**Backend:**

The backend of the Rentify application is responsible for handling data storage, business logic, and serving data to the frontend. It is built using Node.js with Express.js and MongoDB as the database.

Backend Deployed on Render.com: [*[Link to Deployed API on Render]*](https://rentify-akhilesh.netlify.app/)

**Technologies Used:**

* **Node.js**: A JavaScript runtime environment used to build scalable and efficient server-side applications.
* **Express.js**: A web application framework for Node.js used to build APIs and handle HTTP requests.
* **MongoDB**: A NoSQL database used for storing application data in a flexible, JSON-like format.
* **Mongoose**: An Object Data Modeling (ODM) library for MongoDB and Node.js, providing a straightforward schema-based solution to model application data.

**Components and Features:**

1. **API Endpoints**: Define routes and endpoints to handle various operations such as user authentication, property management, property exploration, etc.
2. **Authentication Middleware**: Middleware functions to handle user authentication using JSON Web Tokens (JWT).
3. **Database Models**: Define Mongoose models to represent application data entities such as users, properties, etc.
4. **Controllers**: Implement controller functions to handle incoming requests, interact with the database, and send appropriate responses.
5. **Error Handling**: Implement error handling middleware to handle errors and send appropriate error responses.
6. **Middleware**: Implement middleware functions for logging, request validation, etc.
7. **Security**: Implement security measures such as input validation, password hashing, etc., to protect against common web vulnerabilities.

**Instructions:**

**1. Setting Up the Backend Environment:**

* Install Node.js and npm (Node Package Manager) on the system if not already installed.
* Create a new directory for the backend code and navigate into it using the command line.
* Initialize a new Node.js project by running npm init -y.

**2. Installing Dependencies:**

* Install required dependencies like Express.js, Mongoose, etc., by running:

npm install express mongoose dotenv jsonwebtoken bcryptjs cors

**3. Setting Up MongoDB:**

* Install MongoDB on the system if not already installed.
* Create a new MongoDB database for the Rentify application.

**4. Building the Backend:**

* Create necessary directories like models, controllers, middlewares, etc., for organizing the backend code.
* Define Mongoose models for users, properties, etc., in the models directory.
* Implement Express.js routes and controllers to handle CRUD operations for users, properties, etc.
* Set up middleware functions for user authentication, error handling, etc.
* Configure environment variables using a .env file for sensitive data like database connection strings, JWT secret, etc.

**5. Testing the Backend:**

* Use tools like Postman or curl to test the backend API endpoints.
* Verify that user authentication, property management, and other functionalities work as expected.

**6. Deployment:**

* Deploy the backend application to a hosting service like Heroku, AWS, or Azure.
* Set up continuous integration and deployment pipelines if required.

**Frontend:**

The frontend of the Rentify application is responsible for presenting the user interface and interacting with users. It is built using React.js, a popular JavaScript library for building user interfaces.

Frontend Deployed on Netlify.com: [*[Link to Live Site]*](https://rentify-akhilesh.netlify.app/)

**Technologies Used:**

* **React.js**: A JavaScript library for building user interfaces, developed by Facebook.
* **React Router**: A routing library for React applications, used for managing navigation and rendering different components based on the URL.
* **Material-UI**: A popular React UI framework that provides pre-designed components and styling for building responsive and visually appealing user interfaces.
* **Axios**: A promise-based HTTP client for making AJAX requests to the backend server.
* **React Context API / Redux**: State management libraries used to manage application state and share data between components.

**Components and Features:**

1. **User Authentication**: Login and registration forms for users to authenticate and create accounts.
2. **Dashboard**: Separate dashboards for sellers and clients to manage properties and explore available properties, respectively.
3. **Property Management**: Allow sellers to add, update, and delete properties, including features such as image upload, property details, etc.
4. **Property Exploration**: Allow clients to search and explore available properties based on various filters such as location, price, etc.
5. **User Interactions**: Implement features such as liking properties, showing interest, etc., using buttons and interactive components.
6. **Form Validation**: Client-side form validation to ensure that user inputs are correct before submitting data to the server.
7. **Error Handling**: Display appropriate error messages to users in case of invalid inputs, network errors, etc.

**Instructions:**

**1. Setting Up the Frontend Environment:**

* Install Node.js and npm if not already installed.
* Create a new directory for the frontend code and navigate into it using the command line.
* Initialize a new React.js project by running npx create-react-app rentify-frontend.

**2. Installing Dependencies:**

* Install required dependencies like React Router, Material-UI, Axios, etc., by running:

npm install @mui/material @emotion/react @emotion/styled react-router-dom axios

**3. Building the Frontend:**

* Organize the frontend code into components, pages, stylesheets, etc.
* Implement user authentication components like login, registration forms, etc.
* Create separate dashboards for sellers and clients with respective functionalities.
* Develop components for property management, property exploration, user interactions, form validation, error handling, etc.
* Use React Router for managing navigation between different pages and components.

**4. Styling and UI Design:**

* Utilize Material-UI components and theming for building a responsive and visually appealing user interface.
* Customize styles and layouts as per the application requirements.

**5. Testing the Frontend:**

* Test the frontend application in different browsers and devices to ensure cross-browser compatibility and responsiveness.
* Verify that user authentication, property management, property exploration, and other functionalities work smoothly.

**6. Deployment:**

* Build the frontend application for production using npm run build.
* Deploy the built files to a hosting service like Netlify, Vercel, GitHub Pages, etc.
* Set up custom domains, SSL certificates, and other configurations as required.