Title : Cruise Cabs FSD Project

Akshath Kheria

CHRIST (Deemed to be University) Yeshwantpur Road Campus, Bangalore

CLASS CODE - BCA 261-2

Dr.Prof.Mahalakshmi J

Date 13/3/2025

ACKNOWLEDGEMENT

I express my sincere gratitude to Dr. Mahalakshmi J, Subject teacher, for their valuable guidance throughout this project. Their insightful suggestions and continuous support have played a crucial role in the successful completion of this project.

I also extend my heartfelt appreciation to Christ University for providing the necessary infrastructure and resources that facilitated the development of ShopEase. I am thankful to my professors and peers for their valuable feedback and motivation throughout the process.

My deepest gratitude goes to my family and friends, who have constantly encouraged me and supported me throughout this journey. Lastly, I acknowledge the open-source communities, online tutorials, and documentation that have been instrumental in my learning and development process.

Cruise Cabs is a full-stack web application designed to streamline cab reservations by offering users a seamless, efficient, and secure platform for booking rides. This project integrates user authentication, ride booking, ride history tracking, and an interactive user interface to ensure a smooth experience for both passengers and drivers.

The frontend of Cruise Cabs is built using HTML, CSS, and JavaScript, ensuring a dynamic, responsive, and user-friendly interface. The backend leverages Firebase for real-time database management, user authentication, and secure data storage. Firebase's cloud-based architecture allows for quick and efficient ride bookings and updates.

Key Features:

User Authentication: Secure login and signup using Firebase Authentication.

Cab Booking System: Users can select pickup and drop-off locations, view available cabs, and confirm bookings.

Ride History Tracking: Users can access past ride details for easy reference.

Real-Time Database Management: Firebase ensures instant updates for ride requests and bookings.

Interactive UI: A modern, responsive design optimized for various devices.

Security Measures:

Input Validation: Prevents incorrect or malicious data entry.

Real-Time Data Storage: Ensures safe and instant storage of user and ride information.

Authentication Protocols: Secure handling of user login credentials.

Future Enhancements:

Payment Gateway Integration: Allowing users to make secure online payments for rides.

Driver Management System: Assigning and tracking driver availability.

AI-Based Ride Suggestions: Offering personalized ride recommendations based on user preferences.

Live Ride Tracking: Implementing real-time GPS tracking for booked rides.

Cruise Cabs aims to provide a reliable, user-friendly, and scalable solution for modern cab reservations, ensuring efficiency and convenience for all users.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Overview of the System

Cruise Cabs is a modern cab reservation web application designed to address the challenges of traditional ride-booking platforms. It leverages the power of full-stack development to provide a seamless, efficient, and user-friendly cab booking experience. The system ensures smooth navigation, real-time ride updates, secure authentication, and effective database management.

Key Features:

• User Registration & Authentication – Secure sign-up and login system using Firebase Authentication.

• Cab Booking System – Users can book rides by selecting pickup and drop-off locations.

• Ride History Tracking – Users can view and manage past ride details.

• Database Management – Firebase handles real-time storage and updates for user and ride data.

• Contact & Inquiry System – Customers can reach out via the contact form for support or ride-related inquiries.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM (Cruise Cabs)

Traditional cab reservation systems involve manual booking processes or inefficient web platforms, leading to long wait times and poor user experience. Most platforms require users to go through multiple steps to book a ride, often lacking real-time updates and user-friendly interfaces.

2.1.1 LIMITATIONS OF EXISTING SYSTEM

• Time-Consuming: Traditional ride-booking systems require multiple steps, making the process lengthy and inefficient. Cruise Cabs streamlines booking with a simple and fast reservation system.

• Limited Flexibility: Many existing systems lack real-time updates on ride availability. Cruise Cabs leverages Firebase to provide real-time booking updates.

• Resource Intensive: Traditional systems rely heavily on backend processing for handling ride data. Cruise Cabs optimizes performance using Firebase's cloud-based architecture, reducing server-side processing.

• Inefficient Communication: Many cab platforms lack an integrated communication system for users and drivers. Cruise Cabs incorporates a contact form for direct inquiries.

• Scalability Challenges: Some platforms face performance issues as demand increases. Cruise Cabs, built with Firebase, efficiently scales to accommodate high user traffic.

2.2 PROPOSED SYSTEM –

Cruise Cabs Cruise Cabs is designed to offer a smooth and efficient cab reservation experience by integrating real-time booking, authentication, and ride history tracking.

Key Enhancements:

• User Authentication & Profile Management – Secure registration and login system.

• Ride Booking System – Users can book cabs instantly with an intuitive UI.

• Real-Time Ride Updates – Firebase ensures real-time availability and booking updates.

• Ride History Tracking – Users can review and manage past ride details.

• Interactive & Responsive UI – Optimized for desktops, tablets, and mobile devices.

2.2.1 BENEFITS OF THE PROPOSED SYSTEM – Cruise Cabs

• Efficiency: Cruise Cabs ensures quick and hassle-free cab bookings.

• Enhanced User Experience: With real-time ride updates and a modern UI, the system offers a superior user experience.

• Secure Data Management: Firebase provides robust security for user authentication and ride data storage.

• Scalability: The system is designed to handle increased traffic and growing user demands.

• Real-Time Communication: Users can reach out for support via the integrated inquiry system.

2.3 FUNCTIONAL REQUIREMENTS – Cruise Cabs

• User Authentication & Management:

- Secure login and registration system.
- User roles:
    - Passengers: Book rides and view ride history.
    - Admins: Manage bookings and monitor user activity.

• Cab Booking System:

- Select pickup and drop-off locations.
- View available cabs in real-time.
- Instant ride confirmation.

• Ride History Tracking:

- Users can view their past rides.
- Admins can analyze ride data for improvements.

• Security & Data Protection:

- Secure authentication with Firebase.
- Encrypted storage of user credentials.
- Prevent unauthorized access.

• Error Handling & User Feedback:

- Clear messages for booking failures or login issues.
- Help center for resolving common queries.

• Scalability & Performance Optimization:

- Efficient handling of ride requests and bookings.
- Firebase ensures real-time synchronization and quick data retrieval.

2.4 SOFTWARE REQUIREMENTS – Cruise Cabs

• HTML, CSS, and JavaScript:

- HTML structures the web pages.
- CSS styles the UI for a responsive experience.
- JavaScript handles interactivity and real-time updates.

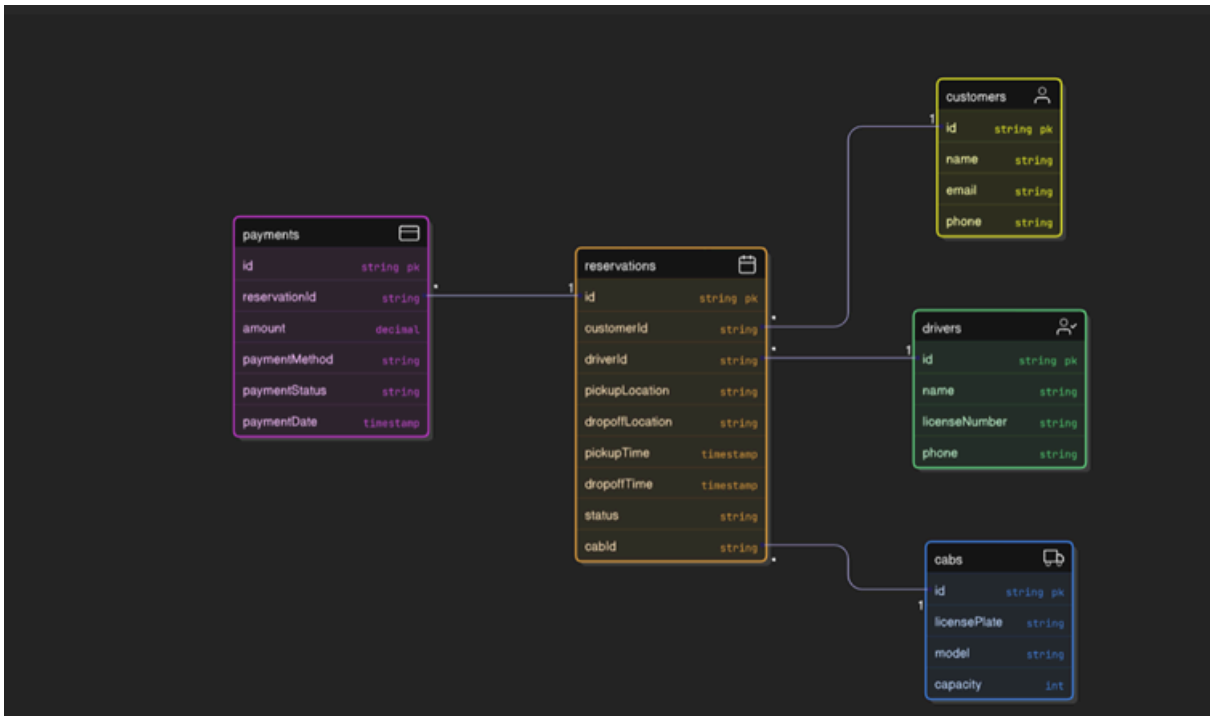
• Firebase:

- Real-time database for storing ride and user data.
- Authentication service for secure login.
- Hosting for seamless application deployment.

Cruise Cabs is built with a focus on efficiency, security, and user experience, making cab reservations faster and more accessible.

3. SYSTEM DESIGN

3.1 Database Design



3.2 User Flow Diagram

Step 1: User Registration & Login

- New users sign up using email and password authentication via Firebase.
- Existing users log in securely to access their profile and ride history.

