PHASE 1

ABAAN NOOR 24911
SAAD THAPLAWALA 27172
SAMEED AHMAD 26956

Problem Statement

Introduction :

The current system for managing carpools among IBA students is largely manual and inefficient. Students seeking carpool options rely on informal communication methods, such as word of mouth or social media platforms like WhatsApp and Facebook groups. These methods are **unorganized**, **time-consuming**, and lack centralized data storage, making it difficult to coordinate rides, verify participants, and ensure safety. Additionally, there is no structured way to handle cost-sharing or provide feedback on the quality of rides. This fragmented process not only limits the reach of carpooling but also discourages potential users due to concerns over **reliability** and **convenience**.

Identified Issues:

The limitations of the current system include:

1. **Inefficiency**: The manual process of finding or offering rides is time-consuming and prone to errors. Without a centralized platform, matching drivers with passengers is inconsistent and often unreliable.

2. **Lack of Data Organization**: There is no central database to store and manage information about rides, drivers, and passengers. This leads to data redundancy and makes it difficult to track ride history, availability, and user behavior.

3. **Security and Trust Concerns**: In the absence of user verification or profiling, students may feel uncertain about traveling with strangers. This lack of trust and accountability discourages broader participation.

4. **Difficulty in Cost Sharing**: No formal method exists to split travel costs fairly, leading to potential disputes or financial burdens falling unevenly on drivers.

5. **Limited Accessibility**: Students with different schedules and commute preferences face challenges in finding suitable carpooling options due to the lack of a dedicated matching system.

Need for a Database System:

Improved Data Management: A **well-designed** database organizes and stores user profiles, rides, and history efficiently, ensuring easy access, updating, and retrieval, reducing errors and confusion.
Enhanced Decision-Making: The database allows analysis of usage patterns and user behavior, helping users and administrators make **informed** decisions, like identifying peak ride times or popular routes.
Elimination of Redundancy: The database prevents **duplicate** data entries, ensuring data integrity and reducing storage waste by maintaining single records for users and rides.
Streamlining Operations: The database automates key processes like **authentication**, ride matching, and notifications, improving efficiency and speeding up operations.

- **User Registration and Authentication:** The database ensures that only IBA students, faculty, staff can register by verifying their **credentials**, securing sensitive user data (e.g., passwords), and managing account information.
- **Ride Listings & Notifications:** When a driver creates a ride listing, the system stores this information in the database, making it accessible to users looking for rides. Notifications related to ride confirmations and updates are triggered based on database entries and updates.
- **Ride History Tracking:** Both drivers and passengers will have their rides tracked, creating a record that can be accessed later, adding a layer of security and accountability.*****isko implement karna muskhil hosakta hai so remove kardena chahiya**

By leveraging a database system, the carpool app can offer a reliable, efficient, and user-friendly platform that meets the needs of IBA students while promoting sustainable travel solutions.

Proposed Solution:

The Carpool for IBA Students project aims to build an efficient, secure, and user friendly platform that allows IBA students to organize and participate in carpools.

1. Ride Matching and Search Functionality :

The system will offer a **search** functionality that allows students to find carpool options based on their starting point, destination, and schedule. A central database will store details about drivers and available seats, making it easy for passengers to find rides that align with their commuting needs.
Example: For IBA students, this feature will save time by automating the process of finding potential carpools, ensuring students can easily match with others traveling from nearby areas.

2. User Authentication and Profile Verification:

the system will implement a user authentication process, requiring students to sign up with their IBA credentials (e.g., student ID or email address).
Example: Similar to **Uber** where drivers and passengers are required to create verified profiles, our platform will build trust within the student community.Verified profiles will include details such as the user's name, picture, and carpooling history, allowing students to make informed decisions about whom to travel with. This will help reduce concerns about safety and create a more reliable carpooling environment.

3. User Ratings and Feedback System:

The platform will include a rating system where both drivers and passengers can rate each other after a trip. This feature will help build accountability, ensuring that users maintain a certain level of professionalism and safety while participating in carpools. Consistently low ratings will lead to warnings or temporary suspension from the system.
Example: If a driver is punctual and provides a safe, comfortable ride, they will receive **positive** ratings, which will encourage others to carpool with them in the future. Conversely, if a passenger cancels at the last minute without notice, the driver can leave feedback to help future carpoolers make informed choices.

4. Cost Sharing and Ride Optimization:

The system will include a cost-sharing feature, allowing drivers and passengers to split fuel costs. This will not only incentivize students to participate as drivers but also make carpooling a more affordable option for passengers.

