

**MINOR PROJECT - III**

**Submitted in Partial Fulfilment of the Requirements for the Award of the Degree of**

**MASTER OF COMPUTER APPLICATIONS**

**(M.C.A.)**

**BY**

**SHUBHAM JOSHI**

**(03811604422)**

**MCA –2nd SEMESTER**



**BHARATI VIDYAPEETH’S INSTITUTE OF COMPUTER APPLICATIONS AND MANAGEMENT (BVICAM)**

**(GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY)**

**NEW DELHI**

**December, 20****23**

**Table of Contents**

1. Chapter 1 – INTRODUCTION…………………………………………………...………….1
   1. **PURPOSE……………………………………………………………...…………………….1**
   2. **SCOPE………………………………………………………………………..…….………..2**
   3. **OBJECTIVES………………………………………………………………………...……..3**
2. Chapter 2- SYSTEM ANALYSIS STUDY……………………………………………………..…..5
   1. **PROJECT OVERVIEW…………………………………………………………………………..5**
   2. **FEASIBILITY STUDY……………………………………………………………………………6**
      1. **Technical Feasibility…………………………………………………………………………..6**
      2. **Economical Feasibility………………………………………………….……………………..7**

**2.2.3 Operational Feasibility…………………………………………………………..………………..7**

1. Chapter- 3 SYSTEM DESIGN…………………………………………………………….8
   1. **INTRODUCTION………………………………………………………………………….8**
   2. **DATA FLOW CHARTS…………………………………………………………………9**
   3. **USE CASE DIAGRAM…………………………………………………………………15**
   4. **DATABASE DESIGN:………………………………………………….....…………………16**
      1. **ER Diagram………………………………………………………………...………………..16**
      2. **Database Schema…………………………………………………………………………17**
2. Chapter 4 - SOFTWARE CODING, TESTING & IMPLEMENTATION……..………18
   1. **SYSTEM CODING ENVIRONMENT AND STANDARDS FOLLOWED………19**
   2. **CODES AND LAYOUTS………………………………………………………..…….21**
3. OUTPUT FORMS & REPORTS……………………………………….…………………..22
   1. **INPUT / OUTPUT FORMS WITH SCREENSHOTS………………………………………..23-61**
4. CONCLUSION & FUTURE ENHANCEMENT(S)………………………………….…….62
   1. **Conclusion…………………………………………………………………………..……..64**
   2. **Future Scope……………………………………………………………………...………..65**

**Bibliography……………………………………………………………………………...……………66**

**1.INTRODUCTION**

**1.1 PURPOSE**

Welcome to our small real estate app, built using Mern Stack . This minor project aims to build a robust web application using the MERN stack with advanced authentication and search Functionality.

The platform serves as a centralized hub for users to showcase their properties available for rent and sale, providing a seamless experience for both property owners and potential renters or buyers.

**1.2 SCOPE**

**In the traditional real estate landscape, the predominant model relies on intermediaries or property dealers who serve as middlemen between property owners and potential tenants or buyers. These intermediaries play a central role in the property transaction process, controlling access to a limited selection of properties and charging a commission for their services. This conventional approach has both advantages and limitations, addressing challenges for both property seekers and owners.**

**Advantages of the Traditional Model:**

1. **Convenience for Property Seekers:**
   * **Property seekers often find convenience in dealing with intermediaries who curate a selection of properties. This saves time and effort that would otherwise be spent on individually searching for suitable options.**
2. **Expertise and Guidance:**
   * **Property dealers bring expertise and local market knowledge to the table. Their understanding of the real estate landscape helps guide property seekers in making informed decisions about their investments or rental choices.**
3. **Negotiation Assistance:**
   * **Intermediaries are skilled negotiators who can facilitate smoother transactions. They act as intermediaries during price negotiations, helping to reach mutually agreeable terms between property owners and seekers.**
4. **Reduced Burden for Property Owners:**
   * **Property owners benefit from the services of intermediaries by offloading the responsibility of managing property inquiries, screenings, and negotiations. This allows property owners to focus on other aspects of their investments or personal priorities.**

**Limitations of the Traditional Model:**

1. **Limited Property Selection:**
   * **Property seekers are often restricted to the inventory held by the intermediary. This limitation may result in missing out on potentially suitable properties that are not part of the intermediary's portfolio.**
2. **Commission Costs:**
   * **Property dealers typically charge a commission for their services, adding an additional cost to both property seekers and owners. This can be a significant financial consideration, especially for those on a tight budget.**
3. **Dependency on Intermediaries:**
   * **Both property seekers and owners become dependent on intermediaries for the property transaction process. Overreliance on these middlemen may lead to challenges if the intermediary's services do not meet expectations.**
4. **Lack of Direct Communication:**
   * **In the traditional model, direct communication between property owners and seekers is often limited. This can result in delays, miscommunications, or a lack of clarity during the negotiation and decision-making process.**
5. **Technology and Innovation Lag:**
   * **The traditional model may lag in adopting modern technologies and innovative solutions that could streamline the real estate transaction process and enhance user experiences.**

**In conclusion, while the traditional real estate model offers convenience and expertise through intermediaries, it also comes with limitations. The evolving landscape of real estate is witnessing the emergence of alternative models, leveraging technology to address these limitations and provide more transparent, efficient, and cost-effective solutions for property seekers and owners**

**1.3 OBJECTIVES**

* **Responsive designs**

A responsive design where an optimal test experience is available for a wide range of devices. A responsive app scales with the size of the screen without sacrificing the text readability or usability of the user interface.

* **Login/Signup with Firebase Authentication:**

Every users and assistant will be authorized through Email id and password (set by the user) through Firebase database.

* **News Feed**

It allows users to constantly benefit from the surge of activity in their network neighborhood. . With News Feed, this app is no longer about staying connected with a friend or even with a group. It’s about staying connected with one’s entire network.

* **Manage / Edit your profile page**

Editing profile page option include:

1. Changing your academic field
2. You can view how many people you are following as well as followers you have
3. Changing your email id.
4. View, add and delete your name and about yourself section.

* **Notifications**

You regularly get an update about what’s new happening in your chosen community. If you get a request it will be shown in Notification section as well as many more.

* **Create new Posts / Edit your Post**

You can ask any query related to your subject in your community and you can also share any brand new information with your community. Your post will have the features of like and comments and share which will make it more interactive for the users.

**2.SYSTEM STUDY**

**2.1 PROJECT OVERVIEW**

In today’s world, Smart phones have changed our lives and have become an indispensable part of our lives because of its specialty to simplify our routine work and thereby saving our time. A Smartphone with an Android OS offers excellent functionality to the users offering a distinct experience. Android is a Linux based operating system and it was bought by Google in 2007.There are tons of application available and one of the prime reason for this vast number is android being an open source. On the other hand, android based device like mobile, tab are very user friendly.

A survey done by “Statista” which indicates that though other operating system mobile users exist but the majority users use Android operating system.

**2.2 FEASIBILITY STUDY**

**2.2.1 Technical Feasibility**

Today, very little is technically impossible. Consequently, technical feasibility looks at what is practical and reasonable. There are not much technical requirements for this project. Both software and hardware requirements could be met very easily and are basically available in most of the systems. The application could also run on android phones very easily. So it will be very easy simple for both the user and the administrators of the test to use this and hence any profound technical knowledge is not required to use the application. Anyone can use it very easily. It’s requirements are as follows:

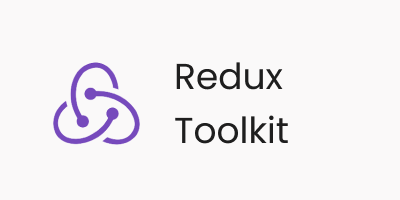
Tools / Platform, Hardware and Software Requirements

**Tools / Platform, Hardware and Software Requirements**

These requirements are separated based on development of the app or running the app on a device.

* **Software requirements for development**
* **Platform:**  VS Code IDE



* **Technologies used:** Mern Stack, Redux Toolkit
* 

. 

* **Debugger:** Vs Code
* **Hardware Requirements for development**
* **OS**: Windows 64-bit Microsoft Windows 8/10
* **Processor:** Intel Pentium IV or i3/i5/i7.
* **RAM:** 8 GB or more
* **Space on disk:** minimum 20 GB or more
* **Hardware/software requirements on User-end:-**
* **Processor:** Intel i-core Processor 1.2 GHZ or faster processor
* **Storage:** Between 850 MB and 1.2 GB
* **RAM:** minimum of 512 MB | 2 GB recommended
* **Device:** Phone or tablet running Android 4.4 or higher

**2.2.2 Economical Feasibility**

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly knows as cost/benefit analysis the procedure is to determine the benefits and saving that are expected from Sociofy application and compare them with costs, decisions is made to design and implement the system.

The organization is satisfied by economic feasibility because, if the organization implements this application, it need not require any additional hardware resources thus it will be saving lot of time and cost thereby making it economically feasible.

**2.2.3 Operational Feasibility**

It is mainly related to human organizational and political aspects. The points to be considered are:

Generally project will be affected because of operational infeasibility but such considerations critically affect the nature and scope of the eventual recommendations.

The application (Sociofy) offers greater levels of user friendliness combined with greater processing speed. Therefore, the cost of maintenance can be reduced. Since, processing speed is very high and the work is reduced in the maintenance point of view management convince that the application is operationally feasible. The application can be used in wide variety of Android devices which make it preferable choice and thus it is operationally feasible

**3.SYSTEM DESIGN**

**3.1 INTRODUCTION**

The final deliverable from system analysis is a document containing an unambiguous statement of the client’s requirements from the new system and what the development project will have to deliver in order to be considered a success.

The functional specification is the starting point for designing, which depends to a large extent on the accuracy and thoroughness with which the analysis has been carried out. Understanding of the business, appreciation of the client’s problems and documentation of requirements provide the foundation on which the designing is based.

A key factor in this approach is the use of the structured techniques. Analysis ends with a description of what the new system must do to fulfill the requirements of the organization, while design specifies how this will be done by selecting one of the many ways of doing it.

The structured techniques used during the analysis which provide this logical view are:

• Data flow diagrams – representing the process, which manipulate the data as it passes through the system.

• Entity model – showing the relationship within the data items held within the system.

• A data dictionary – providing an overall consistent definition of the data used during the system development. This definition includes the content of the data stores, data flows and the process shown on the data flow diagrams, and the entities that make up the entity model.

**3.2 DATA FLOW CHARTS**

The data flowchart is a means of visually presenting the flow of data through an information processing system, the operations performed within the system and the sequence in which they are performed. In this lesson, we shall concern ourselves with the program flowchart, which describes what operations (and in what sequence) are required to solve a given problem. The program flowchart can be likened to the blueprint of a building. As we know a designer draws a blueprint before starting construction on a building. Similarly, a programmer prefers to draw a flowchart prior to writing a computer program. As in the case of the drawing of blueprint, the flowchart is drawn according to defined rules and standard flowchart symbols prescribed by the American National Standard Institute, Inc.

Symbols used in database and ER diagrams

Start or end of program

Computational steps or processing functions of a program.

Input or output

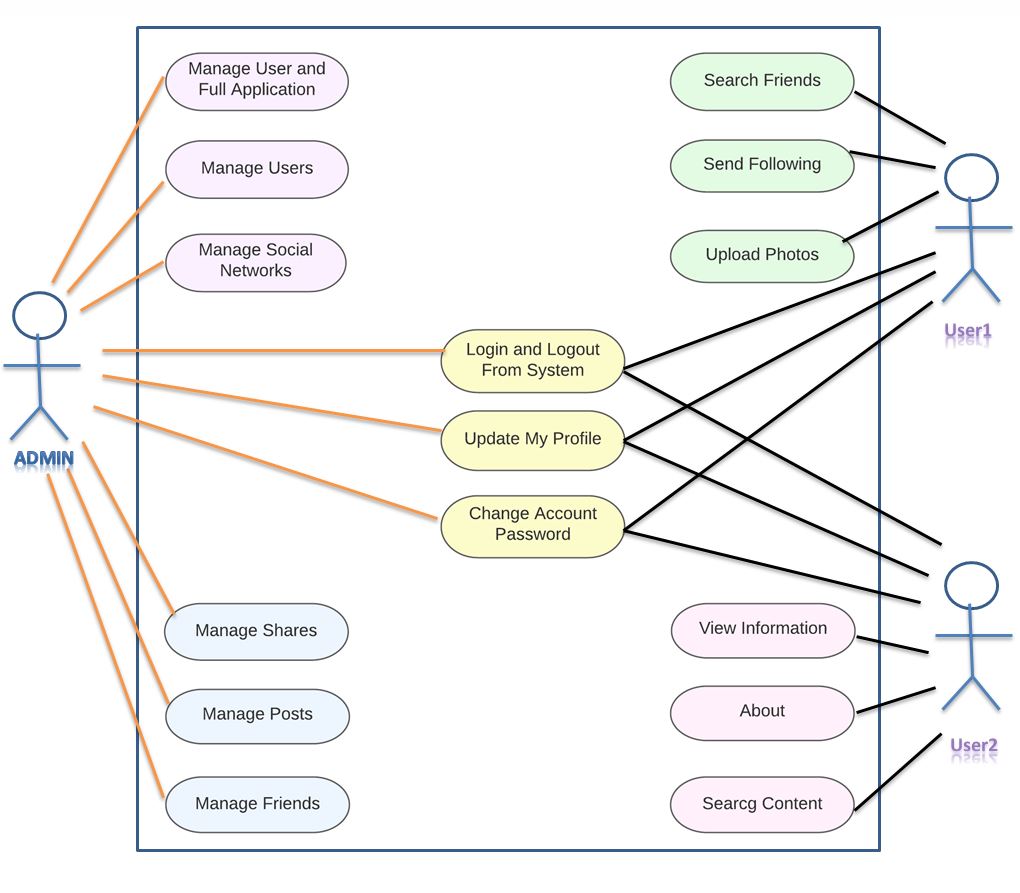
Decision making or branching.

Flow lines

**3.3 USE CASE DIAGRAM:**

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system.

