ParkMate-Easy Way To Find Nearest Parking Zone

by

Md. Niamat Ullah
ID: CSE2101022119

Sahadat Hossain Apu
ID: CSE2101022022

Md. Hafijul Islam
ID: CSE2002020040

Poretosh Chandra Das
ID: CSE2101022025

Tania Roy
ID: CSE2101022042

Supervised by

Sadia Tasnim Barsha
Lecturer & Exam Coordinator
Department of CSE

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

October 2024

ParkMate-Easy Way To Find Nearest Parking Zone

by

Md. Niamat Ullah
ID: CSE2101022119

Sahadat Hossain Apu
ID: CSE2101022022

Md. Hafijul Islam
ID: CSE2002020040

Poretosh Chandra Das
ID: CSE2101022025

Tania Roy
ID: CSE2101022042

Supervised by

Sadia Tasnim Barsha
Lecturer & Exam Coordinator
Department of CSE

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

October 2024

APPROVAL

The project titled “ParkMate-Easy Way To Find Nearest Parking Zone” submitted by Md. Niamat Ullah (CSE2101022119), Sahadat Hossain Apu (CSE2101022022), Md. Hafijul Islam (CSE2002020040), Poretosh Chandra Das (CSE2101022025) and Tania Roy (CSE2101022042) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Sadia Tasnim Barsha
Lecturer & Exam Coordinator,
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Sadia Tasnim Barsha,** Lecturer & Exam Coordinator**,** Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Sadia Tasnim Barsha)
Supervisor

Signature

Md. Niamat Ullah
ID: CSE2101022119

Sahadat Hossain Apu
ID: CSE2101022022

Md. Hafijul Islam
ID: CSE2002020040

Poretosh Chandra Das
ID: CSE2101022025

Tania Roy
ID: CSE2101022042

ABSTRACT

ParkMate platform tackles the pressing challenges of urban parking by connecting vehicle owners seeking parking spaces with parking owners who have available, unused spaces. By leveraging real-time location-based services, ParkMate allows users to effortlessly locate, reserve, and pay for parking while providing parking owners with the opportunity to monetize their underutilized spaces. This paper outlines the system's architecture, functionality, and the technologies employed in the development of the platform. Through a user-friendly interface and efficient management processes, ParkMate aims to enhance parking solutions in urban areas, ultimately contributing to a more organized and accessible city environment.

ACKNOWLEDGMENT

At the very beginning, we would like to express my deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the schedule time.

We are auspicious that we had the kind association as well as supervision of **Sadia Tasnim Barsha**, Lecturer & Exam Coordinator, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary recourse to carry out our project.

We would like to convey our special gratitude to **Bulbul Ahamed**, Associate Professor and Head, Department of Computer Science and Engineering for his kind concern and precious suggestions.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to my parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

API	Application programming interface
CPU	Central Processing Unit
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
MySQL	My Structured Query Language
RAM	Random Access Memory
ROM	Read Only Memory
UI	User Interface

TABLE OF CONTENTS

Title		Page No.
DECLARATION		iii
ABSTRACT		iv
ACKNOWLEDGEMENT		v
LIST OF ABBREVIATION		vi
CHAPTER 1		1 – 2
INTRODUCTION TO "ParkMate"		
1.1	Introduction	1
1.2	Project Description	1
1.3	Project Tools ……..…….....	2
1.4	Project Objectices	2
CHAPTER 2		3-5
SYSTEM OVERVIEW		
2.1	System Architecture …….….....	3
2.2	System Modules …….….....	3
2.3	System Features	4
2.4	Data Flow.....	4
2.5	Technologies Used.....	4
2.6	Limitations of the System.....	5
CHAPTER 3		6-8
IMPLEMENTATION		
3.1	Front-End Development.....	6
3.2	Back-End Development	6
3.3	Database Design.....	6-7
3.4	Parking Link Sharing Integration.....	7
3.5	Payment Gateway Integration.....	7
3.6	Version Control with Git.....	7-8

CHAPTER 4 9-11

TESTING AND RESULTS

4.1 Testing Methodology..... 9

4.2 Testing Tools..... 9

4.3 Test Cases and Results 10

4.4 Bug Fixes and Enhancements..... 10-11

4.5 User Feedback..... 11

CHAPTER 5 12-14

DISCUSSION AND ANALYSIS

5.1 Additional Challenges Faced Faced..... 12

5.2 Additional Solutions Implemented..... 12

5.3 Lessons Learned..... 13

5.4 Future Improvements..... 13-14

CHAPTER 6 15-16

CONCLUSION AND FUTURE WORKS

6.1 Conclusion 15

6.2 Future Works 15-16

REFERENCES ……..... 17

Fial Steps For Completion…… 32

LIST OF FIGURES

Figure No.	Title	Page No.
Fig.1.1	System Architecture Diagram.....	18
Fig.2.1	Database Schema.....	19
Fig.3.1	Sample Code Snippets.....	20-21
Fig.3.2	Parking search functionality.....	22
Fig.3.3	Reservation handling processes.....	23-24
Fig.4.1	User Interface Screenshots-Frontend.....	25-28
Fig.4.2	User Interface Screenshots-Frontend (Client View).....	29-30
Fig.4.3	User Interface Screenshots-Backtend (Admin View).....	30-31

CHAPTER 1

INTRODUCTION TO "PARKMATE"

1.1 Introduction

Finding available parking in urban areas is a persistent challenge for vehicle owners, particularly in high-traffic locations where demand often exceeds supply. At the same time, parking owners frequently have unused spaces that could be transformed into profitable assets. This project proposes a solution by developing ParkMate, a web-based platform designed to connect vehicle owners in search of parking with parking owners offering their available spaces for rent. The platform acts as a digital marketplace, bridging the gap between parking demand and supply. With a user-friendly interface and real-time location-based services, ParkMate aims to enhance convenience and accessibility for users, ultimately streamlining the parking experience in urban environments.

1.2 Project Description

The ParkMate project involves the development of a fully functional web platform aimed at simplifying the search for parking spaces while allowing parking owners to list their available spots for rent. The system features two distinct user roles: vehicle owners seeking parking and parking owners offering their unused spaces. Key functionalities of the platform include location-based search capabilities, user management, parking spot reservations, and real-time updates on parking availability. By creating an efficient and user-friendly interface, ParkMate seeks to enhance the parking experience for users and optimize the utilization of available parking resources in urban areas.

1.3 Project Tools

The following tools and technologies were used in the development of the ParkMate project:

- **HTML**: Provides the structure for the web pages. [1] [2]
- **CSS**: Styles the user interface for an appealing design. [2]
- **JavaScript**: Adds interactivity and dynamic content to enhance user experience.
- **jQuery**: Simplifies DOM manipulation and event handling.
- **Laravel**: Serves as the backend framework for data handling and user authentication. [4]
- **MySQL:** Functions as the database management system for storing user and parking data. [3]
- **Map Link Sharing:** Enables parking owners to share links to their available parking spaces.
- **Git:** Manages version control to track changes in the project. [4]
- **GitHub:** Hosts the project repository, enabling collaboration and version management.

- **Visual Studio Code:** Acts as the primary development environment for coding. [5]

1.4 Project Objectives

- **Facilitate Easy Access:** Empower vehicle owners to effortlessly locate and reserve nearby parking spaces, enhancing their overall parking experience.
- **Monetization Opportunities:** Enable parking owners to effectively generate income by listing their unused spaces, turning idle assets into profitable opportunities.
- **Enhance Urban Parking Management:** Streamline the parking process by significantly reducing the time vehicle owners spend searching for available spots, contributing to better urban mobility.
- **Scalable and User-Friendly Design:** Develop a robust, intuitive platform that can easily adapt to diverse urban environments, ensuring a seamless user experience for all.
- **Secure Transactions:** Implement secure and efficient transaction processes to foster trust and reliability between vehicle owners and parking owners.

CHAPTER 2

SYSTEM OVERVIEW

2.1 System Architecture

Parkmate's architecture integrates front-end and back-end components:

- **Front-End**: Developed using HTML, CSS, and JavaScript, the platform features an intuitive and responsive user interface that facilitates the search and booking of parking spots..

- **Back-End**: Powered by Laravel, the back end manages user authentication, business logic, and communication between the database and front-end components.

- **Database**: MySQL serves as the database management system, handling data related to users, parking spots, reservations, and listings.

- **Link Sharing**: Parking owners can share links to their available parking spaces, allowing vehicle owners to easily access and reserve spots without the need for an external API.

2.2 System Modules

- **User Module:** Manages user registration, login, and overall management for both vehicle owners and parking owners. It ensures that users are authenticated before they can proceed with booking a parking space.
- **Parking Search Module:** Facilitates location-based searches for available parking spots, allowing users to search by the city name registered by parking owners. This module provides users with a seamless way to find suitable parking options in their desired locations.
- **Reservation Module:** Handles the booking process, allowing users to select an estimated duration for their reservation. It calculates the total cost by multiplying the unit price by the number of hours booked. The module also ensures that if users park for less time than booked, any excess payment is refunded automatically.
- **Parking Owner Management Module:** Empowers parking owners to list, price, and manage their parking spots effectively. They can view earnings and monitor reservations, while also managing their parking space details from an intuitive dashboard.
- **Admin Dashboard:** Displays all bookings made by users, providing an overview of parking availability and transactions. This module assists in managing the overall system and ensuring a smooth user experience for both vehicle and parking owners.

