# A MINI-PROJECT REPORT

***Submitted by***

**ORRIN LEROY H 220701191**

***In partial fulfillment of the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

**RAJALAKSHMI ENGINEERING COLLEGE**

**AUTONOMOUS, CHENNAI**

**NOV/DEC, 2024**

# BONAFIDE CERTIFICATE

Certified that this mini project **“ Smart Parking Solution ”** is the bonafide work of **“ORRIN LEROY H(2116220701191)”** who carried out the project work under my supervision.

**SIGNATURE**

**Mrs. JANANEE V,**

Assistant Professor,

Computer Science and Engineering,

Rajalakshmi Engineering College,

Thandalam, Chennai - 602105.

Submitted for the End Semester Practical examination to be held on \_\_\_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# 

# ACKNOWLEDGEMENT

I express my sincere thanks to my beloved and honorable chairman

**Mr. S. MEGANATHAN** and the chairperson  **Dr. M. THANGAM MEGANATHAN** for their timely support and encouragement.

I am greatly indebted to my respected and honourable principal

**Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by my head of the department, **Dr. P. KUMAR**, and my Academic Head **Dr. R. SABITHA**, for being every supporting force during my project work.

I also extend my sincere and hearty thanks to my internal guide **Mrs. JANANEE V** for her valuable guidance and motivation during the completion of this project.

My sincere thanks to my family members, friends and other staff members of Computer Science and Engineering.

# ABSTRACT

# Urban areas face significant challenges in parking management due to increasing vehicle density, leading to traffic congestion, fuel wastage, and driver frustration. PRAWN (Parking Reservation and Wayfinding Network) aims to tackle these issues by providing an IoT-enabled smart parking system that offers real-time monitoring, slot reservations, and online payments through a web-based platform.

# PRAWN features both a user interface for reservations and an admin dashboard to monitor parking availability, manage bookings, and track revenue. By providing real-time data, it ensures transparency, prevents overbooking, and streamlines parking management. This system helps reduce traffic congestion and carbon emissions by guiding drivers directly to available slots.

# With its modular architecture, PRAWN is easily scalable for different locations. Future enhancements could include dynamic pricing, AI-based slot availability predictions, and location-based notifications to further improve efficiency. In conclusion, PRAWN offers an innovative, scalable, and cost-effective solution for cities looking to adopt smart city initiatives and improve urban mobility.

# TABLE OF CONTENTS

**TITLE PAGE**

**ABSTRACT** 4

1. **INTRODUCTION** 6
   1. INTRODUCTION 6
   2. SCOPE OF THE WORK 6
   3. AIM AND OBJECTIVES OF THE PROJECT 6
2. **SYSTEM SPECIFICATIONS**  8
   1. HARDWARE SPECIFICATIONS 8
   2. SOFTWARE SPECIFICATIONS 8
3. **ARCHITECTURE DIAGRAM** 9
4. **MODULE DESCRIPTION** 10
5. **SYSTEM DESIGN** 12
   1. USE CASE DIAGRAM
   2. ER DIAGRAM
   3. DATA FLOW DIAGRAM
   4. ACTIVITY DIAGRAM
6. **SAMPLE CODING** 16
7. **SCREENSHOTS** 29
8. **CONCLUSION** 32

**REFERENCES** 33

# CHAPTER 1

## **INTRODUCTION**

### 1.1 INTRODUCTION

### With the rapid growth of urbanization and the increasing number of vehicles, cities are facing significant challenges in managing parking spaces efficiently. Traditional parking systems often lead to unnecessary delays, fuel wastage, and increased pollution as drivers struggle to find available parking. This not only affects individual drivers but also contributes to traffic congestion and environmental degradation. To address these challenges, PRAWN (Parking Reservation and Wayfinding Network) introduces a smart parking management system that leverages IoT technologies and web-based platforms to provide a seamless parking experience.

### 1.2 SCOPE OF WORK

### This project aims to develop a web-integrated parking system that ensures the following:

### Real-time monitoring of parking slot availability using IoT sensors.

### User-friendly reservation portal to allow booking and payment.

### Google Pay integration to enable secure online transactions.

### Admin dashboard to manage slot availability, bookings, and payments efficiently.

### Scalable and modular architecture, enabling expansion to other locations.

### Integration with real-time traffic management systems to optimize routes.

### 1.3 AIM AND OBJECTIVES OF THE PROJECT

The primary aim of PRAWN is to develop an automated parking management system that minimizes driver inconvenience, reduces traffic congestion, and optimizes parking space utilization through real-time monitoring and reservation features. The system is designed to enhance user convenience by offering online payment options and guiding users to available slots efficiently. Additionally, PRAWN aims to reduce environmental impact by minimizing the time and fuel wasted while searching for parking. With its scalable and modular design, the platform can be adapted to different urban environments, promoting seamless integration with smart city initiatives.

**OBJECTIVES:**

1. Provide a real-time booking and monitoring system using IoT **sensors**.
2. Ensure easy accessibility through a web interface for users to search, book, and pay for parking.
3. Improve urban mobility by reducing the time spent searching for parking.
4. Streamline parking operations through automated slot management and payments.
5. Enable administrators to monitor bookings, payments, and slot occupancy through an intuitive dashboard.

By fulfilling these objectives, the PRAWN system contributes towards smart city initiatives, offering cost-effective, eco-friendly, and scalable parking solutions that enhance the overall urban experience.

# CHAPTER 2

## **SYSTEM SPECIFICATIONS**

### 2.1 HARDWARE SPECIFICATIONS

To run the application smoothly, the following hardware is recommended:

* **Microcontroller**: ESP8266 or ESP32.
* **Sensors**: Infrared Sensor.
* **Computer**: Intel Core i5 or higher, Minimum of 8GB RAM, 500GB of storage.
* **Other**: Breadboard, Wires.

