VEHICLE PARKING MANAGEMENT SYSTEM - PHASE 1 REPORT

DELIVERABLE-3

The Parking Management System is a web-based application designed to efficiently manage parking slots. It allows users to book parking slots, view availability, and manage bookings. Administrators have functionalities to oversee users, parking slots, and reservations. The system is built using Node.js, Express.js, MongoDB, and React.js. We made significant changes by updating technology stack to improve performance, scalability, and user experience. Initially, the system was designed with a traditional backend structure using PhP laravel, but based on peer feedback and performance evaluations, we have changed it to a Node.js and Express.js backend with MongoDB. This change was primarily made to improve scalability, ensure better API handling, and also to improve database flexibility compared to a relational database structure.

For the frontend, we opted for React.js with Material UI and Tailwind CSS instead of a basic HTML/CSS or outdated frontend framework. This shift was made as it's needed for a modern, responsive user interface with reusable components and faster rendering. The combination of Material UI and Tailwind CSS ensures a good design while maintaining customization flexibility. Moreover, we have also introduced pagination for parking slots and bookings to improve performance when handling large datasets as it prevents excessive data loading, improving response times and user experience. Enhanced error handling and validation mechanisms were also implemented in the backend using Express Validator and better middleware practices to ensure strong security and data integrity. These updates were made to make sure that system is more updated to improve efficiency, and provide a smoother user experience while maintaining security and maintainability.

a. Requirements

Phase 1 Requirements (From Deliverable 2)

User Authentication

- The system provides a secure authentication mechanism allowing users to register, log in, and log out.
- JWT (JSON Web Token)-based authentication is implemented to ensure secure access and prevent unauthorized actions.
- Role-based access control (RBAC) is enforced, where users are assigned one of the following roles:
    - **Admin**: Full control over system operations.
    - **Moderator**: Limited administrative privileges, such as managing parking slots and bookings.
    - **User**: Can book parking slots and manage their own reservations.

Parking Slot Management

- **Admins have complete control over parking slots and can perform the following actions:**

- Create new parking slots with attributes such as slot number, location, size, and availability status.
- Update existing parking slots, including modifying their status, type, or location.
- Delete parking slots that are no longer needed.

- **Users can:**
    - View the list of available parking slots.
    - Filter parking slots based on criteria like location, type, and availability.
    - Book available parking slots.
- **Admins and Moderators can:**
    - Update parking slot statuses to Available, Occupied, or Reserved based on real-time usage.

Booking Management:

- **Users can:**
    - Create new bookings for available parking slots by selecting a slot and specifying the duration.
    - View their current and past bookings.
    - Cancel their bookings before the scheduled time.
- **Admins and Moderators** can:
    - View all bookings across the system.
    - Manage booking statuses, which include:
        - **Pending**: Awaiting confirmation.
        - **Confirmed**: Successfully booked.
        - **Cancelled**: Canceled by the user or admin.

User Management

- **Admins** have full control over user accounts and can:
    - View all registered users.
    - Update user details such as **name, email, and role assignments**.
    - Delete users who violate platform policies or are inactive.
    - Assign or modify user roles (User, Moderator, Admin) as needed.

API Endpoints

- A RESTful API is provided for seamless communication between the frontend and backend, enabling efficient data exchange.

- API endpoints include:
  - **User Authentication**: Sign-up, login, logout.
  - **Parking Slot Management**: Creating, updating, deleting, and retrieving parking slots.
  - **Booking Management**: Creating, viewing, modifying, and canceling bookings.
  - **User Management**: Retrieving user lists, updating user roles, and managing accounts.
- The API follows best practices for security and scalability, including proper error handling and validation.

Security Measures

- CORS (Cross-Origin Resource Sharing) Middleware is used to manage and restrict cross-origin requests, ensuring secure communication between frontend and backend.
- Secure cookie settings for authentication tokens prevent unauthorized session hijacking.
- Input validation using express-validator ensures that only correctly formatted and safe data is processed by the system, protecting against common security vulnerabilities like SQL injection and XSS attacks.

Database and Data Management

- MongoDB is used as the primary database for storing user, parking slot, and booking information.
- The database schema includes:
  - **User Model**: Stores user details, authentication credentials (hashed passwords), and assigned roles.
  - **Parking Slot Model**: Stores slot details, including location, availability status, and slot type.
  - **Booking Model:** Stores booking details such as user ID, parking slot ID, booking time, and status.
- **Database relationships:**
  - Each user can have multiple bookings, establishing a one-to-many relationship.
  - Each parking slot can be associated with multiple bookings over time.
- Data validation is created at the database level to maintain consistency and integrity.

Changes to Requirements

Reason for Changes:

- Feedback from the peer review highlighted the need for better error handling and validation in the backend.
- Additional requirements were added to improve user experience, such as pagination for parking slots and bookings.

