RentEase

by

Suraj Singh Mehra (2200320140170)

Under the guidance of

Ms. Savita Singh (Assistant Professor)

Department of Computer Applications

Estd. 2000

ABES Engineering College

19th Km Stone, NH-09, Ghaziabad (U.P)

May, 2024

RentEase

by

Suraj Singh Mehra (2200320140170)

Submitted to the Department of Computer Applications
in partial fulfillment of the requirements
for the degree of
Master of Computer Application

Under the guidance of

Ms. Savita Singh (Assistant Professor)
Department of Computer Applications

ABES Engineering College
19th Km Stone, NH-09, Ghaziabad (U.P)
May, 2024

DECLARATION

I hereby declare that the work being presented in this report entitled "**RentEase**" is an authentic record of my work carried out under the supervision of "**Ms. Savita Singh**".

The matter embodied in this report has not been submitted by me for the award of any other degree.

Date:	**Signature of Student**
	Suraj Singh Mehra
	Dept.: Computer Applications

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of HOD	**Signature of Supervisor**
Prof. (Dr.) Devendra Kumar	**Ms. Savita Singh**
HOD-MCA	**Assistant Professor**
Dept.:-Computer Applications	
Date:	

CERTIFICATE

This is to certify that Project Report entitled **“RentEase”** which is submitted by **Suraj Singh Mehra** in partial fulfillment of the requirement for the award of degree **Master of Computer Application** in Department of Computer Applications of **Dr. A.P.J. Abdul Kalam Technical University,** is a record of the candidate own work carried out by him under my supervision.

The matter embodied in this Major Project Report is original and has not been submitted for the award of any other degree.

The plagiarism percentage evaluated for the content presented is **10%.**

Supervisor Signature

Ms. Savita Singh

Assistant Professor

Date:

ACKNOWLEDGEMENT

Introducing the report on the MCA project finished during MCA Final Year fills me with incredible happiness. I owe an exceptional obligation of appreciation to **Ms. Savita Singh (Assistant Professor) Department of Computer Applications, ABESEC, Ghaziabad** for his/her constant support and guidance throughout the course of my work. Her sincerity, thoroughness, and perseverance have been a constant source of inspiration for me. It is just her perceptive endeavors that our undertakings have come around.

I also take the opportunity to recognize the contribution of **Prof. (Dr.) Devendra Kumar Head, Department of Computer Applications, ABESEC, Ghaziabad** for his support and assistance during the development of the project.

I also don't want to miss the opportunity to thank the entire department's faculty members for their helpful support and cooperation during the project's development. Last but not least, I want to thank my friends for their help in getting the project done.

Signature of Student

Suraj Singh Mehra

2200320140170

ABSTRACT

The RentEase project is an innovative platform designed to streamline the process of finding student accommodations. In response to the challenges faced by students in securing suitable living arrangements, RentEase employs the MERN (MongoDB, Express.js, React, Node.js) stack to create a comprehensive solution. This project focuses on enhancing transparency, trust, and user experience by providing a user-friendly interface for students to search and connect with accommodation providers.

The objective of RentEase is to offer a centralized hub where students can effortlessly discover, assess, and communicate with accommodation options that align with their preferences. The platform incorporates features such as real-time search, detailed accommodation listings, secure communication channels, and a robust feedback system.

The development and planning of RentEase involve careful consideration of user requirements, technological aspects, and future scalability. The project aims to not only address immediate accommodation needs but also contribute to the creation of a supportive online community for students.

By utilizing the MERN stack, RentEase leverages the strengths of MongoDB for data storage, Express.js for efficient backend logic, React for dynamic and responsive user interfaces, and Node.js for scalable server-side operations. The integration of these technologies enables the creation of a versatile and user-centric platform.

Looking forward, the future scope of RentEase includes international expansion, integration of smart technologies, and continuous innovation through artificial intelligence and predictive analytics. The project envisions becoming a global leader in student accommodation solutions, constantly evolving to meet the dynamic needs of the student community.

In conclusion, RentEase stands as a transformative force in the realm of student accommodations, offering a technologically advanced and user-friendly platform that reshapes the student living experience. The project's success lies in its commitment to transparency, community building, and continuous adaptation to emerging trends and technologies.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

*	Multiplication Symbol
$\neq$	Not Equal
$<$	Smaller than

LIST OF ABBREVIATIONS

AWS	Amazon Web Service
S3	Simple Storage Service
RDS	Relational Database Service
IAM	Identity and Access Management
EC2	Elastic Compute Cloud

Chapter 1: Introduction

1.1 Problem Definition / Statement

Students encounter numerous hurdles when searching for suitable accommodation. Traditional methods such as word-of-mouth referrals, newspaper listings, or generic property websites often prove inefficient and fail to cater specifically to the needs of students. These challenges include:

High Rental Costs: Many students operate within tight budgets, making affordability a crucial factor when searching for accommodation.

Limited Availability: Popular student areas often face high demand, resulting in limited availability of suitable housing options.

Lack of Detailed Information: Students require comprehensive information about available properties, including rent prices, amenities, proximity to educational institutions, and safety features.

Difficulty in Contacting Property Owners: Communicating with property owners or landlords can be challenging, leading to delays or missed opportunities in securing accommodation.

RentEase aims to address these challenges by providing a dedicated platform designed specifically for students, offering an efficient and effective solution to their accommodation needs.

1.2 Objective / Project Objective

The primary objective of RentEase is to develop a user-friendly digital platform that simplifies the process of finding accommodation for students. The project aims to achieve the following objectives:

Developing a Searchable Database: Create a comprehensive database of available properties, making it easier for students to find suitable accommodation options.

Detailed Property Listings: Provide detailed property listings with comprehensive information, including descriptions, photos, pricing, amenities, and proximity to educational institutions.

Advanced Filtering Options: Implement advanced filtering options to enable students to refine their search based on specific criteria such as rent range, location, amenities, and property type.

Facilitating Communication: Enable direct communication between students and property owners through integrated messaging functionality, allowing for seamless coordination and negotiation.

Ensuring User-Friendly Interface: Design and develop a secure, intuitive, and responsive user interface that enhances the overall user experience and encourages engagement with the platform.

1.3 Scope of Project

The scope of the RentEase project encompasses the entire lifecycle of developing and deploying a web-based platform using the MERN stack (MongoDB, Express.js, React.js, Node.js). Key components of the project include:

Design: Defining the architecture, user interface, and user experience design of the platform.

Development: Writing code to implement the features and functionalities outlined in the project objectives.

Deployment: Deploying the platform on a web server and ensuring it is accessible to users.

Testing: Conducting thorough testing to identify and address any issues or bugs in the system.

Future Enhancements: Considering potential future enhancements and additional features that could further improve the platform's functionality and user experience.

1.4 Need

The need for RentEase arises from the significant challenges students face in securing suitable accommodation. Traditional methods are often time-consuming and inefficient, leading to frustration and stress among students. RentEase seeks to fill this gap by providing a dedicated platform that streamlines the search process, saves time, and alleviates the burden associated with finding housing. By addressing these needs, RentEase aims to enhance the overall educational experience for students and contribute to their well-being and success.

Chapter 2: Literature Review

2.1 Existing Solutions

Existing student accommodation platforms have become increasingly prevalent in recent years, aiming to streamline the process of finding housing for students. This section of the literature review involves a detailed examination of these platforms, including:

Review of Existing Platforms: An overview of popular student accommodation platforms, such as Airbnb, Zillow, and Student.com, among others. This review will encompass a comprehensive analysis of their features, functionalities, and user interfaces.

Analysis of Strengths and Weaknesses: Each platform will be scrutinized to identify its strengths and weaknesses. Strengths may include user-friendly interfaces, extensive property listings, or innovative features like virtual tours. Conversely, weaknesses may include limited search capabilities, lack of transparency in pricing, or inadequate communication tools.

2.2 Technological Frameworks

Technological frameworks play a crucial role in the development of student accommodation platforms, shaping their functionality, scalability, and performance. This section will provide an overview of the technologies commonly used in building similar platforms, with a focus on the advantages of the MERN stack, which consists of:

MongoDB: A NoSQL database used for storing property listings, user data, and other relevant information. Its flexibility and scalability make it well-suited for handling large volumes of data.

Express.js: A lightweight and flexible Node.js framework for building web applications and APIs. Express.js simplifies the process of routing, middleware integration, and handling HTTP requests, enhancing the efficiency of backend development.

React.js: A JavaScript library for building user interfaces. React.js enables the creation of interactive and dynamic frontend components, facilitating a seamless and intuitive user experience.

Node.js: A server-side JavaScript runtime environment that allows developers to build scalable and high-performance web applications. Node.js enables asynchronous, event-driven programming, making it ideal for handling concurrent requests and real-time interactions.

The advantages of the MERN stack, such as its versatility, performance, and ease of development, will be highlighted, along with examples of successful implementations in the context of student accommodation platforms.

2.3 Identified Gaps

Despite the proliferation of student accommodation platforms, there are several notable gaps and limitations in existing solutions that hinder the user experience. This section will focus on:

Identification of Gaps: Common shortcomings observed in existing platforms, such as limited search functionality, lack of verified listings, or inadequate communication channels between users and property owners.

Explanation of RentEase's Approach: RentEase aims to address these gaps and provide a better user experience by implementing innovative features and enhancements. For example, RentEase may introduce advanced search filters, incorporate user verification mechanisms, or develop a robust messaging system to facilitate seamless communication between students and property owners.

By identifying these gaps and proposing solutions, RentEase seeks to differentiate itself from existing platforms and provide a more comprehensive and user-centric solution for students seeking accommodation.

Chapter 3: Feasibility Study

3.1 Purpose

The feasibility study serves as a critical assessment tool to determine the viability and potential success of RentEase. By evaluating various aspects of the project, including technical, economic, and operational factors, the study aims to provide insights into the feasibility of developing and implementing the platform. The key purposes of the feasibility study are as follows:

Understanding Objectives: The study seeks to gain a comprehensive understanding of RentEase's objectives, including its intended functionalities, target users, and expected outcomes. By clarifying the project's goals, the feasibility study sets the foundation for subsequent assessments.

Assessing Potential Challenges: Identifying potential challenges and obstacles that may arise during the development and deployment of RentEase is crucial. This includes technical complexities, market competition, regulatory compliance issues, and resource constraints. By anticipating these challenges early on, the feasibility study enables stakeholders to devise strategies to mitigate risks effectively.

Evaluating Expected Outcomes: The feasibility study aims to evaluate the expected outcomes and benefits of RentEase upon its successful implementation. This involves assessing the potential impact on users, such as improved access to housing options, enhanced user experience, and increased efficiency in the accommodation search process.

3.2 Scope

The feasibility study encompasses a comprehensive assessment of various factors to determine the feasibility of RentEase. It covers the following key areas:

Technical Feasibility: This aspect evaluates whether the MERN stack, consisting of MongoDB, Express.js, React.js, and Node.js, is suitable for developing the RentEase platform. It assesses factors such as the availability of relevant tools and frameworks, scalability, compatibility with desired functionalities, and the expertise of the development team. Additionally, the study examines potential technical challenges and explores strategies to address them effectively.

Economic Feasibility: Economic feasibility involves estimating the costs associated with developing, deploying, and maintaining the RentEase platform. This includes expenses related to software development, infrastructure setup, marketing, staffing, and ongoing operational costs. The study also considers potential revenue streams, such as subscription fees, advertisements, or transaction commissions, to determine the platform's financial viability. By comparing the projected costs and benefits, stakeholders can assess the economic feasibility of RentEase and make informed decisions regarding investment.

Operational Feasibility: Operational feasibility focuses on evaluating whether RentEase can be successfully operated and maintained in the long term. It considers factors such as user acceptance, usability, support requirements, regulatory compliance, and scalability. The study assesses the platform's ability to meet user needs effectively, handle increasing user loads, and adapt to evolving market conditions. By addressing operational considerations upfront, stakeholders can ensure that RentEase remains sustainable and competitive in the long run.

Chapter 4: System Requirements

4.1 Functional Requirements

Functional requirements define the specific functionalities and features that the RentEase platform must possess to meet the needs of its users. These requirements are essential for the platform to perform its core functions effectively. The functional requirements for RentEase are as follows:

User Authentication: Users should be able to register for an account, log in securely, and manage their profiles. This functionality ensures that user data is protected and that each user has a personalized experience on the platform.

Property Listings: Property owners should have the ability to add new property listings, update existing ones, and remove listings when necessary. This feature allows property owners to showcase their available accommodations to potential tenants.

Search and Filter: Users should be able to search for properties using various filters such as location, price range, amenities, and property type. This functionality enables users to find accommodations that meet their specific criteria quickly and efficiently.

Reviews and Ratings: Users should be able to view and submit reviews and ratings for properties they have interacted with. This feature allows users to share their experiences with others and helps future users make informed decisions about potential accommodations.

Messaging System: RentEase should include a messaging system that facilitates communication between students seeking accommodation and property owners. This feature enables users to ask questions, negotiate rental terms, and finalize rental agreements directly through the platform.

4.2 Non-Functional Requirements

Non-functional requirements specify the attributes of the system that are not related to its specific functionalities but are crucial for ensuring its overall effectiveness, usability, and performance. The non-functional requirements for RentEase are as follows:

Security: The platform must implement robust security measures to protect user data, prevent unauthorized access, and ensure privacy and confidentiality.

Performance: RentEase should provide fast load times and a responsive user interface to enhance the user experience and minimize waiting times.

Scalability: The platform should be designed to handle a growing number of users and property listings without compromising performance or usability.

Usability: RentEase should offer an intuitive and easy-to-navigate interface for all users, regardless of their technical expertise or familiarity with the platform.

4.3 Hardware Requirements

The hardware requirements specify the minimum hardware specifications that the server hosting the RentEase platform should meet to ensure optimal performance. These requirements include:

Minimum 8GB RAM to ensure sufficient memory for handling user requests and database operations.

500GB Hard Disk to accommodate the storage of property listings, user data, and other platform-related files.

Intel Core i5 Processor or equivalent to provide adequate processing power for handling concurrent user requests and maintaining system responsiveness.

4.4 Software Requirements

The software requirements outline the necessary software components and technologies that the RentEase platform relies on for its development and operation. These requirements include:

Operating System: RentEase should be compatible with popular operating systems such as Windows, macOS, and Linux to ensure broad accessibility for users.

Backend: The backend of the platform should be built using Node.js and Express.js, which provide a robust and scalable framework for developing server-side applications.

Frontend: The frontend of RentEase should be developed using React.js, a powerful JavaScript library for building dynamic user interfaces.