Example: Platforms like **SPLT**, a corporate carpool service, provide ride cost-splitting to ensure that the burden of fuel expenses is shared fairly.

5. Scalability and Future Expansion:
While the initial platform will focus solely on IBA students, the system can be easily scaled to accommodate more users or extended to other universities or campuses within Karachi. Example: **Zimride**, which started as a university-focused carpool platform, gradually expanded to serve corporate clients and large communities. Similarly, the carpool platform for IBA students can scale up to serve other educational institutions or even city-wide commuters, offering a much broader impact on transportation solutions.

Functional Requirements

Core Functionalities & Detailed Descriptions

1) User Registration and Authentication

The platform ensures only authorized IBA students can access the system by validating their credentials during the signup process.

Features:

User Authentication:

- Every person needs to sign up using their IBA credentials to ensure the platform remains exclusive to IBA community. This will prevent unauthorized users and maintain a secure environment.
- Email verification via OTP (One-Time Password) or confirmation link ensures authenticity.
- Password creation will contain security policies e.g., minimum length, special characters).

Data Validation:

- Email format validation i.e email must end with `@khi.iba.edu.pk`
- Verification of ID to ensure only valid person can access the platform.

Permissions:

- Only registered person can access the core features (carpools, profiles, etc.).
- Admins can manage user roles (e.g., deactivate fraudulent accounts or ban users).

- **Example**: When a person registers, they provide their ID and IBA email, which the system verifies. Once authenticated, the user gains access to the platform, including their profile and available carpools.

2) Profile Management

Each user has a profile where they can input their commuting details, such as location, preferred pickup/drop-off points, travel schedule, and whether they prefer to drive or be a passenger.

Features:

Personal Information:

- Users can update name, contact details, profile picture, and bio
- Fields like home location, travel preferences, and commuting schedule.

2 Travel Preferences:

- Specify whether the user is a driver or passenger.
- Indicate availability e.g., mornings only or both morning and evening rides.

3 Data Validation:

- Location fields must be structured e.g., dropdown for predefined areas like DHA, Clifton.
- Users can select multiple pickup/drop-off points.

4 Permissions:

- Users can only edit their own profiles.
- Admins may review profiles flagged with low ratings or complaints.
- **Example**: A student living in Gulshan can set their location and select whether they're available for morning and evening rides, making it easier for the system to match them with compatible carpools.

3) Carpool Creation and Management

Users can create carpools by specifying details like route, departure time, pickup points, and the number of available seats. Drivers can add a carpool to the system, which then makes it visible to others looking for rides along the same route.

Features:

1) **Carpool Setup:**

- Input departure time, pickup/drop-off points**,** available seats, and route.
- Set one-time or recurring carpools (e.g., every weekday).

2 Availability Management:

- Drivers can edit carpool details e.g., cancel trips or update departure times.
- A notification will be sent to affected passengers in case of changes.

3 Permissions:

- Only drivers can create and modify carpools.
- Passengers can **request to join or cancel a seat** reservation.
- **Example**: A student commuting from DHA can create a carpool departing at 8 a.m., with 3 seats available. The system will list this carpool, allowing other students along the route to request a seat.

4) Ride Matching and Search

The system will allow students to search for carpools that match their route, schedule, and preferences, automatically filtering options based on location and time.

Features:

1 Ride Search:

- Filter carpools by departure time, route, and available seats.
- Real-time search updates based on user inputs.

2 Automated Matching:

- Suggests rides based on user preferences and proximity to pickup points.
- If no exact matches are available, the system can suggest nearby routes.

3 Permissions:

- Users can search and request seats in any available carpool.
- Drivers can accept or reject seat requests.
- **Example**: A student in Clifton looking for a ride to campus between 7:30 a.m. and 8 a.m. can search the system. The system will display available carpools departing from nearby locations within that timeframe.

5) Notifications and Real-Time Updates

The system sends notifications for carpool availability, booking confirmations, cancellations, or schedule changes, helping students stay updated in real time.

Features:

1) Notification Types:

- Booking confirmation which alerts the passenger when the driver approves their request.
- Cancellation notifications which informs passengers if a trip is canceled.
- Trip reminders which sends reminders for upcoming rides e.g., 30 minutes before departure.

2 Real-Time Updates:

- Notify users if departure times change.
- Send notifications about available carpools matching user preferences.

3 Communication Options:

- In-app chat between drivers and passengers for better coordination.
- Option to enable or disable notifications.

4 Permissions:

- All users receive system notifications relevant to their bookings.
- Admins can send broadcast notifications e.g., platform maintenance announcements.
- **Example**: If a driver needs to cancel a trip, the system automatically sends a notification to the passengers who have reserved a seat, allowing them to make alternate arrangements.

