**FULL STACK DEVELOPMENT WITH MERN**

Submitted in accordance with the requirement for the degree of

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Under the faculty Guideship**

**MR.SK. AHAMAD SHARIF**



**DEPARTMENT OF**

**COMPUTERSCIENCE ANDENGINEERING**

**KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY**

(Autonomous)

**Approved by AICTE- New Delhi, Accredited by NAAC with ‘A’ Grade**

**Permanently Affiliated to JNTUK, Kakinada**

**NH-5, Chowdavaram, Guntur – 522019**

**FULL STACK DEVELOPMENT WITH MERN**

**PROJECT DOCUMENTATION**

**PROJECT TITLE:**

**Ride Ready: your ready to go ride booking app**



**Submitted in complete fulfillment**

**of the requirement**

**by**

**Team Members:**

|  |  |
| --- | --- |
| A.TRIVENI | 218X1A0501 |
| D.HARI PRIYA | 218X1A0516 |
| M.BHARATH | 218X1A0541 |
| D.RAM MOHAN REDDY | 218X1A0547 |

**CONTENTS:**

1. Introduction

2. Project Overview

* Purpose
* features

3. Architecture

* frontend
* backend
* datebase

4. Setup Instructions

* pre requisites
* installations

5. Folder Structure

* client
* server

6. Running the Application

7. API Documentation

8. Authentication

9. User Interface

10. Testing

11. Screenshots or Demo

12. Known Issues

13. Future Enhancements

**INTRODUCTION:**

* RideEase is a cab booking app built with the MERN stack, offering a convenient and user-friendly platform for seamless ride bookings. With features like real-time cab tracking, secure payments, and user profile management, RideEase ensures stress-free transportation.
* Users can easily register, schedule rides, and select their preferred vehicle, making travel simple and efficient. The app aims to transform the cab booking experience, providing a reliable and accessible solution for all.

**Project overview:**

* **Purpose :**

1. **Primary Purposes:**

1**.** Provide a convenient and safe ride-hailing experience for users.

2. Connect riders with verified and trusted drivers.

3. Offer real-time tracking and estimated time of arrival (ETA) for rides.

4. Ensure secure payment processing and multiple payment options.

5. Foster a community of respectful and courteous riders and drivers.

1. **Secondary Purposes:**

1. Reduce transportation costs for users.

2. Increase mobility and accessibility for seniors, disabled individuals, and those without vehicles.

3. Provide employment opportunities for drivers.

4. Promote sustainability through eco-friendly transportation options.

* **Features:**

**A. Rider Features:**

1. Easy Booking: Book rides with simplicity and ease.

2. Real-time Tracking: Track your ride in real-time.

3. Estimated Time of Arrival (ETA): Get accurate ETA for your ride.

**B. Driver Features:**

1**.** Ride Requests: Receive ride requests with passenger details.

2. Navigation: Get turn-by-turn navigation to pickup and dropoff locations.

3. Real-time Tracking: Track ride status and passenger location.

**C. Safety Features:**

1. Panic Button: Emergency panic button for riders.

2. GPS Tracking: Real-time GPS tracking for rides.

3. Driver Verification: Background checks and verification drivers.

4. Ride Sharing: Share ride details with friends and family.

5. Emergency Contact: Store emergency contact information**.**

**ARCHITECTURE:**

* **FRONT END:**
* **Header:**

1. Logo and navigation menu

2. Search bar for entering pickup location

3. Buttons for selecting ride type (e.g., Economy, Premium)

* **Home Page:**

1. Hero image or video showcasing Ride Ready benefits

2. Call-to-action (CTA) button for booking ride

3. Featured sections highlighting Ride Ready features

* **Ride Request Page:**

1. Map view displaying pickup and dropoff location.

2. Ride details.

* **BACK END:**

1**.** API Gateway (AWS API Gateway)