Step 2: Booking a Ride

- User enters pickup and drop-off locations in the booking form.
- System fetches available cabs and displays estimated ride fare.
- User confirms the booking.

Step 3: Ride Confirmation & Updates

- The system sends a confirmation message with ride details.
- Firebase updates ride status in real time.
- Admins can monitor ongoing bookings and user activity.

Step 4: Ride Completion & Payment (Future Enhancement)

- Upon reaching the destination, the ride status is updated as completed.
- Users can leave feedback and rate their experience.
- Future updates may include online payment integration.

Step 5: Viewing Ride History

- Users can access a list of past rides from their profile.
- Ride details, including fare and time, are stored securely in Firebase.

Step 6: Inquiry & Customer Support

- Users can submit queries or complaints via the contact form.
- Admins can respond to inquiries in real time.

Cruise Cabs is built with a focus on efficiency, security, and user experience, making cab reservations faster and more accessible.

4.1 Technology Stack

• Frontend: HTML, CSS, JavaScript
• Database: Firebase (for real-time data storage and authentication)
• Authentication: Firebase Authentication (Google Sign-in, Email/Password)
• Hosting: Firebase Hosting (for seamless deployment)
• Storage: Firebase Firestore (for ride and user data management)

4.2 Source Code Structure

• Frontend:

- Handles UI components for booking, ride history, and authentication.
- Uses JavaScript for real-time updates and user interactions.

• Database (Firebase Firestore):

- Stores user information, ride bookings, and history.
- Provides real-time synchronization for seamless ride updates.

• Authentication (Firebase Auth):

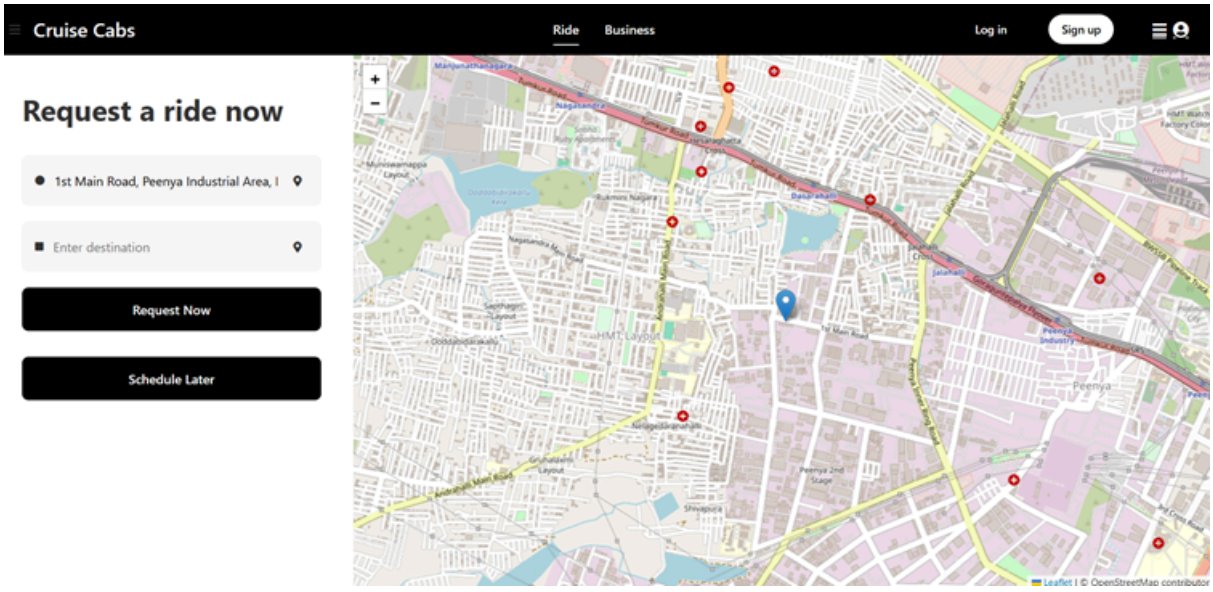
- Manages user login and registration securely.

• Error Handling & Security Measures:

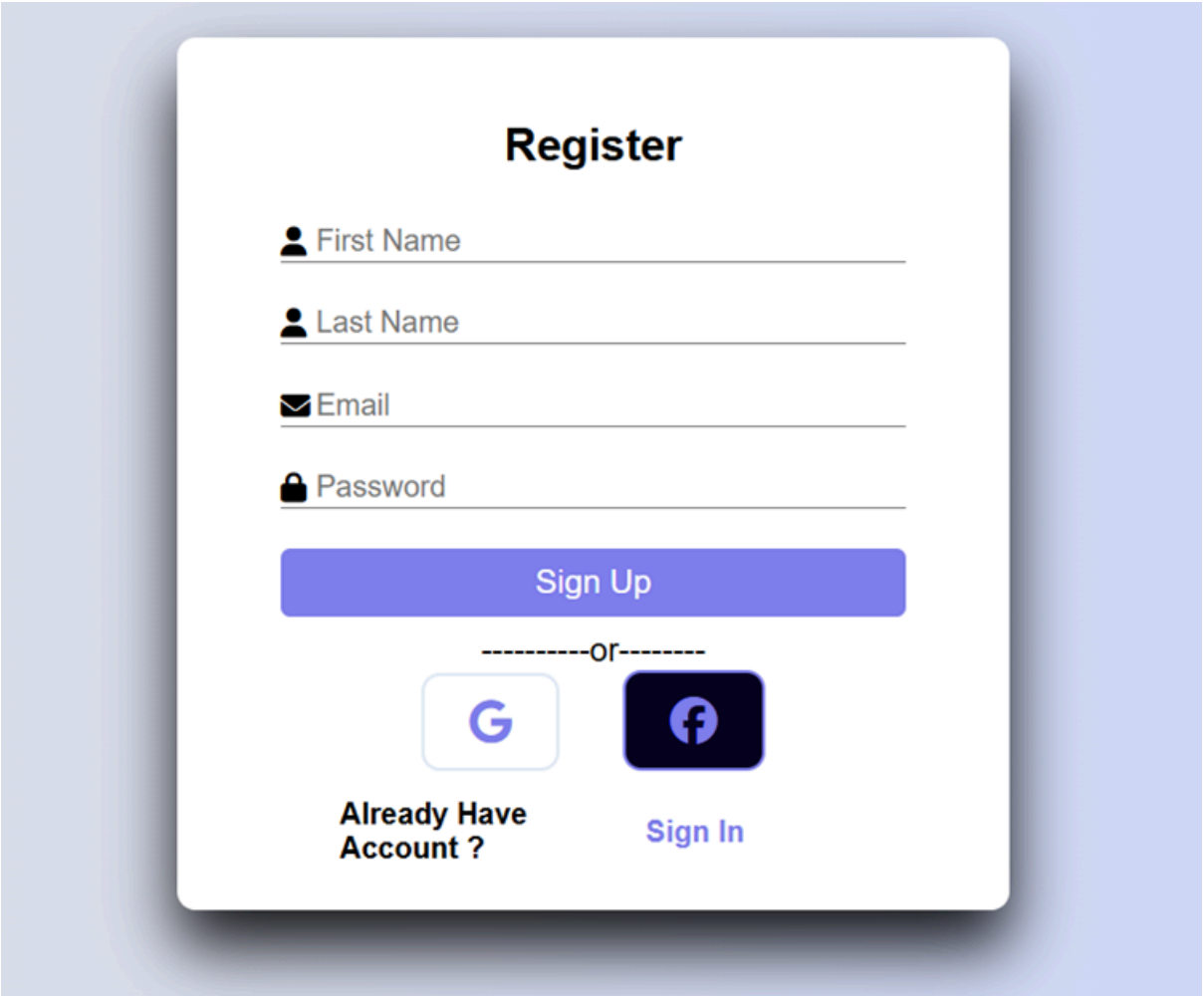
- Input validation for user details and booking information.
- Firebase rules ensure data security and restricted access.

4. Screenshot

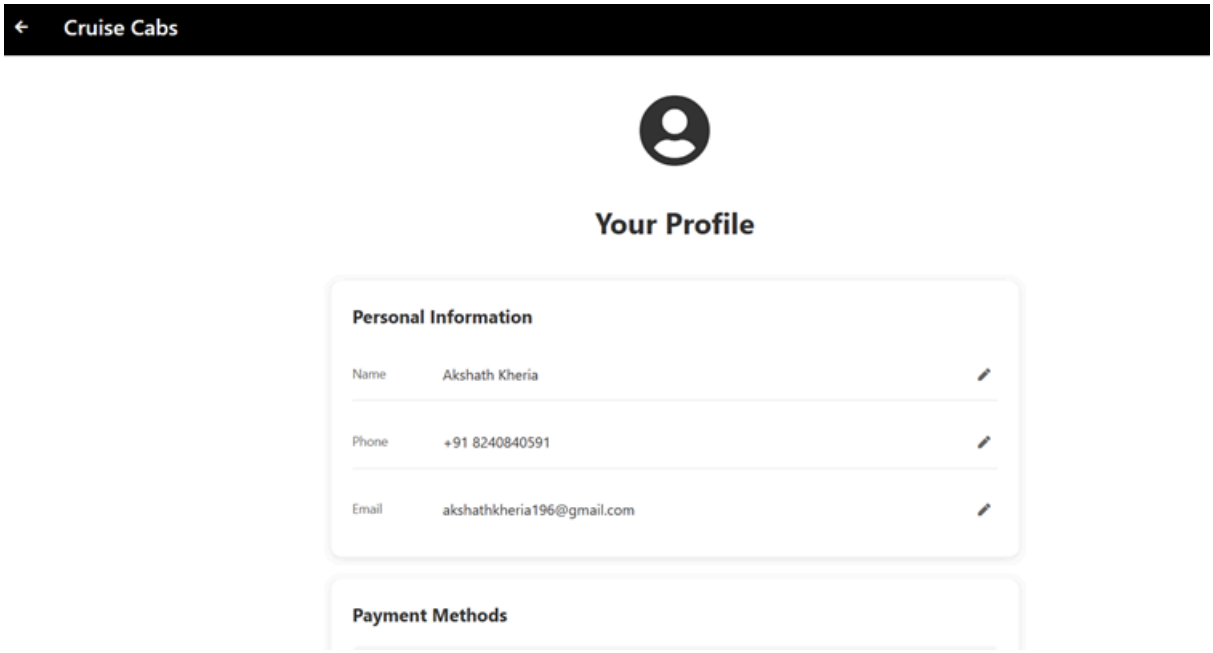
Main page


Cruise Cabs
Ride
Business
Log in
Sign up
Request a ride now
1st Main Road, Peenya Industrial Area, I
Enter destination
Request Now
Schedule Later
Peenya
Leaflet | © OpenStreetMap contributors