* Scenarios in which your system or application interacts with people, organizations, or external systems.
* Goals that application helps those entities (known as actors) achieve
* The scope of system

****

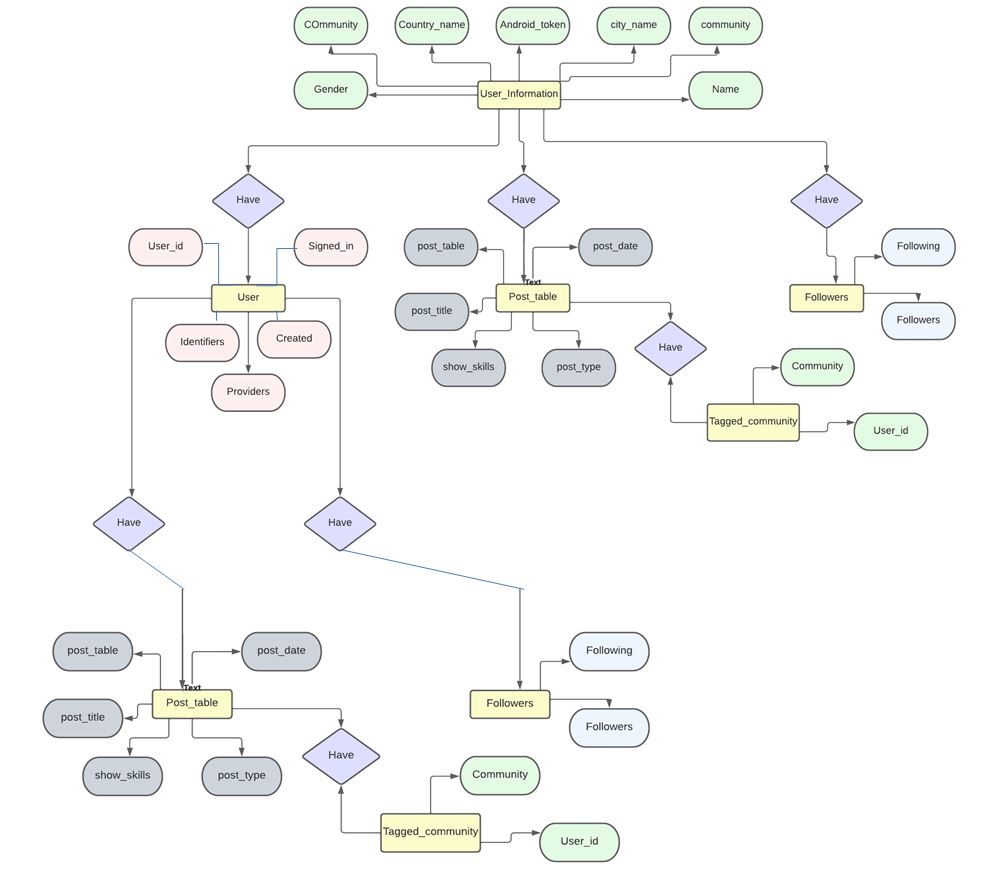
**3.4 DATABASE DESIGN**

**3.4.1 ER Diagram-**

Symbols used in making ER Diagram:

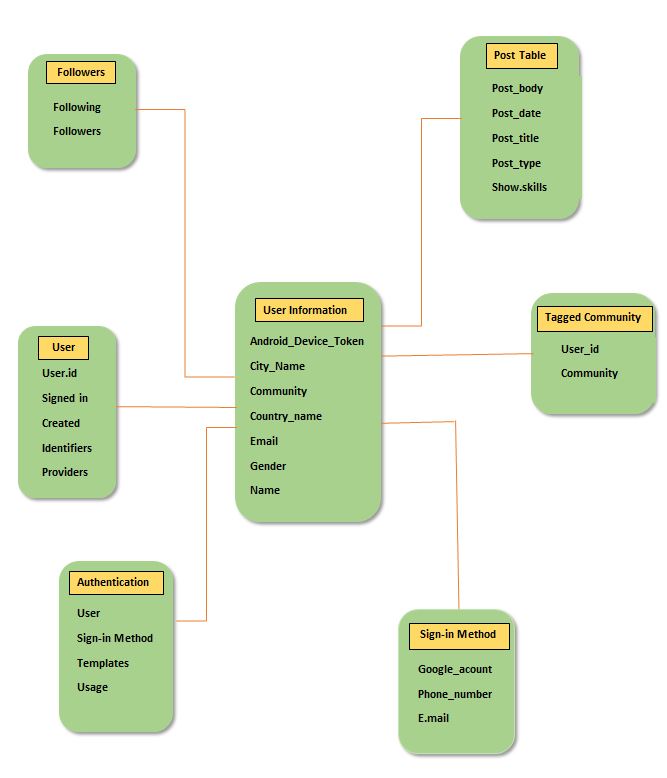
**🡪 Entity**

* **Relationship**
* **Attribute**
* **Weak Entity**
* **Weak Entity Relationship**
* **Multivalued Attribute**
* **Key Attribute**
  + - **Composite Attribute**

****

**3.4.2. Database Schema**

Database schema is a visual representation of [database](https://examplanning.com/what-is-database-its-types-and-examples/). The structure is described by formal language and is supported by database management system (DBMS). It represents logical view of the database.A schema is a representation of design or idea in the shape of model. A database may contain one or multiple schemas.A database schema consists of objects like tables, views and stored procedure etc. In other words, we can say that database schema is a container of objects.



**4. SOFTWARE CODING, TESTING & IMPLEMENTATION**

* 1. **SYSTEM CODING ENVIRONMENT AND STANDARDS FOLLOWED**

### 1. **MongoDB: The NoSQL Powerhouse**

**Overview:** MongoDB is a robust NoSQL database designed for the modern era of application development. As a document-oriented database, MongoDB stores data in flexible, JSON-like BSON documents. This flexibility allows developers to work with dynamic schemas, making it particularly suitable for applications with evolving and diverse data structures.

**Key Concepts:**

* **Document-Oriented Storage:** MongoDB stores data in flexible, JSON-like BSON documents, allowing for varied and hierarchical data structures.
* **Dynamic Schema:** Unlike traditional relational databases, MongoDB doesn't require a predefined schema. This flexibility simplifies development and accommodates changes in data structure over time.
* **Scalability:** MongoDB is horizontally scalable, enabling seamless distribution of data across multiple servers for increased performance.

**Use Cases:**

* MongoDB is ideal for applications with rapidly changing data requirements, such as content management systems, e-commerce platforms, and real-time analytics.

### 2. **Express.js: Crafting Robust Backends**

**Overview:** Express.js is a minimalist web application framework for Node.js, designed to simplify the process of building robust and scalable server-side applications. It provides a set of features for web and mobile application development and serves as the backend component in the MERN stack.

**Key Concepts:**

* **Routing:** Express enables the definition of routes to handle HTTP requests. This allows for the creation of a RESTful API, facilitating communication between the frontend and backend.
* **Middleware:** Express uses middleware functions to perform tasks such as authentication, logging, and error handling. Middleware enhances the functionality and behavior of the application.

**Use Cases:**

* Express.js is commonly used to build APIs and server-side applications for web and mobile platforms. It is well-suited for creating scalable and maintainable server-side code.

### 3. **React: Transformative Front-End Development**

**Overview:** React is a declarative and efficient JavaScript library for building user interfaces. Developed by Facebook, React allows developers to create reusable UI components and efficiently manage the state of an application. It facilitates the development of interactive and dynamic front-end experiences.

**Key Concepts:**

* **Component-Based Architecture:** React follows a component-based architecture, where UI elements are encapsulated into reusable and self-contained components. This promotes code reusability and maintainability.
* **Virtual DOM:** React utilizes a virtual DOM to optimize the rendering process. Changes to the UI are first made to the virtual DOM, and only the differences are applied to the actual DOM, minimizing performance overhead.
* **One-Way Data Binding:** React adopts a unidirectional data flow, making it easier to manage the state of components and ensuring predictable updates to the UI.

**Use Cases:**

* React is widely used for building interactive and dynamic user interfaces in single-page applications (SPAs) and for creating reusable UI components in larger web applications.

### 4. **Node.js: Empowering Server-Side Development**

**Overview:** Node.js is a JavaScript runtime built on the V8 JavaScript engine. It allows developers to run JavaScript on the server side, enabling the development of scalable and high-performance server-side applications. In the context of the MERN stack, Node.js serves as the runtime environment for running the backend server.

**Key Concepts:**

* **Event-Driven Architecture:** Node.js is designed with an event-driven, non-blocking I/O model. This makes it well-suited for handling concurrent connections and building scalable applications.
* **NPM (Node Package Manager):** NPM is the package manager for Node.js, providing a vast ecosystem of open-source packages and modules that developers can leverage to enhance their applications.

**Use Cases:**

* Node.js is commonly used for building server-side applications, APIs, and real-time applications. Its event-driven nature makes it particularly suitable for handling a large number of simultaneous connections.

### In Summary:

The MERN stack combines MongoDB, Express.js, React, and Node.js to create a powerful, end-to-end solution for building modern web applications. MongoDB provides a flexible and scalable NoSQL database, Express.js handles server-side logic, React facilitates the creation of dynamic user interfaces, and Node.js empowers server-side development. Together, these technologies form a cohesive and efficient stack, enabling developers to build feature-rich and scalable applications. The MERN stack is well-regarded for its flexibility, ease of use, and the ability to create seamless, real-time, and interactive user experiences

* 1. **CODES AND LAYOUTS**

**LOGIN PAGE**

import { useState } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { useDispatch, useSelector } from 'react-redux';

import {

  signInStart,

  signInSuccess,

  signInFailure,

} from '../redux/user/userSlice';

import OAuth from '../components/OAuth';

export default function SignIn() {

  const [formData, setFormData] = useState({});

  const { loading, error } = useSelector((state) => state.user);

  const navigate = useNavigate();

  const dispatch = useDispatch();

  const handleChange = (e) => {

    setFormData({

      ...formData,

      [e.target.id]: e.target.value,

    });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      dispatch(signInStart());

      const res = await fetch('/api/auth/signin', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify(formData),

      });

      const data = await res.json();

      console.log(data);

      if (data.success === false) {

        dispatch(signInFailure(data.message));

        return;

      }

      dispatch(signInSuccess(data));

      navigate('/');

    } catch (error) {

      dispatch(signInFailure(error.message));

    }

  };

  return (

    <div className='p-3 max-w-lg mx-auto'>

      <h1 className='text-3xl text-center font-semibold my-7'>Sign In</h1>

      <form onSubmit={handleSubmit} className='flex flex-col gap-4'>

        <input

          type='email'

          placeholder='email'

          className='border p-3 rounded-lg'

          id='email'

          onChange={handleChange}

        />

        <input

          type='password'

          placeholder='password'

          className='border p-3 rounded-lg'

          id='password'

          onChange={handleChange}

        />

        <button

          disabled={loading}

          className='bg-slate-700 text-white p-3 rounded-lg uppercase hover:opacity-95 disabled:opacity-80'

        >

          {loading ? 'Loading...' : 'Sign In'}

        </button>

        <OAuth/>

      </form>

      <div className='flex gap-2 mt-5'>

        <p>Dont have an account?</p>

        <Link to={'/sign-up'}>

          <span className='text-blue-700'>Sign up</span>

        </Link>

      </div>

      {error && <p className='text-red-500 mt-5'>{error}</p>}

    </div>

  );

}

**Sign Up Page**

import { useState } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import OAuth from '../components/OAuth';

export default function SignUp() {

  const [formData, setFormData] = useState({});

  const [error, setError] = useState(null);

  const [loading, setLoading] = useState(false);

  const navigate = useNavigate();

  const handleChange = (e) => {

    setFormData({

      ...formData,

      [e.target.id]: e.target.value,

    });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      setLoading(true);

      const res = await fetch('/api/auth/signup', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify(formData),

      });

      const data = await res.json();

      console.log(data);

      if (data.success === false) {

        setLoading(false);

        setError(data.message);

        return;

      }

      setLoading(false);

      setError(null);

      navigate('/sign-in');

    } catch (error) {

      setLoading(false);

      setError(error.message);

    }

  };

  return (

    <div className='p-3 max-w-lg mx-auto'>

      <h1 className='text-3xl text-center font-semibold my-7'>Sign Up</h1>

      <form onSubmit={handleSubmit} className='flex flex-col gap-4'>

        <input

          type='text'

          placeholder='username'

          className='border p-3 rounded-lg'

          id='username'

          onChange={handleChange}

        />

        <input

          type='email'

          placeholder='email'

          className='border p-3 rounded-lg'

          id='email'

          onChange={handleChange}

        />

        <input

          type='password'

          placeholder='password'

          className='border p-3 rounded-lg'

          id='password'

          onChange={handleChange}

        />

        <button

          disabled={loading}

          className='bg-slate-700 text-white p-3 rounded-lg uppercase hover:opacity-95 disabled:opacity-80'

        >

          {loading ? 'Loading...' : 'Sign Up'}

        </button>

        <OAuth/>

      </form>

      <div className='flex gap-2 mt-5'>

        <p>Have an account?</p>

        <Link to={'/sign-in'}>

          <span className='text-blue-700'>Sign in</span>

        </Link>

      </div>

      {error && <p className='text-red-500 mt-5'>{error}</p>}

    </div>

  );

}

**Packages Installed**

{

  "name": "client",

  "private": true,

  "version": "0.0.0",

  "type": "module",

  "scripts": {

    "dev": "vite",

    "build": "vite build",

    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",

    "preview": "vite preview"

  },

  "dependencies": {

    "@reduxjs/toolkit": "^1.9.7",

    "firebase": "^10.7.0",

    "react": "^18.2.0",

    "react-dom": "^18.2.0",

    "react-icons": "^4.11.0",

    "react-redux": "^8.1.3",

    "react-router-dom": "^6.17.0",

    "redux-persist": "^6.0.0",

    "swiper": "^11.0.5"