Database: MongoDB, a NoSQL database, should be used to store property listings, user profiles, and other platform-related data due to its flexibility and scalability.

Other Tools: Redux should be used for state management, while JWT (JSON Web Tokens) should be employed for user authentication to ensure secure and reliable user authentication and authorization mechanisms.

4.5 Use Cases

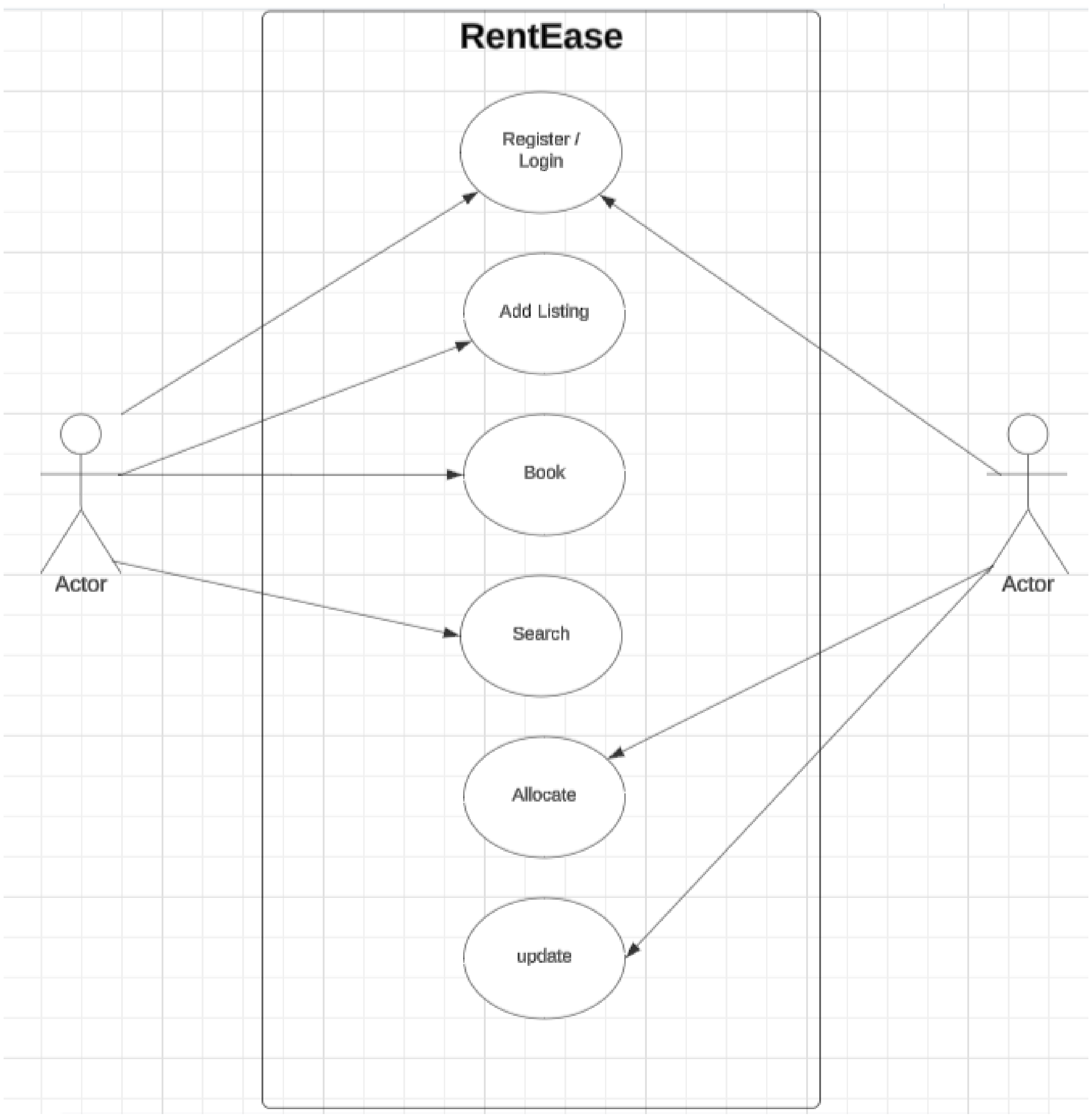
Use cases describe the specific interactions and workflows that users engage in when using the RentEase platform. These use cases provide detailed step-by-step scenarios of how users interact with the platform and the expected outcomes of each interaction. The use cases for RentEase include:

User Registration and Login: Describes the steps a user takes to register for an account and log in to the platform securely.

Property Search and Filter: Outlines how users search for properties and apply filters to narrow down search results based on their preferences.

Property Listing Management: Details how property owners add, update, and manage their property listings on the platform.

Messaging: Describes the process of sending and receiving messages between students seeking accommodation and property owners to facilitate communication and negotiation.

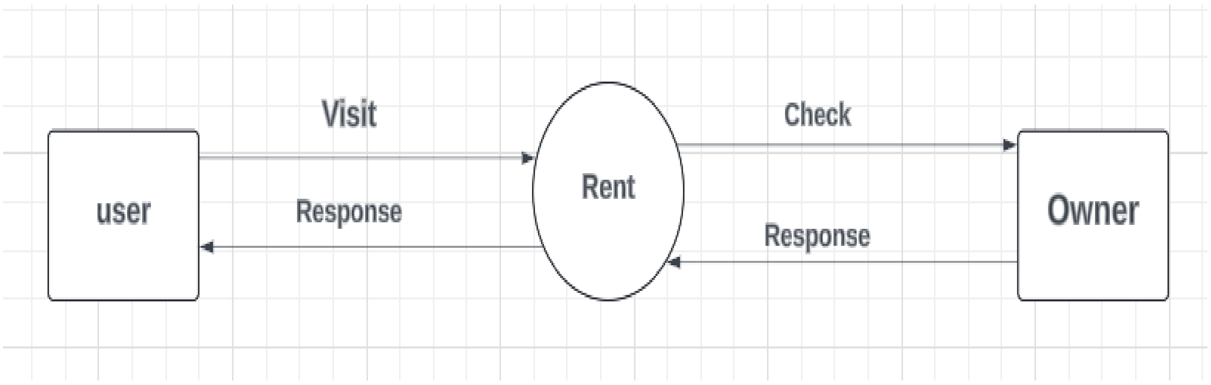


Chapter 5: System Design

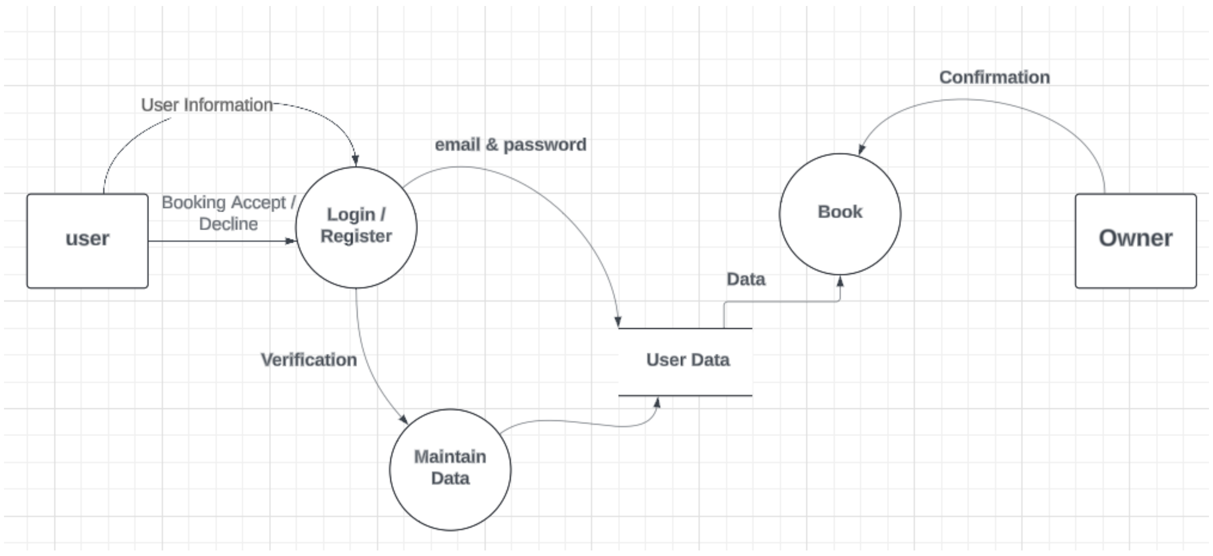
5.1 Data Flow Diagram (DFD)

Data Flow Diagrams (DFDs) provide a graphical representation of the flow of data within a system. They illustrate how data moves through different processes and entities within the system. The DFD for RentEase is structured into multiple levels:

Level 0 DFD: This level represents the overall system as a single process with its major inputs and outputs. It provides a high-level overview of the entire system and its interactions with external entities such as users, property owners, and the database.

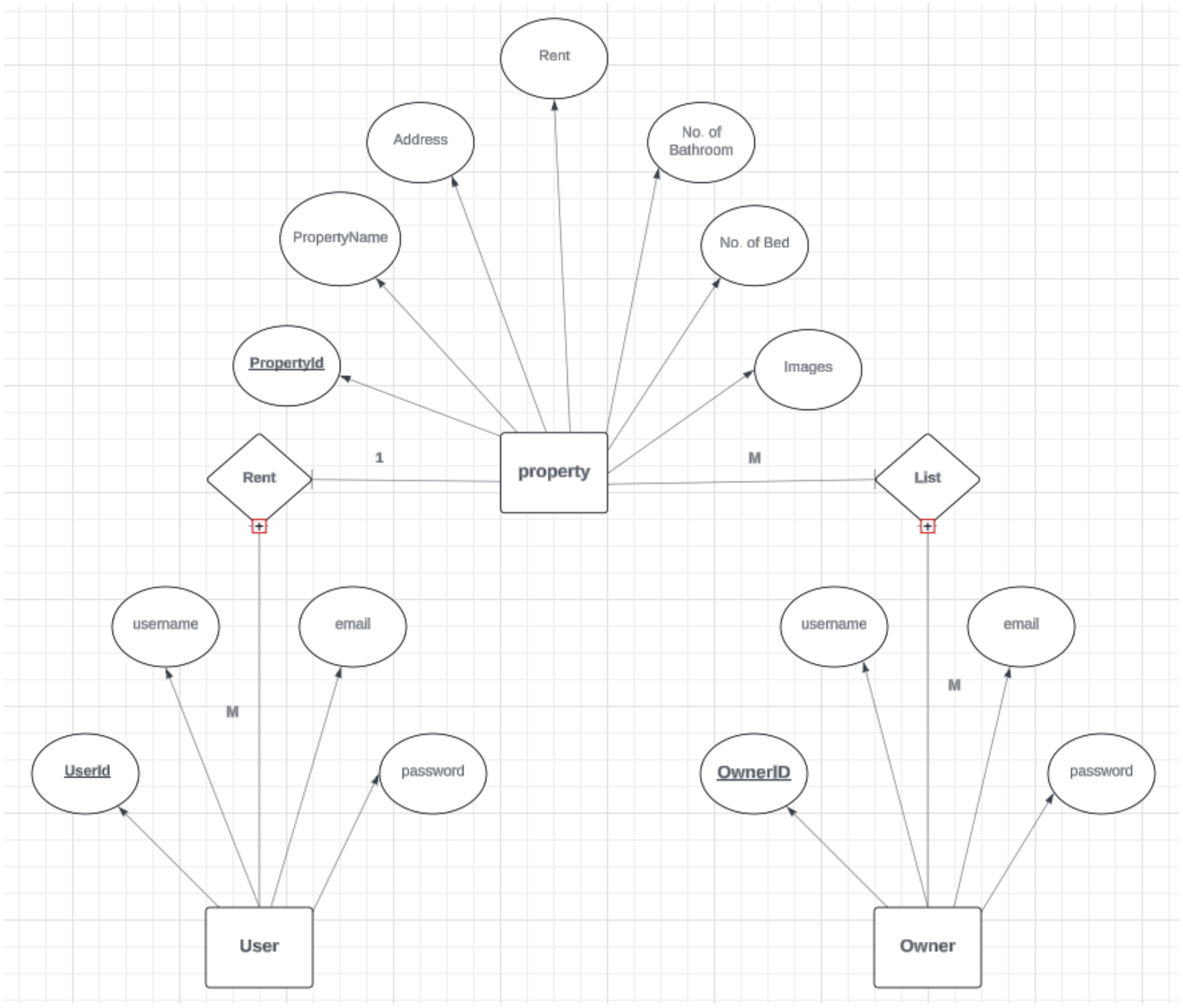


Level 1 DFD: The Level 1 DFD breaks down the main process into sub-processes, providing more detail about the data flow within the system. It identifies individual processes such as user registration, property search, messaging, and reviews, and illustrates how data moves between them.



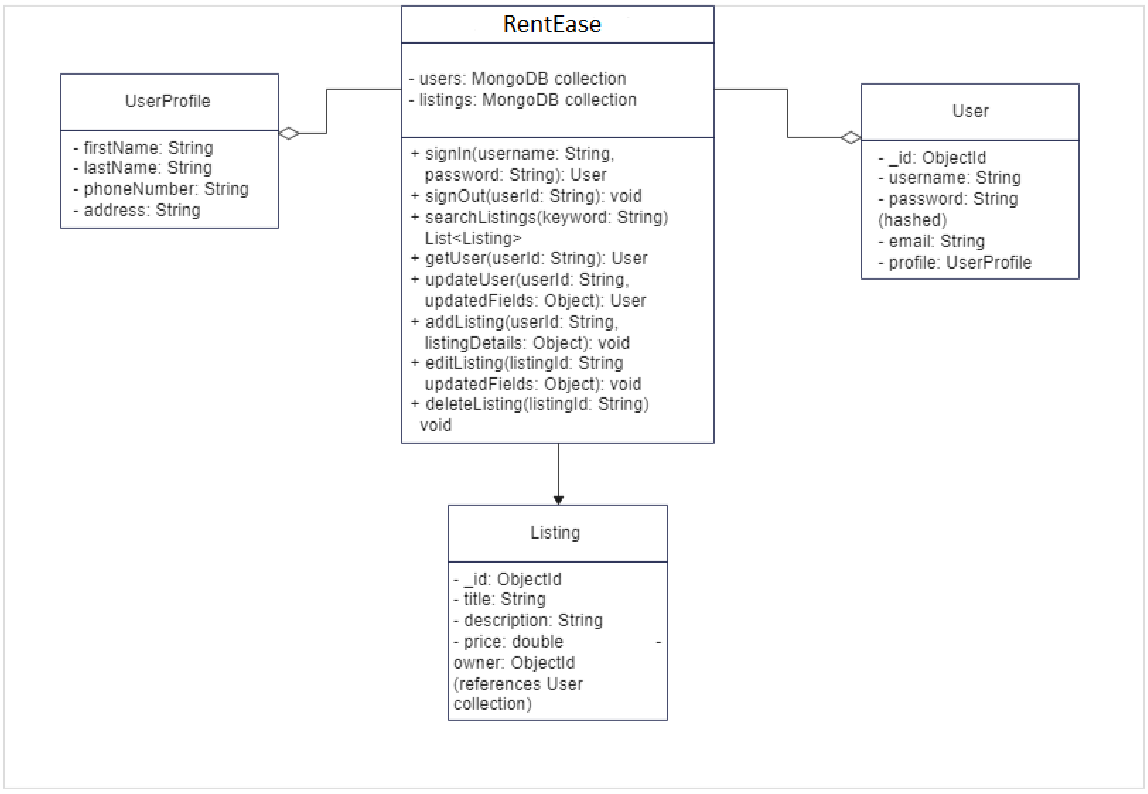
5.2 Entity-Relationship Diagrams (ERD)

Entity-Relationship Diagrams (ERDs) visualize the relationships between entities in a system. For RentEase, the ERD illustrates how entities such as Users, Properties, Reviews, and Messages are related to each other. It identifies the attributes of each entity and the relationships between them, such as one-to-one, one-to-many, or many-to-many relationships. The ERD provides a comprehensive view of the system's data model, helping to design the database schema and ensure data integrity.



