ExpoWheels Technical Design Document

Providing a One Stop Destination for Buying Selling and Renting Cars on Affordable Price

---

Author :- Harshit Singla
Date :- 24 - December - 2024
Version - 2

Preface

The Technical Document Outlines the Architecture, Features and Implementation Strategies for ExpoWheels, A *MERN* Stack Application to Facilitate Car Buying, Selling and Renting. It serves as a blueprint for the development process, ensuring alignment among team members and stakeholders.

Scope

ExpoWheels is a web-based platform that connects car owners with potential buyers and renters. The system will include features such as user registration, car listings, search and filter capabilities, secure payments, and an admin panel for platform management.

Document Structure

This document is organized into the following sections:

- **System Overview**: High-level description of the system architecture and deployment strategy.
- **Requirements**: Detailed breakdown of functional and non-functional requirements.
- **Technical Design**: Database schemas, API endpoints, and UI/UX components.
- **Testing and Deployment**: Strategies for testing and deploying the application.
- **Future Enhancements**: Planned features and scalability considerations

TABLE OF CONTENTS

Introduction

- **Project Overview**
  Expo Wheels is a MERN stack-based platform for buying, selling, and renting cars at affordable prices. It simplifies vehicle transactions by connecting buyers, sellers, and renters in one seamless, user-friendly interface. Whether you're purchasing, selling, or renting, Expo Wheels ensures transparency, convenience, and affordability for all your automotive needs.

- **Objectives**
  - Facilitate Seamless Vehicle Transactions
  - Offer Affordable Solutions for Users
  - Create a Comprehensive Automotive Marketplace
  - Ensure Transparency and Trust in Transactions
  - Deliver a User-Friendly and Accessible Platform
  - Enable Secure and Scalable Operations

- **Target Audience**
  The primary audience for *Expo Wheels* includes individual buyers, sellers, and renters of vehicles, as well as automotive businesses like dealerships and rental services. The platform is designed to cater to

their diverse needs, offering a seamless and reliable experience for all stakeholders involved in vehicle transactions.

System Overview

- **Architecture**
  ExpoWheels utilizes a *MERN* stack architecture that integrates MongoDB for data storage, Express.js as a backend web application framework, React.js for the frontend user interface, and Node.js as the runtime environment. This architecture enables seamless, full-stack JavaScript development, facilitating efficient and dynamic operations.

- **Core Components**
  - **Frontend (React.js) :-**
    The frontend is the user-facing part of the platform, built using React.js.

    Key Features:

    - *Dynamic and Responsive Interface:*
      React's component-based architecture allows for building reusable UI components, ensuring consistency and responsiveness across all pages.

    - *Routing:*
      Implements smooth navigation using React Router for Single Page Application (SPA) behavior, avoiding full-page reloads.

    - *State Management:*
      Utilizes tools like Redux or Context API to manage the state of the application, including user data, vehicle listings, and search filters.

    - *Third-Party Libraries:*
      Tailwind CSS is to Provide an Immersive User Interface and Experience.

- **Backend (Node.js and Express.js) :-**
  The backend is the server-side part of the platform, responsible for handling business logic, API requests, and data processing.

  Key Features:

  - *RESTful APIs:*
    Exposes endpoints for various operations like fetching car listings, managing users, and processing payments.
  - *Middleware Management:*
    Uses middleware for request validation, authentication, and error handling.
  - *Scalability*:
    Node.js's non-blocking, event-driven architecture makes the backend capable of handling multiple concurrent requests efficiently.
  - *Data Validation*:
    Implements robust validation with libraries like Joi or express-validator to ensure data integrity.
  - *Authentication and Authorization:*
    Uses JWT (JSON Web Tokens) for secure user authentication and role-based access control (RBAC).

- **Database (MongoDB) :-**
  The frontend is the user-facing part of the platform, built using React.js.

  Key Features:

  - *Flexible Schema:*
    MongoDB's document-based structure allows dynamic schema definitions, ideal for storing car details with varying attributes (e.g., make, model, year, condition).
  - *Scalability:*
    Supports horizontal scaling to handle increased data and user loads.

