AY 2023-24

IV Semester

ITL406

A PROJECT REPORT
ON

Parkivia: Optimizing Urban Parking Spaces for Efficient City Mobility

Bachelor of Technology
in
School of Computing

By
Harshit Mishra (22323)

SCHOOL OF COMPUTING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA
HIMACHAL PRADESH

MAY 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled *Parkivia: Optimizing Urban Parking Spaces for Efficient City Mobility* is a bonafide record of the work done by

Harshit Mishra (22323)

in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in BRANCH NAME of the INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA, HIMACHAL PRADESH, during the year 2022 - 2026.

under the guidance of

Dr Nishtha Hooda

Project viva-voce held on: 26/04/2024

Internal Examiner External Examiner

ORIGNALITY / NO PLAGARISM DECLARATION

We certify that this project report is our original report and no part of it is copied from any published reports, papers, books, articles, etc. We certify that all the contents in this report are based on our personal findings and research and we have cited all the relevant sources which have been required in the preparation of this project report, whether they be books, articles, reports, lecture notes, and any other kind of document. We also certify that this report has not previously been submitted partially or as whole for the award of degree in any other university in India and/or abroad.

We hereby declare that, we are fully aware of what constitutes plagiarism and understand that if it is found at a later stage to contain any instance of plagiarism, our degrees may be cancelled.

Harshit Mishra (22323)

ABSTRACT

Parkivia: Optimizing Urban Parking Spaces for Efficient City Mobility" is a pioneering project aimed at transforming the landscape of urban parking management. By integrating user-friendly interfaces, RFID technology, and real-time data processing, this project seeks to revolutionize the parking experience. With a keen emphasis on enhancing user convenience and administrative efficacy, Parkivia introduces a suite of features including seamless parking booking, automated billing, and comprehensive management tools. Leveraging the robust backend infrastructure powered by Node.js and MongoDB, Parkivia promises to deliver a sophisticated solution tailored to the dynamic needs of modern cities.

The project's primary objective is to design and develop a comprehensive parking management system that seamlessly integrates user-facing features, administrative tools, and real-time monitoring through RFID technology. Addressing the challenge of efficient parking lot management, Parkivia aims to streamline booking processes, automate billing, and enhance communication between users and administrators. To achieve these goals, a diverse skill set is required, encompassing proficiency in programming languages such as JavaScript, HTML, CSS, Node.js, MongoDB, Tailwind CSS, as well as knowledge in IoT and Arduino. The project timeline spans four months, during which the acquisition of these skills is prioritized.

Key components of Parkivia include a user section (website) facilitating parking space bookings, an administrative section (website) for monitoring and managing parking lots, and an IoT module for RFID integration. The IoT module employs RFID sensors and readers strategically placed at entry and exit points of parking lots to detect vehicles, transmit data, calculate parking fees, and generate automated billing. Additionally, a notification system ensures users are promptly informed of their allocated parking space and payment details via email or SMS.

Backend development plays a crucial role in Parkivia, involving the establishment of a server using Node.js, database management utilizing MongoDB, and API development for seamless communication between the frontend and backend. Parkivia aspires to be a game-changer in urban parking management, offering a sophisticated yet user-friendly solution that optimizes city mobility and contributes to a more efficient urban infrastructure. Through rigorous testing, experimentation, and documentation, Parkivia aims to set new standards in parking management, paving the way for smarter, more sustainable cities.

***Keywords*:** RFID, Smart City, Parking System, IoT

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

We would like to express our sincere gratitude and heartfelt thanks to Dr Nishta Hooda for their unflinching support and guidance, valuable suggestions and expert advice. Their words of wisdom and expertise in subject matter were of immense help throughout the duration of this project.

We also take the opportunity to thank our director and all the faculty of School of Computing, IIIT Una for helping us by providing necessary knowledge base and resources.

We would also like to thank our parents and friends for their constant support.

Harshit Mishra (22323)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

RFID – Radio Frequency Identification

IoT – Internet of things

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In an era marked by urbanization and the relentless pace of technological advancement, the management of urban parking spaces has emerged as a critical challenge. Traffic congestion, wasted time searching for parking, and inefficient space utilization plague both drivers and city administrators alike. Parkivia is our ambitious endeavor to transform parking management through a cohesive blend of cutting-edge technology and user-centric design.

Our project sets out to revolutionize the way cities manage their parking infrastructure by leveraging the power of RFID technology, real-time data processing, and intuitive user interfaces. At its core, Parkivia aims to provide a seamless parking experience for users while empowering administrators with powerful tools for efficient space allocation and revenue management.

Fig 1.1 Reference image of parkivia project

1.2 Problem Definition:

The challenge at hand is multifaceted: designing and implementing a comprehensive parking management system that not only caters to the needs of users but also streamlines administrative processes. With the proliferation of vehicles in urban areas, the demand for parking spaces has never been higher. Parkivia seeks to address this by developing a system that can efficiently handle parking lot bookings, automate billing, and provide real-time monitoring capabilities.

1.3 Objectives:

Our primary objective is to design and implement a robust parking management system that seamlessly integrates user-facing features, administrative tools, and real-time monitoring. By doing so, we aim to streamline the booking process, automate billing procedures, and enhance communication between users and administrators.

1.4 Sections in the project:

User Section (Website):

Homepage: A user-friendly interface displaying available parking lots with pertinent details such as location, capacity, and real-time availability.

Booking System: Empowering users to select a parking lot, choose a specific space, and book it for a specified duration seamlessly.

Admin Section (Website):

Dashboard: Providing administrators with a comprehensive overview of all parking lots, including utilization rates, available spaces, and revenue generated.

RFID Integration: Utilizing RFID technology to monitor vehicle entry and exit in real-time, updating parking lot status accordingly.

IoT Module (RFID Integration):

RFID Sensors: Strategically placed at entry and exit points to detect vehicles.

RFID Reader: Identifying vehicles by reading RFID tags attached to them.

Data Transmission: Sending vehicle entry and exit data to the server for processing.

Automated Billing: Calculating parking fees based on entry and exit times, generating bills automatically.

Notification System: Keeping users informed about their allocated parking space and payment details via email or SMS.