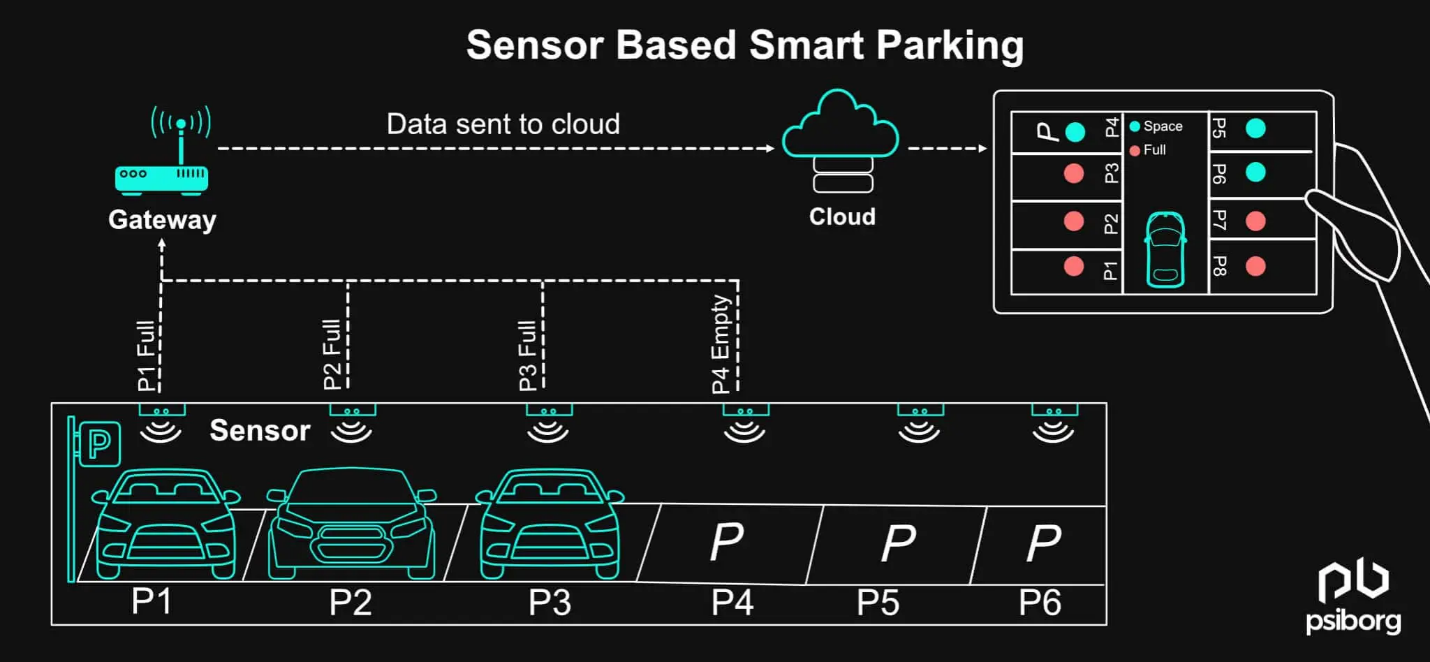
### 2.2 SOFTWARE SPECIFICATIONS

The project relies on a set of modern software tools and technologies to build, test, and deploy the system:

* **Front-End:** HTML, CSS, Bootstrap, and JavaScript
* **Framework:** Flask
* **Backend:** Python3, Firebase (NoSQL cloud-based storage)
* **Machine Learning:** scikit-learn(library), Regression model.

# CHAPTER 3

## **ARCHITECTURE DIAGRAM**



# CHAPTER 4

## **MODULE DESCRIPTION**

### 4.1 USER AUTHENTICATION MODULE

### This module handles user sign-up, login, and profile management, including password encryption and secure session handling. It also manages different roles, such as user and admin, for controlled access to specific features within the system.

### 4.2 REAL-TIME SLOT MONITORING MODULE

### Integrated with IoT sensors, this module monitors and displays the current status of parking slots (occupied, vacant, reserved) in real time. It updates the slot availability data on the frontend for accurate booking and usage visibility.

### 4.3 RESERVATION & BOOKING MANAGEMENT MODULE

This core module enables users to search, reserve, and manage parking slots through the platform. It manages booking requests, time slots, and cancellations, and provides users with booking history and confirmations.

### 4.4 PAYMENT PROCESSING MODULE

### This module handles secure payments for parking reservations, integrating with Google Pay for transaction processing. It includes billing, transaction history, and receipt generation to ensure a smooth and transparent payment experience for users.

### 4.5 ADMIN DASHBOARD MODULE

Designed for system administrators, this module provides insights and controls for managing parking slots, monitoring usage patterns, overseeing bookings, and accessing financial reports. It includes data visualizations and management tools for streamlined system operations.

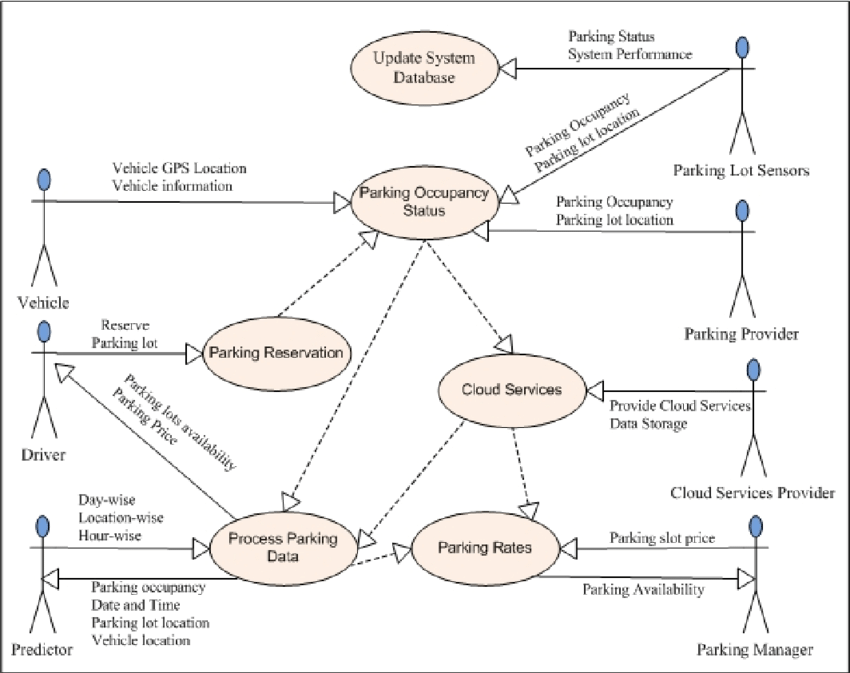
### 4.6 ANALYTICS & REPORTING MODULE

### This module provides insights into parking slot usage patterns, peak booking times, user demographics, and financial statistics. Equipped with data visualizations and exportable reports, it allows administrators to make data-driven decisions to optimize parking space management and improve user experience.

