Server running on port ${port}`));
```

mongo.js

```
const mongoose=require("mongoose");

mongoose.connect("mongodb://localhost:27017/Parking_management")
```

```
.then(()=>{
    console.log('mongoose connected');
})
.catch((e)=>{
    console.log('failed');
})


const LogInSchema = new mongoose.Schema({
    Username: { type: String, required: true },
    Email: { type: String, required: true },
    Password: { type: String, required: true },
    MobileNo:{type:String,required:true}
});

const LogInCollection = mongoose.model('LogIn', LogInSchema);

module.exports = LogInCollection;
```

car.hbs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link href="https://fonts.googleapis.com/css?family=Roboto&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Parking Lot Management App</title>
</head>

<body action="/car" method="post">
    <header class="shadow">
        <div class="header-content d-flex justify-content-center p-2">
            <div id="header-msg" class="ml-5 align-self-center">Parking Lot
Management System</div>
        </div>
    </header>
    <div class="form-container mt-5">
        <form class="w-50 mx-auto" id="entryForm">
            <h5 class="text-center">Add Car to Parking Lot</h5>
            <div class="form-group">
                <label id="aa" for="owner">Owner:</label>
                <input type="text" class="form-control rounded-0 shadow-sm"
id="owner" placeholder="Owner">
            </div>
            <div class="form-group">
                <label id="aa"  for="car">Car:</label>
                <input type="text" class="form-control rounded-0 shadow-sm" id="car"
placeholder="Car">
            </div>
            <div class="form-group">
                <label id="aa"  for="licensePlate">License Plate:</label>
                <input type="text" class="form-control rounded-0 shadow-sm"
id="licensePlate" placeholder="NN-NN-LL,NN-LL-NN,....etc">
```

```
        </div>
        <div class="row">
          <div class="col-6">
            <label id="aa" for="entryDate">Entry Time:</label>
            <input type="time" class="form-control rounded-0 shadow-sm"
id="entryDate">
          </div>
          <div class="col-6">
            <label id="aa"  for="exitDate">Exit Time:</label>
            <input type="time" class="form-control rounded-0 shadow-sm"
id="exitDate">
          </div>
        </div>
        <button type="submit" style="color: rgb(110, 119, 206);;" class="btn mx-
auto d-block mt-5 rounded-0 shadow" id="btnOne">Add Car</button>
      </form>
  </div>
  <div class="table-container mt-5 mb-5 w-75 mx-auto" >
        <h5 class="text-center mb-3">List of Cars in Parking Lot</h5>
        <input type="text" class="w-100 mb-3" id="searchInput"
placeholder="Search...">
        <table class="table table-striped shadow " id="parkingTable">
            <thead class="text-black" id="tableHead">
              <tr>
                <th scope="col">Owner</th>
                <th scope="col">Car</th>
                <th scope="col">License Plate</th>
                <th scope="col">Entry Time</th>
                <th scope="col">Exit Time</th>
                <th scope="col">Actions</th>
              </tr>
```

```
            </thead>
            <tbody id="tableBody">

            </tbody>
          </table>
    </div>

    <script src="script.js"></script>
</body>

</html>
```

home.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Parking Lot System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: linear-gradient(135deg, #74ebd5, #acb6e5);
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
```

```
    height: 100vh;
    transition: background-color 0.5s ease;
}

h1 {
    color: #333;
    transition: transform 0.5s ease, color 0.3s ease;
    margin-bottom: 20px;
}

h1:hover {
    color: #007bff;
    transform: scale(1.05);
}

.container {
    background-color: white;
    border-radius: 15px;
    box-shadow: 0 6px 30px rgba(0, 0, 0, 0.15);
    padding: 25px;
    width: 320px;
    text-align: center;
    transition: transform 0.5s ease, box-shadow 0.3s ease;
    position: relative;
}

.container:hover {
    transform: scale(1.05);
    box-shadow: 0 10px 40px rgba(0, 0, 0, 0.25);
}
```

```
select {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    margin: 10px 0;
    transition: border-color 0.3s ease, background-color 0.3s ease;
}

select:focus {
    border-color: #007bff;
    outline: none;
    background-color: #e7f3ff;
}