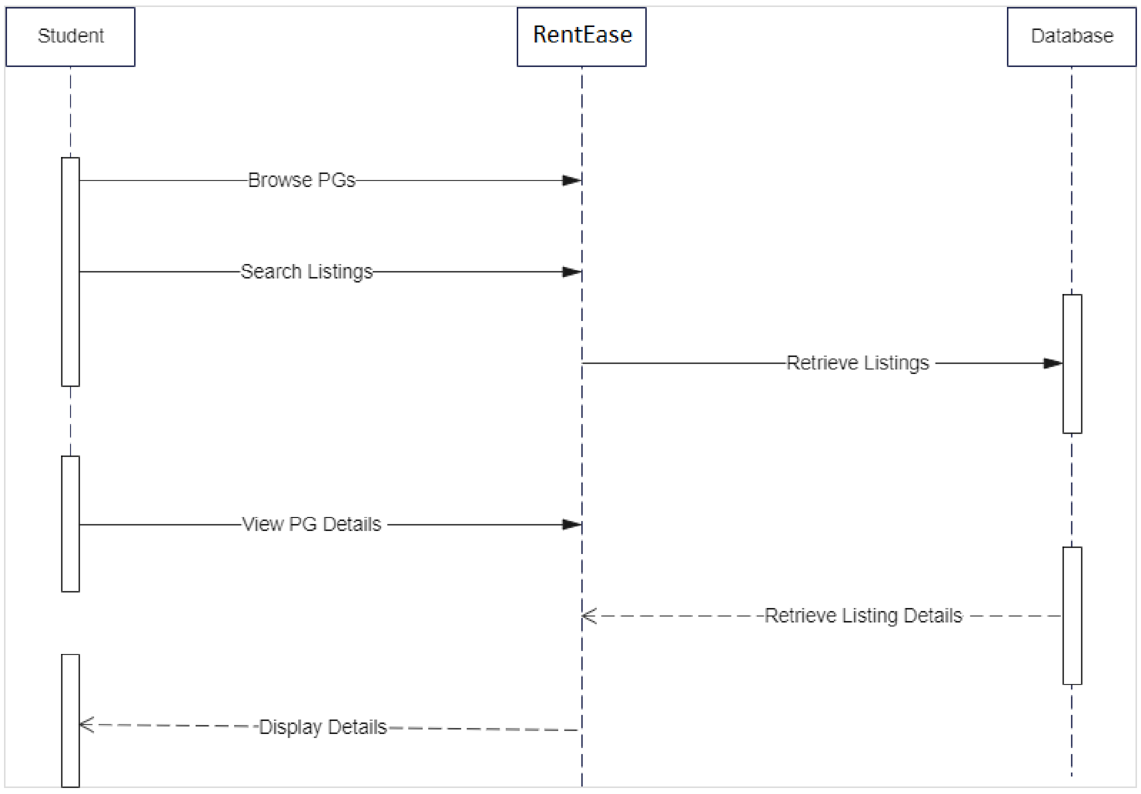
5.3 Class Diagrams

Class Diagrams are part of the Unified Modeling Language (UML) and depict the structure of the system's classes, their attributes, methods, and relationships. In RentEase, the Class Diagram represents the various classes and their interactions within the system. This includes classes such as User, Property, Review, Message, etc., along with their attributes and methods. The relationships between classes, such as associations, aggregations, and inheritances, are also depicted in the diagram, providing a blueprint for the system's code implementation.


RentEase
- users: MongoDB collection
- listings: MongoDB collection
+ signIn(username: String, password: String): User
+ signOut(userId: String): void
+ searchListings(keyword: String) List<Listing>
+ getUser(userId: String): User
+ updateUser(userId: String, updatedFields: Object): User
+ addListing(userId: String, listingDetails: Object): void
+ editListing(listingId: String updatedFields: Object): void
+ deleteListing(listingId: String) void
UserProfile
- firstName: String
- lastName: String
- phoneNumber: String
- address: String
User
- _id: ObjectId
- username: String
- password: String (hashed)
- email: String
- profile: UserProfile
Listing
- _id: ObjectId
- title: String
- description: String
- price: double
- owner: ObjectId (references User collection)

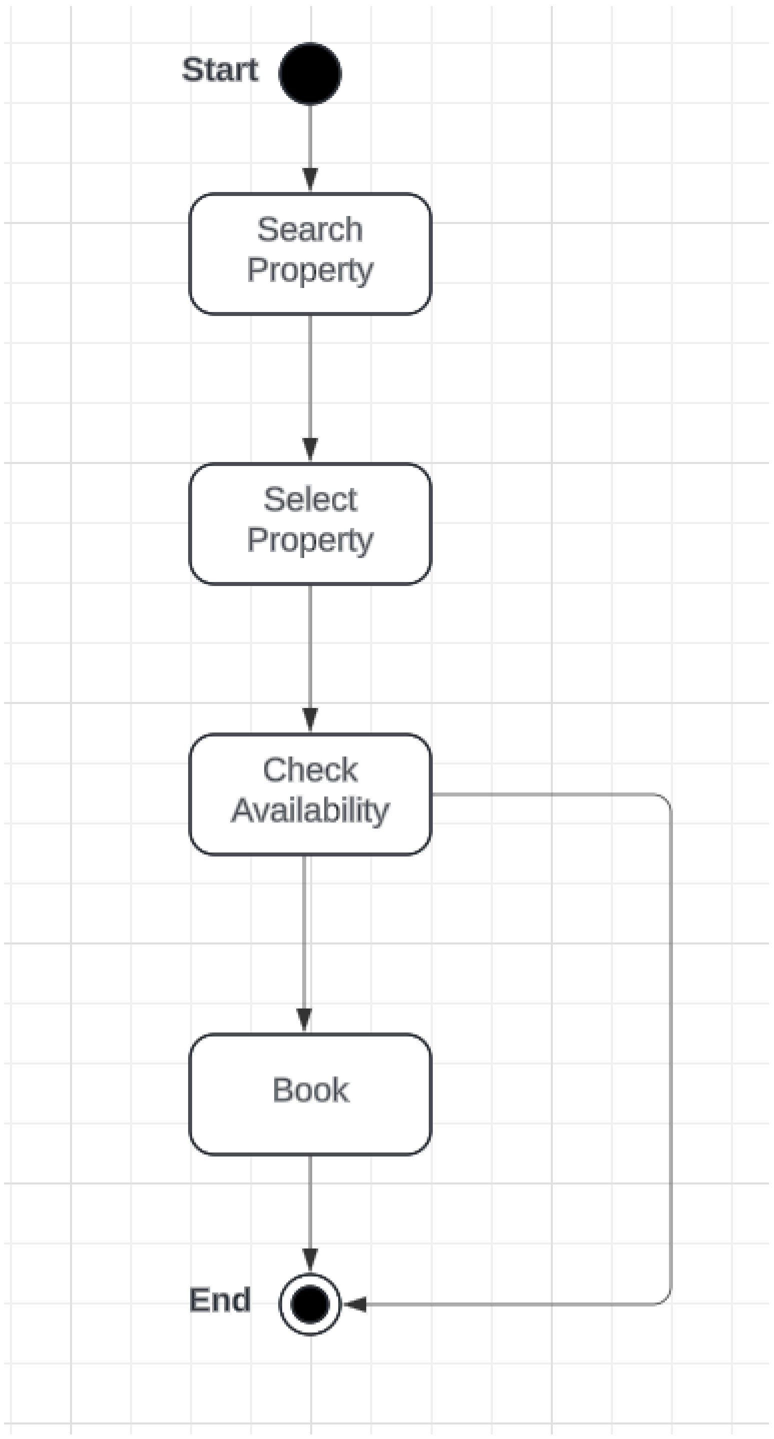
5.4 Sequence Diagrams

Sequence Diagrams illustrate the sequence of interactions between objects in a particular scenario or use case. For RentEase, Sequence Diagrams depict the interactions between different components of the system, such as users, controllers, and the database, during key processes such as user registration, property search, and messaging. They show the chronological order of messages exchanged between objects, helping to understand how the system behaves in response to user actions and events.



5.5 Activity Diagram

Activity Diagrams visualize the flow of activities or processes within a system. In RentEase, the Activity Diagram illustrates the flow of activities for various processes, such as searching for properties, booking a property, and communicating with property owners. It shows the sequence of actions taken by users and the system in response to user inputs, providing a high-level overview of the system's functionality and workflow.



Chapter 6: GUI / Coding

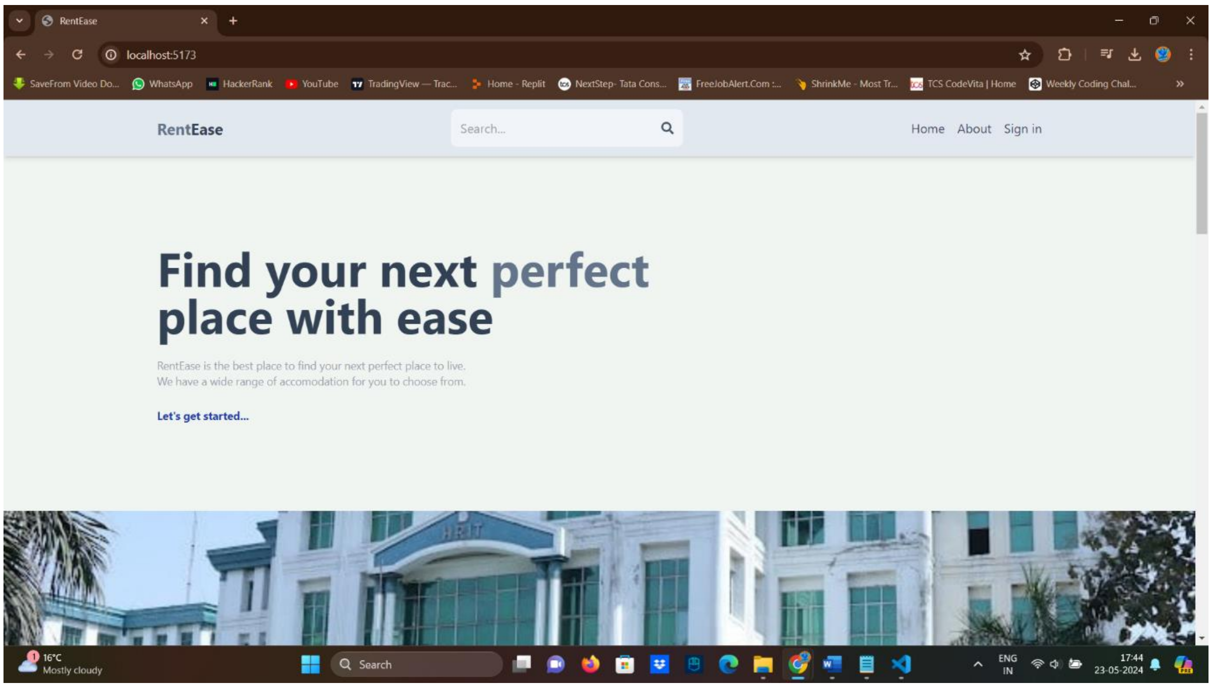
6.1 Graphical User Interface (GUI)

Graphical User Interface (GUI) is crucial for RentEase as it directly impacts user experience. It encompasses various screens and interfaces that users interact with while using the platform. Below are the key GUI components of RentEase:

Homepage: The homepage serves as the entry point for users and typically features a search bar prominently displayed. It also showcases featured property listings to attract user attention.

Search Results: This page displays search results based on user input and selected filters. It presents property listings in a structured manner, allowing users to easily browse through available options.

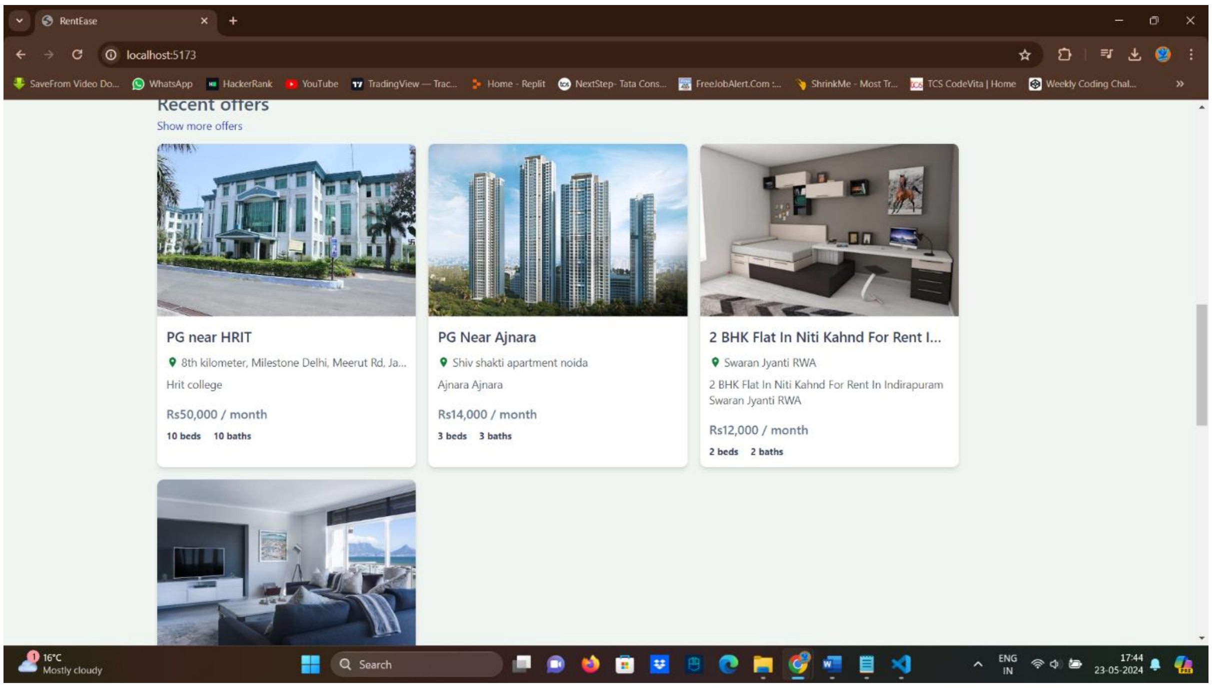
Property Details: When users click on a specific property listing, they are directed to a detailed page that provides comprehensive information about the property. This includes images, descriptions, amenities, pricing, and user reviews.