- *Indexes and Search*:
  Leverages indexes for faster search and retrieval, essential for features like filtering and sorting vehicle listings.

- *Relationships*:
  Manages relationships using references (e.g., user-car relationships) or embedded documents (e.g., reviews within car listings).

- **Deployment**
  The hosting and deployment components ensure the platform is available and accessible to users.

  Key Features:

  - *Frontend Hosting*:
    Platforms like Vercel, Netlify, or AWS amplify are used to host the React-based frontend, ensuring global availability with CDN integration.

  - *Backend Hosting*:
    Services like Heroku, AWS EC2, or DigitalOcean host the Express.js server, providing high uptime and scalable infrastructure.

  - *Database Hosting*:
    MongoDB Atlas or similar cloud solutions are used to host the database, offering automatic backups, clustering, and monitoring.

  - *CI/CD Pipelines*:
    Tools like GitHub Actions or Jenkins automate testing, building, and deployment processes to maintain a smooth development workflow.

REQUIREMENTS

- **Core Features**
  - Buy Car
  - Sell Car
  - Rent Car

- **User Management**
  - Registration
  - Login
  - Profile Management

- **Product Listings**
  - Adding a new Car
  - Editing a Car Details
  - Deleting a Car
  - Putting a Car on Sale
  - Retracting the Car or Sale Itself

- **Search and Filters**
  - By Client Budget or Car Price
  - By Rent Time
  - By Wait Time
  - By Car Details
    - Brand
    - Milege
    - Fuel Type

- **Payment**
  - Will Use RazorPay API for Fast Payments

- **Admin**
  - User and Feedback Management
  - Listing Moderation

- **Security Considerations**
    1. **Authentication**
        - **JWT Tokens:**
            - Securely generated and signed with a secret key to authenticate users.
            - Stored in the client's localStorage for maintaining user sessions.
            - Tokens are set to expire after a specific duration to mitigate risks of token misuse.
        - **Password Security:**
            - User passwords are securely hashed using bcrypt before being stored in the database.
            - Plain-text passwords are never stored or logged to prevent potential breaches.
            - A password reset mechanism is in place, using email-based verification links.

    2. **Authorization**
        - **Protected Routes:**
            - Backend API routes are accessible only with a valid JWT token included in the request header.
            - Frontend routes for sensitive actions (e.g., adding a car, managing listings) are restricted using React's higher-order components (HOCs) or context-based checks.
        - **Input Validation:**
            - All user inputs are sanitized to prevent injection attacks (e.g., SQL Injection, XSS).
            - Validation rules are enforced both on the client and server sides to ensure data integrity.

3. **CORS Configuration**
    - **Access-Control Policies:**
        - The backend is configured to allow requests only from trusted origins (e.g., the Expo Wheels frontend URL hosted on Vercel).
        - Proper headers are set for:
            - Access-Control-Allow-Origin: Restricts origins to authorized domains.
            - Access-Control-Allow-Methods: Limits HTTP methods (e.g., GET, POST, PUT, DELETE) based on functionality.
            - Access-Control-Allow-Headers: Specifies the required headers like Authorization, Content-Type, and Accept.

