Documentation and Report

Overview:

The Car Vault app is a MERN stack application designed to manage and store car information. Users can add, view, update, and delete car entries. The app includes a secure backend, a responsive frontend, and seamless communication between the two.

Features:

- Add new car entries with details like make, model, year, etc.
- View a list of all stored cars.
- Update existing car information.
- Delete car entries.
- Secure user authentication.
- Responsive design for all devices.

Technologies Used:

- Frontend: React.js, HTML, CSS, JavaScript
- Backend: Node.js, Express.js
- Database: MongoDB
- State Management: Context API (optional)
- Security: JWT (JSON Web Tokens), bcryptjs
- API Integration: Axios or Fetch API

Car Vault App - Backend Documentation

Overview

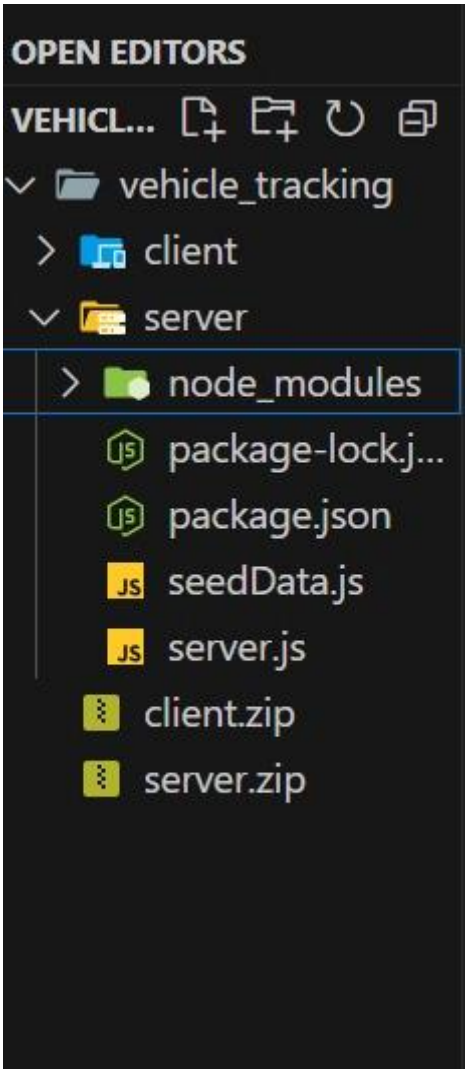
The backend of the Car Vault App is built using Node.js with Express.js. It connects to a MongoDB database using Mongoose to perform CRUD (Create, Read, Update, Delete) operations on car data. This documentation provides an overview of the backend setup, API routes, and how to use the backend service.

Prerequisites

Before setting up the backend, ensure you have the following installed on your system:

- Node.js (v14 or later)
- npm or yarn (for package management)
- MongoDB (for the database)

Project Structure



Step 1: Set Up the Backend Directory

1. **Create Backend Directory:**
    - In your project root directory, create a folder named server.
      mkdir server
      cd server

2. Initialize the Express Application:

- Initialize a new Node.js project inside the server directory.
  npm init -y

Step 2: Install Required Dependencies

1. **Install Backend Dependencies:**

   npm install express mongoose body-parser cors nodemon
2. **Verify package.json:**
   Your package.json should include the following dependencies:

```
"dependencies": {
"body-parser": "^1.20.2",
"cors": "^2.8.5",
"express": "^4.18.2",
"mongoose": "^8.0.3",
"nodemon": "^3.0.2"
}
```

Example: Create the required files and add the below code.