Find your next perfect place with ease

RentEase is the best place to find your next perfect place to live.
We have a wide range of accomodation for you to choose from.

Let's get started...



Recent offers

Show more offers

PG near HRIT

8th kilometer, Milestone Delhi, Meerut Rd, Ja...

Hrit college

Rs50,000 / month

10 beds 10 baths

PG Near Ajnara

Shiv shakti apartment noida

Ajnara Ajnara

Rs14,000 / month

3 beds 3 baths

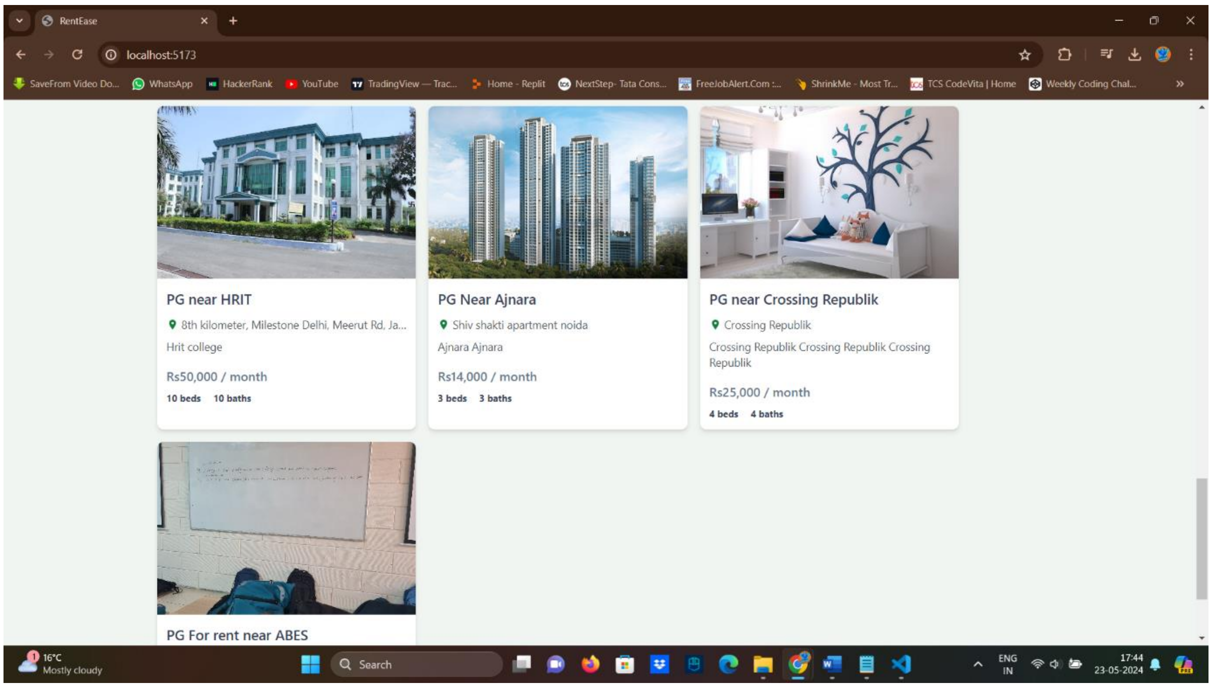
2 BHK Flat In Niti Kahnd For Rent I...

Swaran Jyanti RWA

2 BHK Flat In Niti Kahnd For Rent In Indirapuram Swaran Jyanti RWA

Rs12,000 / month

2 beds 2 baths



PG near HRIT

8th kilometer, Milestone Delhi, Meerut Rd, Ja...

Hrit college

Rs50,000 / month

10 beds 10 baths

PG Near Ajnara

Shiv shakti apartment noida

Ajnara Ajnara

Rs14,000 / month

3 beds 3 baths

PG near Crossing Republik

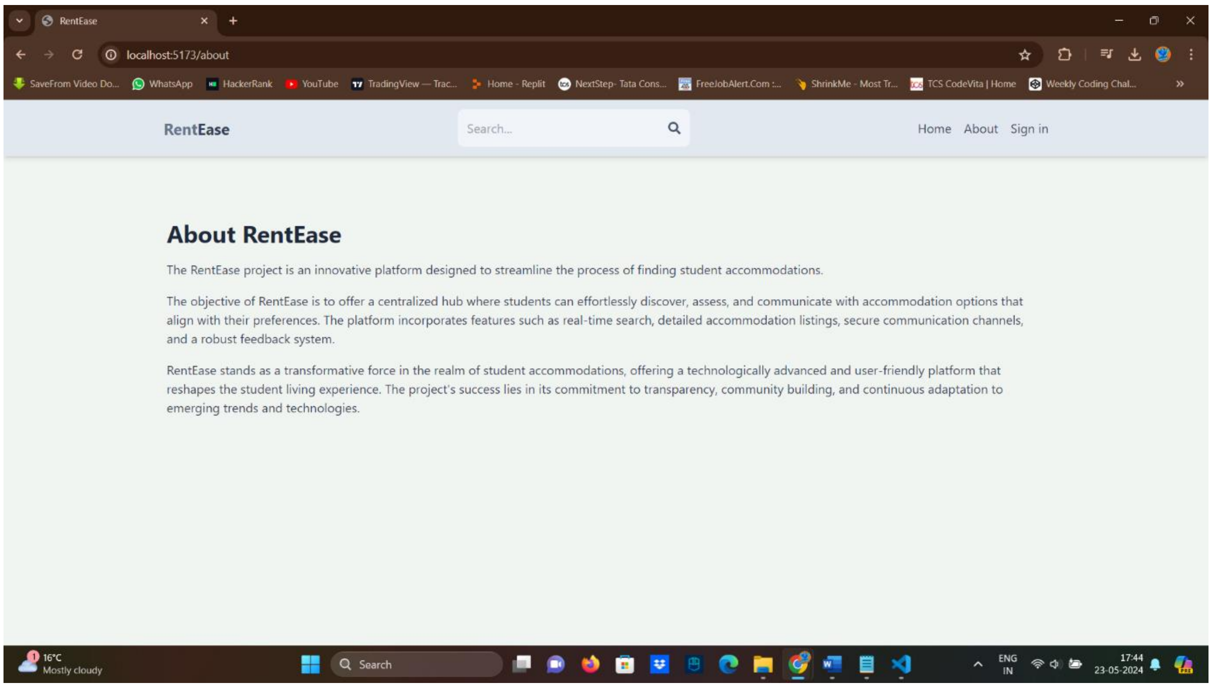
Crossing Republik

Crossing Republik Crossing Republik Crossing Republik

Rs25,000 / month

4 beds 4 baths

PG For rent near ABES

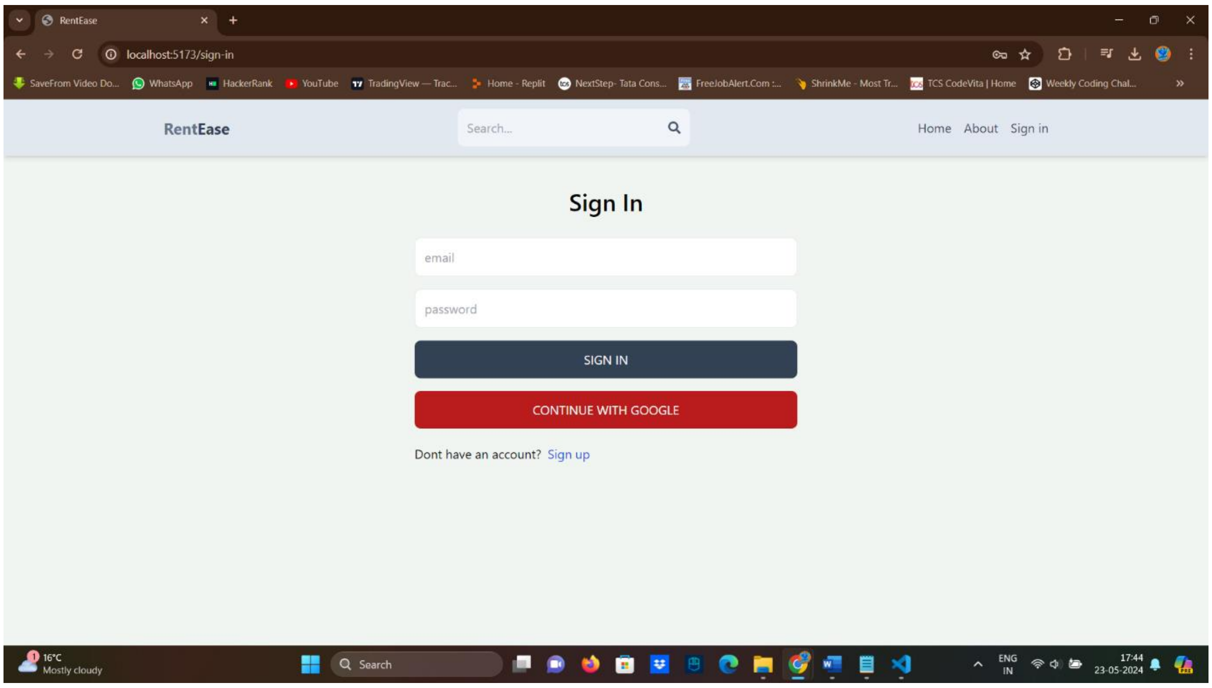


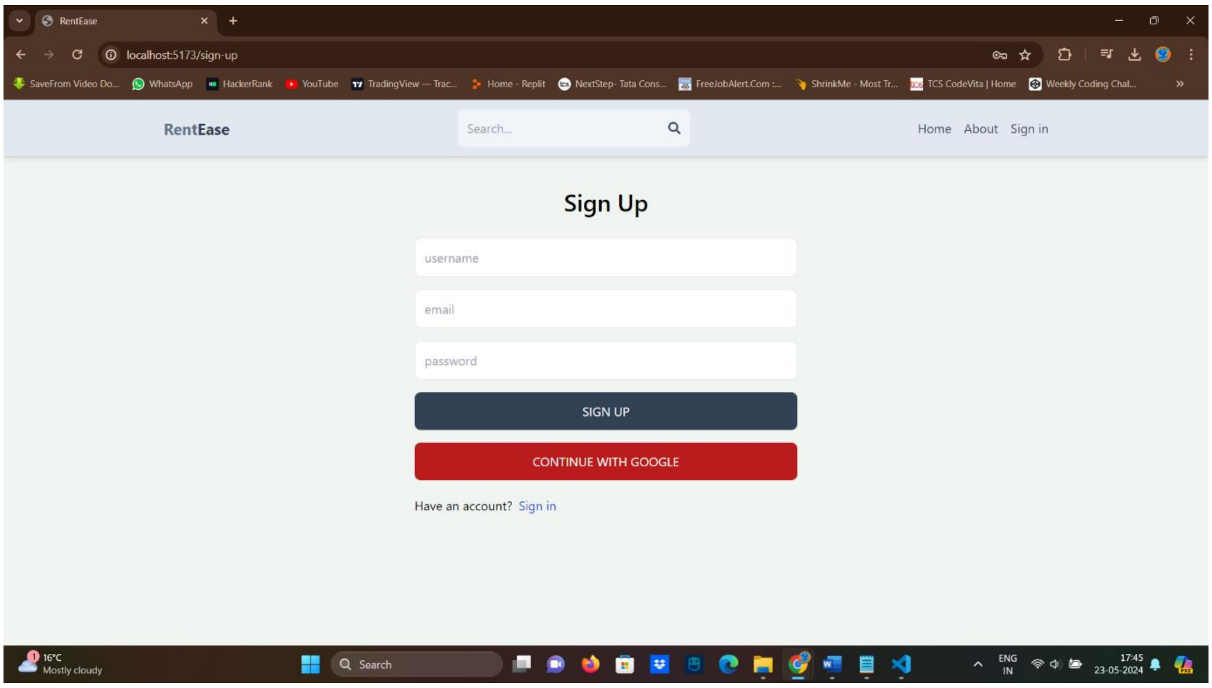
About RentEase

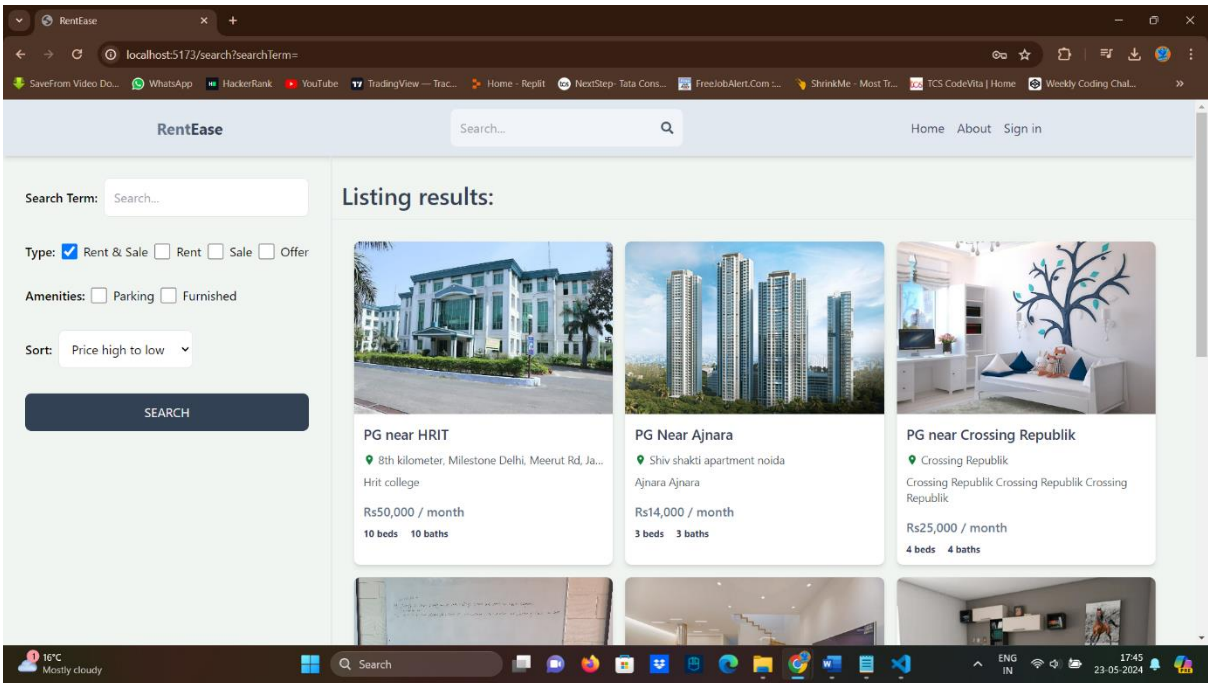
The RentEase project is an innovative platform designed to streamline the process of finding student accommodations.

