jeeny

Case Study

Instructions

Mini Ride Booking System

📘 Scenario:

You are planning to launch a **lite version** of a ride-hailing app, designed for testing in smaller cities. You've been brought in as a junior software engineer to create a **prototype** of the basic ride booking system.

This prototype **does not need to connect to real-time maps or GPS** — we're only focusing on **backend logic and frontend flows** to simulate booking rides between locations.

🧠 Goals:

Create a **simple app or backend API** that allows a passenger to:

1. **Register/Login** (Basic Auth or dummy login is fine)
2. **Request a ride**
    - Enter pickup and drop-off locations (just names like "Mall Road" or "Airport")
    - Choose ride type (Bike, Car, Rickshaw)
3. **View ride status**
    - e.g., Requested → Accepted → In Progress → Completed
4. **View ride history**

For bonus points, allow a driver to:

- Accept or reject a ride request

📒 What to Submit:

1. Design Document

- Which tech stack you used and why
- Any assumptions you made
- Entity Relationship Diagram or data model (e.g., for users, rides)

2. Functioning Code (any one of the below is fine):

- Web-based frontend using HTML/CSS/JS or React
- Android app in Java/Kotlin
- Backend-only API using Node.js/PHP/Java/Spring etc.
- SQLite/local DB simulation if no backend is available

3. Features to Implement

Feature	Required?
User Registration/Login	✅ Basic or mocked
Request a Ride	✅
View Ride Status	✅
Ride History	✅
Driver Accepts Ride	❌ Optional
Basic UI	✅ if frontend-based
API documentation (if backend)	✅ if backend-based

💻 Sample Entities You Might Create:

- **User** (id, name, type: passenger/driver)
- **Ride** (id, passenger_id, driver_id, pickup_location, drop_location, ride_type, status)
- **Driver** (id, name, availability_status)

📊 Evaluation Criteria:

Area	Weight
Problem Understanding	10%
Code Functionality	30%
Simplicity & Cleanliness	20%
Documentation	15%
Creativity & Extras	10%
Communication	15%