**FORMAN CHRISTIAN COLLEGE**

**(A CHARTERED UNIVERSITY)**

****

**Mobile Application Development (CSCS468)**

**Section A, Fall 2024**

**Assignment 1**  
**Ride Sharing App**

**Faraz Waqar (251707570)**

# Design Decisions

The Smart Ride-Sharing System is designed with simplicity and modularity in mind. The key focus areas are:

1. The system can be easily extended by adding more features, like tracking multiple ongoing rides, adding payments, etc.
2. Each entity (Driver, Rider, Ride) was designed as a separate class, handling its own state and behaviour. This keeps the logic organized and easy to maintain.
3. The app matches riders with drivers based on proximity using the Haversine formula, which calculates the distance between two geographical points.
4. A random factor simulates traffic conditions affecting the estimated time for each ride, making it realistic.
5. To keep the interface simple, a console-based UI is used for basic input and output. This is helpful for testing purposes, though it can easily be extended to a graphical UI.

# Class Structure

## Driver Class:

* + **Attributes**:
    - name: The driver’s name.
    - currentLocation: The driver's current latitude and longitude.
    - isAvailable: Whether the driver is available to accept a ride.
    - rating: The driver’s average rating.
  + **Methods**:
    - setAvailability(available: Boolean): Updates the driver’s availability.
    - updateRating(newRating: Double): Updates the driver’s rating after feedback from the rider.

## Rider Class:

* + **Attributes**:
    - name: The rider’s name.
    - currentLocation: The rider's current latitude and longitude.
    - destinationLocation: The destination's latitude and longitude.
  + **Methods**:
    - requestRide(): Finds the closest available driver and assigns a ride if one is found.

## Ride Class:

* + **Attributes**:
    - rider: The rider object requesting the ride.
    - assignedDriver: The driver assigned to the ride.
    - distanceToDestination: Distance between the rider’s location and destination (calculated using the Haversine formula).
    - estimatedTime: Estimated time for the ride, calculated based on distance and a random traffic factor.
  + **Methods**:
    - calculateDistance(): Calculates the distance between the rider and destination using the Haversine formula.
    - calculateEstimatedTime(): Calculates the estimated time for the ride, factoring in simulated traffic.

## Haversine Utility:

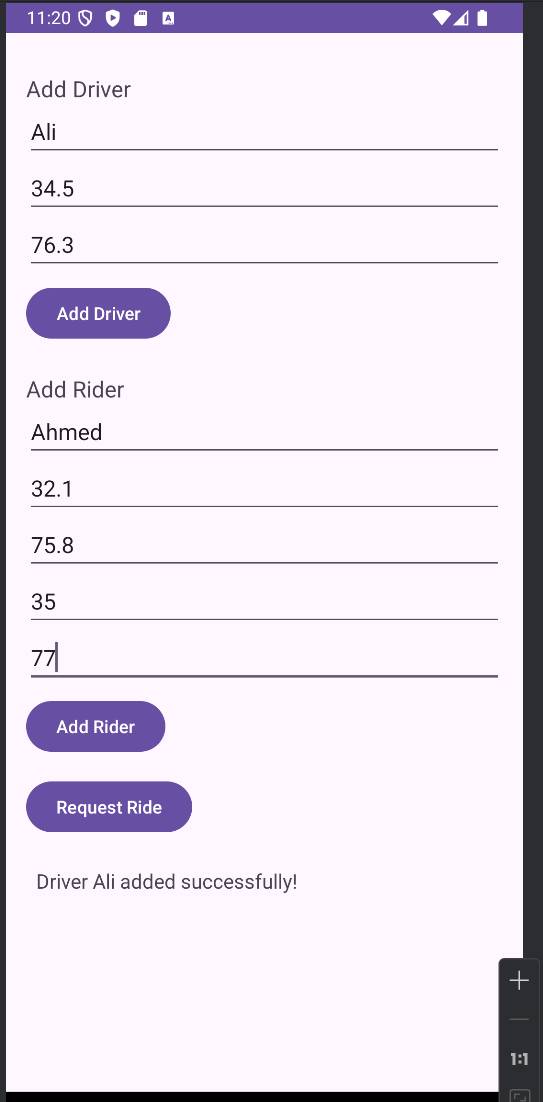
* + **Method**:
    - haversine(): Calculates the distance between two points on Earth given their latitude and longitude.

# Steps:

1. Open **Android Studio**.
2. Create a new project with **Empty Views Activity**.
3. Copy the Kotlin classes (Driver, Rider, Ride, Utils) into your project under the appropriate directories.
4. Update MainActivity.kt with the logic provided.
5. Modify activity\_main.xml with the provided XML layout for input fields and buttons.
6. Click **Run** in Android Studio and test the app.