  },

  "devDependencies": {

    "@tailwindcss/line-clamp": "^0.4.4",

    "@types/react": "^18.2.15",

    "@types/react-dom": "^18.2.7",

    "@vitejs/plugin-react-swc": "^3.3.2",

    "autoprefixer": "^10.4.16",

    "eslint": "^8.45.0",

    "eslint-plugin-react": "^7.32.2",

    "eslint-plugin-react-hooks": "^4.6.0",

    "eslint-plugin-react-refresh": "^0.4.3",

    "postcss": "^8.4.31",

    "tailwindcss": "^3.3.3",

    "vite": "^4.4.5"

  }

}

**(in admin side)**

{

  "name": "real\_estate",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "type": "module",

  "scripts": {

    "dev": "nodemon api/index.js",

    "start": "node api/index.js",

    "build": "npm install && npm install --prefix client && npm run build --prefix client"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "dependencies": {

    "bcryptjs": "^2.4.3",

    "cookie-parser": "^1.4.6",

    "dotenv": "^16.3.1",

    "express": "^4.18.2",

    "jsonwebtoken": "^9.0.2",

    "mongoose": "^7.6.3",

    "nodemon": "^3.0.1"

  }

}

**Create Listing**

import { useState } from 'react';

import {

  getDownloadURL,

  getStorage,

  ref,

  uploadBytesResumable,

} from 'firebase/storage';

import { app } from '../firebase';

import { useSelector } from 'react-redux';

import { useNavigate } from 'react-router-dom';

export default function CreateListing() {

  const { currentUser } = useSelector((state) => state.user);

  const navigate = useNavigate();

  const [files, setFiles] = useState([]);

  const [formData, setFormData] = useState({

    imageUrls: [],

    name: '',

    description: '',

    address: '',

    type: 'rent',

    bedrooms: 0,

    bathrooms: 1,

    regularPrice: 50,

    discountPrice: 0,

    offer: false,

    parking: false,

    furnished: false,

  });

  const [imageUploadError, setImageUploadError] = useState(false);

  const [uploading, setUploading] = useState(false);

  const [error, setError] = useState(false);

  const [loading, setLoading] = useState(false);

  console.log(formData);

  const handleImageSubmit = (e) => {

    if (files.length > 0 && files.length + formData.imageUrls.length < 7) {

      setUploading(true);

      setImageUploadError(false);

      const promises = [];

      for (let i = 0; i < files.length; i++) {

        promises.push(storeImage(files[i]));

      }

      Promise.all(promises)

        .then((urls) => {

          setFormData({

            ...formData,

            imageUrls: formData.imageUrls.concat(urls),

          });

          setImageUploadError(false);

          setUploading(false);

        })

        .catch((err) => {

          setImageUploadError('Image upload failed (2 mb max per image)');

          setUploading(false);

        });

    } else {

      setImageUploadError('You can only upload 6 images per listing');

      setUploading(false);

    }

  };

  const storeImage = async (file) => {

    return new Promise((resolve, reject) => {

      const storage = getStorage(app);

      const fileName = new Date().getTime() + file.name;

      const storageRef = ref(storage, fileName);

      const uploadTask = uploadBytesResumable(storageRef, file);

      uploadTask.on(

        'state\_changed',

        (snapshot) => {

          const progress =

            (snapshot.bytesTransferred / snapshot.totalBytes) \* 100;

          console.log(`Upload is ${progress}% done`);

        },

        (error) => {

          reject(error);

        },

        () => {

          getDownloadURL(uploadTask.snapshot.ref).then((downloadURL) => {

            resolve(downloadURL);

          });

        }

      );

    });

  };

  const handleRemoveImage = (index) => {

    setFormData({

      ...formData,

      imageUrls: formData.imageUrls.filter((\_, i) => i !== index),

    });

  };

  const handleChange = (e) => {

    if (e.target.id === 'sale' || e.target.id === 'rent') {

      setFormData({

        ...formData,

        type: e.target.id,

      });

    }

    if (

      e.target.id === 'parking' ||

      e.target.id === 'furnished' ||

      e.target.id === 'offer'

    ) {

      setFormData({

        ...formData,

        [e.target.id]: e.target.checked,

      });

    }

    if (

      e.target.type === 'number' ||

      e.target.type === 'text' ||

      e.target.type === 'textarea'

    ) {

      setFormData({

        ...formData,

        [e.target.id]: e.target.value,

      });

    }

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      if (formData.imageUrls.length < 1)

        return setError('You must upload at least one image');

      if (+formData.regularPrice < +formData.discountPrice)

        return setError('Discount price must be lower than regular price');

      setLoading(true);

      setError(false);

      const res = await fetch('/api/listing/create', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify({

          ...formData,

          userRef: currentUser.\_id,

        }),

      });

      const data = await res.json();

      setLoading(false);

      if (data.success === false) {

        setError(data.message);

      }

      navigate(`/listing/${data.\_id}`);

    } catch (error) {

      setError(error.message);

      setLoading(false);

    }

  };

  return (

    <main className='p-3 max-w-4xl mx-auto'>

      <h1 className='text-3xl font-semibold text-center my-7'>

        Create a Listing

      </h1>

      <form onSubmit={handleSubmit} className='flex flex-col sm:flex-row gap-4'>

        <div className='flex flex-col gap-4 flex-1'>

          <input

            type='text'

            placeholder='Name'

            className='border p-3 rounded-lg'

            id='name'

            maxLength='62'

            minLength='10'

            required

            onChange={handleChange}

            value={formData.name}

          />

          <textarea

            type='text'

            placeholder='Description'

            className='border p-3 rounded-lg'

            id='description'

            required

            onChange={handleChange}

            value={formData.description}

          />

          <input

            type='text'

            placeholder='Address'

            className='border p-3 rounded-lg'

            id='address'

            required

            onChange={handleChange}

            value={formData.address}

          />

          <div className='flex gap-6 flex-wrap'>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='sale'

                className='w-5'

                onChange={handleChange}

                checked={formData.type === 'sale'}

              />

              <span>Sell</span>

            </div>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='rent'

                className='w-5'

                onChange={handleChange}

                checked={formData.type === 'rent'}

              />

              <span>Rent</span>

            </div>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='parking'

                className='w-5'

                onChange={handleChange}

                checked={formData.parking}

              />

              <span>Parking spot</span>

            </div>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='furnished'

                className='w-5'

                onChange={handleChange}

                checked={formData.furnished}

              />

              <span>Furnished</span>

            </div>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='offer'

                className='w-5'

                onChange={handleChange}

                checked={formData.offer}

              />

              <span>Offer</span>

            </div>

          </div>

          <div className='flex flex-wrap gap-6'>

            <div className='flex items-center gap-2'>

              <input

                type='number'

                id='bedrooms'

                min='1'

                max='10'

                required

                className='p-3 border border-gray-300 rounded-lg'

                onChange={handleChange}

                value={formData.bedrooms}

              />

              <p>Beds</p>

            </div>

            <div className='flex items-center gap-2'>

              <input

                type='number'

                id='bathrooms'

                min='1'

                max='10'

                required

                className='p-3 border border-gray-300 rounded-lg'

                onChange={handleChange}

                value={formData.bathrooms}

              />

              <p>Baths</p>

            </div>

            <div className='flex items-center gap-2'>

              <input

                type='number'

                id='regularPrice'

                min='50'

                max='10000000'

                required

                className='p-3 border border-gray-300 rounded-lg'

                onChange={handleChange}

                value={formData.regularPrice}

              />

              <div className='flex flex-col items-center'>

                <p>Regular price &#8377;</p>

                {formData.type === 'rent' && (

                  <span className='text-xs'>( Rs / month)</span>

                )}

              </div>

            </div>

            {formData.offer && (

              <div className='flex items-center gap-2'>

                <input

                  type='number'

                  id='discountPrice'

                  min='0'

                  max='10000000'

                  required

                  className='p-3 border border-gray-300 rounded-lg'

                  onChange={handleChange}

                  value={formData.discountPrice}

                />

                <div className='flex flex-col items-center'>

                  <p>Discounted price  &#8377;</p>

                  {formData.type === 'rent' && (

                    <span className='text-xs'>(  Rs / month)</span>

                  )}

                </div>

              </div>

            )}

          </div>

        </div>

        <div className='flex flex-col flex-1 gap-4'>

          <p className='font-semibold'>

            Images:

            <span className='font-normal text-gray-600 ml-2'>

              The first image will be the cover (max 6)

            </span>

          </p>

          <div className='flex gap-4'>

            <input

              onChange={(e) => setFiles(e.target.files)}

              className='p-3 border border-gray-300 rounded w-full'

              type='file'

              id='images'

              accept='image/\*'

              multiple

            />

            <button

              type='button'

              disabled={uploading}

              onClick={handleImageSubmit}

              className='p-3 text-green-700 border border-green-700 rounded uppercase hover:shadow-lg disabled:opacity-80'

            >

              {uploading ? 'Uploading...' : 'Upload'}

            </button>

          </div>

          <p className='text-red-700 text-sm'>

            {imageUploadError && imageUploadError}

          </p>

          {formData.imageUrls.length > 0 &&

            formData.imageUrls.map((url, index) => (

              <div

                key={url}

                className='flex justify-between p-3 border items-center'

              >

                <img

                  src={url}

                  alt='listing image'

                  className='w-20 h-20 object-contain rounded-lg'

                />

                <button

                  type='button'

                  onClick={() => handleRemoveImage(index)}

                  className='p-3 text-red-700 rounded-lg uppercase hover:opacity-75'

                >

                  Delete

                </button>

              </div>

            ))}

          <button

            disabled={loading || uploading}

            className='p-3 bg-blue-700 text-white rounded-lg uppercase hover:opacity-95 disabled:opacity-80'

          >

            {loading ? 'Creating...' : 'Create listing'}

          </button>

          {error && <p className='text-red-700 text-sm'>{error}</p>}

        </div>

      </form>

    </main>

  );

}

**Update Listing**

import { useEffect, useState } from 'react';

import {

  getDownloadURL,

  getStorage,

  ref,

  uploadBytesResumable,

} from 'firebase/storage';

import { app } from '../firebase';

import { useSelector } from 'react-redux';

import { useNavigate, useParams } from 'react-router-dom';

export default function CreateListing() {

  const { currentUser } = useSelector((state) => state.user);

  const navigate = useNavigate();

  const params = useParams();

  const [files, setFiles] = useState([]);

  const [formData, setFormData] = useState({

    imageUrls: [],

    name: '',

    description: '',

    address: '',

    type: 'rent',

    bedrooms: 1,

    bathrooms: 1,

    regularPrice: 50,

    discountPrice: 0,

    offer: false,

    parking: false,

    furnished: false,

  });

  const [imageUploadError, setImageUploadError] = useState(false);

  const [uploading, setUploading] = useState(false);

  const [error, setError] = useState(false);

  const [loading, setLoading] = useState(false);

  useEffect(() => {

    const fetchListing = async () => {

      const listingId = params.listingId;

      const res = await fetch(`/api/listing/get/${listingId}`);

      const data = await res.json();

      if (data.success === false) {

        console.log(data.message);

        return;

      }

      setFormData(data);

    };

    fetchListing();

  }, []);

  const handleImageSubmit = (e) => {

    if (files.length > 0 && files.length + formData.imageUrls.length < 7) {

      setUploading(true);

      setImageUploadError(false);

      const promises = [];

      for (let i = 0; i < files.length; i++) {

        promises.push(storeImage(files[i]));

      }

      Promise.all(promises)

        .then((urls) => {

          setFormData({

            ...formData,

            imageUrls: formData.imageUrls.concat(urls),

          });

          setImageUploadError(false);

          setUploading(false);

        })

        .catch((err) => {

          setImageUploadError('Image upload failed (2 mb max per image)');

          setUploading(false);

        });

    } else {

      setImageUploadError('You can only upload 6 images per listing');

      setUploading(false);

    }

  };

  const storeImage = async (file) => {

    return new Promise((resolve, reject) => {

      const storage = getStorage(app);

      const fileName = new Date().getTime() + file.name;

      const storageRef = ref(storage, fileName);

      const uploadTask = uploadBytesResumable(storageRef, file);

      uploadTask.on(

        'state\_changed',

        (snapshot) => {

          const progress =

            (snapshot.bytesTransferred / snapshot.totalBytes) \* 100;

          console.log(`Upload is ${progress}% done`);

        },

        (error) => {

          reject(error);

        },

        () => {

          getDownloadURL(uploadTask.snapshot.ref).then((downloadURL) => {

            resolve(downloadURL);

          });

        }

      );

    });

  };

  const handleRemoveImage = (index) => {

    setFormData({

      ...formData,

      imageUrls: formData.imageUrls.filter((\_, i) => i !== index),

    });

  };

  const handleChange = (e) => {

    if (e.target.id === 'sale' || e.target.id === 'rent') {

      setFormData({

        ...formData,

        type: e.target.id,

      });

    }

    if (

      e.target.id === 'parking' ||

      e.target.id === 'furnished' ||

      e.target.id === 'offer'

    ) {

      setFormData({

        ...formData,

        [e.target.id]: e.target.checked,

      });

    }

    if (

      e.target.type === 'number' ||

      e.target.type === 'text' ||

      e.target.type === 'textarea'

    ) {

      setFormData({

        ...formData,

        [e.target.id]: e.target.value,

      });

    }

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      if (formData.imageUrls.length < 1)

        return setError('You must upload at least one image');

      if (+formData.regularPrice < +formData.discountPrice)

        return setError('Discount price must be lower than regular price');

      setLoading(true);

      setError(false);

      const res = await fetch(`/api/listing/update/${params.listingId}`, {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify({

          ...formData,

          userRef: currentUser.\_id,

        }),

      });

      const data = await res.json();

      setLoading(false);

      if (data.success === false) {

        setError(data.message);

      }

      navigate(`/listing/${data.\_id}`);

    } catch (error) {

      setError(error.message);

      setLoading(false);

