: REAL ESTATE WEBSITE :

A Project Report for Industrial Training and Internship
submitted by

SOUMYAJEET SAHA
ARPON SANA
ANUPAM JANA
ARITRA HALDAR
SOUMYADIP DAS

In the partial fulfillment of the award of the degree of

B.Tech
in the
CSE Department
Of

DR. SUDHIR CHANDRA SUR INSTITUTE OF TECHNOLOGY AND SPORTS COMPLEX

SurTech

At

Ardent Computech Pvt. Ltd.

ARDENT COMPUTECH PVT. LTD.
High-End Technology Training and Project

ARDENT®
GROUP OF COMPANIES
INTERNSHIP | DEVELOPMENT | PROJECT
An ISO 9001 : 2015 Certified Company

CERTIFICATE FROM SUPERVISOR

This is to certify that “Soumyajeet Saha(25500122042), Arpon Sana(25500122026), Anupam Jana(25500122020), Aritra Haldar(25500122024), Soumyadip Das(25500122094) ” have completed the project titled "Real Estate Web Browser – **Home Connect**" under my supervision during the period from “14 April 2025” to “10 August 2025” which is in partial fulfillment of requirements for the award of the B.Tech degree and submitted to the Department of “**Computer Science And Engineering**”of “**Dr. Sudhir Chandra Sur Institute Of Technology & Sports Complex**”.

_____

Signature of the Supervisor

Date: dd/mm/yy

Name of the Project Supervisor:

Ardent Computech Pvt. Ltd.
Drives you to the Industry
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision
"Real Estate Web Browser " is the bonafide work of

Name of the student: Soumyajeet Saha *Signature:*
Name of the student: Arpon Sana *Signature:*
Name of the student: Anupam Jana *Signature:*
Name of the student: Aritra Haldar *Signature:*
Name of the student: Soumyadip Das *Signature:*

SIGNATURE

Name :

PROJECT MENTOR

SIGNATURE

Name:

EXAMINERS

Ardent Original Seal

ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, ***Subhajit Santra*** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of Ardent Computech Pvt. Ltd. for their support.

CONTENT PAGE

1. COMPANY PROFILE

ARDENT (Ardent Computech Pvt. Ltd.), formerly known as Ardent Computech Private Limited, is an ISO 9001:2015 certified Software Development and Training Company based in India. Operating independently since 2003, the organization has recently undergone a strategic merger with ARDENT Technologies, enhancing its global outreach and service offerings.

ARDENT Technologies

ARDENT Technologies delivers high-end IT services across the UK, USA, Canada, and India. Its core competencies lie in the development of customized application software, encompassing end-to-end solutions including system analysis, design, development, implementation, and training. The company also provides expert consultancy and electronic security solutions. Its clientele spans educational institutions, entertainment companies, resorts, theme parks, the service industry, telecom operators, media, and diverse business sectors.

ARDENT Collaborations

ARDENT Collaborations, the Research, Training, and Development division of ARDENT (Ardent Computech Pvt. Ltd.), offers professional IT-enabled services and industrial training programs. These are tailored for freshers and professionals from B.Tech, M.Tech, MBA, MCA, BCA, and MSc backgrounds. ARDENT (Ardent Computech Pvt. Ltd.) provides Summer Training, Winter Training, and Industrial Training to eligible candidates. High-performing students may qualify for stipends, scholarships, and additional benefits based on performance and mentor recommendations.

Associations and Accreditations

ARDENT (Ardent Computech Pvt. Ltd.**)** is affiliated with the National Council of Vocational Training (NCVT) under the Directorate General of Employment & Training (DGET), Ministry of Labour & Employment, Government of India. The institution upholds strict quality standards under ISO 9001:2015 certification and is dedicated to bridging the gap between academic knowledge and industry skills through innovative training programs.

2. INTRODUCTION

The Real Estate Web Application is an innovative digital platform designed to simplify the process of buying, selling, and renting properties. With the increasing demand for transparent, fast, and user-friendly solutions in the real estate sector, this application aims to bridge the gap between property seekers, sellers, and agents.

This system provides users with an interactive interface to browse property listings, filter search results based on preferences such as location, budget, property type, and amenities, and view detailed descriptions with images and maps. Registered users can list their properties, manage inquiries, and connect directly with interested buyers or tenants.

By integrating modern web technologies, responsive design, and secure data handling, the application ensures an efficient and trustworthy experience. The project not only addresses the traditional challenges of the real estate market—such as limited accessibility, time-consuming searches, and lack of updated information.

Ultimately, this Real Estate Web Application serves as a comprehensive platform that streamlines property transactions, improves market visibility for sellers, and helps buyers make informed decisions quickly and effectively.

2A.OBJECTIVE

1. To simplify property transactions by providing a single online platform for buying, selling, and renting real estate.
2. To offer an intuitive user interface that allows users to easily search, filter, and view property details.
3. To enhance market accessibility by enabling property owners and agents to list properties with images, descriptions, and location maps.
4. To ensure up-to-date information through real-time updates on property availability and pricing.
5. To improve decision-making for buyers and tenants with advanced search filters, and detailed property insights.
6. To provide secure and reliable transactions by implementing safe user authentication and data protection measures.
7. To reduce time and effort involved in traditional property search and listing processes.

2B.SCOPE

The Real Estate Web Application is designed to serve as a comprehensive platform for property-related activities, catering to buyers, sellers and real estate agents. The scope of the project covers the following aspects:

1. **User Registration and Authentication** –
   Secure sign-up/login system for buyers, sellers, and agents to manage their accounts.
2. **Property Listings** –
   Ability for property owners and agents to post detailed property information, including images, pricing, location, and amenities.
3. **Advanced Search and Filters** –
   Search functionality based on location, price range, property type, and other key parameters.
4. **Interactive Property View** –
   Display of property details with photos, descriptions, contact information .
5. **Communication System** –
   Direct inquiry and messaging features between buyers and sellers/agents.
6. **Responsive Web Design** –
   Accessible from desktops, tablets, and smartphones for maximum user convenience.
7. **Real-Time Updates** –
   Live status changes for property availability, pricing, and new listings.
8. **Data Security and Privacy** –
   Secure handling of user data and prevention of unauthorized access.

3. SYSTEM ANALYSIS

3A.IDENTIFICATION OF NEED

System analysis is a crucial phase in the development of our project Real-Estate Web Application, involving creating an efficient and user-friendly platform.

The real estate industry has traditionally relied on physical visits, newspaper ads, and agent-based networking to connect buyers and sellers. While effective in the past, these methods often lead to delays, limited accessibility, high costs, and lack of transparency. With the rapid growth of internet usage and smartphone penetration, there is a growing demand for digital platforms that provide instant access to property information.

Key Needs Identified:

- **Time Efficiency –**

  Users require a faster way to search, compare, and shortlist properties without traveling extensively.

- **Wider Market Reach –**

  Sellers and agents need a platform to showcase their properties to a larger audience beyond local boundaries.

- **Real-Time Information –**

  Buyers and tenants expect updated listings with accurate availability, pricing, and property details.

- **Convenience –**

  Offering a platform accessible 24/7 from any location via web or mobile devices.

3B.FEASIBILITY STUDY

The feasibility study of our Real Estate Web project indicates that the concept is viable and promising across several key areas.

The Real Estate Web project is feasible and practical across all key areas. Technically, it can be developed using widely available tools like Flutter or React Native, with scalable backend solutions like Node.js. Economically, it has strong revenue potential through subscriptions, and premium content. Operationally, it is easy to use, scalable, and meets user needs. Legally, it is viable with compliance to data protection and content licensing laws. The estimated development time is 6 to 8 months, making it achievable within a reasonable timeframe and budget.

Overall, The Real Estate Web Application is feasible in terms of technical execution, economic viability, operational practicality, and timely completion, making it a valuable solution for the real estate sector .

3C.WORKFLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirements of the system. It is meant for use by the developers and will be the basic during the testing phase. Any changes made to the requirements in the future will have to go through a formal change approval process.

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In the waterfall model phases do not overlap.

Waterfall Model Design:

The waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure the success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In the Waterfall model, typically, the Outcome of one phase acts as the input for the next phase sequentially.

Iterative Waterfall Design:

Definition: The Iterative Waterfall Model is a variation of the traditional Waterfall model, which is a linear and sequential software development methodology. In the Iterative Waterfall Model, the development process is divided into small, manageable cycles, allowing for the revisiting and refinement of phases before progressing to the next stage. It combines the systematic structure of the Waterfall model with the flexibility of iterative development.

The sequential phases in Iterative Waterfall model are:

➢ **Requirement Gathering and Analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

➢ **System Design:** The requirement specifications from the first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

➢ **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

➢ **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing each unit. Post integration the entire system is tested for any faults and failures.

➢ **Deployment of the system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

➢ **Maintenance:** Some issues come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in progress and are seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name "Iterative Waterfall Model". In this model, phases do not overlap.

- **<u>Advantages:</u>**

1 . Flexibility: Iterations permit adjustments based on feedback.

2 . Early Delivery: Partial systems can be delivered incrementally.

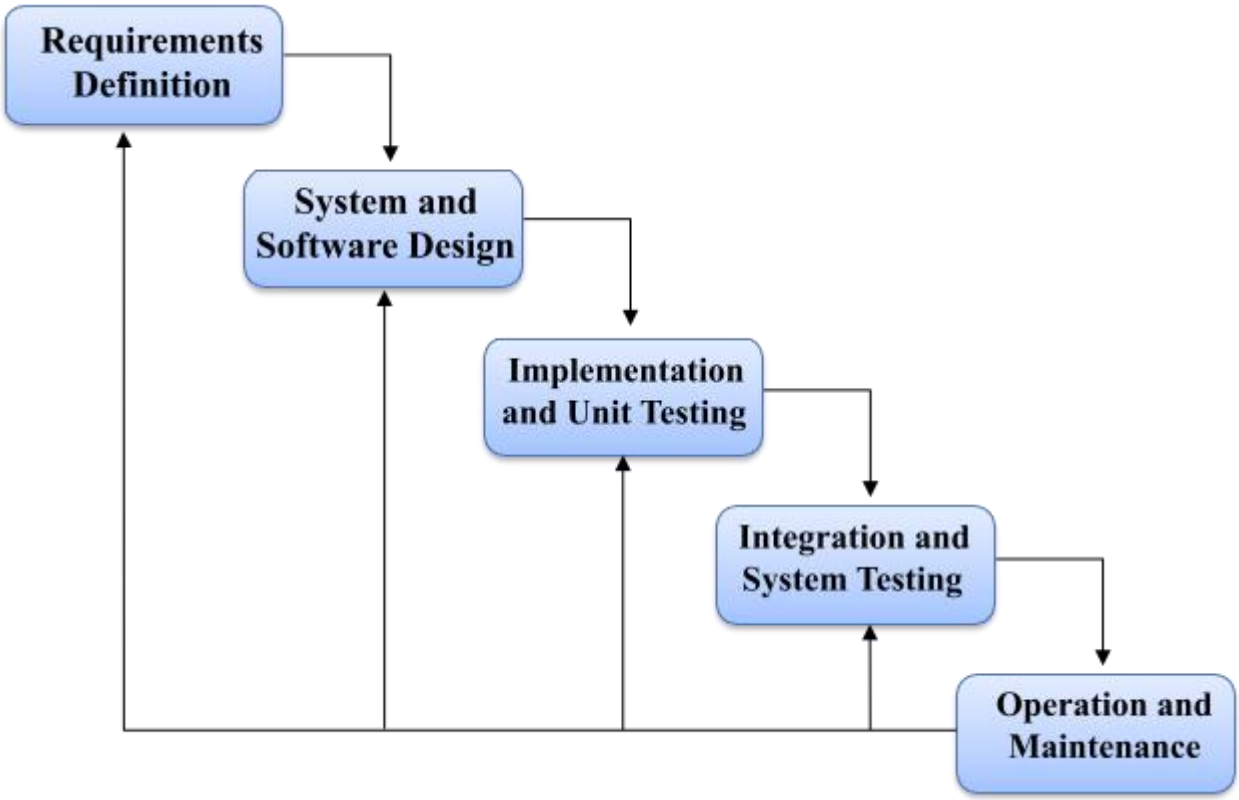
3 . Risk Management: Identifying and addressing issues early in the process.

- **<u>Disadvantages</u>:**

1. Increased Complexity: The iterative nature can make the process more complex.

2. Potential for Scope Creep: Frequent iterations may lead to scope changes.

3. Resource Intensive: Continuous revisiting of phases may demand more resources.



- **Applications:**

The Iterative Waterfall Model is suitable for projects with evolving or unclear requirements. It is commonly used in software development projects where regular feedback and refinement are essential.

Additionally, it is applicable in scenarios where partial system delivery is beneficial, allowing stakeholders to assess progress and make adjustments.

3D .STUDY OF THE SYSTEM

Modules: The modules used in this software are as follows:

- **Register:**

  1 . User Register: Here, the user will register to buy any property.
  2 . Admin Register: Here, the admin will register to handle all the databases.

- **Verify Account:** When a user login an OTP sends to the user Email to check the validity of the data.

- **Login:**

  1 . User Login: Here users will login to see all properties and buy any property.
  2 . Admin Login: Here admin will log in to handle all the data.

- **Home:** This page is to see feedback received from users.

- **Properties:**
  1. **User interface:** This page shows the available properties that the users can purchase. They can also search for properties.
  2. **Admin interface:** In this page, the Admin can view or delete properties.

- **Property details view:**
  1. **User interface**: If the users purchase any property then they can access the details view inside the property.
  2. **Admin interface**: Admin can add images, amenities of any properties.

- **About:** This page will show the details about the website developers.

- **Account:**
  1. **User Interface:**
     a) **My Profile:** Here, the user can see the details after login.
     b) **Dashboard:** Here, the user can see the all properties he/she has purchased.
  2. **Admin Interface:**

a) **Admin Profile:** Here, the Admin can see the details after login.

b) **Admin Dashboard:** Here the Admin can manage property listings.