Backend Development (Node.js, MongoDB):

Server Setup: Establishing a robust backend server using Node.js to handle client requests efficiently.

Database Management: Setting up a MongoDB database to store crucial information such as user details, parking lot information, bookings, and RFID data.

API Development: Creating APIs for seamless communication between the frontend and backend, managing user authentication, booking requests, and RFID data processing.

CHAPTER 2

LITERATURE REVIEW

Urban parking management is a critical aspect of city planning, aiming to optimize parking spaces for efficient city mobility while addressing challenges such as congestion and limited availability. Various solutions and products have been developed to tackle these issues, offering a range of features from real-time monitoring to automated billing. This literature review explores some of the existing solutions and products in urban parking management and evaluates their capabilities in meeting the objectives of the Parkivia project.

2.1 ParkMobile:

ParkMobile is a popular mobile parking app that allows users to find, reserve, and pay for parking spaces in cities across the world. The app provides real-time information on parking availability, pricing, and location, enabling users to make informed decisions about where to park. ParkMobile also offers features such as parking reminders and digital payments, enhancing user convenience and experience. While ParkMobile focuses primarily on user-facing features, its integration with backend systems and data analytics makes it a comprehensive solution for urban parking management.

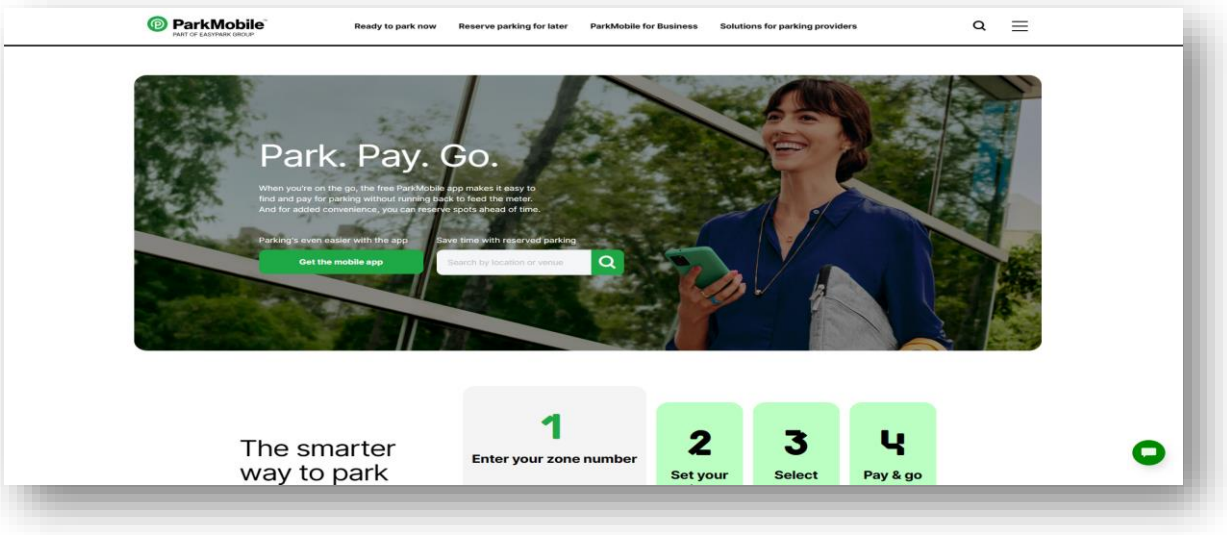

Fig 2.1 Landing page of ParkMobile

2.2 TIBA Parking Systems:

TIBA Parking Systems offers a range of products and solutions for parking management, including parking access control systems, revenue management software, and parking guidance systems. Their products utilize RFID technology, license plate recognition, and IoT sensors to monitor parking spaces, automate billing processes, and provide real-time data to administrators. TIBA's parking management solutions are widely used in commercial parking facilities, airports, and municipalities, demonstrating their effectiveness in optimizing parking operations and enhancing user experience.

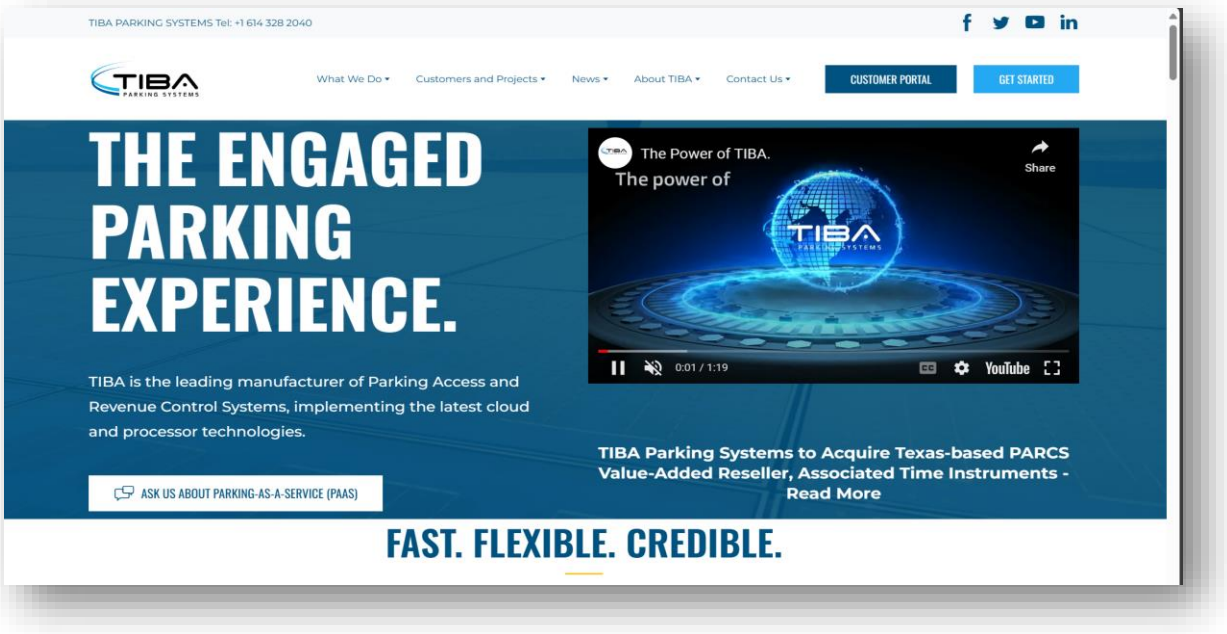

Fig 2.2 Landing page of ParkMobile

2.3 ParkAssist by ParkHelp:

ParkAssist is an intelligent parking guidance system developed by ParkHelp, designed to improve parking efficiency and reduce traffic congestion. The system utilizes ultrasonic sensors, LED indicators, and digital signage to guide drivers to available parking spaces quickly and efficiently. ParkAssist also offers backend analytics and reporting tools for administrators to monitor parking space utilization, revenue generation, and customer satisfaction. With its user-friendly interfaces and real-time monitoring capabilities, ParkAssist contributes to the optimization of urban parking spaces and city mobility