TECHNICAL DESIGN

1. DataBase Design

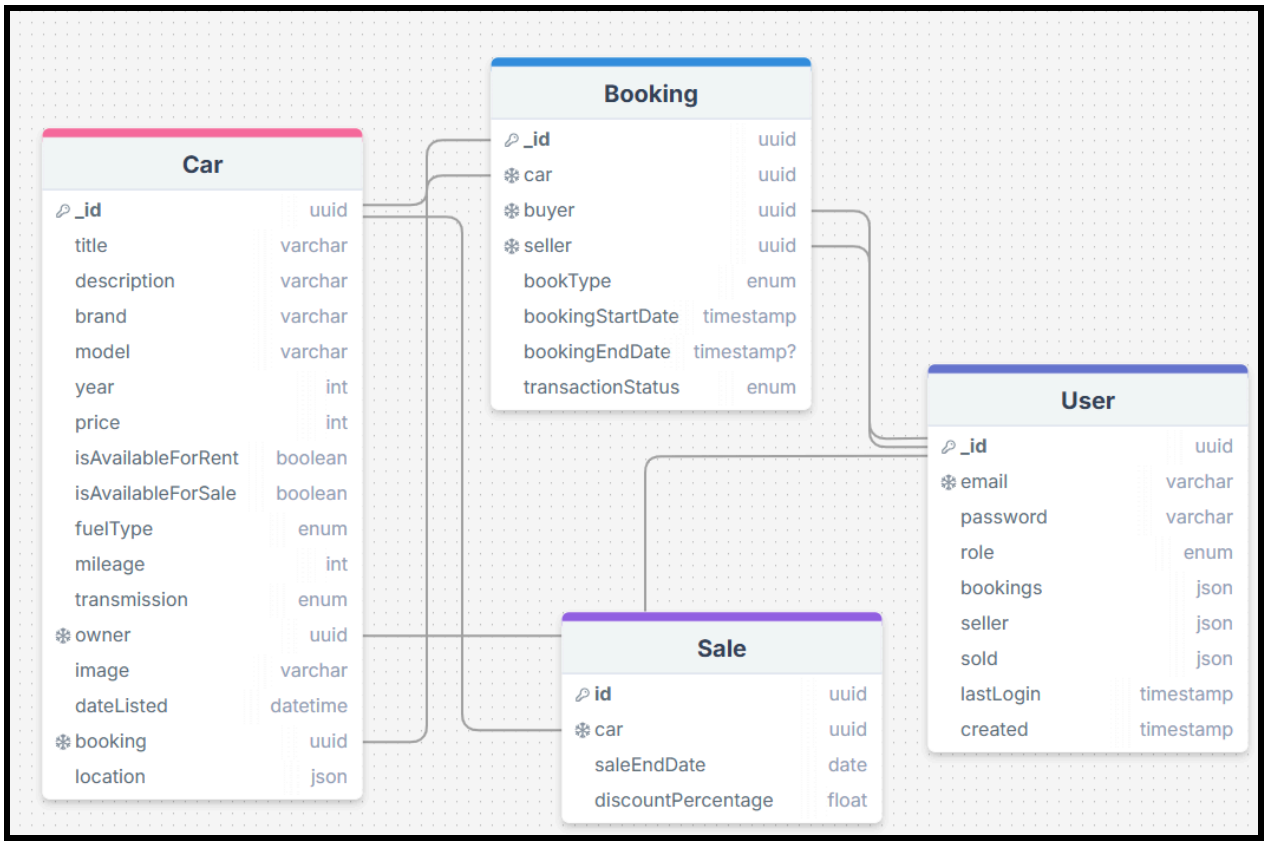
- Car
    - Id :- Mongo Schema ID (Primary)
    - Title :- String
    - Description :- String
    - Brand :- String
    - Model :- String
    - Year :- Date
    - Price :- Number
    - isAvailableForRent :- Boolean
    - isAvailableForSale :- Boolean
    - FuelType :- Enum ["Petrol", "Diesel", "Electric", "Hybrid", "CNG", "LPG"]
    - Mileage :- Number
    - Transmission :- Enum [“Manual”, “Automatic”, “Semi - Automatic”]
    - Owner :- Mongo Schema ID (Unique) Reference to User Id
    - Image :- String
    - Date Listed :- Date

    - Booking :- Mongo Schema ID (Unique) Reference to Booking
    - Location :- Schema
        - City :- String
        - State :- String
        - Country :- String
        - ZipCode :- String

- User
    - Id :- Mongo Schema ID (Primary)
    - Name :- String
    - Email :- String
    - Password :- String
    - Role :- Enum ["Buyer", "Seller", "Admin"]
    - Bookings :- Array of Mongo Scheme ID (Reference to Booking)
    - Seller :- Array of Mongo Scheme ID (Reference to Car)
    - Sold :- Array of Mongo Scheme ID (Reference to Car)
    - lastLogin :- Date
    - Created :- Date