The objective of RentEase is to offer a centralized hub where students can effortlessly discover, assess, and communicate with accommodation options that align with their preferences. The platform incorporates features such as real-time search, detailed accommodation listings, secure communication channels, and a robust feedback system.

RentEase stands as a transformative force in the realm of student accommodations, offering a technologically advanced and user-friendly platform that reshapes the student living experience. The project's success lies in its commitment to transparency, community building, and continuous adaptation to emerging trends and technologies.







Search Term: Search...

Type: [✓ Rent & Sale] Rent Sale Offer

Amenities: Parking Furnished

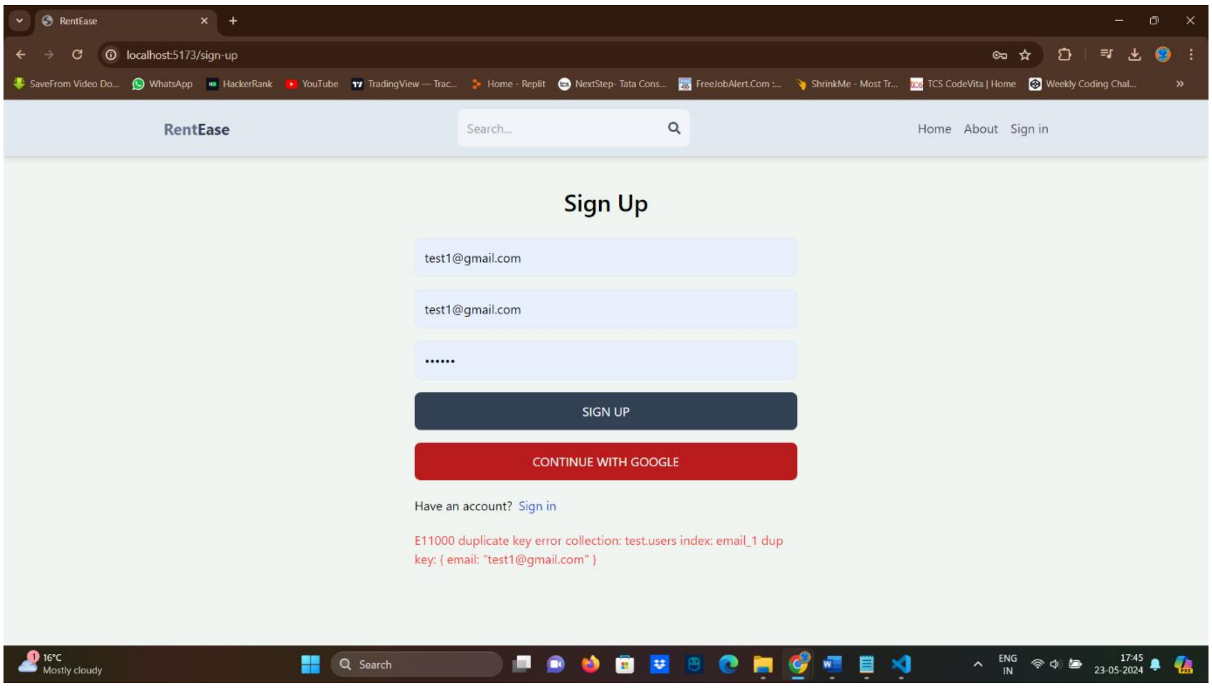
Sort: Price high to low

Listing results:

PG near HRIT
8th kilometer, Milestone Delhi, Meerut Rd, Ja...
Hrit college
Rs50,000 / month
10 beds 10 baths

PG Near Ajnara
Shiv shakti apartment noida
Ajnara Ajnara
Rs14,000 / month
3 beds 3 baths

PG near Crossing Republik
Crossing Republik
Crossing Republik Crossing Republik Crossing Republik
Rs25,000 / month
4 beds 4 baths



localhost:5173/sign-up

RentEase

Home About Sign in

Sign Up

test1@gmail.com

test1@gmail.com

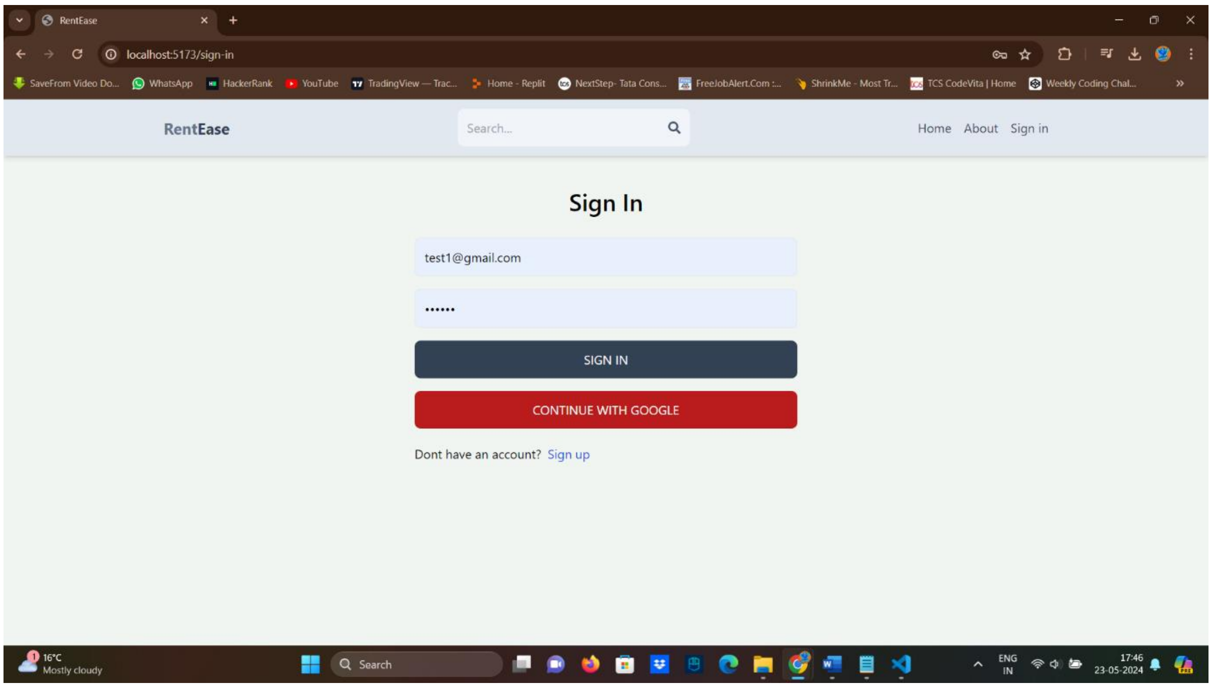
••••••

SIGN UP

CONTINUE WITH GOOGLE

Have an account? Sign in

E11000 duplicate key error collection: test.users index: email_1 dup key: { email: "test1@gmail.com" }



