Project Report file

On

Ashiana: The Renting Home Near You

Submitted in the partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

In

Computer Science & Artificial Intelligence

Batch (2022-2026)

Submitted by:
Priyanshu Sharma
12200974

Submitted to:
Dr. Ridhi kapoor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING DAV UNIVERSITY

JALANDHAR-PATHANKOT NATIONAL HIGHWAY, NH44, SARMASTPUR PUNJAB-144012

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project guide, Dr. Ridhi Kapoor, for their valuable guidance, constructive feedback, and continuous support throughout the duration of this project. I am also thankful to Dr. Rahul Hans and the Department of engineering for providing the necessary resources and an encouraging academic environment.My appreciation extends to all faculty members and classmates whose suggestions and cooperation helped strengthen the quality of this work. Lastly, I am deeply grateful to my family for their constant encouragement and support, which motivated me to complete this project successfully.

DECLARATION

I, Priyanshu Sharma hereby declare that the work which is being presented in this project titled **"Ashiana: The Renting Home Near You"** by me, in partial fulfilment of the requirements for the award of Bachelor of Technology **(B. Tech**) Degree in "Computer Science & Artificial Intelligence" is an authentic record of my own work carried out under the guidance of **DR. Ridhi Kapoor**. To the best of my knowledge, the matter embodied in this report has not been submitted to any other University/ Institute for the award of any degree or diploma.

ABSTRACT

This project report contains the work that I have done for **Ashiana: The Renting Home Near You** based on the problems that are faced by our industries and factories. The motive of this project is to help the Industries and Factories in managing their data in a very systematic way. I have used the basic to advanced level concepts of Web Development and Mongo DB to solve the problems that are generally faced in Industries and Factories and provided the platform for managing the data of machines and their services. Even the service scheduled facility is available.

Table Of Content

INTRODUCTION

1) Introduction of Project Report

The project report is a document prepared by experts that contains all information regarding the proposed project. It is served as a blueprint of all operations. The project report is the business plan of action and clearly describes its goals and objectives .It helps in transforming the business idea into a productive venture without any confusion as it defines strategies for project execution.

Project Reports are important tools to both project teams and stake holders. Through these reports, we track the current progress of the project and compare it against the original plan. They can identify risks early on, and take corrective action. Reports estimate all costs of operations and possible profitability of the proposed project.

Here are the points that validate the Importance of Project Report:

- Project reports are an important source for managers and stakeholders, to monitor the current progress and measure against the original schedule.
- It helps to predict the threats and develop proper steps to recover.
- There port makes it easier to control the co-stand budget apart from the budgeted cost.
- It will be a source of information to respond to success, stagnation, team results, or quality of work.
- The project report requires completeness and accuracy, also ensures coverage of all dimensions of the project, makes the data more viable.
- It helps the project manage to deal with potential or upcoming risks during.

PROJECT TITLE AND OVERVIEW.

1) Project Title
Ashiana: The Renting Home Near You

Project Overview-

The House Renting System offers a centralized platform that connects landlords and tenants. The system includes features such as:

- **Property Listings:** Add, view, and manage rental properties.
- **Search and Filters:** Easy navigation through listings based on location, price, and property type.
- **Authentication:** Secure login and registration for tenants and landlords.
- **Communication:** Built-in messaging for inquiries and updates.
- **Responsive Design:** Compatibility with mobile, tablet, and desktop devices.

OBJECTIVE

The main objectives of the project are:

1. To provide a secure and efficient platform for house renting.
2. To simplify the process of listing and renting properties.
3. To ensure scalability and adaptability for future requirements.

PROGRAMMING LANGUAGES USED

1. MongoDB (Database):

- Data Storage: Stores data in flexible JSON-like documents. This is different from relational databases (like MySQL or PostgreSQL) which use tables and rows. The flexibility is advantageous when dealing with evolving data structures or semi-structured data.
- Scalability: MongoDB is designed for horizontal scalability. You can easily add more servers to handle increased data volume and traffic.
- NoSQL: It's a NoSQL database, meaning it doesn't enforce strict schema definitions. This makes development faster and more 1. MongoDB (Database):
- Data Storage: Stores data in flexible JSON-like documents. This is different from relational databases (like MySQL or PostgreSQL) which use tables and rows. The flexibility is advantageous when dealing with evolving data structures or semi-structured data.

2. Express.js (Backend Framework):

- **Server-Side Logic:** Handles requests from the client (React frontend), performs server-side operations (e.g., database interactions, data processing, authentication), and sends responses back to the client.
- **Routing:** Defines how different URLs map to specific functions or handlers.

- **Middleware:** Allows you to add functionalities to your request-response cycle (e.g., authentication, logging, data validation).
- **APIs (Application Programming Interfaces):** Express.js is used to create RESTful APIs, which provide a standardized way for the frontend (React) to communicate with the backend (Node.js/Express.js).

3. React (Frontend Framework):

- **User Interface (UI):** Builds interactive and dynamic user interfaces.
- **Component-Based Architecture:** Breaks down the UI into reusable components, making the code organized, maintainable, and easier to test.
- **Virtual DOM:** Optimizes UI updates by comparing changes in the virtual DOM (a representation of the actual DOM) before updating the actual DOM. This improves performance significantly.
- **JSX:** Uses JSX (JavaScript XML), a syntax extension to JavaScript, that allows you to write HTML-like code within your JavaScript. This makes it easier to create and manage UI elements.

4. Node.js (JavaScript Runtime Environment):

- **Server-Side JavaScript:** Allows you to run JavaScript code on the server. This is crucial for running Express.js.
- **Non-Blocking, Event-Driven Architecture:** Node.js is designed to handle multiple requests concurrently without blocking the main thread. This makes it highly efficient for handling real-time applications and I/O-bound operations.
- **npm (Node Package Manager):** Provides access to a vast ecosystem of JavaScript libraries and packages, simplifying development by allowing you to easily integrate pre-built functionalities into your application.

Advantages of the MERN Stack:

- **JavaScript Everywhere:** Using JavaScript for both frontend and backend simplifies development and reduces the learning curve.
- **Full-Stack JavaScript:** Developers can work on both the frontend and backend using the same language.

- **Scalability:** MongoDB and Node.js are highly scalable, making MERN suitable for applications with large user bases.
- **Large Community:** A large and active community provides ample resources, support, and libraries.
- **Rapid Prototyping:** The ease of use and the availability of numerous pre-built components allow for faster development and prototyping.