//server.js

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const seedData = require('./seedData');
const cors = require('cors');
const app = express();
app.use(cors());
const PORT = process.env.PORT || 3001;
// Connect to MongoDB (make sure MongoDB is running)
mongoose.connect('mongodb://localhost:27017/vehicle-tracking', {
useNewUrlParser: true,
useUnifiedTopology: true,
});
// Middleware
app.use(bodyParser.json());
// MongoDB Schema for Car
const carSchema = new mongoose.Schema({
companyName: String,
distanceCovered: Number,
mileage: Number,
serviceDates: [Date],
owner: {
name: String,
email: String,
},
image: String,
});
```

```
const Car = mongoose.model('Car', carSchema);

// Function to seed data

const seedDatabase = async () => {

try {
const existingCars = await Car.find();
```

```
    if (existingCars.length > 0) {
    // If there are existing cars, delete them
    await Car.deleteMany();
    console.log('Existing cars deleted.');
    }
// Seed the database with new data
    await Car.create(seedData);
    console.log('Seed data added to the database.');
    } catch (error) {
    console.error('Error seeding the database:', error);
    }
    };

    // Call the seedDatabase function when the server starts
    seedDatabase();

    // API Endpoints
    // Get all cars
    app.get('/api/cars', async (req, res) => {
    try {
    const cars = await Car.find();
    res.json(cars);
    } catch (error) {
    res.status(500).json({ error: 'Internal Server Error' });
    }
    });

    // Get a specific car by ID
    app.get('/api/cars/:id', async (req, res) => {
    try {
    const car = await Car.findById(req.params.id);
    if (!car) {
    return res.status(404).json({ error: 'Car not found' });
    }
    res.json(car);
    } catch (error) {
    res.status(500).json({ error: 'Internal Server Error' });
    }
    });

    // Add a new car
    app.post('/api/cars', async (req, res) => {
```

```
try {
const newCar = await Car.create(req.body);
res.status(201).json(newCar);
} catch (error) {
res.status(400).json({ error: 'Bad Request' });
}
});

// Update a car by ID
app.put('/api/cars/:id', async (req, res) => {
try {
const updatedCar = await Car.findByIdAndUpdate(
req.params.id,
req.body,
{ new: true }
);
if (!updatedCar) {
return res.status(404).json({ error: 'Car not found' });
}
res.json(updatedCar);
} catch (error) {
res.status(400).json({ error: 'Bad Request' });
}
});

// Delete a car by ID
app.delete('/api/cars/:id', async (req, res) => {
try {
const deletedCar = await Car.findByIdAndDelete(req.params.id);
if (!deletedCar) {
return res.status(404).json({ error: 'Car not found' });
}
console.log('Car deleted successfully');
res.json({ message: 'Car deleted successfully' });
} catch (error) {
res.status(500).json({ error: 'Internal Server Error' });
}
});

// Start the server
app.listen(PORT, () => {
console.log(`Server is running on http://localhost:${PORT}`);
});
```

// seedData.js

```
module.exports = [
{
companyName: 'Toyota',
distanceCovered: 10000,
mileage: 25,
serviceDates: [new Date('2022-01-15'), new Date('2022-03-20')],
owner: {
name: 'John Doe',
email: 'john.doe@example.com',
},
image: 'https://media.geeksforgeeks.org/wp-
content/uploads/20240122182422/images1.jpg',
},
{
companyName: 'Honda',
distanceCovered: 8000,
mileage: 28,
serviceDates: [new Date('2022-02-10'), new Date('2022-04-25')],
owner: {
name: 'Jane Smith',
email: 'jane.smith@example.com',
},
image: 'https://media.geeksforgeeks.org/wp-
content/uploads/20240122184958/images2.jpg',
},
{
companyName: 'Volkswagen',
distanceCovered: 10500,
mileage: 26,
serviceDates: [new Date('2022-10-05'), new Date('2022-12-15')],
owner: {
name: 'Ava Anderson',
email: 'ava.anderson@example.com',
},
image: 'https://media.geeksforgeeks.org/wp-
content/uploads/20240122184958/images2.jpg',
},
{
companyName: 'Toyota',
distanceCovered: 10000,
mileage: 25,
serviceDates: [new Date('2022-01-15'), new Date('2022-03-20')],
owner: {
name: 'John Doe',
```