3E.INPUT AND OUTPUT

The main inputs, outputs and the major function the details are:

INPUT:

1. Users can log in by entering their **credentials** on the login page.

OUTPUT:

1. Users can view the **available properties ,buy , rent and any properties and access the property's details view.**
2. The **admin** can access a **centralized database** that includes details of **users, add any properties and add any images, amenities** ensuring efficient system management.

3F.SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and non-functional requirements. The software requirements specification document enlists enough necessary requirements that are required for the project development. To derive the requirements we need to have a clear and thorough understanding of the project to be developed. This is prepared after detailed communication with the project team and the customer.

The developer is responsible for:

- Developing the system, which meets the SRS and solves all the requirements of the system.
- Demonstrating the system and installing the system at the client's location after acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

Functional Requirements:

A. User Registration & Authentication :

1. Secure sign-up/login for buyers, sellers, and agents.
2. Password encryption and account verification.

B. Search and Filter :

1. Search properties based on location, price, property type, and amenities.
2. Apply multiple filters for refined results.

C. Property Details View :

1. Display property description, images, location map, and contact details.

Hardware Requirements:

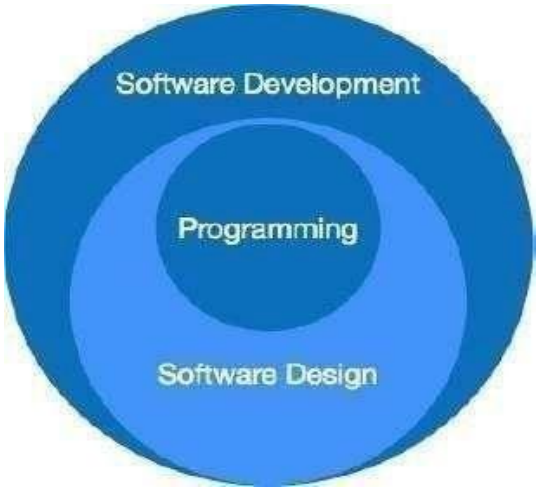
1 . Computer has Intel I3 Processor 2 .
8 GB RAM
3. SSD-ROM Drive

Software Requirements:

1. Windows 11 OS
2. Visual Studio Code
3. Mongo DB Atlas

3G.SOFTWARE ENGINEERING PARADIGM APPLIED

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in software engineering these paradigms stand. These can be combined into various categories, though each of them is containedin one another.



The programming paradigm is a subset of Software design paradigm which is further a subset of the Software development paradigm.

There are two levels of reliability. The first is meeting the right requirements. A careful and thorough systems study is needed to satisfy this aspect of reliability. The second level of systems reliability involves the actual work delivered to the user. At this level, the system's reliability is interwoven with software engineering and development.

There are three approaches to reliability.

1. **Error avoidance:** Prevents errors from occurring in software.
2. **Error detection and correction:** In this approach, errors are recognized whenever they are encountered, and correcting the error by the effect of the error of the system does not fail.
3. **Error to lerance:** In this approach, errors are recognized whenever they occur, but enables the system to keep running through degraded performance or Applying values that instruct the system to continue process.

4.SYSTEM DESIGN

4A. DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

DFD can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present for the system to do its job and shows the flow of data between the various parts of the system.

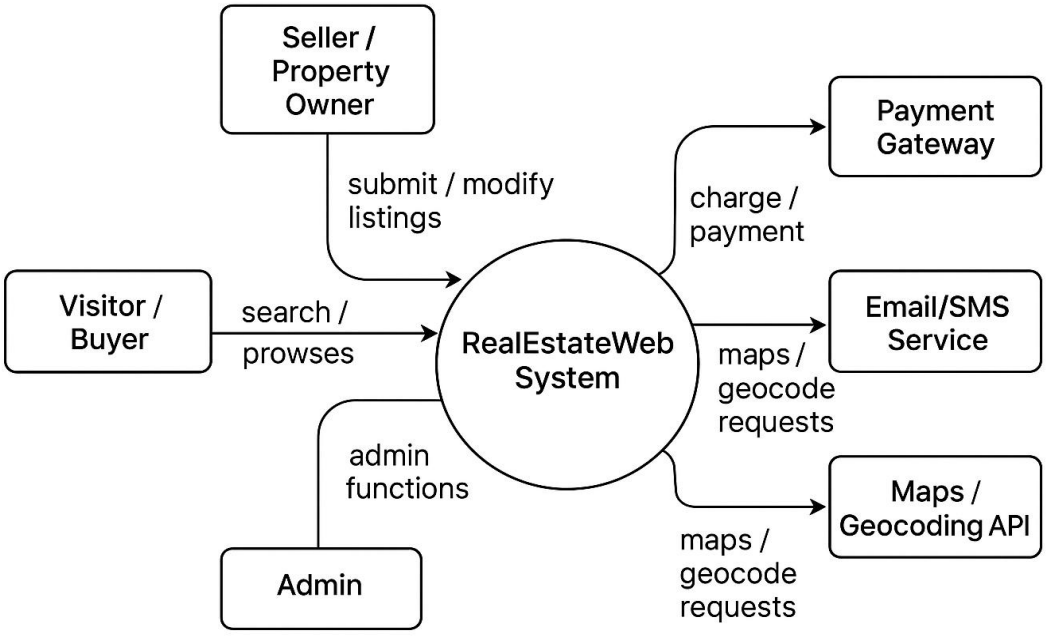
Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users can visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's data-flow diagrams can be drawn up and compared with.

How any system is developed can be determined through a data flow diagram model. In the course of developing a set of leveled data flow diagrams, the analyst/designer is forced to address how the system may be decomposed into component sub-systems and to identify the transaction data in the data model. Data flow diagrams can be used in both the Analysis and Design phase of the SDLC. There are different notations to draw data flow diagrams. Defining different visual representations for processes, data stores, data flow, and external entities.

DFD Notation:

	dataflow	Arrows showing direction of flow
	process	circles
	file	horizontal pair of lines
	data-source, sink	rectangular box

DFD Example:



Real Estate Web System

Steps to Construct Data Flow Diagram:

Four Steps are generally used to construct a DFD.

- Process should be named and referred for easy reference. Each name should be representative of the reference.
- The destination of flow is from top to bottom and from left to right.
- When a process is distributed into lower-level details they are numbered.
- The names of data stores, sources, and destinations are written in capital letters.

Rules for constructing a Data Flow Diagram:

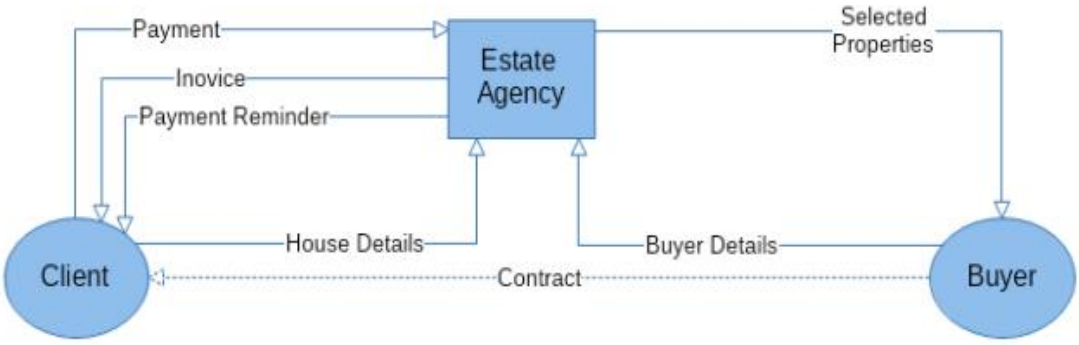
Arrows should not cross each other.

Squares, Circles, and Files must bear a name.

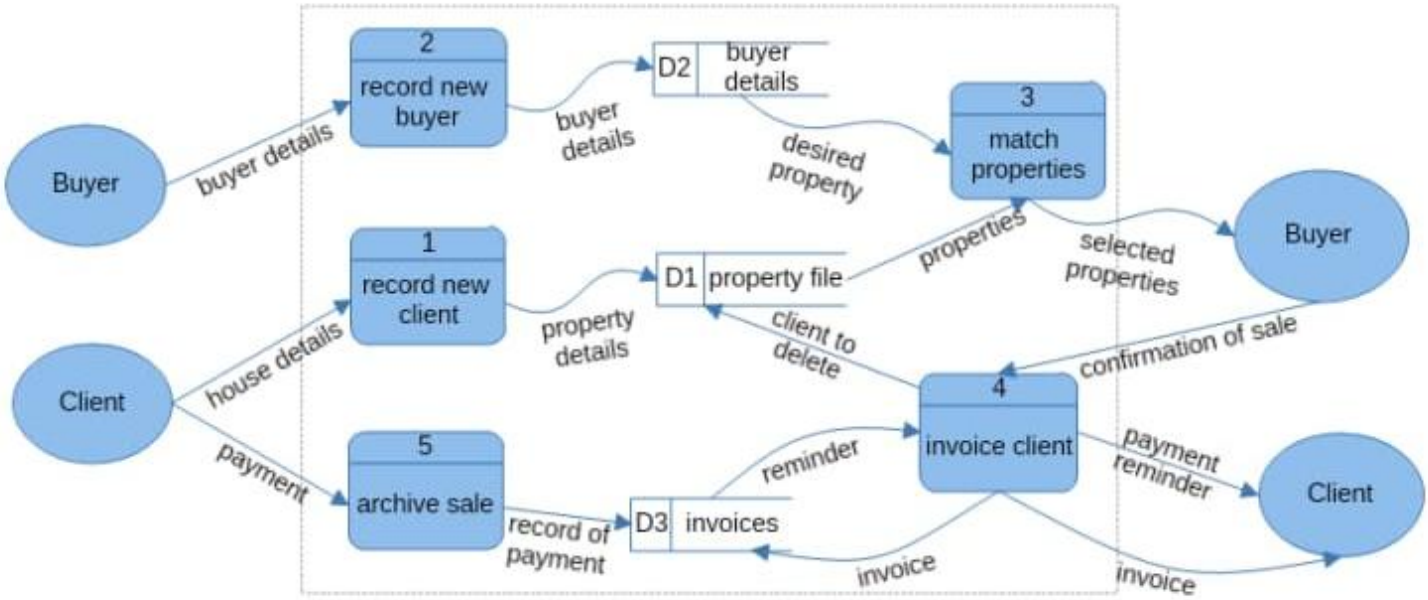
Decomposed data flow squares and circles can have the same names.

Draw all data flow around the outside of the diagram.

- **LEVEL 0 DFD OR CONTEXT DIAGRAM:**



- **LEVEL 1 DFD:**


Buyer
buyer details
2
record new buyer
buyer details
D2
buyer details
desired property
3
match properties
properties
selected properties
Buyer
Client
house details
1
record new client
property details
D1
property file
client to delete
confirmation of sale
4
invoice client
payment
5
archive sale
record of payment
D3
invoices
reminder
payment reminder
Client
invoice
invoice

4B.SEQUENCE DIAGRAM

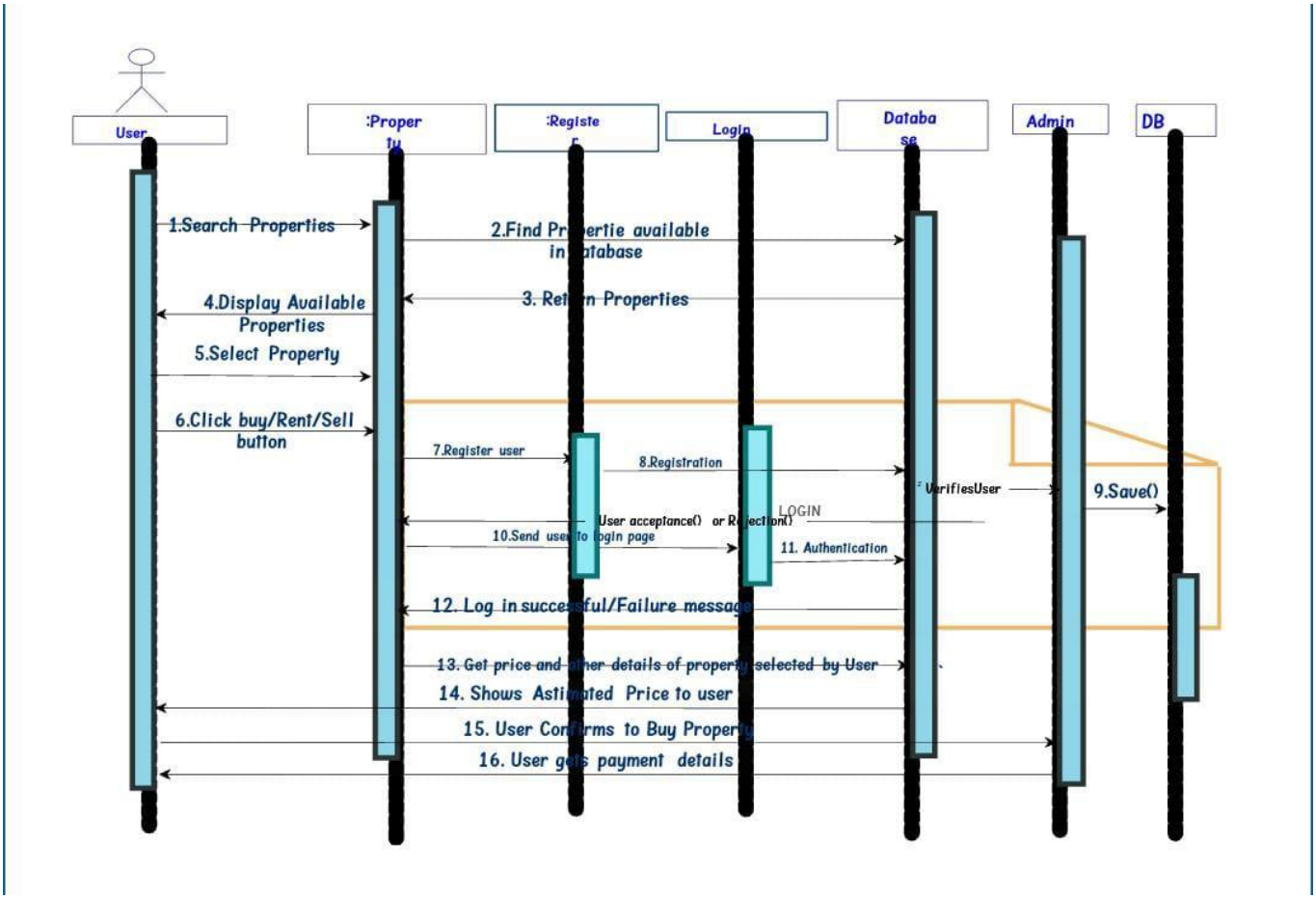
A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between several life lines .A sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, fragment, interaction, state invariant, continuation, and destruction occurrence.



A Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the use rand the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and a use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases, and their relationships. The diagram is used to model the system/subsystem of an application. A single-use case diagram captures a particular functionality of a system.

So, to model the entire system numbers of use case diagrams are used. The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because the other four diagrams (activity, sequence, collaboration, and State chart) are also having the same purpose. So, we will look into some specific purpose that will distinguish it from the other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modeled to present the outside view. So, in brief, the purposes of use case diagrams can be as follows:

Used to gather requirements of a system.

Used to get an outside view of a system.

Identify external and internal factors influencing the system.

How to draw Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed, the functionalities are captured in use cases.

So, we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So, in a brief when we are planning to draw use case diagram, we should have the following items identified.

Functionalities to be represented as a use case

Actors

Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So, after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

The name of a use case is very important. So, the name should be chosen in such a way so that it can identify the functionalities performed.

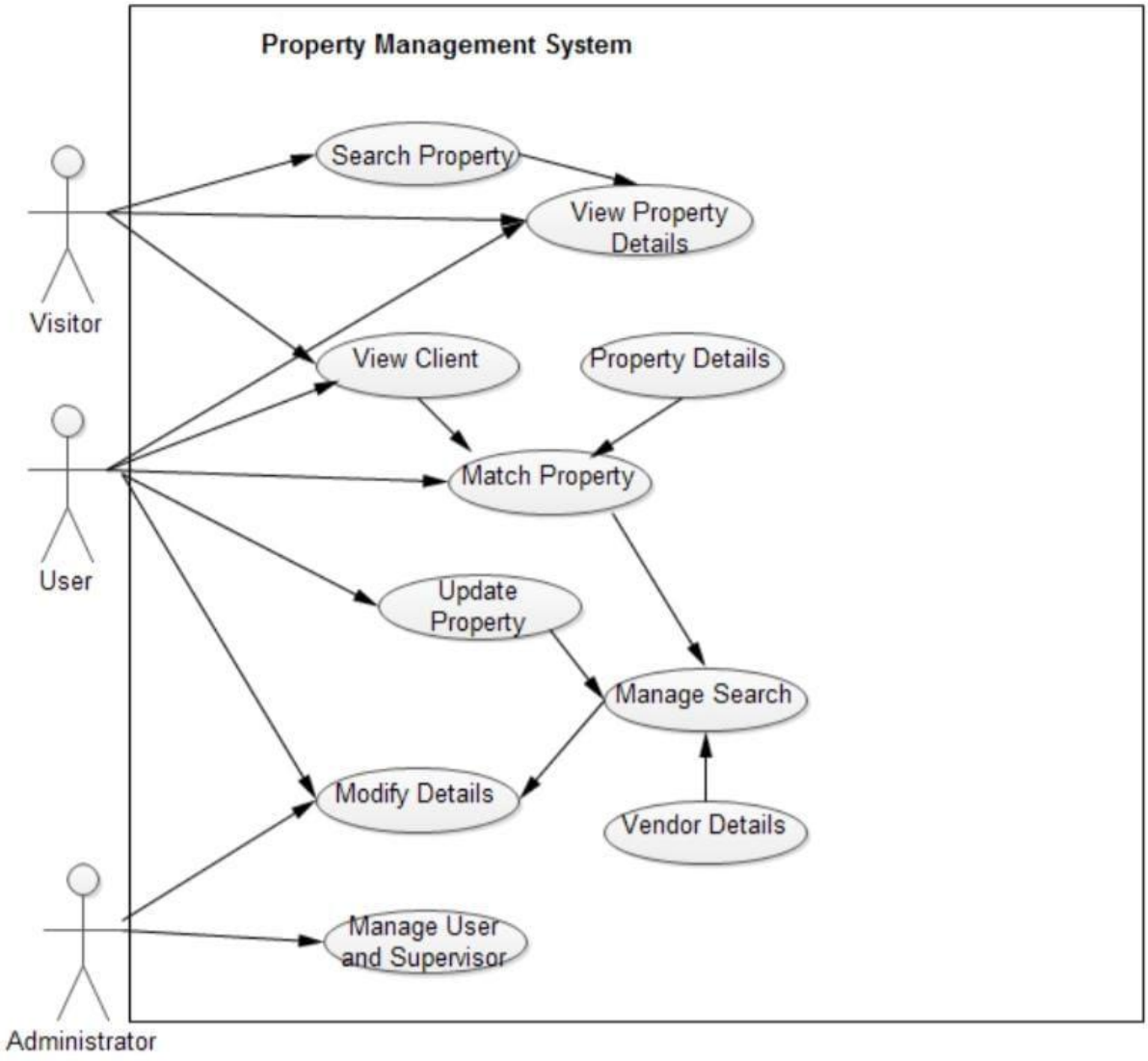
Give a suitable name for actors.

Show relationships and dependencies clearly in the diagram.

Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.

Use note whenever required to clarify some important point

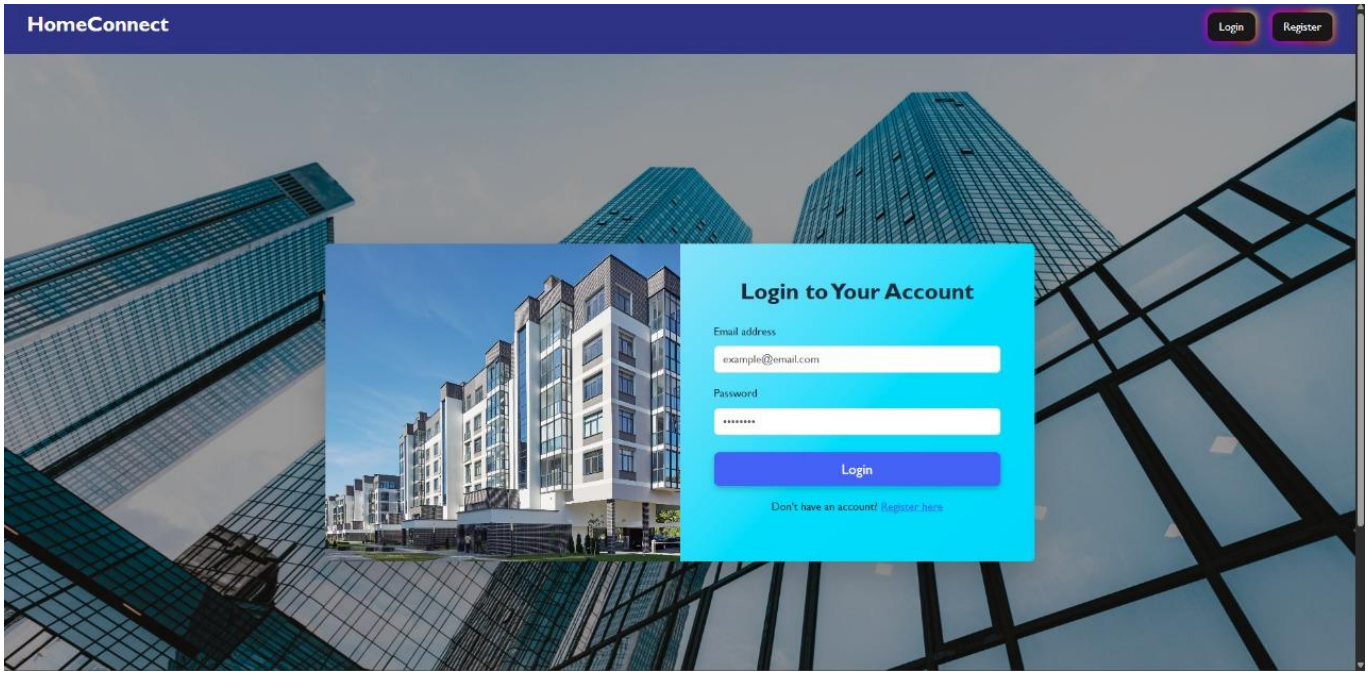
- **USE CASE DIAGRAM:**



5. UI SNAPSHOT

❖ **FRONTEND : -**

1) Login Page:



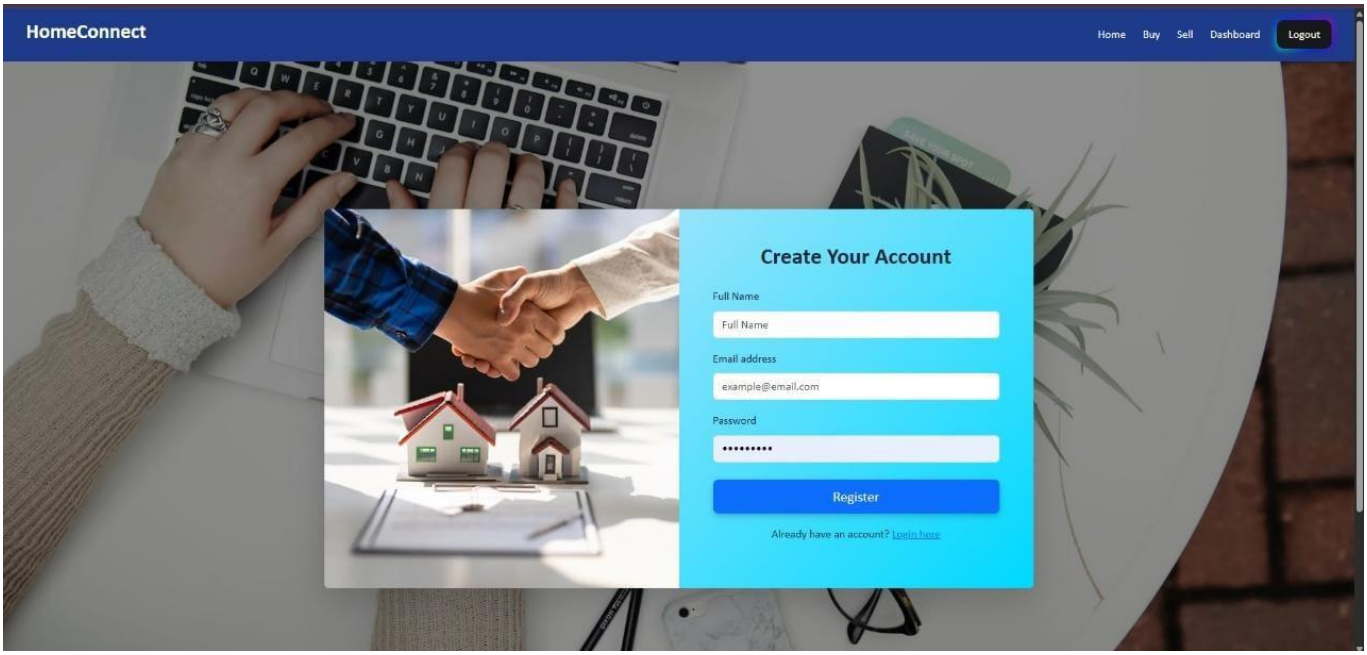
✓ **CODE**

```
import React from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import { useState } from 'react';

const Login = () => {
   const[form,setForm] = useState({
      email: '',
      password: ''
   });
   const navigate = useNavigate();
   const handlechange = (e) => {
      setForm({
         ...form,
         [e.target.name]: e.target.value
      });
   }
   const handlesubmit = async (e) => {
      e.preventDefault();
      try {
```

```
        const response = await axios.post('http://localhost:6005/api/auth/login', form);
        localStorage.setItem('token',response.data.token);
        navigate('/home');
      } catch (error) {
        console.error('Login error:', error);
          alert('An error occurred during login. Please try again.');
       }
  }

return <>
<div className="container mt-5">
   <div className="row justify-content-center">
     <div className="col-md-5">
       <div className="card shadow">
         <div className="card-body">
           <h3 className="card-title text-center mb-4">Login</h3>
           <form onSubmit={handlesubmit} noValidate>
             <div className="mb-3">
               <label className="form-label">Email address</label>
               <input
                 type="text"
                 className={`form-control `}
                 name="email"
                 onChange={handlechange}
                 required
               />
             </div>

             <div className="mb-3">
               <label className="form-label">Password</label>
               <input
                 type="password"
                 className="form-control"
                 name="password"
                 onChange={handlechange}
                 required
               />
             </div>

             <div className="d-grid">
               <button type="submit" className="btn btn-primary">Login</button>
             </div>
           </form>
           <p className="text-center mt-3">
             Don't have an account? <a href="/">Register here</a>
           </p>
```

```
          </div>
        </div>
      </div>
    </div>
  </div>
 </>
}

export default Login ;
```

2) Register Page:



✓ **CODE**

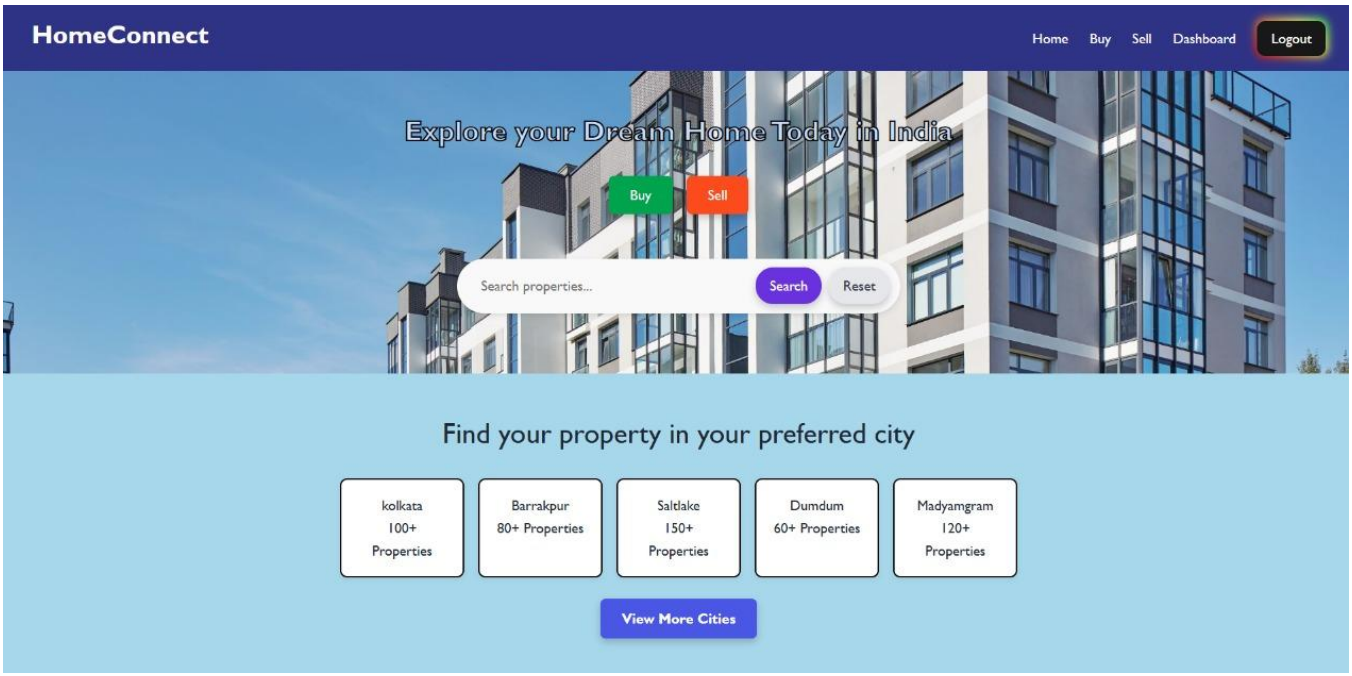
```
import React from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import { useState } from 'react';
import registerImage from '../assets/0x0.webp';

const Register = () => {
   const [form, setForm] = useState({
      name: '',
      email: '',
      password: ''
```

```
  });
  const navigate = useNavigate();
  const handlechange = (e) => {
    setForm({
      ...form,
      [e.target.name]: e.target.value
    });
  }
  const handlesubmit = async (e) => {
    e.preventDefault();
    if (!form.name || !form.email || !form.password) {
    alert("All fields are required!");
    return;
  }
  try {
    await axios.post('http://localhost:6005/api/auth/register', form);
    alert('Registration successful. Please login.');
    navigate('/login');
  } catch (err) {
    console.error('Registration error:', err);
    alert(err.response?.data?.message || 'Registration failed');
  }
  }
  return <>
  <div style={{
  minHeight: '100vh',
  backgroundImage: `linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0, 0.4)), url("\corinne-kutz-tMI2_-
r5Nfo-unsplash.jpg")`,
  backgroundSize: 'cover',
  backgroundPosition: 'center',
  backgroundRepeat: 'no-repeat',
  display: 'flex',
  alignItems: 'center',
  justifyContent: 'center',
  padding: '20px',
 }}>
  <div className="container-fluid min-vh-100 d-flex align-items-center justify-content-center ">
    <div className="row w-100 shadow-lg rounded overflow-hidden" style={{ maxWidth: '1000px',
background: 'linear-gradient(135deg, #e0eafc, #00d9ff)' }}>
      {/* Left Side - Image */}
```

```
<div className="col-md-6 d-none d-md-block p-0">
  <img
    src={registerImage}
    alt="Register Visual"
    className="img-fluid h-100 w-100"
    style={{ objectFit: 'cover' }}
  />
</div>

{/* Right Side - Form */}
<div className="col-md-6 p-5">
  <h2 className="text-center mb-4 fw-bold">Create Your Account</h2>
  <form onSubmit={handlesubmit} noValidate>
    <div className="mb-3">
      <label className="form-label">Full Name</label>
      <input
        type="text"
        className="form-control"
        name="name"
        onChange={handlechange}
        placeholder="Full Name"
        required
      />
    </div>

    <div className="mb-3">
      <label className="form-label">Email address</label>
      <input
        type="email"
        className="form-control"
        name="email"
        onChange={handlechange}
        placeholder="example@email.com"
        required
      />
    </div>

    <div className="mb-3">
      <label className="form-label">Password</label>
```

```
            <input
              type="password"
              className="form-control"
              name="password"
              onChange={handlechange}
              placeholder="••••••••"
              required
            />
          </div>

          <div className="d-grid mt-4">
            <button type="submit" className="btn btn-primary btn-lg">
              Register
            </button>
          </div>
        </form>
        <p className="text-center mt-3">
          Already have an account? <a href="/login">Login here</a>
        </p>
      </div>
    </div>
  </div>
  </div>

  </>
}

export default Register
```

3) PROPERTY PAGE(for admin and user):



● **components-pages-Home.jsx**

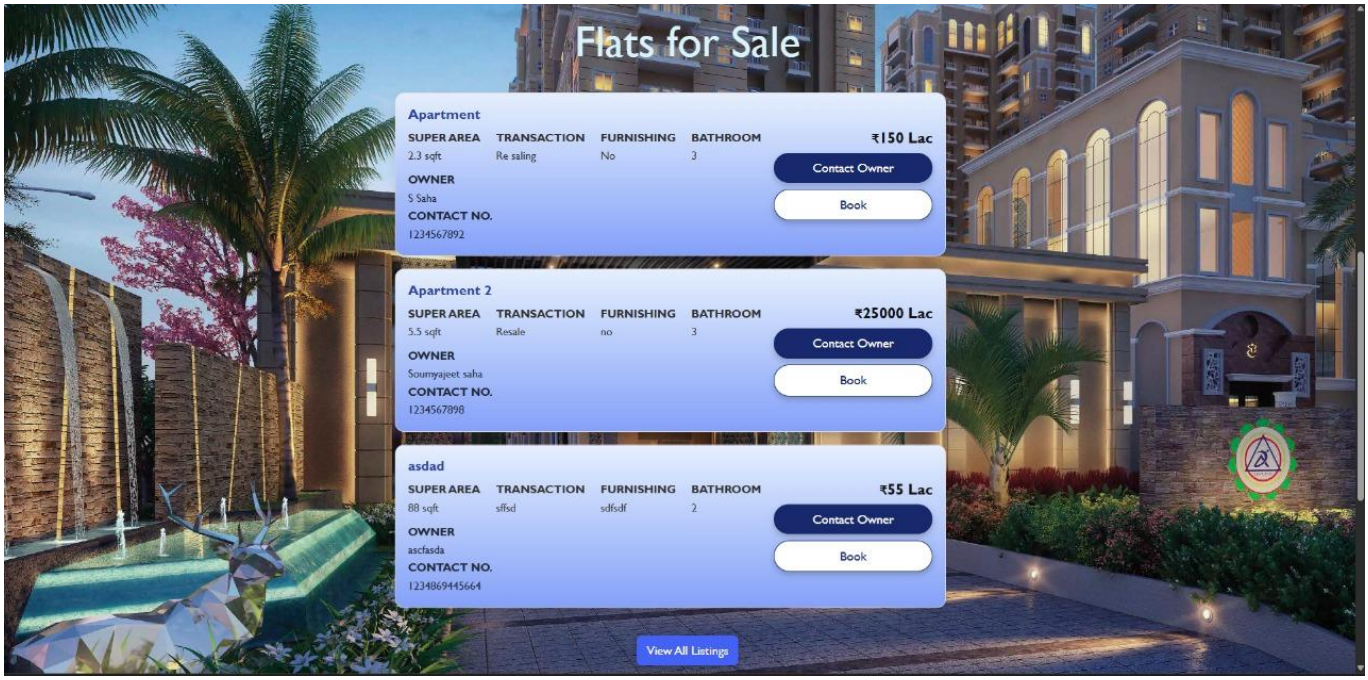
```
import React from 'react';
import {Link} from 'react-router-dom';
import '../components/Styles/Home.css';
import Button from 'react-bootstrap/Button';
import '../components/Styles/Buy.css';
import PropertyCard from './Propertycard.jsx';
import listings from '../data/properties.js';
import BuyCard from './BuyCard.jsx';
import axios from 'axios';
import { useEffect } from 'react';

const Home = () => {
    const [properties, setProperties] =
  React.useState([]);
 const [selectedproperty, setSelectedproperty]
  = React.useState(null);

 const fetchproperties = async () => {
        const response = await
  axios.get('http://localhost:6005/api/pr
  operties');
```

```
  setProperties(response.data);
};
useEffect(() => {
  fetchproperties();
}, []);
return <>
  <section class="hero">
   <h2>Explore your Dream Home Today in
  India</h2>
  <div class="btn-group">
                  <Link          to="/buy"><Button
  className="active">Buy</Button></Link>
                   <Link          to="/sell"><Button
  className="sell">Sell</Button></Link>
  </div>
  <div class="search-bar">
    <select>
      <option>Location</option>
      <option>Kolkata</option>
      <option>Dumdum</option>
      <option>Barrakpur</option>
  </select>
    <select>
      <option>Price</option>
      <option>Below 10 Lakh</option>
      <option>10 Lakh - 50 Lakh</option>
      <option>50 Lakh - 1 Crore</option>
      <option>Above 1 Crore</option>
    </select>
    <select>
      <option>Type</option>
      <option>Apartment</option>
      <option>Villa</option>
      <option>Plot</option>
      <option>Commercial</option>
    </select>
    <button>Search</button>
  </div>
```

```
    <p class="post-property">Are you an owner? <a href="#">Post property for free</a></p>
</section>

<section class="cities">
  <h2>Find your property in your preferred city</h2>
  <div class="city-grid">
     <div class="city-box">kolkata<br/>100+ Properties</div>
    <div class="city-box">Barrakpur<br/>80+ Properties</div>
     <div class="city-box">Saltlake<br/>150+ Properties</div>
    <div class="city-box">Dumdum<br/>60+ Properties</div>
                        <div class="city-box">Madyamgram<br/>120+ Properties</div>
  </div>
    <button class="view-more">View More Cities</button>
</section>

<section class="latest-properties">
  <div className="p-6 max-w-6xl mx-auto">
      <h1 className='text-3xl font-semibold text-gray-800 text-center mb-6 mt-10' style={{ fontSize:'3.5rem',marginTop: '20px', marginLeft: '20px',marginBottom:'30px' }}>Flats for Sale</h1>
    {/* Listings */}
       <div className="flex flex-col items-center">
    <BuyCard
  properties={properties}
  fetchproperties={fetchproperties}
  setSelectedproperty={setSelectedproperty}
```

```
      />
        </div>
      </div>
      <div className='mx-auto d-flex justify-content-
      center align-items-center '><Link
      to='/buy'><Button className="btn btn-
      primary " style={{ marginTop: '20px',
      marginLeft: '20px' }}> View All Listings
      </Button></Link></div>
    </section>

    <section class="services">
      <h2>Explore our real estate services</h2>
      <div class="service-grid">
          <div class="service-card"><div class="img-
      box"><img src='https://lawsikho.com/blog/wp-
      content/uploads/2021/08/9-most-common-types-
      of-business-
      agents.png'></img></div>Agents</div>
          <div class="service-card"><div class="img-
      box"><img src='https://rsbuilder.in/wp-
      content/uploads/2024/04/Builder-
      e1475660822261.jpg'></img></div>Builders</
      div>
          <div class="service-card"><div class="img-
      box"><img src='https://encrypted-
      tbn0.gstatic.com/images?q=tbn:ANd9GcT8AEA
      sXU60B5M0E3171NgBPVe37hn4WPQHfQ&s'
      ></img></div>Interior Decorators</div>
          <div class="service-card"><div class="img-
      box"><img src='https://encrypted-
      tbn0.gstatic.com/images?q=tbn:ANd9GcRmtBd
      KwiW97g0pAwl1zqv6Aj_H8PXNHKpwSA&s'
      ></img></div>Other</div>
      </div>
    </section>
    </>
  }
export default Home
```