2.3 System Features

- **Real-Time Parking Availability:** Displays live data on available parking spots, allowing users to see which spaces are currently open for reservation.
- **User-Friendly Interface:** Offers an intuitive, easy-to-navigate platform where vehicle owners can search for parking spots based on the city name registered by parking owners.
- **Separate Dashboards:** Provides distinct dashboards for vehicle owners and parking owners, enabling them to manage reservations, listings, and monitor earnings efficiently.
- **Booking Notifications:** Sends alerts to users regarding reservation confirmations, updates, and reminders about their bookings.
- **Secure Transactions:** Ensures the safety of user data and payment information through encryption and secure payment methods, fostering trust between vehicle owners and parking owners.

2.4 Data Flow

1. **Search Request:** The user initiates a search for parking by entering a desired location.
2. **Data Retrieval:** The request is processed by the back end, which queries the database for available parking spots.
3. **Display Results:** The front end presents the available parking spots to the user, enhanced with a visual map interface for easier navigation.
4. **Reservation Processing:** When a reservation is made, the details are stored in the database, and both the vehicle owner and parking owner receive notifications.
5. **Real-Time Updates:** Parking availability is continuously updated in real time, ensuring that users have access to the most current information.

2.5 Technologies Used

- Front-End: HTML, CSS, JavaScript, jQuery. [6] [7]

- Back-End: Laravel.

- Database: MySQL. [8]

- Link Sharing: Allows parking owners to share links to their available parking spaces directly with users.

- Version Control: Git.

- Development Environment: Visual Studio Code.

2.6 Limitations of the System

- **Scalability Challenges:** The system may face scalability issues when accommodating a high volume of users, particularly in densely populated urban areas where parking demand is elevated.
- **Mapping Limitations:** The link-sharing feature may encounter challenges in areas with limited internet connectivity or where users may not have access to mapping services, impacting the ease of finding available parking spaces.
- **Real-Time Data Updates:** While the system aims to provide real-time availability, there may be occasional delays in data updates due to network connectivity issues, which could affect the accuracy of parking spot information.

CHAPTER 3

IMPLEMENTATION

3.1 Front-End Development

The front end of ParkMate was crafted using HTML, CSS, and JavaScript to deliver a responsive and user-friendly interface. Key features include:

- **Responsive Design:** Implemented to ensure seamless compatibility across a variety of devices and screen sizes, providing an optimal user experience on smartphones, tablets, and desktops.
- **User Interface (UI):** Designed with a clean and intuitive layout that allows users to navigate the platform effortlessly, enhancing usability and accessibility.
- **Interactive Elements:** Leveraged JavaScript to incorporate interactive features, such as dynamic search results and real-time updates, enabling users to engage with the platform in a more meaningful way.

3.2 Back-End Development

The back end of ParkMate is built on the Laravel framework, providing a robust and scalable structure for managing server-side operations. [6] Key components include:

- **Secure User Authentication:** A robust system for registering and logging in users, ensuring that both vehicle owners and parking owners have protected accounts with secure access to the platform.
- **Core Business Logic:** The heart of the system, managing essential functions such as listing parking spaces, processing reservations, and calculating payments. This ensures smooth operation and accurate handling of user transactions.
- **Link Sharing for Parking Owners:** Instead of relying on external APIs, parking owners can generate and share unique links for their available parking spots, simplifying the reservation process for users.

3.3 Database Design

The MySQL database for ParkMate is structured to efficiently manage user data, parking information, reservations, payments, and system operations. [3] The key tables include:

- **Users:** Stores information about vehicle owners and parking owners, including credentials, roles, and other personal details.
- **Roles:** Defines user roles, distinguishing between vehicle owners, parking owners, and administrators for system access control.
- **Parking:** Contains detailed information about available parking spaces, including location, pricing, and availability status.

- **Books:** Manages reservation data, tracking the parking spot ID, user ID, start and end times of bookings, and the calculated total cost for each reservation.
- **Booking Payments:** Handles all transaction data, storing details about payments made for reservations, including amounts, timestamps, and refund processes where applicable.
- **Contact Forms:** Stores messages and inquiries submitted through the platform's contact form.
- **Personal Access Tokens & Password Resets:** These tables ensure secure user authentication and password management, providing safe access to the platform.
- **Migrations & Failed Jobs:** Tracks system migrations and logs any job failures to ensure smooth system operations and troubleshooting.

3.4 Parking Link Sharing Integration

Instead of relying on an API, ParkMate incorporates a link-sharing mechanism that allows parking owners to provide direct access to their available parking spaces. This approach simplifies the reservation process and ensures ease of use for both parking owners and vehicle owners. Key functionalities include:

- **Link Sharing:** Parking owners can generate and share unique links to their parking spots, which vehicle owners can use to access the parking details directly.
- **Location-Based Search:** Vehicle owners can search for parking spaces by city name. Once a city is selected, a list of available parking spots is displayed, each with a link for further details and booking.
- **User Navigation:** The shared link provides parking spot details, booking options, and payment processing, streamlining the entire reservation experience.

3.5 Payment Gateway Integration

ParkMate utilizes a secure card payment system to process transactions between vehicle owners and parking owners. Key features include:

- **Card Payments Only:** Currently, the platform supports credit and debit card payments for booking parking spaces.
- **Security**: Employs encryption and secure protocols to safeguard users' financial data during transactions.
- **Transaction Management**: Efficiently tracks payments, processes refunds when necessary, and maintains financial records for accurate reporting.

3.6 Version Control with Git

Git was implemented as the version control system for managing code changes and ensuring smooth collaboration throughout the development of ParkMate. Key aspects include:

- **Centralized Repository:** A single repository was maintained to store and manage all project code, ensuring consistent access for developers.
- **Branching and Merging:** Developers were able to create separate branches for working on new features or fixes, enabling parallel development without conflicts. Merging branches ensured all changes were integrated smoothly.
- **Commit History:** Every code modification was documented through commits, allowing for a detailed history of project progress, which helps in tracking and troubleshooting.

CHAPTER 4

TESTING AND RESULTS

4.1 Testing Methodology

A comprehensive testing process was implemented to ensure the functionality, security, and performance of the ParkMate platform. The following testing methods were employed:

- **Unit Testing:** Each individual component, such as user authentication, booking, and payment processing, was tested in isolation to verify their correctness and reliability.
- **Integration Testing:** Modules like user registration, parking search, and reservation management were tested together to confirm their interaction and data flow across the system.
- **System Testing:** The complete platform was evaluated as a whole to ensure it met all functional and non-functional requirements, including security and performance benchmarks.
- **User Acceptance Testing (UAT):** Real users, comprising vehicle owners and parking owners, tested the platform to validate usability, user experience, and overall functionality from an end-user perspective.
- **Performance Testing:** The system was subjected to varying load conditions to assess its responsiveness, identify bottlenecks, and evaluate scalability during high-demand scenarios.
- **Security Testing**: A dedicated security assessment was conducted to identify vulnerabilities in the system, ensuring that sensitive user data and transactions were protected against potential threats.
- **Compatibility Testing:** The platform was tested across multiple devices, operating systems, and browsers to ensure consistent functionality and appearance.
- **Regression Testing:** After each update or bug fix, regression testing was performed to ensure that existing functionalities remained unaffected.

4.2 Testing Tools

The following tools were utilized to ensure the quality, functionality, and performance of the ParkMate platform:

- **PHP Unit:** Employed for unit and integration testing within the Laravel framework to verify backend logic, including user authentication, booking, and payment processes. [7] [11] [13]

- **Manual Testing:** Conducted to validate the front-end user interface (UI) and user interactions, ensuring seamless workflows for searching for parking, making reservations, and processing payments.
- **Browser Developer Tools:** Utilized for testing and debugging front-end components (HTML, CSS, JavaScript) to ensure responsiveness and proper functionality across various devices and browsers. [1] [14]
- **Database Testing:** Manually performed to verify the accuracy of data storage and retrieval for users, reservations, and payment records within the MySQL database. [15]lara
- **Load Testing Tools:** Tools such as JMeter or Locust were used to simulate high traffic scenarios and assess how the platform handles peak loads.
- **Security Testing Tools:** Software like OWASP ZAP or Burp Suite was employed to conduct security assessments, identifying vulnerabilities and ensuring robust data protection.
- **Cross-Browser Testing Tools:** Tools like BrowserStack or Sauce Labs were used to validate the platform's performance and appearance across different browsers and devices, ensuring compatibility.