2. Application Server (Node.js, Express.js)

3. Database (MongoDB)

4. Authentication Server (OAuth 2.0)

5. Payment Gateway (Stripe)

6. Location Service (Google Maps API)

7. Notification Service (Twilio API)

8. Queue Service (RabbitMQ)

* **Datebase:**
* **Collections:**

1. Users

2. Drivers

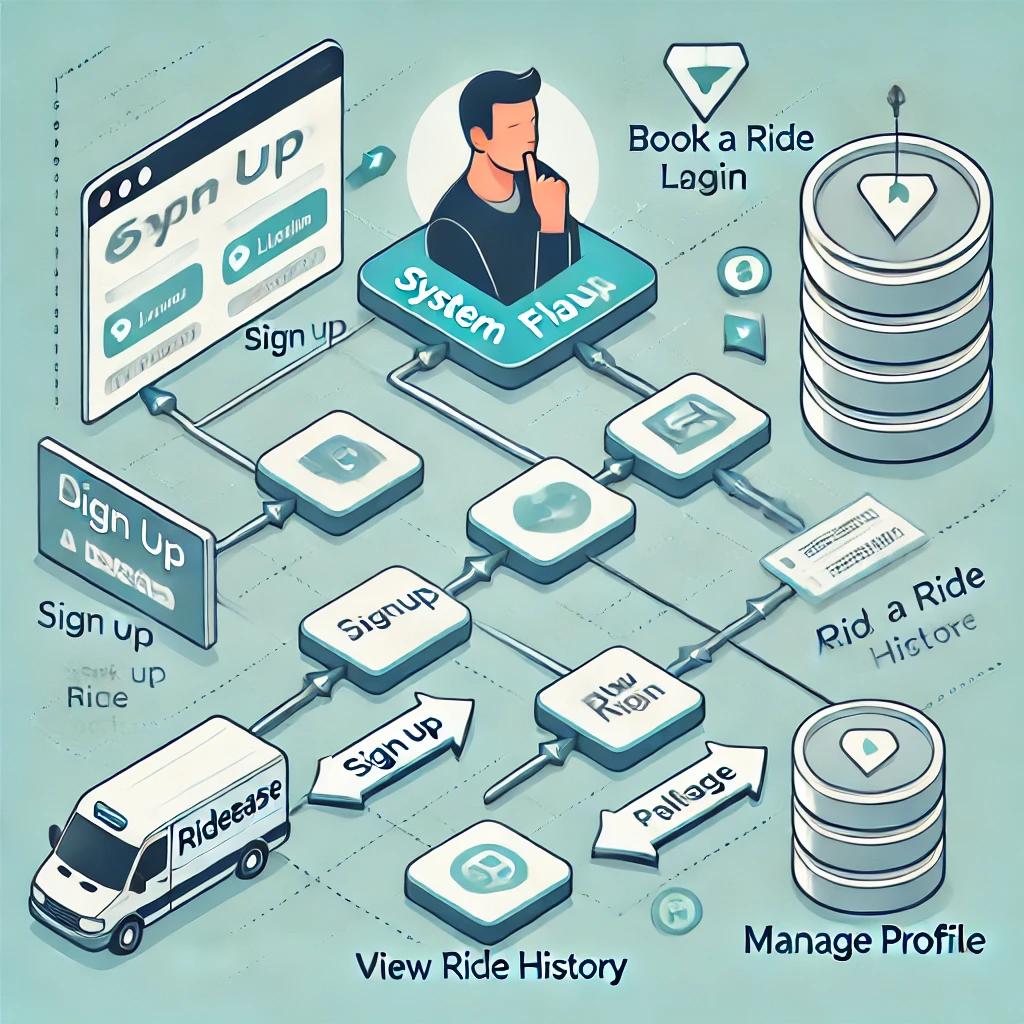
3. Rides

4. Payments

5. Vehicles

6. Locations

7. Reviews



**Setup Instructions:**

* **Pre requisites :**
* **Technical Prerequisites:**

1.Programming languages:JavaScript (Node.js),Java or Python

2. Frameworks: Express.js, React Native or Flutter

3. Databases: MongoDB, MySQL or PostgreSQL

* **Infrastructure Prerequisites:**

1. Servers: Cloud providers (AWS, Google Cloud or Azure)

2. Storage: Cloud storage (AWS S3 or Google Cloud Storage)

3. Networking: Internet connectivity

* **Human Resources Prerequisites:**

1. Development team: Mobile and backend developers

2. Design team: UX/UI designers

3. Quality Assurance team: Testers

4. Project Manager: Coordinate development and launch

* **Financial Prerequisites:**

1. Initial investment: Development costs

2. Ongoing expenses: Server maintenance, marketing

3. Revenue streams: Commission-based ride fees

* **Installations :**

**iOS Installation:**

1. Open App Store on iOS device

2. Search "Ride Ready"

3. Click "Get"

4. Wait for download and installation

5. Open Ride Ready app

**Android Installation:**

1. Open Google Play Store on Android device

2. Search "Ride Ready"

3. Click "Install"

4. Wait for download and installation

5. Open Ride Ready app

**Folder Structure:**

* **Client:**

The React frontend of the Ride Ready app is typically organized into the following components:

* **Components:**

1. Functional Components: Reusable UI elements (e.g., buttons, forms) that manage their own state and lifecycle.
2. Class Components: (if used) for more complex state management or lifecycle methods.

* **Containers**:

Higher-level components that connect to the Redux store or manage application state, often containing multiple child components.

* **Routing**:

Utilizes react-router for managing navigation and rendering different views based on the URL.

* **State Management**:

Uses Context API or Redux for global state management, allowing components to share state easily.

* **Server:**

The Node.js backend of the Ride Ready app is structured for clarity and scalability:

* **Entry Point:**

The main server file (e.g., server.js or app.js) initializes the Express app and sets up middleware.

* **Routes:**

Organized in separate files for different resources(e.g.,userRoutes.js, rideRoutes.js),defining API endpoints.

* **Controller:**

Functions that handle the business logifor each route, separating concerns and improving maintainability.

* **Model:**

Database models, often using an ORM like Sequelize or Mongoose, that define the structure of the data and interactions with the database**.**

* **Configuration:**

Environment variables and application settings managed vi

a config file or package like dotenv.

* **Utilities:**

functions and modules that are used throughout the application**.**

* **Testing:**

Unit and integration tests for API endpoints and business

logic, typically using frameworks like Mocha or Jest**.**

**Running the Application:**

Here are the steps to run the Ride Ready client using npm:

**Step 1:** Navigate to Client Directory

Open your terminal/command prompt and navigate to the client directory:

1. bash
2. cd ride-ready-client

**Step 2**: Install Dependencies

Install the required dependencies using npm:

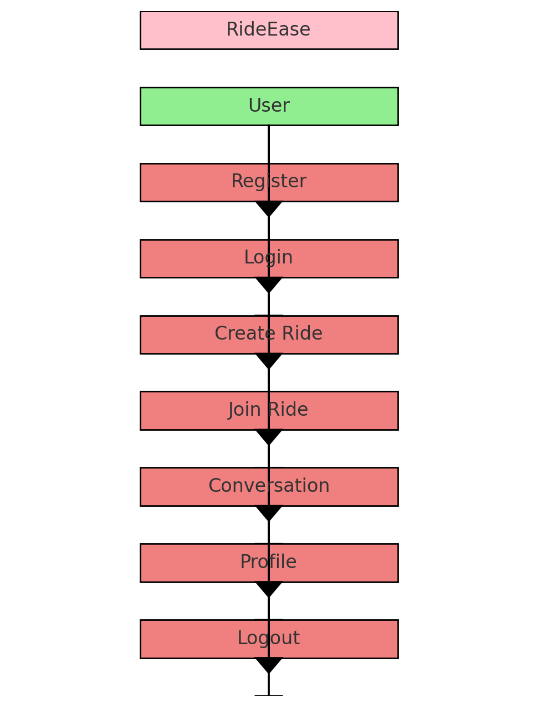
1. bash
2. npm install

**Step 3**: Start Client

Start the client using npm:

1. bash
2. npm start**Top of Form**

**Bottom of Form**



**API Documentation:**

* **Endpoints Overview:**
* **User Registration:**

1. Endpoint: /users/register
2. Method: POST
3. Parameters: username, email, password
4. Responses:

Success: User ID and success message

Error: Email already in use

* **User Login:**

1. Endpoint: /users/login
2. Method: POST
3. Parameters: email, password
4. Responses:

Success: JWT token and success message

Error: Invalid credentials

* **Create Ride Request:**

1. Endpoint: /rides
2. Method: POST
3. Headers: Authorization (Bearer token)
4. Parameters: pickupLocation, dropoffLocation, rideType
5. Responses:

Success: Ride ID and status

Error: Invalid location format

* **Get Ride Status:**

1. Endpoint: /rides/{rideId}
2. Method: GET
3. Headers: Authorization (Bearer token)
4. Parameters: rideId
5. Responses:

Success: Ride details and status

Error: Ride not found

* **Get All Rides for User:**

1. Endpoint: /users/{userId}/rides
2. Method: GET
3. Headers: Authorization (Bearer token)
4. Parameters: userId
5. Responses:

