Phase 1&2 Project Documents

Ideation Phase
Define the Problem Statements

Date	22 June 2024
Team ID	SWTID1720019632
Project Name	HOUSE RENT APP USING MERN
Maximum Marks	3 Marks
Team Members	● Madhavv Viswanath K ● Mohnishwar D ● Hiruthick SM ● Ganesh S

Customer Problem Statement:

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	Tenent/Renter	Rent a house posted on the site using mobile/laptop	It takes much time to load the available houses and Images	The site's server is slow and it takes time to fetch the image and load in the UI	Frustrated and May does not visit the site again
PS-2	Owner	To Post a House in the site for sale using phone/laptop by filling a Form and uploading Documents	Difficult to upload the necessary documents and doesn't support all image formats.	The website may only support specific file formats. If you are trying to upload a file format that	Confused and Looks for Alternative site if the issue persists

				is not supported, the upload will fail.	

Project Design Phase-I
Proposed Solution

Date	22 June 2024
Team ID	SWTID1720019632
Project Name	HOUSE RENT APP USING MERN
Maximum Marks	3 Marks

Proposed Solution:

Project team shall fill the following information in proposed solution template.

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	Develop a user-friendly house rental platform to streamline property searches, automate tenant verification, manage leases, ensure timely payments, and efficiently handle maintenance requests, enhancing the rental experience for all stakeholders.
2.	Idea / Solution description	Create an integrated rental platform featuring advanced search filters, automated tenant screening, digital lease management, secure online payment systems, and efficient maintenance request handling to simplify and enhance the rental process.
3.	Novelty / Uniqueness	Combines advanced search algorithms, automated tenant verification and seamless rental experience.

4.	Social Impact / Customer Satisfaction	1. Increasing Access to Housing: By providing a platform for landlords to list them properties, the app can increase the availability of rental housing options, especially in areas where housing is scarce. 2. Empowering Tenants: Tenants can use the app to find suitable housing that meets their needs and budget, giving them more control over their living arrangements.
5.	Business Model (Revenue Model)	● Listing Fees ● Subscription Plans ● Market Expansion
6.	Scalability of the Solution	The platform can easily scale to accommodate a growing number of users, properties, and locations, with robust cloud infrastructure ensuring reliable performance and efficient handling of increased demand and data.

Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	22 June 2024
Team ID	SWTID1720019632
Project Name	HOUSE RENT APP USING MERN
Maximum Marks	4 Marks

Product Backlog, Sprint Schedule, and Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Authentication	USN-1.1	As an Owner or Renter, I can register for the application by	3	Low	Madhavv, Hiruthick

			entering my email, password, and confirming my password. In the Registration page			
Sprint-1		USN-1.2	Implement separate registration API's for handling Owner and Renter Data	5	Medium	Monish, Ganesh
Sprint-1		USN-1.3	Integrate frontend with backend for registration	8	High	Monish, Ganesh
Sprint-1		USN-1.4	As an Owner or Renter, I can login from any device through Login page by entering my registered email, password	2	Low	Madhavv, Hiruthick
Sprint-2	Property Listing	USN-2.1	Renter Should be able to See the List of properties in the property Listing page	3	Low	Madhavv, Hiruthick
		USN-2.2	Implement property listing API	5	Medium	Madhavv, Hiruthick
		USN-2.3	Integrate frontend with backend for listing the properties posted by	8	High	Monish, Ganesh

			owners			
Sprint-3	Property Submission	USN-3.1	Owner Should be able to submit the Properties details in a form in the property Submission page	3	Low	Monish, Ganesh
		USN-3.2	Implement property submission API	5	Medium	Monish, Ganesh
		USN-3.3	Integrate frontend with backend for submission of property by Owner which will be posted in property listing page of Renters.	8	High	Madhavv, Hiruthick
Sprint-4	Property Details	USN-4.1	Renters Should be able to see the details of each listed property in The Details Page	3	Low	Madhavv, Hiruthick
		USN-4.2	Implement property details API by fetching the data posted by the Owners in the Property Submission Forms	5	Medium	Madhavv, Hiruthick
		USN-4.3	Integrate	8	High	Monish,

			frontend with backend for details			Ganesh
Sprint-5	Booking Property	USN-5.1	Renter should be able to request Booking by filling form details, in the Property Details page and, the form is sent to the property Owner	5	Medium	Monish, Ganesh
		USN-5.2	Implement booking API	5	Medium	Ganesh, Hiruthick
		USN-5.3	Integrate frontend with backend for booking status update	8	High	Monish, Madhavv
Sprint-6	Admin Dashboard	USN-6.1	Admin should be able to Handle Users, Owners and the Booking Status of Property	5	Medium	Monish, Ganesh
		USN-6.2	Implement admin monitoring API	8	High	Madhavv, Hiruthick
		USN-6.3	Integrate frontend with backend for monitoring tools	8	High	Madhavv, Hiruthick
Sprint-7	Property Management	USN-7.1	Owner should be able to Manage	5	Medium	Ganesh, Hiruthick

			Property by approving or rejecting the Renter and updating the Booking Status.			
		USN-7.2	Implement property CRUD API	8	High	Ganesh, Hiruthick
		USN-7.3	Integrate frontend with backend for property CRUD	8	High	Monish, Madhavv

Project Tracker, Velocity & Burndown Chart: (2 Marks)

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	18	3 Days	5 Jul 2024	7 Jul 2024	10	8 Jul 2023
Sprint-2	16	3 Days	7 Jul 2024	10 Jul 2024		
Sprint-3	16	5 Days	5 Jul 2024	10 Jul 2024		
Sprint-4	16	6 Days	5 Jul 2024	11 Jul 2024		
Sprint-5	18	7Days	5 Jul 2024	12 Jul 2024		
Sprint-6	19	7 Days	5 Jul 2024	12Jul 2024		
Sprint-7	21	7 Days	5 Jul 2024	12 Jul 2024		

Velocity:

Sprint	**Average Velocity (story points/Duration)**
Sprint-1	**18/3=6**
Sprint-2	**16/3=5.33**
Sprint-3	**16/5=3.2**
Sprint-4	**16/6=2.66**
Sprint-5	**18/7=2.57**
Sprint-6	**19/7=2.71**
Sprint-7	**19/7=2.71**

Requirement Gathering and Analysis Phase
Solution Requirements (Functional & Non-functional)

Date	06 July 2024
Team ID	SWTID1720019632
Project Name	HOUSE RENT APP USING MERN
Maximum Marks	

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	**Functional Requirement**	**Sub Requirement (Story / Sub-Task)**
FR-1	User Registration	● Registration through Form

FR-2	Property Listing	● Type of Property ● Price and Availability ● Location of Property
FR-3	Property Submission	● Registration of Owner Details through Form ● Property Details and Proof Document ● Uploading Property Images and Features
FR-4	Property Details	● Detailed Description of Property ● Image Carousel ● Status of Property (Booked or Requested)
FR-5	Booking Property	● Requesting Property through Form ● Filling The User Details to Request to Owner ● Cancel Booking
FR-6	Admin Dashboard	● Handle The User and Owner List ● Approve The Owner to post Property in the site ● Monitor the Property Booking Status
FR-7	Property Management	● Reject Rental Request by User ● Approve Rental and Change Status by Owner ● Add Property to the site by Owner ● Delete the posted Property ● Update the Property Details