4.3 Test Cases and Results

- **User Registration and Login:** Users successfully registered and authenticated without any security vulnerabilities. The system effectively enforced login requirements before allowing bookings.
- **Parking Space Search**: Users could accurately search for parking spaces based on city names provided by parking owners. Search results were returned swiftly, ensuring minimal delay in user experience.
- **Reservation Process**: The booking process was seamless, allowing users to select an estimated duration for their parking needs. Payments were processed securely, and the system correctly calculated total costs, automatically refunding any excess payment if users parked for less than the reserved time.
- **Dashboard Functionality**: Both vehicle owners and parking owners had access to user-friendly dashboards. Vehicle owners could view their reservations, while parking owners effectively managed their listings and earnings, with real-time updates.
- **Performance Under Load**: The system performed optimally with up to 500 concurrent users, maintaining responsiveness. Beyond this threshold, minor delays were noted in processing bookings and updating reservation statuses.

4.4 Bug Fixes and Enhancements

During the testing phase, several bugs were identified and addressed to enhance the functionality and user experience of the ParkMate platform:

- **UI Glitches:** Corrected alignment and responsiveness issues to ensure a consistent and visually appealing interface across all device sizes, particularly for mobile users.
- **Reservation Conflicts:** Implemented a locking mechanism to prevent multiple users from booking the same parking spot simultaneously, ensuring that each reservation is honored without conflict.
- **Payment Failures:** Enhanced the payment processing system with improved error handling and retry logic for failed transactions, reducing user frustration and increasing successful payment rates.
- **Data Syncing:** Optimized the real-time data update processes to ensure timely reflections of parking availability, significantly minimizing delays and providing users with accurate information.

4.5 User Feedback

Feedback gathered during User Acceptance Testing (UAT) emphasized the ParkMate platform's user-friendliness and effectiveness in addressing urban parking challenges. Key suggestions for improvement included:

- **Mobile App Development**: Users expressed a strong interest in a dedicated mobile application to facilitate easier and more convenient access to parking services on the go.
- **Advanced Search Filters:** Requests for additional search filters emerged, enabling users to specify parking preferences, such as covered parking spaces or locations with electric vehicle charging stations.
- **Enhanced Notifications:** Users desired more comprehensive notifications regarding reservation statuses and promotional offers, helping them stay informed and engaged with the platform.

CHAPTER 5

DISCUSSION AND ANALYSIS

5.1 Additional Challenges Faced

Apart from the primary challenges mentioned earlier, the ParkMate platform faced several other complexities during its development:

- **User Experience (UX) Design:** Creating an intuitive and user-friendly interface required extensive user research and multiple iterations. Striking the right balance between simplicity and functionality posed a challenge in ensuring that both tech-savvy users and less experienced individuals could easily navigate the platform.

- **Integration with Payment Gateways:** Facilitating smooth transactions through third-party payment gateways presented its own set of challenges. Ensuring compatibility with multiple payment providers while maintaining a seamless user experience demanded extensive testing and troubleshooting.

- **Dynamic Pricing Implementation:** Incorporating a dynamic pricing feature, which allows parking rates to fluctuate based on demand, added a layer of complexity. The system needed to efficiently manage these pricing algorithms without impacting performance or confusing users.

- **Cross-Platform Compatibility:** Ensuring that the platform functioned consistently across different devices and operating systems, including mobile phones, tablets, and desktops, was challenging. It required thorough testing and optimization for various screen sizes and performance constraints.

- **Customer Support Management:** Developing an integrated customer support system within the platform to address user issues in real-time was crucial. The challenge was to make this system responsive, ensuring that users could get immediate assistance, especially in urgent parking situations.

5.2 Additional Solutions Implemented

To address the additional challenges encountered, the following solutions were introduced:

- **User-Centered Design Approach:** A user-centered design process was adopted to improve the overall user experience. This included frequent user feedback sessions, usability testing, and continuous refinement of the interface to ensure a smooth and enjoyable experience for all users.

- **Payment Gateway Optimization:** A standardized API integration framework was developed for handling multiple payment gateways. This framework enabled smooth transaction processing and ensured that the platform could easily scale to support new payment providers in the future.

- **Automated Pricing Algorithms:** To handle dynamic pricing, automated pricing algorithms were implemented, which adjusted rates based on real-time data such as parking demand and availability. These algorithms were thoroughly tested to ensure they were accurate, fair, and efficient without causing delays in the platform's performance.

- **Cross-Platform Testing Framework:** A comprehensive cross-platform testing framework was developed to ensure that the ParkMate platform functioned seamlessly across various devices and operating systems. Regular testing cycles were performed to identify and resolve compatibility issues quickly.

- **Integrated Help Desk and Chatbot:** To improve customer support, an integrated help desk system with an AI-powered chatbot was implemented. This system provided users with real-time assistance and automated responses for frequently asked questions, reducing the time required for human intervention.

5.3 Lessons Learned

- **Importance of Planning**: Comprehensive planning and requirement analysis are vital for identifying potential challenges early in the development process, facilitating smoother execution and minimizing risks.
- **User-Centric Design**: Focusing on user experience is essential for achieving higher satisfaction and adoption rates. Engaging with users throughout the development cycle provides valuable insights for creating a more intuitive platform.
- **Continuous Testing**: Implementing regular and thorough testing at every stage of development ensures timely identification and resolution of issues, contributing to overall system reliability and performance.
- **Adaptability**: Maintaining flexibility and openness to feedback fosters continuous improvement, enabling the team to enhance the platform based on user needs and market trends.

5.4 Future Improvements

Based on the project's current state and user feedback, the following improvements are proposed:

- **Mobile Application Development**: Creating native mobile apps for Android and iOS to provide users with on-the-go access.AI-Based Parking Predictions: Incorporating machine learning algorithms to predict parking availability based on historical data and trends.
- **Enhanced Payment Options**: Adding support for more payment gateways and cryptocurrency transactions.
- **Multi-Language Support:** Expanding the platform's language options to cater to a broader user base.
- **Integration with Smart City Initiatives**: Collaborating with city planners to integrate ParkMate with smart city infrastructure for optimized urban parking management.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

The ParkMate platform successfully addresses the common issue of finding available parking in urban areas by connecting vehicle owners with landowners who have unused parking spaces. Through the integration of real-time location-based services, secure transaction mechanisms, and an intuitive user interface, ParkMate delivers a scalable and efficient solution for both parties. This project showcases the effective application of modern web technologies, resulting in a system that positively impacts urban parking management by streamlining the process and enhancing convenience for users

6.2 Future Works

To further enhance and evolve the ParkMate platform, the following future improvements are suggested:

- **Mobile Application Development:** Developing native mobile applications for iOS and Android to enhance user accessibility, increase engagement, and leverage mobile-specific features like push notifications and location tracking.

- **AI-Driven Smart Parking Suggestions:** Integrating artificial intelligence algorithms to offer smart parking suggestions based on user preferences, traffic patterns, and historical data, thereby improving the overall parking experience.

- **Electric Vehicle (EV) Charging Integration:** Expanding the platform to include EV charging stations, allowing users to locate parking spots equipped with charging facilities, catering to the growing population of electric vehicle owners.

- **Dynamic Pricing Model:** Introducing a dynamic pricing system that adjusts parking fees based on demand, time of day, and special events, optimizing the revenue potential for parking owners while providing flexible pricing for users.

- **Subscription-Based Services:** Implementing subscription-based options for frequent users, allowing them to reserve parking spaces at discounted rates for a defined period.

- **Integration with Smart City Initiatives:** Collaborating with smart city projects to integrate the ParkMate platform with other urban mobility systems, such as public transportation, bike-sharing, and carpooling services, contributing to a more holistic urban transportation ecosystem.

- **Parking Space Optimization:** Using data analytics to optimize parking space usage and guide landowners on how to maximize their revenue, offering insights on peak usage times and preferred spots.

- **Carbon Footprint Reduction Initiatives:** Implementing features that help users track and reduce their carbon footprint by identifying eco-friendly parking spaces and encouraging greener transportation choices, such as parking spots that support carpooling.

REFERENCES

[1] K. Kowalczyk, "HTLM Documentation," in *Essential HTML5 Canvas*, Programming Books, 2023.

[2] R. Nixon, "Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5," in *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5, 4th ed*, O'Reilly Media, 2015, p. 750.

[3] Tailwindcomponents.com, "A free repository for community," [Online]. Available: https://www.creative-tim.com/twcomponents/.

[4] F. Zandstra, "Laravel: Up and Running," in *Laravel: Up and Running, 2nd ed*, O'Reilly Media, 2019, p. 370.

[5] N. Smyth, "MySql Manual," in *MySQL Essentials*, Payload Media, 2007, p. 254.

[6] Git-Scm.com, "Git Documentation," [Online]. Available: https://git-scm.com/doc.

[7] code.visualstudio, "Visual Studio Code Documentation," [Online]. Available: https://code.visualstudio.com/docs.

[8] M. Nixon, "JavaScript: The Definitive Guide," in *JavaScript: The Definitive Guide 7th ed*, O'Reilly Media, 2020, p. 706.

[9] C. Schmitt, "CSS: The Definitive Guide," in *CSS: The Definitive Guide*, O'Reilly Media, 2017, p. 900.

[10] K. Williams, "Learning PHP, MySQL & JavaScript," in *Learning PHP, MySQL & JavaScript 5th Ed*, O'Reilly Media, 2018, p. 824.

[11] D. Rees, "Laravel Installation," in *Code Bright - Web application development for the Laravel framework for beginners.*, Leanpub.com, 2013, p. 260.