Success: List of rides

Error: User not found

* + - **Cancel a Ride:**

1. Endpoint: /rides/{rideId}/cancel
2. Method: DELETE
3. Headers: Authorization (Bearer token)
4. Parameters: rideId
5. Responses:

Success: Success message

Error: Ride not found

* **Update User Profile:**

1. Endpoint: /users/{userId}
2. Method: PUT
3. Headers: Authorization (Bearer token)
4. Parameters: userId, username, email
5. Responses:

Success: Success message

Error: User not found

**Authentication:**

Here's an overview of how authentication and authorization are handled in Ride Ready:

* **Authentication:**

1. User Registration: Users register with email, password, and other details.

2. Password Hashing: Passwords are hashed using bcrypt.

3. Token Generation: Upon successful registration, a JSON Web Token (JWT) is generated.

* **Authorization:**

1. Token Verification: Tokens are verified on each request to ensure authenticity.

2. Role-Based Access Control (RBAC): Roles (e.g., user, driver, admin) determine access levels.

3.Route Protection: Routes are protected using middleware to check token validity and role permissions

* **Token Management**

1. Token Type: JWT (JSON Web Token)

2. Token Expiration: 1 hour (configurable)

3. Token Renewal: Tokens are renewed upon expiration

4.Token Storage: Tokens are stored in cookies or local storage

* **Session Management:**

1. Session Type: Stateless (using tokens)

2. Session Expiration: Tokens expire after 1 hour (configurable)

**User Interface:**

* **Home Screen:**
* Highlight the ride request button and location input field.
* Show available ride options (e.g., standard, premium).
* **Map View:**
* Display the interactive map with user and driver locations.
* Highlight nearby drivers and estimated arrival times**.**
* **Ride Request Confirmation:**

Show the confirmation screen after a ride is requested, including driver details and estimated time of arrival.

* **In-Ride Interface:**

Showcase the screen during an active ride, displaying route navigation, driver info, and emergency contact options.

* **Payment Options**:

Highlight the payment screen, showcasing various payment methods (credit card, PayPal, etc.).

* **Ride History:**

Display the user’s ride history, with options to review past trips and details

* **Profile Settings**:

Showcase the user profile screen, where users can update personal information, payment methods, and preferences.

* **Notifications:**

Display notification banners for ride confirmations, driver arrivals, and important updates.

* **Help and Support:**

Show the help section where users can access FAQs or contact support.

* **Accessibility Features**:

Highlight options for accessibility settings, such as voice commands or high-contrast modes.



**Testing:**

* **Unit testing:**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it isdone after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

* **Integration testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unt testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

* **Acceptance Testing:**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

* **Functional testing:**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised

* **White Box Testing:**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

* **Black Box Testing:**

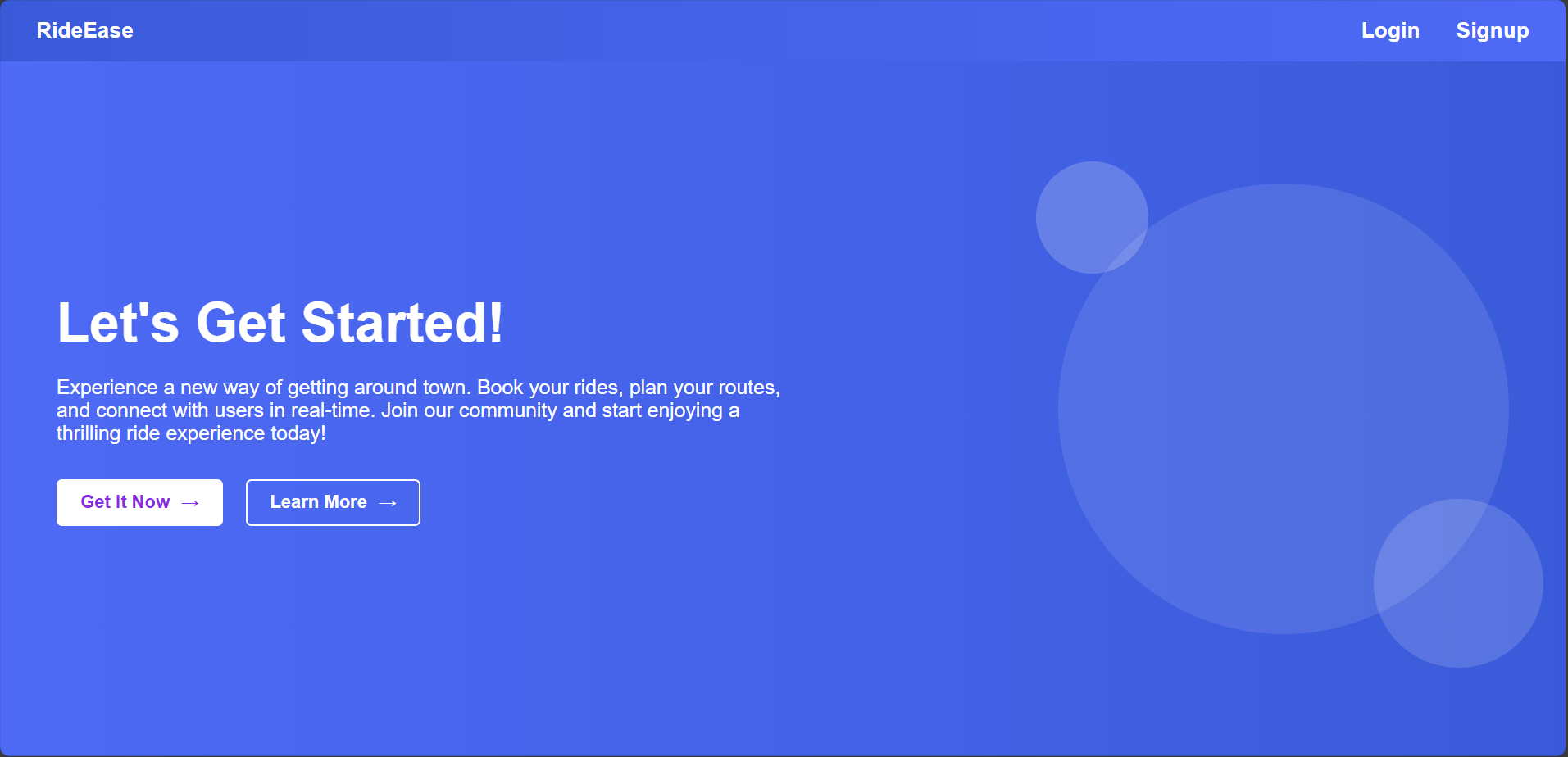
Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Precondition** | **Test Steps** | **Expected Result** |
| TC01 | Register a new user | User is on the registration page | 1. Enter valid user details (e.g., username, email, password).  2. Click on "Register" button. | User is successfully registered, and a confirmation message appears. |
| TC02 | Register with existing email | User is on the registration page | 1. Enter details with an existing email.  2. Click on "Register" button. | Registration fails with an error message stating that the email is already in use. |
| TC03 | Login with valid credentials | User is on the login page | 1. Enter correct email and password.  2. Click on "Login" button. | User is successfully logged in and redirected to the dashboard. |
| TC04 | Login with invalid credentials | User is on the login page | 1. Enter incorrect email or password.  2. Click on "Login" button. | Login fails with an error message indicating incorrect email or password. |
| TC05 | Create a video conference link | User is logged in | 1. Navigate to the "Create Conference" section.  2. Click on "Generate Link" button. | A unique conference link is generated and displayed to the user. |
| TC06 | Join a video conference | User has a valid conference link | 1. Click on the shared conference link.  2. Wait for the conference to load. | User successfully joins the video conference, and audio/video streams are established. |
| TC07 | Join a conference with an invalid link | User has an invalid conference link | 1. Click on an invalid or expired conference link.  2. Wait for the conference to load. | User receives an error message indicating that the link is invalid or expired. |
| TC08 | View user profile | User is logged in | 1. Navigate to the "Profile" section. | User profile details (e.g., name, email, etc.) are displayed. |
| TC09 | Update profile information | User is logged in and on Profile page | 1. Click on "Edit Profile".  2. Modify profile details.  3. Click on "Save" button. | Profile information is updated, and a success message is shown. |
| TC10 | Logout from the application | User is logged in | 1. Click on the "Logout" button. | User is logged out, and the session is terminated, redirecting to the login page. |