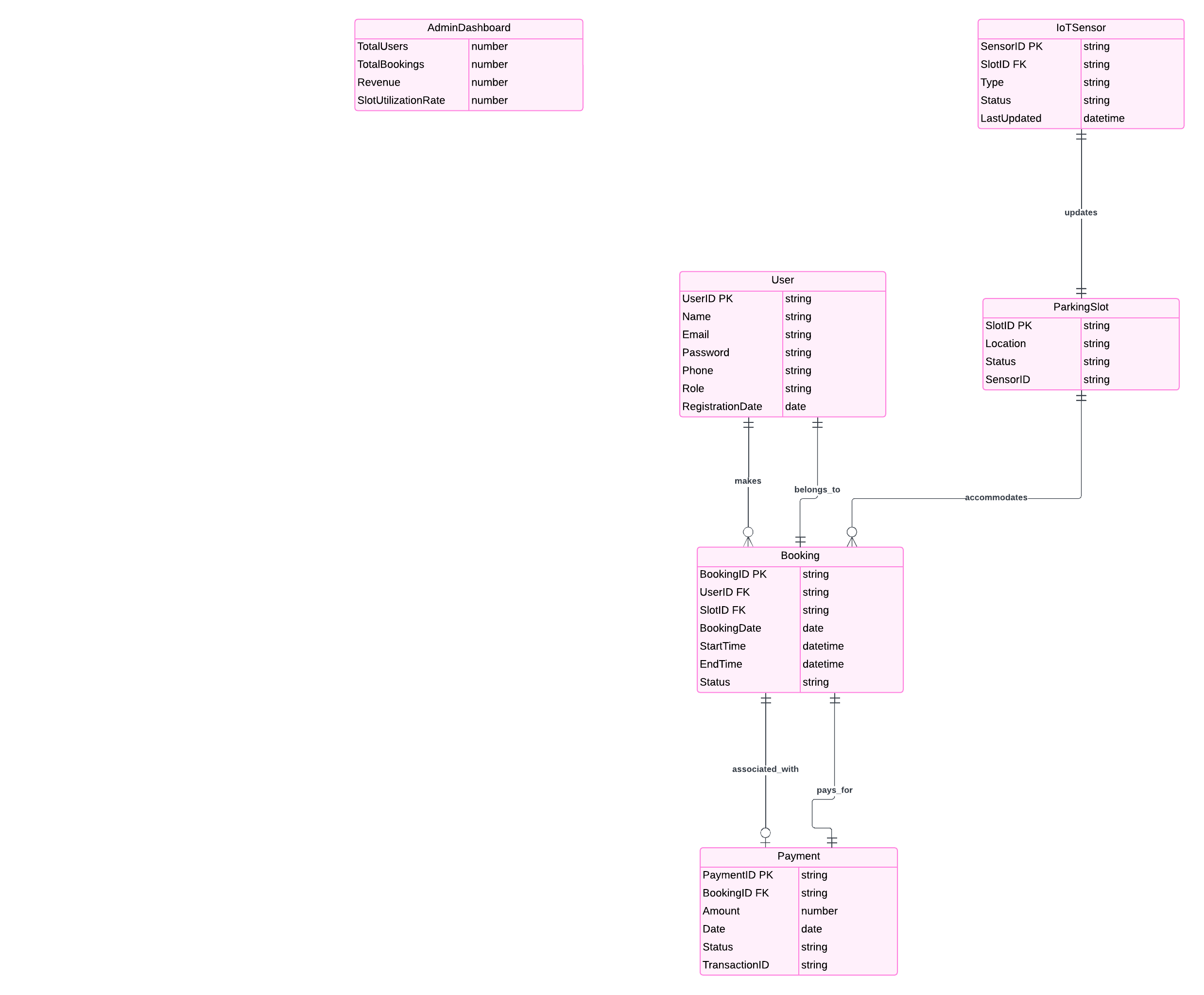
# CHAPTER 5

## **SYSTEM DESIGN**

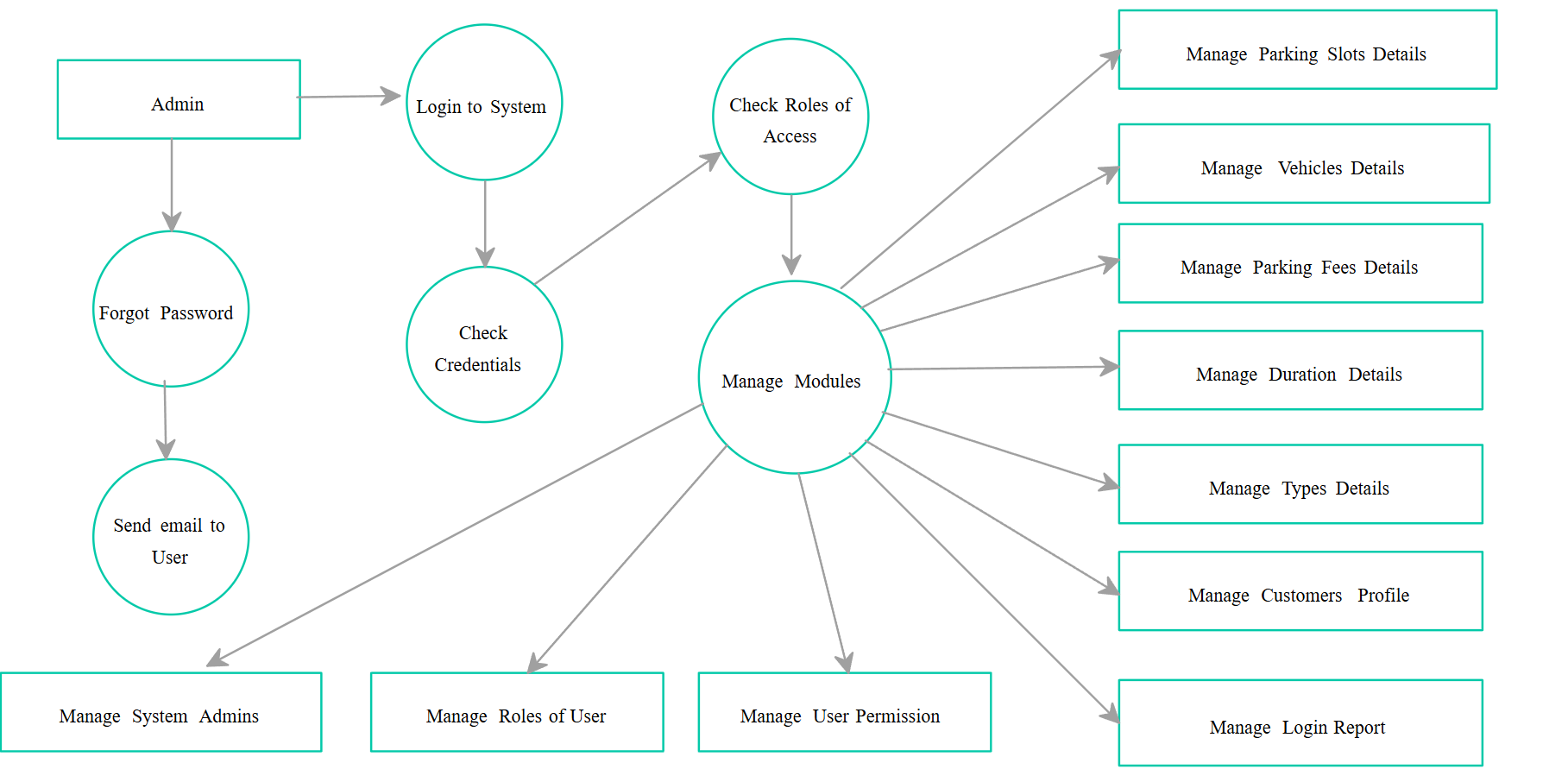
### 5.1 USE CASE DIAGRAM



### 5.2 ER DIAGRAM

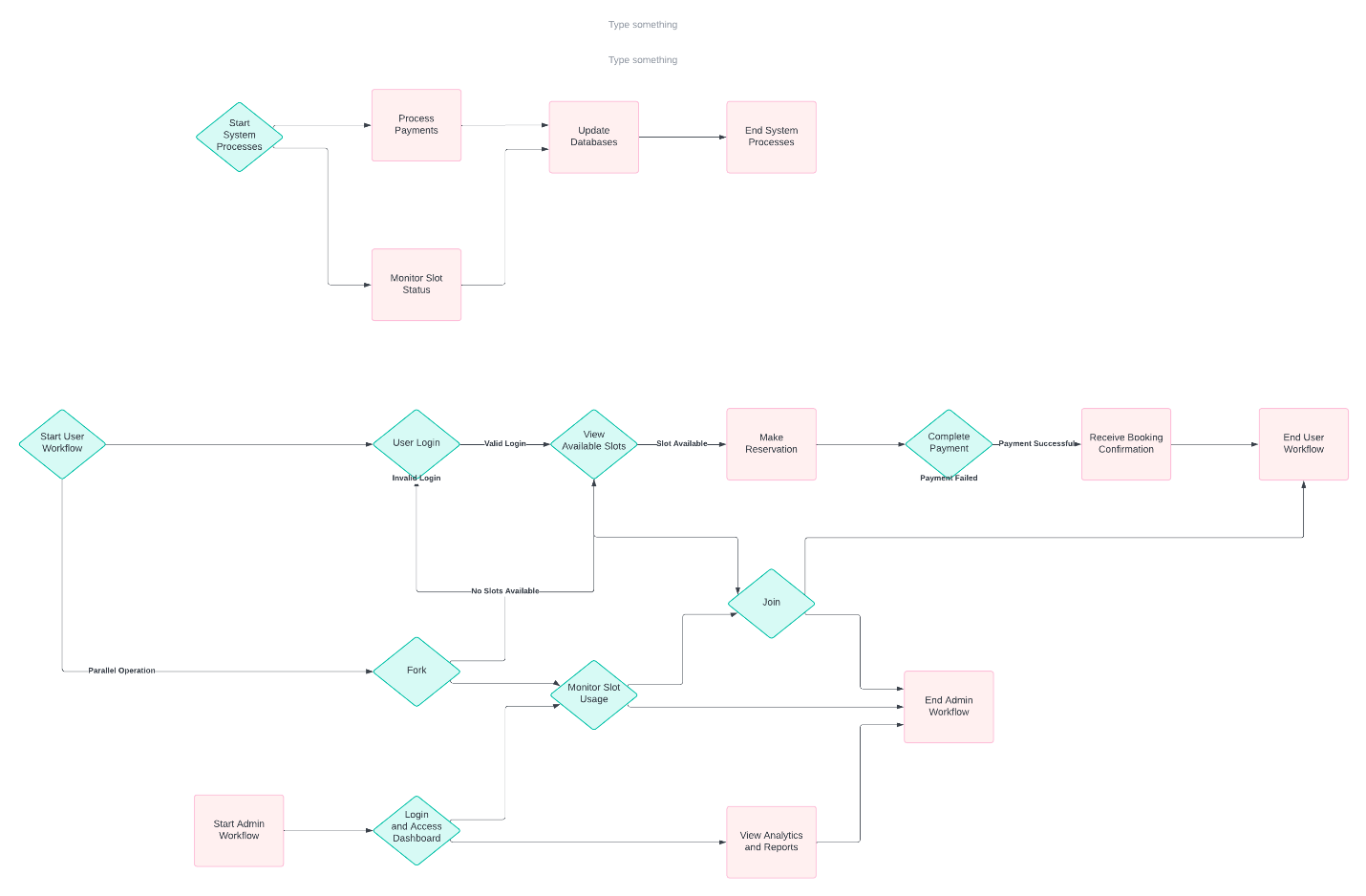


### 5.3 DATA FLOW DIAGRAM



#### 

### 5.4 ACTIVITY DIAGRAM



# CHAPTER 6

## **SAMPLE CODING**

home.html :-

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>PRAWN</title>

<link rel="icon" href="/static/logo.png" type="image/x-icon">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom CSS -->

<link rel="stylesheet" href="/static/style.css">

</head>

<body>

<!-- Header -->

<header class="container-fluid bg-dark py-3">

<div class="d-flex justify-content-between align-items-center">

<div class="d-flex align-items-center">

<img src="/static/logo.png" alt="PRAWN Logo" class="logo">

<h1 class="text-white ms-3 mb-0">PRAWN</h1>

</div>

<button class="btn btn-danger rounded-pill px-4" onclick="window.location.href='/register';">Reserve Now</button>

</div>

</header>

<!-- Hero Section with Full-width Image -->

<section class="hero-section d-flex justify-content-center align-items-center flex-column text-center text-white">

<img src="/static/park.jpg" alt="Hero Background" class="hero-img img-fluid w-100">

<div class="overlay-text position-absolute">

<h2 class="display-3">PRAWN</h2>

<p class="lead fst-italic">Parking Reservation And Wayfinding Network</p>

</div>

</section>

<!-- Features Section with Background Image -->

<section class="container-fluid features-section text-center text-white py-5" style="background-image: url('/static/features-bg.jpg'); background-size: cover;">

<div class="container">

<div class="row">

<div class="col-lg-4 col-md-6 col-sm-12">

<div class="feature p-4 rounded shadow-sm">

<h3>Real-Time Availability</h3>

<p>Find available parking spots in real time using advanced IoT sensors.</p>

</div>

</div>

<div class="col-lg-4 col-md-6 col-sm-12 mt-4 mt-md-0">

<div class="feature p-4 rounded shadow-sm">

<h3>Advanced Reservations</h3>

<p>Reserve your spot in advance and reduce the hassle of searching for parking.</p>

</div>

</div>

<div class="col-lg-4 col-md-6 col-sm-12 mt-4 mt-lg-0">

<div class="feature p-4 rounded shadow-sm">

<h3>Navigation Assistance</h3>

<p>Get precise navigation to your reserved parking spot with ease.</p>

</div>

</div>

</div>

</div>

</section>

<!-- Footer -->

<footer class="bg-dark text-white text-center py-3">

<p>&copy; 2024 PRAWN - All Rights Reserved</p>

</footer>

<!-- Bootstrap JS -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

Style.css :-

/\* General styling \*/

body {

font-family: Arial, sans-serif;

background-color: #f8f9fa;

margin: 0;

color: white;

}

/\* Header \*/

header .logo {