```
email: 'john.doe@example.com',
},
image: 'https://media.geeksforgeeks.org/wp-
content/uploads/20240122185312/images3.jpg',
},
{
companyName: 'Honda',
distanceCovered: 8000,
mileage: 28,
serviceDates: [new Date('2022-02-10'), new Date('2022-04-25')],
owner: {
name: 'Jane Smith',
email: 'jane.smith@example.com',
},
image: 'https://media.geeksforgeeks.org/wp-
content/uploads/20240122185312/images3.jpg',
},
{
companyName: 'Volkswagen',
distanceCovered: 10500,
mileage: 26,
serviceDates: [new Date('2022-10-05'), new Date('2022-12-15')],
owner: {
name: 'Ava Anderson',
email: 'ava.anderson@example.com',
},
image: 'https://media.geeksforgeeks.org/wp-
content/uploads/20240122185312/images3.jpg',
},
];
```

Step 3: To start the server run the following command.

```
nodemon server.js
```

Car Vault App - Frontend Documentation

Overview

The Car Vault App is a web application that allows users to store and manage information about their cars securely. The app's frontend is built using React, and it communicates with a backend API to perform CRUD (Create, Read, Update, Delete) operations. This document provides a detailed guide to the frontend structure, components, and how to use the app.

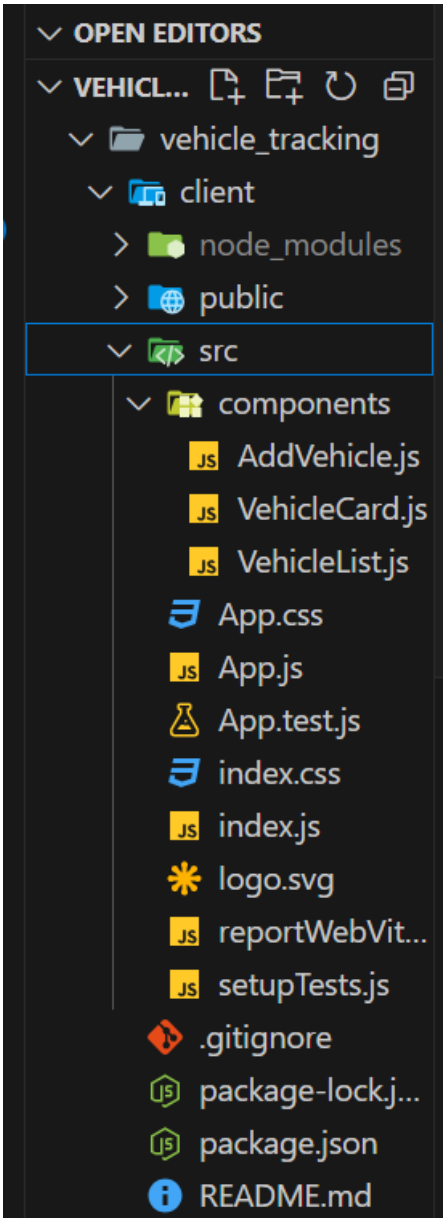
Prerequisites

Before setting up the frontend, ensure you have the following installed on your system:

- Node.js (v14 or later)

- npm or yarn (for package management)
- React Developer Tools (browser extension for easier debugging)

Project Structure



Step 1: Set Up the Frontend Directory

1. **Open VS Code:**
    - Open your terminal in VS Code.
    - Create a new folder for your project (if you haven't already).
    - Open the root directory of your project in VS Code.
2. **Create the Frontend Directory:**
    - In the root directory, create a new folder called client.

    mkdir client

    cd client

3. Initialize the React Application:

Inside the client directory, create a new React app using Create React App.

npx create-react-app

Step 2: Install Required Dependencies

1. **Install Frontend Dependencies:**
   - Install the necessary packages for building the frontend API.

   npm install axios

2. **Verify package.json:**

- Your package.json file should have the following dependencies:

```
"dependencies": {
   "@testing-library/jest-dom": "^5.17.0",
   "@testing-library/react": "^13.4.0",
   "@testing-library/user-event": "^13.5.0",
   "axios": "^1.6.3",
   "moment": "^2.30.1",
   "react": "^18.2.0",
   "react-dom": "^18.2.0",
   "react-scripts": "5.0.1",
   "web-vitals": "^2.1.4"
 };
```

Example: Create the required files and add the below code.

/* Src/App.css */

```
.vehicle-list {
display: flex;
flex-wrap: wrap;
gap: 20px;
}
.vehicle-card {
border: 1px solid #ddd;
border-radius: 18px;
padding: 15px;
width: -moz-fit-content;
width: fit-content;
box-shadow: 0 14px 18px rgba(0, 0, 0, 0.1);
background-color: #fff;
}
```

```
.vehicle-card h3 {
margin-bottom: 10px;
}
.vehicle-card button {
background-color: #007BFF;
color: #fff;
border: none;
padding: 8px;
cursor: pointer;
border-radius: 4px;
}
.vehicle-card button:hover {
background-color: #0056b3;
}
img {
height: 200px;
width: 300px;
border-radius: 10px;
margin-bottom: 10px;
}
.list-container {
display: flex;
flex-wrap: wrap;
justify-content: center;
gap: 15px;
overflow-x: hidden;
}

h2 {
text-align: center;
}
.vehicle-card {
border: 1px solid #d3c1c1;
```

```
border-radius: 10px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
padding: 20px;
margin: 20px;
width: 300px;
background-image: url('https://img.pikbest.com/backgrounds/20191019/multi-color-gradient-background-v_1576675jpg!bwr800 ');
}
.vehicle-card h2 {
margin-bottom: 15px;
color: #333;
}
.vehicle-card label {
display: block;
margin-bottom: 10px;
color: #333333;
}
.vehicle-card input {
width: 100%;
padding: 8px;
margin-bottom: 10px;
box-sizing: border-box;
}
.vehicle-card button {
background-color:orange;
color: #fff;
border: none;
padding: 10px;
cursor: pointer;
border-radius: 4px;
}
.vehicle-card button:hover {
background-color: #0056b3;
```

```
}
.form-container {
max-width: 300px;
margin: 20px auto;
padding: 20px;
background-color: #c5ebed;
border-radius: 8px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.form-row {
display: flex;
gap: 20px;
margin-bottom: 15px;
}
.form-row label {
flex: 1;
}
.form-row input {
flex: 2;
padding: 8px;
width: 100%;
box-sizing: border-box;
}
button {
background-color:orange;
color: #fff;
border: none;
padding: 10px;
cursor: pointer;
border-radius: 4px;
}
button:hover {
background-color: #0056b3;
```

```
}
form {
padding: 10px;
}
.gfg {
background-color: green;
text-align: center;
color: white;
padding: 15px;
border-radius: 10px;
margin-bottom: -20px;
}
.list {
margin-top: -50px;
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
}
.vehicle-image {
width: 100%;
border-radius: 10px;
transition: transform 0.3s ease-in-out;
}
.vehicle-image:hover {
transform: scale(1.1);
}
.button-container {
display: flex;
justify-content: space-evenly;
margin-top: 15px;
}
.main-container {
```

```
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
}
body {
background-color: #e8f5e9;
background-image: url('https://wallpapercave.com/wp/wp2869302.jpg');
background-size: cover;
background-repeat: no-repeat;
}
.card {
box-shadow: 0 4px 8px rgba(58, 55, 55, 0.1);
}
h1{
color:rgb(14, 213, 193);
}
```

// src/App.js

```
import React, { useState, useEffect } from "react";
import VehicleList from "./components/VehicleList";
import AddVehicle from "./components/AddVehicle";
import axios from "axios";
import "./App.css";
const App = () => {
const [vehicles, setVehicles] = useState([]);
const [showForm, setShowForm] = useState(false);
useEffect(() => {
axios
.get("http://localhost:3001/api/cars")
.then((response) => setVehicles(response.data))
.catch((error) => console.error("Error fetching vehicles:", error));
}, []);
const handleAddVehicle = (newVehicle) => {
```