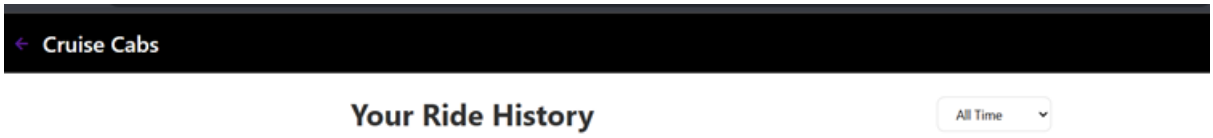
Log in


Register
First Name
Last Name
Email
Password
Sign Up
----------or--------
Already Have Account ?
Sign In

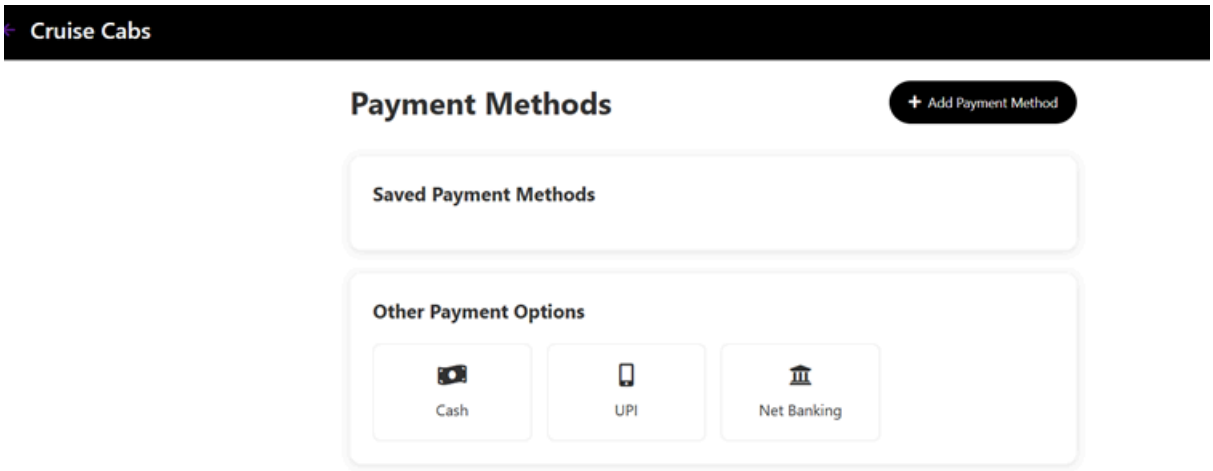
Profile



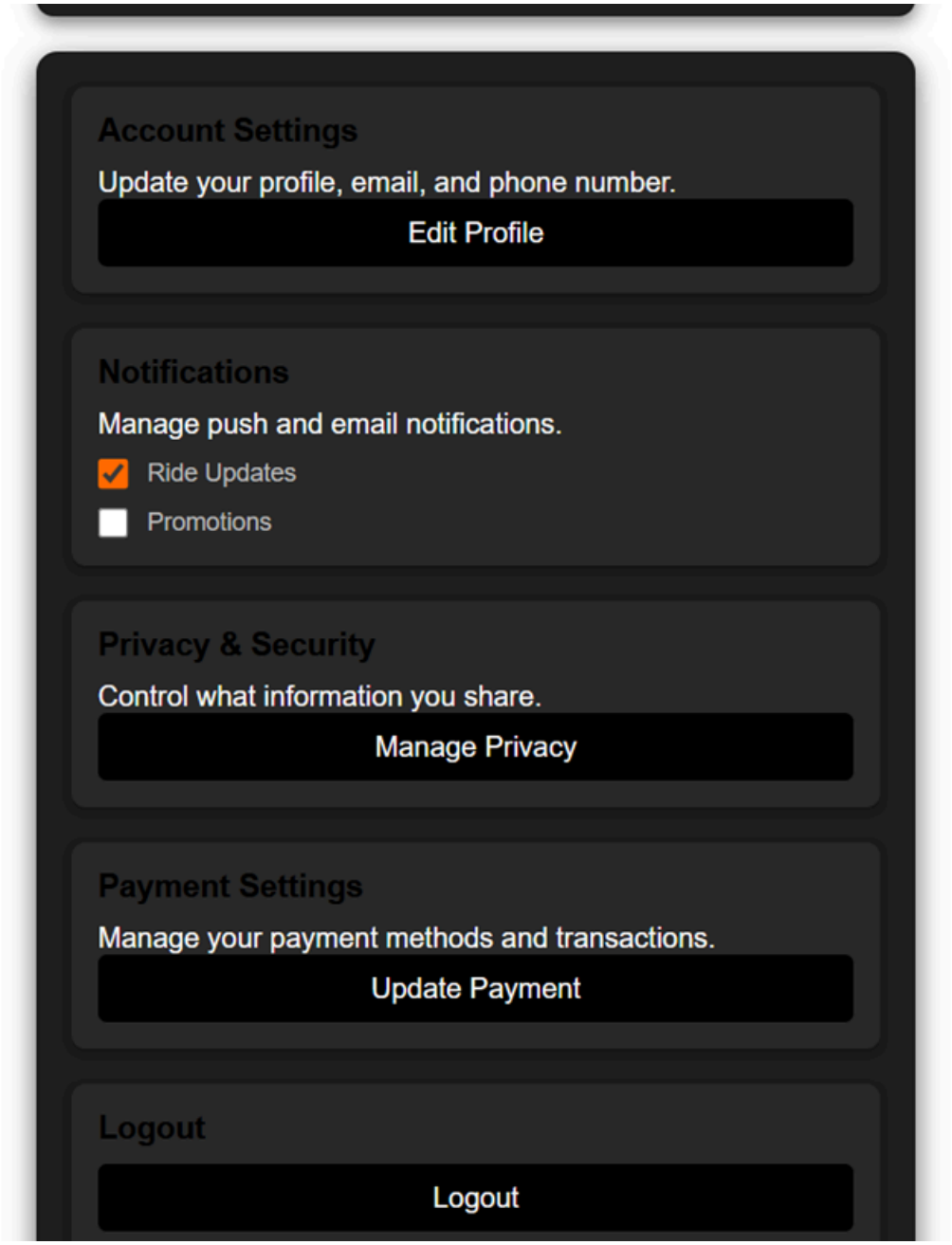
Trip History



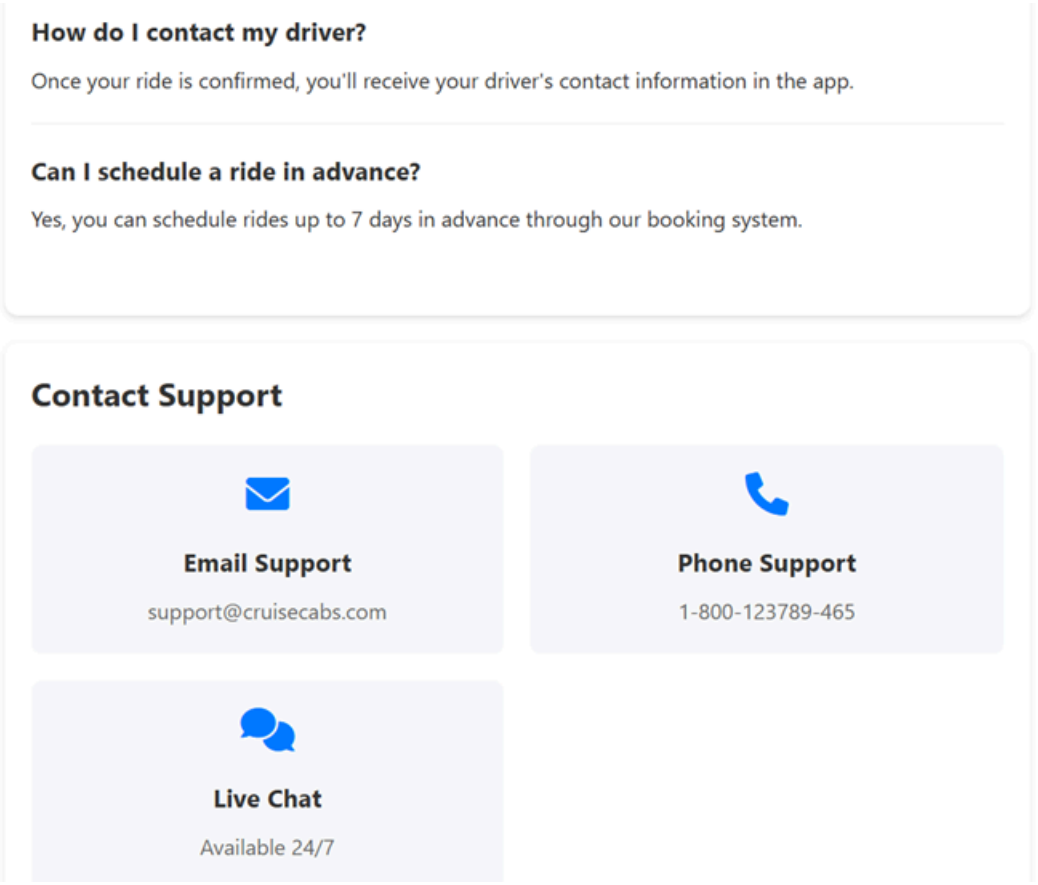
Payment Method


Cruise Cabs
Payment Methods
+ Add Payment Method
Saved Payment Methods
Other Payment Options
Cash
UPI
Net Banking