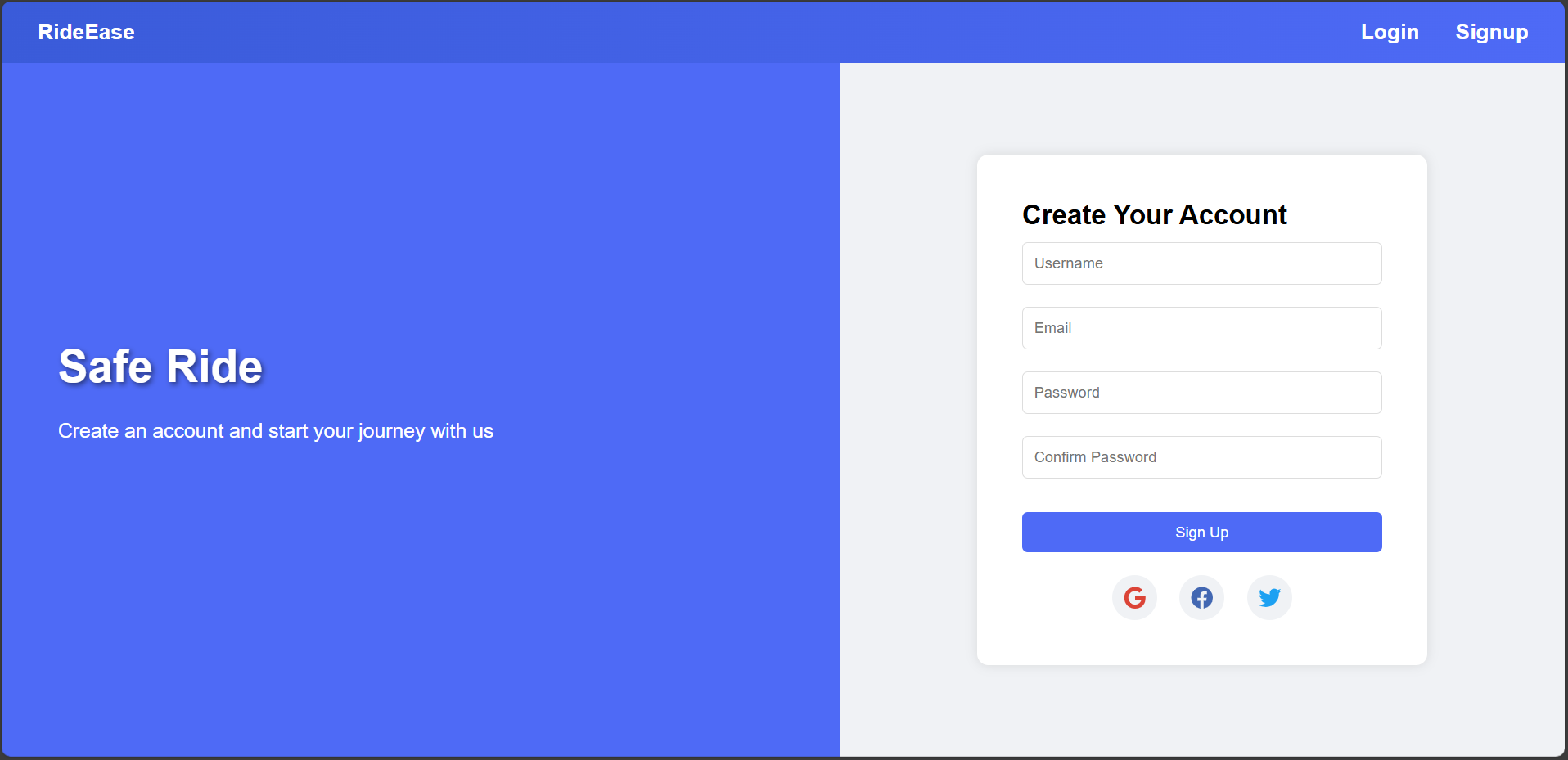
**Screenshots or Demo:**

**Figure 1:Homepage:**

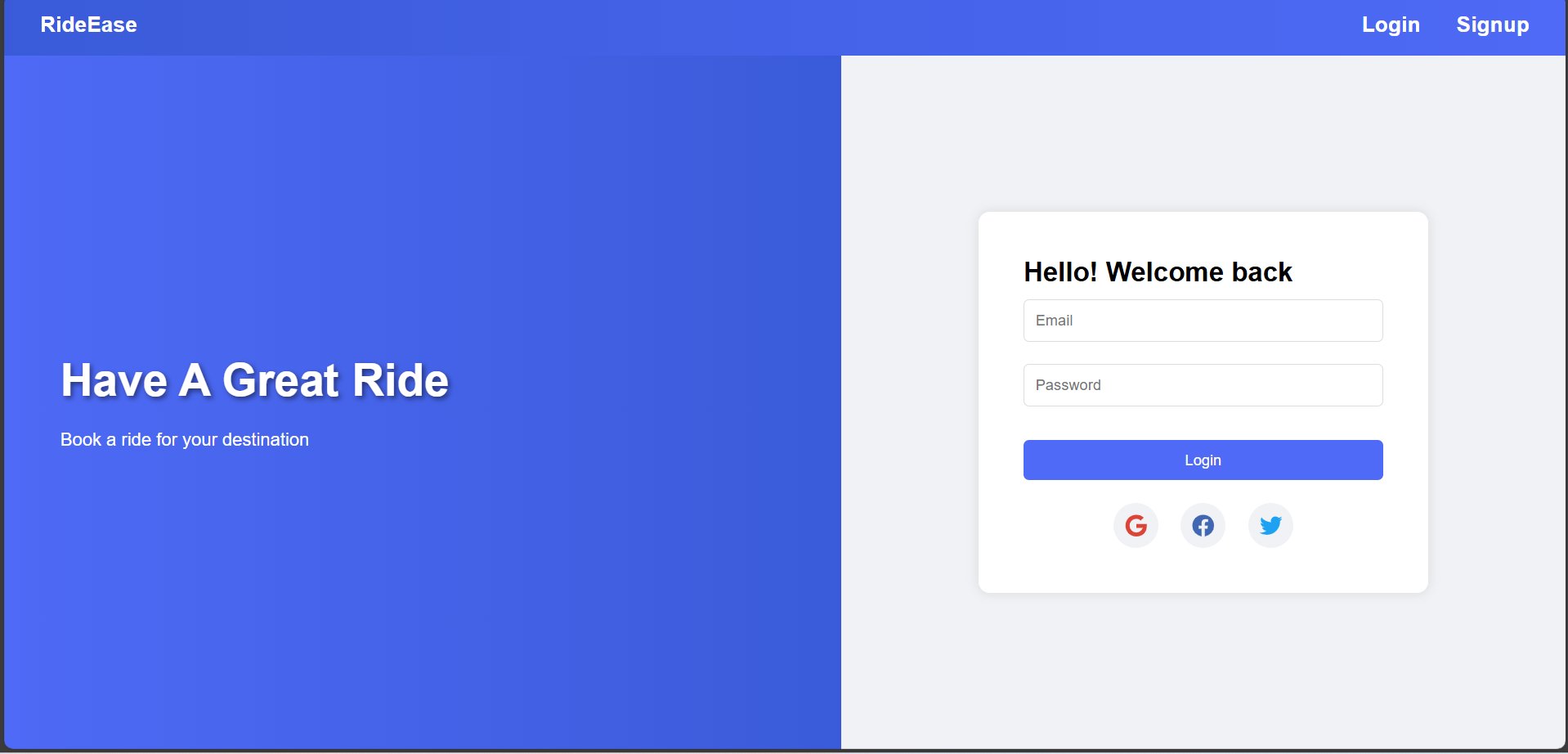
Default landing page for user where he will get the information about the project