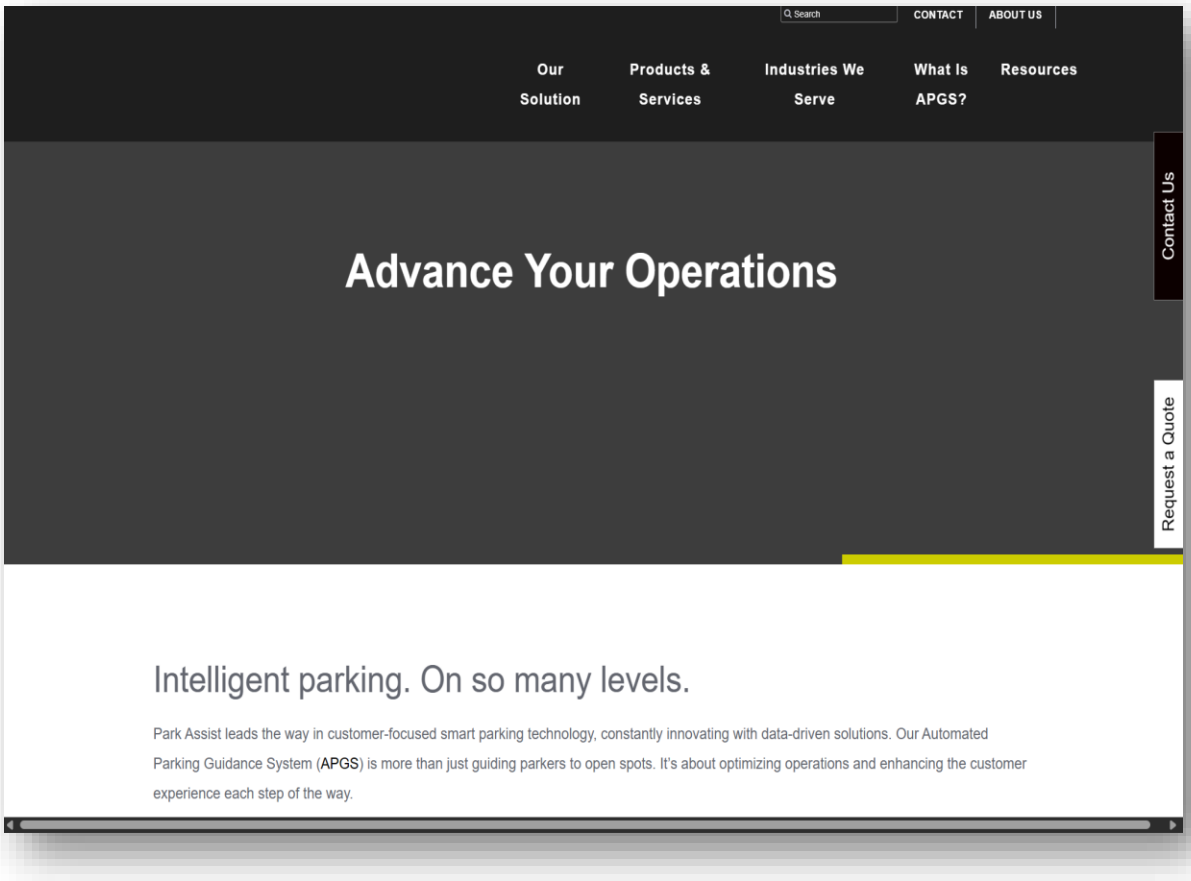

Fig 2.3 Landing page of ParkAssist

2.4 Flowbird:

Flowbird is a leading provider of smart parking and mobility solutions, offering a comprehensive suite of products and services for parking management. Their offerings include pay-by-plate parking meters, mobile payment apps, and parking enforcement tools. Flowbird's solutions leverage RFID technology, cloud-based platforms, and data analytics to streamline parking operations, improve revenue collection, and enhance user engagement. With a focus on user convenience and administrative efficiency, Flowbird's products contribute to the overall optimization of urban parking spaces and city mobility.