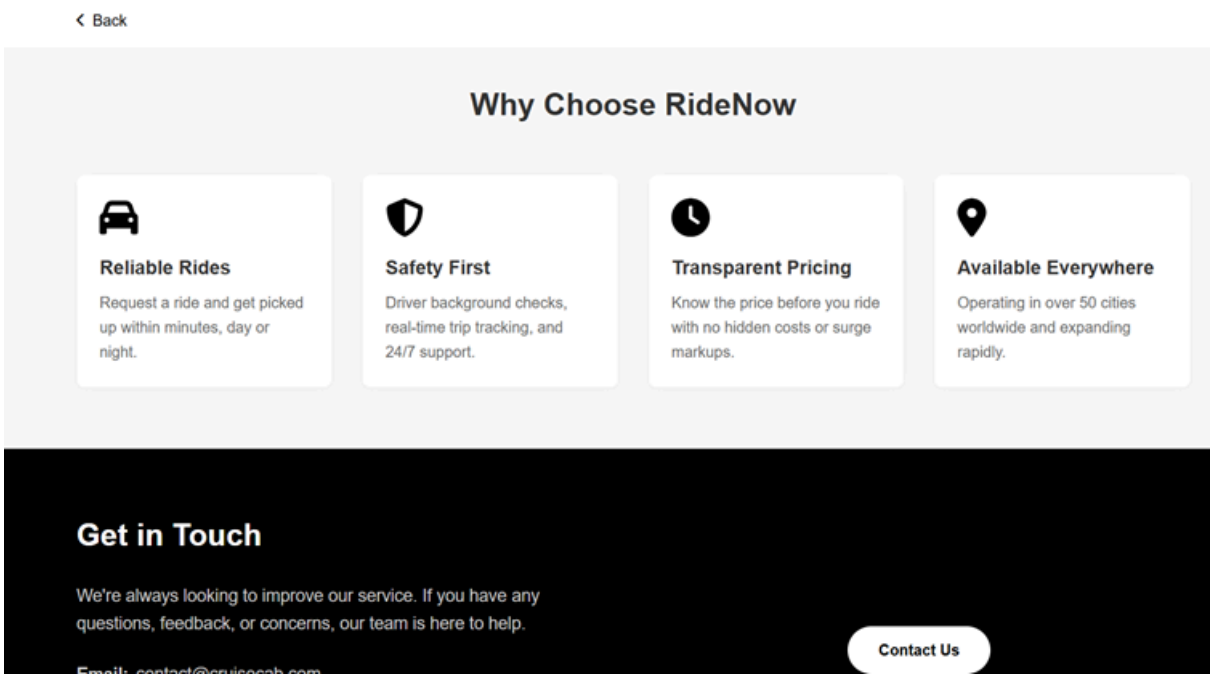
Settings


Account Settings
Update your profile, email, and phone number.
Edit Profile
Notifications
Manage push and email notifications.
Ride Updates
Promotions
Privacy & Security
Control what information you share.
Manage Privacy
Payment Settings
Manage your payment methods and transactions.
Update Payment
Logout
Logout

Help



About section



Code:

Html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cruise Cabs - Book a Ride</title>
  <link rel="stylesheet" href="styles.css">
```

```
    <link
href="https://fonts.googleapis.com/css2?family=UberMove:wght@400;500;700&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
    <script type="module" src="firebaseauth.js"></script>
    <link rel="icon" type="image/png" href="favicon.png">


</head>
<body>
    <!-- Add this right after the opening <body> tag -->
    <div class="sidebar" id="sidebar">
        <div class="sidebar-header">
            <button class="close-sidebar">
                <i class="fas fa-times"></i>
            </button>
        </div>
        <div class="sidebar-content">
            <div class="user-info">
                <i class="fas fa-user-circle user-avatar"></i>
                <div class="user-details">
                    <h3>Your Name</h3>
```

```
      <a href="profile.html" class="view-account-link">View account</a>
    </div>
  </div>
  <nav class="sidebar-nav">
    <div class="nav-section">
      <a href="ride-history.html" class="nav-item">
        <i class="fas fa-clock-rotate-left"></i>
        <span>Trip History</span>
      </a>
      <a href="payment-methods.html" class="nav-item">
        <i class="fas fa-credit-card"></i>
        <span>Payment Methods</span>
      </a>
      <a href="Settings.html" class="nav-item">
        <i class="fas fa-cog"></i>
        <span>Settings</span>
      </a>
    </div>
    <div class="nav-section">
      <a href="help.html" class="nav-item">
        <i class="fas fa-question-circle"></i>
        <span>Help</span>
```

```
            </a>
            <a href="about.html" class="nav-item">
              <i class="fas fa-info-circle"></i>
              <span>About</span>
            </a>
            <a href="#" class="nav-item logout">
              <i class="fas fa-sign-out-alt"></i>
              <span>Log out</span>
            </a>
          </div>
        </nav>
      </div>
    </div>

    <!-- Add overlay div for sidebar background -->
    <div class="sidebar-overlay" id="sidebar-overlay"></div>

    <!-- Header -->
    <header>
      <div class="header-left">
        <div class="menu-toggle" id="sidebar-toggle">
          <i class="fas fa-bars"></i>
```

```
    </div>
    <div class="logo">
      <span class="logo-text">Cruise Cabs</span>
    </div>
  </div>

  <nav class="header-nav">
    <div class="nav-item-group">
      <a href="#" class="nav-item active">Ride</a>
      <a href="#" class="nav-item">Business</a>
    </div>
  </nav>

  <div class="header-right">
    <a href="login.html" class="auth-nav-btn">Log in</a>
    <a href="signup2.html" class="auth-nav-btn signup">Sign up</a>
    <a href="profile.html" class="nav-button" id="profile-button">
      <i class="fas fa-bars"></i>
      <i class="fas fa-user-circle"></i>
    </a>
  </div>
</header>
```

```
<!-- Main Section -->
<main class="main-container">
  <div class="left-panel">
    <h1>Request a ride now</h1>
    <form id="booking-form" class="ride-form">
      <div class="location-input-group">
        <i class="fas fa-circle pickup-icon"></i>
        <input type="text" id="pickup-location" placeholder="Enter pickup location" required>
        <button type="button" class="map-btn" onclick="showMapModal('pickup')">
          <i class="fas fa-map-marker-alt"></i>
        </button>
      </div>
      <div class="location-input-group">
        <i class="fas fa-square destination-icon"></i>
        <input type="text" id="destination" placeholder="Enter destination" required>
        <button type="button" class="map-btn" onclick="showMapModal('destination')">
          <i class="fas fa-map-marker-alt"></i>
        </button>
      </div>
      <button type="submit" class="request-ride-btn">Request Now</button>
```

```
        <button id="scheduleLaterBtn">Schedule Later</button>

        <!-- Add a hidden date-time picker -->
        <input type="datetime-local" id="scheduleDateTime" style="display: none;">
            </form>
    </div>

    <div class="right-panel">
      <div id="map" class="main-map"></div>

    </div>
</main>

<!-- Ride Options Modal -->
<div id="ride-options" class="ride-options-modal">
    <div class="ride-options-content">
      <h3>Choose a ride</h3>
      <div class="ride-cards">
        <div class="ride-card" data-type="Bike">
          <i class="fas fa-motorcycle"></i>
          <div class="ride-details">
            <h4>Bike</h4>
```

```
      <p>Affordable bike rides</p>
      <span class="price">Rs 65.00</span>
    </div>
  </div>
  <div class="ride-card" data-type="Economy">
    <i class="fas fa-car"></i>
    <div class="ride-details">
      <h4>Economy</h4>
      <p>Affordable everyday rides</p>
      <span class="price">Rs 125.00</span>
    </div>
  </div>
  <div class="ride-card" data-type="Sedan">
    <i class="fas fa-car-side"></i>
    <div class="ride-details">
      <h4>Sedan</h4>
      <p>Comfortable rides with extra space</p>
      <span class="price">Rs 335.00</span>
    </div>
  </div>
  <div class="ride-card" data-type="Luxury">
    <i class="fas fa-car-alt"></i>
```

```
            <div class="ride-details">
              <h4>Luxury</h4>
              <p>Premium rides in high-end cars</p>
              <span class="price">Rs 550.00</span>
            </div>
          </div>
        </div>
      </div>
    </div>