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

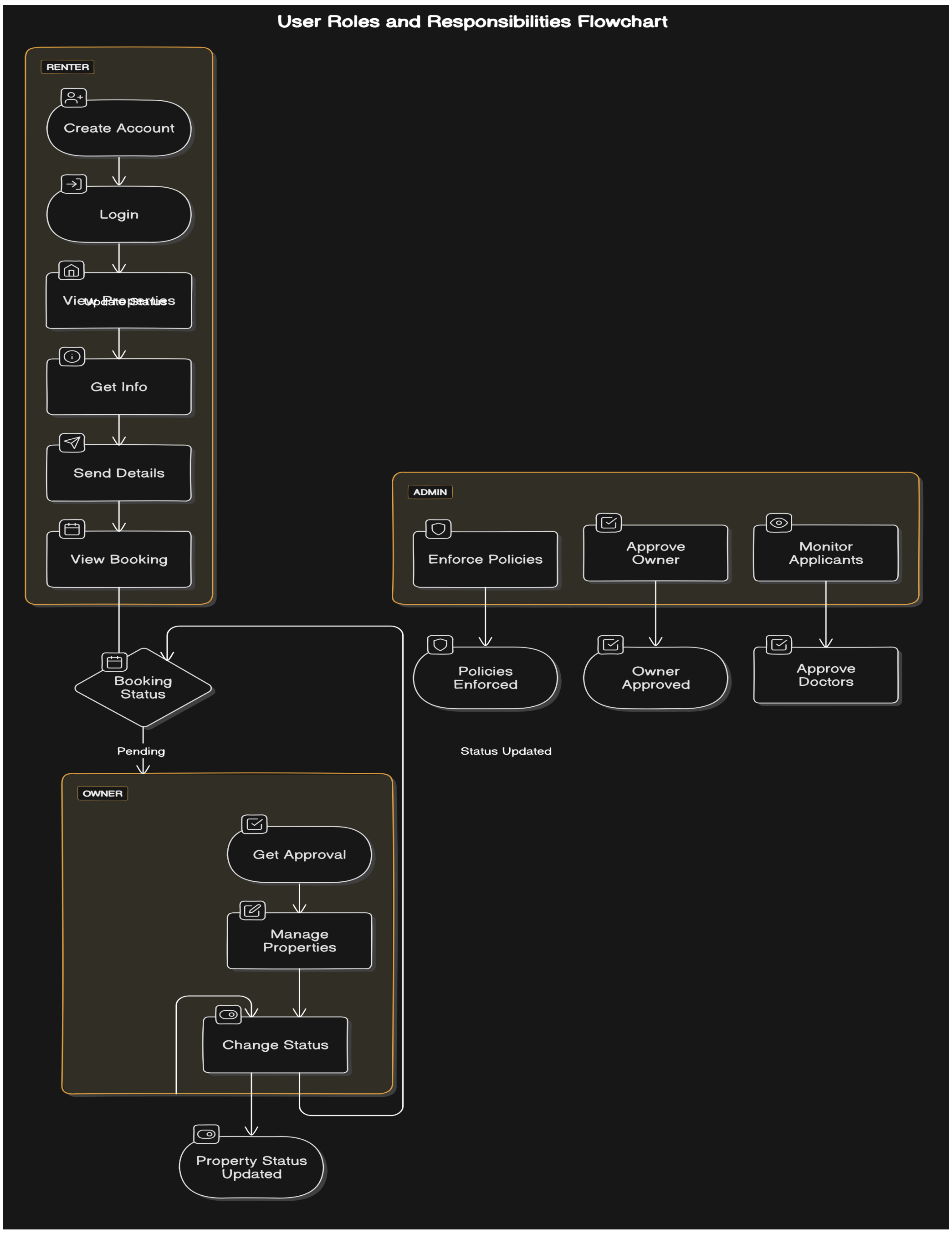
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	● The user interface should be intuitive and easy to navigate for users of all technical backgrounds. ● The application should provide a clear and responsive List of Property Cards for users of all kinds of device. ● The property viewing and renting experience should be easy and user-

		friendly
NFR-2	**Security**	● The application must implement secure user authentication and authorization mechanisms. ● All the Booking and User Details filling must be secure, and the API request and response data must be encrypted. ● The app must run in https protocol and be hosted in a genuine hosting site.
NFR-3	**Reliability**	● The application should be highly reliable with minimal downtime or disruptions.
NFR-4	**Performance**	● The application should provide high quality images of the property and the booking status must reflect changes without any delay ● It must be responsive in all devices with all sizes.
NFR-5	**Availability**	● The application should be available to users 24/7 with minimal downtime for maintenance. ● The application should be able to handle high volumes of concurrent users without performance degradation.
NFR-6	**Scalability**	● The platform must easily scale to accommodate a growing number of users, properties, and locations. ● With robust cloud infrastructure it must ensure reliable performance and efficient handling of increased demand and data

Requirement Gathering and Analysis Phase
Solution Architecture

Date	06 July 2024
Team ID	SWTID1720019632
Project Name	HOUSE RENT APP USING MERN
Maximum Marks	

Solution Architecture:



EXPLANATION:

1. Renter/Tenent:
   - Create an account and log in to the system using their email and password.
   - They will be shown automatically all the properties in their dashboard.
   - After clicking on the Get Info, all the information of the property and owner will come, and small form will generate in which the renter needs to send his\her details.
   - After that they can see their booking in booking section where the status of booking will be showing “pending”. It will be change by owner of the property.

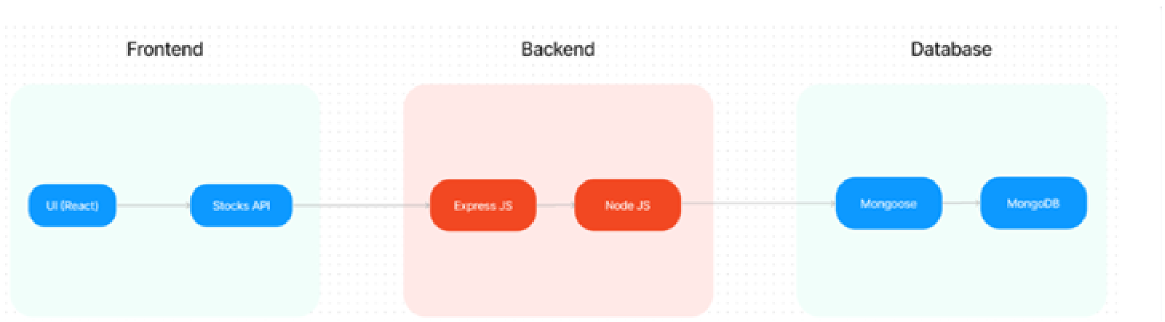
2. **Admin:**
   - He/she can approve the user as “owner” for the legit user to add properties in his app
   - He monitors the applicant of all doctors and approve them and then doctors are registered in the app.
   - Implement and enforce platform policies, terms of service, and privacy regulations.

3. **Owner:**
   - Gets the approval from the admin for his Owner account.
   - After approval, he/she can do all CRUD operation of the property in his/her account
   - He/she can change the status and availability of the property.

Requirement Gathering and Analysis Phase
Technology Stack (Architecture & Stack)

Date	06 July 2024
Team ID	SWTID1720019632
Project Name	HOUSE RENT APP USING MERN
Maximum Marks	

Technical Architecture



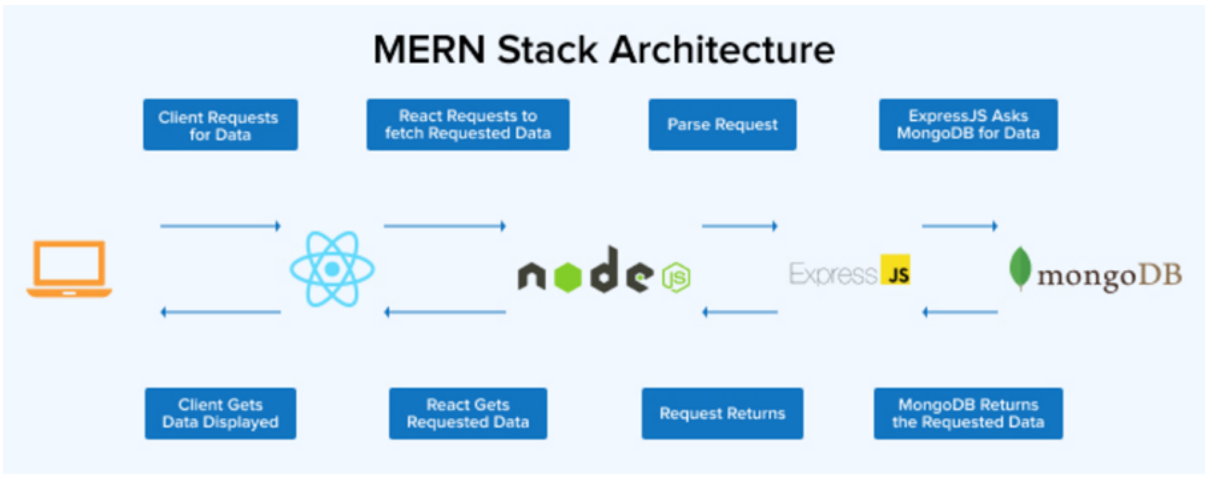

MERN Stack Architecture
Client Requests for Data
React Requests to fetch Requested Data
Parse Request
ExpressJS Asks MongoDB for Data
node JS
Express JS
mongoDB
Client Gets Data Displayed
React Gets Requested Data
Request Returns
MongoDB Returns the Requested Data