6)Rating and Feedback System

After each ride, users can rate and leave feedback on their experience. This feature will improve user accountability and help others choose reliable drivers or passengers.

Features:

1) Rating System:

- After each ride, passengers and drivers can rate the experience on a scale e.g., 1-5 stars.
- Option to rate specific aspects such as punctuality, cleanliness, and safety.

2) Feedback Mechanism:

- Users can leave written feedback alongside the rating.
- Reports for inappropriate behavior trigger admin review.

3) Permissions:

- All users can provide ratings and feedback.
- Admins can suspend or warn users based on consistent low ratings.

- **Example**: A student passenger can rate their driver for punctuality, cleanliness, and safety. High ratings will build a driver's reputation, while consistent low ratings may flag a user to the admin.

7) Administrative Features

- **User Management:** Admins can view and manage all users, including banning or warning accounts with repeated violations.
- **Report Generation:** Admins can generate reports on user activity, ratings, and complaints to ensure platform health.
- **System Maintenance:** Admins have access to platform-wide settings, including broadcast messaging and managing notifications.

Use case scenarios

Use Case #1: Register as a Student

Actors: Student, staff, faculty
Description: Allows a user to create an account using their IBA credentials.

- **Workflow**:
    - The user opens the registration page.
    - They enter their IBA email, create a password, and provide any required personal information.
    - The system verifies their email by sending a confirmation link to their IBA email.
    - The user clicks the confirmation link to complete registration.
- **System Response**:
    - The system creates a new user account with the provided details.
    - A confirmation email is sent to the student.
    - The user is redirected to the login page.

Use Case #2: Log In

Actors: Student, staff, faculty
Description: Allows a registered user to log into their account.

- **Workflow**:
    - The user enters their email and password on the login page.

    - The system verifies their credentials.
    - If valid, the student is logged into the system.
- **System Response**:
    - The user is directed to the dashboard.
    - An error message appears if login fails due to incorrect credentials.

Use Case #3: Create a Ride Listing

Actors: Student, staff, faculty
Description: Allows a driver to create a new ride listing by specifying pickup/drop-off locations and time slots.

- **Workflow**:
    - The driver selects "Create Ride" from the dashboard.
    - They specify pickup/drop-off locations, time, date, and available seats.
    - The system validates the details.
    - The driver submits the listing.
- **System Response**:
    - The new ride listing is saved and becomes visible to potential passengers.
    - A confirmation notification is sent to the driver.

Use Case #4: Book a Ride

Actors: Student, staff, faculty (Passenger)
Description: Allows a passenger to search for available rides and book a seat in a ride that fits their schedule.

- **Workflow**:
    - The passenger searches for available rides by entering their desired pickup/drop-off locations and times.
    - The system displays a list of matching rides.
    - The passenger selects a ride and books a seat.
- **System Response**:
    - The ride is marked as booked for the passenger.
    - The system sends a booking confirmation notification to both the driver and the passenger.

Use Case #5: View Ride History

Actors: Student, staff, faculty (Driver and Passenger)
Description: Allows drivers and passengers to view their ride history.

- **Workflow**:
    - The user selects "Ride History" from the dashboard.
    - The system retrieves all previous rides (as driver or passenger).
- **System Response**:
    - The system displays past ride details.
    - Allows the user to click on each ride to view specific details.

Use Case #6: Receive Ride Confirmation

Actors: Student, staff, faculty (Driver and Passenger)
Description: Sends notifications to the driver and passenger when a ride is confirmed.

- **Workflow**:
    - When a passenger books a ride, the system triggers a notification to both the driver and passenger.
- **System Response**:
    - Both parties receive a notification (email, in-app, or SMS) confirming the ride.
    - Ride details and contact information of each party are included in the notification.

Use Case #7: Receive Ride Updates

Actors: Student, staff, faculty (Driver and Passenger)
Description: Sends notifications for any changes made to the ride (e.g., time changes, pickup location changes).

- **Workflow**:
    - The driver updates the ride details (e.g., time or pickup location).
    - The system verifies and saves the changes.
- **System Response**:
    - Both driver and passenger receive a notification about the update.
    - Updated details are accessible in the ride information.

Use Case #8: Receive Ride Cancellation Notification

Actors: Student, staff, faculty (Driver and Passenger)
Description: Notifies the driver and passenger if a ride is canceled.

- **Workflow**:
    - Either the driver cancels the ride, or the passenger cancels their booking.
    - The system confirms the cancellation action.
- **System Response**:
    - A cancellation notification is sent to both the driver and passenger.
    - The ride status is updated to "canceled" in the system.