<!-- Map Modal -->
<div id="map-modal" class="map-modal">
  <div class="map-modal-content">
    <span class="close-map">&times;</span>
    <div id="modal-map"></div>
    <button id="confirm-location" class="submit-btn">Confirm Location</button>
  </div>
</div>

<!-- Add this after the ride-options modal -->
<div id="ride-confirmation-modal" class="ride-confirmation-modal">
  <div class="confirmation-content">
```

```
<div class="driver-info">
  <div class="driver-header">
    <h3>Your ride is confirmed!</h3>
    <button class="close-confirmation">&times;</button>
  </div>
  <div class="driver-details">
    <div class="driver-avatar">
      <i class="fas fa-user-circle"></i>
    </div>
    <div class="driver-text">
      <h4 id="driver-name">Loading driver...</h4>
      <p id="car-details">Loading vehicle details...</p>
    </div>
  </div>
</div>
<div class="ride-info">
  <div class="location-info">
    <div class="pickup">
      <i class="fas fa-circle"></i>
      <p id="pickup-text">Loading pickup...</p>
    </div>
    <div class="destination">
```

```
                <i class="fas fa-square"></i>
                <p id="destination-text">Loading destination...</p>
              </div>
            </div>
            <div class="price-info">
              <h4>Fare</h4>
              <p id="ride-price">Loading price...</p>
            </div>
          </div>
          <button id="start-ride-btn" class="start-ride-btn">Start Ride</button>
        </div>
      </div>

    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
    <script src="script.js"></script>
</body>
</html>
```

CSS

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'UberMove', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}

body {
    background-color: #fff;
    color: #333;
    line-height: 1.5;
}

/* Main Container Layout */
.main-container {
    display: grid;
    grid-template-columns: 450px 1fr;
    height: calc(100vh - 70px);
    margin-top: 64px;
}
```

```
/* Left Panel Styles */
.left-panel {
    padding: 40px;
    background: #fff;
    overflow-y: auto;
}

.left-panel h1 {
    font-size: 36px;
    margin-bottom: 30px;
    font-weight: 700;
}

/* Form Styles */
.ride-form {
    display: flex;
    flex-direction: column;
    gap: 20px;
}

.location-input-group {
    position: relative;
```

```
    display: flex;
    align-items: center;
    background: #f6f6f6;
    border-radius: 8px;
    padding: 8px 16px;
}

.location-input-group input {
    flex: 1;
    border: none;
    background: transparent;
    padding: 12px;
    font-size: 16px;
    outline: none;
}

.pickup-icon, .destination-icon {
    font-size: 12px;
    color: #333;
}

.map-btn {
```

```
    background: none;
    border: none;
    color: #333;
    cursor: pointer;
    padding: 8px;
}

/* Button Styles */
.request-ride-btn, .schedule-btn {
    width: 100%;
    padding: 16px;
    border: none;
    border-radius: 8px;
    font-size: 16px;
    font-weight: 500;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.request-ride-btn {
    background-color: #000;
    color: #fff;
```

```
    margin-bottom: 12px;
}

.schedule-btn {
    background-color: #f6f6f6;
    color: #333;
}

.request-ride-btn:hover {
    background-color: #333;
}

.schedule-btn:hover {
    background-color: #eee;
}

/* Right Panel Map */
.right-panel {
    height: 100%;
}

.main-map {
```

```
    height: 100%;
    width: 100%;
}

/* Ride Options Modal */
.ride-options-modal {
    display: none;
    position: fixed;
    bottom: 0;
    left: 0;
    width: 100%;
    background: white;
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    box-shadow: 0 -2px 10px rgba(0,0,0,0.1);
    z-index: 1000;
    animation: slideUp 0.3s ease-out;
}

.ride-options-content {
    padding: 24px;
    max-width: 1200px;
```

```
    margin: 0 auto;
}

.ride-cards {
    display: flex;
    flex-direction: column;
    gap: 16px;
    margin-top: 20px;
}

.ride-card {
    display: flex;
    align-items: center;
    padding: 16px;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.2s ease;
}

.ride-card:hover {
    background-color: #f6f6f6;
}
```

```
.ride-card img {
  width: 64px;
  height: 64px;
  margin-right: 16px;
}

.ride-details {
  flex: 1;
}

.ride-details h4 {
  font-size: 18px;
  margin-bottom: 4px;
}

.ride-details p {
  color: #666;
  font-size: 14px;
}

.price {
```

```
    font-weight: 500;
    font-size: 18px;
}

/* Animations */
@keyframes slideUp {
    from {
        transform: translateY(100%);
    }
    to {
        transform: translateY(0);
    }
}

/* Responsive Design */
@media (max-width: 1024px) {
    .main-container {
        grid-template-columns: 1fr;
    }

    .right-panel {
        display: none;
```

```
  }
}

@media (max-width: 768px) {
  .left-panel {
    padding: 20px;
  }

  .left-panel h1 {
    font-size: 28px;
  }
}

/* Header Styles */
header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0 24px;
  height: 64px;
  background-color: #000;
  position: fixed;
```

```
    top: 0;
    left: 0;
    right: 0;
    z-index: 1000;
}

.header-left {
    display: flex;
    align-items: center;
    gap: 16px;
}

.logo-text {
    color: #fff;
    font-size: 24px;
    font-weight: 500;
    text-decoration: none;
}

.header-nav {
    position: absolute;
    left: 50%;
```

```
    transform: translateX(-50%);
}

.nav-item-group {
    display: flex;
    gap: 32px;
}

.nav-item {
    color: #fff;
    text-decoration: none;
    font-size: 16px;
    font-weight: 500;
    padding: 8px 0;
    position: relative;
}

.nav-item.active::after {
    content: '';
    position: absolute;
    bottom: 0;
    left: 0;
```

```
    width: 100%;
    height: 2px;
    background-color: #fff;
}

.nav-item:hover {
    color: #e5e5e5;
}

.header-right {
    display: flex;
    align-items: center;
    gap: 24px;
}

.nav-button {
    display: flex;
    align-items: center;
    gap: 8px;
    background: transparent;
    border: none;
    color: #fff;
```

```
    padding: 8px 12px;
    cursor: pointer;
    font-size: 16px;
}

.nav-button i {
    font-size: 20px;
}

.nav-button:hover {
    background-color: #333;
    border-radius: 24px;
}

/* Responsive adjustments */
@media (max-width: 768px) {
    .header-nav {
        display: none;
    }

    .header-right {
        gap: 16px;
```

```
    }
}

/* Sidebar Styles */
.sidebar {
    position: fixed;
    left: -320px;
    top: 0;
    width: 320px;
    height: 100vh;
    background-color: #fff;
    z-index: 2000;
    transition: left 0.3s ease;
    box-shadow: 2px 0 8px rgba(0,0,0,0.1);
}

.sidebar.active {
    left: 0;
}

.sidebar-overlay {
    position: fixed;
```

```
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0,0,0,0.5);
    z-index: 1999;
    display: none;
}

.sidebar-overlay.active {
    display: block;
}

.sidebar-header {
    padding: 20px;
    border-bottom: 1px solid #eee;
}

.close-sidebar {
    background: none;
    border: none;
    font-size: 24px;
```

```
    cursor: pointer;
    color: #333;
}

.user-info {
    padding: 24px 20px;
    display: flex;
    align-items: center;
    gap: 16px;
    border-bottom: 1px solid #eee;
}

.user-avatar {
    font-size: 48px;
    color: #333;
}

.user-details h3 {
    font-size: 18px;
    margin-bottom: 4px;
}
```

```
.user-details p {
    color: #666;
    font-size: 14px;
}

.sidebar-nav {
    padding: 16px 0;
}

.nav-section {
    padding: 8px 0;
    border-bottom: 1px solid #eee;
}

.nav-section:last-child {
    border-bottom: none;
}

.sidebar-nav .nav-item {
    display: flex;
    align-items: center;
    gap: 16px;
```

```
    padding: 16px 20px;
    color: #333;
    text-decoration: none;
    transition: background-color 0.2s ease;
}

.sidebar-nav .nav-item:hover {
    background-color: #f5f5f5;
}

.sidebar-nav .nav-item i {
    font-size: 20px;
    width: 24px;
    text-align: center;
}

.logout {
    color: #ff0000 !important;
}

/* Add these styles for the ride confirmation modal */
.ride-confirmation-modal {
```

```
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    z-index: 2000;
}

.confirmation-content {
    position: fixed;
    bottom: 0;
    left: 0;
    width: 100%;
    background: white;
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    padding: 24px;
    animation: slideUp 0.3s ease-out;
}
```

```
.driver-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

.close-confirmation {
    background: none;
    border: none;
    font-size: 24px;
    cursor: pointer;
}

.driver-details {
    display: flex;
    align-items: center;
    gap: 16px;
    padding: 16px;
    background: #f8f8f8;
    border-radius: 12px;
    margin-bottom: 20px;
```

```
}

.driver-avatar i {
    font-size: 48px;
}

.driver-text h4 {
    margin: 0;
    font-size: 18px;
}

.driver-text p {
    margin: 4px 0 0;
    color: #666;
}

.location-info {
    border-top: 1px solid #eee;
    border-bottom: 1px solid #eee;
    padding: 16px 0;
    margin-bottom: 16px;
}
```

```
.pickup, .destination {
    display: flex;
    align-items: center;
    gap: 12px;
    margin: 8px 0;
}

.price-info {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

.start-ride-btn {
    width: 100%;
    padding: 16px;
    background: #000;
    color: white;
    border: none;
    border-radius: 8px;
```

```
    font-size: 16px;
    cursor: pointer;
}

.start-ride-btn:hover {
    background: #333;
}

/* Add these styles for the view account link */
.view-account-link {
    color: #666;
    text-decoration: none;
    font-size: 14px;
    transition: color 0.2s ease;
}

.view-account-link:hover {
    color: #000;
    text-decoration: underline;
}

/* Auth navigation buttons */
```

```
.auth-nav-btn {
  text-decoration: none;
  padding: 8px 16px;
  margin-right: 12px;
  border-radius: 24px;
  font-size: 14px;
  font-weight: 500;
  transition: all 0.2s ease;
}

.auth-nav-btn:not(.signup) {
  color: #fff;
}

.auth-nav-btn.signup {
  background: #fff;
  color: #000;
}

.auth-nav-btn:hover {
  background: rgba(255,255,255,0.1);
}
```

```
.auth-nav-btn.signup:hover {
    background: rgba(255,255,255,0.9);
}

/* Help page styles */
.help-container {
    max-width: 800px;
    margin: 20px auto;
    padding: 0 20px;
}

.faq-section, .contact-section {
    background: white;
    border-radius: 10px;
    padding: 20px;
    margin-bottom: 20px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.faq-item {
    margin-bottom: 20px;
```

```
    padding-bottom: 20px;
    border-bottom: 1px solid #eee;
}

.faq-item:last-child {
    border-bottom: none;
}

.faq-item h3 {
    color: #333;
    margin-bottom: 10px;
}

.contact-options {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    margin-top: 20px;
}

.contact-card {
    text-align: center;
```

```
    padding: 20px;
    background: #f8f9fa;
    border-radius: 8px;
    transition: transform 0.2s;
}

.contact-card:hover {
    transform: translateY(-5px);
}

.contact-card i {
    font-size: 2em;
    color: #007bff;
    margin-bottom: 10px;
}

.contact-card h3 {
    margin: 10px 0;
    color: #333;
}

.contact-card p {
```

```
    color: #666;
}

/* Home page styles */
.home-container {
    max-width: 1200px;
    margin: 20px auto;
    padding: 0 20px;
}

.welcome-section {
    text-align: center;
    padding: 20px;
}

.quick-actions {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    margin-top: 30px;
}
```

```
.action-card {
    background: white;
    border-radius: 10px;
    padding: 20px;
    text-align: center;
    box-shadow: 0 2px 4px rgba(0,0,0)
}

/* Loading state for input */
.location-input-group input[readonly] {
    background-color: #f5f5f5;
    cursor: wait;
}

/* Style for the marker button when active */
.map-btn:active {
    background-color: #e0e0e0;
}

#scheduleLaterBtn {
    width: 100%;
    padding: 16px;
```

```
    border: none;
    border-radius: 8px;
    font-size: 16px;
    font-weight: 500;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

#scheduleLaterBtn:hover {
    background-color: #0056b3; /* Darker blue on hover */
}

.request-ride-btn, .schedule-btn {
    width: 100%;
    padding: 16px;
    border: none;
    border-radius: 8px;
    font-size: 16px;
    font-weight: 500;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
```

```
#scheduleLaterBtn {
    background-color: #000;
    color: #fff;
    margin-bottom: 12px;
}

#scheduleLaterBtn:hover {
    background-color: #333;
}
```