CSS (Cascading Style Sheets):

CSS is used for styling and presentation. It allows developers to control the layout, appearance, and design of HTML elements on a webpage.

Key Points:

Selectors: Define which HTML elements to style.

Properties: Determine the visual aspects like color, size, font, etc.

Layout: CSS3 introduced advanced layout features, such as Flexbox and Grid.

Backend Development:

1. Node.js:

- **The Foundation:** Node.js is the runtime environment. It's not a framework itself, but rather the engine that allows JavaScript to run outside a web browser (on the server). This is key because it allows you to use the same language (JavaScript) for both the frontend (React) and the backend.
- **Non-blocking, Event-Driven Architecture:** This is a crucial aspect of Node.js's performance. Instead of waiting for a task to complete before moving on to the next (like in traditional synchronous programming), Node.js handles multiple requests concurrently. When a task (like fetching data from a database) is initiated, Node.js continues processing other requests without waiting. Once the database operation is complete, Node.js receives a notification (an "event") and processes the result. This makes Node.js very efficient for handling many concurrent connections, ideal for real-time applications and applications with I/O-bound operations.

- **npm (Node Package Manager):** Node.js comes with npm, a powerful package manager. This allows you to easily install and manage external libraries and modules that extend Node.js's functionality. Many essential libraries for backend development (like Express.js itself) are available via npm.

2. Express.js:

- **The Web Framework:** Express.js is a minimalist and flexible Node.js web application framework. It's built *on top of* Node.js, providing a structured way to build web servers and APIs. Think of Node.js as the engine, and Express.js as the chassis and steering wheel of a car – it gives you the tools to control and structure how the engine operates within a web application context.

- **Routing:** Express.js provides a mechanism for defining routes – essentially, mapping URLs to specific functions or handlers. For example, you might define a route that handles requests to /users, fetching user data from the database.

- **Middleware:** Middleware functions are placed between the request and response cycles. They perform tasks like authentication (verifying if a user is logged in), authorization (checking if a user has permission to access a resource), logging requests, and parsing request bodies (e.g., converting JSON data from a POST request into a JavaScript object). This modular approach keeps your code organized and allows you to add functionality easily.

- **API Creation:** Express.js is commonly used to create RESTful APIs (Representational State Transfer APIs). These APIs follow a standard structure for handling requests (GET, POST, PUT, DELETE) and sending responses (usually in JSON format). The React frontend will communicate with the backend primarily through these APIs.

3. MongoDB (Database):

- **Data Persistence:** MongoDB is a NoSQL, document-oriented database. It stores data in flexible JSON-like documents, making it highly scalable and suitable for applications where the data structure might change over time.

- **Schema-less:** Unlike relational databases (like MySQL or PostgreSQL), MongoDB doesn't enforce a rigid schema. This flexibility is beneficial for rapid development and adapting to evolving data requirements. However, it requires careful data modeling to maintain data consistency and integrity.

- **Driver:** You'll need a MongoDB driver (a library that allows your Node.js application to interact with MongoDB). The official MongoDB Node.js driver is commonly used. This driver provides functions to connect to the database, insert, update, delete, and query data.

MERN Features

I. Development Experience & Efficiency:

- **JavaScript Full-Stack:** The most significant advantage is using JavaScript throughout the entire stack (frontend, backend, database). This eliminates the need to learn multiple languages and streamlines development. Developers can easily switch between frontend and backend tasks, improving productivity.

- **Rapid Prototyping:** The combination of relatively straightforward frameworks (Express.js, React) and a flexible database (MongoDB) allows for rapid prototyping and iteration. This speeds up the development process, especially in the early stages of a project.

- **Large and Active Community:** MERN enjoys a large and active community of developers. This means ample resources are available online (tutorials, documentation, libraries, support forums), making it easier to find solutions to problems and learn new techniques. The vast number of pre-built packages available via npm simplifies development.

- **Easy to Learn (Relatively):** Compared to some other full-stack technologies, MERN is considered relatively easier to learn, particularly for developers already familiar with JavaScript.

II. Scalability and Performance:

- **Scalability:** Both Node.js (with its non-blocking I/O model) and MongoDB (with its horizontal scaling capabilities) are designed for scalability. This means MERN applications can handle a large number of concurrent users and growing data volumes efficiently.

- **Non-blocking I/O:** Node.js's non-blocking, event-driven architecture allows it to handle multiple requests concurrently without waiting for each operation to complete. This enhances performance and responsiveness, especially important for real-time applications.
- **Flexible Data Modeling (MongoDB):** MongoDB's schema-less nature allows for flexible data modeling. This can be beneficial for projects with evolving data structures, making it easier to adapt to changing requirements.

III. Architecture and Design:

- **Modular Design (Express.js and React):** Both Express.js and React promote a modular design, making code more organized, maintainable, and reusable. This improves code quality and makes it easier for developers to collaborate.
- **RESTful APIs:** Express.js is well-suited for creating RESTful APIs, a standard and widely adopted way for different parts of an application (or different applications) to communicate.
- **Component-Based UI (React):** React's component-based architecture allows for the creation of reusable UI elements, improving consistency and reducing development time. The virtual DOM optimizes UI updates for enhanced performance.

IV. Deployment and Ecosystem:

- **Cloud-Friendly:** MERN applications are easily deployable to various cloud platforms (AWS, Google Cloud, Heroku, etc.).
- **Extensive Libraries and Tools:** The npm ecosystem provides access to a vast collection of libraries and tools that extend the capabilities of the stack.

HOW TO MERN STACK

1. Install Node.js and npm (or yarn):

- Download: Go to the official Node.js website (https://nodejs.org/). Download the installer for your operating system (Windows, macOS, or Linux).

- Install: Run the installer. This will typically also install npm (Node Package Manager), which is used to manage JavaScript packages. npm comes bundled with Node.js.

- Verify Installation: Open your terminal or command prompt and type node -v and npm -v (or yarn -v if you choose to use yarn). You should see the versions of Node.js and npm (or yarn) printed.

    - Yarn (Optional but Recommended): Yarn is an alternative package manager that often provides faster and more reliable installations. You can install it separately by following the instructions on the official Yarn website (https://yarnpkg.com/). If you use Yarn, replace npm with yarn in the subsequent commands.

2. Install MongoDB:

- Download: Visit the official MongoDB website (https://www.mongodb.com/). Download the appropriate installer for your operating system.