```
console.log("Adding new vehicle:", newVehicle);
setVehicles((prevVehicles) => [...prevVehicles, newVehicle]);
};
const handleContactOwner = (email) => {
alert(`Contacting the owner of the vehicle at ${email}`);
};
const handleDeleteVehicle = (vehicleId) => {
console.log(`Deleting vehicle with ID: ${vehicleId}`);
axios
.delete(`http://localhost:3001/api/cars/${vehicleId}`)
.then(() => {
setVehicles((prevVehicles) =>
prevVehicles.filter((vehicle) => vehicle._id !== vehicleId)
);
})
.catch((error) => console.error("Error deleting vehicle:", error));
};
const handleUpdateVehicle = async (updatedVehicle) => {
try {
const response = await axios.put(
`http://localhost:3001/api/cars/${updatedVehicle._id}`,
updatedVehicle
);
console.log("Vehicle updated successfully:", response.data);
setVehicles((prevVehicles) =>
prevVehicles.map((vehicle) =>
vehicle._id === updatedVehicle._id ? response.data : vehicle
)
);
} catch (error) {
console.error("Error updating vehicle:", error);
}
};
```

```
return (
<div className="main-container">
<div style={{ display: "flex", alignItems: "center", justifyContent: "space-between" }}>
<a href="https://www.google.com/maps" target="_blank" rel="noopener noreferrer">
<img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS7Z_eDt4CehABV6bfkWWOW3bDjiC_6WVrWUQ&s" alt="Logo" style={{ width: "50px", height: "50px", marginLeft: "50px",marginRight:"10px" }} />
</a>
<h1 style={{ flexGrow: 1, textAlign: "center", margin: 0 }}>Havaki Vault</h1>
</div>
<button onClick={() => setShowForm(!showForm)}>
{showForm ? "Close" : "Add New Vehicle"}
</button>
<div>
{showForm && <AddVehicle onAddVehicle={handleAddVehicle} />}
<VehicleList
vehicles={vehicles}
onDeleteVehicle={handleDeleteVehicle}
onUpdateVehicle={handleUpdateVehicle}
onContactOwner={handleContactOwner}
/>
</div>
</div>
);
};
export default App;
```

//src/components/AddVehicle.js

```
import axios from "axios";
import React, { useState } from "react";
const AddVehicle = ({ onAddVehicle }) => {
const [newVehicle, setNewVehicle] = useState({
companyName: "",
distanceCovered: "",
```

```
mileage: "",
serviceDates: "",
owner: {
name: "",
email: "",
},
image: "",
});
const handleAddVehicle = () => {
// Submit a new vehicle
axios
.post("http://localhost:3001/api/cars", newVehicle)
.then((response) => {
// Notify the parent component about the new vehicle
onAddVehicle(response.data);
// Clear the newVehicle state for the next entry
setNewVehicle({
companyName: "",
distanceCovered: "",
mileage: "",
serviceDates: [],
owner: {
name: "",
email: "",
},
image: "", // Reset image URL field
});
})
.catch((error) => console.error(error));
};
return (
<div className="form-container">
<h2 style={{ color: "#007BFF", textAlign: "center" }}>
```

```
Add a New Vehicle
</h2>
<form
onSubmit={(e) => {
e.preventDefault();
handleAddVehicle();
}}
>
<div className="form-row">
<label>
Company Name:
<input
type="text"
value={newVehicle.companyName}
onChange={(e) =>
setNewVehicle({
...newVehicle,
companyName: e.target.value,
})
}
required
className="form-input"
/>
</label>
<label>
Distance Covered:
<input
type="number"
value={newVehicle.distanceCovered}
onChange={(e) =>
setNewVehicle({
...newVehicle,
distanceCovered: e.target.value,
```