Js :

```
window.addEventListener("load", function() {
    let preloader = document.getElementById("preloader");
    let mainContent = document.getElementById("main-content");

    // Fade out effect
    setTimeout(() => {
        preloader.style.animation = "fadeOut 1s ease-out";

        // Remove preloader after animation
        setTimeout(() => {
```

```
            preloader.style.display = "none";
            mainContent.style.display = "block";
        }, 1000);
    }, 4000); // Changed to 4000ms (4 seconds) before starting fade out
});

// Add sidebar toggle functionality
document.addEventListener('DOMContentLoaded', function() {
    const sidebarToggle = document.getElementById('sidebar-toggle');
    const sidebar = document.getElementById('sidebar');

    sidebarToggle.addEventListener('click', function() {
        sidebar.classList.toggle('active');
    });

    // Close sidebar when clicking outside
    document.addEventListener('click', function(event) {
        if (!sidebar.contains(event.target) && !sidebarToggle.contains(event.target)) {
            sidebar.classList.remove('active');
        }
    });
});
```

```
// Add these variables at the top
const BASE_PRICES = {
    'Bike': 15,
    'Economy': 25,
    'Sedan': 35,
    'Luxury': 50
};

const PRICE_PER_KM = {
    'Bike': 2,
    'Economy': 3,
    'Sedan': 4,
    'Luxury': 6
};

// Function to calculate distance between two points using Haversine formula
function calculateDistance(lat1, lon1, lat2, lon2) {
    const R = 6371; // Earth's radius in km
    const dLat = (lat2 - lat1) * Math.PI / 180;
    const dLon = (lon2 - lon1) * Math.PI / 180;
    const a = Math.sin(dLat/2) * Math.sin(dLat/2) +
```

```
            Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
            Math.sin(dLon/2) * Math.sin(dLon/2);
    const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
    return R * c; // Distance in km
}

// Function to update prices based on distance
function updatePrices(distance) {
    Object.keys(BASE_PRICES).forEach(rideType => {
        const basePrice = BASE_PRICES[rideType];
        const pricePerKm = PRICE_PER_KM[rideType];
        const totalPrice = (basePrice + (distance * pricePerKm)).toFixed(2);

        // Find the price element for this ride type and update it
        const priceElement = document.querySelector(`.ride-card:has(h4:contains('${rideType}')) .price`);
        if (priceElement) {
            priceElement.textContent = `CHF ${totalPrice}`;
        }
    });
}

// Modify the form submission handler
```

```
document.addEventListener('DOMContentLoaded', function() {
    const bookingForm = document.getElementById('booking-form');
    const rideOptions = document.getElementById('ride-options');

    bookingForm.addEventListener('submit', function(e) {
        e.preventDefault();
        // Show ride options
        rideOptions.style.display = 'block';
        // Smooth scroll to ride options
        rideOptions.scrollIntoView({ behavior: 'smooth' });
    });

    // Add click handlers for ride selection
    const selectButtons = document.querySelectorAll('.select-ride');
    selectButtons.forEach(button => {
        button.addEventListener('click', function() {
            const rideType = this.textContent.replace('Select ', '');
            alert(`You have selected ${rideType}. Proceeding to confirmation...`);
            // Here you can add code to proceed to the next step (payment, confirmation, etc.)
        });
    });
});
```

```
let map;
let pickupMarker = null;
let destinationMarker;
let currentField;
let directionsLine;
let modalMap = null;

// Initialize map functionality
document.addEventListener('DOMContentLoaded', function() {
    // Initialize main map
    map = L.map('map').setView([47.3769, 8.5417], 13);
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '© OpenStreetMap contributors'
    }).addTo(map);

    // Initialize form handlers
    const bookingForm = document.getElementById('booking-form');
    const rideOptions = document.getElementById('ride-options');
    const pickupInput = document.getElementById('pickup-location');
    const destinationInput = document.getElementById('destination');
```

```
// Get user's current location
if ("geolocation" in navigator) {
    navigator.geolocation.getCurrentPosition(function(position) {
        const userLat = position.coords.latitude;
        const userLng = position.coords.longitude;

        map.setView([userLat, userLng], 15);

        // Set pickup location
        reverseGeocode(userLat, userLng, pickupInput);
        if (pickupMarker) map.removeLayer(pickupMarker);
        pickupMarker = L.marker([userLat, userLng]).addTo(map);
    });
}

// Handle form submission
bookingForm.addEventListener('submit', async function(e) {
    e.preventDefault();

    if (pickupMarker && destinationMarker) {
        // Calculate route
        const distance = calculateDistance(
```

```
            pickupMarker.getLatLng().lat,
            pickupMarker.getLatLng().lng,
            destinationMarker.getLatLng().lat,
            destinationMarker.getLatLng().lng
        );

        // Update prices based on distance
        updatePrices(distance);

        // Show ride options
        rideOptions.style.display = 'block';

        // Draw route line
        drawRoute(pickupMarker.getLatLng(), destinationMarker.getLatLng());
    }
  });

  // Add input listeners for location search
  setupLocationSearch(pickupInput, 'pickup');
  setupLocationSearch(destinationInput, 'destination');
});
```

```
// Location search setup
function setupLocationSearch(input, type) {
    let timeout = null;

    input.addEventListener('input', function() {
        if (timeout) clearTimeout(timeout);

        timeout = setTimeout(() => {
            const query = this.value;
            if (query.length > 2) {
                searchLocation(query, type);
            }
        }, 300);
    });
}

// Search location using Nominatim
async function searchLocation(query, type) {
    try {
        const response = await fetch(

`https://nominatim.openstreetmap.org/search?format=json&q=${encodeURIComponent(query)}`
```

```
    );
    const data = await response.json();

    if (data.length > 0) {
      const location = data[0];
      const latlng = [parseFloat(location.lat), parseFloat(location.lon)];

      if (type === 'pickup') {
        if (pickupMarker) map.removeLayer(pickupMarker);
        pickupMarker = L.marker(latlng).addTo(map);
      } else {
        if (destinationMarker) map.removeLayer(destinationMarker);
        destinationMarker = L.marker(latlng).addTo(map);
      }

      map.setView(latlng, 15);

      if (pickupMarker && destinationMarker) {
        drawRoute(pickupMarker.getLatLng(), destinationMarker.getLatLng());
      }
    }
  } catch (error) {
```

```
        console.error('Error searching location:', error);
    }
}

// Draw route between two points
function drawRoute(start, end) {
    if (directionsLine) map.removeLayer(directionsLine);

    directionsLine = L.polyline([start, end], {
        color: 'black',
        weight: 3,
        opacity: 0.7
    }).addTo(map);

    map.fitBounds(directionsLine.getBounds(), {
        padding: [50, 50]
    });
}

// Reverse geocode coordinates to address
async function reverseGeocode(lat, lon, input) {
    try {
```

```
      const response = await fetch(
        `https://nominatim.openstreetmap.org/reverse?format=json&lat=${lat}&lon=${lon}`
      );
      const data = await response.json();
      input.value = data.display_name;
    } catch (error) {
      console.error('Error reverse geocoding:', error);
    }
}

// Add this to your existing JavaScript
document.addEventListener('DOMContentLoaded', function() {
    const sidebar = document.getElementById('sidebar');
    const sidebarOverlay = document.getElementById('sidebar-overlay');
    const sidebarToggle = document.getElementById('sidebar-toggle');
    const closeSidebar = document.querySelector('.close-sidebar');

    // Open sidebar
    sidebarToggle.addEventListener('click', function() {
        sidebar.classList.add('active');
        sidebarOverlay.classList.add('active');
        document.body.style.overflow = 'hidden'; // Prevent scrolling when sidebar is open
```

```
});


// Close sidebar function
function closeSidebarMenu() {
    sidebar.classList.remove('active');
    sidebarOverlay.classList.remove('active');
    document.body.style.overflow = '';
}


// Close sidebar when clicking close button
closeSidebar.addEventListener('click', closeSidebarMenu);


// Close sidebar when clicking overlay
sidebarOverlay.addEventListener('click', closeSidebarMenu);


// Close sidebar when pressing Escape key
document.addEventListener('keydown', function(e) {
    if (e.key === 'Escape' && sidebar.classList.contains('active')) {
        closeSidebarMenu();
    }
});
```

```
// Profile navigation
const profileButton = document.getElementById('profile-button');
if (profileButton) {
    profileButton.addEventListener('click', function(e) {
        e.preventDefault(); // Prevent default anchor behavior
        window.location.href = 'profile.html';
    });
}

// Add help navigation
const helpButton = document.getElementById('help-button');
if (helpButton) {
    helpButton.addEventListener('click', function(e) {
        e.preventDefault();
        window.location.href = 'help.html';
    });
}

// Ride selection functionality
const rideCards = document.querySelectorAll('.ride-card');