width: 50px;

height: 50px;

}

header h1 {

font-size: 28px;

}

/\* Hero Section \*/

.hero-section {

position: relative;

height: 60vh;

}

.hero-img {

height: 100%;

object-fit: cover;

}

.overlay-text {

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

position: absolute;

z-index: 1;

}

.hero-section h2 {

font-size: 3rem;

color: white;

}

.hero-section p {

font-size: 1.25rem;

color: white;

}

/\* Features Section \*/

.features-section {

background-color: #212529; /\* Dark background to match the theme \*/

padding: 60px 0;

color: white; /\* Ensure the text is visible on a darker background \*/

}

.feature {

background-color: #333; /\* Dark background for the feature cards \*/

color: white;

transition: transform 0.3s ease;

border-radius: 10px;

box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.3);

padding: 20px;

}

.feature h3 {

font-size: 1.5rem;

color: #ff007f; /\* Consistent with button color \*/

}

.feature p {

font-size: 1rem;

}

.feature:hover {

transform: translateY(-10px);

}

.features-row {

display: flex;

justify-content: space-around;

margin-top: 40px;

}

/\* Footer \*/

footer {

background-color: #200024; /\* Ensure footer background matches \*/

padding: 20px 0;

}

footer p {

margin: 0;

color: white;

}

/\* Responsiveness \*/

@media (max-width: 768px) {

.features-row {

flex-direction: column;

margin-top: 20px;

}

.feature {

margin-bottom: 20px;

}

}

login.html :-

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login - PRAWN</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

<link rel="stylesheet" href="/static/login.css">

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container">

<img src="/static/logo.png" alt="PRAWN Logo" class="logo" width="50px" height="50px">

<a class="navbar-brand" href="#">&nbsp;PRAWN</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse justify-content-end" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link" href="/">Home</a>

</li>

<li class="nav-item">

<a class="nav-link active" href="#">Login</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/register">Register</a>

</li>

</ul>

</div>

</div>

</nav>

<section class="login-section d-flex justify-content-center align-items-center">

<div class="login-box p-5 rounded">

<h2 class="text-center mb-4">Login</h2>

<form method="POST" action="/login">

<div class="mb-3">

<label for="email" class="form-label">Email</label>

<input type="email" class="form-control" id="email" name="email" placeholder="Enter your email" required>

</div>

<div class="mb-3">

<label for="password" class="form-label">Password</label>

<input type="password" class="form-control" id="password" name="password" placeholder="Enter your password" required>