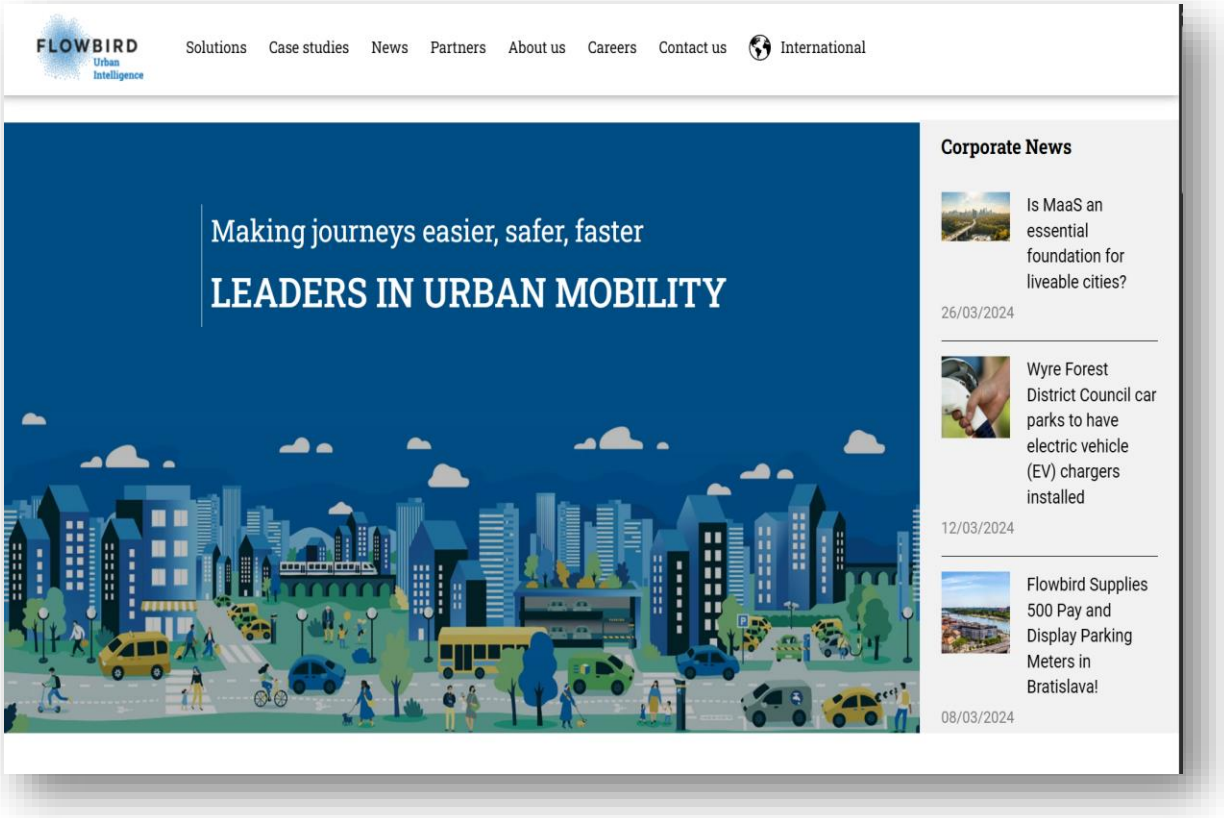

Fig 2.4 Landing page of FlowBird

2.5 ParkWhiz:

ParkWhiz is a parking reservation platform that allows users to book parking spaces in advance at various locations, including garages, lots, and valet services. The platform provides real-time availability information, pricing options, and detailed location maps to help users find and reserve parking spaces seamlessly. ParkWhiz also offers integration with navigation apps and digital payment systems for added convenience. By offering pre-booking options and real-time updates, ParkWhiz contributes to reducing parking congestion and improving overall city mobility.

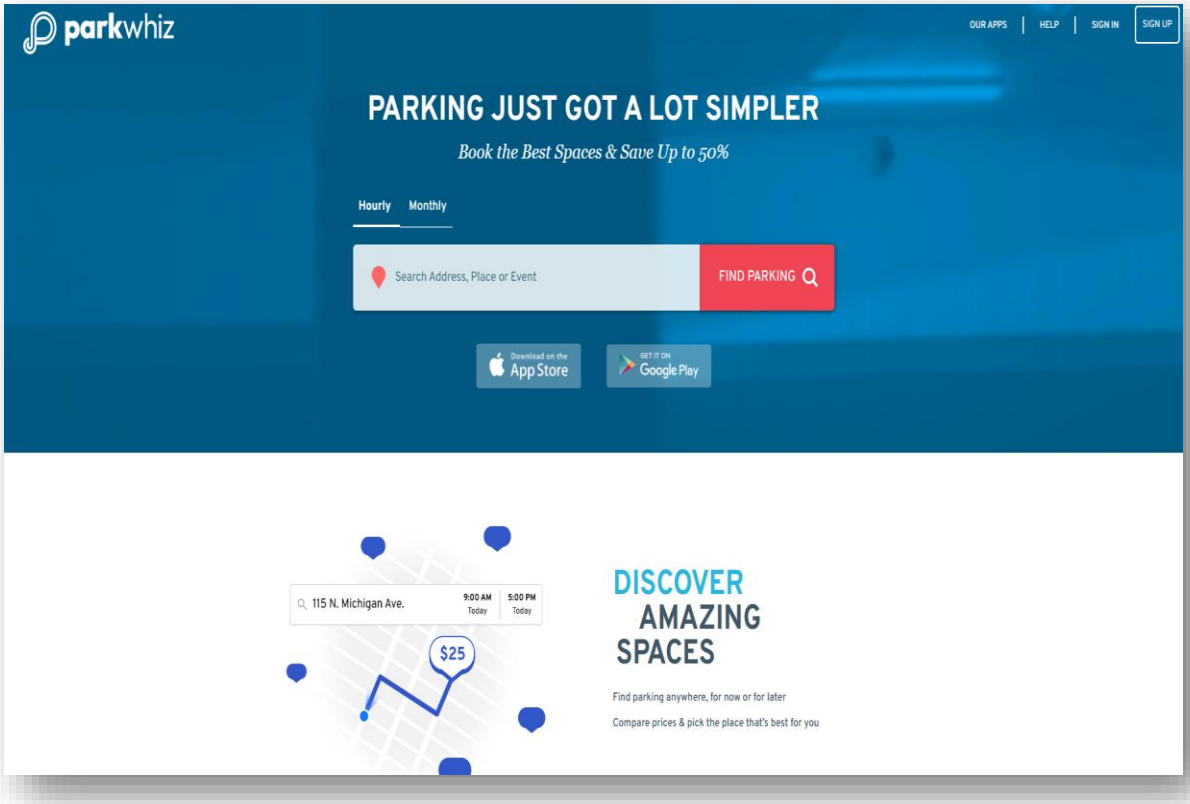

Fig 2.5 Landing page of ParkWhiz

CHAPTER 3

Worked accomplished in the Project

1. User Section:

1) Booking site:

The parking lots can be reserved via the website that is part of this project. Verify whether parking spaces are available at a specific area. From this website, users can immediately reserve parking lots.