const confirmRideBtn = document.querySelector('.confirm-ride-btn');
```

```
rideCards.forEach(card => {
    card.addEventListener('click', function() {
        // Remove selected class from all cards
        rideCards.forEach(c => c.classList.remove('selected'));
        // Add selected class to clicked card
        this.classList.add('selected');

        const rideType = this.getAttribute('data-type');
        const price = this.querySelector('.price').textContent;
        const pickup = document.getElementById('pickup-location').value;
        const destination = document.getElementById('destination').value;

        // Create and store current ride
        currentRide = createRide(rideType, price, pickup, destination);
    });
});

if (confirmRideBtn) {
    confirmRideBtn.addEventListener('click', function() {
        const selectedRide = document.querySelector('.ride-card.selected');
        if (selectedRide && currentRide) {
            // Add ride to history
```

```
            rideHistory.unshift(currentRide); // Add to beginning of array
            localStorage.setItem('rideHistory', JSON.stringify(rideHistory));

            // Show confirmation
            alert(`Booking confirmed!\nRide Type: ${currentRide.type}\nPrice: ${currentRide.price}`);

            // Update ride history in profile if we're on the profile page
            updateRideHistory();

            // Hide ride options
            document.getElementById('ride-options').style.display = 'none';

            // Reset current ride
            currentRide = null;
        } else {
            alert('Please select a ride type');
        }
    });
  }
});

// Function to update ride history display
```

```
function updateRideHistory() {
    const tripsList = document.querySelector('.trips-list');
    if (tripsList) {
        tripsList.innerHTML = ''; // Clear existing trips

        // Get latest rides from localStorage
        const rides = JSON.parse(localStorage.getItem('rideHistory')) || [];

        // Display latest 5 rides
        rides.slice(0, 5).forEach(ride => {
            const date = new Date(ride.date);
            const formattedDate = date.toLocaleDateString('en-US', {
                month: 'short',
                day: 'numeric',
                hour: '2-digit',
                minute: '2-digit'
            });

            const tripHtml = `
                <div class="trip-item">
                    <div class="trip-icon">
                        <i class="fas fa-car"></i>
```

```
            </div>
            <div class="trip-details">
              <h3>${ride.pickup} to ${ride.destination}</h3>
              <p>${formattedDate}</p>
              <span class="trip-price">${ride.price}</span>
            </div>
          </div>
        `;

        tripsList.insertAdjacentHTML('beforeend', tripHtml);
      });
    }
}

// Call this when the profile page loads
if (window.location.pathname.includes('profile.html')) {
    updateRideHistory();
}

/*****************************************
 * GLOBAL VARIABLES AND UTILITIES
 *****************************************/
```

```
// Store user and ride data
let rideHistory = JSON.parse(localStorage.getItem('rideHistory')) || [];
let currentRide = null;

// Arrays for random driver generation
const driverNames = [
    "John Smith", "Maria Garcia", "David Chen", "Sarah Johnson",
    "Michael Brown", "Emma Wilson", "James Taylor", "Lisa Anderson"
];

const carModels = [
    "Toyota Camry", "Honda Civic", "Tesla Model 3", "BMW 3 Series",
    "Mercedes C-Class", "Audi A4", "Volkswagen Passat", "Hyundai Sonata"
];

/*****************************************
 * UTILITY FUNCTIONS
 *****************************************/
// Format date for display
function formatDate(dateString) {
    const date = new Date(dateString);
    return date.toLocaleDateString('en-US', {
```

```
      month: 'short',
      day: 'numeric',
      hour: '2-digit',
      minute: '2-digit'
    });
}

// Generate random car number
function generateCarNumber() {
    const letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    const numbers = "0123456789";
    let carNumber = "";

    carNumber += letters[Math.floor(Math.random() * letters.length)];
    carNumber += letters[Math.floor(Math.random() * letters.length)];
    carNumber += "-";
    for(let i = 0; i < 4; i++) {
        carNumber += numbers[Math.floor(Math.random() * numbers.length)];
    }

    return carNumber;
}
```

```
/****************************************
 * AUTHENTICATION FUNCTIONS
 ****************************************/

// Handle user login
function handleLogin(email, password) {
    const storedUser = JSON.parse(localStorage.getItem('userData'));
    if (storedUser && storedUser.email === email && storedUser.password === password) {
        localStorage.setItem('isLoggedIn', 'true');
        window.location.href = 'Homepage.html';
    } else {
        alert('Invalid credentials');
    }
}


// Handle user signup
function handleSignup(userData) {
    localStorage.setItem('userData', JSON.stringify(userData));
    localStorage.setItem('isLoggedIn', 'true');
    alert('Account created successfully! Welcome to Cruise Cabs!');
    window.location.href = 'Homepage.html';
}
```

```
/****************************************
 * RIDE MANAGEMENT FUNCTIONS
 ****************************************/
// Create new ride
function createRide(rideType, price, pickup, destination) {
    const driverName = driverNames[Math.floor(Math.random() * driverNames.length)];
    const carModel = carModels[Math.floor(Math.random() * carModels.length)];
    const carNumber = generateCarNumber();

    return {
        id: Date.now(),
        type: rideType,
        price: price,
        pickup: pickup,
        destination: destination,
        date: new Date().toISOString(),
        status: 'Active',
        driver: {
            name: driverName,
            car: carModel,
            carNumber: carNumber
```

```
        }
    };
}

// Update ride history display
function updateRideHistory(filter = 'all') {
    const ridesList = document.querySelector('.rides-list');
    const noRides = document.querySelector('.no-rides');

    if (!ridesList) return;

    let rides = JSON.parse(localStorage.getItem('rideHistory')) || [];

    // Apply time filter
    if (filter !== 'all') {
        const now = new Date();
        const filterDate = new Date();
        if (filter === 'month') {
            filterDate.setMonth(filterDate.getMonth() - 1);
        } else if (filter === 'week') {
            filterDate.setDate(filterDate.getDate() - 7);
        }
```

```
        rides = rides.filter(ride => new Date(ride.date) > filterDate);
    }

    if (rides.length === 0) {
        ridesList.style.display = 'none';
        noRides.style.display = 'block';
    } else {
        ridesList.style.display = 'block';
        noRides.style.display = 'none';
        ridesList.innerHTML = rides.map(createRideCard).join('');
    }
}

/*****************************************
 * PAYMENT FUNCTIONS
 *****************************************/
// Format card input
function formatCardNumber(input) {
    let value = input.value.replace(/\s+/g, '').replace(/[^0-9]/gi, '');
    let formattedValue = '';
    for (let i = 0; i < value.length; i++) {
```

```
        if (i > 0 && i % 4 === 0) {
            formattedValue += ' ';
        }
        formattedValue += value[i];
    }
    input.value = formattedValue;
}

// Format expiry date
function formatExpiry(input) {
    let value = input.value.replace(/\s+/g, '').replace(/[^0-9]/gi, '');
    if (value.length > 2) {
        value = value.slice(0, 2) + '/' + value.slice(2);
    }
    input.value = value;
}

/*****************************************
 * UI COMPONENT FUNCTIONS
 *****************************************/
// Create ride card HTML
function createRideCard(ride) {
```

```
return `
    <div class="ride-card">
        <div class="ride-header">
            <span class="ride-date">${formatDate(ride.date)}</span>
            <span class="ride-price">${ride.price}</span>
        </div>
        <div class="ride-details">
            <div class="location-markers">
                <div class="location-marker"></div>
                <div class="location-marker destination"></div>
            </div>
            <div class="locations">
                <div class="location-text">${ride.pickup}</div>
                <div class="location-text">${ride.destination}</div>
            </div>
        </div>
        <div class="ride-info">
            <div class="info-item">
                <i class="fas fa-car"></i>
                <span>${ride.type}</span>
            </div>
            <div class="info-item">
```

```
            <i class="fas fa-user"></i>
            <span>${ride.driver.name}</span>
          </div>
          <div class="info-item">
            <i class="fas fa-tag"></i>
            <span>${ride.driver.car} • ${ride.driver.carNumber}</span>
          </div>
        </div>
      </div>
    `;
}

// Create saved card element
function createSavedCardElement(card) {
    return `
      <div class="saved-card" data-card-id="${card.id}">
        <i class="fas fa-credit-card card-icon"></i>
        <div class="card-details">
          <div class="card-number">•••• •••• •••• ${card.number.slice(-4)}</div>
          <div class="card-expiry">Expires ${card.expiry}</div>
        </div>
        <i class="fas fa-trash delete-card"></i>
```

```
        </div>
    `;
}

/*****************************************