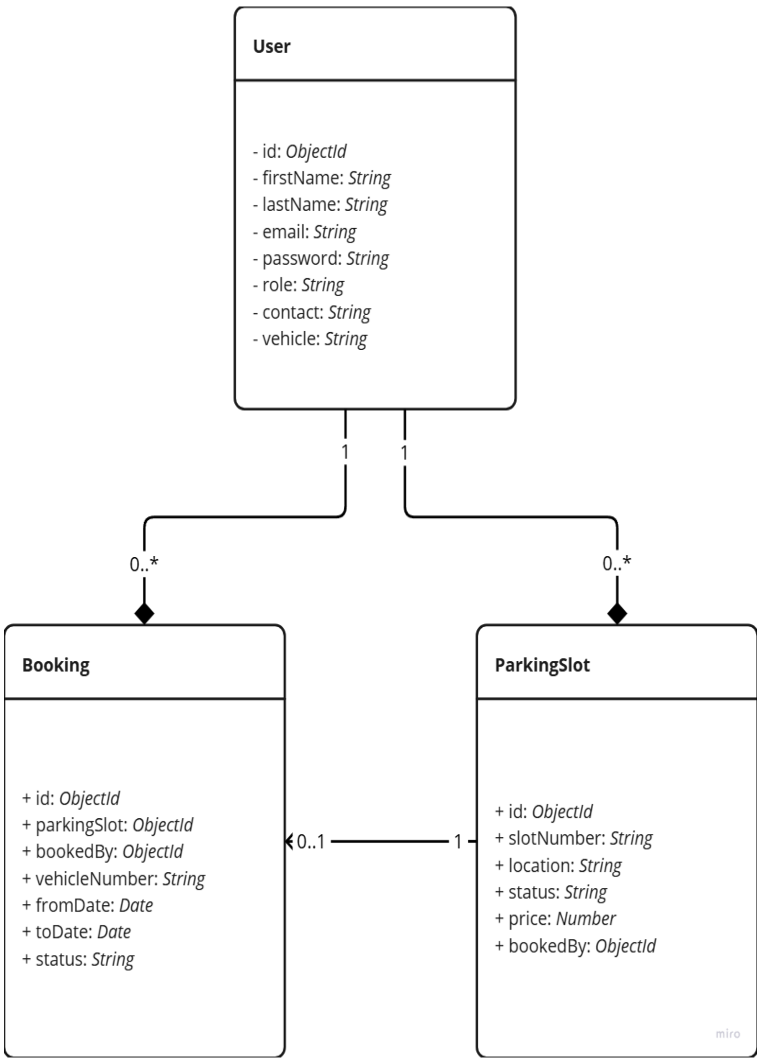
Updated Requirements:

- Added pagination for parking slots and bookings.
- Enhanced error handling and validation in the backend.
- Improved user interface for better usability.