Fig 3.1 Landing page of User section

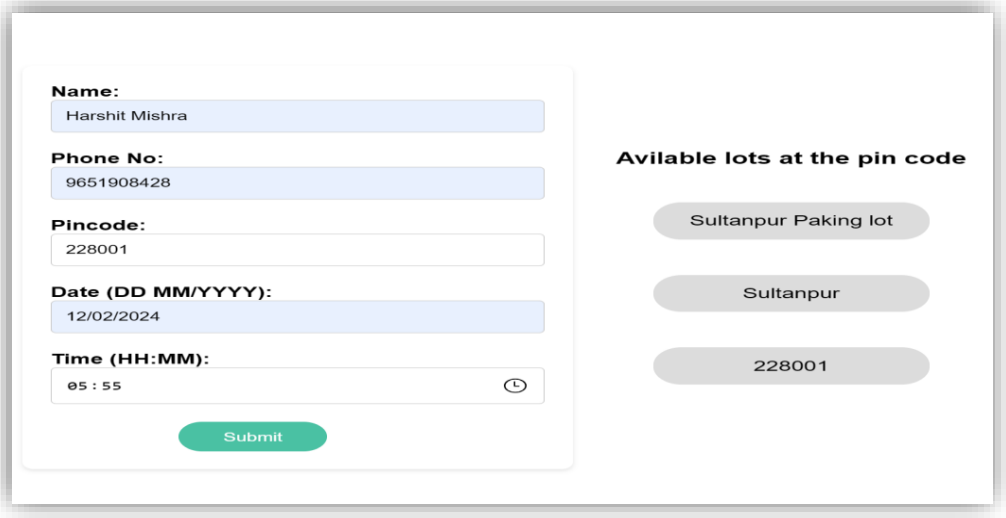

Fig 3.2 Booking and lot check page of User section

2. Admin section:

1) **Login Section:**

First, there's a login part that requests your email address and password. If they match, it maps the parking lot that is associated with that email address and displays lot data.

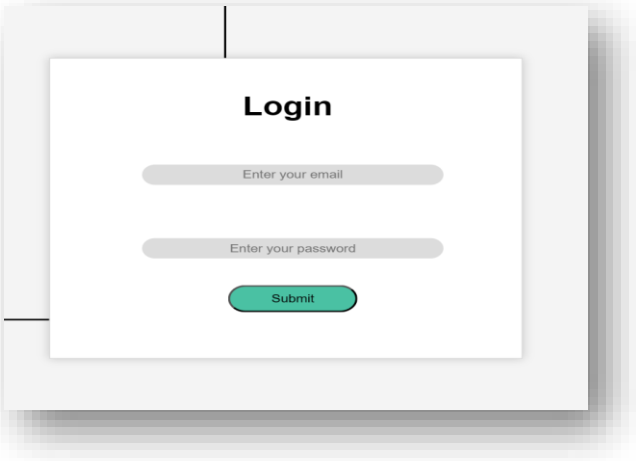

Fig 3.3 Login form of the admin section

2) Information panel:

The information panel employs a pie chart to illustrate the applications of parking lot details, such as occupancy location pin codes and usage.

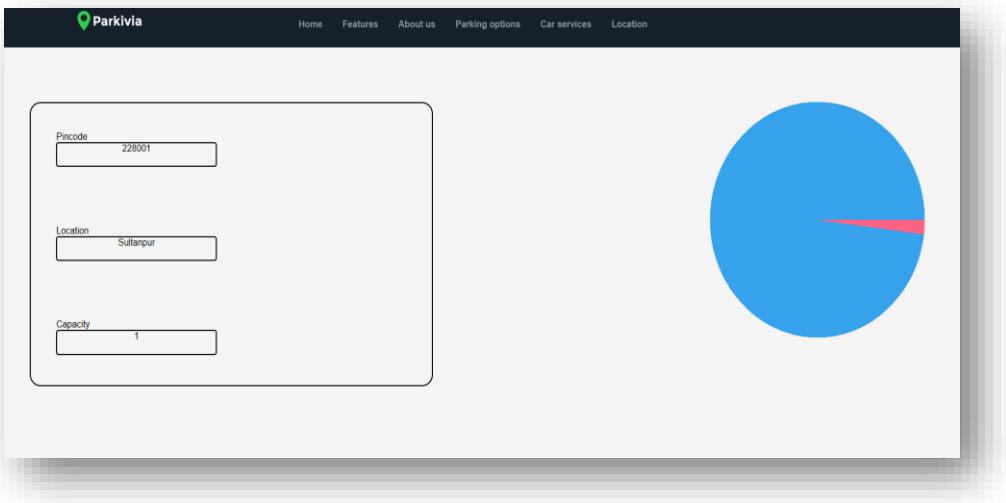

Fig 3.4 Data of the lot

3. IOT module:

In the proposed system, **RFID Sensors** are strategically placed at entry and exit points to detect vehicles. These sensors work in conjunction with an **RFID Reader**, which identifies vehicles by reading RFID tags attached to them. The system is designed for **Data Transmission**, sending vehicle entry and exit data to a server for processing. This data is then used for **Automated Billing**, where parking fees are calculated based on entry and exit times, and bills are generated automatically. Additionally, a **Notification System** is in place to keep users informed about their allocated parking space and payment details via email or SMS. This comprehensive system ensures efficient management of parking spaces and seamless user experience.