[12] PHPUnit, "Documentation for PHPUnit," [Online]. Available: https://phpunit.de/documentation.html.

[13] R. Lerdorf, "Programming PHP," in *Programming PHP 4th ed*, O'Reilly Media, 2020, p. 550.

[14] M. Trachtenberg, "Upgrading to PHP 7," in *Upgrading to PHP 7*, Addison-Wesley, 2016, p. 360.

[15] M. MacDonald, "Creating a Website: The Missing Manual," in *Creating a Website: The Missing Manual 4th ed*, O'Reilly Media, 2011, p. 800.

[16] M. Brown, "The Complete Reference MySQL," in *The Complete Reference MySQL, 2nd ed*, McGraw-Hill, 2010, p. 1050.

[17] A. JMeter, "JMeter User Manual," [Online]. Available: https://jmeter.apache.org/usermanual/index.html.

APPENDICES

Appendix A: System Architecture Diagram

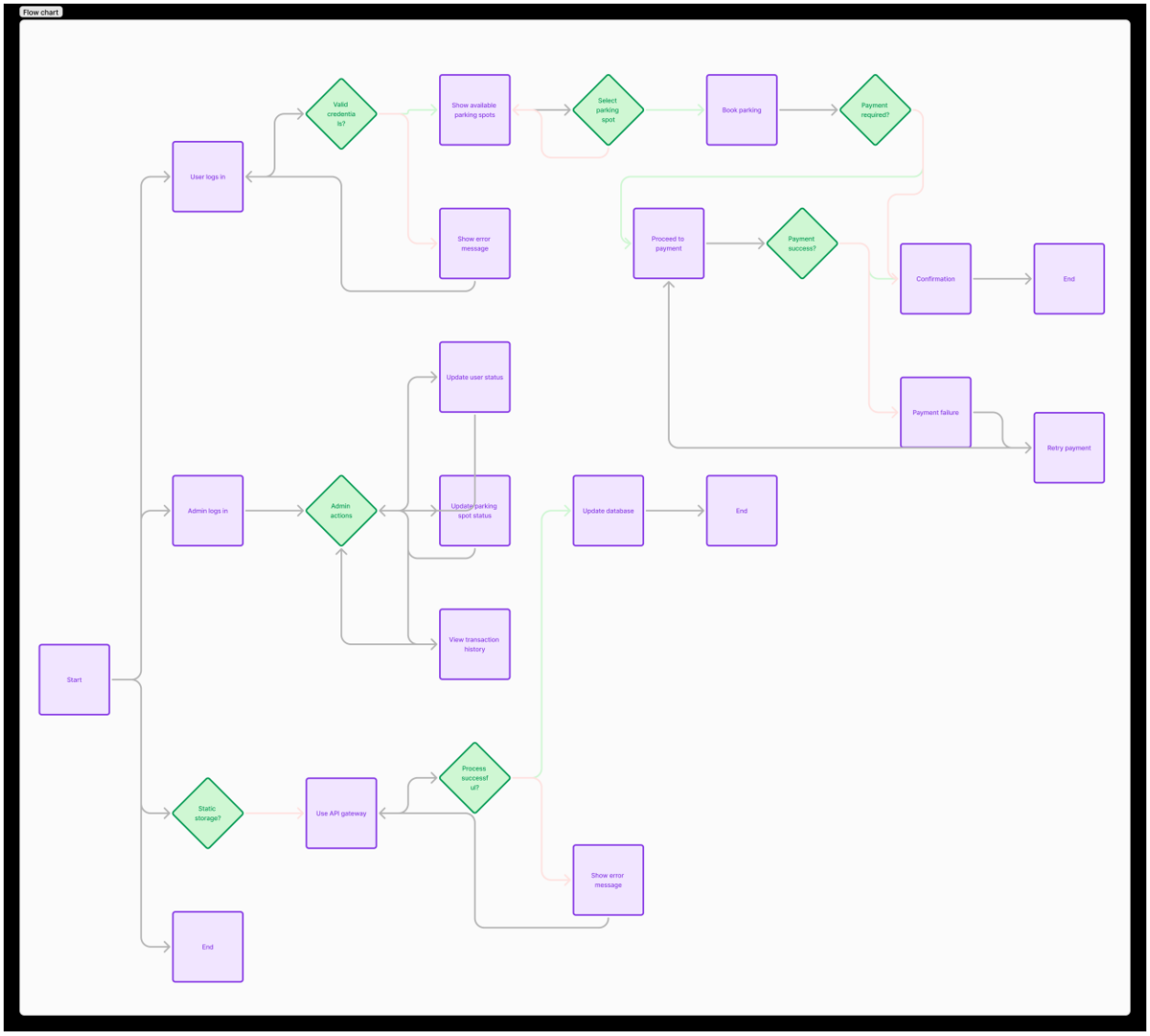


Fig: 1.1

Appendix B: Database Schema



Fig: 2.1

Appendix C: Sample Code Snippets

Included relevant code excerpts that demonstrate key functionalities, such as:

- User authentication logic

```
            <form action="{{ route('user.register.store') }}" class="sign-up-form" method="POST" enctype="multipart/form-data">
@csrf
                    <h2 class="title">Register</h2>
                    <div class="input-field">
    <i class="fas fa-user"></i>
    <input id="name" type="text" class="form-control  @error('name') is-invalid @enderror"
    placeholder="Name" name="name" value="{{ old('name') }}" required autocomplete="name">
    @error('name')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror
</div>
<div class="input-field">
    <i class="fas fa-phone"></i>
    <input type="phone" name="phone" placeholder="Phone Number" />
    @error('phone')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror
</div>
<div class="input-field">
    <i class="fas fa-envelope"></i>
    <input id="email" type="email"  placeholder="Email" class="form-control @error('email') is-invalid @enderror"
        name="email" value="{{ old('email') }}" required autocomplete="email">
    @error('email')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror
</div>
<div class="input-field align-items-center">
    <i class="fa-solid fa-id-card"></i>
    <input id="name" type="file" class="form-control" name="nid" value="{{ old('nid') }}" required>
    @error('nid')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror
</div>
<div class="input-field">
    <i class="fas fa-lock"></i>
    <input id="password" type="password" placeholder="Password" class="form-control @error('password') is-invalid @enderror"
        name="password" required autocomplete="new-password">
    @error('password')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror
</div>
<div class="input-field">
    <i class="fas fa-lock"></i>
    <input id="password-confirm" type="password" placeholder="Confirm Password" class="form-control" name="password_confirmation"
        required autocomplete="new-password">
</div>
<button type="submit" class="btn btn-primary">
    {{ __('Register') }}
</button>
            </form>
```

```
            </div>

            <div class="panels-container">
                    <div class="panel left-panel">
                            <div class="content">
                                    <h3>New here ?</h3>
                                    <p class="text-white">Lorem ipsum, dolor sit amet consectetur adipisicing elit. Debitis,
                                            ex ratione. Aliquid!</p>
                                    <button class="btn transparent" id="sign-up-btn">Register</button>
                            </div>
                            <img  src="./assets/img/login1.png" class="image" alt="" />
                    </div>
                    <div class="panel right-panel">
                            <div class="content">
                                    <h3>One of us ?</h3>
                                    <p class="text-white">rem ipsum dolor sit amet consectetur adipisicing elit. Nostrum
                                            laboriosam ad deleniti.</p>
                                    <button class="btn transparent" id="sign-in-btn">Login</button>
                            </div>
                            <img src="./assets/img/login2.png"  class="image" alt="" />
                    </div>
            </div>
    </div>
<!-- Toastr JS-->
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.js"></script>
<script>
    function flashMessage(status, message) {
        toastr.options = {
            "closeButton": true,
            "debug": false,
            "newestOnTop": false,
            "progressBar": true,
            "positionClass": "toast-top-right",
            "preventDuplicates": false,
            "onclick": null,
            "showDuration": "300",
            "hideDuration": "1000",
            "timeOut": "5000",
            "extendedTimeOut": "1000",
            "showEasing": "swing",
            "hideEasing": "linear",
            "showMethod": "fadeIn",
            "hideMethod": "fadeOut"
        }
        switch (status) {
            case 'success':
                toastr.success(message);
                break;

            case 'error':
                toastr.error(message);
                break;

            case 'info':
                toastr.info(message);
                break;

            case 'warning':
                toastr.warning(message);
                break;
        }
    }
    // session flash message
    @if (Session::get('success'))
        flashMessage('success', "{{ Session::get('success') }}")
    @elseif (Session::get('error'))
        flashMessage('error', "{{ Session::get('error') }}")
    @elseif (Session::get('info'))
        flashMessage('info', "{{ Session::get('info') }}")
    @elseif (Session::get('warning'))
        flashMessage('warning', "{{ Session::get('warning') }}")
    @endif
</script>
```