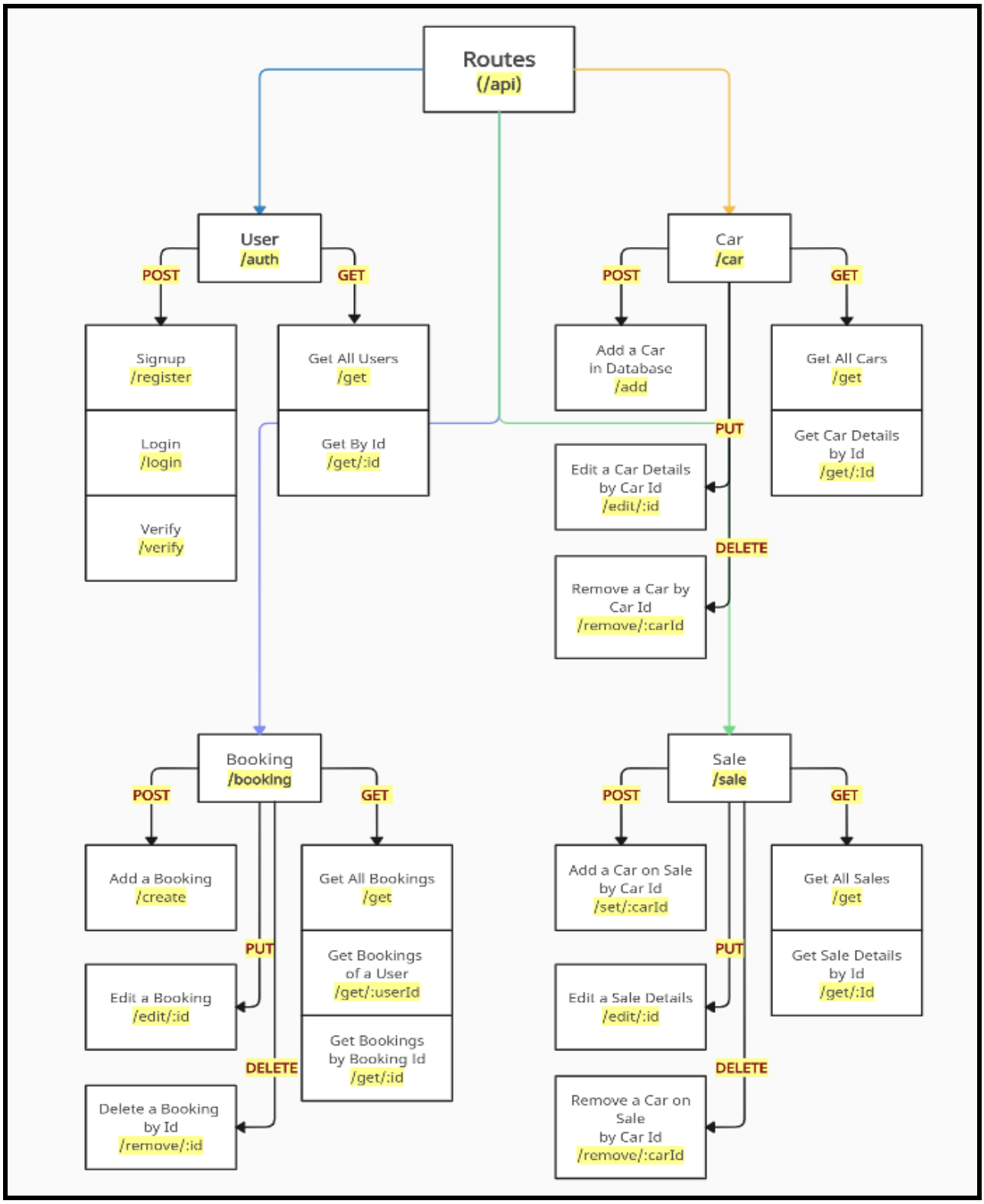
- Sale
    - Id :- Mongo Schema Id
    - Car :- Mongo Schema Id Reference to Car ID
    - SaleDate :- Date
    - DiscountPercentage :- Number (Min - 0, Max - 100)

- Booking
    - Id :- Mongo Schema Id
    - Car :- Mongo Schema Id Reference to Car Id
    - Buyer :- Mongo Schema Id Reference to User ID
    - Seller :- Mongo Schema Id Reference to User ID
    - Book Type :- Enum ["Rent", "Buy"]
    - Booking Start Date :- Date
    - Booking End Date :- Date
    - Transaction Status :- Enum ["Pending", "Completed", "Cancelled"]