```
})
}
required
className="form-input"
/>
</label>
</div>
<div className="form-row">
<label>
Mileage:
<input
type="number"
value={newVehicle.mileage}
onChange={(e) =>
setNewVehicle({
...newVehicle,
mileage: e.target.value,
})
}
required
className="form-input"
/>
</label>
<label>
Service Dates (comma-separated):
<input
type="text"
value={newVehicle.serviceDates}
onChange={(e) =>
setNewVehicle({
...newVehicle,
serviceDates: e.target.value,
})
```

```
}
required
className="form-input"
/>
</label>
</div>
<div className="form-row">
<label>
Owner Name:
<input
type="text"
value={newVehicle.owner.name}
onChange={(e) =>
setNewVehicle({
...newVehicle,
owner: {
...newVehicle.owner,
name: e.target.value,
},
})
}
required
className="form-input"
/>
</label>
<label>
Owner Email:
<input
type="email"
value={newVehicle.owner.email}
onChange={(e) =>
setNewVehicle({
...newVehicle,
```

```
owner: {
...newVehicle.owner,
email: e.target.value,
},
})
}
required
className="form-input"
/>
</label>
</div>
<div className="form-row">
<label>
Image URL:
<input
type="text"
value={newVehicle.image}
onChange={(e) =>
setNewVehicle({
...newVehicle,
image: e.target.value,
})
}
className="form-input"
/>
</label>
</div>
<button type="submit" className="form-button">
Add Vehicle
</button>
</form>
</div>
);
```

```
};

export default AddVehicle;
// src/components/VehicleList.js
import React, { useState, useMemo } from "react";
import VehicleCard from "./VehicleCard";
const VehicleList = ({
vehicles,
onContactOwner,
onDeleteVehicle,
onUpdateVehicle,
}) => {
const [companyFilter, setCompanyFilter] = useState("");
const [sortBy, setSortBy] = useState("distanceCovered"); // Default sorting by
distanceCovered
const filteredAndSortedVehicles = useMemo(() => {
return vehicles
.filter((vehicle) =>
vehicle.companyName
.toLowerCase()
.includes(companyFilter.toLowerCase())
)
.sort((a, b) => (a[sortBy] > b[sortBy] ? 1 : -1));
}, [vehicles, companyFilter, sortBy]);
return (
<div className="list" style={{ marginTop: "20px" }}>
<h2 style={{ color: "orange" }}>Vehicle List</h2>

{/* Filter and Sort Controls */}
<div style={{ marginBottom: "10px", fontSize: "25px" }}>
<label style={{ color: "white", fontWeight: "bold" }}>
Filter by Company Name:
```

```
<input
type="text"
value={companyFilter}
onChange={(e) => setCompanyFilter(e.target.value)}
style={{ padding: "5px", borderRadius: "4px", border: "1px solid #ccc" }}
/>
</label>
<label style={{ marginLeft: "15px", color: "white", fontSize: "18px", fontFamily: "Arial, sans-serif", fontWeight: "bold" }}>
Sort by:
<select
value={sortBy}
onChange={(e) => setSortBy(e.target.value)}
style={{ padding: "8px", borderRadius: "4px", border: "1px solid #ccc", fontSize: "16px", fontFamily: "Arial, sans-serif" }}
>
<option value="distanceCovered" style={{ fontSize: "16px", fontFamily: "Arial, sans-serif" }}>
Distance Covered
</option>
<option value="mileage" style={{ fontSize: "16px", fontFamily: "Arial, sans-serif" }}>
Mileage
</option>
</select>
</label>
</div>
<div className="list-container">
{filteredAndSortedVehicles.map((vehicle) => (
<VehicleCard
key={vehicle._id}
vehicle={vehicle}
onContactOwner={onContactOwner}
```

```
onDeleteVehicle={onDeleteVehicle}
onUpdateVehicle={onUpdateVehicle}
/>
))}
</div>
</div>
);
};
export default VehicleList;
// src/components/VehicleCard.js
import React, { useState } from "react";
import moment from "moment";
const VehicleCard = ({
vehicle,
onContactOwner,
onDeleteVehicle,
onUpdateVehicle,
}) => {
const [isEditing, setIsEditing] = useState(false);
const [updatedVehicle, setUpdatedVehicle] = useState(vehicle);
const handleUpdateClick = () => {
setIsEditing(true);
};
const handleCancelClick = () => {
setIsEditing(false);
setUpdatedVehicle(vehicle); // Reset to original values
};
const handleSaveClick = () => {
// Implement the logic to save the updated details
onUpdateVehicle(updatedVehicle);
setIsEditing(false);
```