localhost:5173/sign-in

RentEase

Home About Sign in

Sign In

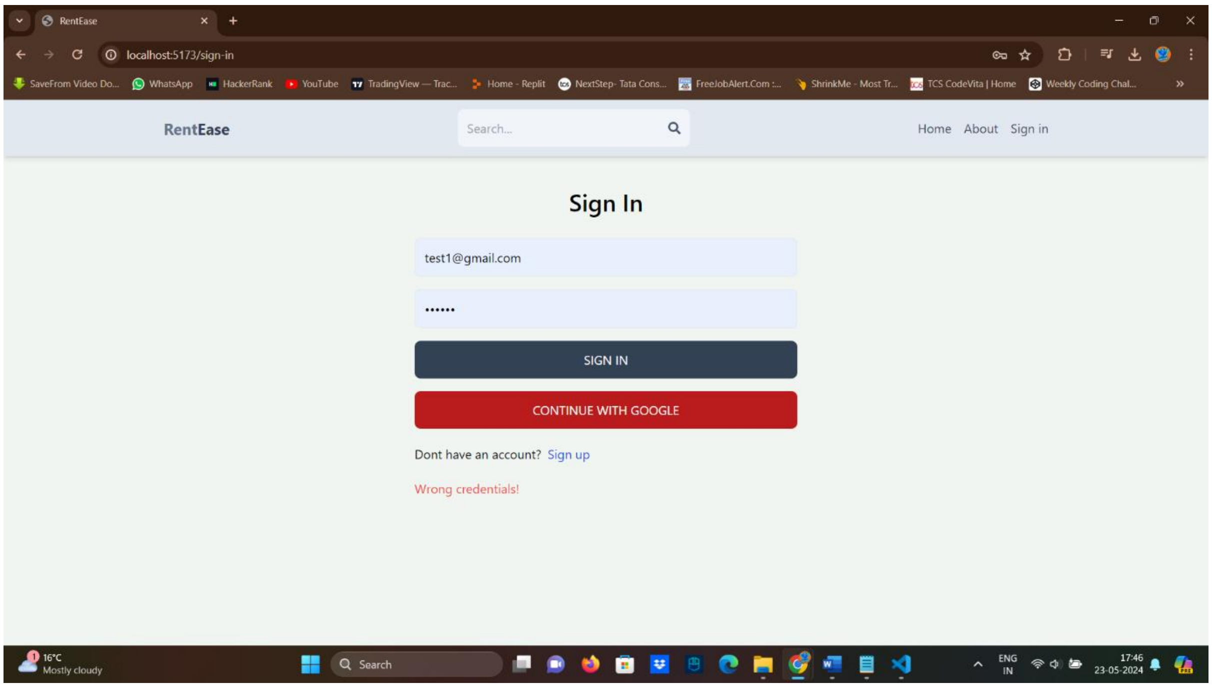
test1@gmail.com

••••••

SIGN IN

CONTINUE WITH GOOGLE

Dont have an account? Sign up



Sign In

test1@gmail.com

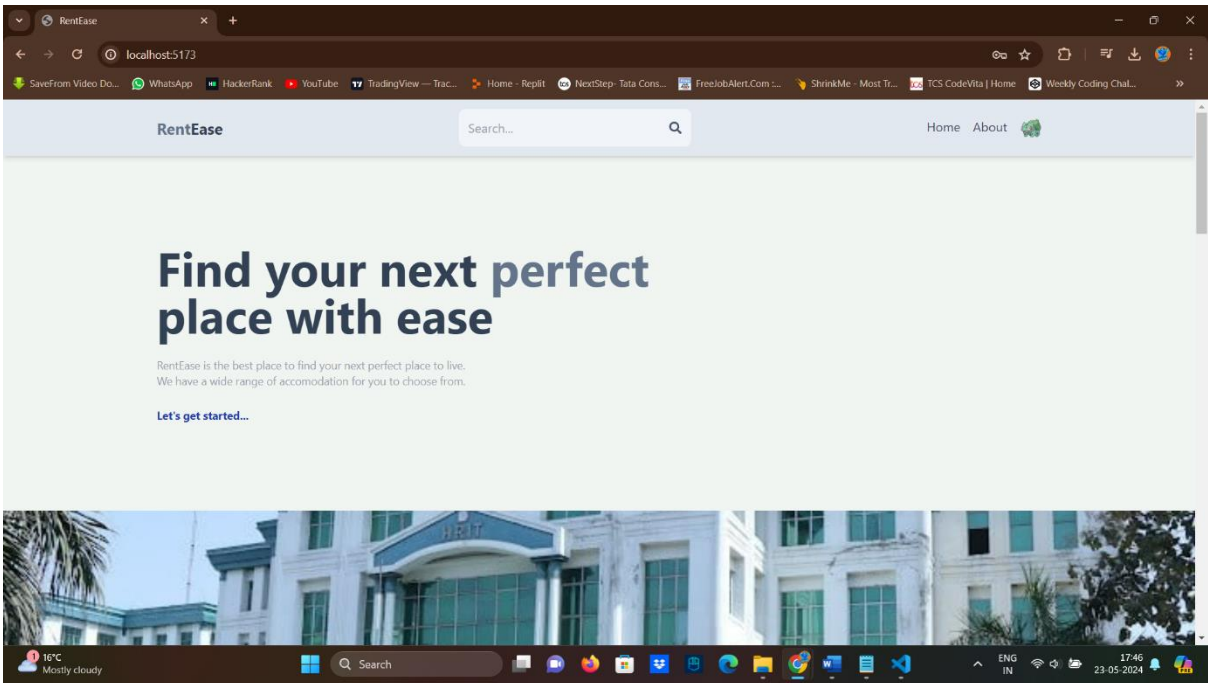
••••••

SIGN IN

CONTINUE WITH GOOGLE

Dont have an account? Sign up

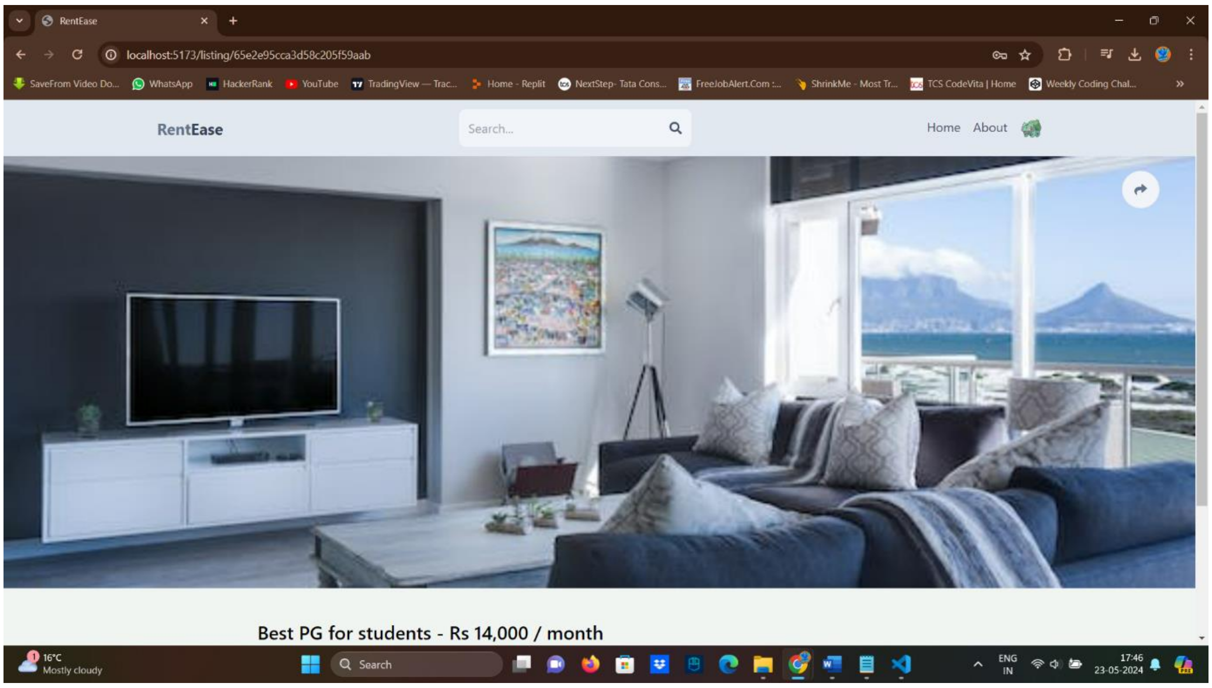
Wrong credentials!



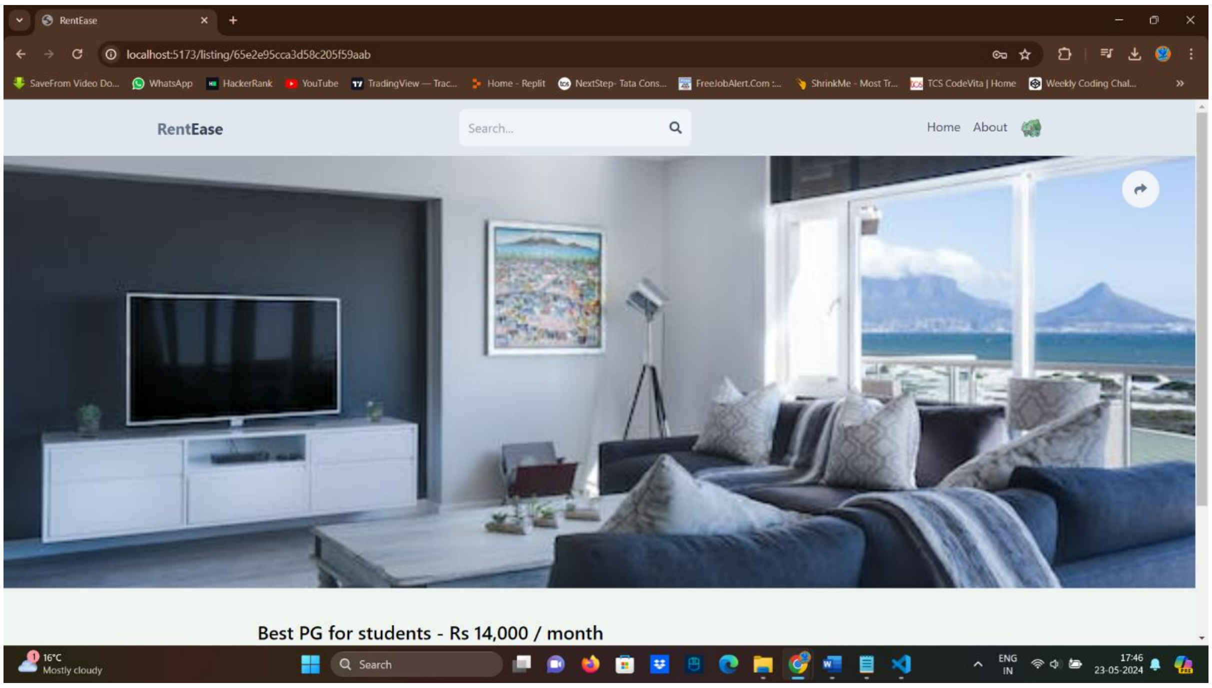
Find your next perfect place with ease

RentEase is the best place to find your next perfect place to live.
We have a wide range of accomodation for you to choose from.

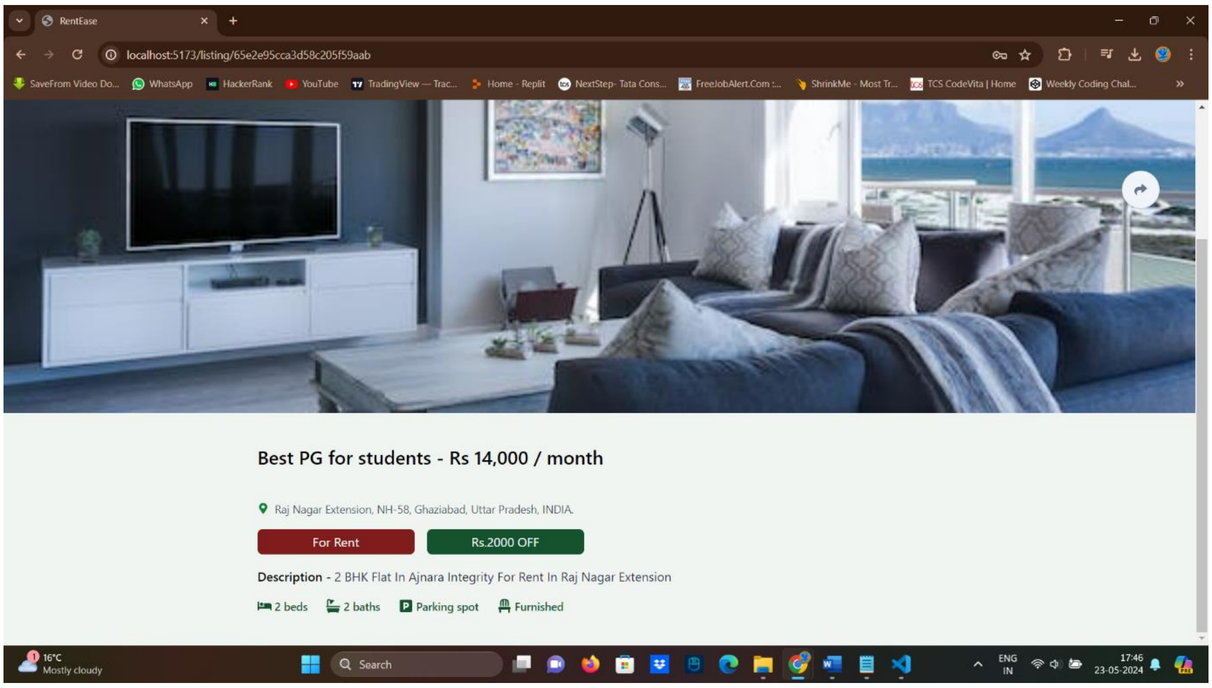
Let's get started...



Best PG for students - Rs 14,000 / month



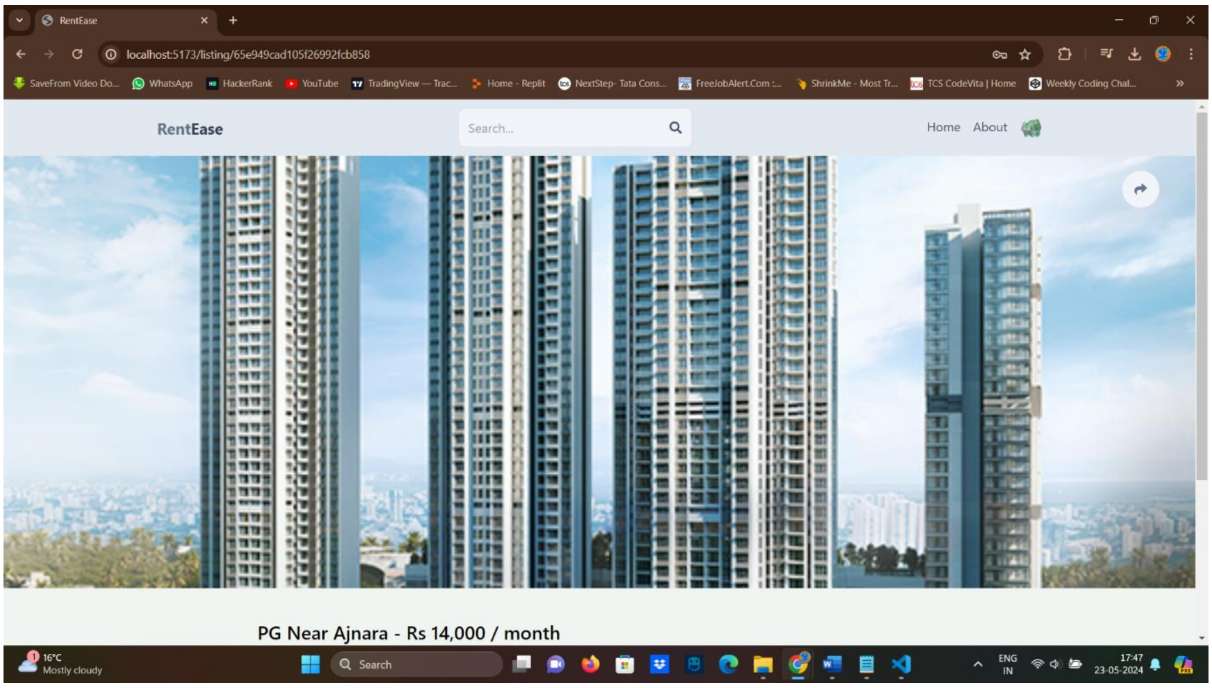
Best PG for students - Rs 14,000 / month



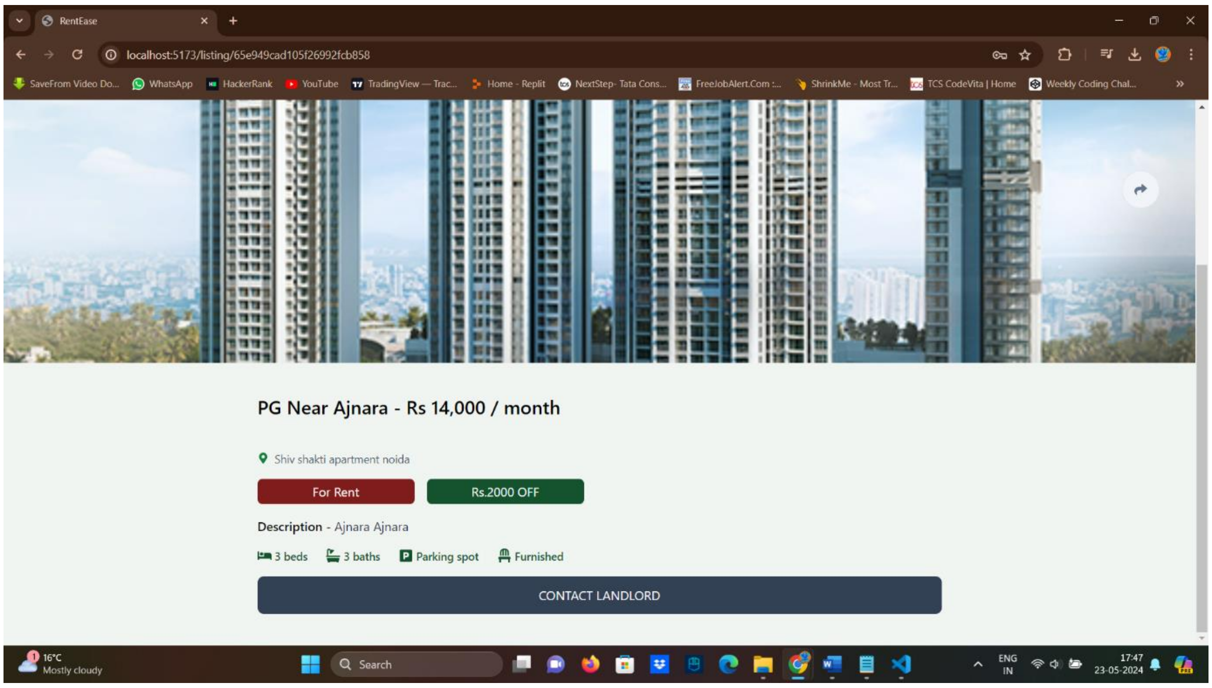
Best PG for students - Rs 14,000 / month

Raj Nagar Extension, NH-58, Ghaziabad, Uttar Pradesh, INDIA.