# Inputs/Outputs:

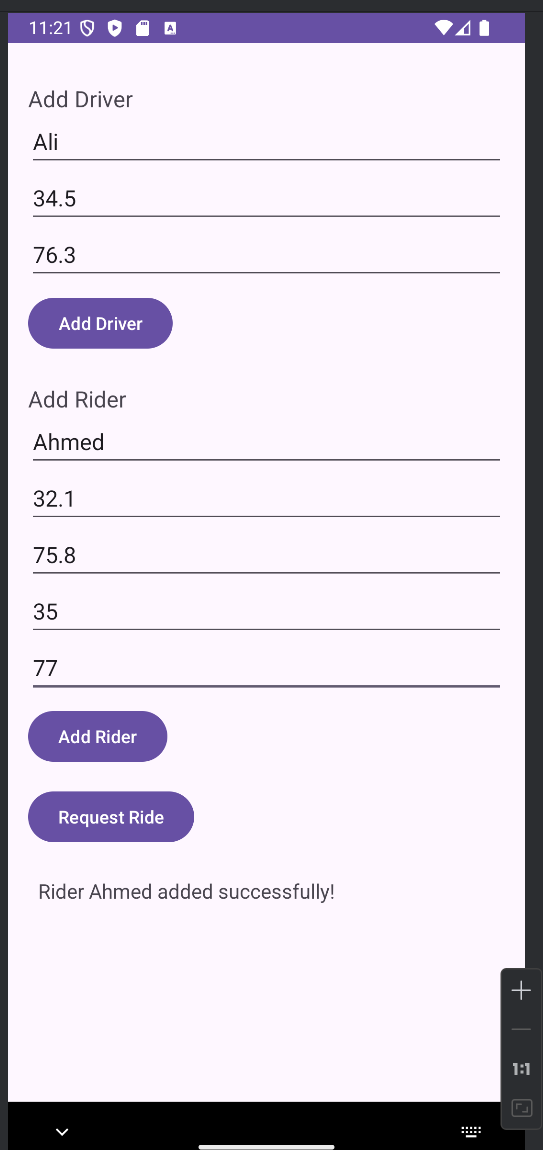
## Adding a Driver:



**Input**: Driver Name = Ali, Latitude = 34.5, Longitude = 76.3

**Output**: "Driver Ali added successfully!"

## Adding a Rider:



**Input:** Rider Name = Ahmed, Latitude = 32.1, Longitude = 75.9, Destination Latitude = 35, Destination Longitude = 77

**Output:** "Rider Ahmed added successfully!"

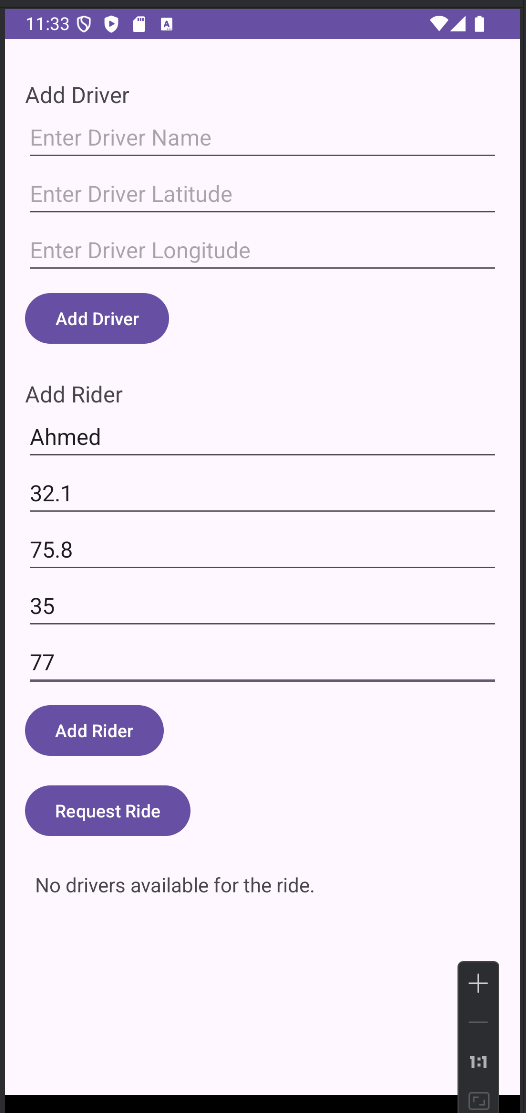
## Requesting a Ride:

**Input:** Rider Ahmed requests a ride

**Output:** "Ride assigned to Ali. Estimated time: 5.625258843297427 minutes."

## 

## No Driver Available:



**Input:** Rider Ahmed requests a ride, but no drivers are available.

**Output:** "No drivers available for the ride."

# Github Link:

https://github.com/FarazWaqar/251707570\_MobApp\_Assignment1.git