</div>

<button type="submit" class="btn btn-primary w-100">Login</button>

</form>

<!-- Display the message below the login form -->

{% if message %}

<div class="mt-3 alert alert-danger" role="alert">

{{ message }}

</div>

{% endif %}

</div>

</section>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

app.py :-

from flask import Flask, render\_template, request, redirect, url\_for, session

from flask\_pymongo import PyMongo

from werkzeug.security import generate\_password\_hash, check\_password\_hash

import os

app = Flask(\_\_name\_\_)

# Configuration

app.config["MONGO\_URI"] = "mongodb://localhost:27017/prawn"

app.secret\_key = os.getenv("SECRET\_KEY", "your\_default\_secret\_key")

mongo = PyMongo(app)

@app.route('/register', methods=['GET', 'POST'])

def register():

message = None

if request.method == 'POST':

username = request.form.get('username')

email = request.form.get('email')

password = request.form.get('password')

confirm\_password = request.form.get('confirm\_password')

if password != confirm\_password:

message = "Passwords do not match!"

elif mongo.db.login.find\_one({"username": username}):

message = "Username already exists!"

elif mongo.db.login.find\_one({"email": email}):

message = "Email already exists!"

else:

hashed\_password = generate\_password\_hash(password)

mongo.db.login.insert\_one({

"username": username,

"email": email,

"password": hashed\_password

})

message = "Registration successful! Please log in."

return render\_template('register.html', message=message)

return render\_template('register.html', message=message)

@app.route('/login', methods=['GET', 'POST'])

def login():

message = None

if request.method == 'POST':

email = request.form.get('email')

password = request.form.get('password')

user = mongo.db.login.find\_one({"email": email})

if user and check\_password\_hash(user['password'], password):

session['username'] = user['username']

return redirect(url\_for('search'))

else:

message = "Invalid email or password"

return render\_template('login.html', message=message)

@app.route('/search', methods=['GET', 'POST'])

def search():

return render\_template('search-parking.html')

@app.route('/marina', methods=['GET', 'POST'])

def marina():

if request.method == 'POST':

slot = request.form.get('slot')

# Debugging output to confirm slot selection

print(f"Selected slot: {slot}")

return redirect(url\_for('confirm\_payment', slot=slot))

return render\_template('marina-beach-parking.html')

@app.route('/confirm-payment', methods=['GET'])

def confirm\_payment():

slot = request.args.get('slot')

if not slot:

return "Error: Slot not specified", 400

return render\_template('confirm-payment.html', slot=slot)

@app.route('/', methods=['GET'])

def home():

return render\_template('home.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

parking.ino :-

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

// WiFi credentials

const char\* ssid = "Your\_SSID";

const char\* password = "Your\_PASSWORD";

// Server URL (Replace with your server IP)

const String serverUrl = "http://<your-server-ip>:5000/api/update\_slot";

// Sensor pins

const int trigPin = 5; // GPIO5 (D1)

const int echoPin = 4; // GPIO4 (D2)

int slotId = 1; // Assign each parking slot a unique ID

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

// Connect to WiFi

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("\nConnected to WiFi");

}

void loop() {

long duration, distance;

// Trigger the ultrasonic sensor

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Measure the pulse duration

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2; // Convert to cm

// Determine if the slot is occupied

bool isOccupied = (distance < 30); // Adjust this threshold as necessary

// Send slot status to the Flask server

sendSlotStatus(isOccupied);

delay(5000); // Update every 5 seconds

}

void sendSlotStatus(bool isOccupied) {

if (WiFi.status() == WL\_CONNECTED) {

HTTPClient http;

http.begin(serverUrl);

// Prepare the JSON payload

String payload = "{\"id\":" + String(slotId) + ",\"isOccupied\":" + (isOccupied ? "true" : "false") + "}";

http.addHeader("Content-Type", "application/json");

// Send the POST request

int httpResponseCode = http.POST(payload);

Serial.println("Response Code: " + String(httpResponseCode));

http.end();

} else {

Serial.println("WiFi Disconnected");