- Install: Follow the installer's instructions. You'll likely need to choose a directory for the MongoDB installation.

- Run MongoDB: After installation, you might need to start the MongoDB service manually. Consult the MongoDB documentation for instructions specific to your operating system. Typically, you'll have a mongod executable that you start from the command line. On some systems, there might be a service manager to start and stop MongoDB.

- Verify MongoDB: Once running, connect to the MongoDB shell (usually by typing mongo in your terminal). You should see a prompt indicating a successful connection. If you encounter connection errors, ensure the MongoDB service is started correctly and that your MongoDB configuration is set up correctly.

FLOW OF PROJECT

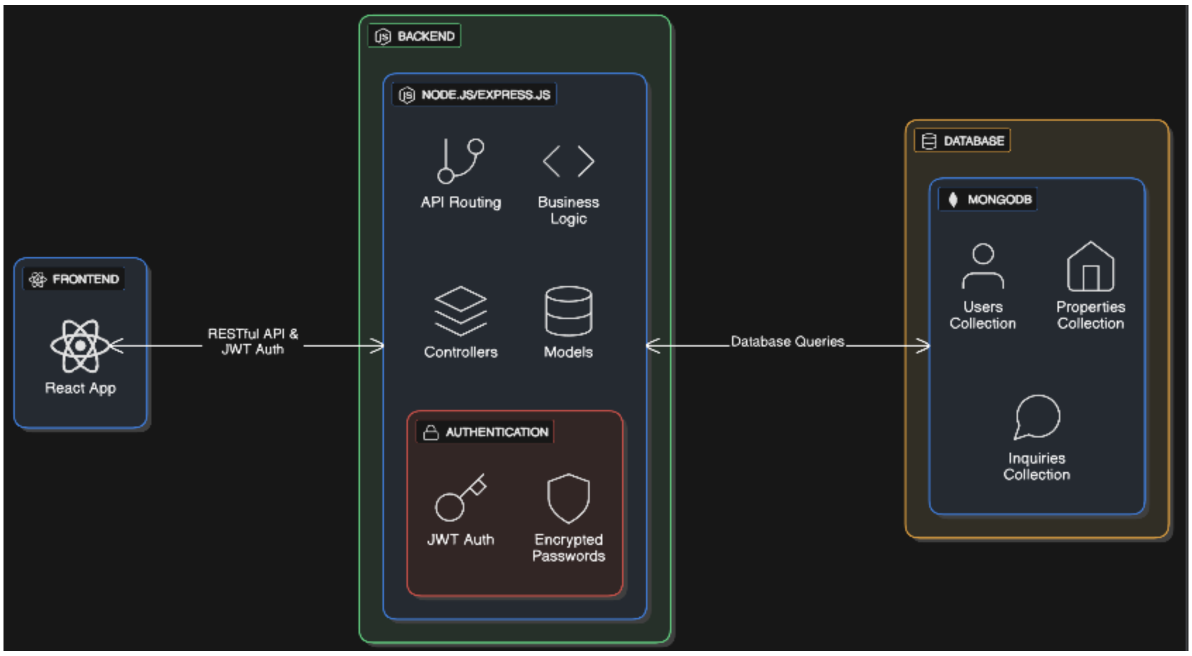
Systems are designed keeping in mind an issue that is to be solved. Every system is designed in its unique keeping in mind the requirement of the problem or the issue. Our system solves the problem of searching for the good that the customer's needs.

System design involves the design of overall architecture, based on which we design components, modules and interfaces. The beginning of any system architecture is by decomposing it into smaller fragments. Decomposition and binding of components makes the architecture easy to understand and makes it easier to understand.

Our system uses algorithm for collecting data which will collect the data of user and we have data analysing algorithm which will analyse and highlight the needs of user.

ARCHITECTURE DIAGRAM

An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.



SYSTEM IMPLEMENTATION

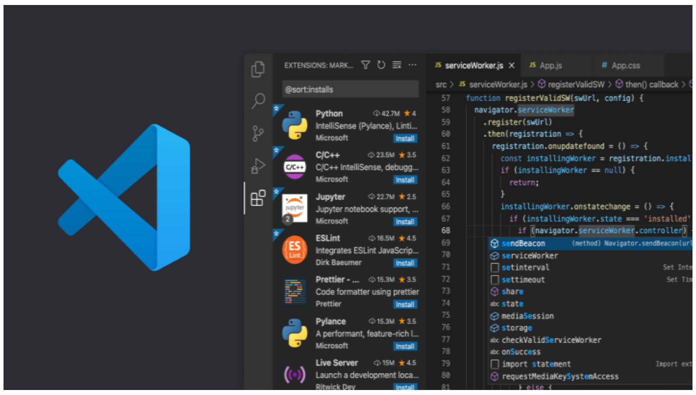
Hardware Used:

Laptop with RYZEN 4700U SERIES processor with 8GB ram

Software's used:

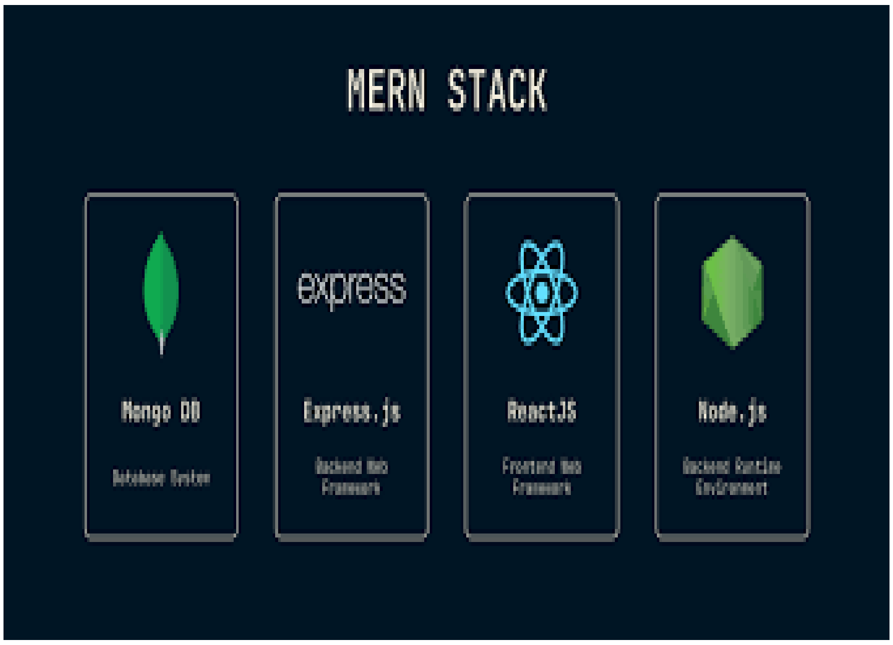
- Microsoft Windows11 Home as Operating System.