Fig: 3.1

- Parking search functionality

```
<div class="col-md-3">
    <div class="filter_option d-flex align-items-center justify-content-end">
        <h6> Search : </h6>
        <input type="text" id="search_city" placeholder="Search by City" onkeyup="searchCity()">
    </div>
</div>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
    function searchCity() {
        let query = $('#search_city').val();

        $.ajax({
            url: "{{ route('parking.search') }}", // A new route you'll define
            type: "GET",
            data: { 'city': query },
            success: function(data) {
                $('.row.clearfix.g-4').html(data); // Replace the parking result section with the new results
            }
        });
    }
</script>
```

Fig: 3.2

- Reservation handling processes

```
@section('content')
    <div class="container">
        <div class="col-12 col-md-12 py-4">
            <form action="{{ route('client.parking.book.store') }}" method="POST" enctype="multipart/form-data">
                @csrf
                <div class="summary entry-summary">
                    <input type="hidden" name="parking_id" value="{{ $singleParking->id }}">
                    <input type="hidden" name="create_user_id" value="{{ $singleParking->user_id }}">
                    <div>
                        <label>
                            <input type="radio" name="vehicle" value="bike" checked onclick="updatePrice('bike')"> Bike
                        </label>
                        <label>
                            <input type="radio" name="vehicle" value="car" onclick="updatePrice('car')"> Car
                        </label>
                        <label>
                            <input type="radio" name="vehicle" value="other" onclick="updatePrice('other')"> Other
                        </label>
                    </div>
                    <p class="total-price my-3">
                        <span class="parkmate-Price-amount amount">
                            <bdi>
                                <span class="parkmate-Price-currencySymbol">Total Price : $</span>
                                <span id="total-price-display">{{ $singleParking->price_bike }}</span>
                            </bdi>
                        </span>
                    </p>
                    <input type="hidden" name="price" id="price-input" value="{{ $singleParking->price_bike }}">

                    <div class="duration-input">
                        <label for="duration">Enter Duration (in hours):</label> <br>
                        <input type="number" class="form-control" name="duration" id="duration" value="1"
                            min="1" required onchange="updateTotalPrice()">
                    </div>
                    <div class="form-group mt-3">
                        <label for="cardnumber">Card Number</label>
                        <input id="cardnumber"  class="form-control" placeholder="Enter Your Card Number" type="text" pattern="[0-9]*" inputmode="numeric"
                            name="cardnumber">
                        @error('cardnumber')
                            <p class="text-danger">{{ $message }}</p>
                        @enderror
                    </div>
                    <div class="form-group mt-3">
                        <label for="expirationdate">Expiration (mm/yy)</label>
                        <input id="expirationdate" class="form-control" type="date" inputmode="numeric"
                            name="expirationdate">
                        @error('expirationdate')
                            <p class="text-danger">{{ $message }}</p>
                        @enderror
                    </div>
                    <div class="form-group my-3">
                        <label for="securitycode">Security Code</label>
                        <input id="securitycode" placeholder="Enter Your Security Code" class="form-control" type="text" inputmode="numeric"
                            name="securitycode">
                        @error('securitycode')
                            <p class="text-danger">{{ $message }}</p>
                        @enderror
                    </div>
                    <input type="hidden" name="total_price" id="total-price-input"
                        value="{{ $singleParking->price_bike }}">
                </div>
                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div>
    </div>
@endsection
```

```
<script>
    function updatePrice(vehicle) {
        let priceBike = {{ $singleParking->price_bike }};
        let priceCar = {{ $singleParking->price_car }};
        let priceOther = {{ $singleParking->price_others }};

        let price = 0;
        if (vehicle === 'bike') {
            price = priceBike;
        } else if (vehicle === 'car') {
            price = priceCar;
        } else {
            price = priceOther;
        }

        // Set the price
        $('#price-display').text(price);
        $('#price-input').val(price);
        updateTotalPrice();
    }

    function updateTotalPrice() {
        let price = parseFloat($('#price-input').val()) || 0;
        let duration = parseFloat($('#duration').val()) || 0;

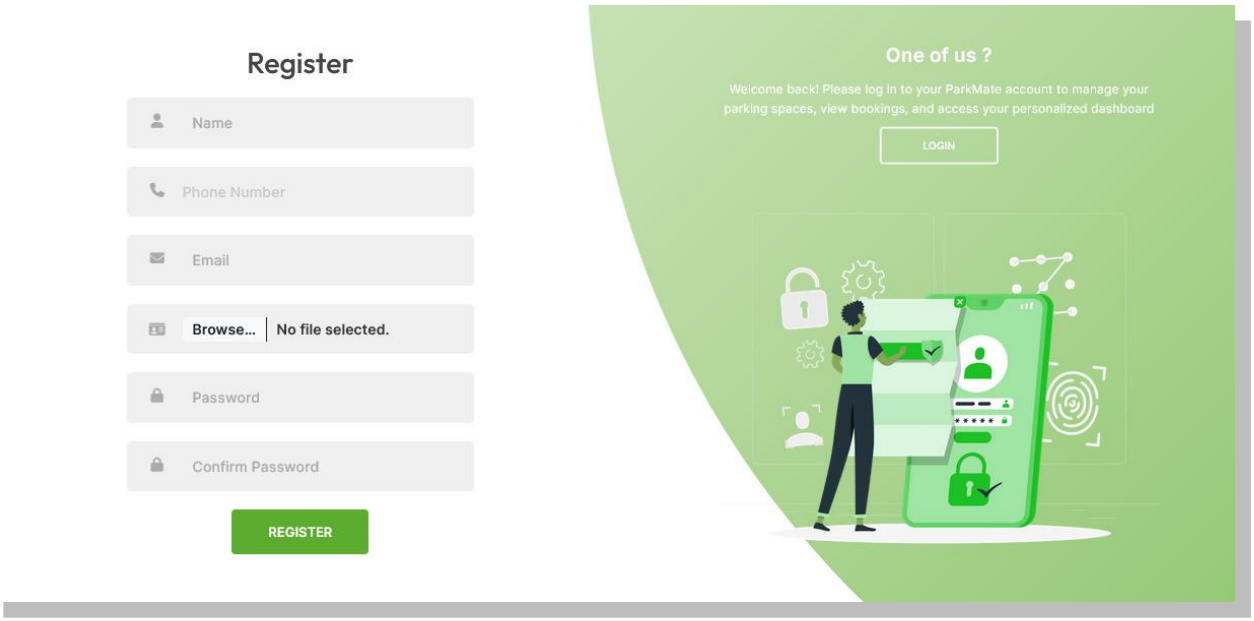
        let totalPrice = price * duration;

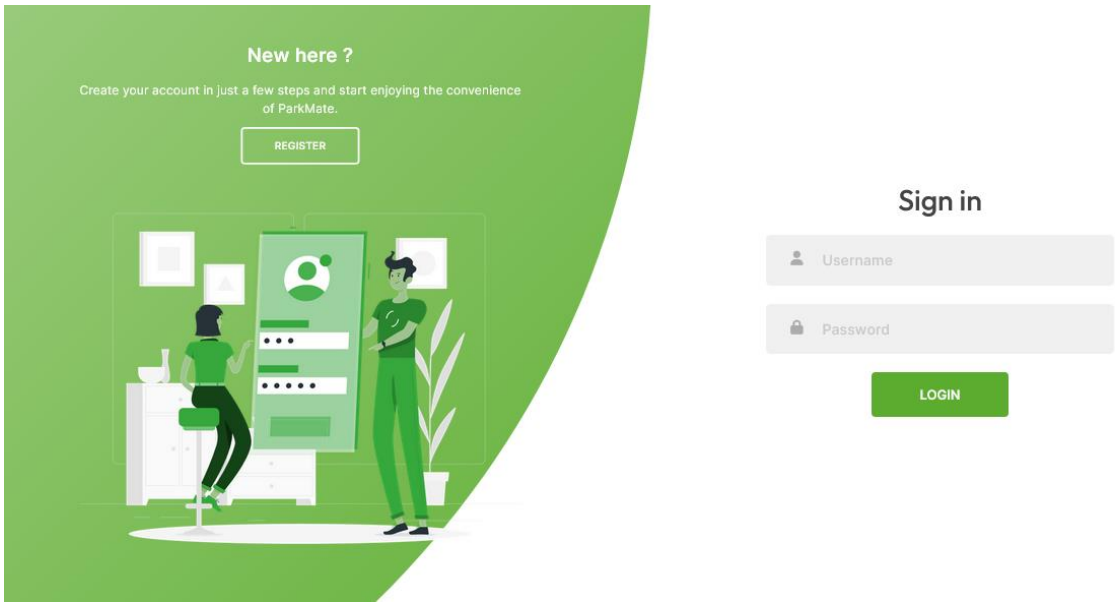
        $('#total-price-display').text(totalPrice.toFixed(2));
        $('#total-price-input').val(totalPrice.toFixed(2));
    }
</script>
```

Fig: 3.3

Appendix D: User Interface Screenshots

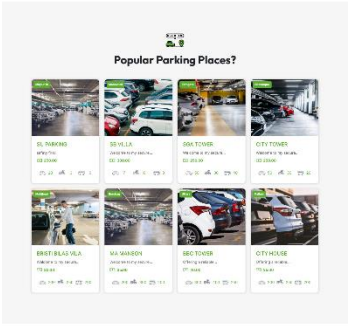
- **Frontend:**





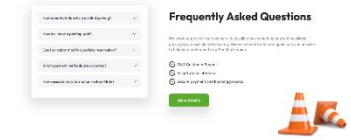

Parking Zone
Easy Way To Find
Near Parking Zone.