```
};
const handleInputChange = (fieldName, value) => {
const [field, subField] = fieldName.split(".");
setUpdatedVehicle((prevVehicle) => ({
...prevVehicle,
[field]: subField
? { ...prevVehicle[field], [subField]: value }
: value,
}));
};
return (
<div className="vehicle-card">
{isEditing ? (
// Render editable fields for updating details
<div>
<label>
Company Name:
<input
type="text"
value={updatedVehicle.companyName}
onChange={(e) =>
handleInputChange("companyName", e.target.value)
}
required
/>
</label>
<label>
Distance Covered:
<input
type="number"
value={updatedVehicle.distanceCovered}
```

```
onChange={(e) =>
handleInputChange(
"distanceCovered",
e.target.value
)
}
required
/>
</label>
<label>
Mileage:
<input
type="number"
value={updatedVehicle.mileage}
onChange={(e) =>
handleInputChange("mileage", e.target.value)
}
required
/>
</label>
<label>
Owner Name:
<input
type="text"
value={updatedVehicle.owner.name}
onChange={(e) =>
handleInputChange("owner.name", e.target.value)
}
required
/>
</label>
```

```
<label>
Owner Email:
<input
type="email"
value={updatedVehicle.owner.email}
onChange={(e) =>
handleInputChange("owner.email", e.target.value)
}
required
/>
</label>
{/* Add input fields for other properties like
owner name, owner email, and image */}
<button
onClick={handleSaveClick}
style={{
padding: "10px 20px",
marginRight: "10px",
backgroundColor: "#4CAF50", // optional: add background color
color: "#fff", // optional: change text color
border: "none", // optional: remove border
borderRadius: "4px", // optional: round the corners
}}
> Save
</button>
<button
onClick={handleCancelClick}
style={{
padding: "10px 20px",
backgroundColor: "#f44336", // optional: add background color
color: "#fff", // optional: change text color
```

```
border: "none", // optional: remove border
borderRadius: "4px", // optional: round the corners
}}
>
Cancel
</button>
</div>
) : (
// Display details
<div>
<h3 style={{ fontWeight: "bold" }}>
{vehicle.companyName}
</h3>
<p>
<span style={{ fontWeight: "bold" }}>
Distance Covered:
</span>{" "}
{vehicle.distanceCovered}
</p>
<p>
<span style={{ fontWeight: "bold" }}>Mileage:
</span>{" "}
{vehicle.mileage}
</p>
{vehicle.owner && (
<div>
<p style={{ fontWeight: "bold" }}>Owner:</p>
<ul style={{ listStyle: "none", padding: 0 }}>
<li>{vehicle.owner.name}</li>
<li>{vehicle.owner.email}</li>
</ul>
```

```
</div>
)}
{vehicle.image && (
<div className="image-container">
<img
src={vehicle.image}
alt={vehicle.companyName}
className="vehicle-image"
/>
</div>
)}
<p>Service Dates:</p>
<ul>
{vehicle.serviceDates.map((item, i) => (
<li key={i}>
{moment(item).format("MMMM D, YYYY")}
</li>
))}
</ul>
<div className="button-container">
<button
onClick={() =>
onContactOwner(
vehicle.owner ? vehicle.owner.email : ""
)
}
>
Contact Owner
</button>
<button onClick={() => onDeleteVehicle(vehicle._id)}>
Delete Vehicle
```

```
</button>
<button onClick={handleUpdateClick}>Update</button>
</div>
</div>
)}
</div>
);
};
export default VehicleCard;
```

Step 6: To start the frontend run the following command.

```
npm start
```