Design Document: Ride-Hailing Backend (Jeeny Case Study)

---

1. Tech Stack

Backend: Node.js with Express.js
Database: MongoDB with Mongoose ODM
Authentication: JSON Web Tokens (JWT)
Password Hashing: bcryptjs

Why this stack?

- **Node.js & Express.js** are lightweight, fast, and well-suited for RESTful APIs.
- **MongoDB** provides flexibility with document-based storage and allows easy relation through references.
- **JWT** provides a stateless authentication mechanism suitable for testing via Postman.
- **bcryptjs** ensures secure password hashing.

---

2. Assumptions

- No real-time tracking, maps, or GPS required, locations are static strings.
- A user can only be either a passenger or a driver, not both.
- Each ride is requested by one passenger and assigned to one driver.
- Ride status flows in this order: `Requested → Accepted → In Progress → Completed`

---

3. Data Model

User

```
{
  _id: ObjectId,
  username: String,
  password: String (hashed),
  type: String ("passenger" | "driver")
}
```

Driver

```
{
  _id: ObjectId,
  user: ObjectId (ref to User),
  availability_status: Boolean
}
```

Ride

```
{
  _id: ObjectId,
  passenger_id: ObjectId (ref to User),
  driver_id: ObjectId (ref to User),
  pickup_location: String,
  drop_location: String,
  ride_type: String ("Bike" | "Car" | "Rickshaw"),
  status: String ("Requested", "Accepted", "In Progress", "Completed")
}
```

These models are managed using Mongoose and mapped to MongoDB collections for persistent storage.

ER Diagram