 * EVENT LISTENERS
 *****************************************/
document.addEventListener('DOMContentLoaded', function() {
    // Check login status and update UI
    checkLoginStatus();

    // Handle signup form
    const signupForm = document.getElementById('signup-form');
    if (signupForm) {
        signupForm.addEventListener('submit', handleSignupSubmit);
    }

    // Handle login form
    const loginForm = document.getElementById('login-form');
    if (loginForm) {
        loginForm.addEventListener('submit', handleLoginSubmit);
    }
```

```
// Handle payment form
const paymentForm = document.getElementById('payment-form');
if (paymentForm) {
    setupPaymentFormListeners(paymentForm);
}

// Handle ride selection
const rideCards = document.querySelectorAll('.ride-card');
if (rideCards.length) {
    setupRideCardListeners(rideCards);
}

// Update ride history if on history page
if (window.location.pathname.includes('ride-history.html')) {
    updateRideHistory();
}

// Add this after your other DOMContentLoaded event listeners
const scheduleLaterBtn = document.getElementById('schedule-later-btn');

if (scheduleLaterBtn) {
```

```
    scheduleLaterBtn.addEventListener('click', function(e) {
      e.preventDefault();
      showScheduleModal();
    });
  }
});

/*****************************************
 * EVENT HANDLER FUNCTIONS
 *****************************************/
function handleSignupSubmit(e) {
  e.preventDefault();
  const userData = {
    name: document.getElementById('fullname').value,
    email: document.getElementById('email').value,
    phone: document.getElementById('phone').value,
    password: document.getElementById('password').value,
    dateJoined: new Date().toISOString()
  };
  handleSignup(userData);
}
```

```
function handleLoginSubmit(e) {
    e.preventDefault();
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    handleLogin(email, password);
}

// Sign out functionality
document.addEventListener('DOMContentLoaded', function() {
    const signoutBtn = document.getElementById('signout-btn');

    if (signoutBtn) {
        signoutBtn.addEventListener('click', function(e) {
            e.preventDefault();

            // Clear any stored user data/tokens
            localStorage.removeItem('userToken');
            localStorage.removeItem('userData');
            sessionStorage.clear();

            // Optional: Show a success message
            alert('Successfully signed out!');
```

```
        // Redirect to login page
        window.location.href = 'login.html';
      });
    }
});

// ... Additional event handler functions ...

// Function to get current location
function getCurrentLocation(inputField) {
  if ("geolocation" in navigator) {
    // Show loading state
    inputField.value = "Getting location...";

    navigator.geolocation.getCurrentPosition(
      function(position) {
        // Get coordinates
        const latitude = position.coords.latitude;
        const longitude = position.coords.longitude;

        // Reverse geocode to get address
```

```
fetch(`https://nominatim.openstreetmap.org/reverse?lat=${latitude}&lon=${longitude}&format=json
`)
        .then(response => response.json())
        .then(data => {
          // Update input field with address
          inputField.value = data.display_name;

          // If map is initialized, update marker
          if (map) {
            // Remove existing marker if any
            if (pickupMarker) {
              map.removeLayer(pickupMarker);
            }

            // Add new marker
            pickupMarker = L.marker([latitude, longitude]).addTo(map);

            // Center map on location
            map.setView([latitude, longitude], 15);
          }
        })
```

```
            .catch(error => {
                console.error('Error getting address:', error);
                inputField.value = `${latitude}, ${longitude}`;
            });
        },
        function(error) {
            // Handle errors
            console.error('Error getting location:', error);
            inputField.value = "";
            alert("Unable to get your location. Please check your location permissions.");
        }
      );
    } else {
      alert("Geolocation is not supported by your browser");
    }
}

// Update the location input groups event listeners
document.addEventListener('DOMContentLoaded', function() {
    const pickupInput = document.getElementById('pickup-location');
    const destinationInput = document.getElementById('destination');
```

```
  // Add click handlers for the map buttons
  document.querySelectorAll('.map-btn').forEach(button => {
    button.addEventListener('click', function(e) {
      e.preventDefault();
      const inputField = this.previousElementSibling; // Get the input field before the button
      if (inputField.id === 'pickup-location') {
        getCurrentLocation(inputField);
      } else {
        // For destination, still show the map modal
        showMapModal('destination');
      }
    });
  });
});

// Update showMapModal function to work with the new functionality
function showMapModal(type) {
  if (type === 'destination') {
    const mapModal = document.getElementById('map-modal');
    mapModal.style.display = 'block';

    // Initialize modal map if not already done
```

```
        if (!modalMap) {
            modalMap = L.map('modal-map').setView([0, 0], 13);
            L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
                attribution: '© OpenStreetMap contributors'
            }).addTo(modalMap);
        }

        // Update modal map size after it's visible
        setTimeout(() => {
            modalMap.invalidateSize();
        }, 100);
    }
}

// Add these new functions
function showScheduleModal() {
    // Create modal HTML
    const modalHTML = `
        <div id="schedule-modal" class="modal">
            <div class="modal-content">
                <span class="close-modal">&times;</span>
                <h2>Schedule Your Ride</h2>
```

```
        <div class="schedule-form">
          <div class="form-group">
            <label for="schedule-date">Date:</label>
            <input type="date" id="schedule-date" min="${getTodayDate()}" required>
          </div>
          <div class="form-group">
            <label for="schedule-time">Time:</label>
            <input type="time" id="schedule-time" required>
          </div>
          <button class="confirm-schedule-btn">Confirm Schedule</button>
        </div>
      </div>
    </div>
`;

// Add modal to body
document.body.insertAdjacentHTML('beforeend', modalHTML);

const modal = document.getElementById('schedule-modal');
const closeBtn = modal.querySelector('.close-modal');
const confirmBtn = modal.querySelector('.confirm-schedule-btn');
```

```
// Show modal
modal.style.display = 'block';

// Close modal functionality
closeBtn.onclick = function() {
    modal.remove();
}

// Close when clicking outside
window.onclick = function(event) {
    if (event.target === modal) {
        modal.remove();
    }
}

// Handle schedule confirmation
confirmBtn.addEventListener('click', function() {
    const date = document.getElementById('schedule-date').value;
    const time = document.getElementById('schedule-time').value;

    if (date && time) {
        const scheduledDateTime = new Date(`${date} ${time}`);
```

```
        if (scheduledDateTime > new Date()) {
            // Store scheduled time
            localStorage.setItem('scheduledRide', JSON.stringify({
                datetime: scheduledDateTime.toISOString(),
                pickup: document.getElementById('pickup-location').value,
                destination: document.getElementById('destination').value
            }));

            alert(`Ride scheduled for ${scheduledDateTime.toLocaleString()}`);
            modal.remove();

            // Show ride options
            document.getElementById('ride-options').style.display = 'block';
        } else {
            alert('Please select a future date and time');
        }
    } else {
        alert('Please select both date and time');
    }
  });
}
```

```
function getTodayDate() {
    const today = new Date();
    const year = today.getFullYear();
    const month = String(today.getMonth() + 1).padStart(2, '0');
    const day = String(today.getDate()).padStart(2, '0');
    return `${year}-${month}-${day}`;
}

document.addEventListener("DOMContentLoaded", function () {
    const scheduleBtn = document.getElementById("scheduleLaterBtn");
    const dateTimeInput = document.getElementById("scheduleDateTime");

    scheduleBtn.addEventListener("click", function () {
        dateTimeInput.style.display = "block"; // Show the date-time picker
        dateTimeInput.focus(); // Focus on it so the user can select a date
    });

    dateTimeInput.addEventListener("change", function () {
        alert("You have scheduled a ride for: " + dateTimeInput.value);
        dateTimeInput.style.display = "none"; // Hide after selection
    });
});
```

```
const signUpButton=document.getElementById('signUpButton');
const signInButton=document.getElementById('signInButton');
const signInForm=document.getElementById('signIn');
const signUpForm=document.getElementById('signup');


signUpButton.addEventListener('click',function(){
   signInForm.style.display="none";
   signUpForm.style.display="block";
})
signInButton.addEventListener('click', function(){
   signInForm.style.display="block";
   signUpForm.style.display="none";
})
```