Table-1: Components & Technologies:

S.No	Component	Description	Technology
1	User Interface	How users interact with the application (Web UI, Mobile)	HTML, CSS, JavaScript, React.js
2	Application Logic-1	Logic for user authentication and authorization	Node.js, Express.js

3	Application Logic-2	Logic for property management (CRUD operations)	Node.js, Express.js
4	Application Logic-3	Logic for booking management (request and status update)	Node.js, Express.js
5	Database	Data storage and configurations	MongoDB, Mongoose
6	Cloud Database	Database service on cloud	MongoDB Atlas
7	File Storage	File storage requirements (property images, documents)	Local Filesystem
8	Infrastructure (Server/Cloud)	Application deployment on local system/cloud	Local

Table-2: Application Characteristics:

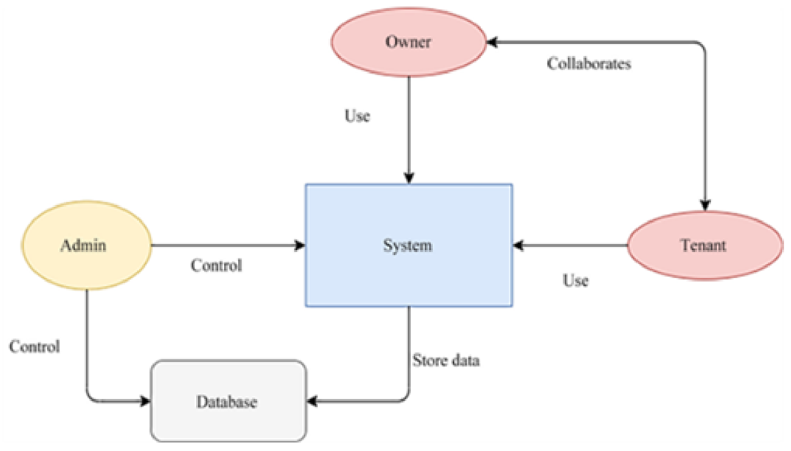
S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	List the open-source frameworks used	React.js, Node.js, Express.js, Mongoose
2	Security Implementations	List all the security/access controls implemented, use of firewalls, etc.	HTTPS, BSON
3	Scalable Architecture	Justify the scalability of architecture (3-tier,Client-Server)	MERN

Requirement Gathering and Analysis Phase
Data Flow Diagram & User Stories

Date	06 July 2024
Team ID	SWTID1720019632
Project Name	HOUSE RENT APP USING MERN
Maximum Marks	

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Renter, Owner	User Authentication	USN-1.1	As an Owner or Renter, I can register for the	Form fields: email, password, confirm password, role	Low	Sprint-1

			application by entering my email, password, and confirming my password. In the Registration page	selection (renter/owner). Successful registration should navigate to dashboard.		
Renter, Owner		USN-1.2	Implement separate registration API's for handling Owner and Renter Data	API should create user in the database, handle errors, and return success/failure response	Medium	Sprint-1
Renter, Owner		USN-1.3	Integrate frontend with backend for registration	Successful integration should allow new users to register and log in immediately.	High	Sprint-1
Renter, Owner		USN-1.4	As an Owner or Renter, I can login from any device through Login page by entering my registered email, password	API should authenticate user, handle errors, and return user-specific token	Low	Sprint-1

Renter	Property Listing	USN-2.1	Renter Should be able to See the List of properties in the property Listing page	Page should display list of available properties with summary information	Low	Sprint-2
Renter		USN-2.2	Implement property listing API	API should retrieve list of properties from the database, handle errors, and return properties	Medium	Sprint-2
Renter		USN-2.3	Integrate frontend with backend for listing the properties posted by owners	Should display properties on the listing page posted by owners	High	Sprint-2
Owner	Property Submission	USN-3.1	Owner Should be able to submit the Properties details in a form in the property Submission page	Form should contain property name, description, address, price, availability. Successful submission should add property to the database	Low	Sprint-3
Owner		USN-3.2	Implement property submission	API should add new property to the	Medium	Sprint-3

			API	database, handle errors, and return success/failure response.		
Owner		USN-3.3	Integrate frontend with backend for submission of property by Owner which will be posted in property listing page of Renters.	On Successful integration should allow owners to submit new properties.	High	Sprint-3
Renter	Property Details	USN-4.1	Renters Should be able to see the details of each listed property in The Details Page	Page should display detailed information about a selected property, including owner contact form.	Low	Sprint-4
Renter		USN-4.2	Implement property details API by fetching the data posted by the Owners in the Property Submission Forms	API should retrieve detailed information of a specific property from the database	Medium	Sprint-4

Renter		USN-4.3	Integrate frontend with backend for details	Successful integration should display detailed property information on the details	High	Sprint-4
Renter	Booking Property	USN-5.1	Renter should be able to request Booking by filling form details, in the Property Details page and, the form is sent to the property Owner	Forms with renter details, booking dates. Successful submission should create a booking request with status "pending".	Medium	Sprint-5
Renter		USN-5.2	Implement booking API	API should create a new booking in the database, handle errors, and return success/failure response.	Medium	Sprint-5
Renter		USN-5.3	Integrate frontend with backend for booking status update	Successful integration should allow renters to submit booking	High	Sprint-5

Admin	Admin Dashboard	USN-6.1	Admin should be able to Handle Users, Owners and the Booking Status of Property	Dashboard should display pending user approval requests with approve/reject buttons.	Medium	Sprint-6
Admin		USN-6.2	Implement admin monitoring API	API should update user approval status in the database	High	Sprint-6
Admin		USN-6.3	Integrate frontend with backend for monitoring tools	successful integration should allow admin to approve/reject user requests.	High	Sprint-6
Owner	Property Management	USN-7.1	Owner should be able to Manage Property by approving or rejecting the Renter and updating the Booking Status.	Owners should be able to create, read, update, and delete properties.	Medium	Sprint-7
Owner		USN-7.2	Implement property CRUD API	API should handle property CRUD operations.	High	Sprint-7

Owner		USN-7.3	Integrate frontend with backend for property CRUD	Successful integration should allow owners to manage their properties.	High	Sprint-7

Phase 3 to 5 Project Documents

Full Stack Development with MERN Frontend Development Report

Date	15th July 2024
Team ID	SWTID1720019632
Project Name	HOUSE RENT APP USING MERN
Maximum Marks	

Project Title: HOUSE RENT APP USING MERN

Date: [20/07/2024]

Prepared by: [Madhavv Viswanath. K]

Objective

The objective of this report is to document the frontend development progress and key aspects of the user interface implementation for the HOUSE RENT APP USING MERN project.

Technologies Used

- **Frontend Framework:** React.js
- **State Management:** Context API
- **UI Framework/Libraries:** Tailwind CSS
- **API Libraries:** Axios

Project Structure

- **The assets** Folder contains all the necessary images for the application.
- **Modules** contains all the necessary pages with appropriate folders
- **Vite.config.js** this makes the react app faster to load than native webpack bundler.
- **node modules** contain all the required dependencies for the frontend.