- Visual Studio Code (VS Code) as Integrated Development Environment


EXTENSIONS: MARK...
@sort:installs
Python 42.7M 4 IntelliSense (Pylance), Linti... Microsoft Install
C/C++ 23.5M 3.5 C/C++ IntelliSense, debugg... Microsoft Install
Jupyter 22.7M 2.5 Jupyter notebook support, ... Microsoft Install
ESLint 16.5M 4.5 Integrates ESLint JavaScrip... Dirk Baeumer Install
Prettier - ... 15.3M 3.5 Code formatter using prettier Prettier Install
Pylance 15.3M 3.5 A performant, feature-rich l... Microsoft Install
Live Server 15M 4.5 Launch a development loca... Ritwick Dey Install
serviceWorker.js
App.js
App.css
src > serviceWorker.js > registerValidSW > then() callback

```
function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
      registration.onupdatefound = () => {
        const installingWorker = registration.instal
        if (installingWorker == null) {
          return;
        }
        installingWorker.onstatechange = () => {
          if (installingWorker.state === 'installed'
            if (navigator.serviceWorker.controller)
              sendBeacon          (method) Navigator.sendBeacon(url
              serviceWorker
              setInterval                                  Set Inte
              setTimeout                                   Set Tim
              share
              state
              mediaSession
              storage
              checkValidServiceWorker
              onSuccess
              import statement                             Import ext
              requestMediaKeySystemAccess
            } else {
```

- MERN STACK



System Design & Development

Working with MERN STACK

Working with the MERN stack involves several key aspects: setting up your development environment, structuring your project, creating the backend (using Node.js and Express.js), building the frontend (using React), and connecting the two. Let's break down each step:

I. Setting Up Your Development Environment:

1. **Node.js and npm (or yarn):** Install the latest LTS (Long Term Support) version of Node.js from https://nodejs.org/. This automatically installs npm (Node Package Manager). Consider using Yarn (https://yarnpkg.com/) as an alternative package manager; it's often faster and more reliable.
2. **MongoDB:** Download and install MongoDB Community Edition from https://www.mongodb.com/. Make sure the MongoDB service is running. You might need to configure it depending on your operating system.
3. **Code Editor:** Choose a code editor or IDE (Integrated Development Environment). Popular choices include Visual Studio Code, Sublime Text, Atom, or WebStorm.
4. **Terminal/Command Prompt:** You'll be using the terminal or command prompt extensively to run commands, manage packages, and start your servers.

II. Project Structure:

A typical MERN project structure might look like this:

```
mern-app/
├── client/         // React frontend
│   ├── public/
│   ├── src/
│   │   ├── App.js
│   │   ├── components/
│   │   └── ...
│   ├── package.json
│   └── ...
└── server/         // Node.js/Express.js backend
    ├── routes/      // API routes
    ├── models/      // Database models (schemas)
    ├── controllers/ // API controllers (logic)
    ├── config/      // Database configuration
    ├── package.json
    └── server.js    // Main server file
```

III. Backend Development (Node.js and Express.js):

1. **Create server/ directory and package.json:** In your terminal, navigate to the server directory and run npm init -y to create a package.json file.

2. **Install Dependencies:** Install Express.js and the MongoDB driver:

```
npm install express mongodb
```

3. **Database Connection:** In config/db.js (or a similar file), establish a connection to your MongoDB database. This will require your MongoDB connection string.

4. **Models:** Define your data models (schemas) in models/. For example, a House model might define fields like address, price, description, images, etc.

5. **Controllers:** Create controllers in controllers/ to handle API requests. Each controller will have functions for creating, reading, updating, and deleting (CRUD) operations on specific data models.

6. **Routes:** Define API routes in routes/ using Express.js. These routes will map HTTP requests (GET, POST, PUT, DELETE) to specific controller functions.

7. **Server File (server.js):** This file starts the server, sets up middleware (like body-parsing), and connects the routes to the Express.js app.

IV. Frontend Development (React):

1. **Create client/ directory:** Create the client directory and navigate to it in your terminal.

2. **Create React App:** Use Create React App to scaffold your React project:

```
npx create-react-app .
```

3. **Install Dependencies:** Install necessary packages, such as axios for making HTTP requests to your backend API.

4. **Components:** Create React components to represent different parts of your UI (e.g., search forms, house listings, user profiles).

5. **API Calls:** Use axios (or fetch) to make API calls to your backend to fetch and manipulate data.

6. **State Management:** For larger applications, consider using a state management library like Redux or Zustand to manage application state effectively.

V. Connecting Frontend and Backend:

1. **API Endpoints:** Ensure that your frontend API calls match the routes you've defined in your Express.js backend.

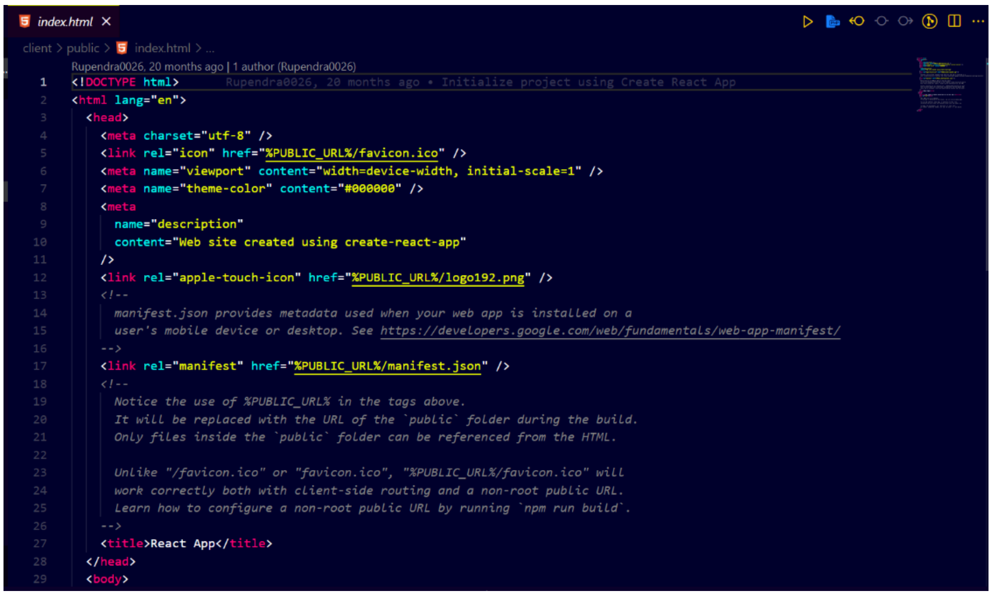
2. **Data Handling:** Handle responses from the backend appropriately in your React components.