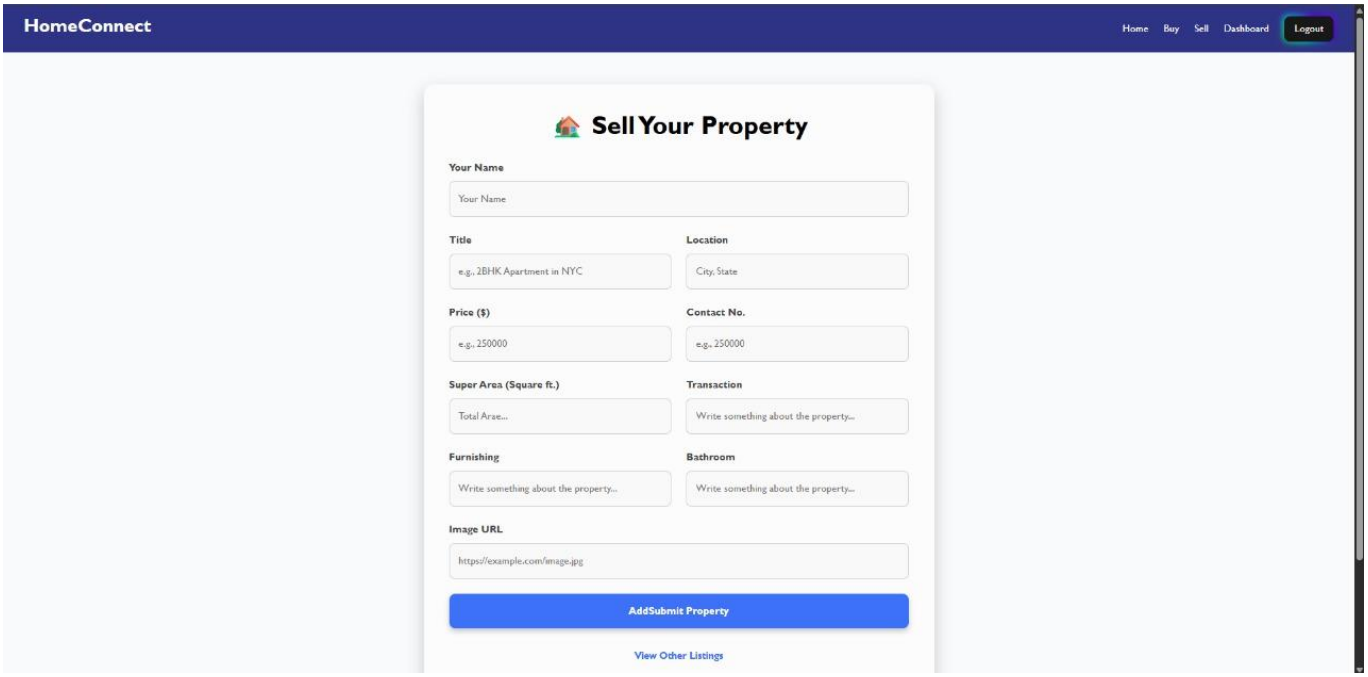
- **components – PropertyCard.jsx :-**




```
import React from 'react';
import listings from '../data/properties.js';
 // Assuming you have a listings data file
 import './Styles/Propertycard.css';
const PropertyCard = ({ listing }) => {
  return (
    <div className="property-card">
      {/* Left: Image Section */}
      <div className="property-image-container">
        <img src={listing.imageUrl} alt="property" className="property-image" />
        <span className="image-tag">{listing.imageCount}+ Photos</span>
        <span className="posted-tag">Posted: {listing.posted}</span>
      </div>

      {/* Middle: Details */}
      <div className="property-details">
        <h3 className="property-title">{listing.title}</h3>
        <div className="property-info">
          <div><strong>SUPER AREA</strong><br />{listing.area}</div>
          <div><strong>TRANSACTION</strong><br />Resale</div>
          <div><strong>FURNISHING</strong><br />{listing.furnishing}</div>
          <div><strong>BATHROOM</strong><br />{listing.bathroom}</div>
        </div>
        <p className="owner-name">Owner: {listing.owner}</p>
      </div>

      {/* Right: Price & Buttons */}
      <div className="property-price-actions">
        <div className="price">₹{listing.price / 100000} Lac</div>
```

```
            <div className="price-per-sqft">₹{listing.pricePerSqft} per sqft</div>
            <button className="contact-btn">Contact Owner</button>
            <button className="phone-btn">Get Phone No.</button>
          </div>
        </div>
      );
    };

    export default PropertyCard;
```

● components – Sell.jsx :-




```
import React from 'react';
import axios from 'axios';
import Sellform from './Sellform';
import Propertylist from './ViewSell';
import { useEffect } from 'react';

const Sell = () => {
  const [properties, setProperties] = React.useState([]);
  const [selectedproperty, setSelectedproperty] = React.useState(null);

  const fetchproperties = async () => {
```

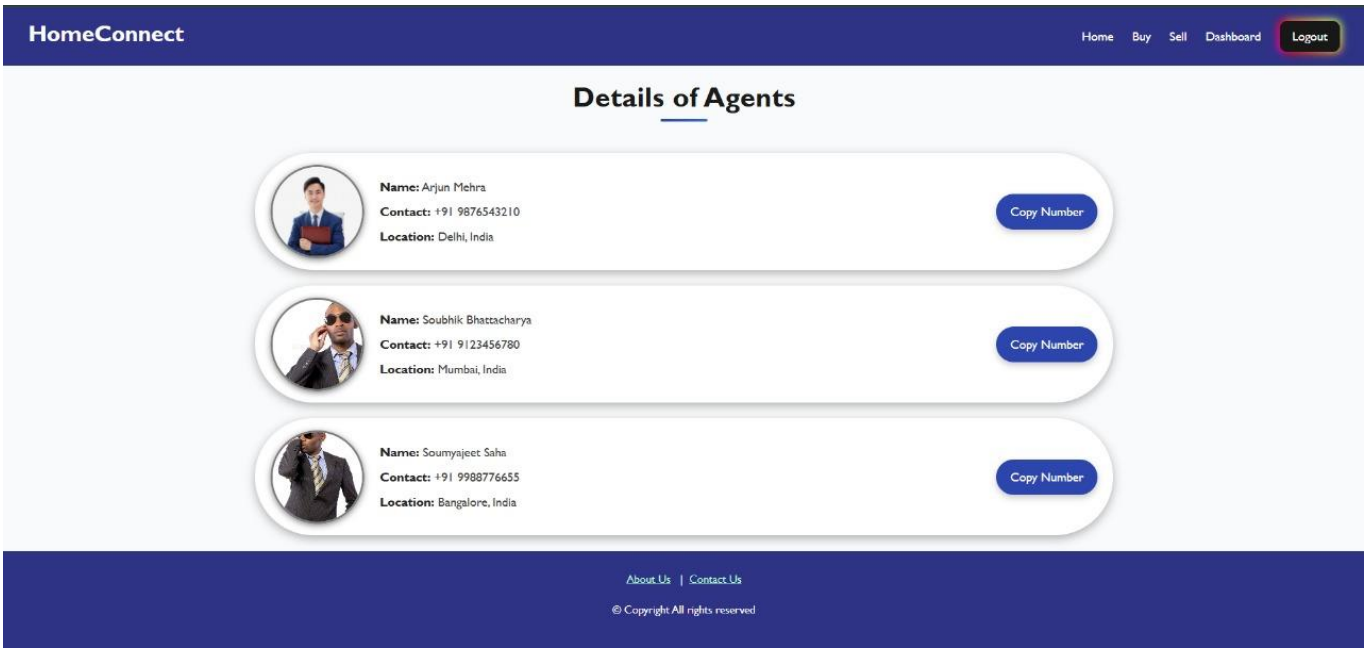
```
    const response = await axios.get('http://localhost:6005/api/properties');
    setProperties(response.data);
  };
  useEffect(() => {
    fetchproperties();
  }, []);

  return <>
  <Sellform
    fetchproperty={fetchproperties}
    selectedproperty={selectedproperty}
    setSelectedproperty={setSelectedproperty}
    />
  <Propertylist
    properties={properties}
    fetchproperties={fetchproperties}
    setSelectedproperty={setSelectedproperty}
    />
  </>
}

export default Sell
```

- **components – Buy.jsx :-**


```
import React from 'react';
import propertyData from '../data/datas';
import Cardslider from './Cardslider';
import '../components/Styles/Buy.css';
import PropertyCard from './Propertycard.jsx';
import listings from '../data/properties.js'; // Assuming you have a listings data file
import Button from 'react-bootstrap/Button';
import BuyCard from './BuyCard.jsx';
import axios from 'axios';
import { useEffect } from 'react';

function Buy() {
  const [properties, setProperties] = React.useState([]);
  const [selectedproperty, setSelectedproperty] = React.useState(null);

  const fetchproperties = async () => {
    const response = await axios.get('http://localhost:6005/api/properties');
    setProperties(response.data);
  };
  useEffect(() => {
```

```
  fetchproperties();
 }, []);
 return <>
 <section className="ss">
 <section>
  <div className="p-6 max-w-6xl mx-auto">
   <h1 className='text-3xl font-semibold text-gray-800 text-center mb-6 mt-10' style={{ fontSize:'3.5rem',marginTop: '20px', marginLeft: '20px',marginBottom:'30px' }}>Flats for Sale</h1>
   {/* Listings */}
   <BuyCard
  properties={properties}
  fetchproperties={fetchproperties}
  setSelectedproperty={setSelectedproperty}
  />
  </div>
 </section>
 <div className='mx-auto d-flex justify-content-center align-items-center '><Button className="btn btn-primary  " style={{ marginTop: '20px', marginLeft: '20px',marginBottom:'30px' }}> View All Listings </Button></div>
 <section className="bg-gray-50 py-8">
 <div className="max-w-6xl mx-auto px-4">
  <h1 className="text-3xl font-semibold text-gray-800 text-center mb-6">
   Properties in Kolkata
  </h1>
  {/* Carousel */}
  <Cardslider properties={propertyData} />
 </div>
</section>
 </section>
 </>
}
export default Buy
```

4) AGENT PAGE (for user);



✓ CODE

```
import React from 'react';
import { Link } from 'react-router-dom';
import '../components/Styles/Agents.css';

const agentsData = [
  {
    img: "https://img.lovepik.com/element/40247/1471.png_1200.png",
    name: "Arjun Mehra",
    contact: "+91 9876543210",
    location: "Delhi, India",
  },
  {
    img: "https://thumbs.dreamstime.com/b/government-agent-black-male-bodyguard-wearing-sunglasses-blacksuit-35529797.jpg?w=768"
    name: "Soubhik Bhattacharya",
    contact: "+91 9123456780",
    location: "Mumbai, India",
  },
  {
    img: "https://thumbs.dreamstime.com/b/government-agent-black-male-bodyguard-wearing-sunglasses-blacksuit-35529813.jpg"
    name: "Soumyajeet Saha",
    contact: "+91 9988776655",
    location: "Bangalore, India",
  },
];
```

```
const Agents = () => {
   const copyToClipboard = (number) => {
   navigator.clipboard.writeText(number);
   alert(`Copied: ${number}`);
  };
  return <>
   <div className="agents-container">
    <h1 className="page-title">Details of Agents</h1>
    <ul className="agents-list">
     {agentsData.map((agent, index) => (
      <li key={index} className="agent-card">
       <img src={agent.img} alt={`${agent.name} Photo`}/>
       <div className="agent-details">
        <p>
         <strong>Name:</strong> {agent.name}
        </p>
        <p>
         <strong>Contact:</strong> {agent.contact}
        </p>
        <p>
         <strong>Location:</strong> {agent.location}
        </p>
       </div>
       <div className="button-container">
        <button onClick={() => copyToClipboard(agent.contact)}>
         Copy Number
        </button>
       </div>
      </li>
     ))}
    </ul>
   </div>

  </>
}

export default Agents
```

5) PAYMENT_PAGE:

✓ **CODE**

```
import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';

const PaymentPage = () => {
   const [amount, setAmount] = useState(100);
   const navigate = useNavigate();

   // Function to handle payment
   const handlePayment = async () => {
      try {
         const { data } = await axios.post('http://localhost:5600/api/payment/create-order', { amount });
         const { orderId, currency } = data;

         const options = {
            key: 'your_razorpay_key_id', // from .env
            amount: amount * 100,
            currency,
            name: "REAL ESTATE CRUD",
            description: "Service Payment",
            order_id: orderId,
            handler: async function (response) {
               await axios.post('http://localhost:5600/api/payment/save-payment', {   // localhost api URL
will be replaced with backend
                  orderId,
                  paymentId: response.razorpay_payment_id,
                  amount,
                  currency,
                  status: 'success'
               });
               alert("Payment Successful!");
               navigate('/dashboard'); // Redirect to dashboard after payment
            },
            prefill: {
               name: "User",
               email: "user@example.com",
               contact: "9999999999"
            },
            theme: { color: "#3399cc" }
         };
```

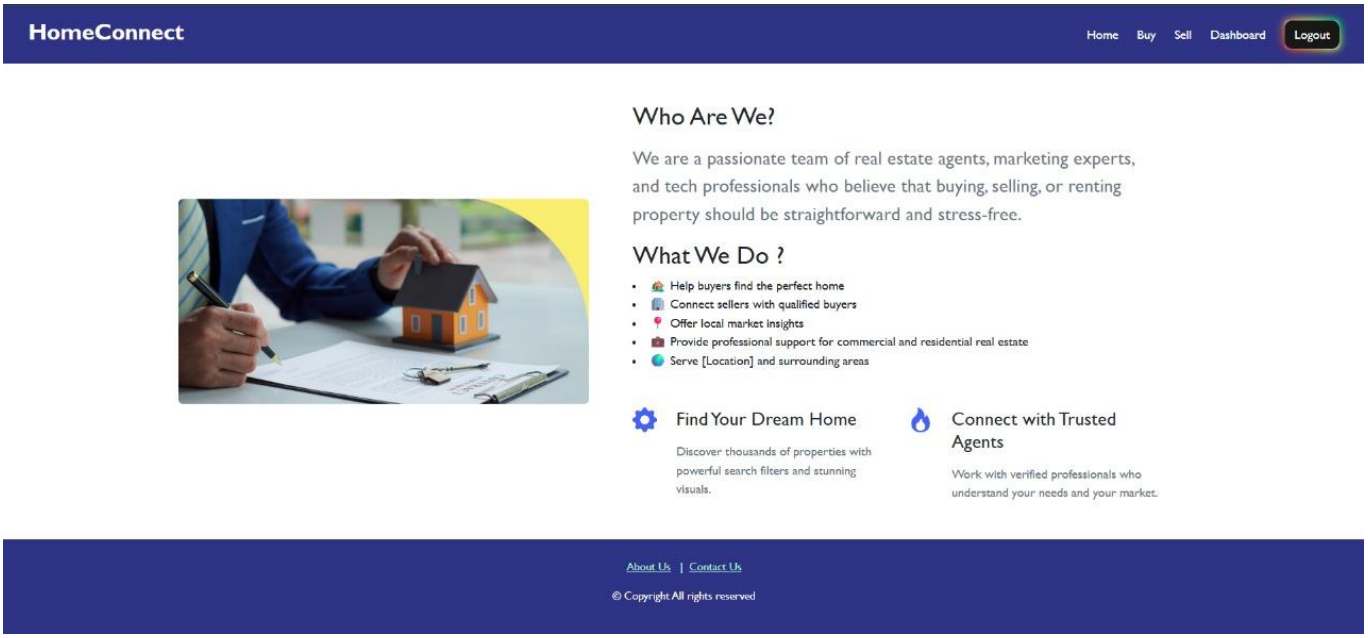
```
            const rzp = new window.Razorpay(options);
            rzp.open();
        } catch (error) {
            console.error("Payment Error", error);
        }
    };

    return (
        <div style={{ padding: 20 }}>
            <h2>Eco Smart City Payment</h2>
            <input type="number" value={amount} onChange={(e) => setAmount(e.target.value)} />
            <button onClick={handlePayment}>Pay Now</button>
        </div>
    );
};

export default PaymentPage;
```