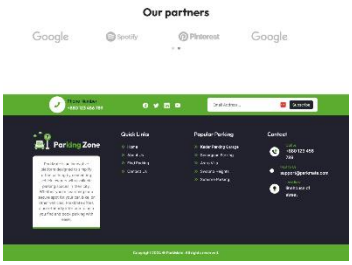

What Advantages will You Get Through ParkMate?
Convenient Parking Search
Flexible Parking Options
Secure Booking Process
Cashless Payments


Popular Parking Places?
SL PARKING
SS VILLA
SSK TOWER
CITY TOWER
BRIST BLAS VILA
MA MARRON
SBC TOWER
CITY HOUSE

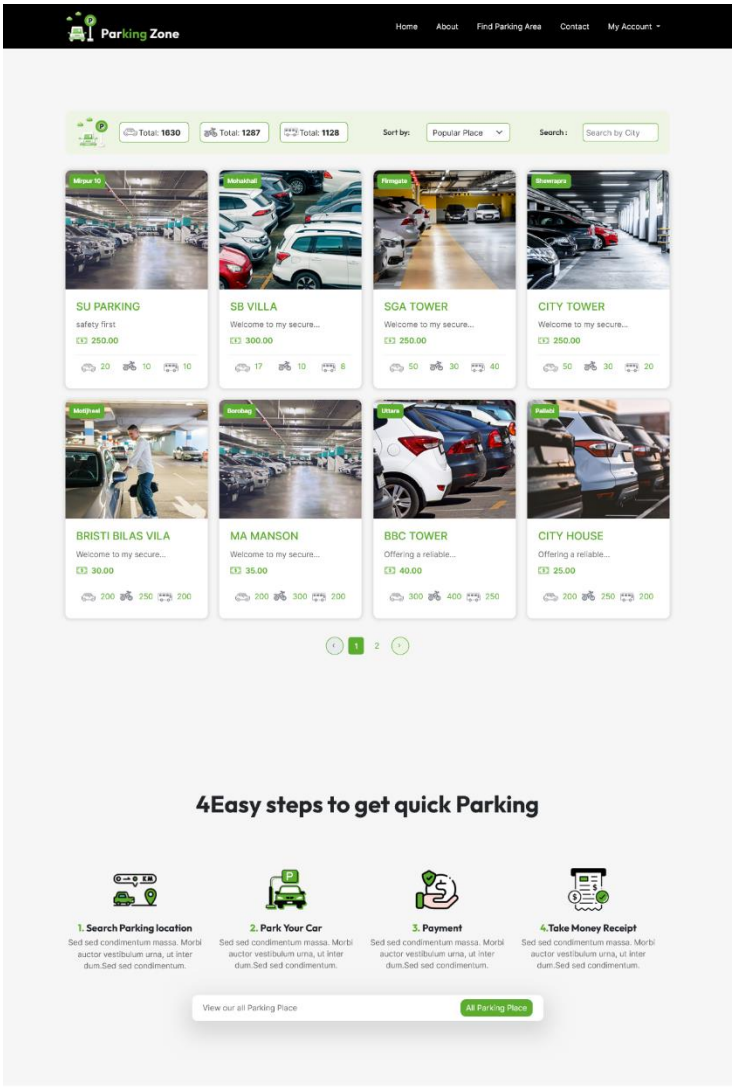

Maximize Your Parking Points
21K
21K
21°
21+


4 Easy Steps to Quick Parking


Frequently Asked Questions


Our partners
Google
Spotify
Pinterest
Google

Parking Zone
Quick Links
Popular Parking
Contact


Parking Zone
Home
About
Find Parking Area
Contact
My Account
Total: 1630
Total: 1287
Total: 1128
Sort by:
Popular Place
Search:
Search by City
SU PARKING
safety first
250.00
SB VILLA
Welcome to my secure...
300.00
SGA TOWER
Welcome to my secure...
250.00
CITY TOWER
Welcome to my secure...
250.00
BRISTI BILAS VILA
Welcome to my secure...
30.00
MA MANSON
Welcome to my secure...
35.00
BBC TOWER
Offering a reliable...
40.00
CITY HOUSE
Offering a reliable...
25.00
4Easy steps to get quick Parking
1. Search Parking location
Sed sed condimentum massa. Morbi auctor vestibulum urna, ut inter dum.Sed sed condimentum.
2. Park Your Car
Sed sed condimentum massa. Morbi auctor vestibulum urna, ut inter dum.Sed sed condimentum.
3. Payment
Sed sed condimentum massa. Morbi auctor vestibulum urna, ut inter dum.Sed sed condimentum.
4.Take Money Receipt
Sed sed condimentum massa. Morbi auctor vestibulum urna, ut inter dum.Sed sed condimentum.
View our all Parking Place
All Parking Place
Our partners
Google
Spotify
Pinterest
Google
Phone Number
+880 123 456 789
Email Address...
Subscribe
Parking Zone
ParkMate is an innovative platform designed to simplify urban parking by connecting vehicle owners with available parking spaces in their city. Whether you're searching for a secure spot for your car, bike, or other vehicles, ParkMate offers a user-friendly interface to help you find and book parking with ease.
Quick Links
Home
About Us
Find Parking
Contact Us
Popular Parking
Kader Parking Garage
Sonargoan Parking
Astro Villa
Swapno Heights
Suborno Parking
Contact
Call us
+880 123 456 789
Mail to Us
support@parkmate.com
Location
linehouse of street
Copyright 2024 © ParkMate. All rights reserved.