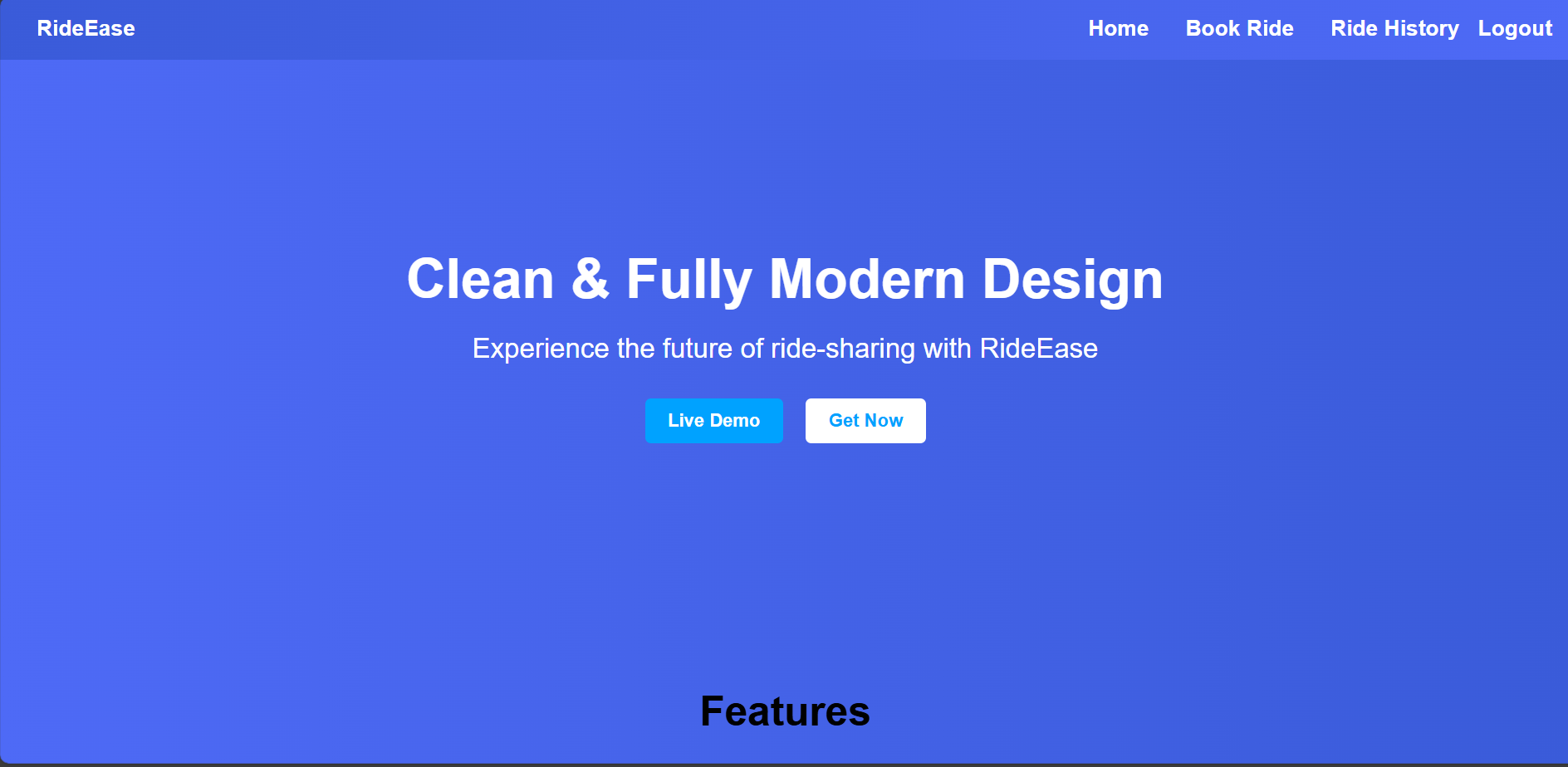


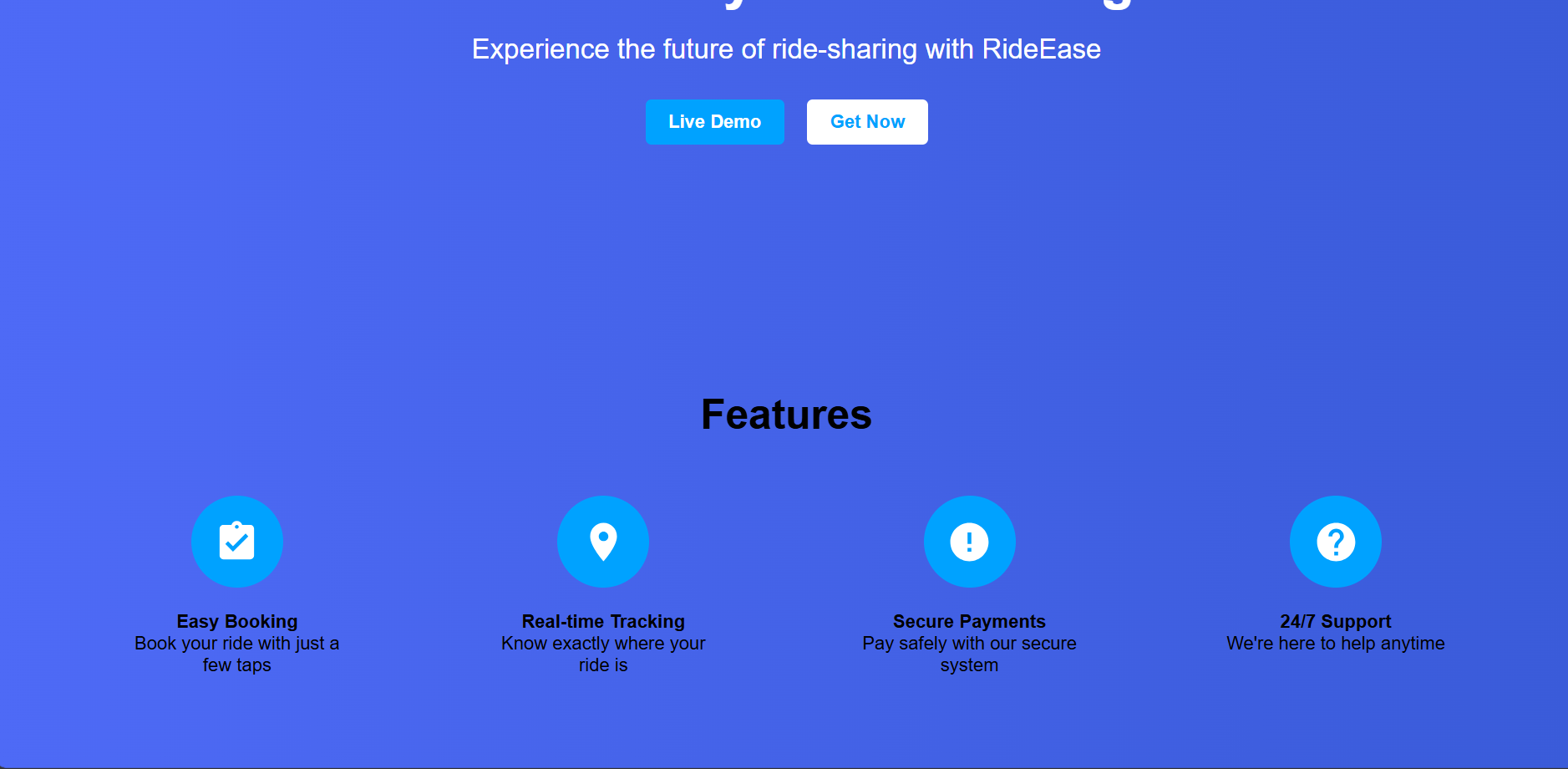
**Figure 2: Register page:** Save user data to the database and send a confirmation email