---

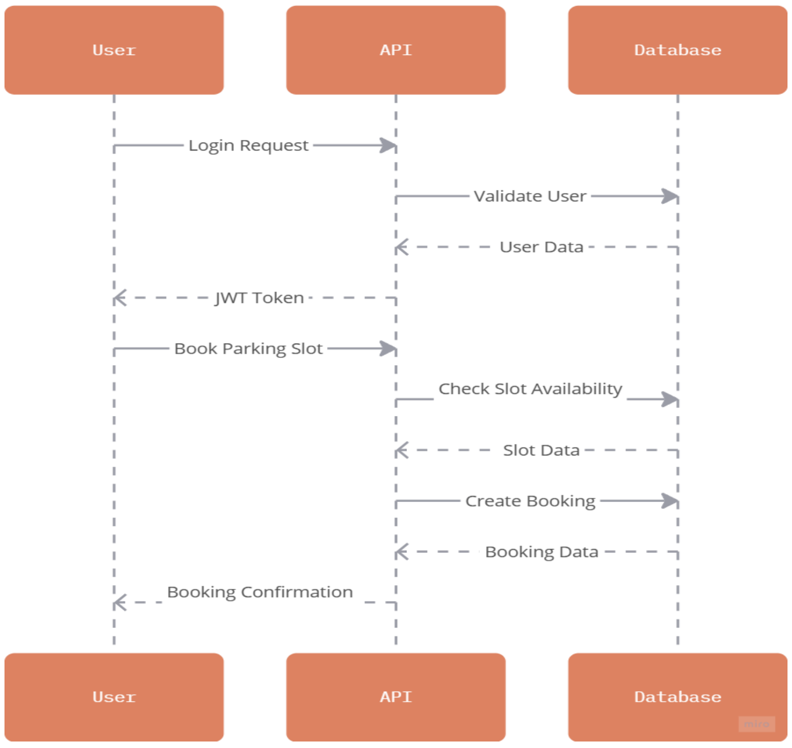
b. UML Design for Phase 1

Class Diagram

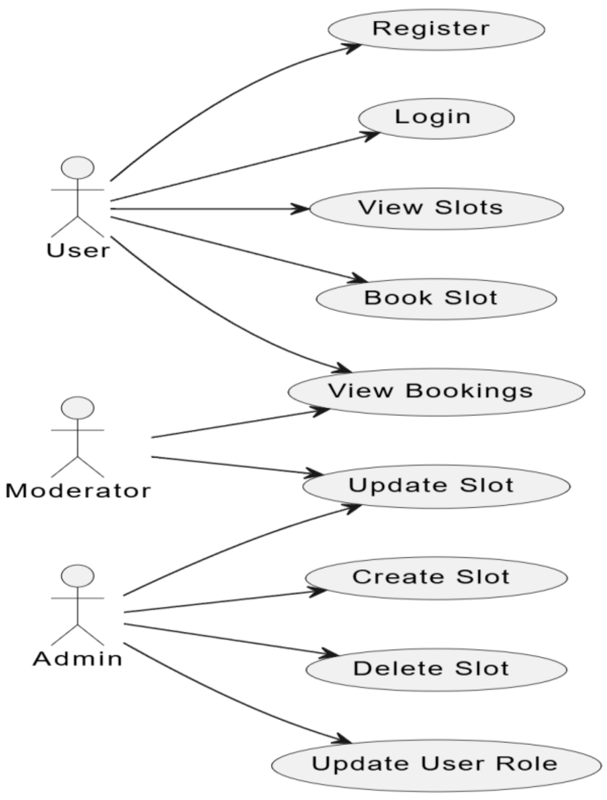


Sequence Diagram

create a sequence diagram for User -> API: Login Request API -> Database: Validate User Database -> API: User Data API -> User: JWT Token User -> API: Book Parking Slot API -> Database: Check Slot Availability Database -> API: Slot Data API -> Database: Create Booking Database -> API: Booking Data API -> User: Booking Confirmation



Use Case Diagram



Normal Case:

Use Case: User books a parking slot.

Steps:

1. User logs in.
2. User views available slots.
3. User selects a slot and books it.
4. System confirms the booking.

Error Case:

Use Case: User tries to book an already booked slot.

Steps:

1. User logs in.
2. User views available slots.
3. User selects a slot that is already booked.
4. System shows an error message.

---

c. Test Cases (Unit Tests)

Test Cases for Phase 1

User Registration:

- **Functionality:** Test user registration.
- **Input:**{ firstName: "John", lastName: "Doe", email: "john@example.com", password: "password123",confirmPassword:”password123” }
- **Expected Output:**{ success: true, user: { ... }, token: "..." }

User Login:

- **Functionality:** Test user login.
- **Input:**{ email: "john@example.com", password: "password123" }
- **Expected Output:**{ success: true, user: { ... }, token: "..." }

Create Parking Slot:

- **Functionality:** Test creating a parking slot (Admin only).
- **Input:**{ slotNumber: "A1", location: "Main Lot", price: 10 }
- **Expected Output:**{ success: true, slot: { ... } }

Book Parking Slot:

- **Functionality:** Test booking a parking slot.
- **Input:**{ parkingSlot: "A1", vehicleNumber: "XYZ123", fromDate: "2023-10-01", toDate: "2023-10-02" }
- **Expected Output:**{ success: true, booking: { ... } }

View Bookings:

- **Functionality:** Test viewing bookings.
- **Input:**{ userId: "123" }
- **Expected Output:**{ success: true, bookings: [...] }

---

d. User Manual

Installation & Usage Instructions

Installation:

1. Clone the repository:

   ```
   git clone https://github.com/your-username/parking-management-system.git
   ```

2. Navigate to the api folder and install dependencies:

   ```
   cd api
   npm install
   ```

3. Navigate to the client folder and install dependencies:

```
cd client
npm install
```

Usage:

1. Start the backend server:

```
cd api
npm run dev
```

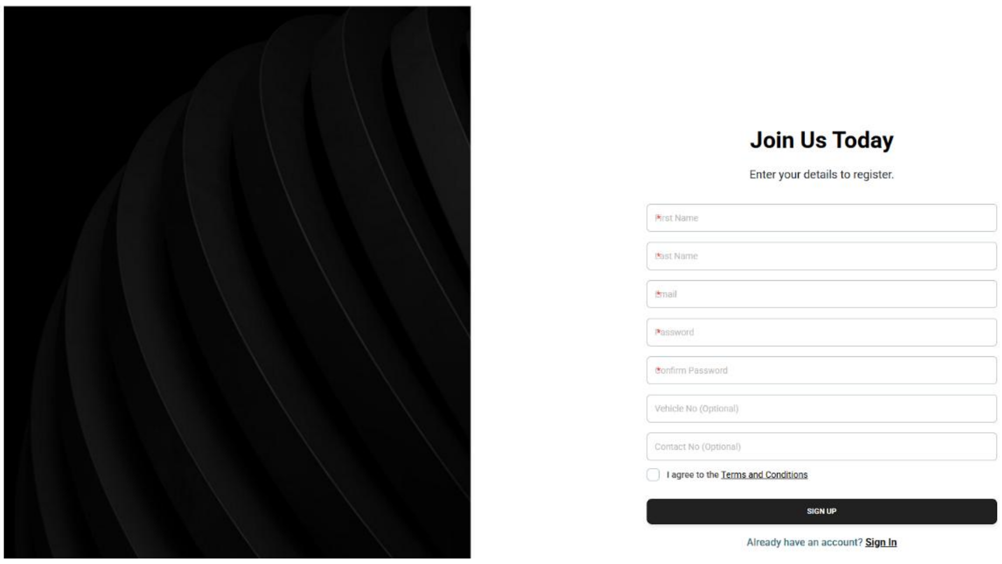
2. Start the frontend development server:

```
cd client
npm run dev
```