Description - 2 BHK Flat In Ajnara Integrity For Rent In Raj Nagar Extension



PG Near Ajnara - Rs 14,000 / month



PG Near Ajnara - Rs 14,000 / month

Shiv shakti apartment noida

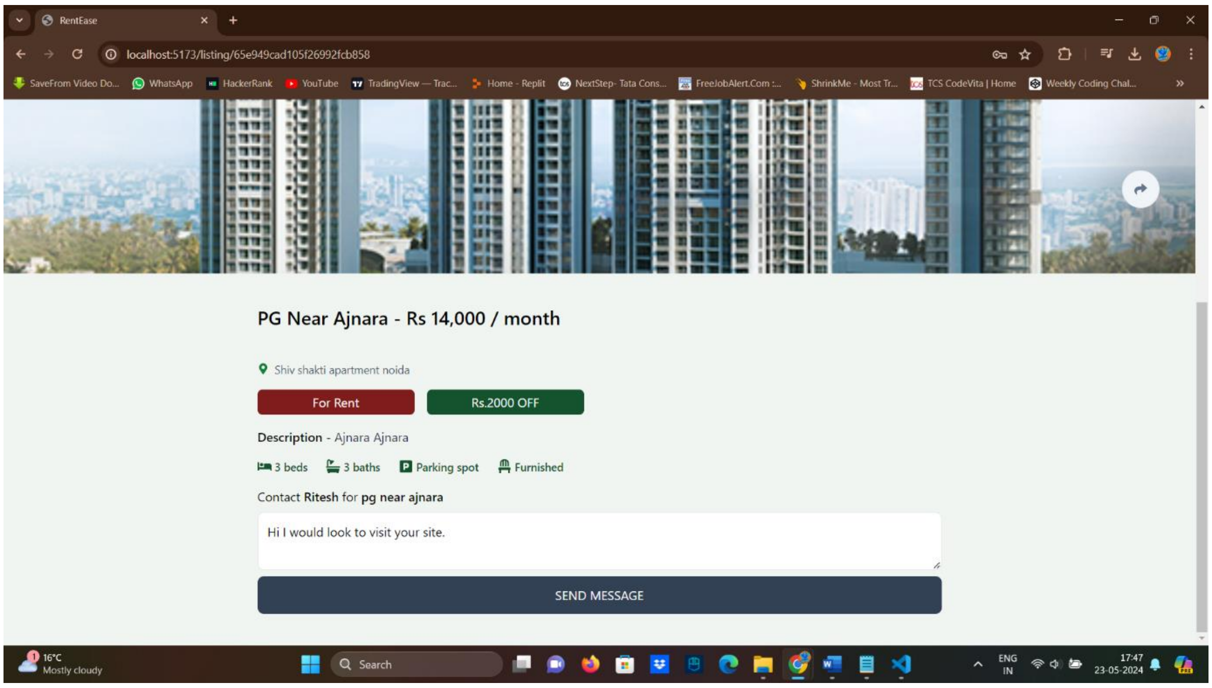
For Rent

Rs.2000 OFF

Description - Ajnara Ajnara

3 beds 3 baths Parking spot Furnished

CONTACT LANDLORD



PG Near Ajnara - Rs 14,000 / month

Shiv shakti apartment noida

For Rent

Rs.2000 OFF

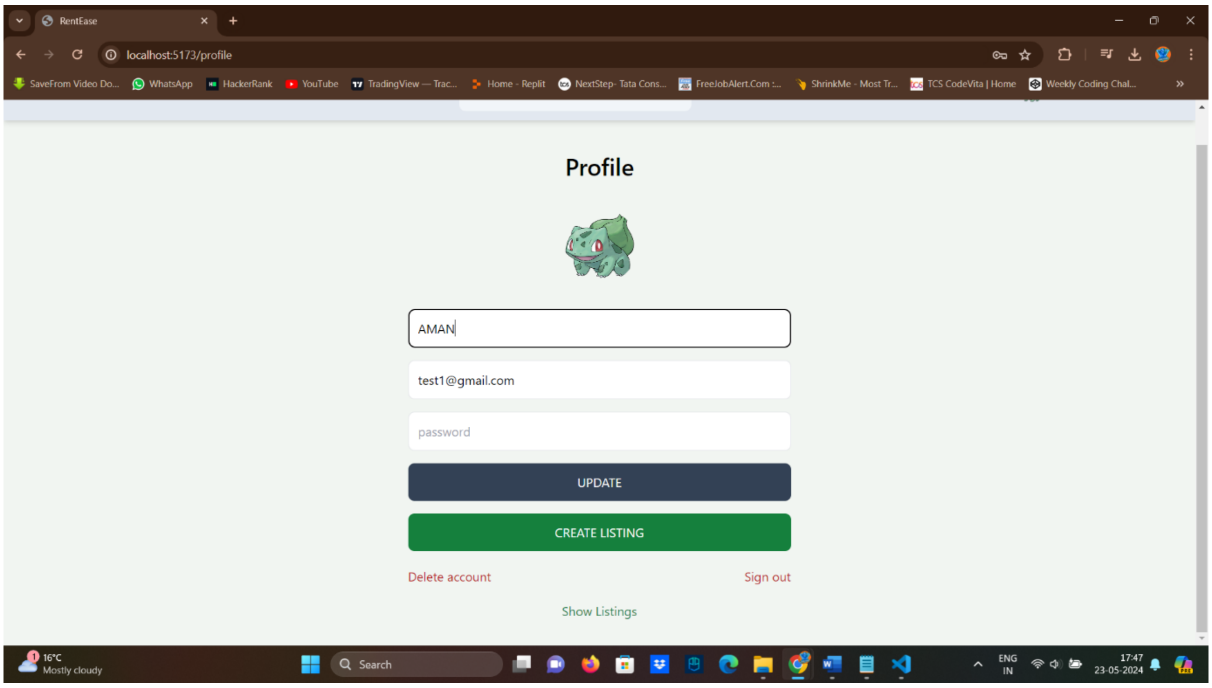
Description - Ajnara Ajnara

3 beds 3 baths Parking spot Furnished

Contact **Ritesh** for **pg near ajnara**

Hi I would look to visit your site.

SEND MESSAGE



Profile

AMAN

test1@gmail.com

password

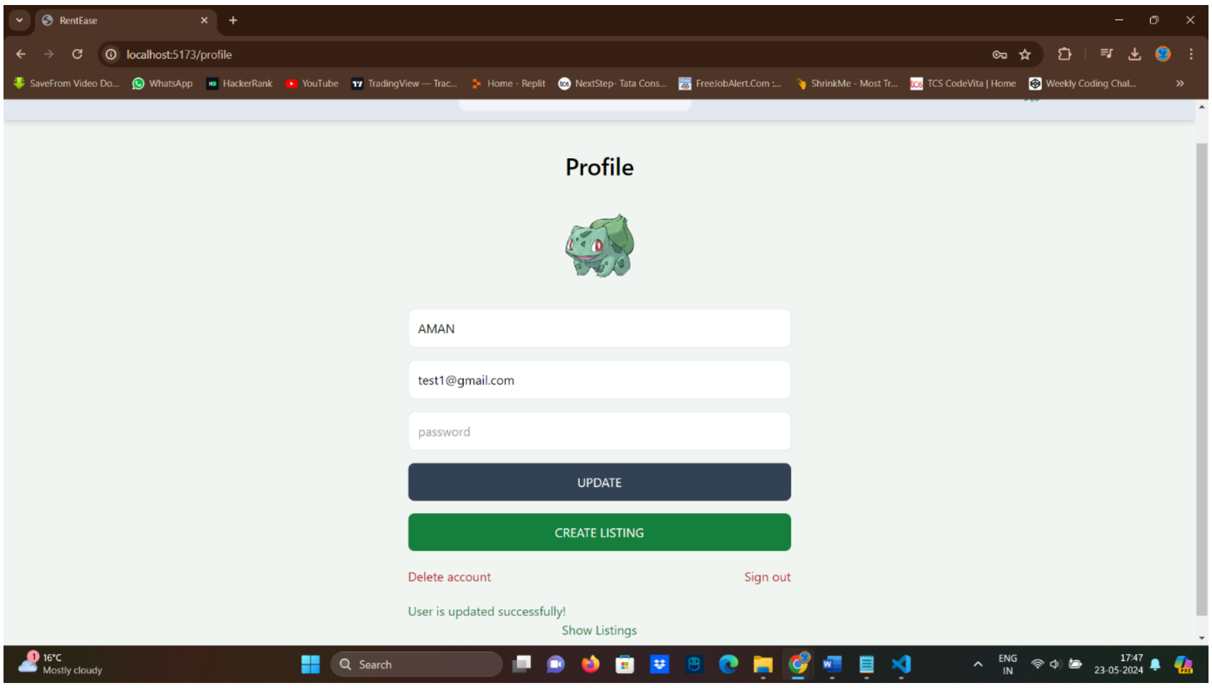
UPDATE

CREATE LISTING

Delete account

Sign out

Show Listings



Profile

AMAN

test1@gmail.com

password

UPDATE

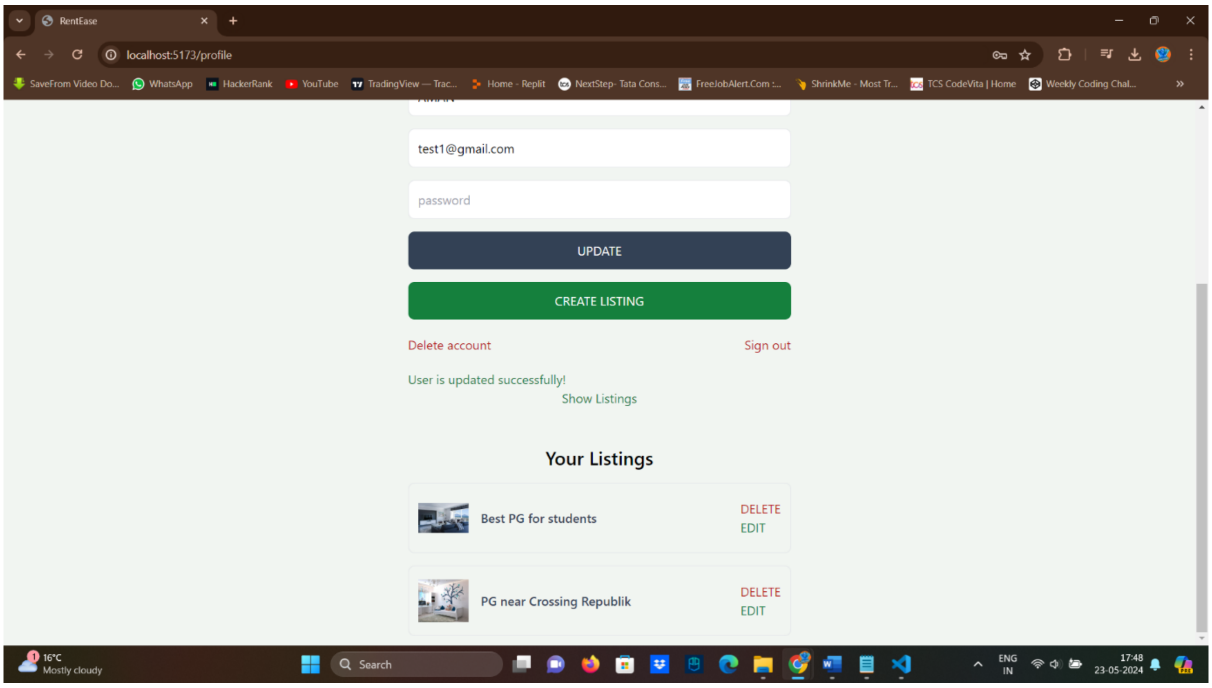
CREATE LISTING

Delete account

Sign out

User is updated successfully!

Show Listings



localhost:5173/profile

test1@gmail.com

password

UPDATE

CREATE LISTING

Delete account Sign out

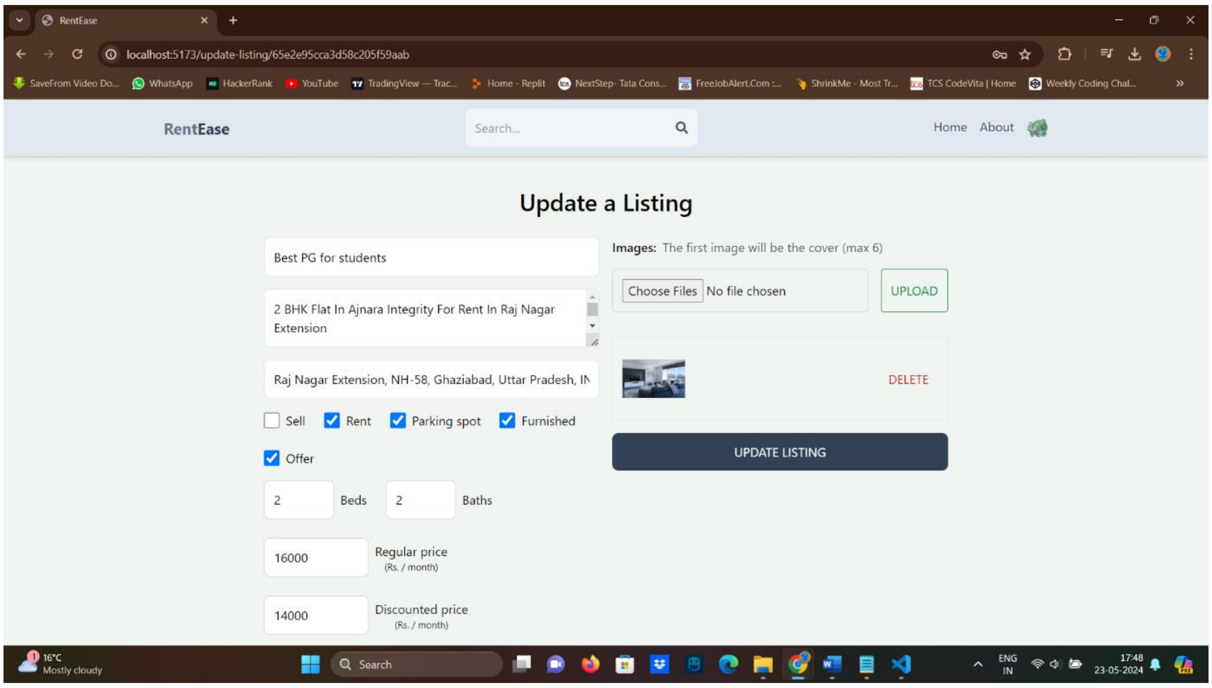
User is updated successfully!

Show Listings

Your Listings

Best PG for students DELETE EDIT

PG near Crossing Republik DELETE EDIT



localhost:5173/update-listing/65e2e95cca3d58c205f59aab

RentEase Search... Home About

Update a Listing

Best PG for students

2 BHK Flat In Ajnara Integrity For Rent In Raj Nagar Extension

Raj Nagar Extension, NH-58, Ghaziabad, Uttar Pradesh, IN

Sell [✓ Rent] [✓ Parking spot] [✓ Furnished]

[✓ Offer]

2 Beds 2 Baths

16000 Regular price (Rs. / month)

14000 Discounted price (Rs. / month)

Images: The first image will be the cover (max 6)

Choose Files No file chosen UPLOAD

DELETE

UPDATE LISTING

6.2 Coding

The coding aspect of RentEase involves the development of backend and frontend components using the MERN stack (MongoDB, Express.js, React.js, Node.js). Below are the key elements of the coding process:

Overview of Code Structure: This section provides an overview of the project's codebase structure, including directory organization and the main files. It outlines how the frontend and backend components are structured and how they interact with each other.

Key Code Snippets: Here, essential code snippets are provided to illustrate key functionalities of RentEase:

User Authentication: Sample code demonstrating user registration, login, and authentication using JSON Web Tokens (JWT).