    }

  };

  return (

    <main className='p-3 max-w-4xl mx-auto'>

      <h1 className='text-3xl font-semibold text-center my-7'>

        Update a Listing

      </h1>

      <form onSubmit={handleSubmit} className='flex flex-col sm:flex-row gap-4'>

        <div className='flex flex-col gap-4 flex-1'>

          <input

            type='text'

            placeholder='Name'

            className='border p-3 rounded-lg'

            id='name'

            maxLength='62'

            minLength='10'

            required

            onChange={handleChange}

            value={formData.name}

          />

          <textarea

            type='text'

            placeholder='Description'

            className='border p-3 rounded-lg'

            id='description'

            required

            onChange={handleChange}

            value={formData.description}

          />

          <input

            type='text'

            placeholder='Address'

            className='border p-3 rounded-lg'

            id='address'

            required

            onChange={handleChange}

            value={formData.address}

          />

          <div className='flex gap-6 flex-wrap'>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='sale'

                className='w-5'

                onChange={handleChange}

                checked={formData.type === 'sale'}

              />

              <span>Sell</span>

            </div>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='rent'

                className='w-5'

                onChange={handleChange}

                checked={formData.type === 'rent'}

              />

              <span>Rent</span>

            </div>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='parking'

                className='w-5'

                onChange={handleChange}

                checked={formData.parking}

              />

              <span>Parking spot</span>

            </div>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='furnished'

                className='w-5'

                onChange={handleChange}

                checked={formData.furnished}

              />

              <span>Furnished</span>

            </div>

            <div className='flex gap-2'>

              <input

                type='checkbox'

                id='offer'

                className='w-5'

                onChange={handleChange}

                checked={formData.offer}

              />

              <span>Offer</span>

            </div>

          </div>

          <div className='flex flex-wrap gap-6'>

            <div className='flex items-center gap-2'>

              <input

                type='number'

                id='bedrooms'

                min='1'

                max='10'

                required

                className='p-3 border border-gray-300 rounded-lg'

                onChange={handleChange}

                value={formData.bedrooms}

              />

              <p>Beds</p>

            </div>

            <div className='flex items-center gap-2'>

              <input

                type='number'

                id='bathrooms'

                min='1'

                max='10'

                required

                className='p-3 border border-gray-300 rounded-lg'

                onChange={handleChange}

                value={formData.bathrooms}

              />

              <p>Baths</p>

            </div>

            <div className='flex items-center gap-2'>

              <input

                type='number'

                id='regularPrice'

                min='50'

                max='10000000'

                required

                className='p-3 border border-gray-300 rounded-lg'

                onChange={handleChange}

                value={formData.regularPrice}

              />

              <div className='flex flex-col items-center'>

                <p>&#8377; Regular price </p>

                {formData.type === 'rent' && (

                  <span className='text-xs'>( Rs / month)</span>

                )}

              </div>

            </div>

            {formData.offer && (

              <div className='flex items-center gap-2'>

                <input

                  type='number'

                  id='discountPrice'

                  min='0'

                  max='10000000'

                  required

                  className='p-3 border border-gray-300 rounded-lg'

                  onChange={handleChange}

                  value={formData.discountPrice}

                />

                <div className='flex flex-col items-center'>

                  <p>&#8377; Discounted price  </p>

                  {formData.type === 'rent' && (

                    <span className='text-xs'>(  Rs / month)</span>

                  )}

                </div>

              </div>

            )}

          </div>

        </div>

        <div className='flex flex-col flex-1 gap-4'>

          <p className='font-semibold'>

            Images:

            <span className='font-normal text-gray-600 ml-2'>

              The first image will be the cover (max 6)

            </span>

          </p>

          <div className='flex gap-4'>

            <input

              onChange={(e) => setFiles(e.target.files)}

              className='p-3 border border-gray-300 rounded w-full'

              type='file'

              id='images'

              accept='image/\*'

              multiple

            />

            <button

              type='button'

              disabled={uploading}

              onClick={handleImageSubmit}

              className='p-3 text-green-700 border border-green-700 rounded uppercase hover:shadow-lg disabled:opacity-80'

            >

              {uploading ? 'Uploading...' : 'Upload'}

            </button>

          </div>

          <p className='text-red-700 text-sm'>

            {imageUploadError && imageUploadError}

          </p>

          {formData.imageUrls.length > 0 &&

            formData.imageUrls.map((url, index) => (

              <div

                key={url}

                className='flex justify-between p-3 border items-center'

              >

                <img

                  src={url}

                  alt='listing image'

                  className='w-20 h-20 object-contain rounded-lg'

                />

                <button

                  type='button'

                  onClick={() => handleRemoveImage(index)}

                  className='p-3 text-red-700 rounded-lg uppercase hover:opacity-75'

                >

                  Delete

                </button>

              </div>

            ))}

          <button

            disabled={loading || uploading}

            className='p-3 bg-blue-700 text-white rounded-lg uppercase hover:opacity-95 disabled:opacity-80'

          >

            {loading ? 'Updating...' : 'Update listing'}

          </button>

          {error && <p className='text-red-700 text-sm'>{error}</p>}

        </div>

      </form>

    </main>

  );

}

**Profile**

import { useSelector } from 'react-redux';

import { useRef, useState, useEffect } from 'react';

import {

  getDownloadURL,

  getStorage,

  ref,

  uploadBytesResumable,

} from 'firebase/storage';

import { app } from '../firebase';

import {

  updateUserStart,

  updateUserSuccess,

  updateUserFailure,

  deleteUserFailure,

  deleteUserStart,

  deleteUserSuccess,

  signOutUserStart,

} from '../redux/user/userSlice';

import { useDispatch } from 'react-redux';

import { Link } from 'react-router-dom';

export default function Profile() {

  const fileRef = useRef(null);

  const { currentUser, loading, error } = useSelector((state) => state.user);

  const [file, setFile] = useState(undefined);

  const [filePerc, setFilePerc] = useState(0);

  const [fileUploadError, setFileUploadError] = useState(false);

  const [formData, setFormData] = useState({});

  const [updateSuccess, setUpdateSuccess] = useState(false);

  const [showListingsError, setShowListingsError] = useState(false);

  const [userListings, setUserListings] = useState([]);

  const dispatch = useDispatch();

  // firebase storage

  // allow read;

  // allow write: if

  // request.resource.size < 2 \* 1024 \* 1024 &&

  // request.resource.contentType.matches('image/.\*')

  useEffect(() => {

    if (file) {

      handleFileUpload(file);

    }

  }, [file]);

  const handleFileUpload = (file) => {

    const storage = getStorage(app);

    const fileName = new Date().getTime() + file.name;

    const storageRef = ref(storage, fileName);

    const uploadTask = uploadBytesResumable(storageRef, file);

    uploadTask.on(

      'state\_changed',

      (snapshot) => {

        const progress =

          (snapshot.bytesTransferred / snapshot.totalBytes) \* 100;

        setFilePerc(Math.round(progress));

      },

      (error) => {

        setFileUploadError(true);

      },

      () => {

        getDownloadURL(uploadTask.snapshot.ref).then((downloadURL) =>

          setFormData({ ...formData, avatar: downloadURL })

        );

      }

    );

  };

  const handleChange = (e) => {

    setFormData({ ...formData, [e.target.id]: e.target.value });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      dispatch(updateUserStart());

      const res = await fetch(`/api/user/update/${currentUser.\_id}`, {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify(formData),

      });

      const data = await res.json();

      if (data.success === false) {

        dispatch(updateUserFailure(data.message));

        return;

      }

      dispatch(updateUserSuccess(data));

      setUpdateSuccess(true);

    } catch (error) {

      dispatch(updateUserFailure(error.message));

    }

  };

  const handleDeleteUser = async () => {

    try {

      dispatch(deleteUserStart());

      const res = await fetch(`/api/user/delete/${currentUser.\_id}`, {

        method: 'DELETE',

      });

      const data = await res.json();

      if (data.success === false) {

        dispatch(deleteUserFailure(data.message));

        return;

      }

      dispatch(deleteUserSuccess(data));

    } catch (error) {

      dispatch(deleteUserFailure(error.message));

    }

  };

  const handleSignOut = async () => {

    try {

      dispatch(signOutUserStart());

      const res = await fetch('/api/auth/signout');

      const data = await res.json();

      if (data.success === false) {

        dispatch(deleteUserFailure(data.message));

        return;

      }

      dispatch(deleteUserSuccess(data));

    } catch (error) {

      dispatch(deleteUserFailure(data.message));

    }

  };

  const handleShowListings = async () => {

    try {

      setShowListingsError(false);

      const res = await fetch(`/api/user/listings/${currentUser.\_id}`);

      const data = await res.json();

      if (data.success === false) {

        setShowListingsError(true);

        return;

      }

      setUserListings(data);

    } catch (error) {

      setShowListingsError(true);

    }

  };

  const handleListingDelete = async (listingId) => {

    try {

      const res = await fetch(`/api/listing/delete/${listingId}`, {

        method: 'DELETE',

      });

      const data = await res.json();

      if (data.success === false) {

        console.log(data.message);

        return;

      }

      setUserListings((prev) =>

        prev.filter((listing) => listing.\_id !== listingId)

      );

    } catch (error) {

      console.log(error.message);

    }

  };

  return (

    <div className='p-3 max-w-lg mx-auto'>

      <h1 className='text-3xl font-semibold text-center my-7'>Profile</h1>

      <form onSubmit={handleSubmit} className='flex flex-col gap-4'>

        <input

          onChange={(e) => setFile(e.target.files[0])}

          type='file'

          ref={fileRef}

          hidden

          accept='image/\*'

        />

        <img

          onClick={() => fileRef.current.click()}

          src={formData.avatar || currentUser.avatar}

          alt='profile'

          className='rounded-full h-24 w-24 object-cover cursor-pointer self-center mt-2'

        />

        <p className='text-sm self-center'>

          {fileUploadError ? (

            <span className='text-red-700'>

              Error Image upload (image must be less than 2 mb)

            </span>

          ) : filePerc > 0 && filePerc < 100 ? (

            <span className='text-slate-700'>{`Uploading ${filePerc}%`}</span>

          ) : filePerc === 100 ? (

            <span className='text-green-700'>Image successfully uploaded!</span>

          ) : (

            ''

          )}

        </p>

        <input

          type='text'

          placeholder='username'

          defaultValue={currentUser.username}

          id='username'

          className='border p-3 rounded-lg'

          onChange={handleChange}

        />

        <input

          type='email'

          placeholder='email'

          id='email'

          defaultValue={currentUser.email}

          className='border p-3 rounded-lg'

          onChange={handleChange}

        />

        <input

          type='password'

          placeholder='password'

          onChange={handleChange}

          id='password'

          className='border p-3 rounded-lg'

        />

        <button

          disabled={loading}

          className='bg-slate-700 text-white rounded-lg p-3 uppercase hover:opacity-95 disabled:opacity-80'

        >

          {loading ? 'Loading...' : 'Update'}

        </button>

        <Link

          className='bg-green-700 text-white p-3 rounded-lg uppercase text-center hover:opacity-95'

          to={'/create-listing'}

        >

          Create Listing

        </Link>

      </form>

      <div className='flex justify-between mt-5'>

        <span

          onClick={handleDeleteUser}

          className='text-red-700 cursor-pointer'

        >

          Delete account

        </span>

        <span onClick={handleSignOut} className='text-red-700 cursor-pointer'>

          Sign out

        </span>

      </div>

      <p className='text-red-700 mt-5'>{error ? error : ''}</p>

      <p className='text-green-700 mt-5'>

        {updateSuccess ? 'User is updated successfully!' : ''}

      </p>

      <button onClick={handleShowListings} className='text-green-700 w-full'>

        Show Listings

      </button>

      <p className='text-red-700 mt-5'>

        {showListingsError ? 'Error showing listings' : ''}

      </p>

      {userListings && userListings.length > 0 && (

        <div className='flex flex-col gap-4'>

          <h1 className='text-center mt-7 text-2xl font-semibold'>

            Your Listings

          </h1>

          {userListings.map((listing) => (

            <div

              key={listing.\_id}

              className='border rounded-lg p-3 flex justify-between items-center gap-4'

            >

              <Link to={`/listing/${listing.\_id}`}>

                <img

                  src={listing.imageUrls[0]}

                  alt='listing cover'

                  className='h-16 w-16 object-contain'

                />

              </Link>

              <Link

                className='text-slate-700 font-semibold  hover:underline truncate flex-1'

                to={`/listing/${listing.\_id}`}

              >

                <p>{listing.name}</p>

              </Link>

              <div className='flex flex-col item-center'>

                <button

                  onClick={() => handleListingDelete(listing.\_id)}

                  className='text-red-700 uppercase'

                >

                  Delete

                </button>

                <Link to={`/update-listing/${listing.\_id}`}>

                  <button className='text-green-700 uppercase'>Edit</button>

                </Link>

              </div>

            </div>

          ))}

        </div>

      )}

    </div>

  );

}

**Home**

import { useEffect, useState } from 'react';

import { Link } from 'react-router-dom';

import { Swiper, SwiperSlide } from 'swiper/react';

import { Navigation } from 'swiper/modules';

import SwiperCore from 'swiper';

import 'swiper/css/bundle';

import ListingItem from '../components/ListingItem';

export default function Home() {

  const [offerListings, setOfferListings] = useState([]);

  const [saleListings, setSaleListings] = useState([]);

  const [rentListings, setRentListings] = useState([]);

  SwiperCore.use([Navigation]);

  console.log(offerListings);

  useEffect(() => {

    const fetchOfferListings = async () => {

      try {

        const res = await fetch('/api/listing/get?offer=true&limit=4');

        const data = await res.json();

        setOfferListings(data);

        fetchRentListings();

      } catch (error) {

        console.log(error);

      }

    };

    const fetchRentListings = async () => {

      try {

        const res = await fetch('/api/listing/get?type=rent&limit=4');

        const data = await res.json();

        setRentListings(data);

        fetchSaleListings();

      } catch (error) {

        console.log(error);

      }

    };

    const fetchSaleListings = async () => {

      try {

        const res = await fetch('/api/listing/get?type=sale&limit=4');

        const data = await res.json();

        setSaleListings(data);

      } catch (error) {

        log(error);

      }

    };

    fetchOfferListings();

  }, []);

  return (

    <div>

      {/\* top \*/}

      <div className='flex flex-col gap-6 p-28 px-3 max-w-6xl mx-auto'>

        <h1 className='text-slate-700 font-bold text-3xl lg:text-6xl'>

          Find your next <span className='text-slate-500'>perfect</span>

          <br />

          place with ease

        </h1>

        <div className='text-gray-400 text-xs sm:text-sm'>

          Apana Flat is the best place to find your next perfect place to

          live.