****

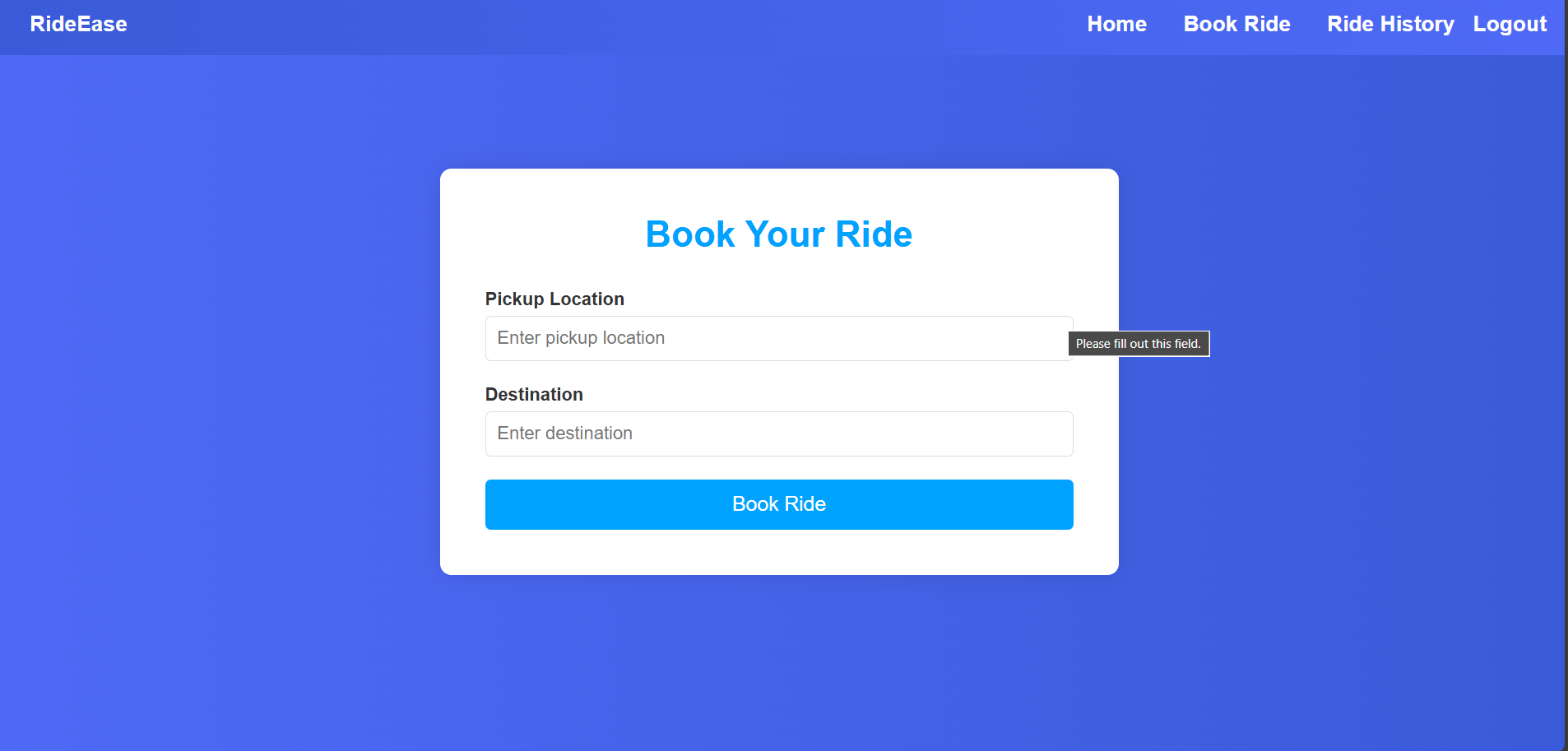
**Figure 3: Login page,** Authenticate user and start session