3. **Error Handling:** Implement proper error handling on both the frontend and backend to manage unexpected situations gracefully.

RESULT AND DISCUSSION

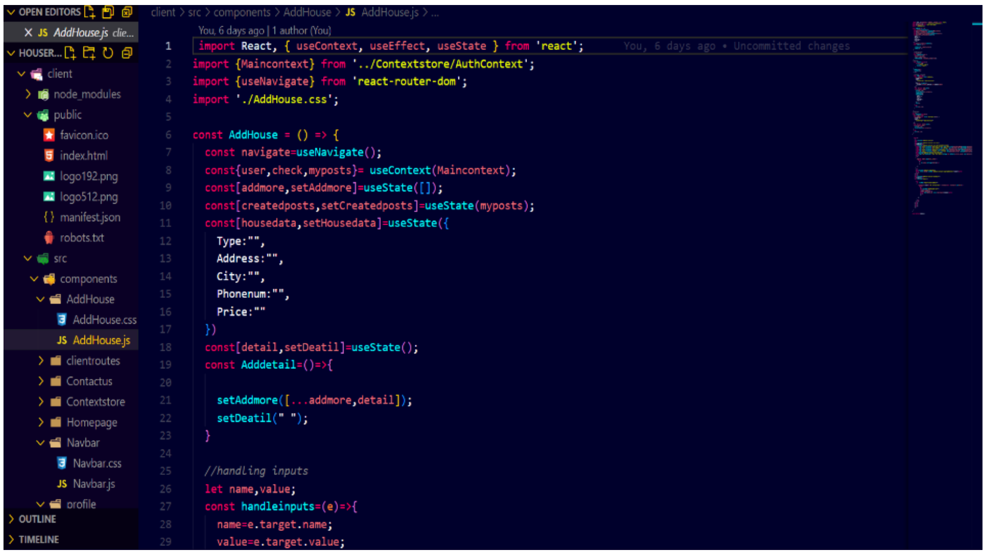
Code

Index page code



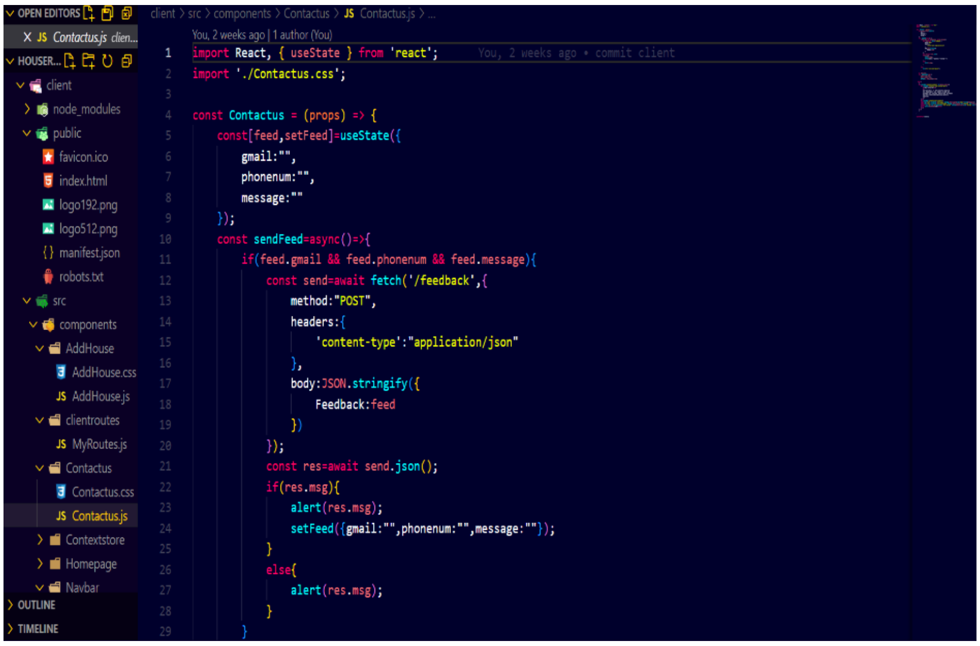
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
```

ADD HOUSE



```javascript
import React, { useContext, useEffect, useState } from 'react';
import {Maincontext} from '../Contextstore/AuthContext';
import {useNavigate} from 'react-router-dom';
import './AddHouse.css';

const AddHouse = () => {
  const navigate=useNavigate();
  const{user,check,myposts}= useContext(Maincontext);
  const[addmore,setAddmore]=useState([]);
  const[createdposts,setCreatedposts]=useState(myposts);
  const[housedata,setHousedata]=useState({
    Type:"",
    Address:"",
    City:"",
    Phonenum:"",
    Price:""
  })
  const[detail,setDeatil]=useState();
  const Adddetail=()=>{