6) ViewSell Page :

```
import React from 'react'
import axios from 'axios'

import './Styles/Viewsell.css';
const Propertylist = ({properties,fetchproperties,setSeletedproperty}) => {
 const deleteproperty = async(id)=> {
  await axios.delete(`http://localhost:6005/api/properties/${id}`);
  fetchproperties();
 }
  return <>
 <div className="container mt-4">
     <h2 className="mb-4">Property List</h2>
     {properties.map((x) => (
   <div className="property-card" key={x._id}>
     <div className="property-left">

     </div>
     <div className="property-middle">
       <h5 className="property-title">{x.title}</h5>
       <div className="property-details">
         <p><strong>SUPER AREA:</strong> {x.superarea || '—'} sqft</p>
         <p><strong>TRANSACTION:</strong> {x.transaction || 'Resale'}</p>
         <p><strong>FURNISHING:</strong> {x.furnishing || 'Unfurnished'}</p>
         <p><strong>BATHROOM:</strong> {x.bathroom || '—'}</p>
         <p><strong>OWNER:</strong> {x.owner || '—'}</p>
         <p><strong>CONTACT NO.:</strong> {x.contact || '—'}</p>
       </div>
     </div>
     <div className="property-right">
       <div className="price">₹{x.price/1000} Lac</div>
```

```
      <button className='btn btn-dark' onClick={()=>setSeletedproperty(x)}>Edit</button>
      <button className='btn btn-danger' onClick={()=>deleteproperty(x._id)}>Delete</button>
    </div>
   </div>
  ))}
</div>
  <div className="empty-state">
    {properties.length === 0 && <p>No properties available. Please add a property.</p>}
  </div>
  </>
}

export default Propertylist;
```

7) About Page:



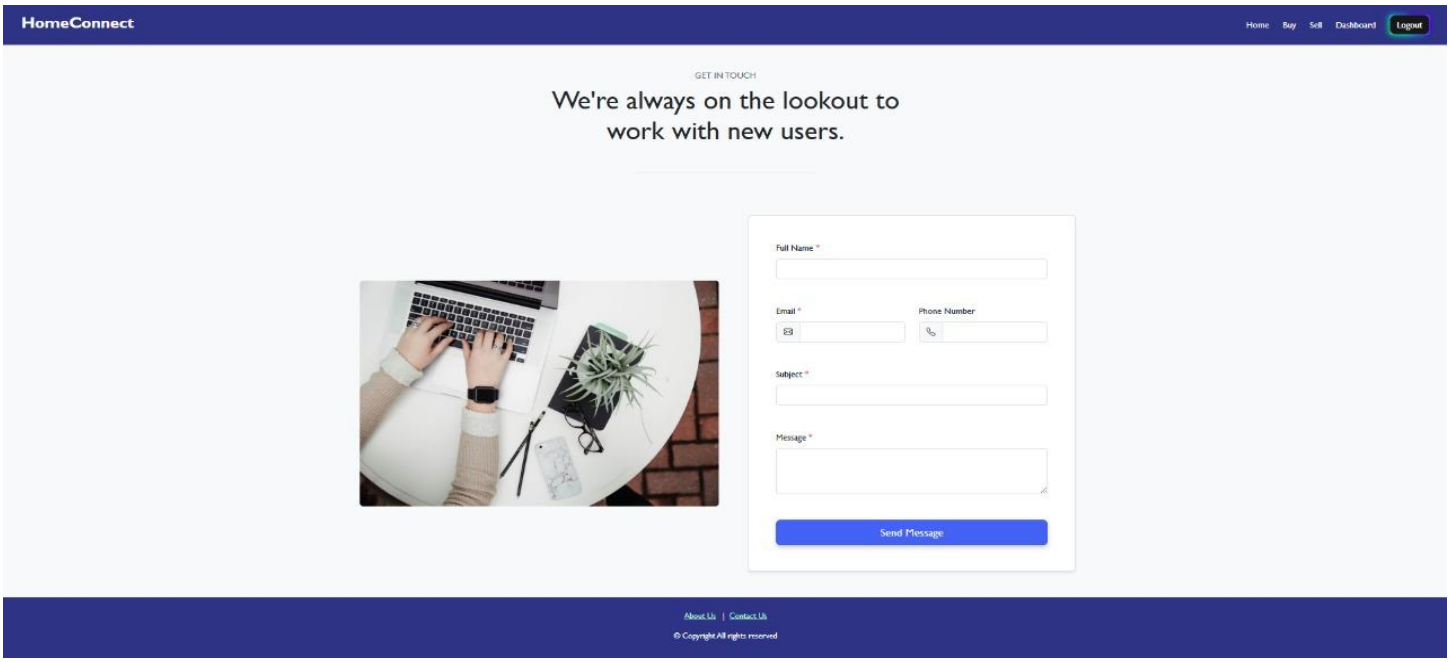

```
import React from 'react' ;

function About() {
  return <>
  <link rel="stylesheet"
href="https://unpkg.com/bootstrap@5.3.3/dist/css/bootstrap.min.css"></link>
<section class="py-3 py-md-5">
  <div class="container">
    <div class="row gy-3 gy-md-4 gy-lg-0 align-items-lg-center">
      <div class="col-12 col-lg-6 col-xl-5">
        <img class= "img-fluid rounded" src="https://cms-assets.themuse.com/media/lead/what-
is-real-estate.png" alt="" />
      </div>
      <div class="col-12 col-lg-6 col-xl-7">
        <div class="row justify-content-xl-center">
          <div class="col-12 col-xl-11">
            <h2 class="mb-3">Who Are We?</h2>
            <p class="lead fs-4 text-secondary mb-3">We are a passionate team of real estate
agents, marketing experts, and tech professionals who believe that buying, selling, or renting
property should be straightforward and stress-free. </p>
            <h2>What We Do ?</h2>
            <p class="mb-5">
            <li>🏠 Help buyers find the perfect home</li>
            <li>🏢 Connect sellers with qualified buyers</li>
            <li> 📍 Offer local market insights</li>
            <li>💼 Provide professional support for commercial and residential real estate</li>
            <li>🌍 Serve [Location] and surrounding areas</li>
```

```
          </p>
          <div class="row gy-4 gy-md-0 gx-xxl-5X">
            <div class="col-12 col-md-6">
              <div class="d-flex">
                <div class="me-4 text-primary">
                  <svg xmlns="http://www.w3.org/2000/svg" width="32" height="32"
fill="currentColor" class="bi bi-gear-fill" viewBox="0 0 16 16">
                    <path d="M9.405 1.05c-.413-1.4-2.397-1.4-2.81 0l-.1.34a1.464 1.464 0 0 1-
2.105.872l-.31-.17c-1.283-.698-2.686.705-1.987 1.987l.169.311c.446.82.023 1.841-.872
2.105l-.34.1c-1.4.413-1.4 2.397 0 2.81l.34.1a1.464 1.464 0 0 1 .872 2.105l-.17.31c-.698
1.283.705 2.686 1.987 1.987l.311-.169a1.464 1.464 0 0 1 2.105.872l.1.34c.413 1.4 2.397 1.4
2.81 0l.1-.34a1.464 1.464 0 0 1 2.105-.872l.31.17c1.283.698 2.686-.705 1.987-1.987l-.169-
.311a1.464 1.464 0 0 1 .872-2.105l.34-.1c1.4-.413 1.4-2.397 0-2.81l-.34-.1a1.464 1.464 0 0 1-
.872-2.105l.17-.31c.698-1.283-.705-2.686-1.987-1.987l-.311.169a1.464 1.464 0 0 1-2.105-
.872l-.1-.34zM8 10.93a2.929 2.929 0 1 1 0-5.86 2.929 2.929 0 0 1 0 5.858z" />
                  </svg>
                </div>
                <div>
                  <h2 class="h4 mb-3">Find Your Dream Home</h2>
                  <p class="text-secondary mb-0">Discover thousands of properties with powerful
search filters and stunning visuals.</p>
                </div>
              </div>
            </div>
            <div class="col-12 col-md-6">
              <div class="d-flex">
                <div class="me-4 text-primary">
                  <svg xmlns="http://www.w3.org/2000/svg" width="32" height="32"
fill="currentColor" class="bi bi-fire" viewBox="0 0 16 16">
                    <path d="M8 16c3.314 0 6-2 6-5.5 0-1.5-.5-4-2.5-6 .25 1.5-1.25 2-1.25 2C11 4 9
.5 6 0c.357 2 .5 4-2 6-1.25 1-2 2.729-2 4.5C2 14 4.686 16 8 16Zm0-1c-1.657 0-3-1-3-2.75 0-
.75.25-2 1.25-3C6.125 10 7 10.5 7 10.5c-.375-1.25.5-3.25 2-3.5-.179 1-.25 2 1 3 .625.5 1 1.364
1 2.25C11 14 9.657 15 8 15Z" />
                  </svg>
                </div>
                <div>
                  <h2 class="h4 mb-3">Connect with Trusted Agents</h2>
                  <p class="text-secondary mb-0">Work with verified professionals who understand
your needs and your market.</p>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```
  </div>
</section>
  </>
}

export default About
```

8) CONTACT PAGE :

✓ **CODE:**

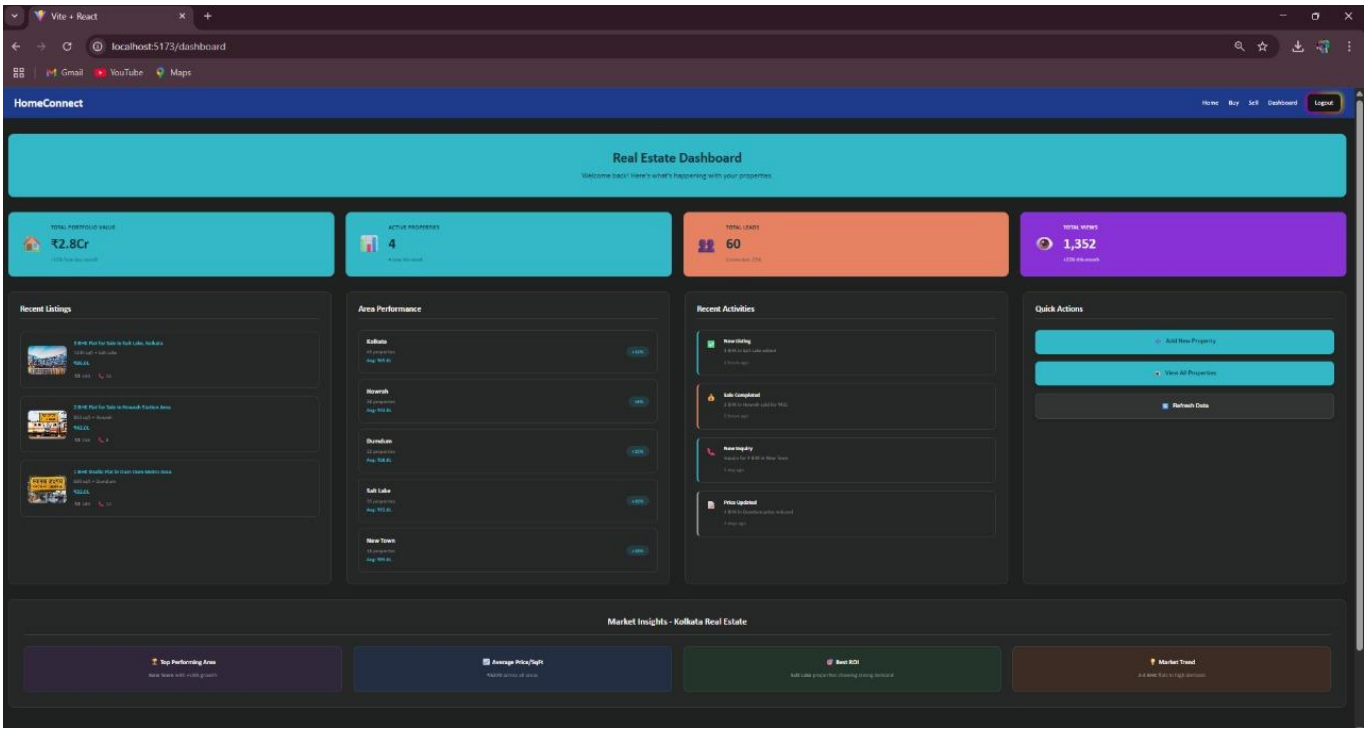



```
import React from 'react'

function Contact() {
  return <>
  <link rel="stylesheet" href="https://unpkg.com/bootstrap@5.3.3/dist/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://unpkg.com/bs-brain@2.0.4/components/contacts/contact-4/assets/css/contact-4.css"></link>
<section class="bg-light py-3 py-md-5" >
  <div class="container">
    <div class="row justify-content-md-center">
      <div class="col-12 col-md-10 col-lg-8 col-xl-7 col-xxl-6">
        <h3 class="fs-6 text-secondary mb-2 text-uppercase text-center">Get in Touch</h3>
        <h2 class="display-5 mb-4 mb-md-5 text-center">We're always on the lookout to work with new users.</h2>
        <hr class="w-50 mx-auto mb-5 mb-xl-9 border-dark-subtle" />
      </div>
    </div>
  </div>

  <div class="container">
    <div class="row gy-3 gy-md-4 gy-lg-0 align-items-xl-center">
      <div class="col-12 col-lg-6">
        <img class="img-fluid rounded" loading="lazy" src="https://images.unsplash.com/photo-1487611459768-bd414656ea10?q=80&w=1170&auto=format&fit=crop&ixlib=rb-4.1.0&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D" alt="Get in Touch" />
      </div>
```

```
    <div class="col-12 col-lg-6">
      <div class="row justify-content-xl-center">
        <div class="col-12 col-xl-11">
          <div class="bg-white border rounded shadow-sm overflow-hidden">

            <form action="#!">
              <div class="row gy-4 gy-xl-5 p-4 p-xl-5">
                <div class="col-12">
                  <label for="fullname" class="form-label">Full Name <span class="text-
danger">*</span></label>
                  <input type="text" class="form-control" id="fullname" name="fullname" value="" required
/>
                </div>
                <div class="col-12 col-md-6">
                  <label for="email" class="form-label">Email <span class="text-danger">*</span></label>
                  <div class="input-group">
                    <span class="input-group-text">
                      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
class="bi bi-envelope" viewBox="0 0 16 16">
                        <path d="M0 4a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v8a2 2 0 0 1-2 2H2a2 2 0 0 1-2-2V4Zm2-
1a1 1 0 0 0-1 1v.217l7 4.2 7-4.2V4a1 1 0 0 0-1-1H2Zm13 2.383-4.708 2.825L15 11.105V5.383Zm-.034
6.876-5.64-3.471L8 9.583l-1.326-.795-5.64 3.47A1 1 0 0 0 2 13h12a1 1 0 0 0 .966-.741ZM1
11.105l4.708-2.897L1 5.383v5.722Z" />
                      </svg>
                    </span>
                    <input type="email" class="form-control" id="email" name="email" value="" required />
                  </div>
                </div>
                <div class="col-12 col-md-6">
                  <label for="phone" class="form-label">Phone Number</label>
                  <div class="input-group">
                    <span class="input-group-text">
                      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
class="bi bi-telephone" viewBox="0 0 16 16">
                        <path d="M3.654 1.328a.678.678 0 0 0-1.015-.063L1.605 2.3c-.483.484-.661 1.169-.45
1.77a17.568 17.568 0 0 0 4.168 6.608 17.569 17.569 0 0 0 6.608 4.168c.601.211 1.286.033 1.77-
.45l1.034-1.034a.678.678 0 0 0-.063-1.015l-2.307-1.794a.678.678 0 0 0-.58-.122l-2.19.547a1.745 1.745
0 0 1-1.657-.459L5.482 8.062a1.745 1.745 0 0 1-.46-1.657l.548-2.19a.678.678 0 0 0-.122-.58L3.654
1.328zM1.884.511a1.745 1.745 0 0 1 2.612.163L6.29 2.98c.329.423.445.974.315 1.494l-.547
2.19a.678.678 0 0 0 .178.643l2.457 2.457a.678.678 0 0 0 .644.178l2.189-.547a1.745 1.745 0 0 1
1.494.315l2.306 1.794c.829.645.905 1.87.163 2.611l-1.034 1.034c-.74.74-1.846 1.065-2.877.702a18.634
18.634 0 0 1-7.01-4.42 18.634 18.634 0 0 1-4.42-7.009c-.362-1.03-.037-2.137.703-2.877L1.885.511z" />
                      </svg>
                    </span>
                    <input type="tel" class="form-control" id="phone" name="phone" value="" />
                  </div>
                </div>
```

```
              <div class="col-12">
                <label for="subject" class="form-label">Subject <span class="text-
danger">*</span></label>
                <input type="text" class="form-control" id="subject" name="subject" value="" required />
              </div>
              <div class="col-12">
                <label for="message" class="form-label">Message <span class="text-
danger">*</span></label>
                <textarea class="form-control" id="message" name="message" rows="3"
required></textarea>
              </div>
              <div class="col-12">
                <div class="d-grid">
                  <button class="btn btn-primary btn-lg" type="submit">Send Message</button>
                </div>
              </div>
            </div>
          </form>

        </div>
      </div>
    </div>
  </div>
 </div>
</section>
 </>
}

export default Contact
```