****

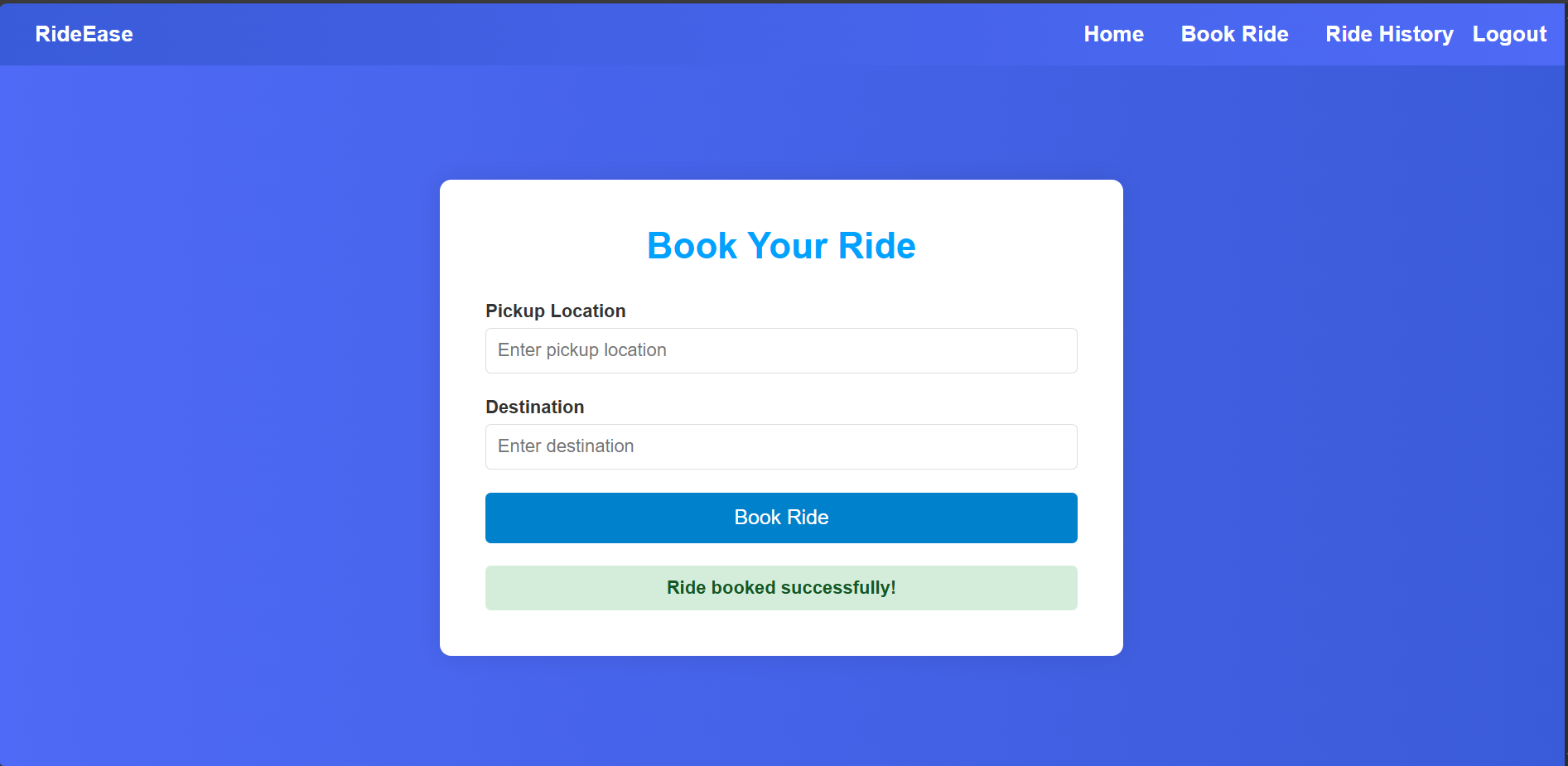
****

****

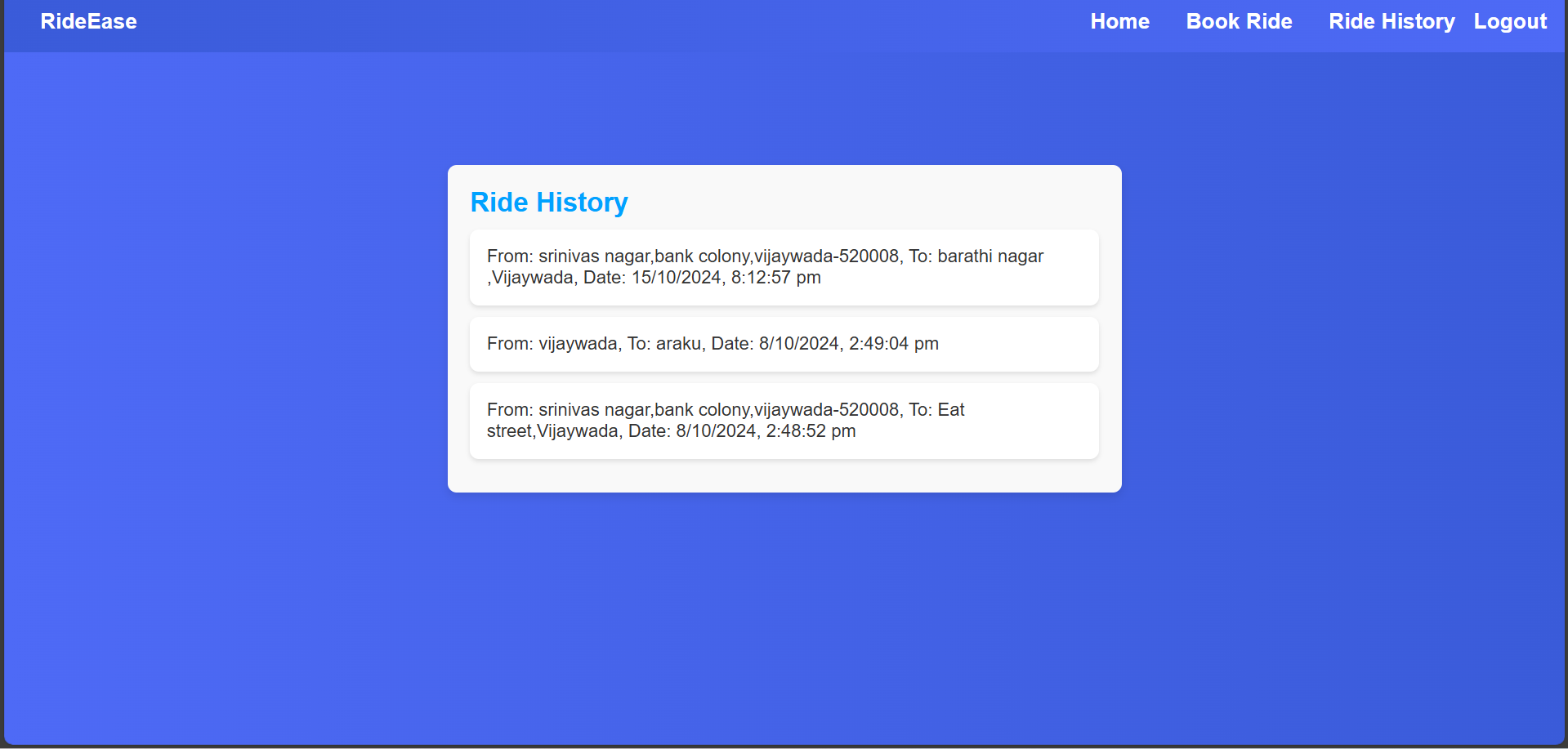
**Figure 4:Booking A Ride :** Here user will Book a Ride

****

This page pop-up for booking ride with user requirement

****

**Figure 5:Ride History :** User can see the Ride History



**Known Issues:**

* **GPS Location Accuracy:**
* Description: Users may experience inaccurate location tracking, which can lead to incorrect pickup/drop-off points.
* Impact: Delayed rides and user frustration.
* **Payment Processing Errors**
* Description: Some users report failed transactions or incorrect billing amounts.
* Impact: Loss of trust and potential loss of customers.
* **App Crashes**
* Description: The app may crash unexpectedly, particularly during high-demand times or when accessing certain features.
* Impact: Interrupts user experience and can lead to missed rides.
* **User Interface Glitches**
* Description: Visual inconsistencies, such as buttons not responding or overlapping text.
* Impact: Confusing user experience and potential navigation issues.
* **Notification Failures**
* Description: Users may not receive notifications for ride confirmations, driver messages, or alerts.
* Impact: Increased uncertainty and missed communications.
* **Delayed Driver Matching**
* Description: Users may experience longer wait times for driver assignments during peak hours.
* Impact: User dissatisfaction and potential loss of business.
* **Inaccurate Ride History**
* Description: Issues with displaying past rides, including missing or incorrect details.
* Impact: Confusion regarding trip history and charges.
* **Profile Update Issues:**
* Description: Difficulty in updating user information, such as payment methods or personal details.
* Impact: Outdated or incorrect user data.
* **Language Localization Problems**
* Description: Inconsistent translations or errors in localized versions of the app.
* Impact: Confusion for non-native speakers**.**
* **Compatibility Issues**