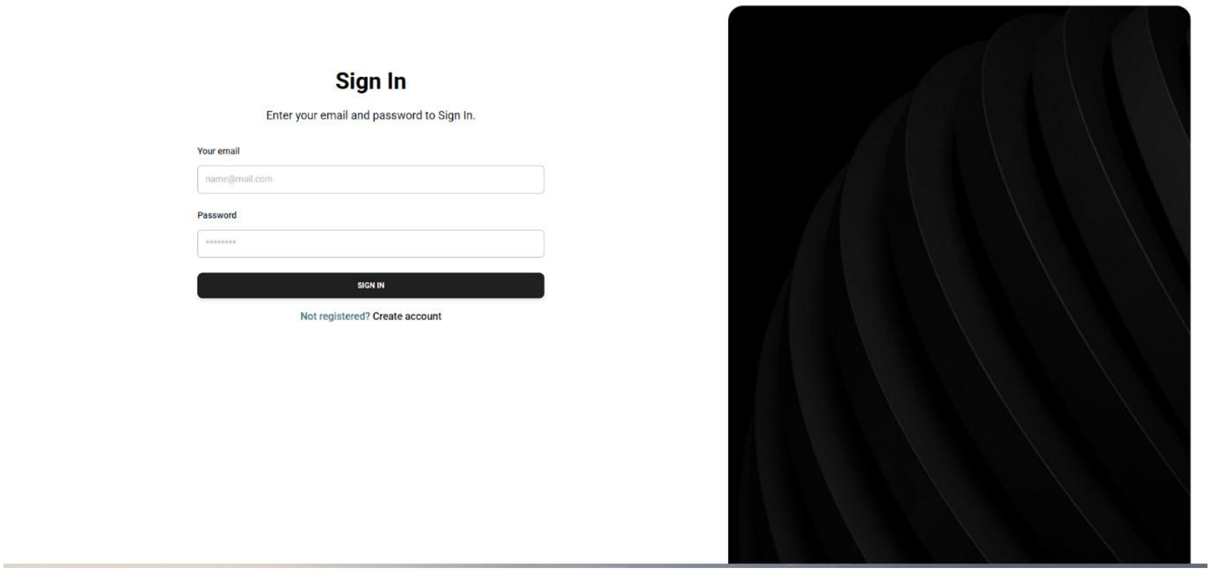
3. Open your browser and navigate to http://localhost:3000 or http://localhost:5173.

User Interfaces

- **Sign Up Page**



- **Sign In Page**



Sign In

Enter your email and password to Sign In.