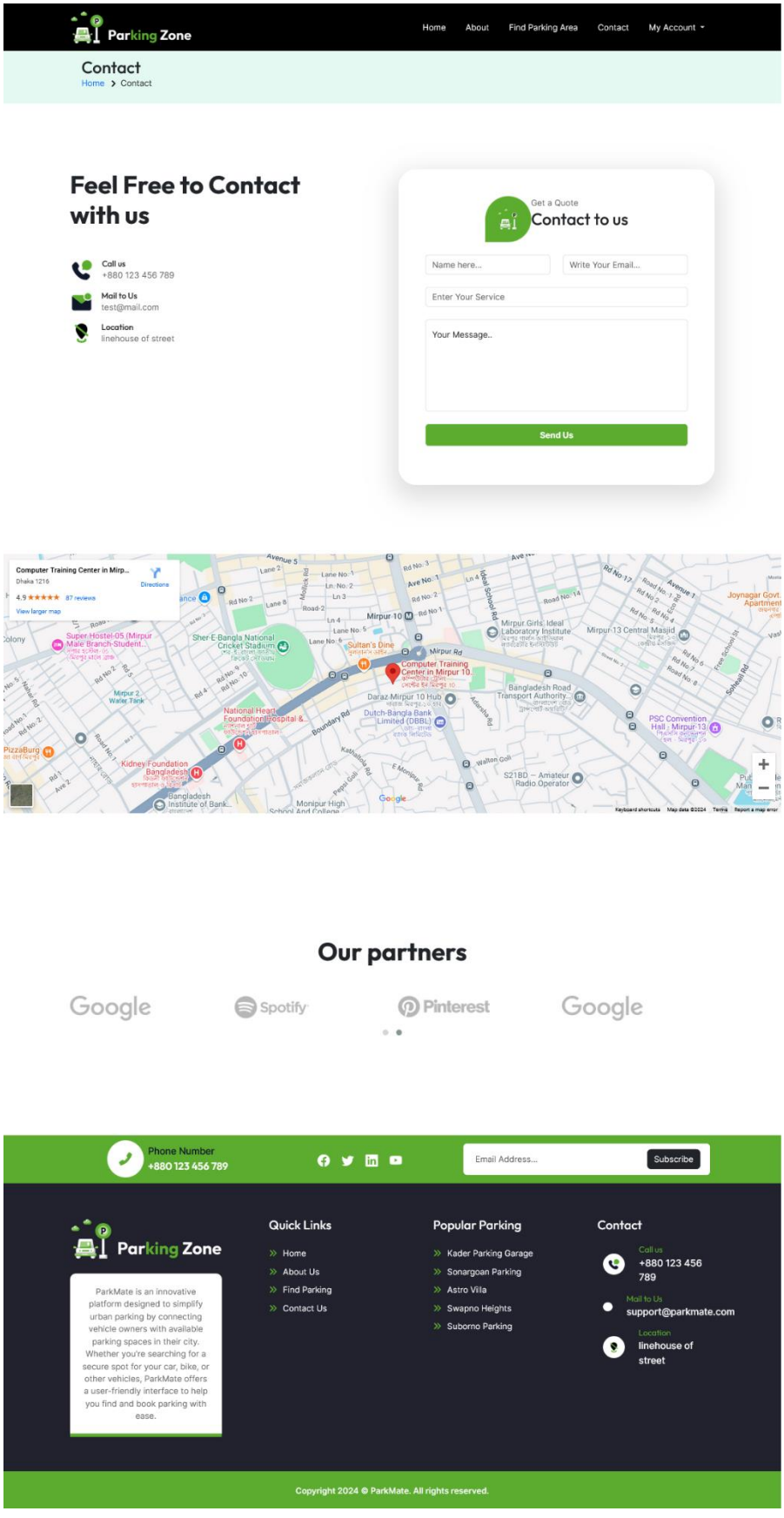
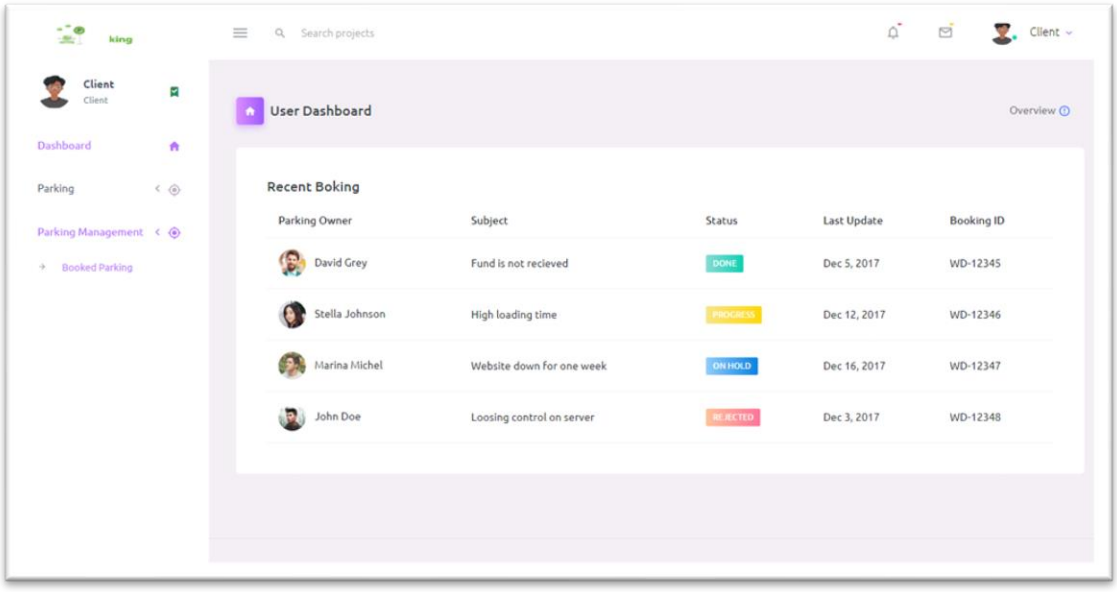
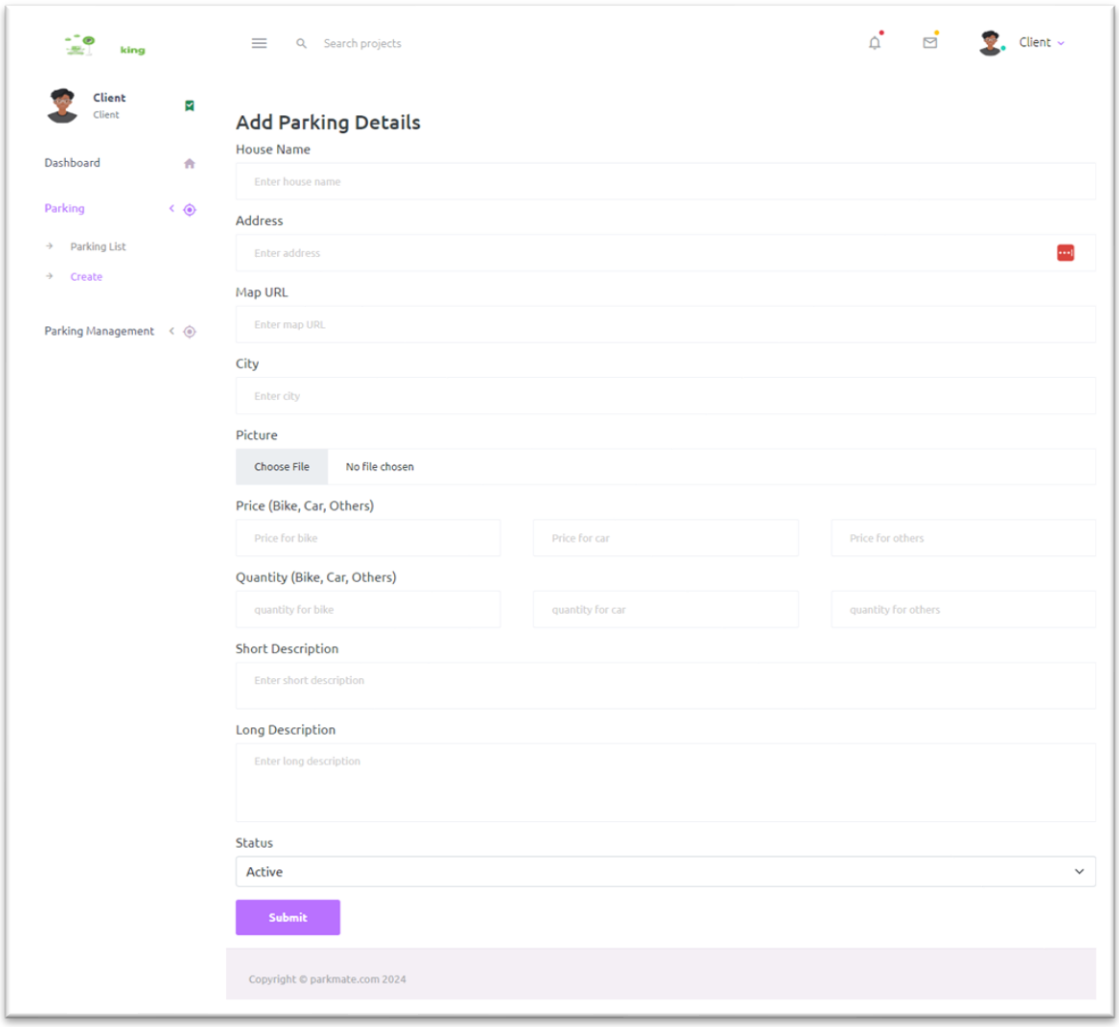

Parking Zone
Home
About
Find Parking Area
Contact
My Account
Contact
Home > Contact
Feel Free to Contact with us
Call us
+880 123 456 789
Mail to Us
test@mail.com
Location
linehouse of street
Get a Quote
Contact to us
Name here...
Write Your Email...
Enter Your Service
Your Message..
Send Us
Our partners
Google
Spotify
Pinterest
Google
Phone Number
+880 123 456 789
Email Address...
Subscribe
Parking Zone
ParkMate is an innovative platform designed to simplify urban parking by connecting vehicle owners with available parking spaces in their city. Whether you're searching for a secure spot for your car, bike, or other vehicles, ParkMate offers a user-friendly interface to help you find and book parking with ease.
Quick Links
Home
About Us
Find Parking
Contact Us
Popular Parking
Kader Parking Garage
Sonargoan Parking
Astro Villa
Swapno Heights
Suborno Parking
Contact
Call us
+880 123 456 789
Mail to Us
support@parkmate.com
Location
linehouse of street
Copyright 2024 © ParkMate. All rights reserved.

