## **🔐 Scenario: Online Cab Booking System (Securing Admin & Ride Data)**

Riya opens an online cab booking application to plan her trip.

### **🚘 Public Content (Open to Everyone)**

Without logging in, she can:

* **Browse cab options** like Mini, Sedan, or SUV.
* **Check the estimated fare** of price per kilo meter

This helps new users quickly explore the service without needing an account.

### **🔑 Protected Content (For Logged-In Users Only)**

Once Riya logs in to her account:

* She can use the **/book-cab** feature to actually book a ride.
* She can open **/my-rides** to see her **past ride history**, including details like from and to destinations.

By separating **public features** (exploring cabs, checking fares) and **protected features** (booking and ride history), the system ensures that only authenticated users can access personal and sensitive ride data.

✨ This flow makes the cab booking app secure, user-friendly, and similar to real-world apps like Uber or Ola.

1. **Public Content (Open to All):**
   * Browse cab options (Mini, Sedan, SUV)
   * /cabs → list all cabs
2. **Protected Content (Requires Authentication):**
   * /book-cab → Book a cab for a logged-in passenger.
   * /my-rides → View ride history.

### **🔹 How JWT & Passport.js are used**

**2.1 Secure routes using JSON Web Tokens (JWT):**

* When a user logs in with phone/email, the server generates a **JWT**.
* This token must be sent in the Authorization: Bearer <token> header for every protected API call.
* Example: Accessing /book-cab or /my-rides requires a valid token.

**2.3 Utilize Passport.js to facilitate JWT use:**

* Passport middleware checks the token before allowing access.

**2.4 Passport.js authentication strategies:**

* **Local Strategy** → Authenticate user via email/phone + password.
* **JWT Strategy** → Verify requests contain a valid JWT.
* **Google OAuth Strategy** → Allow login with Google for faster signup.