Your email

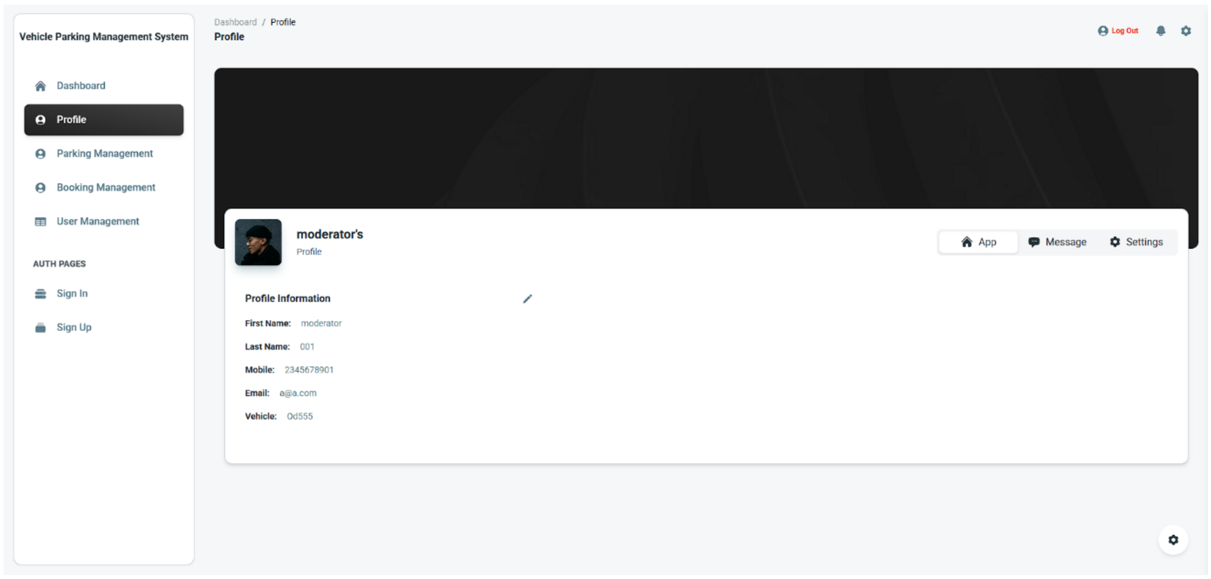
name@mail.com

Password

SIGN IN

Not registered? Create account

- **User Profile**



Vehicle Parking Management System

Dashboard / Profile

Profile

moderator's

Profile

Profile Information

First Name: moderator

Last Name: 001

Mobile: 2345678901

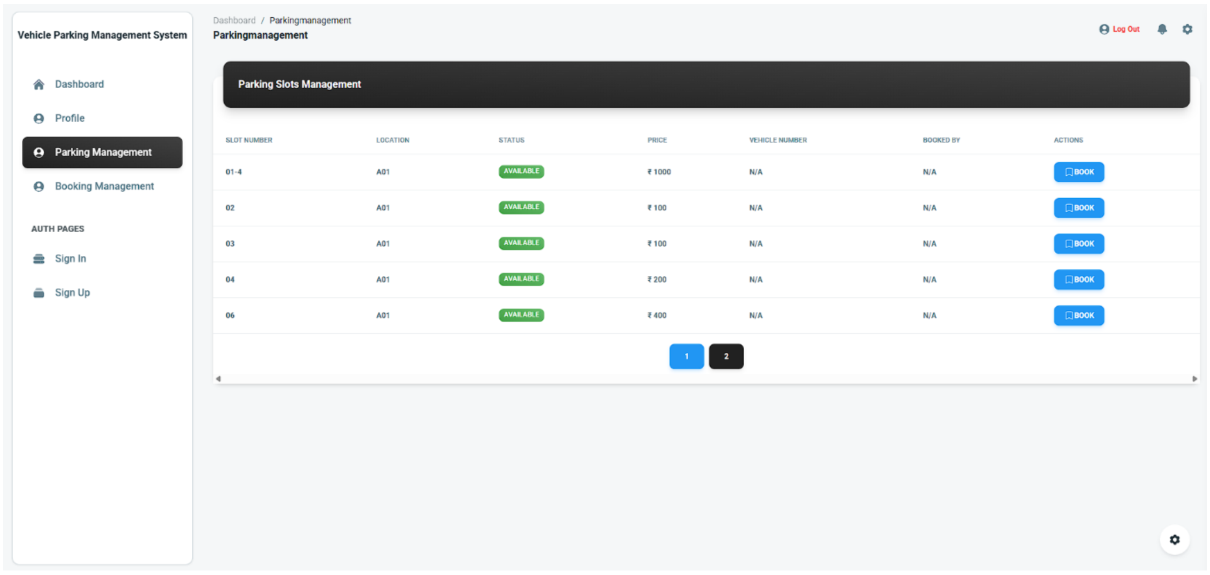
Email: a@a.com

Vehicle: Od555

- **Parking Slot Management**

Team Alpha

- For User



Vehicle Parking Management System

- Dashboard
- Profile
- Parking Management
- Booking Management

AUTH PAGES

- Sign In
- Sign Up

Dashboard / Parkingmanagement

Parkingmanagement

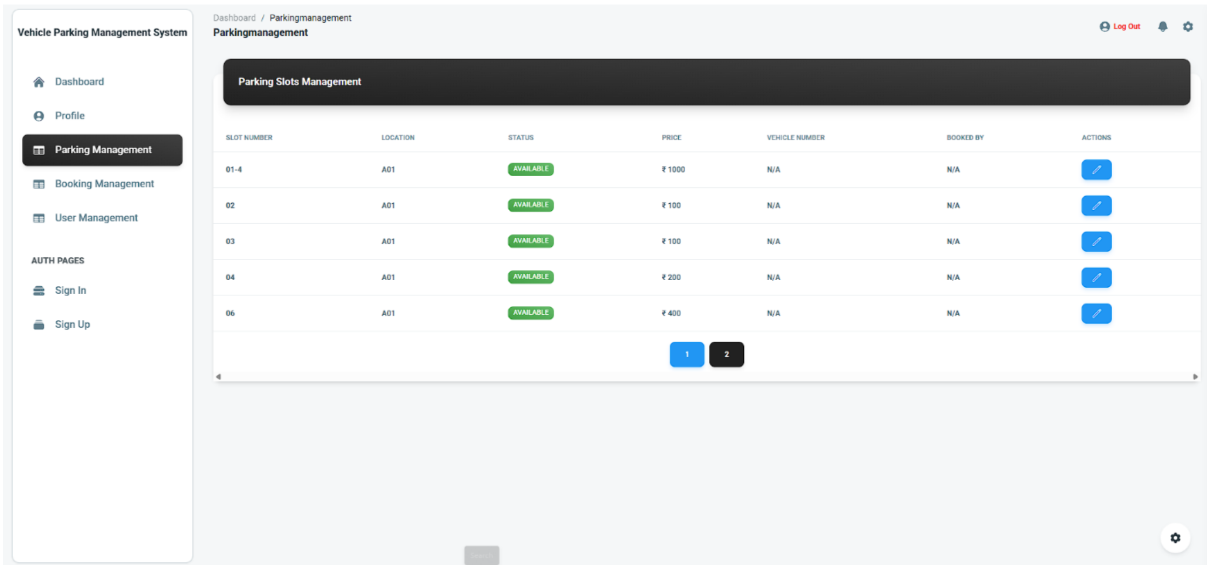
Log Out

Parking Slots Management

SLOT NUMBER	LOCATION	STATUS	PRICE	VEHICLE NUMBER	BOOKED BY	ACTIONS
01-4	A01	AVAILABLE	₹ 1000	N/A	N/A	BOOK
02	A01	AVAILABLE	₹ 100	N/A	N/A	BOOK
03	A01	AVAILABLE	₹ 100	N/A	N/A	BOOK
04	A01	AVAILABLE	₹ 200	N/A	N/A	BOOK
06	A01	AVAILABLE	₹ 400	N/A	N/A	BOOK