8) DASHBOARD PAGE:



CODE :

```
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import './Styles/Dashboard.css';

const Dashboard = () => {
  const navigate = useNavigate();
  const [dashboardData, setDashboardData] = useState({
    properties: [],
    totalValue: 0,
    recentListings: [],
    analytics: {},
    loading: true
  });
```

```
  // Enhanced property data matching your existing structure
  const sampleProperties = [
    {
      id: 1,
      title: "3 BHK Flat for Sale in Salt Lake, Kolkata",
      area: "1200 sqft",
      price: 8500000,
      pricePerSqft: 7083,
      floor: "4 out of 8",
      status: "Ready to Move",
      owner: "Rajesh Kumar",
      imageCount: 22,
      bathroom: 2,
      furnishing: "Fully-Furnished",
      location: "Salt Lake",
      imageUrl: "https://s7ap1.scene7.com/is/image/incredibleindia/howrah-bridge-howrah-west-bengal-city-1-hero?qlt=82&ts=1742154305591",
      listed: "2024-08-05",
      leads: 15,
      views: 340
    },
    {
      id: 2,
      title: "2 BHK Flat for Sale in Howrah Station Area",
      area: "850 sqft",
      price: 4200000,
      pricePerSqft: 4941,
      floor: "2 out of 5",
      status: "Under Construction",
      owner: "Priya Banerjee",
```

```
    imageCount: 18,
    bathroom: 2,
    furnishing: "Semi-Furnished",
    location: "Howrah",
    imageUrl:
"https://st2.indiarailinfo.com/kjfdsuiemjvcya4/0/0/2/7/4226027/0/20190210193319381 0988.jpg",
    listed: "2024-08-08",
    leads: 8,
    views: 156
  },
  {
    id: 3,
    title: "1 BHK Studio Flat in Dum Dum Metro Area",
    area: "600 sqft",
    price: 3200000,
    pricePerSqft: 5333,
    floor: "6 out of 10",
    status: "Ready to Move",
    owner: "Amit Ghosh",
    imageCount: 15,
    bathroom: 1,
    furnishing: "Unfurnished",
    location: "Dumdum",
    imageUrl: "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTTQjr5JkOfSE6vmq2NuBbSIAPz5VOPKOKvPQ&s",
    listed: "2024-08-10",
    leads: 12,
    views: 289
  },
  {
    id: 4,
```

```
    title: "4 BHK Luxury Flat in New Town, Kolkata",
    area: "1800 sqft",
    price: 12000000,
    pricePerSqft: 6667,
    floor: "12 out of 15",
    status: "Ready to Move",
    owner: "Sunita Das",
    imageCount: 28,
    bathroom: 3,
    furnishing: "Fully-Furnished",
    location: "New Town",
    imageUrl: "https://i.ytimg.com/vi/oGSYQusieNM/hq720.jpg?sqp=-oaymwEhCK4FEIIDSFryq4qpAxMIARUAAAAAGAElAADIQj0AgKJD&rs=AOn4CLDtuYXWBz-0Se5F6fK_D4He4VXyHA",
    listed: "2024-07-28",
    leads: 25,
    views: 567
  }
];

const areas = [
  { name: "Kolkata", count: 45, avgPrice: 6500000, growth: "+12%" },
  { name: "Howrah", count: 28, avgPrice: 4200000, growth: "+8%" },
  { name: "Dumdum", count: 22, avgPrice: 3800000, growth: "+15%" },
  { name: "Salt Lake", count: 35, avgPrice: 7200000, growth: "+10%" },
  { name: "New Town", count: 18, avgPrice: 9500000, growth: "+18%" }
];

const recentActivities = [
  { type: "New Listing", message: "3 BHK in Salt Lake added", time: "2 hours ago", status: "success" },
  { type: "Sale Completed", message: "2 BHK in Howrah sold for ₹42L", time: "5 hours ago", status: "sold" },
```

```
    { type: "New Inquiry", message: "Inquiry for 4 BHK in New Town", time: "1 day ago", status:
"inquiry" },
    { type: "Price Updated", message: "1 BHK in Dumdum price reduced", time: "2 days ago", status:
"update" }
  ];


  useEffect(() => {
    // Simulate API call - replace with your actual API endpoint
    const loadDashboardData = async () => {
      try {
        // You can replace this with actual API calls to your backend
        // const response = await axios.get('http://localhost:6005/api/dashboard');


        // Simulated data calculation
        const totalValue = sampleProperties.reduce((sum, prop) => sum + prop.price, 0);
        const totalViews = sampleProperties.reduce((sum, prop) => sum + prop.views, 0);
        const totalLeads = sampleProperties.reduce((sum, prop) => sum + prop.leads, 0);


        setDashboardData({
          properties: sampleProperties,
          totalValue,
          totalViews,
          totalLeads,
          recentListings: sampleProperties.slice(0, 3),
          analytics: {
            totalProperties: sampleProperties.length,
            avgPricePerSqft: Math.round(totalValue / sampleProperties.reduce((sum, prop) => sum +
parseInt(prop.area), 0)),
            conversionRate: "23%"
          },
          loading: false
        });
```

```
    } catch (error) {
      console.error('Error loading dashboard data:', error);
      setDashboardData(prev => ({ ...prev, loading: false }));
    }
  };

  loadDashboardData();
}, []);

const formatPrice = (price) => {
  if (price >= 10000000) {
    return ₹${(price / 10000000).toFixed(1)}Cr;
  } else if (price >= 100000) {
    return ₹${(price / 100000).toFixed(1)}L;
  }
  return ₹${price.toLocaleString()};
};

const formatNumber = (num) => {
  if (num >= 10000000) {
    return ${(num / 10000000).toFixed(1)}Cr;
  } else if (num >= 100000) {
    return ${(num / 100000).toFixed(1)}L;
  }
  return num.toLocaleString();
};

if (dashboardData.loading) {
  return (
    <div className="dashboard-loading">
      <div className="loading-spinner"></div>
```

```
      <p>Loading your dashboard...</p>
    </div>
  );
}

return (
  <div className="dashboard-container">
    {/* Header */}
    <div className="dashboard-header">
      <h1>Real Estate Dashboard</h1>
      <p>Welcome back! Here's what's happening with your properties.</p>
    </div>

    {/* KPI Cards */}
    <div className="kpi-grid">
      <div className="kpi-card total-value">
        <div className="kpi-icon">🏠</div>
        <div className="kpi-content">
          <h3>Total Portfolio Value</h3>
          <p className="kpi-number">{formatPrice(dashboardData.totalValue)}</p>
          <span className="kpi-growth">+12% from last month</span>
        </div>
      </div>

      <div className="kpi-card properties">
        <div className="kpi-icon">📊</div>
        <div className="kpi-content">
          <h3>Active Properties</h3>
          <p className="kpi-number">{dashboardData.analytics.totalProperties}</p>
          <span className="kpi-growth">4 new this week</span>
```

```
    </div>
  </div>

  <div className="kpi-card leads">
    <div className="kpi-icon">👥</div>
    <div className="kpi-content">
      <h3>Total Leads</h3>
      <p className="kpi-number">{dashboardData.totalLeads}</p>
      <span className="kpi-growth">Conversion: {dashboardData.analytics.conversionRate}</span>
    </div>
  </div>

  <div className="kpi-card views">
    <div className="kpi-icon">👁</div>
    <div className="kpi-content">
      <h3>Total Views</h3>
      <p className="kpi-number">{formatNumber(dashboardData.totalViews)}</p>
      <span className="kpi-growth">+25% this month</span>
    </div>
  </div>
</div>

{/* Main Content Grid */}
<div className="dashboard-grid">
  {/* Recent Properties */}
  <div className="dashboard-section recent-properties">
    <h2>Recent Listings</h2>
    <div className="property-list">
      {dashboardData.recentListings.map((property) => (
        <div key={property.id} className="mini-property-card">
```

```
        <img
          src={property.imageUrl}
          alt={property.title}
          className="mini-property-image"
        />
        <div className="mini-property-details">
          <h4>{property.title}</h4>
          <p className="property-area">{property.area} • {property.location}</p>
          <p className="property-price">{formatPrice(property.price)}</p>
          <div className="property-stats">
            <span>👁 {property.views}</span>
            <span>📞 {property.leads}</span>
          </div>
        </div>
      </div>
    ))}
  </div>
</div>

{/* Area Performance */}
<div className="dashboard-section area-performance">
  <h2>Area Performance</h2>
  <div className="area-list">
    {areas.map((area, index) => (
      <div key={index} className="area-item">
        <div className="area-info">
          <h4>{area.name}</h4>
          <p>{area.count} properties</p>
          <p className="area-price">Avg: {formatPrice(area.avgPrice)}</p>
        </div>
```

