B.Tech. 3rd Semester (2025-26) Project: Car Pooling Web Application

Project Overview

Project Start Date: 7th July 2025
Project End Date: 27th September 2025

This document outlines the tasks for each phase of the project. Students are expected to complete all tasks within the specified phases to develop a fully functional Car Pooling Web Application from scratch. The project is divided into three phases:

1. **Phase 1 (Research + Design)**: Weeks 1 to 4
   Research existing carpooling platforms, create wireframes, design static HTML/CSS pages, and design a database schema.
2. **Phase 2 (Node.js Backend)**: Weeks 5 to 8
   Build the backend using Node.js to handle user data, ride requests, and API endpoints, incorporating the database.
3. **Phase 3 (React + CSS + Integration)**: Weeks 9 to 12
   Develop the frontend using React for interactivity, apply CSS for responsive styling, and integrate with the Node.js backend.

Technologies Used

1. Node.js

Node.js is a JavaScript runtime environment that enables server-side development. It will be used to create a backend API to handle user authentication, ride requests, and data storage for the Car Pooling App.

2. React

React is a JavaScript library for building user interfaces, particularly single-page applications. It allows you to create reusable UI components, making the application modular and efficient. In this project, React will manage the dynamic rendering of ride listings, user profiles, and forms.

3. HTML (HyperText Markup Language)

HTML is the standard markup language used to create the structure of web pages. It defines the content and layout of the application, such as navigation bars, forms, and sections for ride listings.

4. CSS (Cascading Style Sheets)

CSS is used to style and visually enhance the web application. It controls the layout, colors, fonts, and responsiveness to ensure the application is appealing across devices.

Project Description

You are required to build a **Car Pooling Web Application** with the following features:

- **Homepage**: Display an overview of the app with a navigation bar linking to other pages.
- **Ride Search Page**: Allow users to search for available rides by entering details like source, destination, and date.
- **Offer Ride Page**: Enable users to post a ride by providing details like car type, seats available, and time.
- **Profile Page**: Display user information (e.g., name, contact, past rides).
- **Contact Us Page**: Include a form for users to submit inquiries or feedback.
- **Responsive Design**: Ensure the application is usable on both desktop and mobile devices.
- **Backend (Node.js)**: Implement API endpoints to store and retrieve ride and user data using a database.

Week-wise TODO List

Week	Dates	Tasks	Deliverable	Detailed Specification
Week 1	7th - 13th July 2025	- Research at least 5 carpooling websites available in the market. - Prepare a document detailing their functionalities. - Understand how these functionalities can guide your project.	Document analyzing 5 carpooling websites and their functionalities.	**Purpose**: Understand the features of existing carpooling platforms to inform your project's design. **Functionality**: Research five carpooling websites (e.g., Uber Pool, Lyft Line, BlaBlaCar, Waze Carpool, Scoop). Document key functionalities like user registration, ride matching, payment systems, route optimization, user ratings, and real-time tracking. For each website, include: (1) a brief overview, (2) key features (e.g., ride matching based on location, cost-sharing options), (3) user interface elements (e.g., search forms, map integration), and (4) how these can be simplified for your project (e.g., basic ride search without payment). Submit a markdown or PDF document summarizing findings and their relevance to your app.
Week 2	14th - 20th July 2025	- Prepare wireframes for the Car Pooling Web Application.	Wireframe designs for all pages.	**Purpose**: Plan the layout and structure of the application's user interface. **Functionality**: Create wireframes for the homepage, Ride Search, Offer Ride, Profile, and Contact Us pages. Include key elements like navigation bars, forms, buttons, and placeholders

Week	Dates	Tasks	Deliverable	Detailed Specification
				for images (e.g., car images, maps). Ensure wireframes reflect the functionalities identified in Week 1 (e.g., ride search form, user profile details). Use tools like Figma, Sketch, or pen-and-paper sketches. Submit wireframes as images or a PDF, explaining the purpose of each page and its components.
Week 3	21st - 27th July 2025	- Design static HTML and CSS pages based on the wireframes.	Static HTML/CSS pages for the application.	**Purpose**: Build the visual structure of the application using static HTML and CSS. **Functionality**: Create HTML files for the homepage, Ride Search, Offer Ride, Profile, and Contact Us pages, following the Week 2 wireframes. Use semantic HTML tags (e.g., `<header>`, `<nav>`, `<section>`) for structure. Apply basic CSS for layout (e.g., navigation bar, form alignment), colors (e.g., blue for navigation), and fonts (e.g., Arial). Include placeholder images from the provided project images. Ensure pages are visually consistent with the wireframes.
Week 4	28th July - 3rd August 2025	- Design a database schema for the application.	Database schema for users and rides.	**Purpose**: Plan the data structure for storing user and ride information. **Functionality**: Design a database schema with at least two tables: `Users` (fields: `id`, `name`, `email`, `phone`) and `Rides` (fields: `id`, `source`, `destination`, `date`, `seats`, `driverId`). Define relationships (e.g., `driverId` in `Rides` references `id` in `Users`). Use a simple in-memory storage (e.g., JavaScript arrays) or a lightweight database like SQLite for this project. Submit a diagram or markdown document describing the schema and its purpose.