1 2

- For Moderator



Vehicle Parking Management System

- Dashboard
- Profile
- Parking Management
- Booking Management
- User Management

AUTH PAGES

- Sign In
- Sign Up

Dashboard / Parkingmanagement

Parkingmanagement

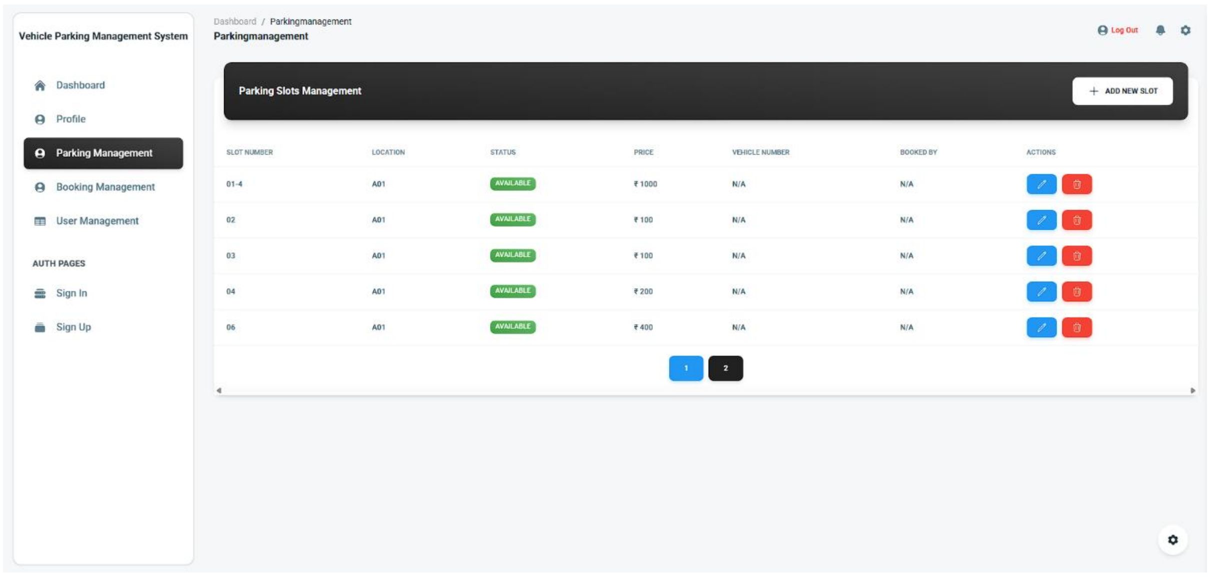
Log Out

Parking Slots Management

SLOT NUMBER	LOCATION	STATUS	PRICE	VEHICLE NUMBER	BOOKED BY	ACTIONS
01-4	A01	AVAILABLE	₹ 1000	N/A	N/A	✎
02	A01	AVAILABLE	₹ 100	N/A	N/A	✎
03	A01	AVAILABLE	₹ 100	N/A	N/A	✎
04	A01	AVAILABLE	₹ 200	N/A	N/A	✎
06	A01	AVAILABLE	₹ 400	N/A	N/A	✎

1 2

- For Admin



Vehicle Parking Management System

- Dashboard
- Profile
- Parking Management
- Booking Management
- User Management