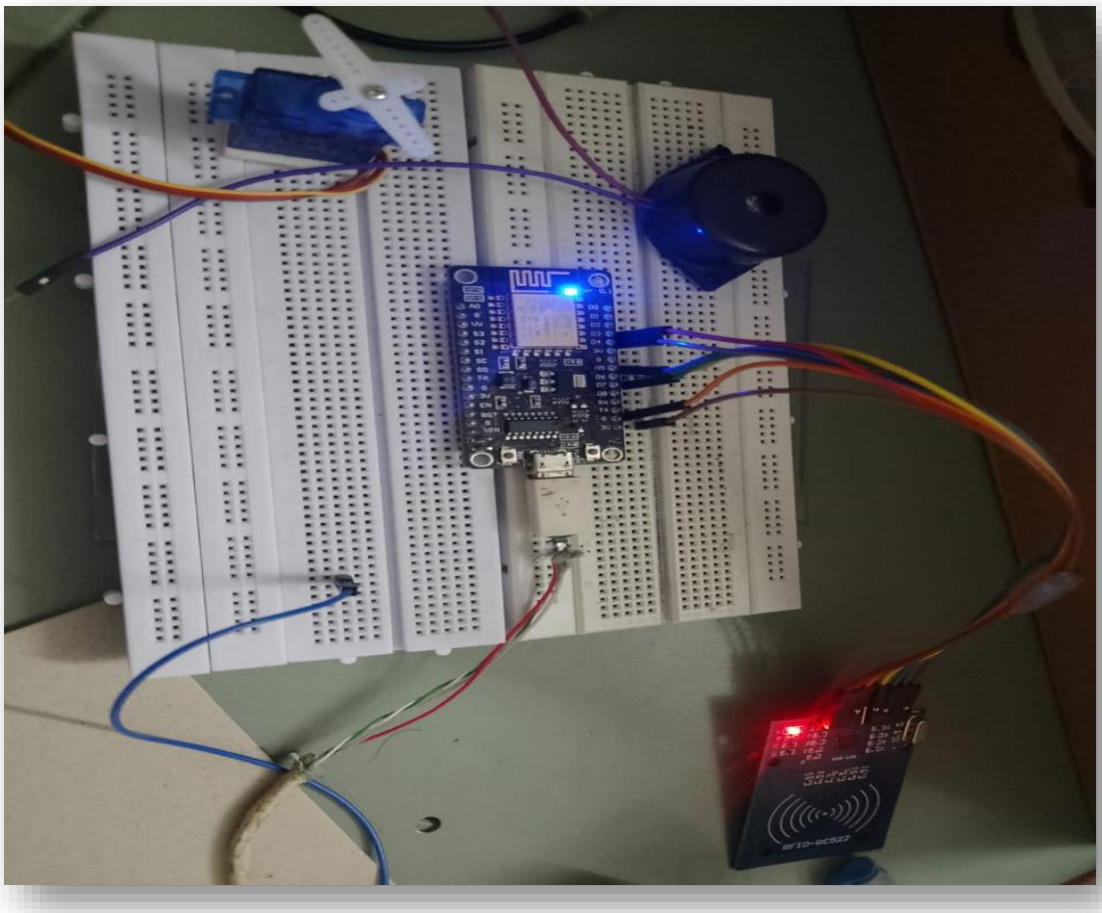

Fig 3.4 Landing page of User section

4. Backend Server:

In the proposed system, a robust backend server is established using **Node.js** to handle client requests efficiently. This is part of the **Server Setup**. A **MongoDB database** is set up for **Database Management**, which stores crucial information such as user details, parking lot information, bookings, and RFID data. **API Development** involves creating APIs for seamless communication between the frontend and backend. These APIs manage user authentication, booking requests, and RFID data processing. Additionally, the backend server also has the capability to send emails on booking, keeping users informed about their booking details. This comprehensive backend setup ensures efficient management and seamless user experience.

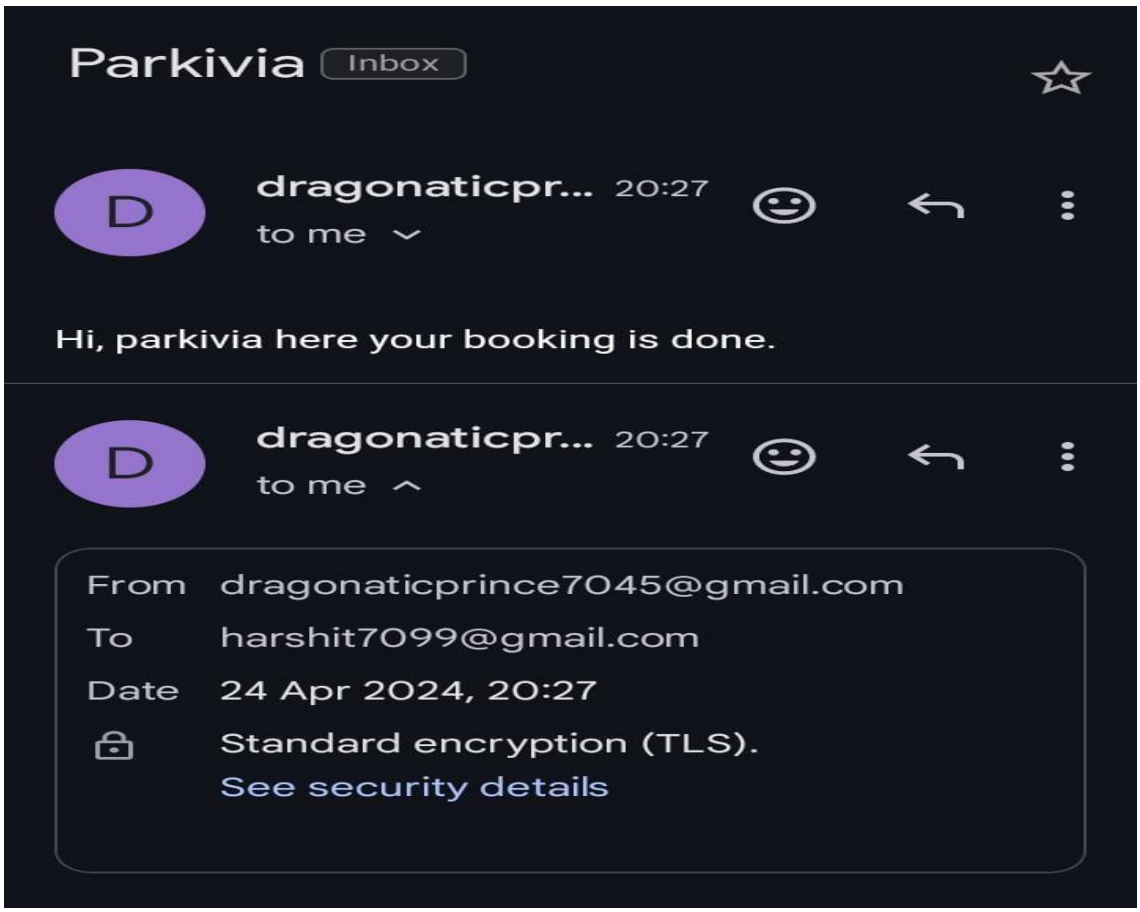

Fig 3.5 Mail sent from backend server

CHAPTER 4

SUMMARY AND CONCLUSION

4.1 SUMMARY:

Parkivia presents a cutting-edge solution to urban parking management, leveraging modern technologies to enhance user experience and optimize city mobility. By integrating user-facing interfaces, RFID technology, and real-time data processing, Parkivia aims to streamline parking operations, automate billing processes, and provide administrators with comprehensive management tools.