Use Case #9: Cancel a Ride Booking

Actors: Student, staff, faculty(Passenger)
Description: Allows a passenger to cancel their ride booking before the ride begins.

- **Workflow**:
    - The passenger navigates to their booked rides and selects a ride they wish to cancel.
    - They confirm the cancellation request.
- **System Response**:
    - The system removes the passenger from the ride and updates the available seats.
    - A cancellation notification is sent to the driver and passenger.

Use Case #10: Manage User Profile

Actors: Student, staff, faculty
Description: Allows a student to update their profile information, such as contact details and password.

- **Workflow**:
    - The student selects "Profile Settings" from the dashboard.
    - They update contact details, profile photo, or password.
    - They save the changes.
- **System Response**:
    - The system updates the profile details.
    - A success message confirms the changes.

Use Case #11: Ban a Student

Actors: Admin
Description: Allows an admin to ban a student, revoking their access to the app and preventing further interactions.

- **Workflow**:
    - The admin logs into the system and navigates to the "User Management" section.
    - They search for the student they wish to ban using filters (e.g., by name, email, or ID).
    - The admin selects the student and clicks on the "Ban" option.
    - The system prompts the admin to confirm the action and provides a field to optionally specify the reason for the ban.
    - The admin confirms, and the system applies the ban.
- **System Response**:
    - The student's account is marked as "banned," and login access is revoked.
    - The system sends a notification to the student informing them of the ban, along with the reason if provided.
    - The banned student's status is updated in the admin dashboard for tracking.

Software Requirements:

Database Management System (DBMS)

1. **Choice of DBMS: MySQL**
    - MySQL is a widely used, open-source relational database management system known for its scalability, support, and performance. It's capable of handling complex queries and large datasets, making it ideal for applications that may grow in user base over time, such as a carpooling system.
    - **Example**: MySQL can efficiently manage data on carpool routes, ride schedules, user information, and booking details. As the platform grows, MySQL's scalability will support additional data without compromising performance.
    - **Benefits**:
        - **Scalability**: MySQL can handle a high volume of users and rides, ensuring the system remains fast and responsive as data grows.
        - **Performance**: MySQL's query optimization helps deliver quick responses to frequent searches, such as "available carpools near Gulshan."
        - **Support**: MySQL has extensive community support and documentation, making it easy to find resources and troubleshoot issues.

Development Tools and Frameworks

Programming Languages:

- **Backend**: *JavaScript* with **Node.js** for handling server-side logic, database interactions, and authentication.
- **Frontend**: *JavaScript* with **React** (or **Vue.js**) for creating a responsive and interactive user interface.
- **Database Queries**: *SQL* for database management and **Stored Procedures** if necessary for complex operations.

Frameworks and Libraries:

- **Express.js** (for Node.js): Provides a minimalist, flexible framework to manage routes and APIs for handling data transfer between frontend and backend.
- **Bootstrap** (CSS Framework): For styling and creating a user-friendly, mobile-responsive design.

Tools:

- **Postman**: For testing APIs and endpoints during development.
- **MySQL Workbench**: For database design, management, and querying

Other choice can be for Backend Development,-Django (Python), Frontend development-Angular, data interaction-SQLAlchemy (Python),