.hidden {
    display: none;
}

.options {
    transition: opacity 0.5s ease;
    opacity: 0;
}

.options.show {
    opacity: 1;
}

.vehicle-button {
```

```
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 5px;
            padding: 10px 20px;
            margin-top: 15px;
            cursor: pointer;
            font-size: 16px;
            display: none;
            transition: background-color 0.3s ease, transform 0.2s ease;
        }

        .vehicle-button:hover {
            background-color: #0056b3;
            transform: translateY(-3px);
        }
    </style>
</head>
<body>

    <h1> Hi {{naming}}!,Welcome to Parking Lot Management System<h1>
    <div class="container">
        <label for="vehicleType">Select Vehicle Type:</label>
        <select id="vehicleType" onchange="showOptions()">
            <option value="">--Select--</option>
            <option value="two-wheeler">Two-Wheeler</option>
            <option value="four-wheeler">Four-Wheeler</option>
        </select>

        <div id="twoWheelerOptions" class="options hidden">
```

```
            <label for="twoWheeler">Select Two-Wheeler:</label>
            <select id="twoWheeler" onchange="toggleButtons()">
                <option value="">--Select--</option>
                <option value="cycle">Cycle</option>
                <option value="motorcycle">Motor Vehicle</option>
            </select>
        </div>

        <div id="fourWheelerOptions" class="options hidden">
            <label for="fourWheeler">Select Four-Wheeler:</label>
            <select id="fourWheeler" onchange="toggleButtons()">
                <option value="">--Select--</option>
                <option value="car">Car</option>
            </select>
        </div>

        <button id="cycleButton" class="vehicle-button">Park Cycle</button>
        <button id="motorcycleButton" class="vehicle-button">Park Motor
Vehicle</button>
        <button id="carButton" class="vehicle-button">Park Car</button>
    </div>
```

login.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
```

```
    <link rel="stylesheet" href="login.css">
    <script src="script.js"></script>
</head>

<body>
    <div class="container">
        <div class="box">
            <form action="/login" method="post">
                <h2>Login</h2>
                <div class="inputbox">
                    <input type="text" name="Username" required>
                    <label for="">Username</label>
                </div>
                <div class="inputbox">
                    <input type="password" name="Password" required>
                    <label for="">Password</label>
                </div>
                <div class="remember">
                    <input type="checkbox" name="remember">
                    <label for="">Remember me</label>
                </div>
                <button type="submit" class="loginbtn">Login</button>
                <div class="signup-link">
                    <p>Don't have an account?</p><a href="/signup" class="signbtn-
link">Sign up</a>
                </div>
            </form>
        </div>
    </div>
</body>
</html>
```

Signup.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sign-Up Form</title>
    <link rel="stylesheet" href="signup.css">
    <script src="script.js"></script>
</head>
<body>
    <div class="container">
        <div class="box">
            <form action="/signup" method="post">
                <h2>Registration</h2>
                <div class="inputbox">
                    <input type="text" name="Username" required>
                    <label for="">Username</label>
                </div>
                <div class="inputbox">
                    <input type="tel" name="MobileNo" required>
                    <label for="">Mobile No.</label>
                </div>
                <div class="inputbox">
                    <input type="email" name="Email" required>
                    <label for="">Email</label>
                </div>
                <div class="inputbox">
                    <input type="password" name="Password" required>
```

```
                <label for="">Password</label>
            </div>
            <button type="submit" class="signupbtn">Sign Up</button>
            <div class="signup-link">
                <p>Already Registered?</p><a href="/login" class="signbtn-
link">Login</a>
            </div>
        </form>
    </div>
  </div>
</body>
</html>
```

login.css

```
@import url('https://fonts.googleapis.com/css2?
family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400
;1,500;1,700;1,900&display=swap');

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;

}

body {
  background: url(background.jpg);
  background-size: 100% 100%;
  display: flex;
  justify-content: center;
```

```
    align-items: center;
    min-height: 100vh;
    font-family: Roboto;
}

.container {
    position: relative;
    width: 400px;
    height: 450px;
    border: 2px solid gray;
    /* background-color: rgb(93, 118, 118); */
    border-radius: 20px;
    padding: 40px;
    align-items: center;
    justify-content: center;
    box-shadow: 0px 0px 30px 10px;
    backdrop-filter: blur(12px);
}