Key Components

1. **App.js**
   - Responsible for routing and main application layout.
2. **/Components**
   - Contains reusable UI components used across the application.
   - **Header Component**
   - **Footer Component**

3. **/Pages**
   - Includes different pages for Web App.
     - **Common Folder**
       - **Contains About Us, Login, Register, Home pages**

```
function AboutUs() {

  return (

    <>



      <div class="flex flex-col min-h-screen">

```

```
o               <main class="flex-1 p-8">
o                 <div class="max-w-4xl mx-auto space-y-8">
o                   <div>
o                     <h1 class="text-4xl font-bold">About Us</h1>
o                     <p class="text-lg text-muted-foreground mt-4">
o                       Rental App is a leading platform that connects renters with the perfect
properties. Our mission is to
o                       simplify the rental process and provide a seamless experience for both
landlords and tenants.
o                     </p>
o                   </div>
o                   <div>
o                     <h2 class="text-2xl font-bold">Our Story</h2>
o                     <p class="text-lg text-muted-foreground mt-4">
o                       Rental App was founded in 2024 by a team of passionate Web Dev enthusiasts
who saw the need for a more
o                       efficient and user-friendly rental platform. We started with a simple goal: to
make the rental process
o                       easier and more accessible for everyone.
o                     </p>
o                     <p class="text-lg text-muted-foreground mt-4">
o                       Over the years, we've grown to become one of the most trusted rental
platforms in the industry. Our team
o                       of experts works tirelessly to curate a diverse selection of properties and
provide exceptional customer
o                       service to our users.
```

```
                </p>
              </div>
              <div>
                <h2 class="text-2xl font-bold">Our Values</h2>
                <ul class="text-lg text-muted-foreground mt-4 space-y-2">
                  <li>
                    <span class="font-bold">Transparency:</span> We believe in being upfront and honest with our users,
                    providing clear and accurate information about our properties and services.
                  </li>
                  <li>
                    <span class="font-bold">Accessibility:</span> Our platform is designed to be user-friendly and
                    accessible to everyone, regardless of their technical expertise or background.
                  </li>
                  <li>
                    <span class="font-bold">Innovation:</span> We are constantly exploring new ways to improve the rental
                    experience, leveraging the latest technologies and industry insights to stay ahead of the curve.
                  </li>
                  <li>
                    <span class="font-bold">Community:</span> We are committed to building a strong and supportive community
                    of renters and landlords, fostering connections and collaboration.
```

```
o                         </li>
o                     </ul>
o                 </div>
o                 <div>
o                     <h2 class="text-2xl font-bold">Our Team</h2>
o                     <div class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-8 mt-4">
o                         <div class="bg-muted rounded-lg p-4 shadow-lg">
o                             <img src={""} alt="John Doe" class="rounded-full w-24 h-24 mx-auto" />
o                             <h3 class="text-xl font-bold mt-4">John Doe</h3>
o                             <p class="text-muted-foreground">Co-Founder and CEO</p>
o                         </div>
o                         <div class="bg-muted rounded-lg p-4 shadow-lg">
o                             <img src={"" } alt="Jane Smith" class="rounded-full w-24 h-24 mx-auto"
    />
o                             <h3 class="text-xl font-bold mt-4">Jane Smith</h3>
o                             <p class="text-muted-foreground">Co-Founder and CTO</p>
o                         </div>
o                         <div class="bg-muted rounded-lg p-4 shadow-lg">
o                             <img src={ ""} alt="Michael Johnson" class="rounded-full w-24 h-24 mx-
    auto" />
o                             <h3 class="text-xl font-bold mt-4">Michael Johnson</h3>
o                             <p class="text-muted-foreground">Head of Operations</p>
o                         </div>
o                     </div>
```

```
                        </div>
                    </div>
                </main>

            </div>

        </>
    )
}

function Home() {
  const navigate=useNavigate()
  return (
    <>
      <div class="flex min-h-[100dvh] flex-col">
        <main class="flex-1">
          <section class="bg-primary text-primary-foreground py-12 md:py-20 lg:py-28">
            <div class="container mx-auto px-4 md:px-6 lg:px-8">
              <div class="grid items-center gap-8 md:grid-cols-2">
                <div>
                  <h1 class="mb-4 text-4xl font-bold md:text-5xl lg:text-6xl">Find Your Perfect Rental</h1>
                  <p class="mb-8 text-lg md:text-xl">Our rental app makes it easy to discover and book the ideal property for your needs.</p>
                  <button onClick={()=>{navigate('/About')}}  class="bg-black text-white border border-gray-300 inline-flex h-10 items-center justify-center whitespace-nowrap rounded-md px-5 text-sm font-medium transition-transform  focus:outline-none focus:ring-2 focus:ring-white focus:ring-offset-2 hover:bg-blue-600 hover:text-white hover:scale-105 hover:shadow-lg disabled:pointer-events-none disabled:opacity-50">About Us</button>

                </div>

                <div>

                  <img src='assets/Profile.jpg' alt="Hero Image" class="rounded-xl" />

                </div>

              </div>

            </div>

          </section>



        </main>



      </div>



    </>

  )

}

export default Home

function Login() {

    const [email, setEmail] = useState();

    const [password, setpassword] = useState();

    const [user_type, setUser_type]=useState();
const navigate=useNavigate();

const handleSubmit = (e) => {


    e.preventDefault();

    axios.post('http://localhost:3001/login', {email, password,user_type})

        .then(result => {

            console.log(result)

            if(result.data==="Success")

            {

                if(user_type==="Renter")

                {

                    navigate('/details')



                }

                else

                {

                    navigate('/owner')

                }

            }

            else

            {

                if(result.data==="The password or Usertype is incorrect")

                {
                        alert(result.data);

                    }

                    if(result.data==="Not Registered")

                    {

                        alert(result.data);

                    }

                }

            })



            .catch(err => console.log(err))



    }



    return (

        <>

            <div class="flex min-h-[100vh] flex-col">



                <main class="flex-1">

                    <section class="w-full py-12 ">

                        <div class="container px-6 md:px-24">

                            <div class="grid gap-6 lg:grid-cols-[1fr_500px] lg:gap-12 xl:grid-cols-
[1fr_550px]">

                                <div class="flex flex-col justify-center space-y-3">

                                    <div class="space-y-2">
```

```
          <h1 class="text-3xl font-bold tracking-tighter sm:text-5xl xl:text-6xl/none">Login to Your Account</h1>

          <p class="text-muted-foreground max-w-[600px] md:text-xl">Welcome</p>

        </div>

        <form onSubmit={handleSubmit}>

          <div class="bg-card text-card-foreground w-full max-w-md rounded-lg border shadow-sm" data-v0-t="card">

            <div class="flex flex-col space-y-1 p-6">

              <h3 class="whitespace-nowrap text-2xl font-semibold tracking-tight">Login</h3>



            </div>

            <div class="relative h-10 w-72 min-w-[200px] ml-6">

              <select name='user-type' id='user-type' onChange={(e) => setUser_type(e.target.value)}

                class="peer h-full w-full rounded-[7px] border border-blue-gray-200 border-t-transparent bg-transparent px-3 py-2.5 font-sans text-sm font-normal text-blue-gray-700 outline outline-0 transition-all placeholder-shown:border placeholder-shown:border-blue-gray-200 placeholder-shown:border-t-blue-gray-200 empty:!bg-gray-900 focus:border-2 focus:border-gray-900 focus:border-t-transparent focus:outline-0 disabled:border-0 disabled:bg-blue-gray-50">

                <option value="Owner">Owner</option>

                <option value="Renter">Renter</option>



              </select>

              <label

                class="before:content[' '] after:content[' '] pointer-events-none absolute left-0 -top-1.5 flex h-full w-full select-none text-[11px] font-normal leading-tight text-
```

```
blue-gray-400 transition-all before:pointer-events-none before:mt-[6.5px] before:mr-1 before:box-border before:block before:h-1.5 before:w-2.5 before:rounded-tl-md before:border-t before:border-l before:border-blue-gray-200 before:transition-all after:pointer-events-none after:mt-[6.5px] after:ml-1 after:box-border after:block after:h-1.5 after:w-2.5 after:flex-grow after:rounded-tr-md after:border-t after:border-r after:border-blue-gray-200 after:transition-all peer-placeholder-shown:text-sm peer-placeholder-shown:leading-[3.75] peer-placeholder-shown:text-blue-gray-500 peer-placeholder-shown:before:border-transparent peer-placeholder-shown:after:border-transparent peer-focus:text-[11px] peer-focus:leading-tight peer-focus:text-gray-900 peer-focus:before:border-t-2 peer-focus:before:border-l-2 peer-focus:before:border-gray-900 peer-focus:after:border-t-2 peer-focus:after:border-r-2 peer-focus:after:border-gray-900 peer-disabled:text-transparent peer-disabled:before:border-transparent peer-disabled:after:border-transparent peer-disabled:peer-placeholder-shown:text-blue-gray-500">