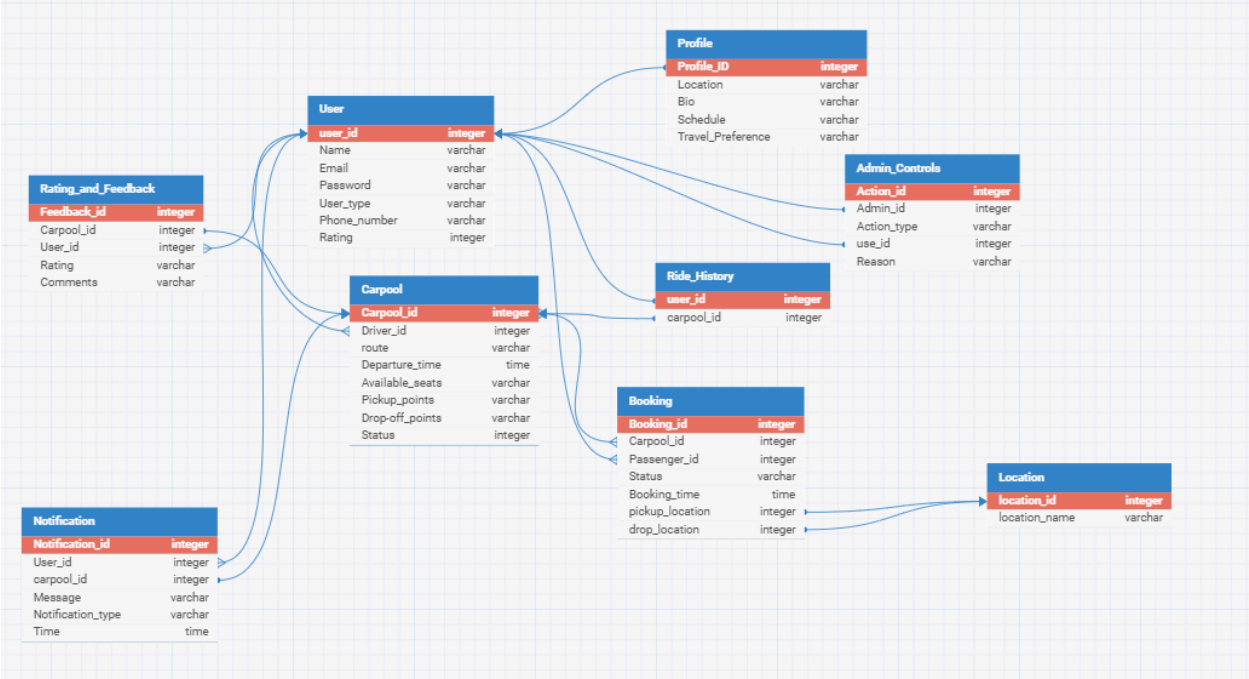
Operating System

Chosen OS: **Windows** (for development and testing)

Other Dependencies

- **Version Control**: **Git** with **GitHub** or **GitLab** for managing source code, collaborating with team members, and tracking changes.
- **Testing Frameworks**: **Jest** or **Mocha** for automated testing of the backend and **Cypress** for frontend UI testing.
- **API Documentation**: **Swagger** for documenting API endpoints, ensuring clarity in system functionalities and integration.
- **Analytics & Reporting**: **Google Analytics** (for monitoring app usage) and **Power BI** or **Tableau** (if reporting on user activity and ride metrics is required in future iterations).
- **Push Notifications**: **Firebase Cloud Messaging (FCM)** for real-time notifications about ride bookings, updates, and cancellations to both drivers and passengers.
- **Containerization**: **Docker** to package the application for easy deployment and to maintain consistency across development, staging, and production environments.

Entity Relationship Diagram



Constraints:

1. **User:**
   `user_id`: Primary Key, unique and not null.
   `Email`: Unique and not null, validating correct email format.
   `Password`: Not null with a minimum length constraint.
   `Phone_number`: Unique and not null, format validation for phone numbers.
   `Rating`: Constraint to keep the rating within a specific range (1–5).

2. **Carpool:**
   `Carpool_id`: Primary Key, unique and not null.
   `Driver_id`: Foreign Key referencing `User(user_id)`, not null.
   `Departure_time`: Not null, should be greater than current system time.
   `Available_seats`: Not null, check constraint to allow only non-negative values within a specified range.
   `Status`: Not null with limited allowed values (`active`, `inactive`).

3. **Booking:**
   `Booking_id`: Primary Key, unique and not null.
   `Carpool_id`: Foreign Key referencing `Carpool(carpool_id)`,`not null`.
   `Passenger_id`: Foreign Key referencing `User(user_id), not null`.
   `Status`: Not null, with limited allowed values (`confirmed`, `pending`, `cancelled`).
   `Booking_time`: Not null, should be greater than current system time.
   `pickup_location` and `drop_location`: Foreign Keys referencing `Location(location_id)`.

4. **Location:**
   `location_id`: Primary Key, unique and not null.
   `location_name`: Not null, unique to avoid duplicate locations.

5. **Rating_and_Feedback:**
   `Feedback_id`: Primary Key, unique and not null.
   `Carpool_id`: Foreign Key referencing `Carpool(carpool_id), not null`.
   `User_id`: Foreign Key referencing `User(user_id), not null`.
   `Rating`: Constraint to ensure rating is within a valid range.