AUTH PAGES

- Sign In
- Sign Up

Dashboard / Parkingmanagement

Parkingmanagement

Log Out

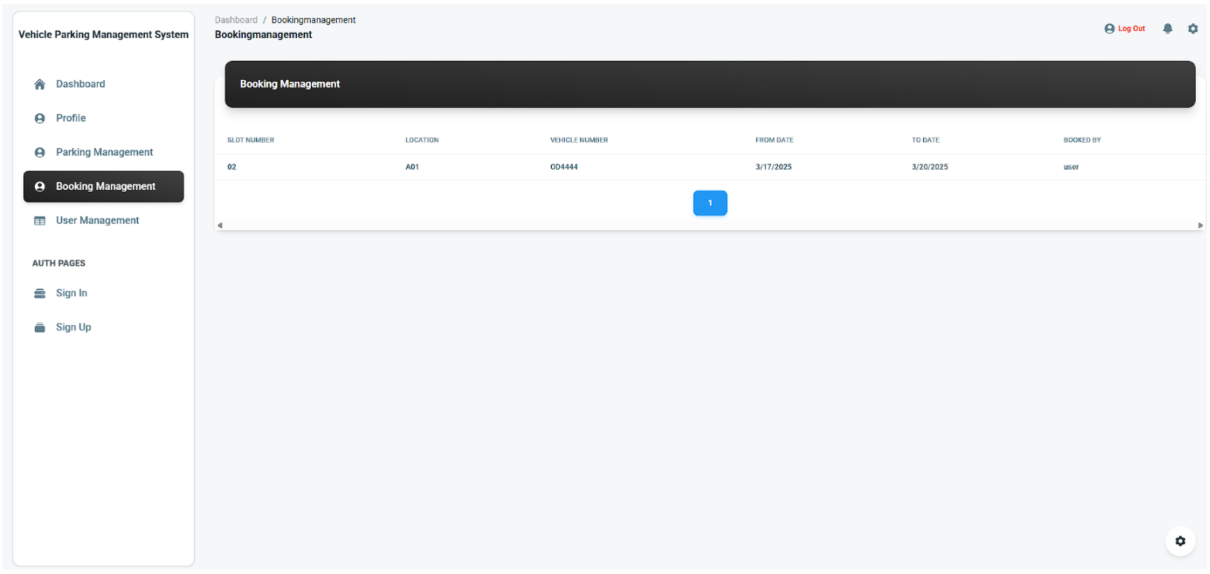
Parking Slots Management

+ ADD NEW SLOT

SLOT NUMBER	LOCATION	STATUS	PRICE	VEHICLE NUMBER	BOOKED BY	ACTIONS
01-4	A01	AVAILABLE	₹ 1000	N/A	N/A	✎ 🗑
02	A01	AVAILABLE	₹ 100	N/A	N/A	✎ 🗑
03	A01	AVAILABLE	₹ 100	N/A	N/A	✎ 🗑
04	A01	AVAILABLE	₹ 200	N/A	N/A	✎ 🗑
06	A01	AVAILABLE	₹ 400	N/A	N/A	✎ 🗑

