**Developer :** *Muhammad Usman Saleem*

**Tech Stack Used:**

* **ReactJS** (Frontend framework)
* **React Router DOM** (Routing between pages)
* **LocalStorage** (Simulated backend)
* **Pure CSS** (For elegant styling)

**Why this tech stack?**

* **No backend required** per case study.
* **React** allows component reusability and fast UI development.
* **LocalStorage** simulates persistence without requiring a database.
* **React Router** enables a clean multi-page app feel in a SPA environment.

**Assumptions Made:**

* Authentication and user roles are mocked.
* Only passengers are allowed (driver role & logic skipped).
* Ride statuses are hardcoded with controlled flow (Requested → Accepted → In Progress → Completed).
* No real-time map or location validation.
* User ID is auto-generated and stored in localStorage.
* One active ride at a time per user.

**Entity Model (Simulated in LocalStorage)**

**User**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | String | Auto-generated UUID |
| name | String | User full name |
| email | String | Unique email (username) |
| password | String | Stored as plain (mock only) |
| rides | Array | History of user ride IDs |

**Ride**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | String | Auto-generated UUID |
| passengerId | String | Linked user ID |
| pickupLocation | String | Free-text |
| dropLocation | String | Free-text |
| rideType | String | Enum: "Bike", "Car", "Rickshaw" |
| status | String | "Requested", "Accepted", etc. |
| createdAt | Date | Timestamp |

**Ride Status Flow:**

A [Requested] **-->** B [Accepted]

B **-->** C [In Progress]

C **-->** D [Completed]

**Implemented Features**

* Register/Login
* Request a Ride
* View Ride Status
* View Ride History

**UI Considerations**

* Minimalist layout
* Consistent gradient buttons
* Mobile-friendly responsiveness
* Clean font and spacing

**Navigation Flow**

Register **→** Login **→** Request Ride **→** View Status **→** Ride History