    setAddmore([...addmore,detail]);
    setDeatil(" ");
  }

  //handling inputs
  let name,value;
  const handleinputs=(e)=>{
    name=e.target.name;
    value=e.target.value;
```

CONTACT US


client > src > components > Contactus > JS Contactus.js > ...
OPEN EDITORS
HOUSER...
OUTLINE
TIMELINE

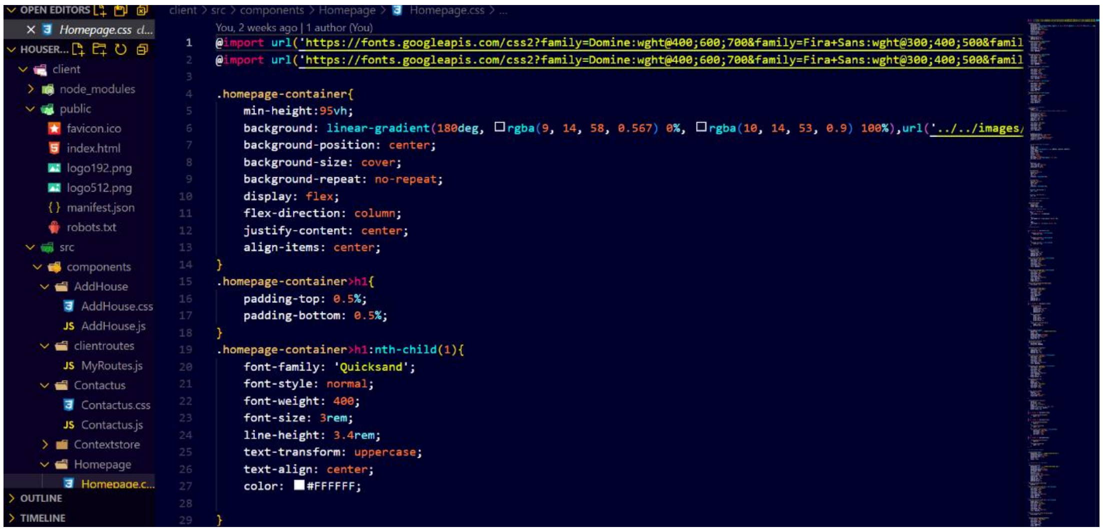
```
You, 2 weeks ago | 1 author (You)
import React, { useState } from 'react';      You, 2 weeks ago • commit client
import './Contactus.css';

const Contactus = (props) => {
    const[feed,setFeed]=useState({
        gmail:"",
        phonenum:"",
        message:""
    });
    const sendFeed=async()=>{
        if(feed.gmail && feed.phonenum && feed.message){
            const send=await fetch('/feedback',{
                method:"POST",
                headers:{
                    'content-type':"application/json"
                },
                body:JSON.stringify({
                    Feedback:feed
                })
            });
            const res=await send.json();
            if(res.msg){
                alert(res.msg);
                setFeed({gmail:"",phonenum:"",message:""});
            }
            else{
                alert(res.msg);
            }
        }
```

HOME PAGE

client > src > components > Contactus > JS Contactus.js > ...
OPEN EDITORS
HOUSER...
OUTLINE
TIMELINE

```
You, 2 weeks ago | 1 author (You)
import React, { useState } from 'react';      You, 2 weeks ago • commit client
import './Contactus.css';

const Contactus = (props) => {
    const[feed,setFeed]=useState({
        gmail:"",
        phonenum:"",
        message:""
    });
    const sendFeed=async()=>{
        if(feed.gmail && feed.phonenum && feed.message){
            const send=await fetch('/feedback',{
                method:"POST",
                headers:{
                    'content-type':"application/json"
                },
                body:JSON.stringify({
                    Feedback:feed
                })
            });
            const res=await send.json();
            if(res.msg){
                alert(res.msg);
                setFeed({gmail:"",phonenum:"",message:""});
            }
            else{
                alert(res.msg);
            }
        }
```

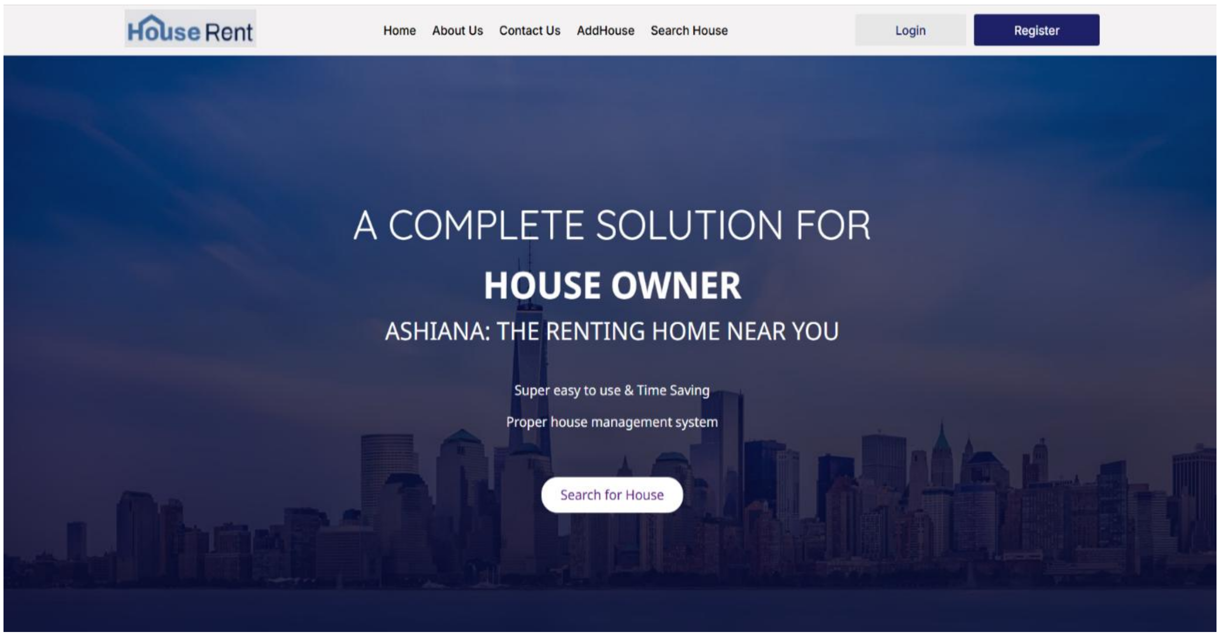
HOME CSS CODE



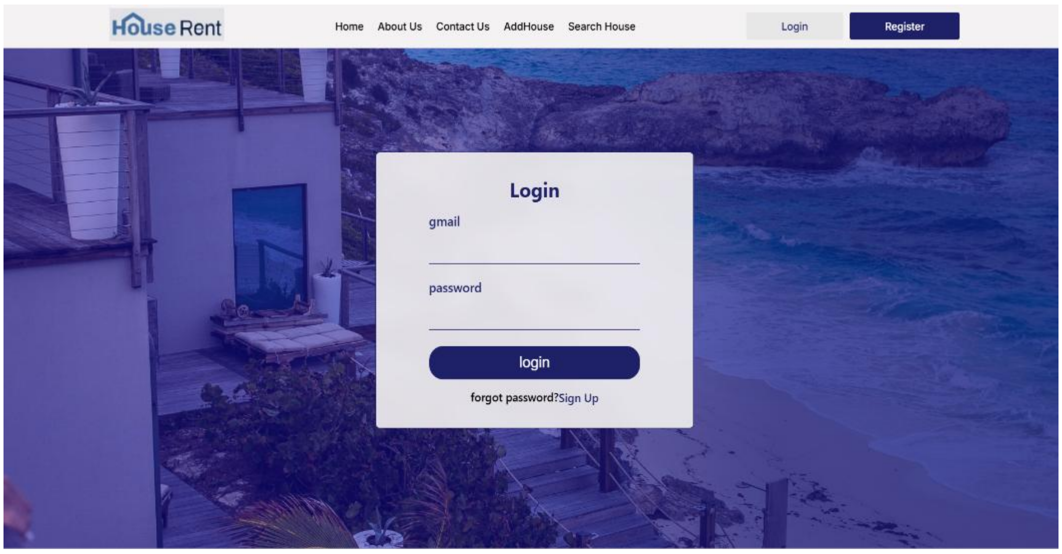
```css
@import url('https://fonts.googleapis.com/css2?family=Domine:wght@400;600;700&family=Fira+Sans:wght@300;400;500&famil
@import url('https://fonts.googleapis.com/css2?family=Domine:wght@400;600;700&family=Fira+Sans:wght@300;400;500&famil