          <br />

          We have a wide range of properties for you to choose from.

        </div>

        <Link

          to={'/search'}

          className='text-xs sm:text-sm text-blue-800 font-bold hover:underline'

        >

          Let's get started...

        </Link>

      </div>

      {/\* swiper \*/}

      <Swiper navigation>

        {offerListings &&

          offerListings.length > 0 &&

          offerListings.map((listing) => (

            <SwiperSlide>

              <div

                style={{

                  background: `url(${listing.imageUrls[0]}) center no-repeat`,

                  backgroundSize: 'cover',

                }}

                className='h-[500px]'

                key={listing.\_id}

              ></div>

            </SwiperSlide>

          ))}

      </Swiper>

      {/\* listing results for offer, sale and rent \*/}

      <div className='max-w-6xl mx-auto p-3 flex flex-col gap-8 my-10'>

        {offerListings && offerListings.length > 0 && (

          <div className=''>

            <div className='my-3'>

              <h2 className='text-2xl font-semibold text-slate-600'>Recent offers</h2>

              <Link className='text-sm text-blue-800 hover:underline' to={'/search?offer=true'}>Show more offers</Link>

            </div>

            <div className='flex flex-wrap gap-4'>

              {offerListings.map((listing) => (

                <ListingItem listing={listing} key={listing.\_id} />

              ))}

            </div>

          </div>

        )}

        {rentListings && rentListings.length > 0 && (

          <div className=''>

            <div className='my-3'>

              <h2 className='text-2xl font-semibold text-slate-600'>Recent places for rent</h2>

              <Link className='text-sm text-blue-800 hover:underline' to={'/search?type=rent'}>Show more places for rent</Link>

            </div>

            <div className='flex flex-wrap gap-4'>

              {rentListings.map((listing) => (

                <ListingItem listing={listing} key={listing.\_id} />

              ))}

            </div>

          </div>

        )}

        {saleListings && saleListings.length > 0 && (

          <div className=''>

            <div className='my-3'>

              <h2 className='text-2xl font-semibold text-slate-600'>Recent places for sale</h2>

              <Link className='text-sm text-blue-800 hover:underline' to={'/search?type=sale'}>Show more places for sale</Link>