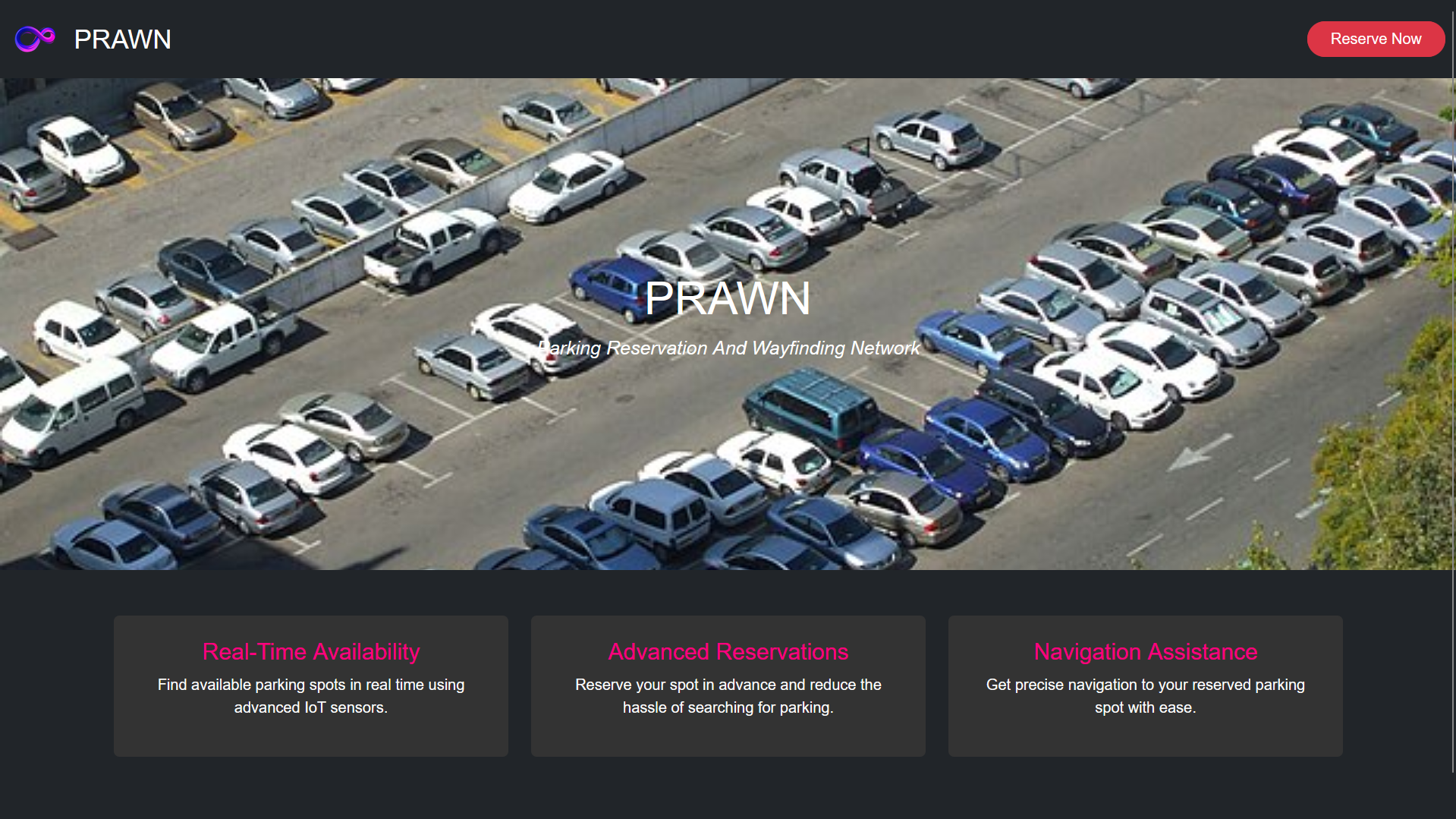
}

}

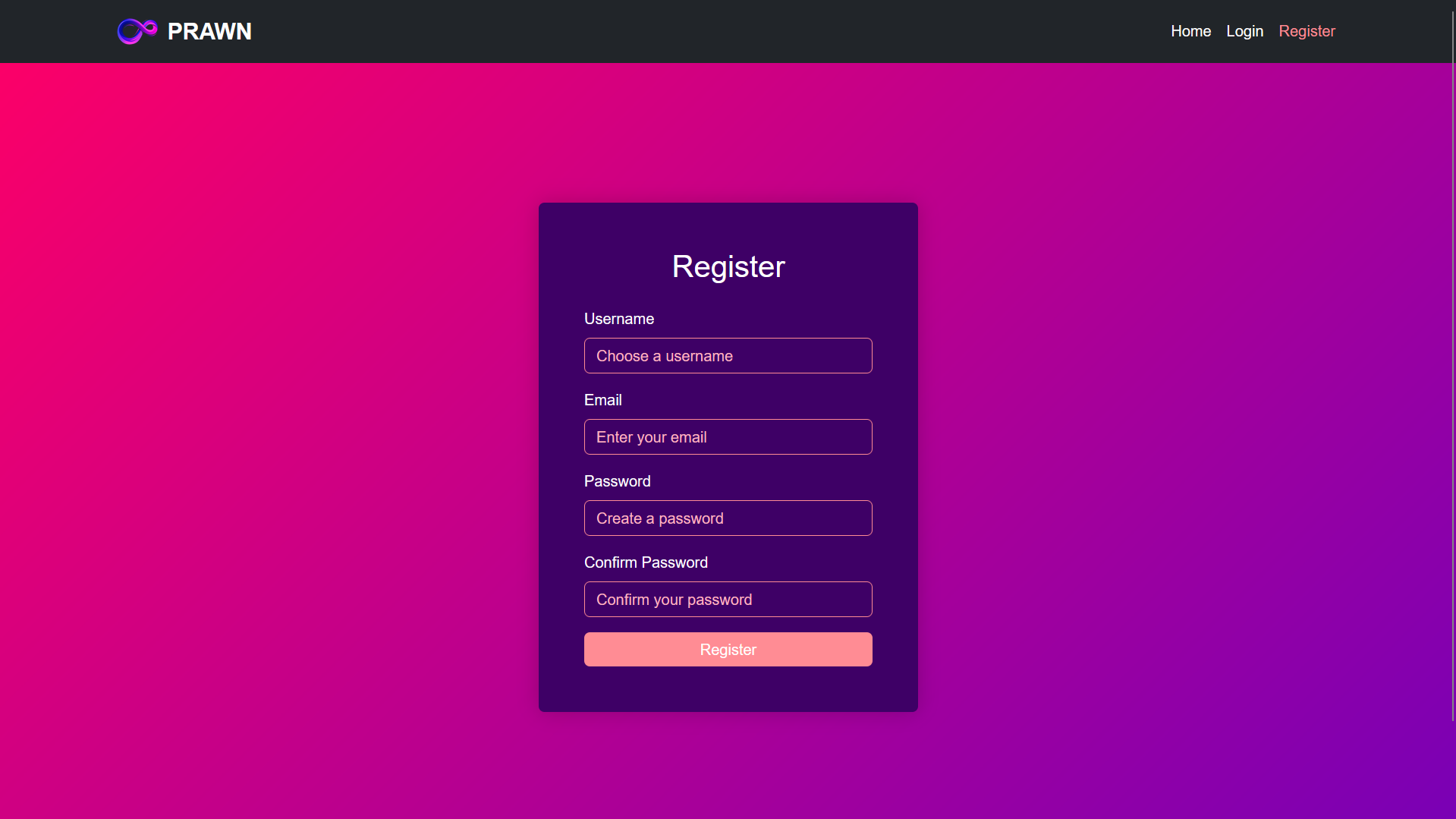
# CHAPTER 7

## **SCREENSHOTS**

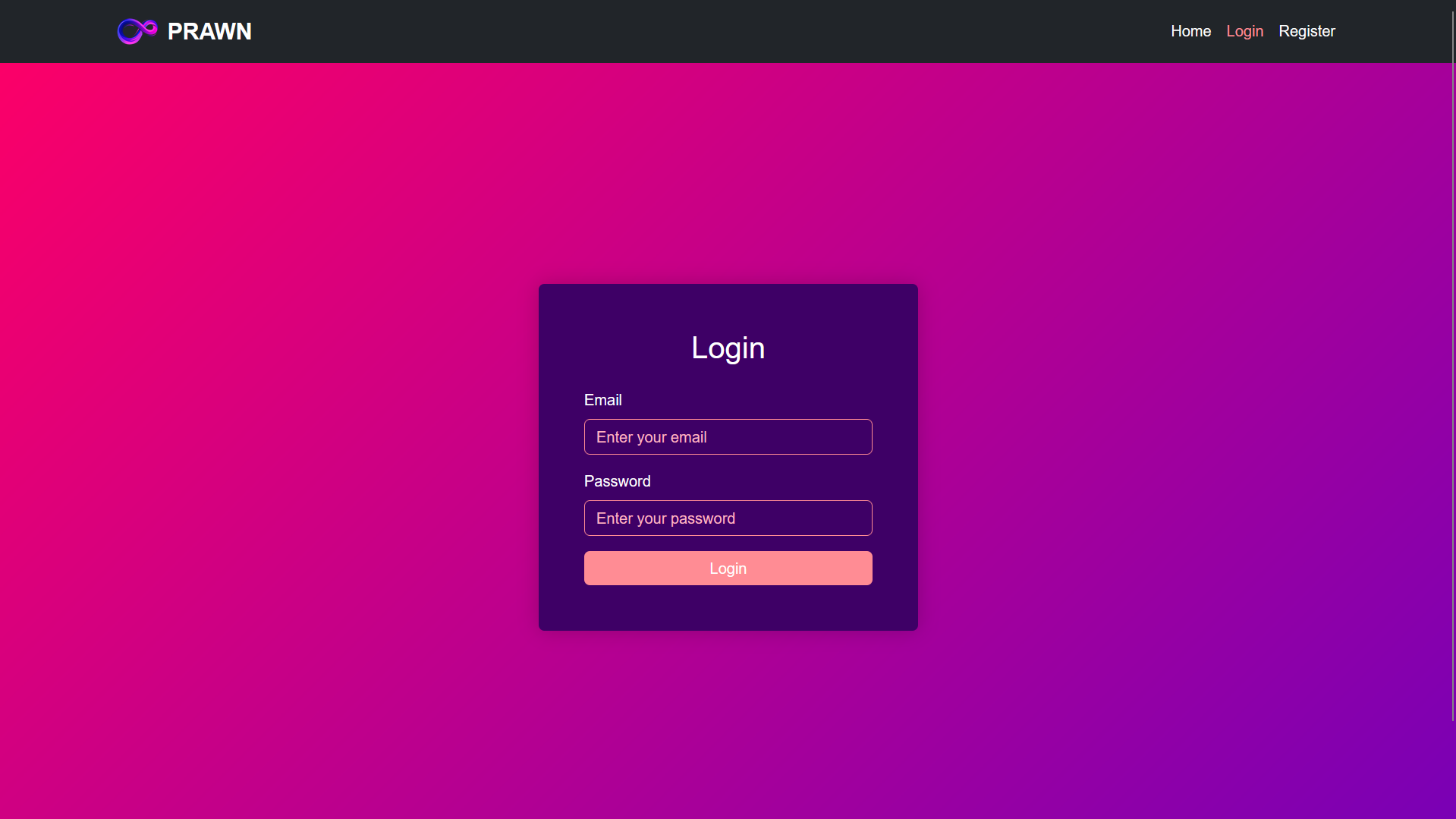
### 7.1 HOME PAGE



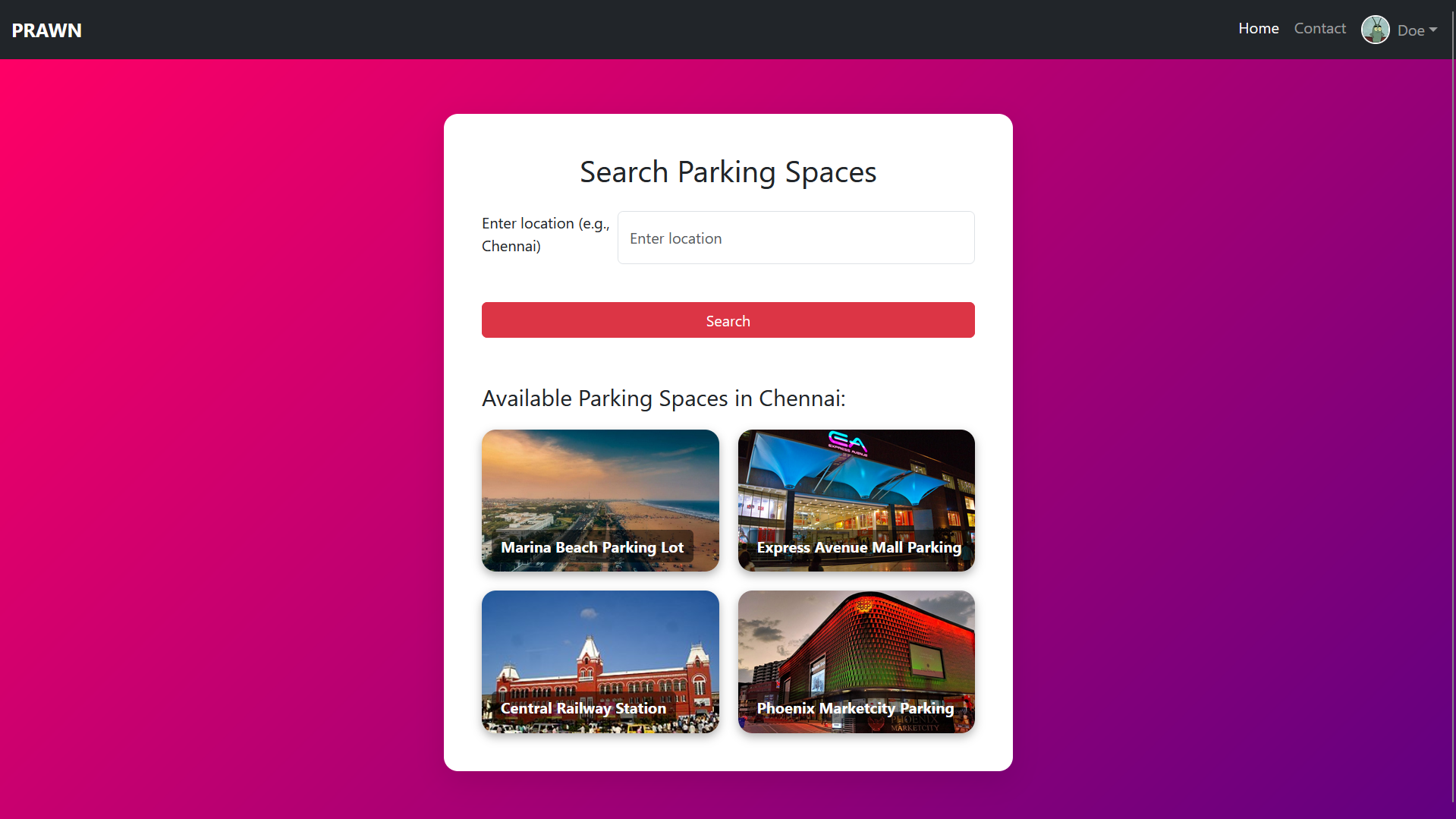
### 7.1 REGISTER PAGE



### 7.2 LOGIN PAGE

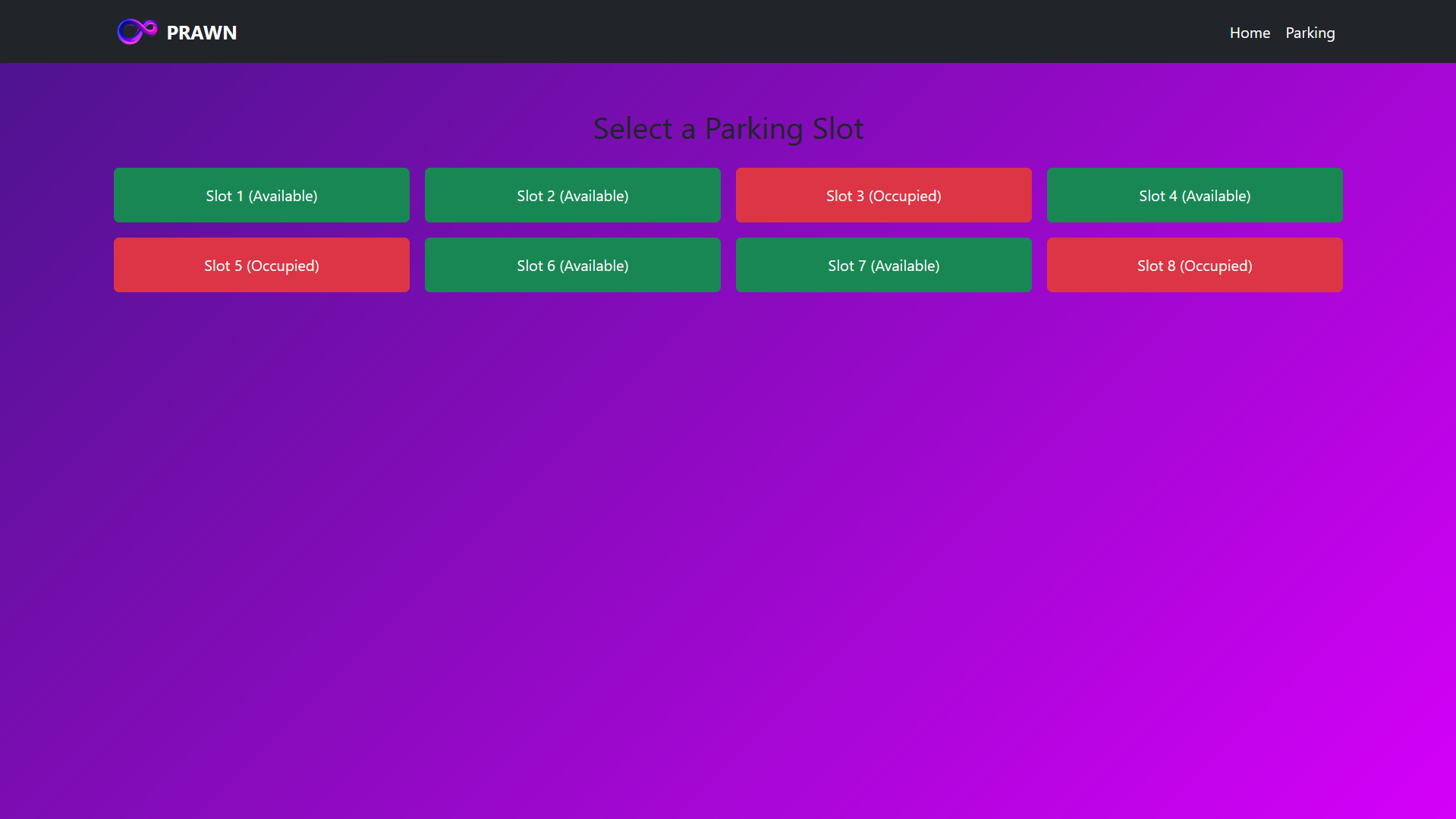


### 7.3 SEARCH PARKING LOT PAGE

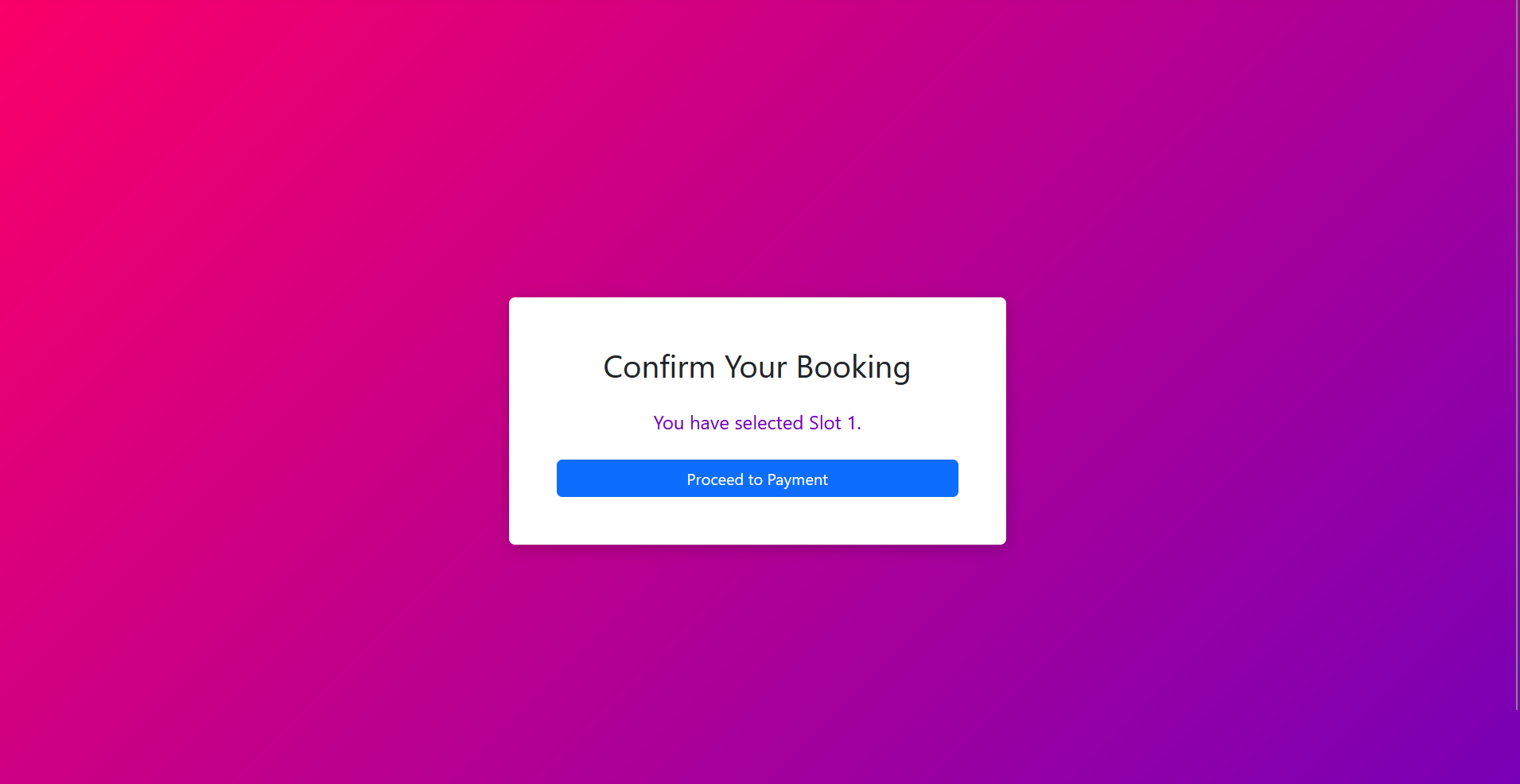


### 

### 7.4 SLOT SELECTION PAGE



### 7.5 BOOKING PAGE



# CHAPTER 8

## **CONCLUSION**

# In conclusion, the PRAWN project presents a practical solution to the growing challenge of parking management in urban areas. As cities become more densely populated, the demand for efficient parking solutions has intensified, making traditional methods increasingly impractical. PRAWN addresses these issues by leveraging IoT technology to provide users with real-time monitoring and a seamless reservation system. This enables users to easily locate and reserve available parking slots, saving