The project's foundation lies in its robust backend infrastructure powered by Node.js and MongoDB, ensuring scalability, reliability, and efficiency. The user section of the system offers a seamless experience, allowing users to view available parking lots, select specific spaces, and book them for desired durations. On the admin side, a comprehensive dashboard provides insights into parking lot utilization, available spaces, and revenue generated. Administrators can efficiently manage parking lots, add or edit details, and monitor real-time updates through RFID integration.

The IoT module, featuring RFID sensors and readers, enables automated vehicle detection and data transmission, facilitating real-time monitoring of parking lot status. Automated billing calculates parking fees based on entry and exit times, while a notification system keeps users informed about their allocated parking spaces and payment details via email or SMS.

4.2 CONCLUSION:

Parkivia offers a holistic solution to urban parking challenges, prioritizing user convenience, efficient administration, and data-driven decision-making. By revolutionizing parking management, Parkivia aims to contribute to more efficient city mobility, reduced congestion, and improved overall urban infrastructure.

4.3 DIRECTION FOR FUTURE WORK:

1. Advanced Analytics: Implement advanced analytics to provide insights into parking lot utilization, peak hours, and revenue generation. This could help in strategic decision-making and improving operational efficiency.
2. Dynamic Pricing: Introduce dynamic pricing based on demand and supply. Higher prices could be charged during peak hours and lower prices during off-peak hours to optimize revenue and space utilization.
3. Integration with Smart Cities: Explore opportunities to integrate the system with smart city infrastructure. This could provide a more holistic and efficient urban mobility solution.
4. Mobile Application: Develop a mobile application to provide users with real-time updates about parking space availability and allow them to make bookings on the go.
5. Enhanced Security: Incorporate advanced security measures such as CCTV integration and automatic number plate recognition (ANPR) systems to enhance security and prevent unauthorized access.
6. User Feedback System: Implement a user feedback system to continuously improve the service based on user suggestions and complaints.

REFERENCES

[1] Y. Agarwal, P. Ratnani, U. Shah and P. Jain, "IoT based Smart Parking System," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2021, pp. 464-470.

[2] https://sansad.in/getFile/lsscommittee/Housing%20and%20Urban%20Affairs/17_Housing_and_Urban_Affairs_21.pdf?source=loksabhadocs https://sansad.in/getFile/lsscommittee/Housing%20and%20Urban%20Affairs/17_Housing_and_Urban_Affairs_21.pdf?source=loksabhadocs

[3] ParkMobile ,Park Hub

Appendices

Appendix A

Code Attachments

A.1 HTML code snippet

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Parkivia</title>
    <link rel="icon" href="/images/car.png" type="image/x-icon" />



    <link rel="stylesheet" href="../css/homepage.css">
</head>

<body>
    <div id="header">
        <div id="header_image">
            <img src="/images/logo-inv.webp" alt="">
        </div>
        <div id="header_elements_div">
            <div class="header_elements">Home</div>
            <div class="header_elements">Features</div>
            <div class="header_elements">About us</div>
            <div class="header_elements">Parking options</div>
            <div class="header_elements">Car services</div>
            <div class="header_elements">Location</div>
        </div>
```

```
<div id="popupContainer" class="popup">
    <div class="container">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name">

        <label for="phone">Phone No:</label>
        <input type="tel" id="phone" name="phone">

        <label for="pincode">Pincode:</label>
        <input type="text" id="pincode" name="pincode">

        <label for="date">Date (DD MM/YYYY):</label>
        <input type="text" id="date" name="date" placeholder="DD MM/YYYY">

        <label for="time">Time (HH:MM):</label>
        <input type="time" id="time" name="time">

        <button id="closeBtn">Submit</button>
    </div>



    <div id="avilable_lots">
        <h3>Avilable lots at the pin code</h3>
        <div id="lotName" class="avilable_items">h</div>
        <div id="lotlocation" class="avilable_items">h</div>
        <div id="lotPincode" class="avilable_items">h</div>
    </div>
```

```
        </div>
    </div>
    </div>
    <div id="booking_form">
        <div class="booking_form_objects">
            <div>Select parking lot</div>
        </div>
        <input type="text" id="fname" placeholder="Enter your phone.No"
class="booking_form_objects">
        <input type="text" id="cname" placeholder="Enter pincode"
class="booking_form_objects">
        <div class="booking_form_objects"><input value="BOOK" type="submit" id="sub"
class="booking_form_objects"></div>

    </div>
    <div id="image">

        <div id="image_div_text">
            <div id="image_div_text1">You Can't <br> Park Closer</div>
            <div id="image_div_text2">Instantly book your space today. <br>Trusted by
millions.</div>

        </div>
    </div>
    <div id="features">
        <div class="feature_block">
            <div class="main">
                <div class="feature_image">
                    <img src="/images/cctv-camera.png" alt="" class="icon">
                </div>
```

```
        <div class="heading1">Safe And <br> Secure</div>
    </div>

    <div class="feature_text">
        24 hour surveillance to ensure your car is safe and secure while you are away.
    </div>
</div>
<div class="feature_block">
    <div class="main">
        <div class="feature_image">
            <img src="/images/car.png" alt="" class="icon">
        </div>
        <div class="heading1">Car <br> Services</div>
    </div>
    <div class="feature_text">
        Simply drive up and go with our Parkivia service. Why not add a car service?
    </div>
</div>
<div class="feature_block">
    <div class="main">
        <div class="feature_image">
            <img src="/images/key.png" alt="" class="icon">
        </div>
        <div class="heading1">Concierge <br>Options</div>
    </div>
    <div class="feature_text">
        Take advantage of our Concierge services and have us help you out.
    </div>
</div>
<div class="feature_block">
    <div class="main">
```

```
            <div class="feature_image">
                <img src="/images/map.png" alt="" class="icon">

            </div>
            <div class="heading1">Close And<br> Handy</div>
        </div>
        <div class="feature_text">Park at the terminal – just a short walk and you are at your departure gate.
        </div>
    </div>
</div>
<div id="why_choose">
    <div id="wc_img">
        <img src="/images/wOMEN_IN_CAR.jpg" alt="">
    </div>
    <div id="wc_question">
        <div id="wc1">
            Why Choose <br> Parkivia?</div>
        <div id="wc2">
            Short Stay and Valet parking options are just a few minutes' walk away from the terminal. If you're
            opting for Long Stay, the car park is just 10 minutes away by bus and shuttles run every 10-15 minutes.