* Description: Certain versions of mobile operating systems may experience performance issues or crashes.
* Impact: Limited accessibility for some users.

**Future Enhancements:**

As the need for efficient ride-sharing services grows, several enhancements could be made to improve the Rideease experience. Integrating dynamic ride matchingbased on real-time traffic and userpreferences could optimize routes and reduce travel time. Additionally, implementing advanced safety features such as ride tracking, emergency alerts, and driver verification using AI could enhance user security. Offering personalized ride options like preferred drivers, music choices, and climate control would cater to individual user comfort. Expanding the app to support multilingual interfaces and currency conversions would make Rideease accessible to a global audience. Introducing features like ride scheduling for recurring trips and integration with calendar apps for automatic reminders would enhance time management. Lastly, utilizing data analytics to provide insights on driver performance, fuel efficiency, and user satisfaction could help optimize both driver and rider experiences.

**CONCLUSION:**

The **Rideease** application, developed using the MERN stack, provides a reliable and user-friendly platform for ride-sharing and commuting. With core features like user registration, secure sign-in, ride booking, and real-time tracking, the app addresses the growing demand for convenient and efficient transportation solutions. Its intuitive interface ensures that users can effortlessly navigate through the app, making it suitable for both daily commuters and casual users. As urban mobility becomes increasingly crucial, Rideease not only enhances the convenience of ride-sharing but also sets the stage for future improvements to further elevate the user experience. With a solid foundation in place, the app is well-equipped to evolve with emerging trends in the transportation industry, ultimately contributing to safer, faster, and more efficient travel options for users worldwide.