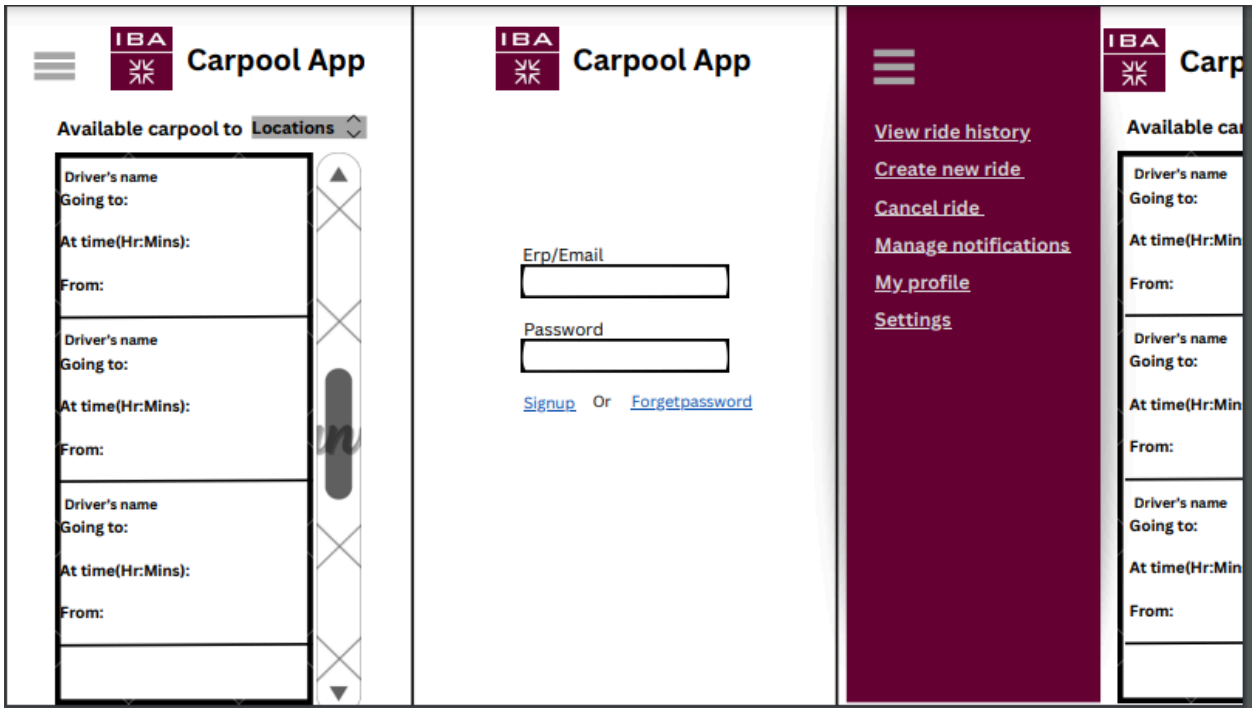
6. **Notification:**
   `Notification_id`: Primary Key, unique and not null.
   `User_id`: Foreign Key referencing `User(user_id), not null`.
   `Message`: Not null, with a character limit.
   `Notification_type`: Not null, limited to specific types (info, warning etc).
   `Time`: Not null, format validation for time.

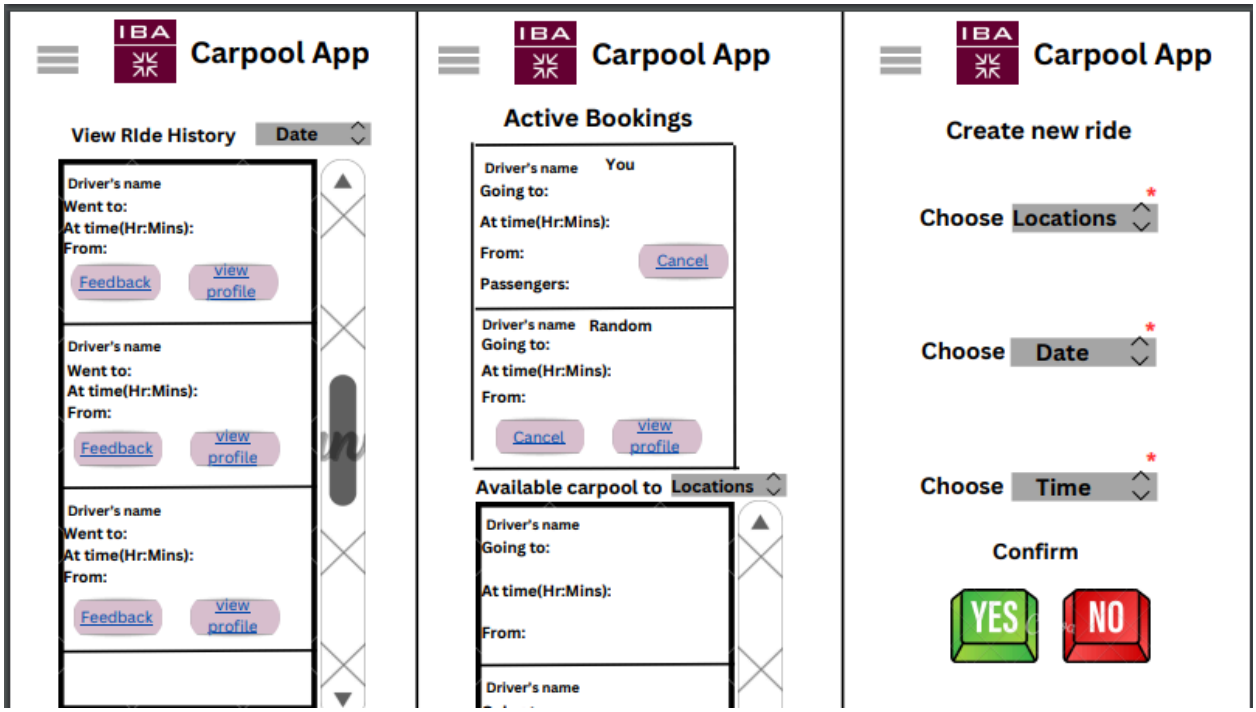
7. **Notification:**
   `Profile_ID`: Primary Key, unique and not null.
   `Location`: Foreign Key referencing `Location(location_id)`.