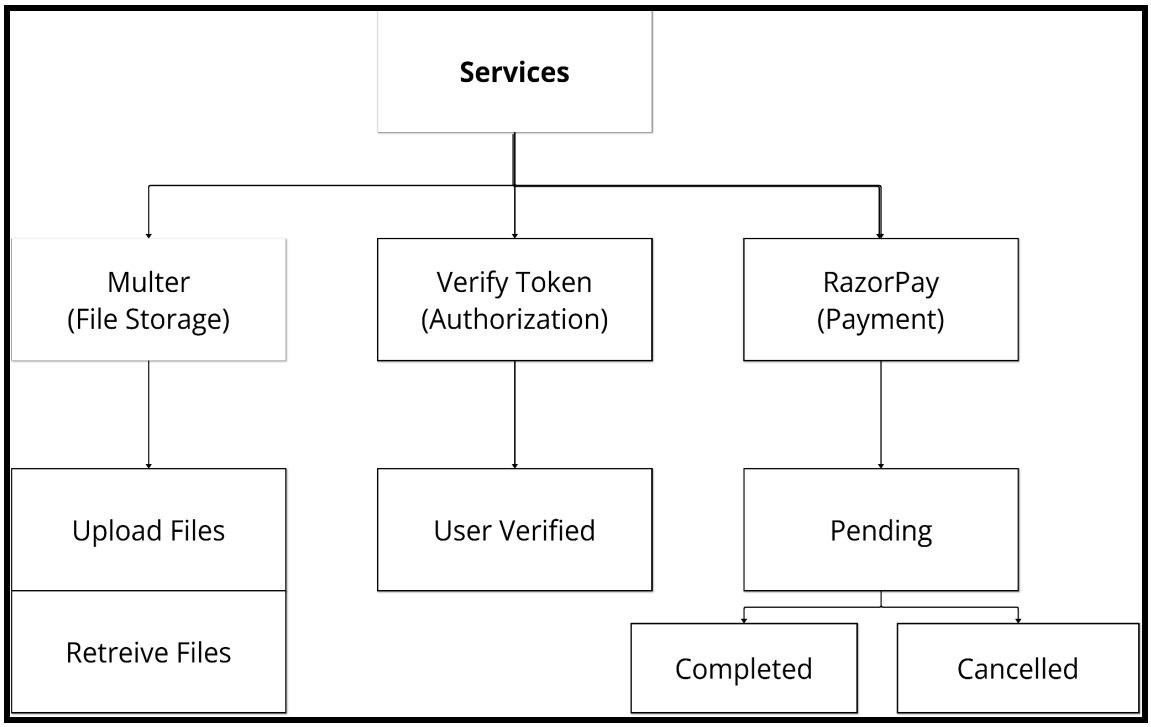

Car
_id uuid
title varchar
description varchar
brand varchar
model varchar
year int
price int
isAvailableForRent boolean
isAvailableForSale boolean
fuelType enum
mileage int
transmission enum
owner uuid
image varchar
dateListed datetime
booking uuid
location json
Booking
_id uuid
car uuid
buyer uuid
seller uuid
bookType enum
bookingStartDate timestamp
bookingEndDate timestamp?
transactionStatus enum
Sale
id uuid
car uuid
saleEndDate date
discountPercentage float
User
_id uuid
email varchar
password varchar
role enum
bookings json
seller json
sold json
lastLogin timestamp
created timestamp