.box {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100%;
    height: 100%;
}

h2 {
    font-size: 30px;
    color: rgb(231, 231, 238);
```

```
    text-shadow: 0px 0px 30px;
    text-align: center;

}

.inputbox {
    position: relative;
    margin: 30px 0;
    border-bottom: 2px solid white;
}

.inputbox label {
    position: absolute;
    top: 50%;
    left: 5px;
    transform: translateY(-50%);
    color: #fff;
    font-size: 16px;
    pointer-events: none;
}

.inputbox input {
    width: 320px;
    height: 40px;
    font-size: 16px;
    color: #fff;
    padding: 0 5px;
    background: transparent;
    border: none;
    outline: none;
```

```
}

.inputbox input:focus~label,
.inputbox input:valid~label {
    top: -5px;
    transition: 0.25s ease-in-out;
}

.remember {
    margin: -5px 0 15px 5px;
}

.remember label {
    color: #fff;
    font-size: 14px;
}

.remember label input {
    accent-color: #0ef;
}

button {
    position: relative;
    width: 100%;
    height: 40px;
    background: rgb(148, 161, 161);
    box-shadow: 0 0 20px rgb(13, 13, 13);
    font-size: 16px;
    color: #000;
    font-weight: 500;
```

```
    cursor: pointer;
    border-radius: 30px;
    border: none;
    outline: none;
    border: 2px solid white;
    width: 100px;
    margin: 10px 110px;
}

button:hover {
    background-color: rgb(25, 95, 123);
    color: rgb(255, 255, 255);
}

.signbtn-link {
    text-decoration: none;
    color: rgb(255, 255, 255);
}

.signbtn-link:hover {
    color: rgb(117, 117, 117);
}
```

signup.css

```
@import url('https://fonts.googleapis.com/css2?
family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400
;1,500;1,700;1,900&display=swap');

* {
    margin: 0;
```

```
    padding: 0;
    box-sizing: border-box;

}

body {
    background: url(background.jpg);
    background-size: 100% 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    font-family: Roboto;
}

.container {
    position: relative;
    width: 400px;
    height: 550px;
    border: 2px solid gray;
    /* background-color: rgb(93, 118, 118); */
    border-radius: 20px;
    padding: 40px;
    align-items: center;
    justify-content: center;
    box-shadow: 0px 0px 30px 10px;
    backdrop-filter: blur(12px);
}

.box {
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100%;
    height: 100%;
}

h2 {
    font-size: 30px;
    color: rgb(231, 231, 238);
    text-shadow: 0px 0px 30px;
    text-align: center;
}

.inputbox {
    position: relative;
    margin: 30px 0;
    border-bottom: 2px solid white;
}

.inputbox label {
    position: absolute;
    top: 50%;
    left: 5px;
    transform: translateY(-50%);
    color: #fff;
    font-size: 16px;
    pointer-events: none;
}
```

```
.inputbox input {
    width: 320px;
    height: 40px;
    font-size: 16px;
    color: #fff;
    padding: 0 5px;
    background: transparent;
    border: none;
    outline: none;
}

.inputbox input:focus~label,
.inputbox input:valid~label {
    top: -5px;
    transition: 0.25s ease-in-out;
}

.remember {
    margin: -5px 0 15px 5px;
}

.remember label {
    color: #fff;
    font-size: 14px;
}

.remember label input {
    accent-color: #0ef;
}
```

```
button {
    position: relative;
    width: 100%;
    height: 40px;
    background: rgb(148, 161, 161);
    box-shadow: 0 0 20px rgb(13, 13, 13);
    font-size: 16px;
    color: #000;
    font-weight: 500;
    cursor: pointer;
    border-radius: 30px;
    border: none;
    outline: none;
    border: 2px solid white;
    width: 100px;
    margin: 10px 110px;
}

button:hover {
    background-color: rgb(25, 95, 123);
    color: rgb(255, 255, 255);
}

.signbtn-link {
    text-decoration: none;
    color: rgb(255, 255, 255);
}
p{
    color: rgb(134, 127, 127);
}
```