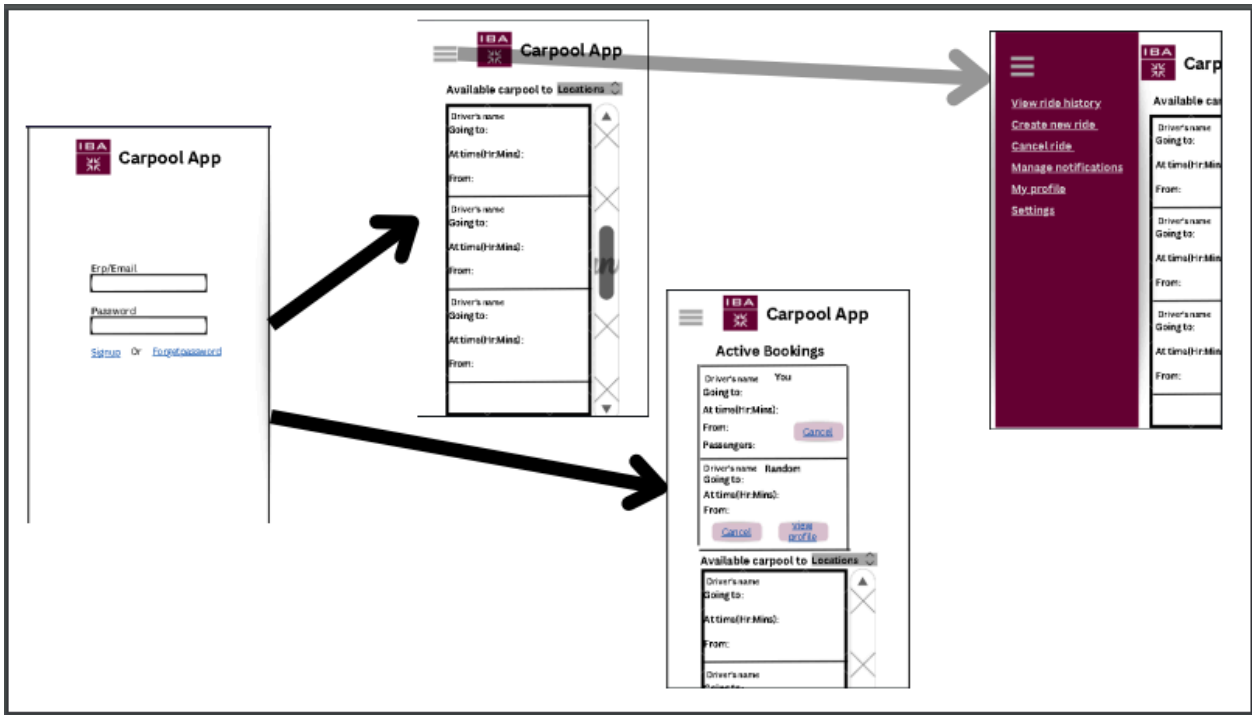
8. **Admin_Controls:**
   `Action_id`: Primary Key, unique and not null.
   `Admin_id`: Foreign Key referencing `User(user_id)`, where `User_type` should be admin.
   `user_id`: Foreign Key referencing `User(user_id)`.
   `Action_type`: Not null, with limited allowed values (`ban`, `warn etc`).
   `Reason`: Not null, providing an explanation for action taken.

