CAR MANAGEMENT APPLICATION

Overview

This project is a full-stack Car Management application that allows users to manage car listings, including adding, updating, deleting, and viewing car entries. The application consists of two parts: the frontend and the backend. The frontend is built using React, while the backend is developed with Node.js, Express, and MongoDB.

- **Frontend Demo**: https://mycarmanagement.netlify.app
- **Backend Demo**: https://car-management-backend-vquj.onrender.com
- **Frontend GitHub Repository**: https://github.com/Venkat5452/Car-Management-FrontEnd
- **Backend GitHub Repository**: https://github.com/your-username/car-management-app

---

Frontend Documentation

Description

The frontend of the Car Management application allows users to interact with car listings. It includes pages for user authentication (signup, login), managing car listings (add, update, delete), and viewing car details.

Features

- **User Authentication**:
    - Sign up, login, and password reset with OTP verification.
- **Car Management**:
    - Add, update, view, and delete car listings.
- **Responsive Interface**:
    - Mobile-friendly and user-friendly design.

Tech Stack

- **Frontend**: React, HTML, CSS, JavaScript
- **Backend**: Node.js, Express.js, MongoDB
- **State Management**: React Context API
- **HTTP Client**: Axios

Setup and Installation

Prerequisites

- Node.js and npm installed on your machine.

Installation Steps

1. Clone the repository:

   git clone https://github.com/Venkat5452/Car-Management-FrontEnd.git

   cd Car-Management-FrontEnd

2. Install dependencies:

   npm install

3. Update the Helper.js file with the backend URL:

   export const BASE_URL="https://car-management-backend-vquj.onrender.com";

4. Start the application:

   npm start

Folder Structure

```
MYCARFRONTEND/
├── build/                    # Production build output
├── node_modules/             # Dependencies
├── public/
│   └── index.html            # Main HTML template
├── src/
│   ├── Components/           # Reusable components
│   │   ├── AddCar/           # AddCar component
│   │   ├── CarDetails/       # CarDetails component
│   │   ├── Dashboard/        # Dashboard component
│   │   ├── Footer/           # Footer component
│   │   ├── ForgotPassword/   # ForgotPassword component
│   │   ├── Header/           # Header component
│   │   ├── Home/             # Home component
│   │   ├── Login/            # Login component
│   │   ├── Signup/           # Signup component
│   │   ├── UpdateCar/        # UpdateCar component
│   │   └── helper.js         # Helper functions
│   ├── App.css               # Main CSS for the app
│   ├── App.js                # Main application component
│   ├── App.test.js           # Test file for App component
│   ├── index.css             # Global CSS styles
│   ├── index.js              # Entry point of the application
│   ├── logo.svg              # Application logo
│   ├── reportWebVitals.js    # Performance reporting
│   └── setupTests.js         # Test setup
├── .gitattributes            # Git attributes
├── .gitignore                # Git ignore file
├── package-lock.json         # Lockfile for npm dependencies
├── package.json              # Project dependencies and scripts
└── README.md                 # Project documentation
```

BACKEND DOCUMENTATION

Description

The backend of the Car Management application is built using Node.js and Express. It handles user authentication, manages car listings, and performs email-based OTP verification for secure login and password resets.

Features

- **User Authentication**:
  - Allows registration, login, and password reset with OTP verification.
- **Car Management**:
  - Allows adding, updating, deleting, and viewing car entries.
- **OTP Verification**:
  - Sends OTPs via email for user registration and password reset.
- **Image Limiting**:
  - Limits the number of images uploaded per car entry to 10.

Tech Stack

- **Express**: Server framework to handle routes and middleware.
- **MongoDB**: Database for storing car listings and user data.
- **Mongoose**: ODM for MongoDB, used for managing data and models.
- **Nodemailer**: Used for sending OTP emails.
- **Bcrypt**: Used for hashing passwords.

Setup and Installation

Prerequisites

- Node.js
- MongoDB (Set up your own MongoDB database or use a cloud MongoDB provider)
- A Gmail account (for sending OTP emails)

Installation Steps

1. Clone the repository:

   git clone https://github.com/your-username/car-management-app.git

   cd car-management-app

2. Install dependencies:

   npm install

3. Set up environment variables:
   - Create a .env file in the root directory and add the following:
   - MYURL=your_mongodb_connection_string
   - EMAIL=your_email_address
   - PASSWORD=your_email_password

4. Start the server:

   node server.js

The server will run on http://localhost:9040.

API Endpoints

User Routes

- **POST** /login: Logs in a user with email and password.
- **POST** /signup: Registers a new user with name, email, password, and OTP.
- **POST** /verify-otp: Verifies OTP during password reset.
- **POST** /updatepassword: Updates the user's password after OTP verification.
- **POST** /makemail: Sends OTP for registration.
- **POST** /send-otp: Sends OTP for password reset.

Car Routes

- **POST** /addcar: Adds a new car entry with details like title, description, tags, and images.
- **GET** /getallcars: Retrieves all car entries.
- **GET** /api/car/:id: Retrieves details of a specific car by its ID.
- **DELETE** /deletecar/:id: Deletes a car entry by its ID.
- **PUT** /update-car/:id: Updates a car's details by its ID.

Dependencies

- **Express**: Server framework for handling routes and API logic.
- **Cors**: Middleware for handling cross-origin requests.
- **Mongoose**: MongoDB ODM for data management.
- **Nodemailer**: Email service for sending OTPs.
- **Bcrypt**: Password hashing and encryption.

Conclusion

This Car Management application provides a secure, scalable, and user-friendly platform for managing car listings and user authentication. The combination of React on the frontend and Node.js with Express and MongoDB on the backend offers a modern solution for users to manage their cars efficiently. The OTP-based authentication ensures secure user login and password management.