2. Backend Design

- API Route

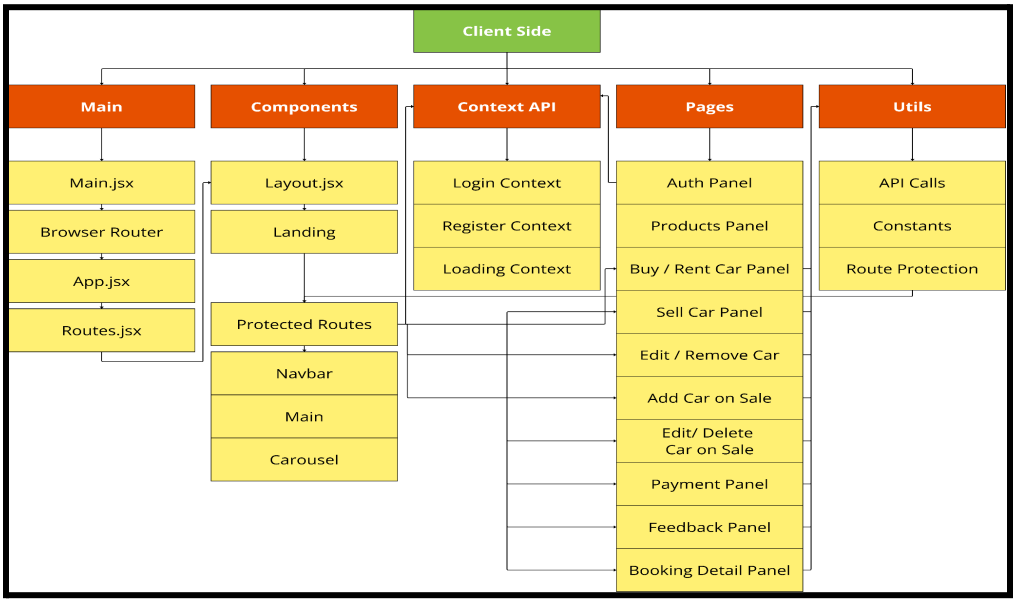


- Services



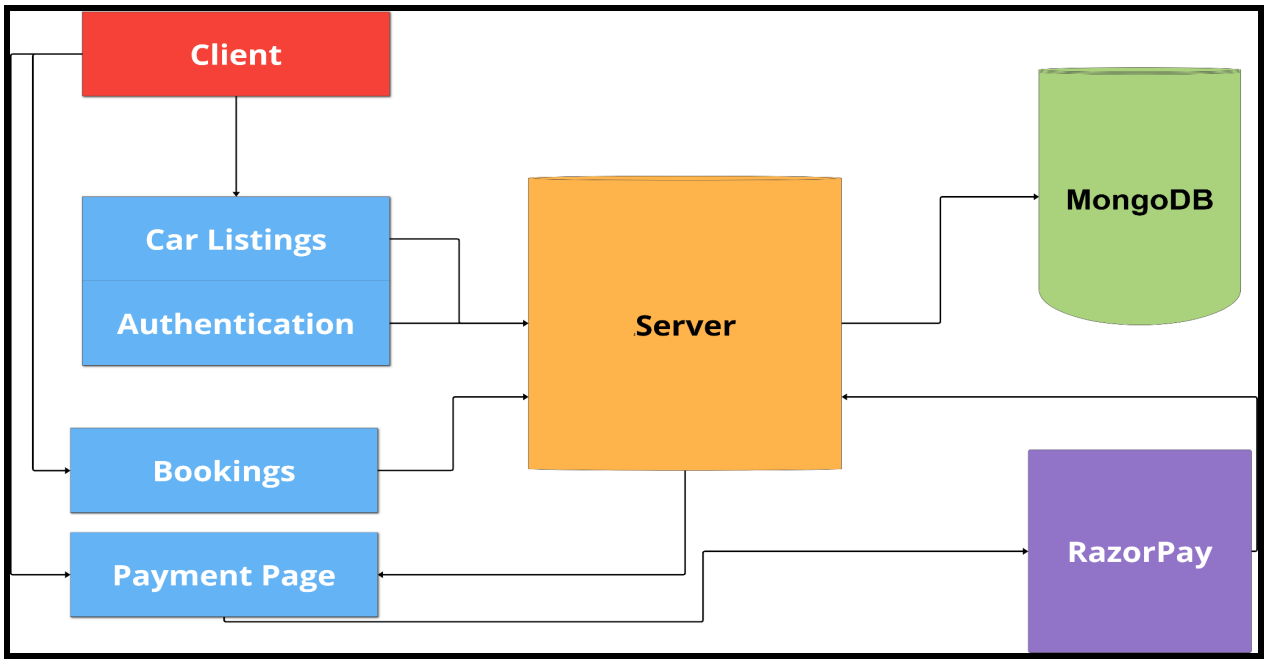
3. Frontend Design

- Primary Design

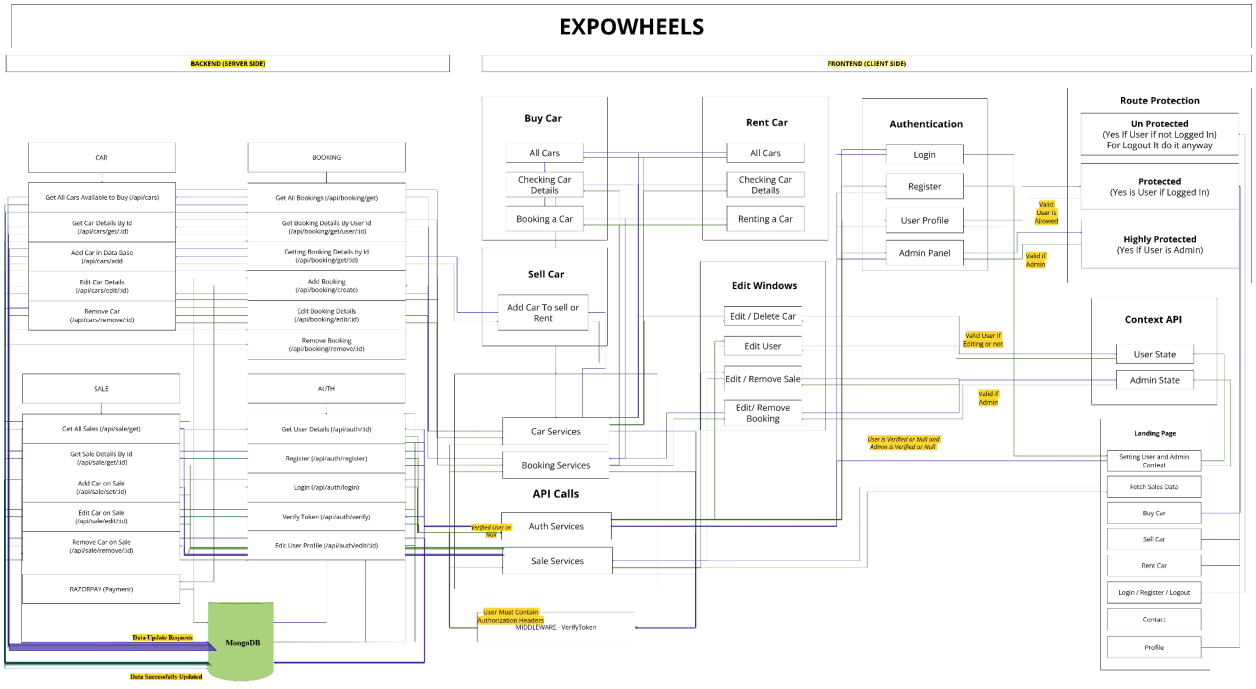


PROJECT WORKFLOW

1. Main Workflow



2. Project WorkFlow



DEPLOYMENT

The deployment strategy for *Expo Wheels* leverages modern cloud-based hosting solutions for the client (frontend), server (backend), and database. This approach ensures high availability, global reach, and seamless integration of all system components.

1. **Frontend Deployment (Client-Side)**
    - **Hosting Platform:**
      The frontend is hosted on **Vercel**, which offers:
        - Automatic builds and deployments from a Git repository (e.g., GitHub).
        - Global content delivery network (CDN) for fast content delivery.
        - Built-in HTTPS for secure connections.
    - **Advantages:**
        - Easy to manage for React-based applications.
        - Optimized for Single Page Applications (SPAs).
        - Minimal configuration required for deployment.

2. **Backend Deployment (Server-Side)**
    - **Hosting Platform:**
      The backend is deployed on **Render** for initial hosting, with plans to leverage **AWS EC2** for scalability in the future.
        - **Render** provides:
            - Simple setup for Node.js applications.
            - Free SSL certificates and custom domains.
            - Automatic scaling for variable workloads.
        - **AWS EC2** will enable:
            - Fine-grained control over instances for backend services.
            - High availability and redundancy with features like auto-scaling and load balancing.
    - **Configuration**:
        - Docker containers may be used to ensure consistent environments across Render and AWS.