```
.signbtn-link:hover {
    color: rgb(117, 117, 117);
}
```

style.css

```
body {
    font-family: 'Roboto', sans-serif;
    margin: 0;
    background: linear-gradient(135deg, #74ebd5, #acb6e5);


}

header {
    margin-top: 20px;
    margin-left: 10px;
    margin-right: 10px;
    border-radius: 150px;
    background-color: transparent;
    opacity: .8;
    padding: 15px 0;
    text-align: center;
    font-size: 24px;
    font-weight: bold;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    transition: all 0.3s ease;
    color: rgb(110, 119, 206);
```

```
}

header:hover {
    background-color: white;
    color: #1e1e1e;
}

.form-container {
    display: flex;
    background-color: transparent;
    opacity: 1.9;
    padding: 20px;
    width: 60%;
    margin: 50px auto;
    border-radius: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    transition: transform 0.3s ease;
    justify-content: center;
    align-items: center;
}

.form-container:hover {
    transform: scale(1.02);
}

h5 {
    color: rgb(110, 119, 206);
    text-align: center;
    margin-bottom: 20px;
```

```
    font-size: 1.3rem;
}

.form-group label {
    color: rgb(110, 119, 206);
    font-weight: bold;
}
.form-control {
    width: 100%;
    border: 2px solid rgb(193, 193, 224);
    border-radius: 5px;
    padding: 10px;
    font-size: 16px;
    margin-top: 5px;
    transition: all 0.3s ease;
    background: linear-gradient(135deg, #74ebd5, #acb6e5);
    outline: none;
    color: rgb(107, 107, 198);
}

.form-control:focus {
    border-color: rgb(193, 193, 224);
    box-shadow: none;
    background: linear-gradient(135deg, #74ebd5, #acb6e5);
}


#btnOne {
    display: flex;
    background-color:transparent;
```

```
    padding: 10px 20px;
    font-size: 16px;
    font-weight: bold;
    cursor: pointer;
    transition: all 0.3s ease;
    border: 2px solid rgb(176, 176, 206);
    border-radius: 15px;
    width: 350px;
    justify-content: center;
    align-items: center;
    margin-top: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    text-decoration: solid;

}

#btnOne:hover {
    background-color: black;
    transform: scale(1.05);
}

.table-container {
    width: 80%;
    margin: auto;
    margin-top: 30px;
    background-color: transparent;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}
```

```
#searchInput {
    background: linear-gradient(135deg, #74ebd5, #acb6e5);
    margin-left: 360px;
    width: 450px;
    padding: 10px;
    font-size: 16px;
    margin-bottom: 20px;
    border: 4px solid rgb(180, 180, 209);
    border-radius: 15px;
    transition: all 0.3s ease;
}

#searchInput:focus {
    border-color: red;
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
}

table {
    width: 100%;
    border-collapse: collapse;
    background-color:rgb(169, 169, 199);
}

thead {
    background-color: red;
    /* color: black; */
}

th, td {
```

```
    padding: 12px;
    text-align: center;
    border-bottom: 1px solid #ddd;
}

tr:hover {
    background-color: blue;
}

@media (max-width: 768px) {
    .form-container, .table-container {
        width: 90%;
    }

    .form-group {
        font-size: 14px;
    }

    #btnOne {
        font-size: 14px;
    }
}
.form-control{
    /* border: 6px solid black; */
    border-radius: 10px;
}
#aa{
    color: rgb(110, 119, 206);
    font-weight: bold;
}
```

MILESTONE

S.No.	Project Activity	Estimated Start Date	Estimated End Date
1.	Synopsis submission		
2.			

MEETING WITH THE SUPERVISOR

Date of the meet	Mode	Comments by the supervisor	Signature of the Supervisor

Bibliography

Books

1. "Smart Parking Systems" by S. S. Iyengar et al., published by Springer, 2019.
2. "Online Parking Management Systems" by A. K. Singh et al., published by CRC Press, 2018.
3. "Parking Management Systems" by R. L. Cheu et al., published by Wiley, 2017.

Online Resources

1. "Online Parking System" by GitHub, 2020.
2. "Smart Parking System" by ResearchGate, 2019.
3. "Parking Management System" by ScienceDirect, 2018.