.homepage-container{
    min-height:95vh;
    background: linear-gradient(180deg, rgba(9, 14, 58, 0.567) 0%, rgba(10, 14, 53, 0.9) 100%),url('../../images/
    background-position: center;
    background-size: cover;
    background-repeat: no-repeat;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
.homepage-container>h1{
    padding-top: 0.5%;
    padding-bottom: 0.5%;
}
.homepage-container>h1:nth-child(1){
    font-family: 'Quicksand';
    font-style: normal;
    font-weight: 400;
    font-size: 3rem;
    line-height: 3.4rem;
    text-transform: uppercase;
    text-align: center;
    color: #FFFFFF;

}
```

GUI

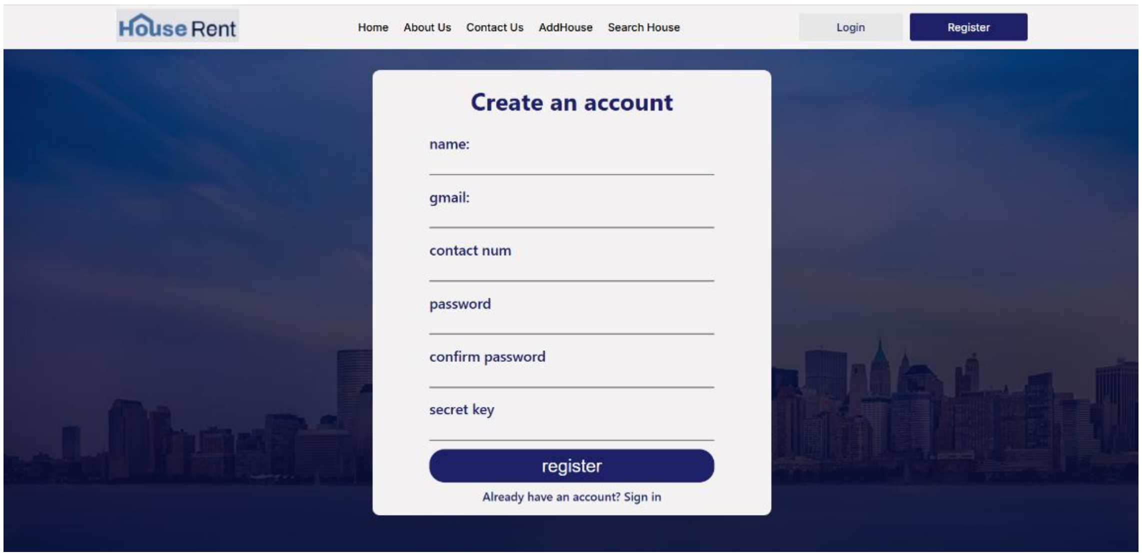
CONTACT US



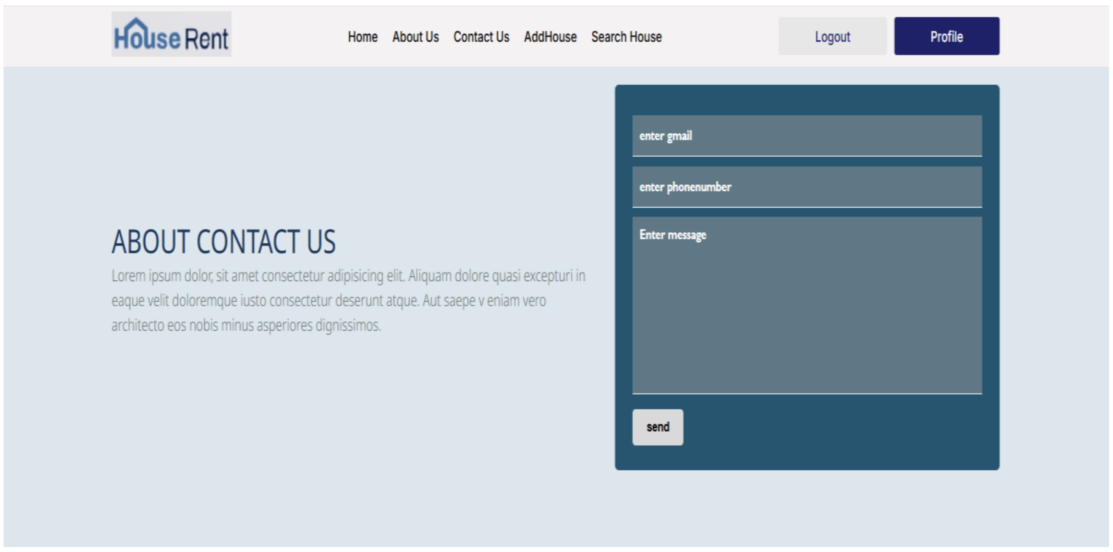
LOGIN PAGE



REGISTER PAGE



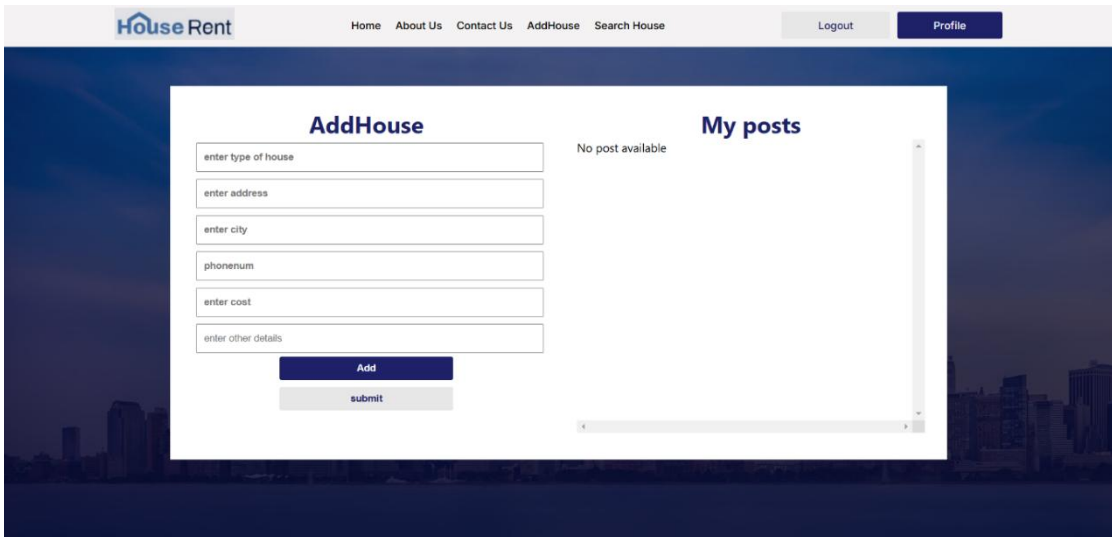
CREATE AN ACCOUNT



ABOUT CONTACT US

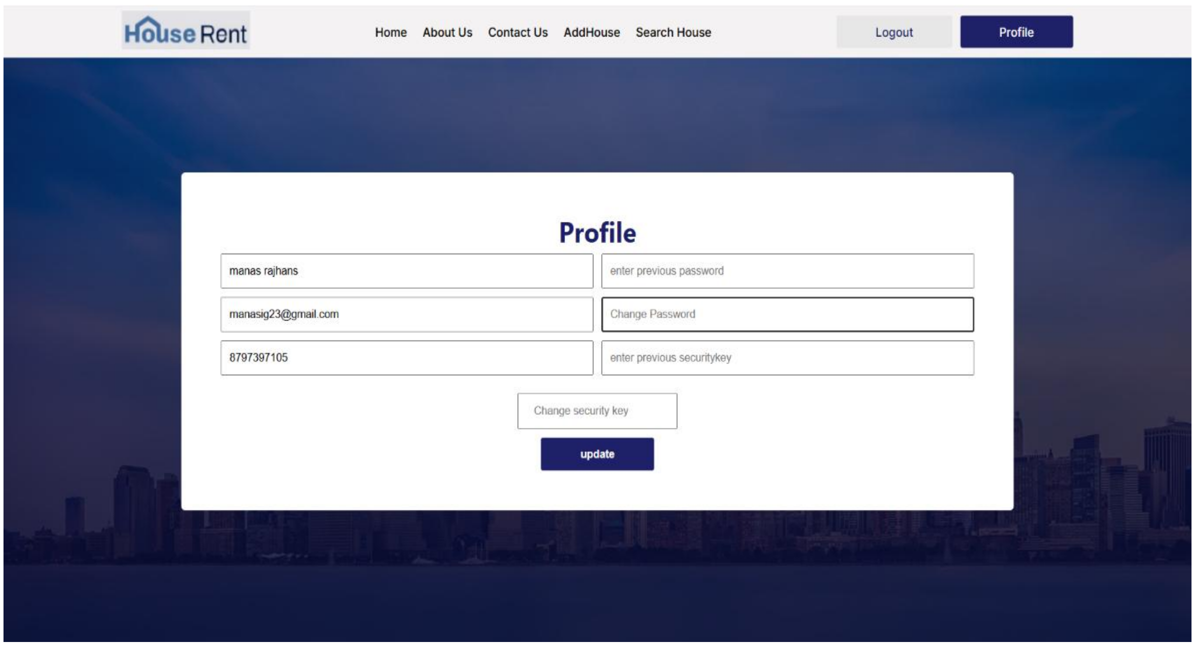
Lorem ipsum dolor, sit amet consectetur adipisicing elit. Aliquam dolore quasi excepturi in eaque velit doloremque iusto consectetur deserunt atque. Aut saepe v eniam vero architecto eos nobis minus asperiores dignissimos.

ADD HOUSE



No post available

PROFILE



Profile

manas rajhans

enter previous password

manasig23@gmail.com

Change Password

8797397105

enter previous securitykey

Change security key

update

Conclusion

The **House Renting System** project demonstrates how modern web technologies can be leveraged to simplify and enhance the rental process for both landlords and tenants. By combining the capabilities of the **MERN stack**—MongoDB, Express.js, React.js, and Node.js—the application provides a secure, scalable, and user-friendly solution to the challenges in traditional renting systems.

Key Achievements

1. **User-Friendly Platform:**