            </div>

            <div className='flex flex-wrap gap-4'>

              {saleListings.map((listing) => (

                <ListingItem listing={listing} key={listing.\_id} />

              ))}

            </div>

          </div>

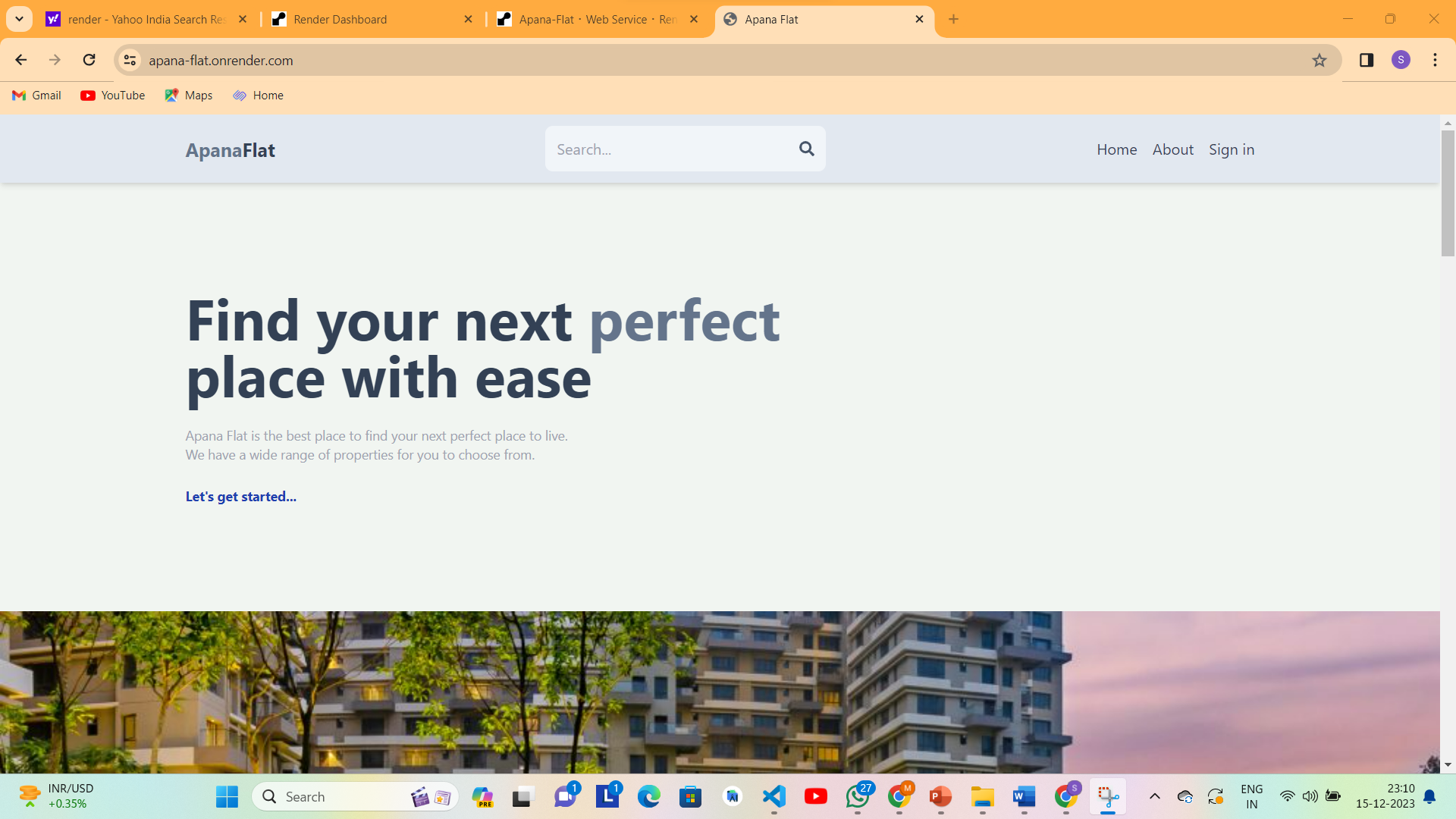
        )}

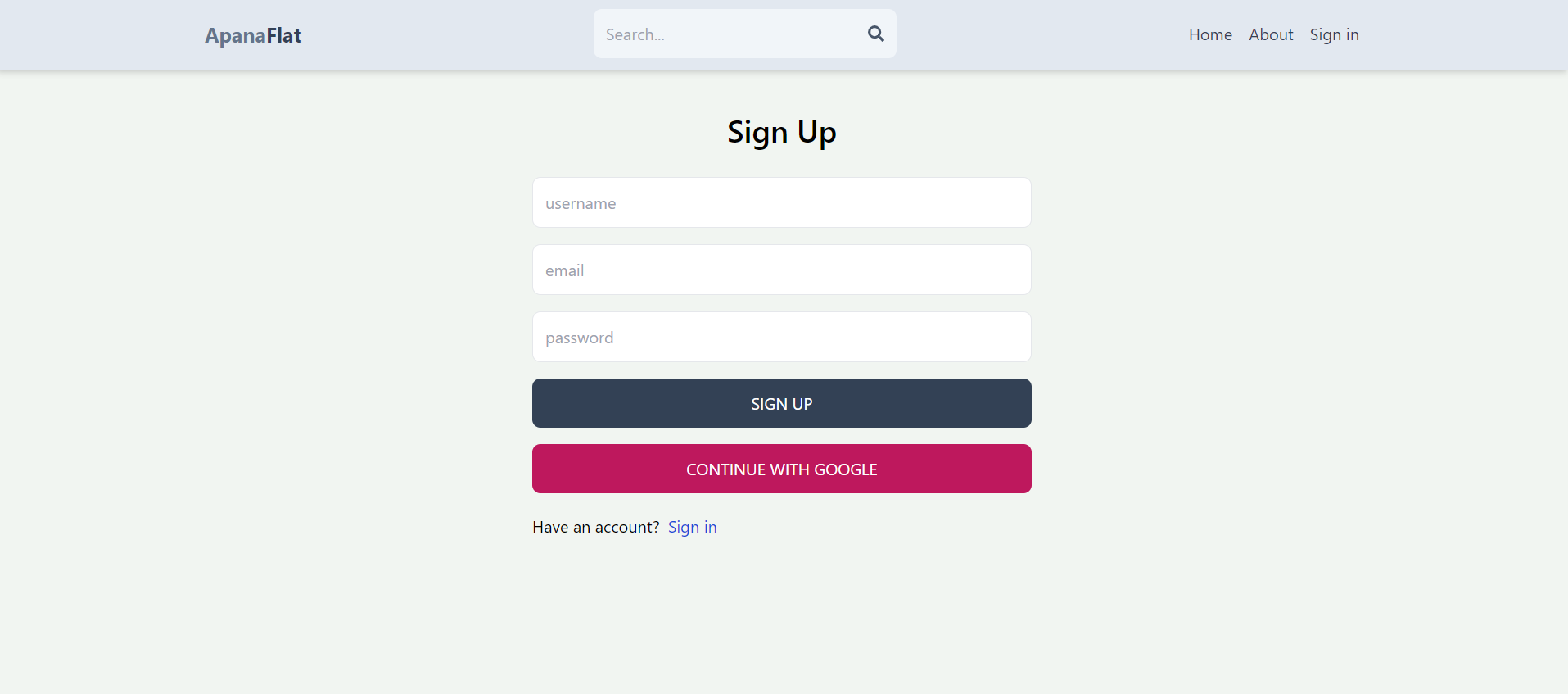
      </div>

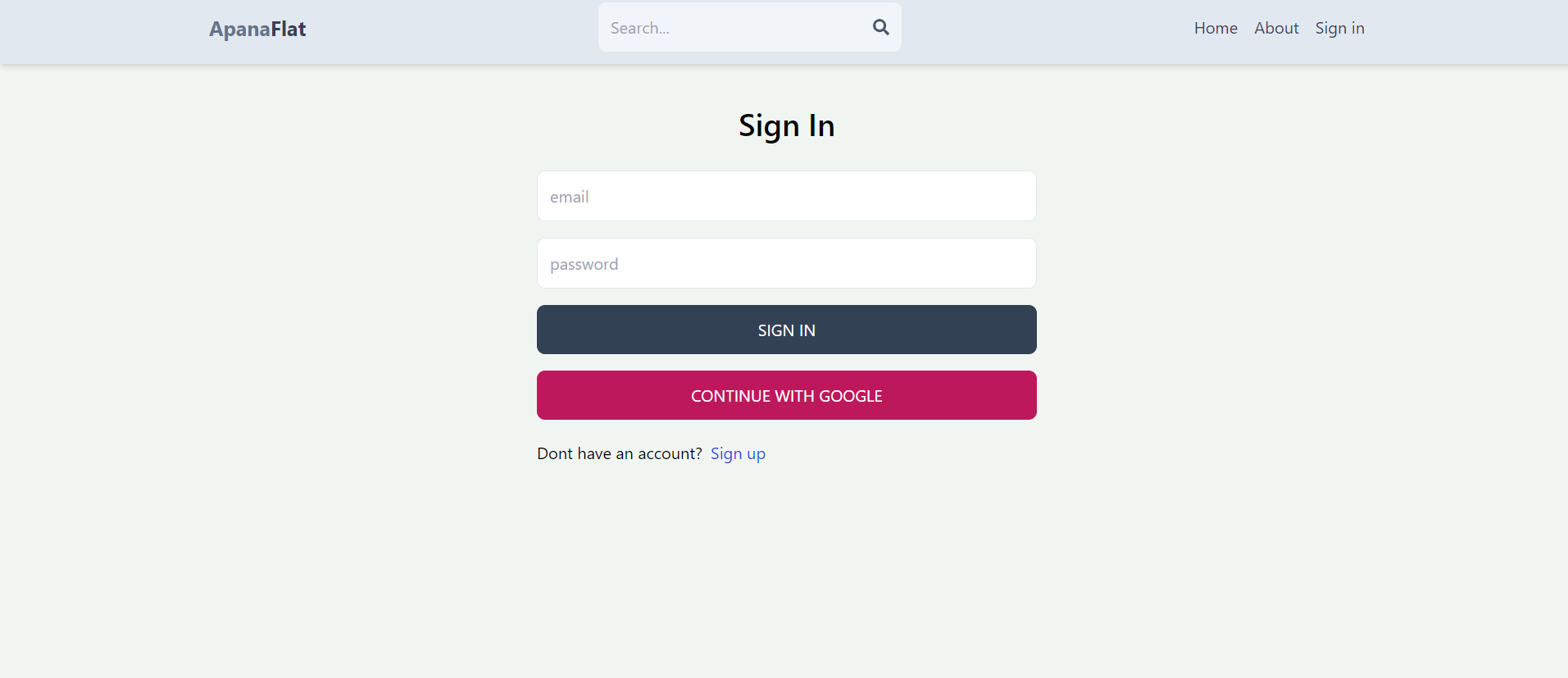
    </div>

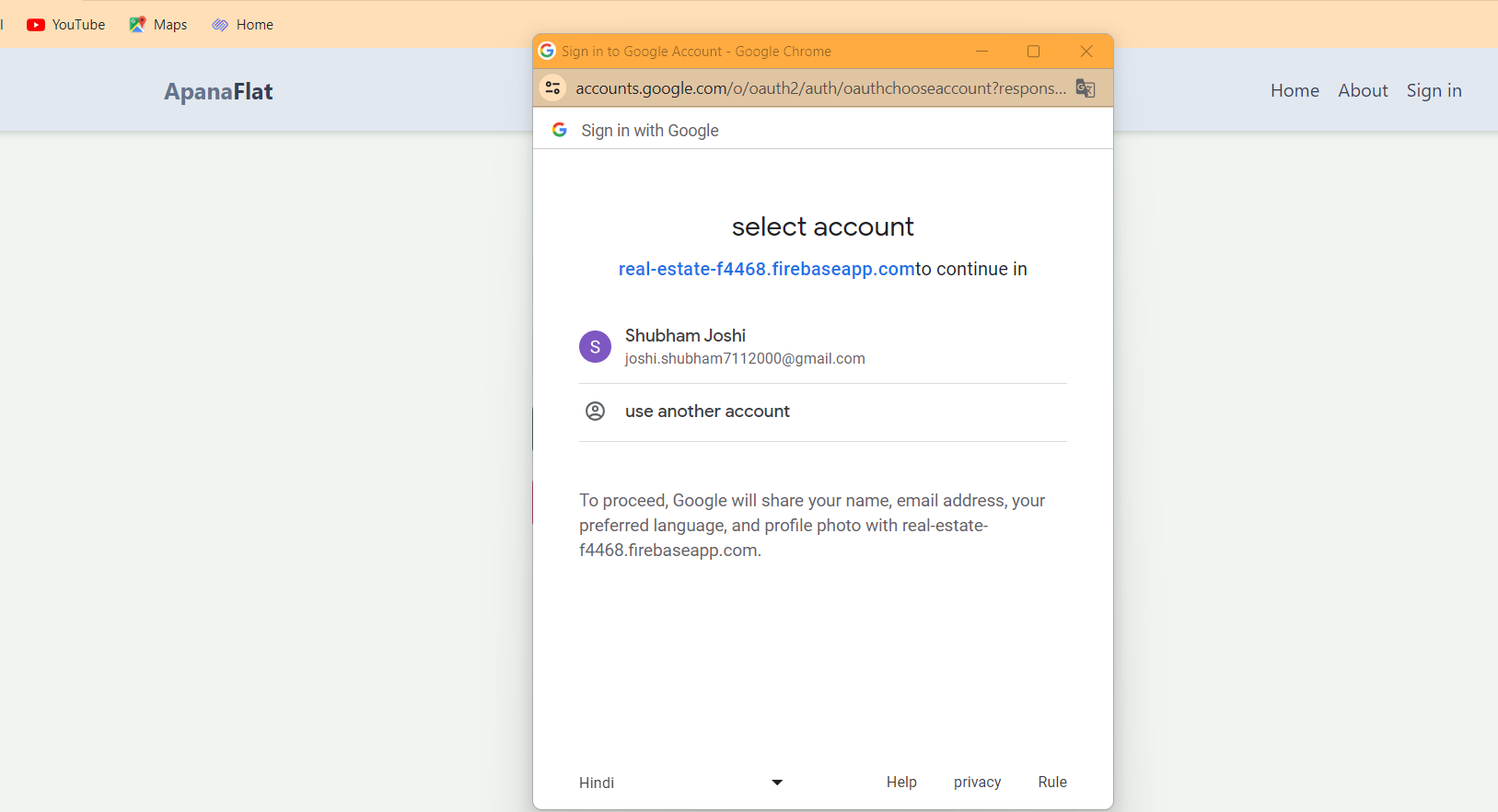
**5. OUTPUT FORMS & REPORTS**

**5.1 INPUT / OUTPUT FORMS WITH SCREENSHOTS**

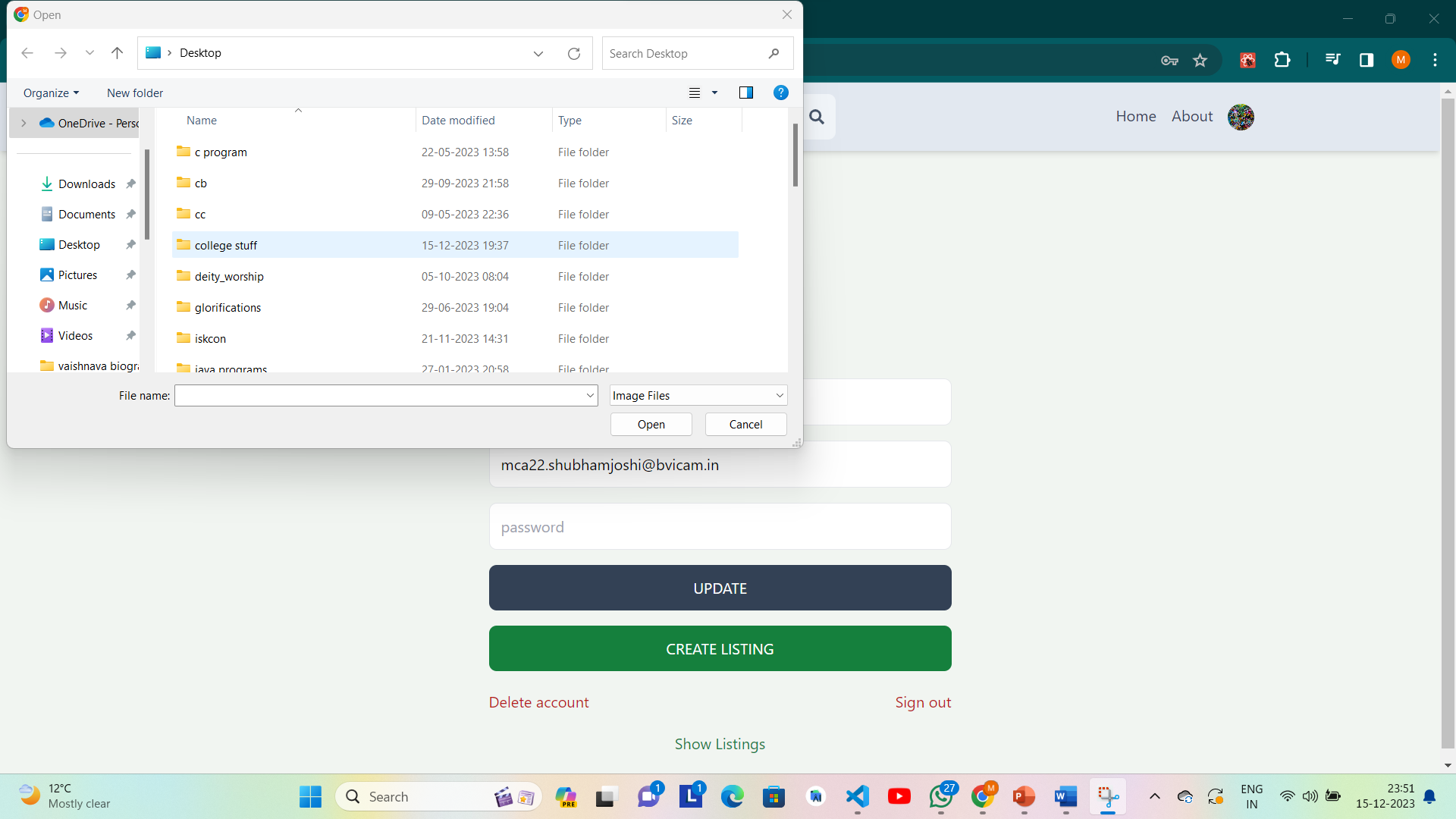
****

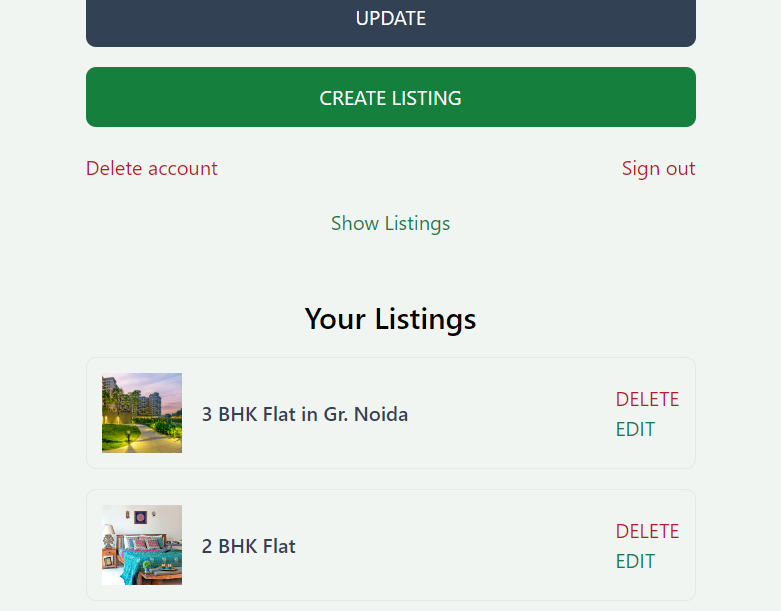


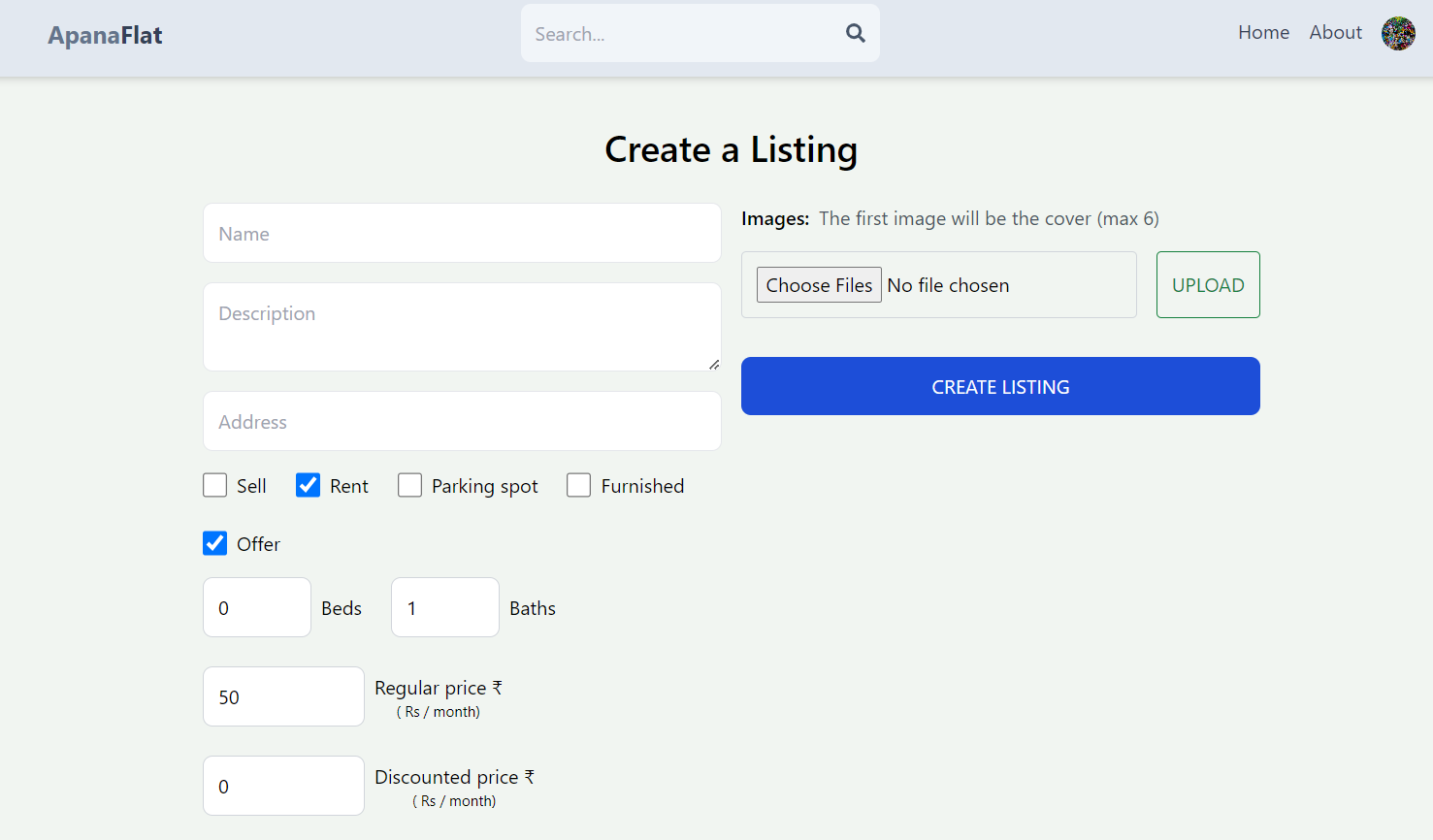


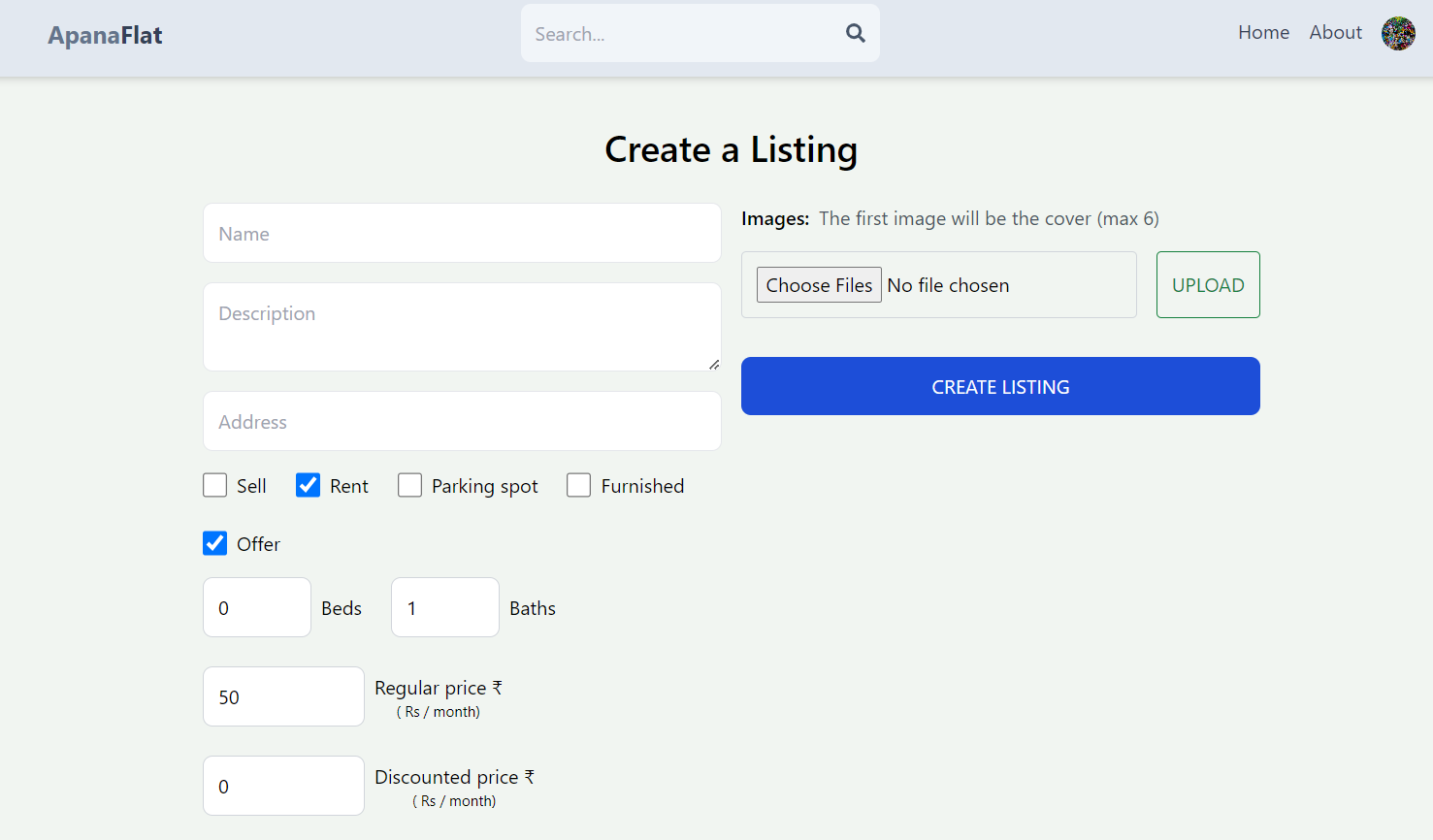


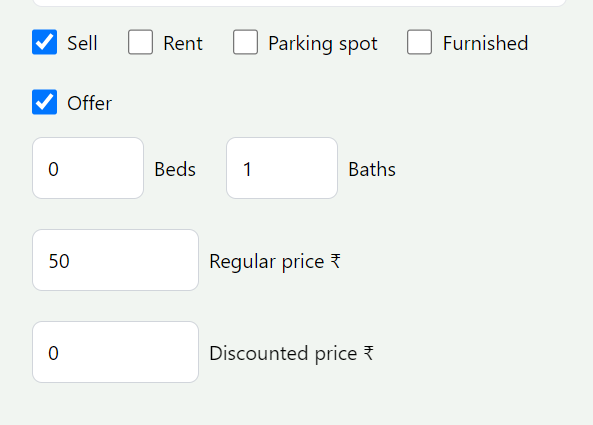


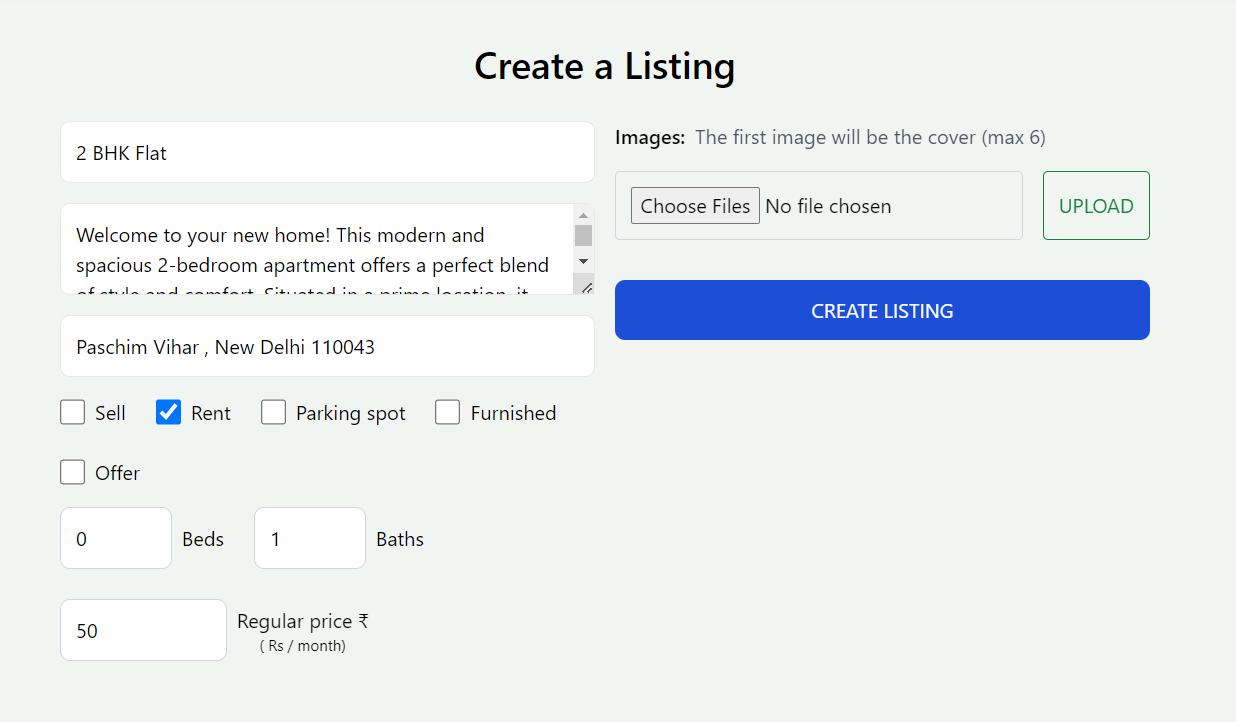


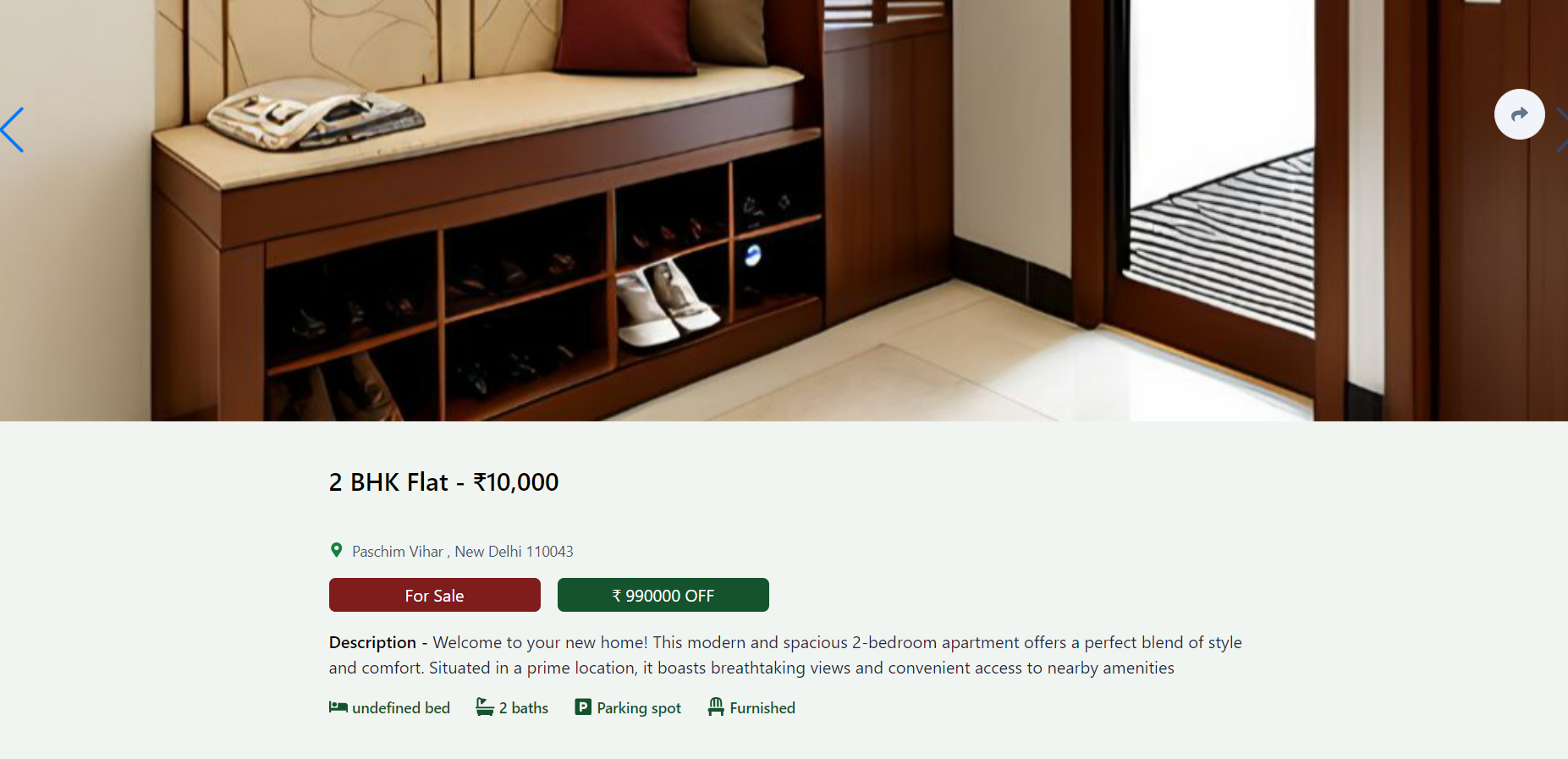


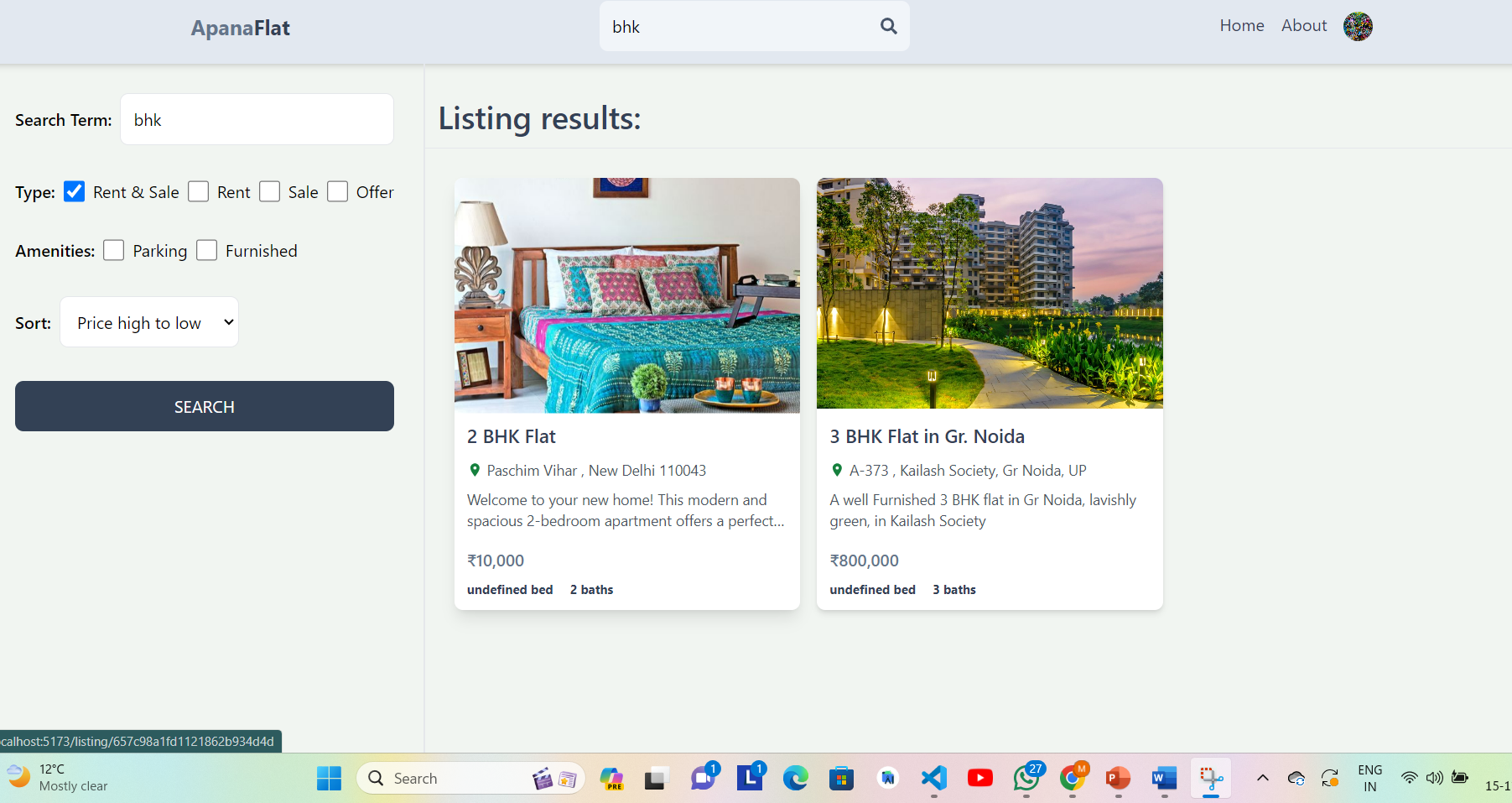


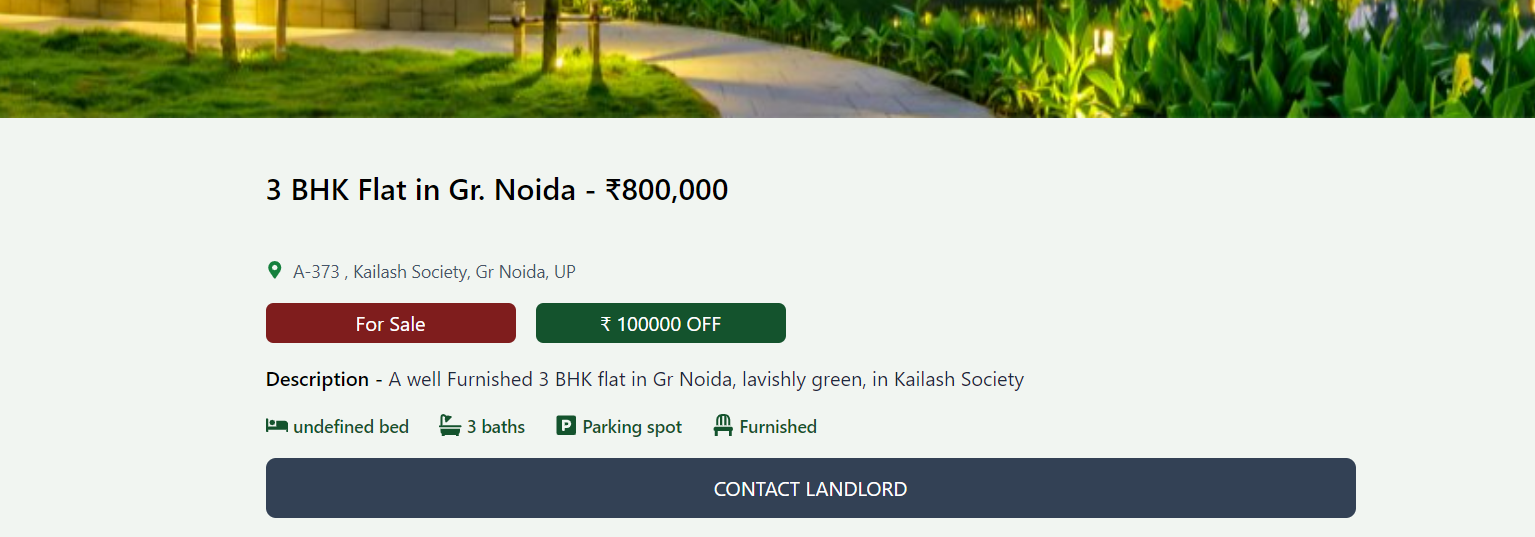


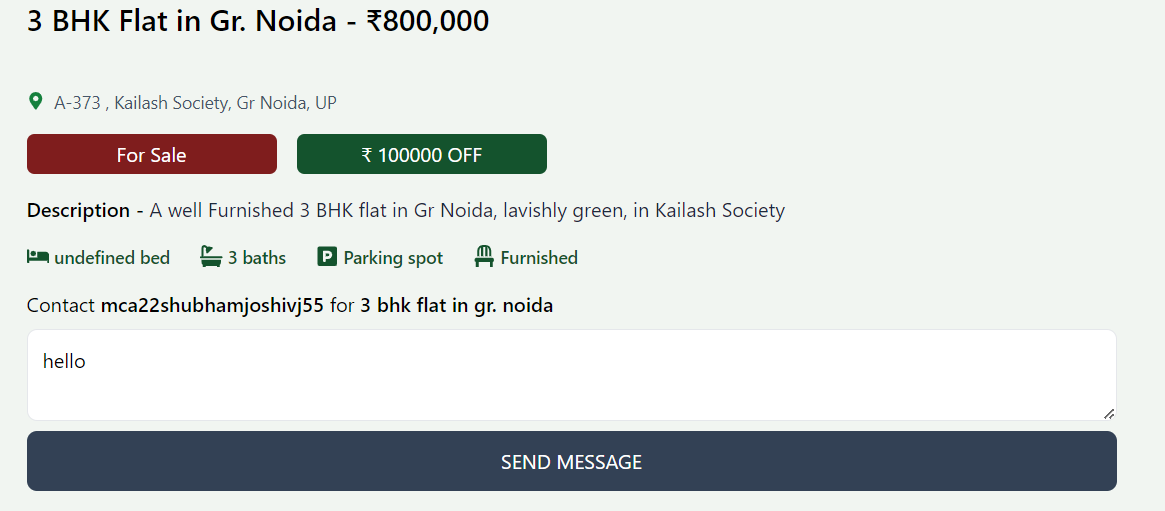


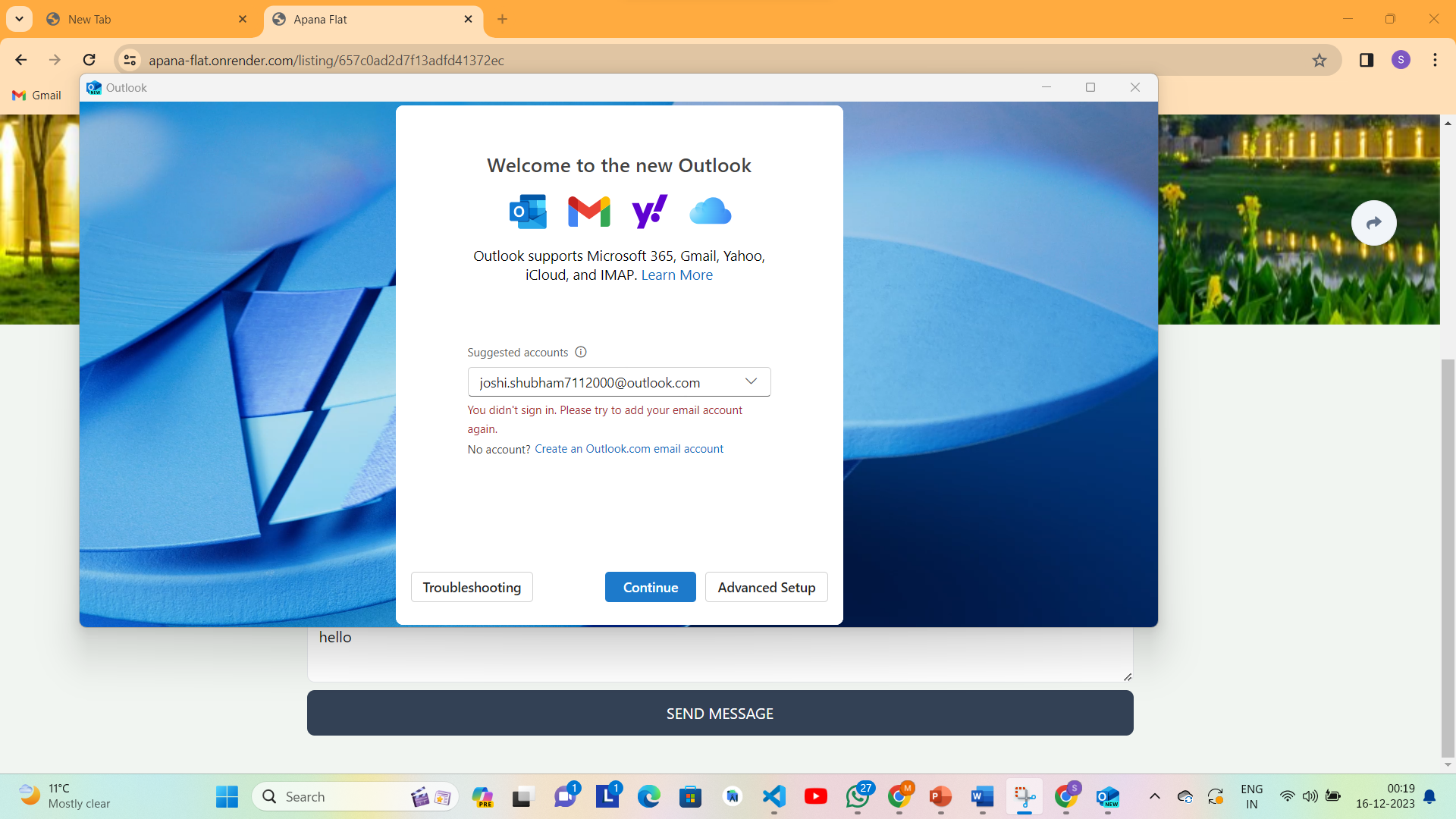


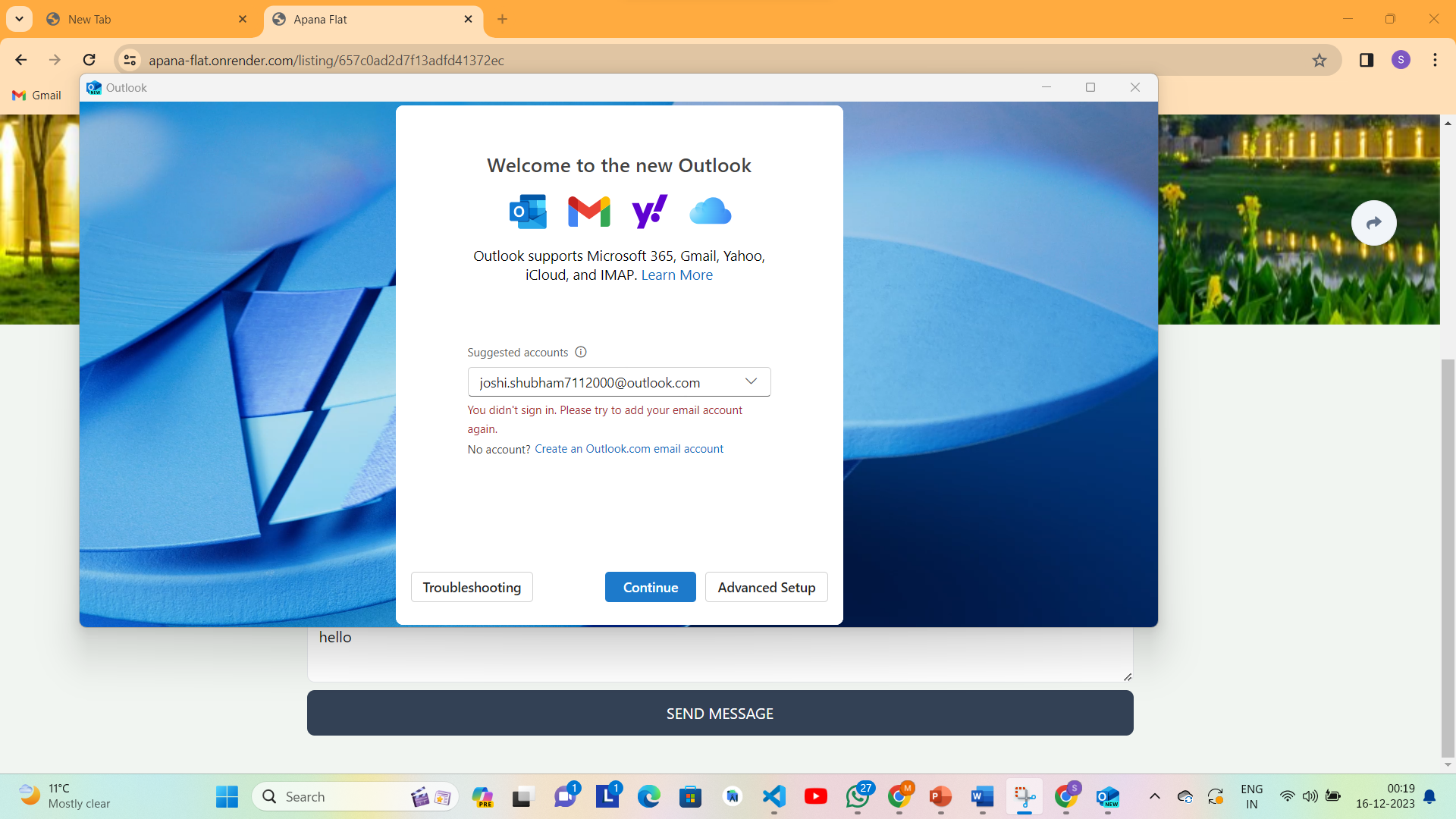












**5. CONCLUSION & FUTURE ENHANCEMENT**

**Future Scope: Enhancing User Experience and Functionality**

1. **Payment Gateway Integration:**