1 2

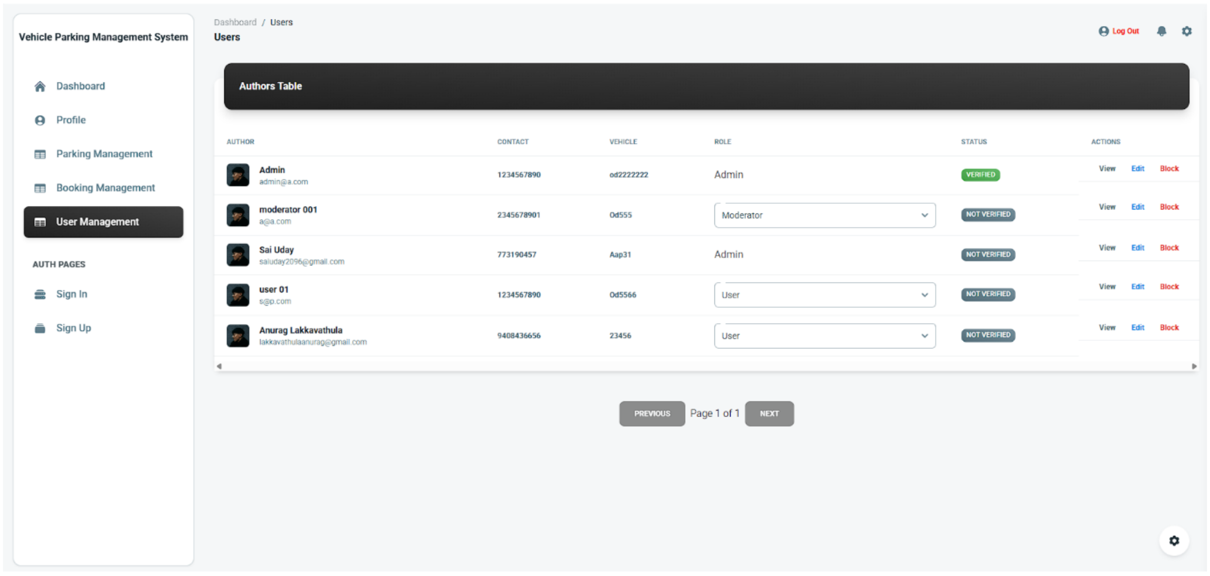
- **Booking Management**



Booking Management

SLOT NUMBER	LOCATION	VEHICLE NUMBER	FROM DATE	TO DATE	BOOKED BY
02	A01	OD4444	3/17/2025	3/20/2025	user

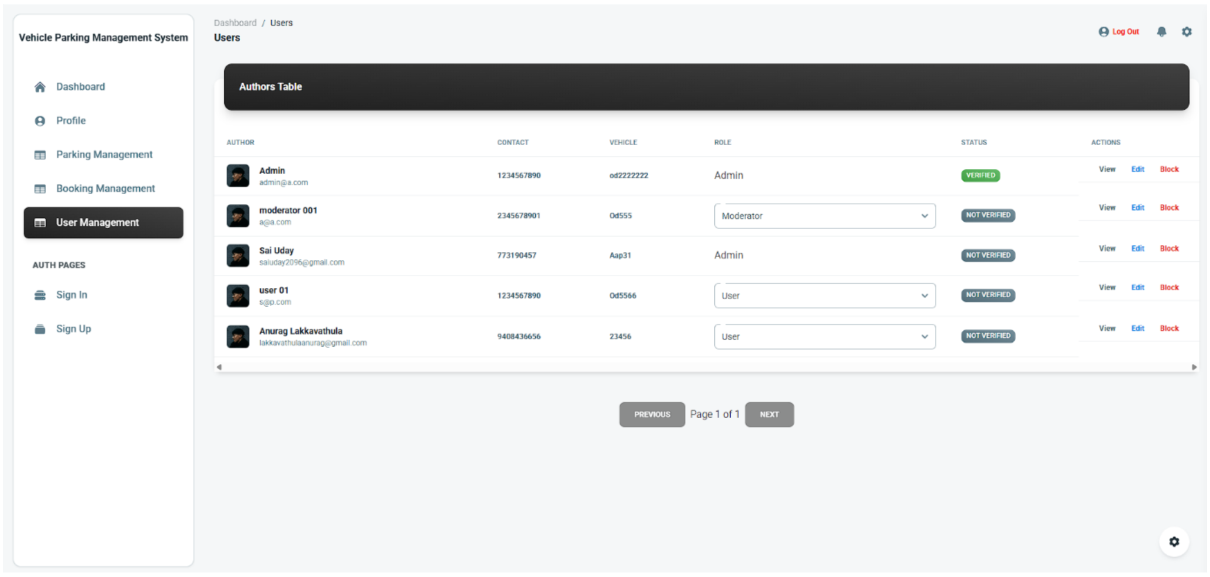
- **User Management**
  - For Moderator



Authors Table

AUTHOR	CONTACT	VEHICLE	ROLE	STATUS	ACTIONS
Admin admin@a.com	1234567890	od2222222	Admin	VERIFIED	View Edit Block
moderator 001 a@a.com	2345678901	0d555	Moderator	NOT VERIFIED	View Edit Block
Sai Uday saiuday2096@gmail.com	773190457	Aap31	Admin	NOT VERIFIED	View Edit Block
user 01 s@p.com	1234567890	0d5566	User	NOT VERIFIED	View Edit Block
Anurag Lakkavathula lakkavathulaanurag@gmail.com	9408436656	23456	User	NOT VERIFIED	View Edit Block

- For Admin



Authors Table

AUTHOR	CONTACT	VEHICLE	ROLE	STATUS	ACTIONS
Admin admin@a.com	1234567890	od2222222	Admin	VERIFIED	View Edit Block
moderator 001 a@a.com	2345678901	0d555	Moderator	NOT VERIFIED	View Edit Block
Sai Uday saiuday2096@gmail.com	773190457	Aap31	Admin	NOT VERIFIED	View Edit Block
user 01 s@p.com	1234567890	0d5566	User	NOT VERIFIED	View Edit Block
Anurag Lakkavathula lakkavathulaanurag@gmail.com	9408436656	23456	User	NOT VERIFIED	View Edit Block

---

e. Steps to Run Test Cases

1. Navigate to the api folder:

```
cd api
```

2. Run the tests:

```
npm test
```

---

f. Peer Review Feedback & Actions Taken

Summary of Feedback Received:

1. **Feedback:** Improve error handling in the backend.
   - **Action Taken:** Enhanced error handling and validation in the backend.
2. **Feedback:** Add pagination for parking slots and bookings.
   - **Action Taken:** Implemented pagination for parking slots and bookings.
3. **Feedback:** Improve user interface for better usability.
   - **Action Taken:** Updated the frontend UI for better user experience.

---

g. Reflection

What Was Accomplished

- Successfully implemented user authentication, parking slot management, booking management, and user management.
- Developed a RESTful API with secure JWT-based authentication.

- Created a responsive frontend using React.js and Material Tailwind.

What Went Well

- The backend API was well-structured and easy to extend.
- The frontend UI was user-friendly and responsive.
- The team collaborated effectively to meet the project deadlines.

Areas for Improvement

- Need to improve error handling and validation further.
- Add more unit tests for better code coverage.
- Enhance the user interface with more features and better design.

---

h. Member Contribution Table:

Member Name	**Contribution Description**	**Overall Contribution (%)**	**Notes (if applicable)**
Jashwanth Nalluri	Project setup (folder structure, MongoDB Atlas, project.env, middleware/routes, JWT token), GitHub management	12%	He has helped creating the whole structure of the Project.
Jaya Sai Reddy Santimalla	Authentication (Sign in, Sign up, Logout)	11%	He has handled the authentication part
Anurag Lakkavathula	User module (User model, User controller, User routes)	11%	He has handled the user management part with respect to backend
Abhishek Darsha	Parking module (Parking model, Parking controller)	11%	He has handled Parking management with respect to backend.
Lakshmi Deepika Yadagiri	Booking module (Booking model, Booking controller)	11%	She has handled Booking Management with respect to backend.
Sai Sri Naga Sashank Pasupuleti	Frontend setup (Dashboard, Profile, and other UI components)	11%	He has handled Front end up for UI displaying and also worked on the Profile section to update the profile.
Dhiresh Venkateshwarlu Katuri	Redux (State management, Sign in, Sign up integration)	11%	He has handled Redux for statement management and integration of sign in and Sign up UI.
Nikhil Chowdary Kollipati	User section (User-related UI components)	11%	He has handled user section i.e. user management with respect to front end.

Jishna Sathvik Vucha	Booking Management and Parking Management functionalities	11%	He has handed Booking and Parking management functionalities with respect to front end UI.