                  Select a User Type

                </label>

              </div>

              <div class="grid gap-4 p-6">

                <div class="grid gap-2">

                  <label for="email" class="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70" > Email </label>

                  <input  name="email" id="email" placeholder="john@example.com" type="email" onChange={(e) => setEmail(e.target.value)} class="border-input bg-background ring-offset-background placeholder:text-muted-foreground focus-visible:ring-ring flex h-10 w-full rounded-md border px-3 py-2 text-sm file:border-0 file:bg-transparent file:text-sm file:font-medium focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50"  required />

                </div>

                <div class="grid gap-2">

                  <label for="password" class="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70" > Password </label>

                  <input name="password"  id="password" type="password" onChange={(e) => setpassword(e.target.value)} class="border-input bg-background ring-offset-background placeholder:text-muted-foreground focus-visible:ring-ring flex h-10 w-full rounded-md border px-3 py-2 text-sm file:border-0 file:bg-transparent file:text-sm file:font-medium focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50"  required />
```

```
                    </div>
                </div>
                <div class="flex items-center p-6">
                    <button class="align-middle select-none font-sans font-bold text-center uppercase transition-all disabled:opacity-50 disabled:shadow-none disabled:pointer-events-none text-xs py-3 px-6 rounded-lg bg-gray-900 text-white shadow-md shadow-gray-900/10 hover:shadow-lg hover:shadow-gray-900/20 focus:opacity-[0.85] focus:shadow-none active:opacity-[0.85] active:shadow-none block w-full"
                        type="submit">Sign In</button>
                </div>
                <div class="flex p-6 justify-items-center" >
                    <p class="pr-4">Not Registered?
                        <span class=" font-medium text-blue-600 dark:text-blue-500 hover:underline"> <NavLink to={'/register'} >Register</NavLink></span>
                    </p>
                </div>
            </div>
        </form>
    </div>
    <div class="flex flex-col justify-center space-y-4">
        <img src="https://images.unsplash.com/photo-1600585154340-be6161a56a0c?q=80&w=2670&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D" class="mx-auto aspect-video overflow-hidden rounded-xl object-cover object-center sm:w-full lg:order-last " />
    </div>

</div>
```

```
                        </div>
                    </section>
                </main>
            </div>

        </>
    )
}

export default Login
function Register() {
    const [name, setName] = useState();
    const [phone, setPhone] = useState();
    const [email, setEmail] = useState();
    const [password, setpassword] = useState();
    const [user_type, setUser_type]=useState();
    const navigate = useNavigate();
    const handleSubmit = (e) => {

        e.preventDefault();
        axios.post('http://localhost:3001/register', { name,user_type,phone, email, password })
            .then(result => {console.log(result)
            navigate('/login')})
            .catch(err => console.log(err))

    }

    return (
        <>
            <div class="bg-card text-card-foreground mx-auto w-full  rounded-lg border shadow-sm max-w-screen-lg mt-16">
                <div class="flex flex-col space-y-1.5 p-6">
                    <h3 class="whitespace-nowrap text-2xl font-semibold tracking-tight">Register for House Rental</h3>
                    <p class="text-muted-foreground text-sm">Sign up to start renting your dream home.</p>
                </div>
                <div class="p-6">
                    <form class="grid gap-4" onSubmit={handleSubmit}>
                        <div class="grid grid-cols-2 gap-4">
                            <div class="space-y-2">
                                <label for="name" class="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70" > Full Name </label>
                                <input name='name' id="name" placeholder="John Doe"  type='text' onChange={(e) => setName(e.target.value)} class="border-input bg-background ring-offset-background placeholder:text-muted-foreground focus-visible:ring-ring flex h-10 w-full rounded-md border px-3 py-2 text-sm file:border-0 file:bg-transparent file:text-sm file:font-medium focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50"  required />
                            </div>
                            <div class="space-y-2">
```

```
                                <label for="phone" class="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70" > Phone Number </label>

                                <input name='phone' id="phone" placeholder="+1 (555) 555-5555" type="tel" onChange={(e) => setPhone(e.target.value)} class="border-input bg-background ring-offset-background placeholder:text-muted-foreground focus-visible:ring-ring flex h-10 w-full rounded-md border px-3 py-2 text-sm file:border-0 file:bg-transparent file:text-sm file:font-medium focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50"  required />

                            </div>

                        </div>

                        <div class="relative h-10 w-full ">

                            <select name='user-type' id='user-type' onChange={(e) => setUser_type(e.target.value)}

                                class="peer h-full w-full rounded-[7px] border border-blue-gray-200 border-t-transparent bg-transparent px-3 py-2.5 font-sans text-sm font-normal text-blue-gray-700 outline outline-0 transition-all placeholder-shown:border placeholder-shown:border-blue-gray-200 placeholder-shown:border-t-blue-gray-200 empty:!bg-gray-900 focus:border-2 focus:border-gray-900 focus:border-t-transparent focus:outline-0 disabled:border-0 disabled:bg-blue-gray-50">

                                <option value="Owner">Owner</option>

                                <option value="Renter">Renter</option>



                            </select>

                            <label for="user-type"

                                class="before:content[' '] after:content[' '] pointer-events-none absolute left-0 -top-1.5 flex h-full w-full select-none text-[11px] font-normal leading-tight text-blue-gray-400 transition-all before:pointer-events-none before:mt-[6.5px] before:mr-1 before:box-border before:block before:h-1.5 before:w-2.5 before:rounded-tl-md before:border-t before:border-l before:border-blue-gray-200 before:transition-all after:pointer-events-none after:mt-[6.5px] after:ml-1 after:box-border after:block after:h-1.5 after:w-2.5 after:flex-grow after:rounded-tr-md after:border-t after:border-r after:border-blue-gray-200 after:transition-all peer-placeholder-shown:text-sm peer-placeholder-shown:leading-[3.75] peer-placeholder-shown:text-blue-gray-500 peer-placeholder-shown:before:border-transparent peer-placeholder-shown:after:border-transparent peer-focus:text-[11px] peer-focus:leading-tight peer-focus:text-gray-900 peer-focus:before:border-t-2 peer-focus:before:border-l-2 peer-focus:before:border-gray-900 peer-
```

```
focus:after:border-t-2 peer-focus:after:border-r-2 peer-focus:after:border-gray-900 peer-disabled:text-transparent peer-disabled:before:border-transparent peer-disabled:after:border-transparent peer-disabled:peer-placeholder-shown:text-blue-gray-500">

o                                   Select a User Type

o                               </label>

o                           </div>

o                           <div class="space-y-2">

o                               <label for="email" class="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70" > Email Address </label>

o                               <input name="email" id="email" placeholder="john@example.com" type="email" onChange={(e) => setEmail(e.target.value)} class="border-input bg-background ring-offset-background placeholder:text-muted-foreground focus-visible:ring-ring flex h-10 w-full rounded-md border px-3 py-2 text-sm file:border-0 file:bg-transparent file:text-sm file:font-medium focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50"  required />

o                           </div>

o                           <div class="space-y-2">

o                               <label  for="password" class="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70"> Password </label>

o                               <input name="password"  id="password" type="password" onChange={(e) => setpassword(e.target.value)} class="border-input bg-background ring-offset-background placeholder:text-muted-foreground focus-visible:ring-ring flex h-10 w-full rounded-md border px-3 py-2 text-sm file:border-0 file:bg-transparent file:text-sm file:font-medium focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50"  required />

o                           </div>

o                           <div class="flex items-center p-6">

o                               <button  type="submit"  class="block w-full select-none rounded-lg bg-gray-900 px-6 py-3 text-center align-middle font-sans text-xs font-bold uppercase text-white shadow-md shadow-gray-900/10 transition-all hover:shadow-lg hover:shadow-gray-900/20 focus:opacity-[0.85] focus:shadow-none active:opacity-[0.85] active:shadow-none disabled:pointer-events-none disabled:opacity-50 disabled:shadow-none"  >Register</button>

o                           </div>
```

```
                    </form>

                </div>

            </div>



        </>

    )

}



export default Register
```