        </div>
    </div>
</div>
<div id="map"></div>
<div id="below_map">
    <div id="sub_below_map">
```

```
<div class="block">
   <div class="hed">
      No Stress
   </div>
   <div class="content">
      <div class="icon1">
         <img src="/images/like.png" alt="">
      </div>
      <div class="content_text">
         Save up to 70% on our site compared to the cost of on-airport parking.
      </div>
   </div>

</div>
```

A.2 Js Server code:

```
// Import necessary modules
import express, { response } from 'express';
import { connectMongoDB } from './mongodb.js';
import { addParkiingLot_repo, booking_repo, user,pakingSingninRepo } from './model.js';
// Import repository
import cors from 'cors';
import { sendbookingEmail } from './mailer.js';
import { checkBooking, checkEntered, dele, makeBooking, updateBooking } from
'../jvavascript/iot.js';
```

```
const app = express();
app.use(cors());
app.use(express.json()); // Middleware to parse JSON in request body

// POST endpoint to book
app.post('/book', async (req, res) => {
  try {
    console.log('Booking request received');
    const data = {
      phoneNo: req.body.phone,
      city: req.body.pincode,
      time: new Date(req.body.date), // Parse req.body.date into a Date object
      entry: req.body.entry,
    };
    const userInput = new user();
    const userdata = await userInput.findUserPhone(req.body.phone)
    sendbookingEmail(userdata[0].email, userdata[0],req);
    console.log(data.time.toTimeString()); // Accessing hours from the parsed Date object
    const booking = new booking_repo();
    await booking.book(data); // Assuming booking.book() is an asynchronous operation
    console.log("Response sent for '/book'");
    console.log(data);
    res.json({ message: 'Booking successful' });
  } catch (error) {
    console.error('Error booking:', error.message);
    res.status(500).json({ error: error.message });
  }
});

// PUT endpoint to find parking lot
app.put('/find', async (req, res) => {
```

```
  try {
    const { date } = req.body;
    if (!date) {
      throw new Error('Date is required');
    }
    const parkingLotRepo = new addParkiingLot_repo();
    const result = await parkingLotRepo.findlot(date);
    res.json({ message: 'Success', value: result });
    console.log('Parking lot found:', result);
  } catch (error) {
    console.error('Error finding parking lot:', error.message);
    res.status(500).json({ error: error.message });
  }
});

// PUT endpoint to handle IoT data
app.put('/iot', async (req, res) => {
  console.log(req.body.message);
  let cbook = await checkBooking(req);
  if (cbook == true) {
    console.log('booking_found');
    let centerd = await checkEntered(req);
    if (centerd == true) {
      console.log('entered');
      dele(req);
    } else {
      console.log('not_entered');
      updateBooking(req);
    }
  } else {
    console.log('booking_not_found');
```

```
    makeBooking(req);
  }
});
app.put('/admin', async (req, res) => {
  console.log(req.body.email);
  let log=new pakingSingninRepo;
  let user=await log.findpakingowner(req.body.email);
  //console.log(user);
  let lot= new addParkiingLot_repo;
  let lotdata=await lot.findlot(user[0].lotId);
  //console.log(lotdata)
  const per= (user[0].password==req.body.password);
  res.json({
    login:per,
    owner:user[0],
    lotd:lotdata[0]
  });

});
const PORT = process.env.PORT || 8000;
app.listen(PORT, async () => {
  try {
    await connectMongoDB(); // Connect to MongoDB Atlas
    console.log('Connected to MongoDB Atlas');
    console.log(`Server is running on port ${PORT}`);

  } catch (error) {
    console.error('Error connecting to MongoDB Atlas:', error);
  }
});
```

A3 Admin page code:

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

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pie Chart</title>
    <link rel="stylesheet" href="../css/homepage.css">
    <link rel="stylesheet" href="../css/adminpage.css">
</head>

<body>
    <div id="header">
        <div id="header_image">
            <img src="/images/logo-inv.webp" alt="">
        </div>
        <div id="header_elements_div">
            <div class="header_elements">Home</div>
            <div class="header_elements">Features</div>
            <div class="header_elements">About us</div>
            <div class="header_elements">Parking options</div>
            <div class="header_elements">Car services</div>
            <div class="header_elements">Location</div>
        </div>
    </div>
    <div id="lot">
        <div id="lotDetails">
            <div class="dele">
                <div class="detailheading">Pincode</div>
                <div class="detailselement" id="pin">p</div>
```

```
        </div>
        <div class="dele">
            <div class="detailheading">Location</div>
            <div class="detailselement" id="locat">L</div>
        </div>
        <div class="dele">
            <div class="detailheading">Capacity</div>
            <div class="detailselement" id="cap">C</div>
        </div>

    </div>
    <div id="pie"><canvas id="pieChart" width="400" height="400"></canvas>
    </div>
</div>


<div id="lookOutForm">
    <h1>Login</h1>
    <input type="text" id="lot_email" placeholder="Enter your email">
    <input type="text" id="password" placeholder="Enter your password">
    <input type="submit" id="parkingLotSignin">
</div>
<canvas id="pieChart" width="400" height="400"></canvas>

<div class="chart">
    <div class="bar" style="height: 100px;"></div>
    <div class="bar" style="height: 200px;"></div>
    <div class="bar" style="height: 150px;"></div>
    <div class="bar" style="height: 250px;"></div>
</div>
```

```
    <script
src="/jvavascript/admin.js"></script>
</body>

</html>
```