🔹 **Week 1: Database & Backend Setup**

📌 **Day 1: Set Up MySQL & Database Schema**

- ✅ Install **MySQL** and create car_rental database
- ✅ Define tables: **Customers, Vehicles, Reservations, Payments, etc.**
- ✅ Set up **relationships** between tables (foreign keys)
- ✅ Insert sample data for testing

📌 **Day 2: Install Flask & Connect to MySQL**

- ✅ Install Flask and required libraries (Flask-MySQLdb, flask-cors, etc.)
- ✅ Set up Flask **app structure** (app.py, models.py, routes.py)
- ✅ Establish **database connection** in Flask
- ✅ Test connection by retrieving sample data

📌 **Day 3: Implement User Authentication**

- ✅ Create users table (id, name, email, password)
- ✅ Implement **signup & login API** using JWT authentication
- ✅ Hash passwords using **bcrypt**
- ✅ Test authentication API in **Postman**

📌 **Day 4: Develop API for Rental Shops & Vehicles**

- ✅ Create APIs to **add, update, delete, and fetch rental shops & vehicles**
- ✅ Implement filters (e.g., location-based search for rental shops)
- ✅ Test using Postman

📌 **Day 5: Build Reservation & Payment APIs**

- ✅ Implement API for **creating a reservation** (date, time, location)
- ✅ Ensure availability check before confirming booking
- ✅ Create API for **processing payments**
- ✅ Store **transaction numbers** in MySQL

📌 **Day 6: Add Feedback & Extra Charges APIs**

- ✅ Implement API for users to **submit feedback**
- ✅ Create API to **add extra charges** (e.g., late fees, damages)
- ✅ Ensure data validation (e.g., no negative amounts)

📌 **Day 7: Debug & Optimize**

- ✅ Handle **errors** (e.g., invalid input, booking conflicts)
- ✅ Optimize **database queries** for speed
- ✅ Ensure all APIs return **proper JSON responses**

🔹 **Week 2: Frontend (React) Setup & Integration**

📌 **Day 8: Set Up React Project**

✅ Install React (npx create-react-app car-rental-ui)
✅ Set up **React Router** for navigation (Home, Booking, Payments)
✅ Install **Axios** for API calls

📌 **Day 9: Display Rental Shops & Vehicles**

✅ Fetch rental shops & vehicles from Flask API
✅ Display data using **cards or tables**

📌 **Day 10: Build Reservation & Payment Forms**

✅ Create **reservation form** (date, time, vehicle selection)
✅ Implement **payment form** (card details, UPI, net banking)

📌 **Day 11: Connect Forms to API**

✅ Use Axios to **send reservation & payment data** to Flask API
✅ Handle **API errors & success responses**

📌 **Day 12: Implement User Authentication in Frontend**

✅ Create **login & signup pages**
✅ Store JWT token after login
✅ Restrict certain pages to **logged-in users only**

📌 **Day 13: Test Frontend & Fix UI Issues**

✅ Ensure **smooth navigation** between pages
✅ Fix UI issues (alignment, responsiveness)

📌 **Day 14: Final Testing for Frontend-Backend Connection**

✅ Test **entire user journey** (signup → book → pay → confirmation)
✅ Debug any API connection issues

🔹 **Week 3: Advanced Features & Security Enhancements**

📌 **Day 15: Implement Admin Panel**

✅ Create an **admin login**
✅ Build UI for **managing cars, users, and payments**

📌 **Day 16: Add Google Maps for Rental Shop Locations**

✅ Use **Google Maps API** to show rental shop locations
✅ Allow users to **select pickup/drop-off points**

📌 **Day 17: Implement Payment Gateway (Razorpay/Stripe)**

✅ Replace mock payments with **real payment gateway**
✅ Ensure **secure transactions** and proper error handling

📌 **Day 18: Optimize API Performance & Security**

✅ Optimize **database queries** for faster response
✅ Secure API endpoints using **JWT authentication**

📌 **Day 19: Implement Email & SMS Notifications**

✅ Send **confirmation emails** after successful booking
✅ Use Twilio/API to send **SMS reminders**

📌 **Day 20: Implement Subscriptions & Discounts**

✅ Add **subscription plans** (e.g., premium users get discounts)
✅ Apply discount logic in **billing API**

📌 **Day 21: Full Testing & Bug Fixes**

✅ Check **database integrity** (no duplicate or missing records)
✅ Test **all edge cases** (invalid input, network errors)

🔹 **Week 4: Deployment & Final Touches**

📌 **Day 22: Deploy Backend on Render/Railway**

✅ Host Flask API on **Render, Railway, or AWS**
✅ Ensure API is accessible via a public URL

📌 **Day 23: Deploy Frontend on Vercel/Netlify**

✅ Host React UI on **Vercel or Netlify**
✅ Connect frontend to **deployed Flask API**

📌 **Day 24: Run Final Full-System Test**

✅ Test booking workflow **on deployed system**
✅ Debug any **cross-origin (CORS) issues**

📌 **Day 25: Improve UI & UX**

✅ Add **loading spinners & animations**
✅ Improve **mobile responsiveness**

📌 **Day 26: Final Documentation**

✅ Write **README.md** with installation steps
✅ Create **API documentation** (Postman collection)

📌 **Day 27: Prepare for Presentation**

✅ Create a **demo video or slide deck**
✅ Highlight key features & challenges faced

📌 **Day 28: Project Wrap-Up & Review**

✅ Review code for **cleanliness & efficiency**
✅ Ensure all **features work smoothly**

🚀 **Bonus Features (If You Have Extra Time)**

✅ Implement **AI-based car recommendation system**
✅ Allow users to **upload driving licenses** for verification
✅ Add **chat support for customer queries**

🎯 **Final Checklist Before Submission**

✅ **Working Flask API (backend)**
✅ **MySQL database properly connected**
✅ **Fully functional React frontend**
✅ **Deployed project online**
✅ **Documentation ready**