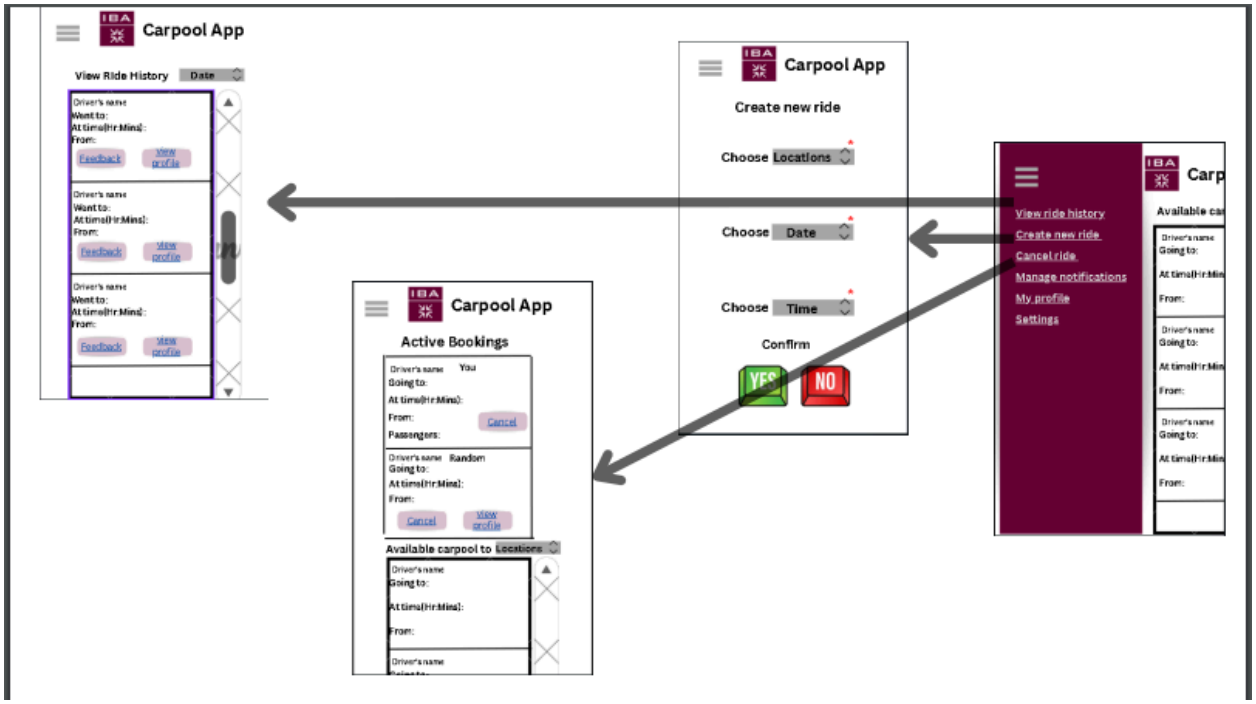
9. **Ride_History:**
   `user_id`: Foreign Key referencing `User(user_id)`, part of a composite Primary Key.
   `carpool_id`: Foreign Key referencing `Carpool(carpool_id)`, part of a composite Primary Key.

User Interface Design




Carpool App
View RIde History
Date
Driver's name
Went to:
At time(Hr:Mins):
From:
Feedback
view profile
Carpool App
Active Bookings
Driver's name
You
Going to:
At time(Hr:Mins):
From:
Cancel
Passengers:
Driver's name
Random
Going to:
At time(Hr:Mins):
From:
Cancel
view profile
Available carpool to
Locations
Carpool App
Create new ride
Choose
Locations
Choose
Date
Choose
Time
Confirm
YES
NO


Carpool App
Erp/Email
Password
Signup
Or
Forgotpassword
Carpool App
Available carpool to
Locations
Carpool App
Active Bookings
View ride history
Create new ride
Cancel ride
Manage notifications
My profile
Settings


Carpool App
View Ride History
Date
Driver's name
Went to:
At time(Hr:Mins):
From:
Feedback
View profile
Carpool App
Create new ride
Choose Locations
Choose Date
Choose Time
Confirm
YES
NO
Carpool App
Active Bookings
Driver's name You
Going to:
At time(Hr:Mins):
From:
Cancel
Passengers:
Driver's name Random
Going to:
At time(Hr:Mins):
From:
Cancel
View profile
Available carpool to Locations
Driver's name
Going to:
At time(Hr:Mins):
From:
View ride history
Create new ride
Cancel ride
Manage notifications
My profile
Settings