Week	Dates	Tasks	Deliverable	Detailed Specification
Week 5	4th - 10th August 2025	- Set up a Node.js project with Express.js. - Create a basic server with a `/health` endpoint.	Initial Node.js server setup.	**Purpose**: Initialize the backend to support the database and API functionality. **Functionality**: Set up a Node.js project and create a server using Express.js on port 3000. Implement a GET `/health` endpoint that returns `{ "status": "Server is running" }`. Test the endpoint to confirm the server is operational.
Week 6	11th - 17th August 2025	- Create API endpoints for users (e.g., GET `/users`, POST `/users`). - Implement database storage for users.	User-related API endpoints with database integration.	**Purpose**: Enable user data management using the database. **Functionality**: Create a GET `/users` endpoint to list all users and a POST `/users` endpoint to register a new user with fields like `name`, `email`, and `phone`. Store user data in the database designed in Week 4. Ensure the POST endpoint returns the created user with a unique `id`. Test by registering users and retrieving the user list.
Week 7	18th - 24th August 2025	- Create API endpoints for rides (e.g., GET `/rides`, POST `/rides`). - Add basic error handling for API requests.	Ride-related API endpoints with error handling.	**Purpose**: Allow creation and retrieval of ride data. **Functionality**: Implement a GET `/rides` endpoint to list all rides and a POST `/rides` endpoint to create a ride with fields like `source`, `destination`, `date`, `seats`, and `driverId`. Store rides in the database. Include error handling to return a 400 status if required fields are missing. Test by creating rides and retrieving the ride list.
Week 8	25th - 31st August 2025	- Implement advanced API endpoints (e.g., GET `/rides/:id`, DELETE `/rides/:id`, PUT `/rides/:id`). - Test APIs using tools like Postman.	Complete Node.js backend with tested APIs.	**Purpose**: Enhance the backend with operations for specific rides. **Functionality**: Create a GET `/rides/:id` endpoint to retrieve a ride by ID, a DELETE `/rides/:id` endpoint to remove a ride, and a PUT `/rides/:id` endpoint to update ride details. Return a 404 status if the ride is not found. Test all endpoints to

Week	Dates	Tasks	Deliverable	Detailed Specification
				ensure they handle user and ride data correctly.
Week 9	1st - 7th September 2025	- Set up a React project using CDN links for React and Babel. - Convert static HTML pages to React components.	Initial React project with page components.	**Purpose**: Transition the static pages to a dynamic React frontend. **Functionality**: Set up an HTML file with React and Babel CDN links. Convert the Week 3 HTML pages (homepage, Ride Search, Offer Ride, Profile, Contact Us) into React components (`Home.js`, `RideSearch.js`, `OfferRide.js`, `Profile.js`, `ContactUs.js`). Include navigation using `react-router-dom`. Ensure the layout matches the wireframes and Week 3 designs.
Week 10	8th - 14th September 2025	- Implement forms and interactivity in React components. - Connect React frontend to Node.js backend using `fetch`.	Interactive React frontend with backend integration.	**Purpose**: Add functionality to the frontend and connect it to the backend. **Functionality**: In `RideSearch.js`, create a form for `source`, `destination`, and `date`, and fetch rides from GET `/rides` to display in a list. In `OfferRide.js`, create a form for `source`, `destination`, `date`, `seats`, and `carType`, and send data to POST `/rides`. In `Profile.js`, fetch user data from GET `/users/:id` (use a static ID). Add form validation and error messages (e.g., “No rides found”).
Week 11	15th - 21st September 2025	- Apply CSS to React components for responsive styling. - Add advanced interactivity (e.g., update/delete rides).	Styled and interactive React frontend.	**Purpose**: Style the application and enhance interactivity. **Functionality**: Update `styles.css` to style navigation (e.g., blue background), forms (e.g., green buttons), and layouts (e.g., grid for ride listings). Use Flexbox or Grid and media queries for responsiveness (320px to 1200px). In `RideSearch.js`, add features to view (GET `/rides/:id`) and delete rides (DELETE `/rides/:id`). In `OfferRide.js`, allow editing rides

Week	Dates	Tasks	Deliverable	Detailed Specification
				(PUT `/rides/:id`). Ensure validation errors are displayed.
Week 12	22nd - 27th September 2025	- Finalize frontend-backend integration. - Test the application across browsers and devices. - Present the project and collect feedback.	Finalized full-stack application with presentation.	**Purpose**: Complete and showcase the application. **Functionality**: Ensure all features (ride search, offer ride, profile, contact form) work seamlessly with the backend. Test the application in Chrome, Firefox, and on mobile/desktop devices. Prepare a presentation summarizing the project, including research, backend, frontend, and styling. Demonstrate creating and searching rides. Document challenges and solutions in `README.md`. Collect feedback during the presentation.