   The project successfully creates a seamless interface that makes it easy for users to navigate, search for properties, and communicate with property owners. Tenants can browse listings and inquire about properties, while landlords can efficiently manage their property details.
2. **Streamlined Processes:**
   The system replaces cumbersome manual processes with digital solutions, such as:
   - Simplifying property searches through filters and sorting options.
   - Automating the management of property details and tenant inquiries.
   - Providing a real-time communication channel.
3. **Security and Authentication:**
   Implementing secure login and registration for both landlords and tenants ensures that data privacy is maintained. The use of modern authentication techniques like **bcrypt** and **JSON Web Tokens (JWT)** enhances the reliability of the system.
4. **Scalability:**
   The system design ensures scalability, making it suitable for managing a growing number of users and properties as the application evolves. The use of MongoDB allows for efficient storage and retrieval of data, even with increasing user demands.

Challenges Addressed

1. **Transparency:**
   The system increases transparency in property transactions by providing detailed information about properties and landlords.
2. **Accessibility:**
   With a responsive design, the platform is accessible across various devices, ensuring users can interact with it conveniently from desktops, tablets, or smartphones.
3. **Efficient Data Management:**
   The integration of MongoDB enables structured storage of user profiles, property details, and communication logs, allowing for smooth data retrieval and updates.

Here is the hyperlink for the project Consisting of

- Project report
- Project code
- PPT of project

https://github.com/PriyasnhuSharma/priyanshu-12200974-B49A