3. **Database Deployment**
    - **Hosting Platform:**
      The database is hosted on **MongoDB Atlas**, a fully managed cloud database solution.
        - **Features**:
            - High availability with multi-region clusters.
            - Automated backups and restoration.
            - Built-in security features like IP whitelisting and encryption at rest.
        - **Configuration:**
            - Cluster tier selection will be based on current traffic and data size, with the ability to scale up as the platform grows.

4. **CI/CD Pipeline**

   To streamline development and deployment, a CI/CD pipeline will be implemented:

    - **Frontend:**
        - GitHub Actions will automate testing, building, and deploying the React application to Vercel.
    - **Backend:**
        - Render provides native CI/CD integration with GitHub for automated deployments upon code push.
        - AWS deployments will use tools like AWS CodePipeline or GitHub Actions.

5. **Security and Monitoring**
    - **Security Measures**:
        - Enable HTTPS for all frontend and backend connections.
        - Use environment variables to store sensitive data like API keys, database credentials, and JWT secrets.

- **Monitoring Tools**:
    - **Frontend**: Vercel's built-in analytics for performance tracking.
    - **Backend**: Tools like Datadog, New Relic, or AWS CloudWatch for monitoring server health and performance.
    - **Database**: MongoDB Atlas monitoring for query performance and resource usage.

6. **Disaster Recovery Plan**
    - Automated backups for MongoDB Atlas will ensure data can be restored in case of failure.
    - AWS EC2 instances will be configured with snapshots and recovery instances.
    - Render provides built-in redundancy for backend services to minimize downtime.

FUTURE ENHANCEMENTS

1. **Providing Real-Time Updates Using Mail Services**
    - Implement automated email notifications for actions like successful transactions, rental reminders, and promotional offers.
    - Users will receive real-time updates about their activities (e.g., booking status or vehicle availability).
2. **Notification and Customer Service Using Socket.IO**
    - Introduce real-time notifications for events such as new car listings, rental approvals, and chat messages.
    - Enable real-time customer support via a live chat system, ensuring quick resolution of user queries.
3. **Separate Organization Accounts for Direct Brand Connects**
    - Allow car manufacturers and dealerships to register as organizations, providing them with a dedicated portal to list vehicles directly.
    - Include bulk listing management and analytics dashboards for these accounts.

4. **Future Features**
    - Advanced Search Filters:
        - Enhance the search functionality to include dynamic filters like fuel type, mileage, and car reviews.
    - Car Inspection Reports:
        - Allow sellers to upload verified inspection reports for increased buyer confidence.
    - Vehicle Financing Options:
        - Integrate third-party financial services to offer loans directly on the platforms.
5. **Future Technology Enhancements**
    - **Using Socket.IO for Direct Communication**
        - Enable real-time communication between users and customer support agents.
        - Allow direct buyer-seller communication via chat for smoother transactions.
    - **Mail Services with Nodemailer**
        - Utilize **Nodemailer** to send transactional emails, promotional offers, and account-related notifications.
        - Add support for email templates to improve branding and user experience.
    - **MySQL Database**
        - Introduce **MySQL** as an auxiliary database for structured data (e.g., transactional records or logs).
        - Use it alongside MongoDB to implement hybrid database architecture, leveraging the strengths of both systems.

APPENDIX

Conclusion

Expo Wheels is a comprehensive MERN stack-based platform that simplifies the buying, selling, and renting of vehicles. With its user-centric design, advanced features, and seamless integration of modern technologies, the platform provides a secure, scalable, and efficient environment for all users. By focusing on transparency, real-time updates, and customer satisfaction, Expo Wheels bridges the gap between buyers, sellers, and renters, setting a new standard for automotive transactions.

References / Documentations

Frontend Technologies

- React
- React Router
- Tailwind CSS
- Framer Motion

Backend Technologies

- Node.js
- Express.js
- MongoDB
- Mongoose
- Bcrypt
- JWT
- Multer

Tools

- Postman

THANK YOU VERY MUCH !!