- **User Folder**

1. **Owner**
2. **Renter**

Routing

Routing is managed using React Router. Here are the main routes:

- **/home -** Routes to home page
- **/login** - Routes to Login page
- **/about**-Routes to About page
- **/register**-Routes to Register page
- **/details**-Routes to all the property cards page
- **/owner**-Routes to Add Property page

- **/prp/:id-**Routes to each individual page of the property

Integration with Backend

The frontend communicates with the backend APIs hosted on [http://localhost:3001]. Key endpoints include:

A summary of the main API endpoints and their purposes:

User Authentication

- POST /register - Registers a new renter
- GET /login - Authenticates a renter

Home page

- GET /home - displays quote and buttons
- GET /about– Retrieves all details about us

Property

- GET /details - Retrieves all property details owner uploads
- POST /owner – add property details
- GET /prp/:id – get the property details that owner uploaded

User Interface (UI) Design

- Implemented using [React Framework and Tailwind CSS for CSS styles].
- This Frontend UI is made by using Mobile First Responsive Design Principle.So this website is fully Responsive in all kinds of Device.

Full Stack Development with MERN

API Development and Integration Report

Date	20-07-2024
Team ID	SWTID1720019632
Project Name	Project - House Rent App Using Mern
Maximum Marks	

Project Title: House Rent App Using Mern
Date: 20-07-2024
Prepared by: Ganesh.S

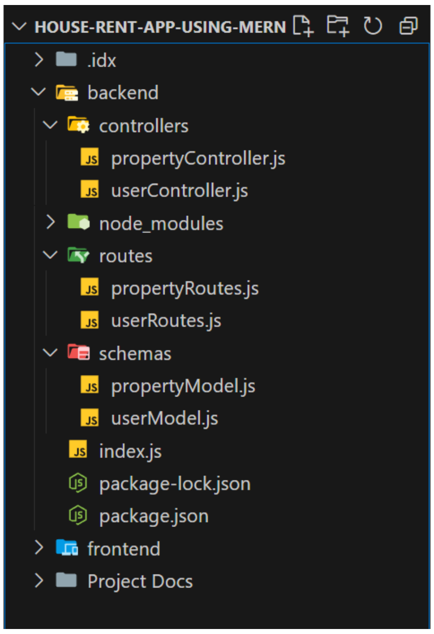
Objective

The objective of this report is to document the API development progress and key aspects of the backend services implementation for the House Rent App Using Mern project.

Technologies Used

- **Backend Framework:** Node.js with Express.js
- **Database:** MongoDB

Project Structure



```
HOUSE-RENT-APP-USING-MERN
  > .idx
  v backend
    v controllers
        propertyController.js
        userController.js
    > node_modules
    v routes
        propertyRoutes.js
        userRoutes.js
    v schemas
        propertyModel.js
        userModel.js
      index.js
      package-lock.json
      package.json
  > frontend
  > Project Docs
```

Key Directories and Files

1. **/Controllers**
   - Contains functions to handle requests and responses.
   - propertyController.js performs Getting Property Details Uploaded by Owner from DB. Individual Property Details for Dynamic Pages. Uploading Property Details from Owner To DB
   - userController .js performs registration and login
2. **/models**
   - Includes Mongoose schemas and models for MongoDB collections.
   - propertyModel.js contains the Mongoose schemas and models for Property collections
   - userModel.js contains the Mongoose schemas for models User collections

3. **/routes**
    - Defines the API endpoints and links them to controller functions.
    - propertyRoutes.js API endpoints (/propertyget, /addproperty, /propertyget/:id) in an Express router and links them to corresponding controller functions (PropertyDetails, uploadProperty, IndvidualDetails) from propertyController. Exports the router for use in the main application.
    - userRoutes.js API endpoints (`/register`, `/login`) in an Express router and links them to corresponding controller functions (`Register`, `Login`) from `userController`. Exports the router for use in the main application.

API Endpoints
A summary of the main API endpoints and their purposes:

User Authentication

- **POST /register -** Registers a new renter
- **GET /login** - Authenticates a renter

Home page

- **GET /home** - displays quote and buttons
- **GET /about**– Retrieves all details about us

Property

- **GET /details** - Retrieves all property details owner uploads
- **POST /owner** – add property details
- **GET /prp/:id –** get the property details that owner uploaded

Integration with Frontend
The backend communicates with the frontend via RESTful APIs. Key points of integration include:

- **User Authentication:** Tokens are passed between frontend and backend to handle authentication.
- **Data Fetching:** Frontend components make API calls to fetch necessary data for display and interaction.

Error Handling and Validation
Describe the error handling strategy and validation mechanisms:

- **Error Handling:** Centralized error handling using middleware.
- **Validation:** Input validation using libraries like Joi or express-validator.

Security Considerations
Outline the security measures implemented:

- **Authentication:** Secure token-based authentication.
- **Data Encryption:** Encrypt sensitive data at rest and in transit.

Full Stack Development with MERN

Database Design and Development Report

Date	20-07-2024
Team ID	SWTID1720019632
Project Name	House Rent App Using Mern
Maximum Marks	

Project Title: House Rent App Using Mern

Date: 20-07-2024

Prepared by: Hiruthick SM

Objective

The objective of this report is to outline the database design and implementation details for the House Rent App Using Mern project, including schema design and database management system (DBMS) integration.

Technologies Used

- **Database Management System (DBMS):** MongoDB
- **Object-Document Mapper (ODM):** Mongoose

Design the Database Schema

The database schema is designed to accommodate the following entities and relationships:

1. Users

- Attributes: name,email,password,user_type, phone, propertyname,propertytype,bedrooms,bathrooms,livingrooms,kitchen,sqft,address,propertyimage, propertydesc,rent

Implement the Database using MongoDB

The MongoDB database is implemented with the following collections and structures:

Database Name: houserent

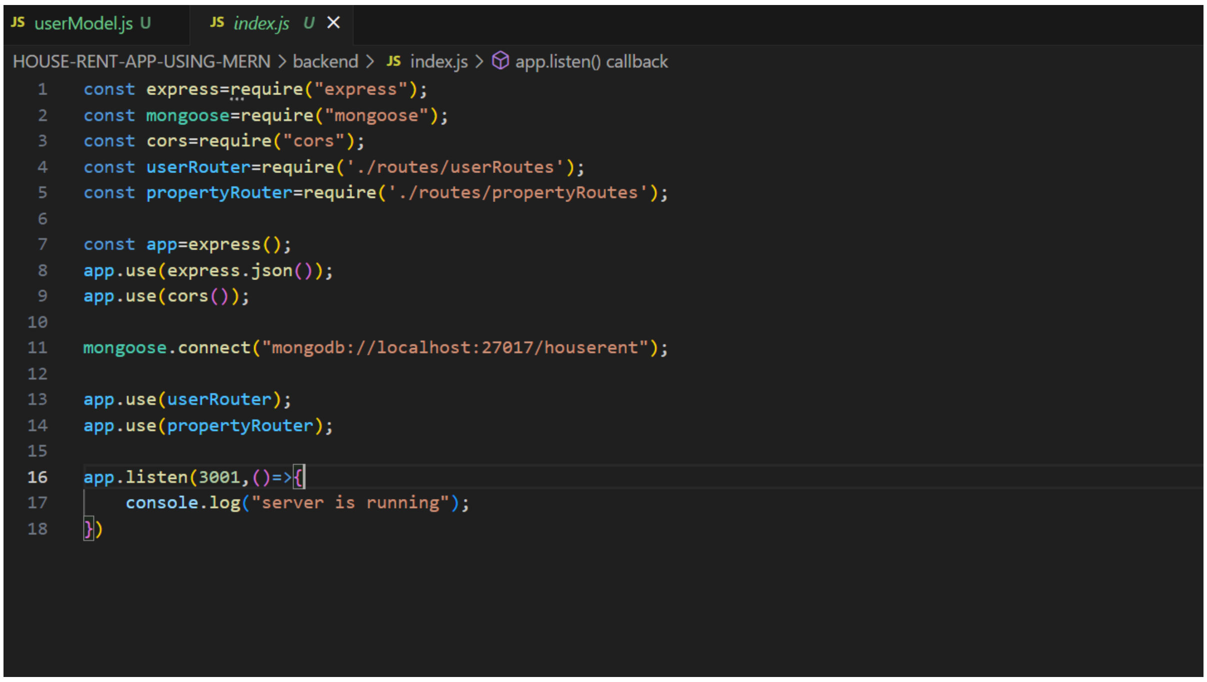
1. Collection: users

   - Schema:

```
const userSchema=new mongoose.Schema({
   name:String,
   email:String,
   password:String,
   user_type:String,
   phone:Number

})
```

Integration with Backend

- Database connection:

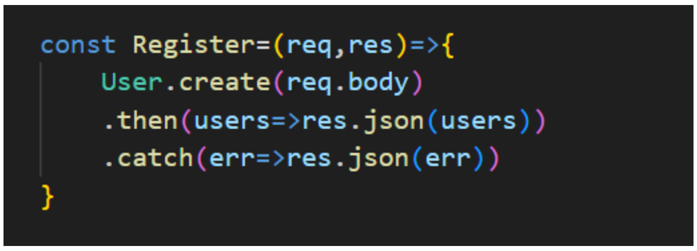


userModel.js U
index.js U
HOUSE-RENT-APP-USING-MERN > backend > index.js > app.listen() callback

```javascript
const express=require("express");
const mongoose=require("mongoose");
const cors=require("cors");
const userRouter=require('./routes/userRoutes');
const propertyRouter=require('./routes/propertyRoutes');

const app=express();
app.use(express.json());
app.use(cors());

mongoose.connect("mongodb://localhost:27017/houserent");

app.use(userRouter);
app.use(propertyRouter);

app.listen(3001,()=>{
    console.log("server is running");
})
```

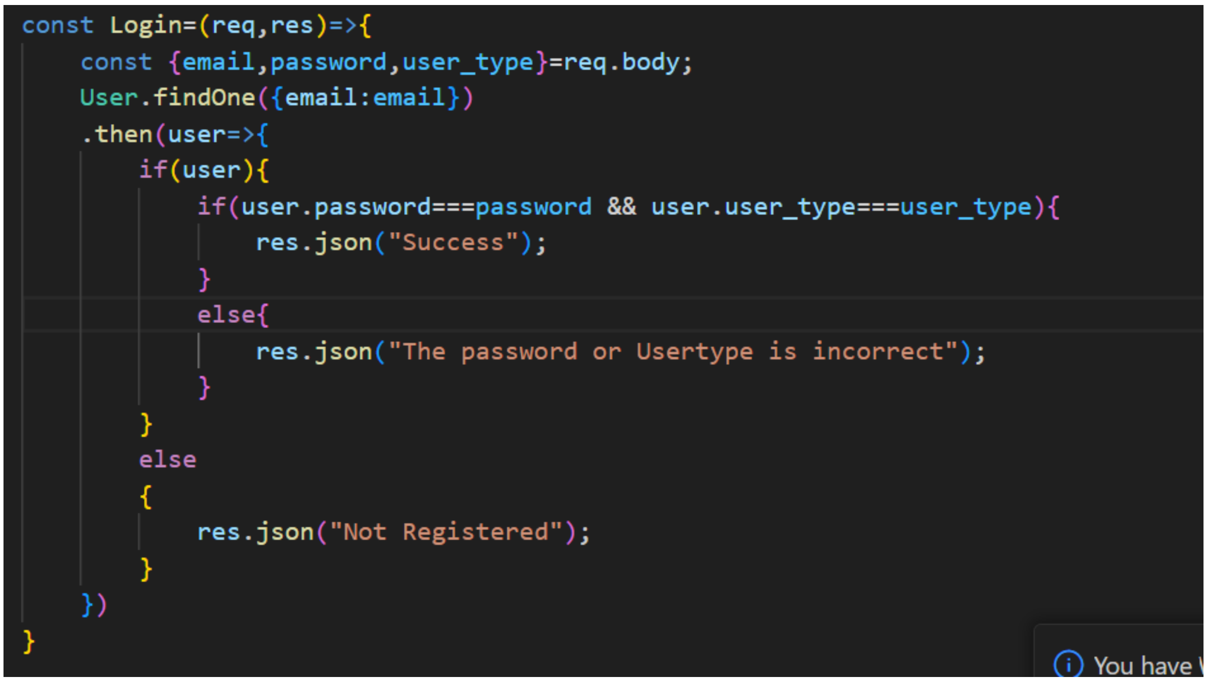
- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
    - User Management:

      Register:



```javascript
const Register=(req,res)=>{
    User.create(req.body)
    .then(users=>res.json(users))
    .catch(err=>res.json(err))
}
```

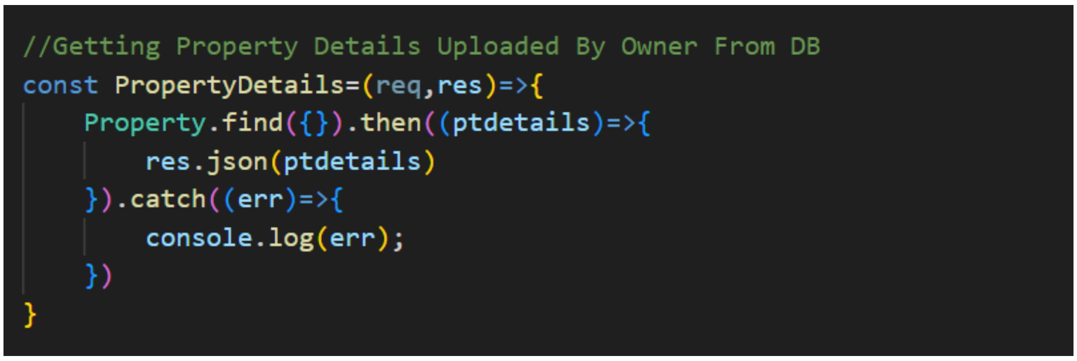
Login:



```
const Login=(req,res)=>{
    const {email,password,user_type}=req.body;
    User.findOne({email:email})
    .then(user=>{
        if(user){
            if(user.password===password && user.user_type===user_type){
                res.json("Success");
            }
            else{
                res.json("The password or Usertype is incorrect");
            }
        }
        else
        {
            res.json("Not Registered");
        }
    })
}
```

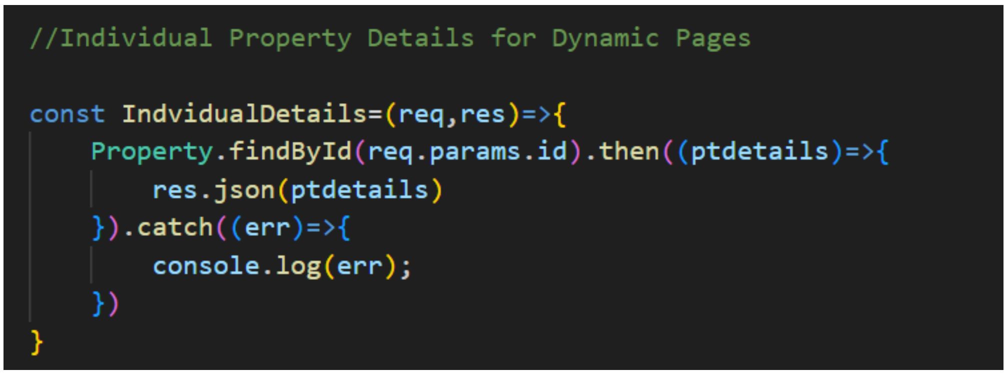
- Property Management:

PropertyDetails:



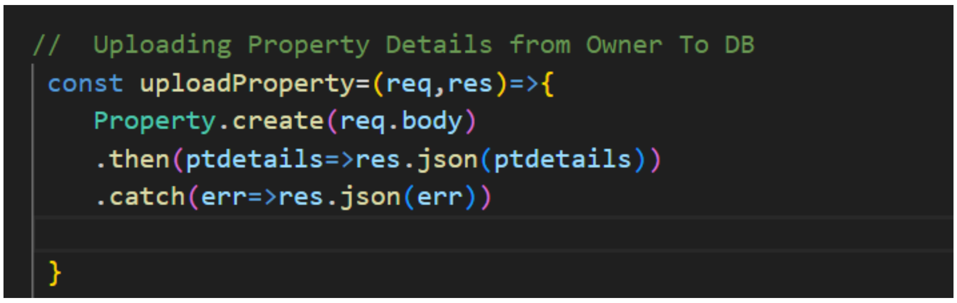
```
//Getting Property Details Uploaded By Owner From DB
const PropertyDetails=(req,res)=>{
    Property.find({}).then((ptdetails)=>{
        res.json(ptdetails)
    }).catch((err)=>{
        console.log(err);
    })
}
```

IndvidualDetails:



```
//Individual Property Details for Dynamic Pages

const IndvidualDetails=(req,res)=>{
    Property.findById(req.params.id).then((ptdetails)=>{
        res.json(ptdetails)
    }).catch((err)=>{
        console.log(err);
    })
}
```

uploadProperty:



```
//  Uploading Property Details from Owner To DB
const uploadProperty=(req,res)=>{
   Property.create(req.body)
   .then(ptdetails=>res.json(ptdetails))
   .catch(err=>res.json(err))

}
```

User Acceptance Testing (UAT) Template

Date	20-07-2024
Team ID	SWTID1720019632
Project Name	House Rent App Using Mern
Maximum Marks	

Prepared by: Mohnishwar D

Project Overview:

Project Name: House Rent App Using Mern

Project Description: A house rent app is typically a mobile or web application designed to help users find rental properties, apartments, or houses for rent. These apps often offer features to make the process of searching for and renting a property more convenient and efficient. Here are some common features you might find in a house rent app:

Property Listings: The app provides a database of available rental properties, complete with detailed descriptions, photos, location, rent amount, and other relevant information.

Search Filters: Users can apply various filters to narrow down their search results based on criteria such as location, rent range, property type (apartment, house, room, etc.), number of bedrooms, amenities, and more.

Contact Landlords/Property Managers: The app might provide a way for users to contact the property owners or managers directly through the app, often through messaging or email.

Project Version: 1

Testing Period: 17-07-2024 to 20-07-2024

Testing Scope:

- Register is to be tested.
- Login page is to be tested.
- AddProperty page is to be tested.

Testing Environment:

1. Register Page:

```
URL/Location: <Route path='/register' element={<Register/>}/>
```

Test Cases:

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
TC-001	Verify page load	1. Open the registration URL 2. Check if the page loads completely	The registration page should load completely without errors	The page loaded completely without errors	Pass
TC-002	Verify mandatory fields	1. Attempt to submit the form without filling any fields	Error messages should appear for all mandatory fields	Error messages are appeared to fill all mandatory fields	Pass
TC-	Valid email	1. Enter a	The form	The form is	

003	address	valid email address in the "Email Address" field 2. Submit the form	should be submitted successfully	submitted successfully	Pass
TC-004	Invalid email address	1. Enter an invalid email address (e.g., missing '@', domain) in the "Email Address" field 2. Submit	An error message should appear indicating an invalid email address	An error message is appeared indicating an invalid email address	Pass
TC-005	Valid phone number	1. Enter a valid phone number in the "Phone Number" field 2. Submit the form	The form should be submitted successfully	The form is submitted successfully	Pass
TC-006	Invalid phone number	1. Enter an invalid phone number (e.g., letters, fewer digits) in the "Phone Number" field 2. Submit	Characters cannot be entered unless it is numbers	Only Numbers can be entered	Pass
TC-007	Select user type	1. Select a user type from the "Select a User Type"	The registration should be done for the selected	The registration is done for the selected correspondi	Pass

		dropdown 2. Submit the form	corresponding user type	ng user type	
TC-008	Register successfully	1. Fill in all fields with valid data 2. Submit the form	Registration should be successful, and user should be redirected to another page	The Registration is successful and user is redirected to another page	Pass

2. Login Page

```
URL/Location: <Route path='/login' element={<Login/>}/>
```

Test Cases:

Test Case ID	Test Scenario	Test Steps	Test Steps	Actual Result	Pass/Fail
TC-001	Verify page load	1. Open the login URL 2. Check if the page loads completely	The login page should load completely without errors	The login page loads completely without errors	Pass
TC-002	Registered email address	1. Enter the email address 2. Check if the entered email is registered	The login page should be redirected to next page after registered email and password is correct	The entered email address is registered	Pass

TC-003	Incorrect email address	1. Enter the email address 2. Check if the entered email is registered	An error message should appear indicating an incorrect email address	An error message appeared indicating incorrect email address	Pass
TC-004	Valid Password	1. Enter the password 2. Check if the entered password is correct to the entered email address	The login page should be redirected to next page after registered email and password is correct	The login is redirected to next page	Pass
TC-005	Incorrect Password	1. Enter the password 2. Check if the entered password is correct to the entered email address	An error message should appear indicating an incorrect password	An error message appeared indicating incorrect password	Pass

3. AddProperty page

```
URL/Location: <Route path='/owner' element={<AddProperty/>}/>
```

Test Case ID	Test Scenario	Test Steps	Test Steps	Actual Result	Pass/Fail
TC-001	Verify page load	1. Open the "Add Your	The page should load	The page load is	Pass

TC-002	Verify mandatory	1. Attempt to submit the form	Error messages	Error message is	Pass
TC-003	Valid property	1. Enter a valid property name	The form should be	The form is submitted	Pass
TC-004	Select property	1. Select a property type	The form should be	The form is submitted	Pass
TC-005	Valid number of rooms	1. Enter a valid number of	The form should be	The form is submitted	Pass
TC-006	Invalid number of rooms	1. Enter an invalid number (e.g., letters, negative numbers) in the bathrooms, bedrooms, living rooms, and kitchens fields 2. Submit	The field cannot accept invalid numbers	The field did not accept invalid numbers	Pass
TC-007		1. Enter a valid square footage	The form should be	The form is submitted	Pass
TC-008	Invalid square footage	1. Enter an invalid square	The field cannot	The field did not	Pass
TC-009	Upload valid image	1. Upload a valid image link in the "Upload Image" field 2. Submit the form	The image should be uploaded successfully	The image was uploaded successful ly	Pass

TC-010	Valid monthly rent	1. Enter a valid monthly rent amount in the "Monthly Rent" field	The form should be submitted successfully	The form is submitted successful ly	Pass
TC-011	Invalid monthly rent	1. Enter an invalid monthly rent amount (e.g., letters, negative numbers) in the "Monthly Rent" field	The field cannot accept invalid numbers	The field did not accept invalid numbers	Pass
TC-012	Add property successfully	1. Fill in all fields with valid data 2. Submit the	Property should be added successfully, and user	The property is successful ly added and the	Pass

Sign-off:

Tester Name: Mohnishwar D

Date: 20-07-2024

Signature: Mohnishwar D