   * Streamline the rental process by integrating a secure and efficient payment gateway, providing users with a seamless transaction experience.
2. **Live Location Facility:**
   * Implement real-time location tracking to empower users with accurate information about the property's surroundings and nearby amenities.
3. **Diversified Contact Methods:**
   * Enhance user engagement by incorporating multiple communication channels, such as in-app messaging, phone calls, and email, offering users flexibility to connect with property owners effortlessly.
4. **Review System for Nearby Places:**
   * Enrich user decision-making by incorporating a review system that allows tenants to share their experiences and insights about nearby places, neighborhoods, and local amenities.
5. **Location-Based Categorization:**
   * Improve property search efficiency by categorizing listings based on specific locations, enabling users to explore and compare options within targeted areas like India, Delhi, Noida, and more.

**Benefits:**

* **User-Friendly Transactions:**
  + Simplify the rental process with a secure payment gateway, ensuring hassle-free transactions for both tenants and property owners.
* **Enhanced User Connectivity:**
  + Offer users a variety of communication options, fostering seamless interaction between tenants and property owners.
* **Informed Decision-Making:**
  + Facilitate better decision-making by providing users with real-time location data and authentic reviews of nearby places.
* **Efficient Property Exploration:**
  + Streamline the property search experience by categorizing listings based on specific locations, allowing users to focus on areas of interest.

**Bibliography**

<https://www.developer.android.com>

<https://www.google.com>

<https://www.tutorialspoint.com/android>

<https://www.javatpoint.com/android-tutorial>

<http://www.vogella.com/tutorials/android.html>

<http://www.coreservlets.com/android-tutorial/>

<http://www.androidauthority.com/android-studio-tutorial-beginners>

<https://www.lucidchart.com/pages/uml-use-case-diagram>

<https://app.creately.com/diagram/IB2Z9tiq8TZ/edit>

https://www.geeksforgeeks.org/android-ui-layouts/?ref=lbp