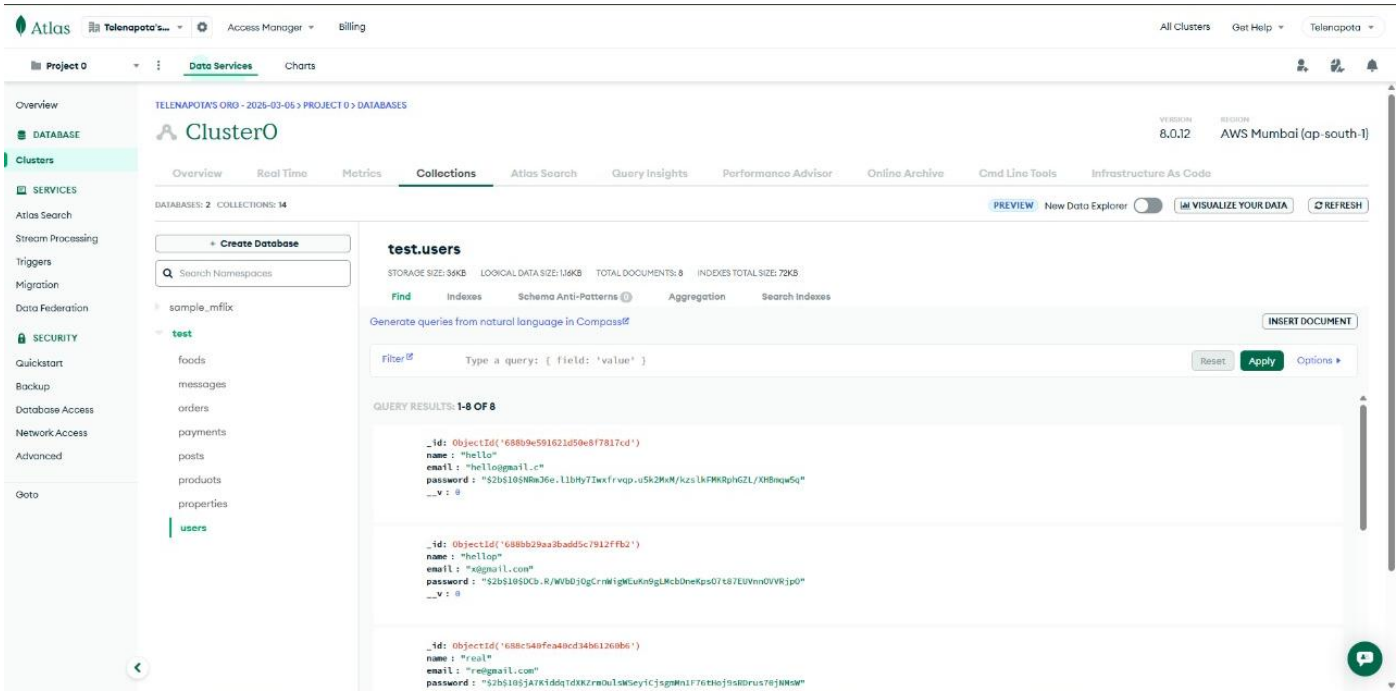
```
        <div className="area-growth">
          <span className={growth ${area.growth.includes('+') ? 'positive' : 'negative'}}>
            {area.growth}
          </span>
        </div>
      </div>
    ))}
  </div>
</div>

{/* Recent Activities */}
<div className="dashboard-section recent-activities">
  <h2>Recent Activities</h2>
  <div className="activity-list">
    {recentActivities.map((activity, index) => (
      <div key={index} className={activity-item ${activity.status}}>
        <div className="activity-icon">
          {activity.status === 'success' && '☑'}
          {activity.status === 'sold' && '💰'}
          {activity.status === 'inquiry' && '📞'}
          {activity.status === 'update' && '📝'}
        </div>
        <div className="activity-content">
          <h4>{activity.type}</h4>
          <p>{activity.message}</p>
          <span className="activity-time">{activity.time}</span>
        </div>
      </div>
    ))}
  </div>
```

```
  </div>

  {/* Quick Actions */}
  <div className="dashboard-section quick-actions">
    <h2>Quick Actions</h2>
    <div className="action-buttons">
      <button
        className="action-btn primary"
        onClick={() => navigate('/sell')}
      >
        <span>➕</span>
        Add New Property
      </button>
      <button
        className="action-btn secondary"
        onClick={() => navigate('/view')}
      >
        <span>👁</span>
        View All Properties
      </button>
      <button
        className="action-btn tertiary"
        onClick={() => window.location.reload()}
      >
        <span>🔄</span>
        Refresh Data
      </button>
    </div>
  </div>
</div>
```

```
      {/* Market Insights */}
      <div className="market-insights">
        <h2>Market Insights - Kolkata Real Estate</h2>
        <div className="insights-grid">
          <div className="insight-card">
            <h3>🏆 Top Performing Area</h3>
            <p><strong>New Town</strong> with +18% growth</p>
          </div>
          <div className="insight-card">
            <h3>📈 Average Price/SqFt</h3>
            <p><strong>₹{dashboardData.analytics.avgPricePerSqft}</strong> across all areas</p>
          </div>
          <div className="insight-card">
            <h3>🎯 Best ROI</h3>
            <p><strong>Salt Lake</strong> properties showing strong demand</p>
          </div>
          <div className="insight-card">
            <h3>💡 Market Trend</h3>
            <p><strong>2-3 BHK</strong> flats in high demand</p>
          </div>
        </div>
      </div>
    </div>
  );
};

export default Dashboard;
```

❖ BACKEND:-

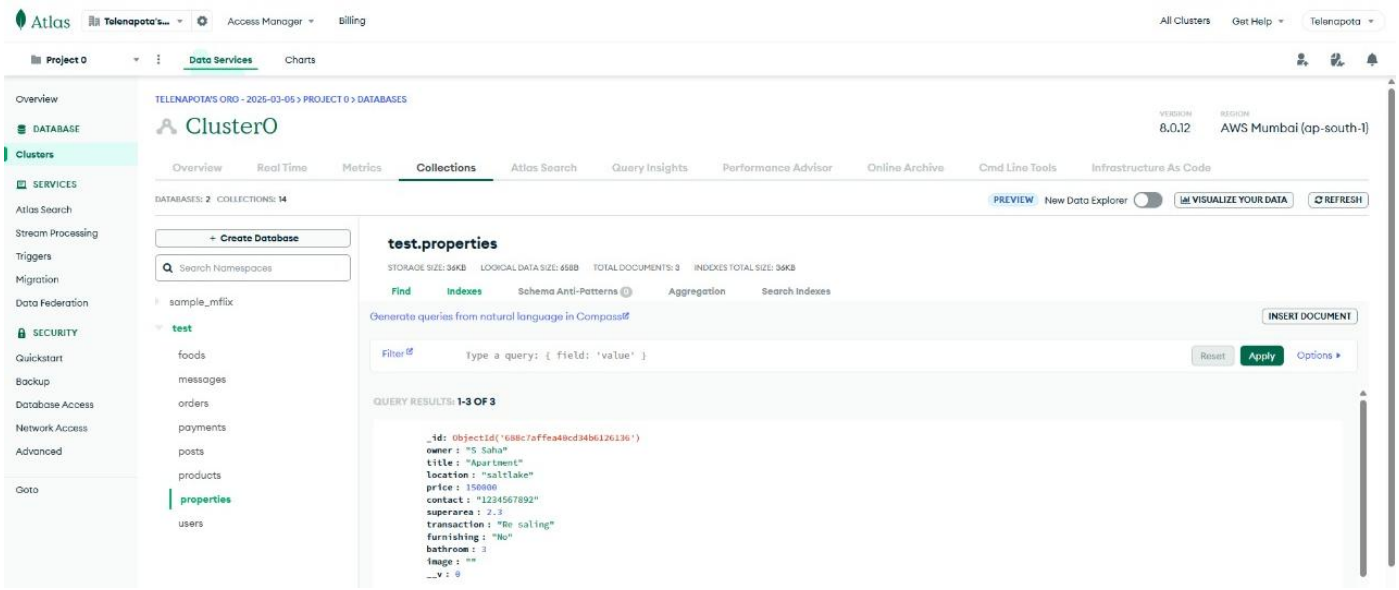
1) USER AND ADMIN DATA:



- **Database - db.js:**

```
const mongoose = require('mongoose');

const connectDB = async () => {
   try {
      await mongoose.connect(process.env.MONGO_URI, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      });
      console.log('MongoDB connected successfully');
   } catch (error) {
      console.error('MongoDB connection failed:', error.message);
      process.exit(1); // Exit process with failure
   }
   }
   module.exports = connectDB;
```

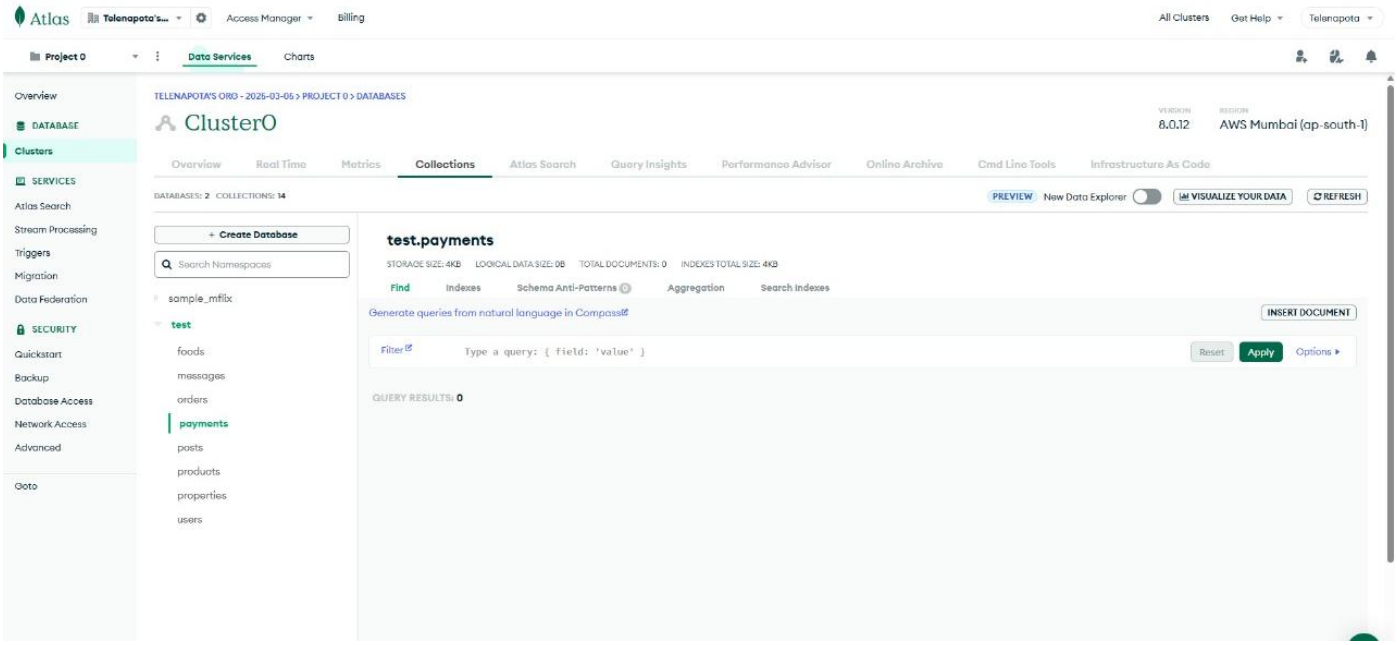
2) PROPERTY DATA:



- **model – Property.js :-**

```
const mongoose = require('mongoose');

const propertySchema = new mongoose.Schema({
    owner: {type: String, required: true },
    title:{type:String, required:true},
    location:{type:String, required:true},
    price:{type:Number, required:true},
    contact:{type:String, required:true},
    superarea:{type:Number, required:true},
    transaction:{type:String},
    furnishing:{type:String},
    bathroom:{type:Number},
    image:{type:String},
});

module.exports = mongoose.model('Property', propertySchema);
```

3) PAYMENT DATA



- **model – Payment.js :-**

```
const mongoose = require('mongoose');

const paymentSchema = new mongoose.Schema({
   orderId: String,
   paymentId: String,
   amount: Number,
   currency: String,
   status: String,

   createdAt: {
      type: Date,
      default: Date.now
   }
});

module.exports = mongoose.model('Payment', paymentSchema);
```

6. CONCLUSION

The Real Estate Web Application successfully addresses the limitations of traditional property transactions by providing a centralized, efficient, and user-friendly digital platform for buyers, sellers, agents, and tenants. Through features like advanced property search, secure user authentication, real-time updates, and direct communication channels, the system enhances transparency, reduces transaction time, and broadens market reach.

The application's responsive design ensures accessibility across devices, while its secure backend architecture safeguards user data and supports smooth operation. By integrating modern web technologies and scalable architecture, the platform is capable of adapting to future requirements and market growth. Ultimately, this project not only improves the overall efficiency of real estate dealings but also delivers a reliable, cost-effective, and convenient solution that benefits all stakeholders in the property market.

7. FUTURE SCOPE & FURTHER ENHANCEMENTS

❖ Future scope:-

a. **Virtual Property Tours** – Integrating 360° photography and VR/AR technology to allow users to explore properties remotely.
b. **AI-Powered Recommendations** – Using machine learning to suggest properties based on user preferences, past searches, and behavior patterns.
c. **Integrated Payment Systems** – Enabling secure online transactions, booking fees, and installment tracking directly within the platform.
d. **Chatbots and Virtual Assistants** – Providing instant customer support and automated responses to frequently asked questions.
e. **Blockchain for Transactions** – Ensuring secure, transparent, and tamper-proof property transaction records.
f. **Multilingual Support** – Expanding reach by supporting multiple languages for diverse user demographics.
g. **Mobile App Version** – Launching a dedicated Android and iOS application for on-the-go accessibility.
h. **Advanced Analytics Dashboard** – Giving sellers and agents deeper insights into market trends, property views, and customer engagement.

❖ Further enhancement:-

1. **Geo-Location Based Search** – Integrating location-based services to automatically suggest nearby properties based on the user's current location.
2. **Property Valuation Tool** – Adding an AI-driven property price estimation feature using historical sales data and market trends.
3. **Smart Filters** – Allowing users to filter results by commute time, neighborhood facilities, crime rates, and school ratings.
4. **Document Management System** – Securely storing and sharing legal documents (agreements, property papers) between buyers, sellers, and agents.
5. **Push Notifications** – Real-time alerts for new property listings, price drops, or messages.
6. **Seller Verification System** – Implementing KYC (Know Your Customer) verification to ensure authenticity of property listings.
7. **Green & Sustainable Property Tagging** – Highlighting eco-friendly properties with energy efficiency ratings.
8. **Crowdsourced Reviews** – Letting users share reviews and ratings of neighborhoods, builders, and agents.
9. **Offline Mode** – Allowing users to browse previously loaded property data without an internet connection.

8. BIBLIOGRAPHY

1) www.w3schools.com

2) www.youtube.com

3) www.pexels.com

4) 4)www.codepen.io

5) www.google.com

6) www.googlefont.com

7) www.react.our

8) www.codingworld.com