Fig: 4.1

- **Backend (Client View):**





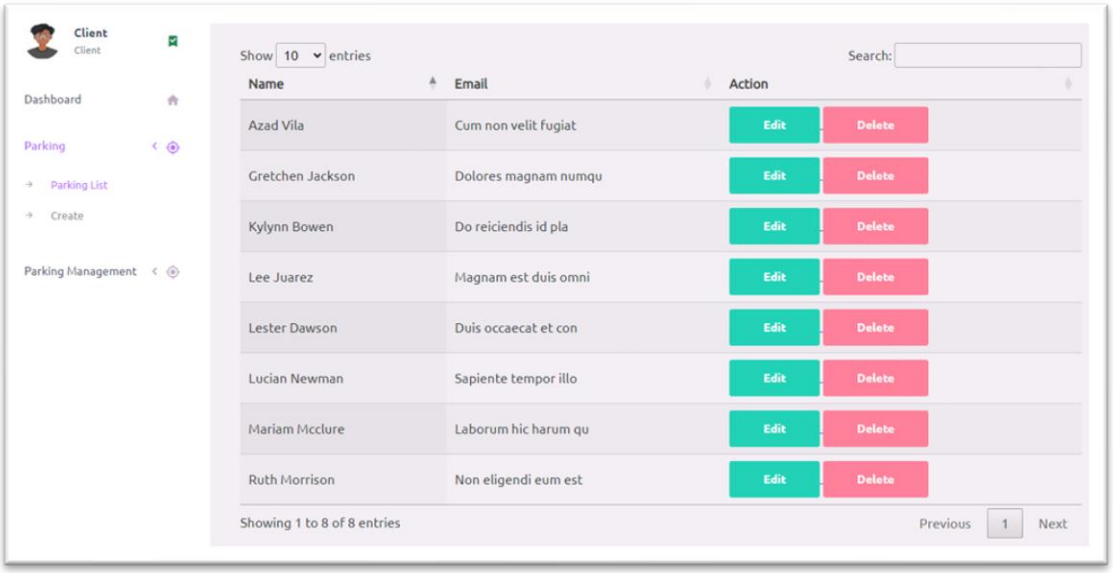
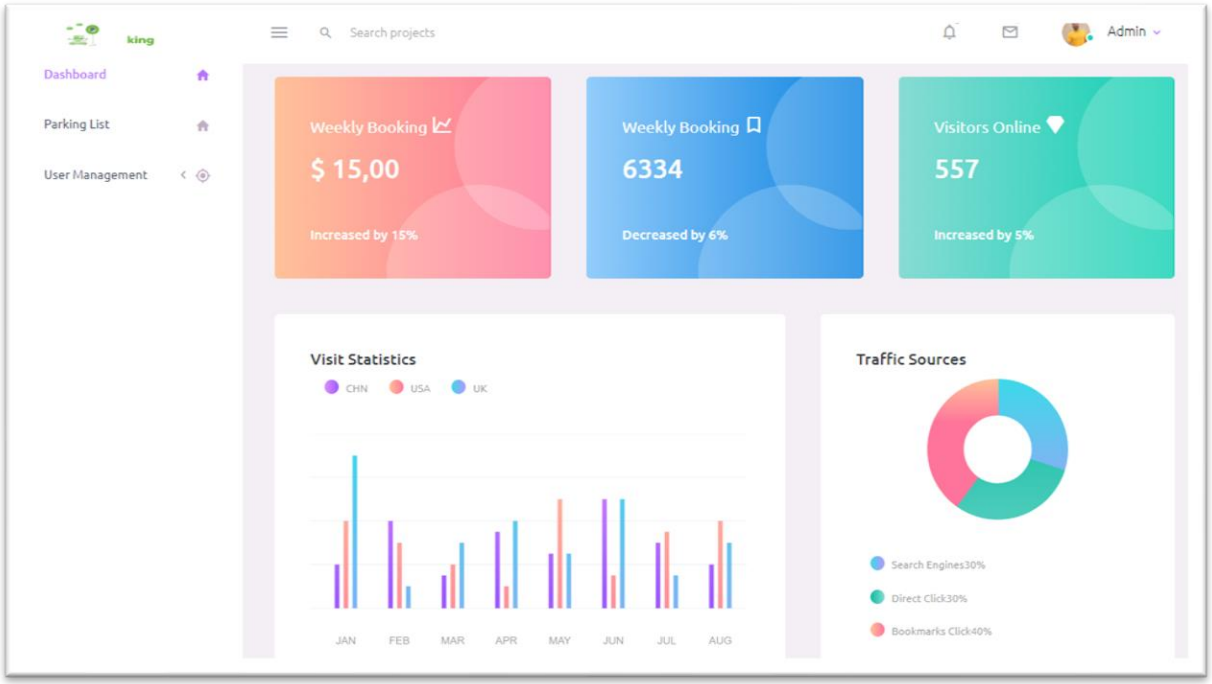


Fig: 4.2

- **Backend (Admin View):**



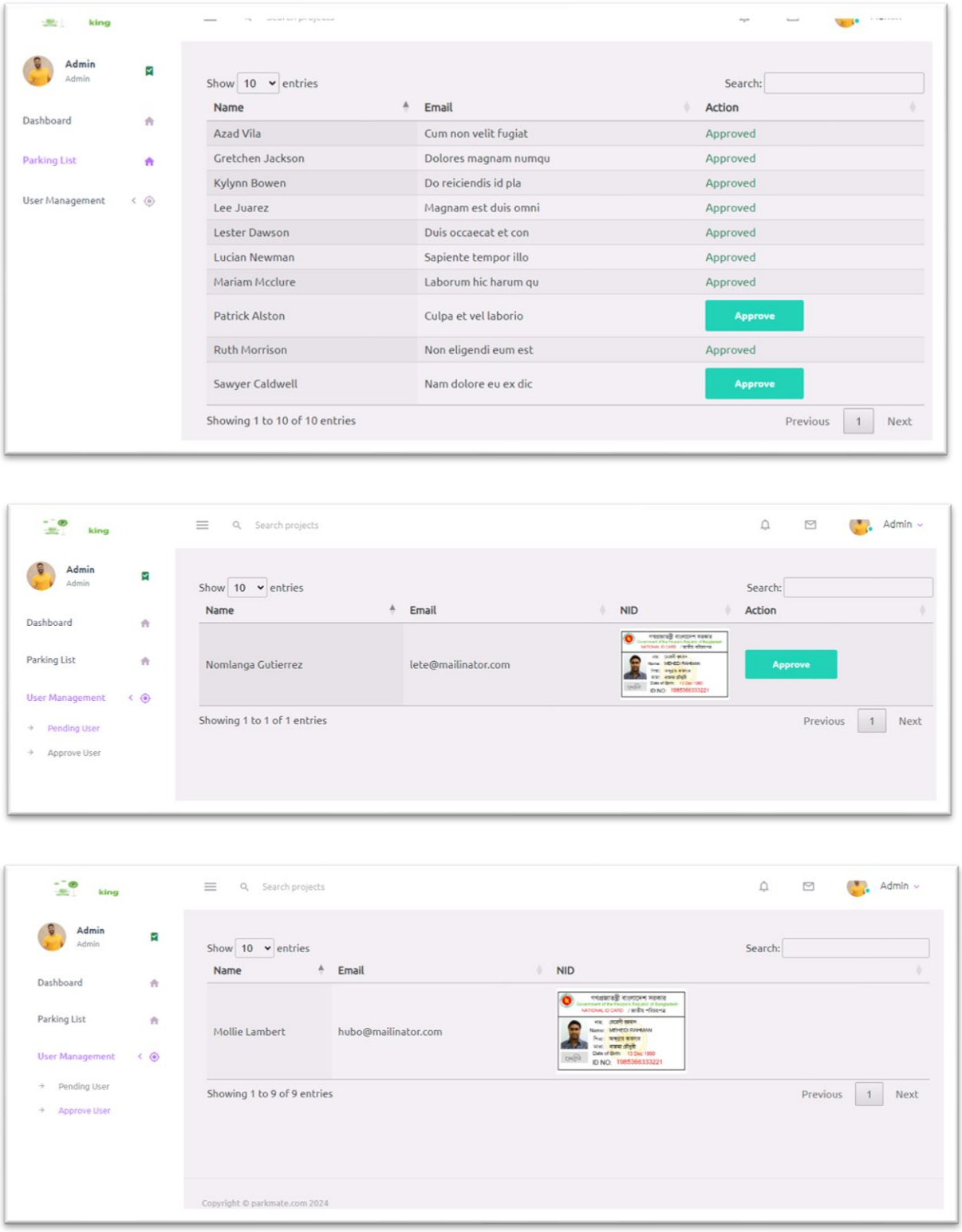

king
Admin
Admin
Dashboard
Parking List
User Management
Show 10 entries
Search:
Name
Email
Action
Azad Vila
Cum non velit fugiat
Approved
Gretchen Jackson
Dolores magnam numqu
Approved
Kylynn Bowen
Do reiciendis id pla
Approved
Lee Juarez
Magnam est duis omni
Approved
Lester Dawson
Duis occaecat et con
Approved
Lucian Newman
Sapiente tempor illo
Approved
Mariam Mcclure
Laborum hic harum qu
Approved
Patrick Alston
Culpa et vel laborio
Approve
Ruth Morrison
Non eligendi eum est
Approved
Sawyer Caldwell
Nam dolore eu ex dic
Approve
Showing 1 to 10 of 10 entries
Previous
1
Next
Search projects
Admin
NID
Nomlanga Gutierrez
lete@mailinator.com
Showing 1 to 1 of 1 entries
Pending User
Approve User
Mollie Lambert
hubo@mailinator.com
Showing 1 to 9 of 9 entries
Copyright © parkmate.com 2024

Fig: 4.3

Final Steps for Completion

- **Complete Appendices**: Finalize all appendices by adding the required system architecture diagrams, database schemas, code snippets, and user interface screenshots to provide thorough visual and technical documentation.
- **Expand and Refine Content**: Enrich each section with more in-depth analysis, real-world examples, and relevant data to enhance the clarity and depth of your discussion.
- **Thorough Proofreading**: Conduct a meticulous proofread to eliminate grammatical errors, improve sentence structure, and ensure the flow and tone of the document are consistent and professional.
- **Formatting and Styling**: Ensure the document meets all university formatting standards, including correct use of headings, consistent font and size, appropriate margins, and proper layout of sections and subsections.
- **Verify Citations and References**: Cross-check all citations to ensure proper academic standards are followed, and confirm that every reference is accurately formatted according to the required citation style.
- **Seek Feedback and Revise**: Solicit feedback from peers, mentors, or supervisors to identify areas for improvement. Incorporate suggestions and revisions that enhance the clarity and quality of the work.
- **Final Review and Submission Readiness**: Perform a comprehensive final review to confirm that the paper is complete, well-organized, error-free, and ready for submission. Double-check for any overlooked details or inconsistencies.