time and reducing frustration. With modules designed for specific functions, such as slot monitoring, reservation management, and secure payment processing, PRAWN prioritizes both convenience and security.

# The modular design of PRAWN allows for scalability, making it adaptable to future needs and updates. By separating features into distinct modules, such as the notifications module for reservation alerts and the analytics module for usage insights, PRAWN provides both users and administrators with a high degree of flexibility. Users benefit from a smooth experience, receiving timely notifications and reminders about their bookings, while administrators gain access to a comprehensive dashboard with valuable data insights. These insights—covering usage patterns, peak booking times, and financial data—enable informed decision-making that optimizes resource allocation and enhances operational efficiency.

# Ultimately, the PRAWN project demonstrates the transformative potential of IoT-enabled systems in creating smarter urban spaces. It not only addresses the current challenges of parking congestion but also establishes a framework that can be expanded with features like dynamic pricing or integration with other urban mobility services. PRAWN’s success underscores the impact of technology-driven solutions in urban planning, paving the way for sustainable, well-organized cities. Through PRAWN, we envision a future where smart city solutions contribute to smoother, more efficient urban life for residents and administrators alike.

# REFERENCES

* <https://arduino-esp8266.readthedocs.io/en/latest/>
* <https://ieeexplore.ieee.org/document/9077885>
* <https://www.researchgate.net/publication/341870728_Smart_Parking_System_based_on_IOT>
* <https://www.ijeat.org/wp-content/uploads/papers/v9i1/A1963109119.pdf>
* <https://webbylab.com/blog/smart-parking-system-using-iot/>
* <https://ieeexplore.ieee.org/document/7562735>