Property Listings: Code snippets for implementing CRUD (Create, Read, Update, Delete) operations on property listings, including adding, updating, and deleting listings.

Search Functionality: Code illustrating the implementation of search and filtering features, allowing users to search for properties based on various criteria.

Messaging System: Code snippets for implementing the messaging functionality, including sending, receiving, and managing messages between users.

<u>auth.controller.js</u>

```
import User from '../models/user.model.js';

import bcryptjs from 'bcryptjs';

import { errorHandler } from '../utils/error.js';

import jwt from 'jsonwebtoken';


export const signup = async (req, res, next) => {
  const { username, email, password } = req.body;
  const hashedPassword = bcryptjs.hashSync(password, 10);
  const newUser = new User({ username, email, password: hashedPassword });
  try {
    await newUser.save();
    res.status(201).json('User created successfully!');
  } catch (error) {
    next(error);
  }
};

export const signin = async (req, res, next) => {
  const { email, password } = req.body;
  try {
    const validUser = await User.findOne({ email });
    if (!validUser) return next(errorHandler(404, 'User not found!'));
    const validPassword = bcryptjs.compareSync(password, validUser.password);
    if (!validPassword) return next(errorHandler(401, 'Wrong credentials!'));
    const token = jwt.sign({ id: validUser._id }, process.env.JWT_SECRET);
    const { password: pass, ...rest } = validUser._doc;
    res
      .cookie('access_token', token, { httpOnly: true })
      .status(200)
      .json(rest);
  } catch (error) {
    next(error);
  }
```

```
};

export const google = async (req, res, next) => {
  try {
    const user = await User.findOne({ email: req.body.email });
    if (user) {
      const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET);
      const { password: pass, ...rest } = user._doc;
      res
        .cookie('access_token', token, { httpOnly: true })
        .status(200)
        .json(rest);
    } else {
      const generatedPassword =
        Math.random().toString(36).slice(-8) +
        Math.random().toString(36).slice(-8);
      const hashedPassword = bcryptjs.hashSync(generatedPassword, 10);
      const newUser = new User({
        username:
          req.body.name.split(' ').join('').toLowerCase() +
          Math.random().toString(36).slice(-4),
        email: req.body.email,
        password: hashedPassword,
        avatar: req.body.photo,
      });
      await newUser.save();
      const token = jwt.sign({ id: newUser._id }, process.env.JWT_SECRET);
      const { password: pass, ...rest } = newUser._doc;
      res
        .cookie('access_token', token, { httpOnly: true })
        .status(200)
        .json(rest);
    }
  } catch (error) {
    next(error);
  }
};

export const signOut = async (req, res, next) => {
  try {
    res.clearCookie('access_token');
    res.status(200).json('User has been logged out!');
  } catch (error) {
    next(error);
  }
};
```

listing.controller.js

```
import Listing from '../models/listing.model.js';
import { errorHandler } from '../utils/error.js';

export const createListing = async (req, res, next) => {
  try {
    const listing = await Listing.create(req.body);
    return res.status(201).json(listing);
  } catch (error) {
    next(error);
  }
};

export const deleteListing = async (req, res, next) => {
  const listing = await Listing.findById(req.params.id);

  if (!listing) {
    return next(errorHandler(404, 'Listing not found!'));
  }

  if (req.user.id !== listing.userRef) {
    return next(errorHandler(401, 'You can only delete your own listings!'));
  }

  try {
    await Listing.findByIdAndDelete(req.params.id);
    res.status(200).json('Listing has been deleted!');
  } catch (error) {
    next(error);
  }
};

export const updateListing = async (req, res, next) => {
  const listing = await Listing.findById(req.params.id);
  if (!listing) {
    return next(errorHandler(404, 'Listing not found!'));
  }
  if (req.user.id !== listing.userRef) {
    return next(errorHandler(401, 'You can only update your own listings!'));
  }

  try {
    const updatedListing = await Listing.findByIdAndUpdate(
      req.params.id,
      req.body,
      { new: true }
    );
    res.status(200).json(updatedListing);
  } catch (error) {
    next(error);
  }
};

export const getListing = async (req, res, next) => {
  try {
    const listing = await Listing.findById(req.params.id);
    if (!listing) {
      return next(errorHandler(404, 'Listing not found!'));
    }
    res.status(200).json(listing);
  } catch (error) {
    next(error);
  }
};

export const getListings = async (req, res, next) => {
  try {
    const limit = parseInt(req.query.limit) || 9;
    const startIndex = parseInt(req.query.startIndex) || 0;
    let offer = req.query.offer;

    if (offer === undefined || offer === 'false') {
      offer = { $in: [false, true] };
    }

    let furnished = req.query.furnished;

    if (furnished === undefined || furnished === 'false') {
      furnished = { $in: [false, true] };
    }
```

```
let parking = req.query.parking;

if (parking === undefined || parking === 'false') {
  parking = { $in: [false, true] };
}

let type = req.query.type;

if (type === undefined || type === 'all') {
  type = { $in: ['sale', 'rent'] };
}

const searchTerm = req.query.searchTerm || '';

const sort = req.query.sort || 'createdAt';

const order = req.query.order || 'desc';

const listings = await Listing.find({
  name: { $regex: searchTerm, $options: 'i' },
  offer,
  furnished,
  parking,
  type,
})
  .sort({ [sort]: order })
  .limit(limit)
```

```
      .skip(startIndex);

    return res.status(200).json(listings);
  } catch (error) {
    next(error);
  }
};
```

user.controller.js

```
import bcryptjs from 'bcryptjs';
import User from '../models/user.model.js';
import { errorHandler } from '../utils/error.js';
import Listing from '../models/listing.model.js';

export const test = (req, res) => {
  res.json({
    message: 'Api route is working!',
  });
};

export const updateUser = async (req, res, next) => {
  if (req.user.id !== req.params.id)
    return next(errorHandler(401, 'You can only update your own account!'));
  try {
    if (req.body.password) {
      req.body.password = bcryptjs.hashSync(req.body.password, 10);
    }

    const updatedUser = await User.findByIdAndUpdate(
      req.params.id,
      {
        $set: {
          username: req.body.username,
          email: req.body.email,
          password: req.body.password,
};

          avatar: req.body.avatar,
        },
      },
      { new: true }
    );

    const { password, ...rest } = updatedUser._doc;

    res.status(200).json(rest);
  } catch (error) {
    next(error);
  }
};

export const deleteUser = async (req, res, next) => {
  if (req.user.id !== req.params.id)
    return next(errorHandler(401, 'You can only delete your own account!'));
  try {
    await User.findByIdAndDelete(req.params.id);
    res.clearCookie('access_token');
    res.status(200).json('User has been deleted!');
  } catch (error) {
    next(error);
  }
};

export const getUserListings = async (req, res, next) => {
  if (req.user.id === req.params.id) {
    try {
      const listings = await Listing.find({ userRef: req.params.id });
      res.status(200).json(listings);
    } catch (error) {
      next(error);
    }
  } else {
    return next(errorHandler(401, 'You can only view your own listings!'));
  }
};

export const getUser = async (req, res, next) => {
  try {

    const user = await User.findById(req.params.id);

    if (!user) return next(errorHandler(404, 'User not found!'));

    const { password: pass, ...rest } = user._doc;

    res.status(200).json(rest);
  } catch (error) {
    next(error);
  }
};
```

listing.model.js

```
import mongoose from 'mongoose';


const listingSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: true,
    },
    description: {
      type: String,
      required: true,
    },
    address: {
      type: String,
      required: true,
    },
    regularPrice: {
      type: Number,
      required: true,
    },
    discountPrice: {
      type: Number,
      required: true,
    },
    bathrooms: {
      type: Number,
      required: true,
    },
    bedrooms: {
      type: Number,
      required: true,
    },
    furnished: {
      type: Boolean,
      required: true,
    },
    parking: {
      type: Boolean,
      required: true,
    },
    type: {
      type: String,
      required: true,
    },
    offer: {
      type: Boolean,
      required: true,
    },
    imageUrls: {
      type: Array,
      required: true,
    },
    userRef: {
      type: String,
      required: true,
    },
  },
  { timestamps: true }
);

const Listing = mongoose.model('Listing', listingSchema);

export default Listing;
```

```
},
```

user.model.js

```
import mongoose from 'mongoose';


const userSchema = new mongoose.Schema(
  {
    username: {
      type: String,
      required: true,
      unique: true,
    },
    email: {
      type: String,
      required: true,
      unique: true,
    },
    password: {
      type: String,
      required: true,
    },
    avatar:{
      type: String,
      default:      "https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-picture-
973460_1280.png"
    },
  },
  { timestamps: true }
);
```

```
const User = mongoose.model('User', userSchema);

export default User;
```

auth.route.js

```
import express from 'express';

import { google, signOut, signin, signup } from '../controllers/auth.controller.js';


const router = express.Router();


router.post("/signup", signup);

router.post("/signin", signin);

router.post('/google', google);

router.get('/signout', signOut)


export default router;
```

listing.route.js

```
import express from 'express';

import { createListing, deleteListing, updateListing, getListing, getListings } from '../controllers/listing.controller.js';

import { verifyToken } from '../utils/verifyUser.js';


const router = express.Router();


router.post('/create', verifyToken, createListing);

router.delete('/delete/:id', verifyToken, deleteListing);

router.post('/update/:id', verifyToken, updateListing);

router.get('/get/:id', getListing);

router.get('/get', getListings);


export default router;
```

user.route.js

```
import express from 'express';

import { deleteUser, test, updateUser,  getUserListings, getUser} from '../controllers/user.controller.js';

import { verifyToken } from '../utils/verifyUser.js';



const router = express.Router();


router.get('/test', test);

router.post('/update/:id', verifyToken, updateUser)

router.delete('/delete/:id', verifyToken, deleteUser)

router.get('/listings/:id', verifyToken, getUserListings)

router.get('/:id', verifyToken, getUser)


export default router;
```

Contact.jsx

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

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


export default function Contact({ listing }) {

  const [landlord, setLandlord] = useState(null);

  const [message, setMessage] = useState('');

  const onChange = (e) => {

    setMessage(e.target.value);

  };


  useEffect(() => {

    const fetchLandlord = async () => {

      try {

        const res = await fetch(`/api/user/${listing.userRef}`);

        const data = await res.json();

        setLandlord(data);

      } catch (error) {

        console.log(error);

      }

    };

    fetchLandlord();

  }, [listing.userRef]);

  return (

    <>

      {landlord && (

        <div className='flex flex-col gap-2'>
```

```
          <p>
            Contact <span className='font-semibold'>{landlord.username}</span>{' '}
            for{' '}
            <span className='font-semibold'>{listing.name.toLowerCase()}</span>
          </p>
          <textarea
            name='message'
            id='message'
            rows='2'
            value={message}
            onChange={onChange}
            placeholder='Enter your message here...'
            className='w-full border p-3 rounded-lg'
          ></textarea>

          <Link
          to={`mailto:${landlord.email}?subject=Regarding
${listing.name}&body=${message}`}
          className='bg-slate-700 text-white text-center p-3 uppercase rounded-lg
hover:opacity-95'
          >
            Send Message
          </Link>
        </div>
      )}
    </>
  );
}
```

Header.jsx

```
import { FaSearch } from 'react-icons/fa';

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

import { useSelector } from 'react-redux';

import { useEffect, useState } from 'react';


export default function Header() {

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

  const [searchTerm, setSearchTerm] = useState('');

  const navigate = useNavigate();

  const handleSubmit = (e) => {

    e.preventDefault();

    const urlParams = new URLSearchParams(window.location.search);

    urlParams.set('searchTerm', searchTerm);

    const searchQuery = urlParams.toString();

    navigate(`/search?${searchQuery}`);

  };


  useEffect(() => {

    const urlParams = new URLSearchParams(location.search);

    const searchTermFromUrl = urlParams.get('searchTerm');

    if (searchTermFromUrl) {

      setSearchTerm(searchTermFromUrl);

    }

  }, [location.search]);

  return (

    <header className='bg-slate-200 shadow-md'>
```

```
<div className='flex justify-between items-center max-w-6xl mx-auto p-3'>
  <Link to='/'>
    <h1 className='font-bold text-sm sm:text-xl flex flex-wrap'>
      <span className='text-slate-500'>Rent</span>
      <span className='text-slate-700'>Ease </span>
    </h1>
  </Link>
  <form
    onSubmit={handleSubmit}
    className='bg-slate-100 p-3 rounded-lg flex items-center'
  >
    <input
      type='text'
      placeholder='Search...'
      className='bg-transparent focus:outline-none w-24 sm:w-64'
      value={searchTerm}
      onChange={(e) => setSearchTerm(e.target.value)}
    />
    <button>
      <FaSearch className='text-slate-600' />
    </button>
  </form>
  <ul className='flex gap-4'>
    <Link to='/'>
      <li className='hidden sm:inline text-slate-700 hover:underline'>
        Home
      </li>
```

```
          </Link>
          <Link to='/about'>
            <li className='hidden sm:inline text-slate-700 hover:underline'>
              About
            </li>
          </Link>
          <Link to='/profile'>
            {currentUser ? (
              <img
                className='rounded-full h-7 w-7 object-cover'
                src={currentUser.avatar}
                alt='profile'
              />
            ) : (
              <li className=' text-slate-700 hover:underline'> Sign in</li>
            )}
          </Link>
        </ul>
      </div>
    </header>
  );
}
```

ListingItem.jsx

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

import { MdLocationOn } from 'react-icons/md';


export default function ListingItem({ listing }) {

  return (

    <div className='bg-white shadow-md hover:shadow-lg transition-shadow overflow-hidden rounded-lg w-full sm:w-[330px]'>

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

        <img

          src={

            listing.imageUrls[0] ||

            'https://53.fs1.hubspotusercontent-na1.net/hub/53/hubfs/Sales_Blog/real-estate-business-compressor.jpg?width=595&height=400&name=real-estate-business-compressor.jpg'

          }

          alt='listing cover'

          className='h-[320px] sm:h-[220px] w-full object-cover hover:scale-105 transition-scale duration-300'

        />

        <div className='p-3 flex flex-col gap-2 w-full'>

          <p className='truncate text-lg font-semibold text-slate-700'>

            {listing.name}

          </p>

          <div className='flex items-center gap-1'>

            <MdLocationOn className='h-4 w-4 text-green-700' />

            <p className='text-sm text-gray-600 truncate w-full'>

              {listing.address}
          </p>
        </div>
        <p className='text-sm text-gray-600 line-clamp-2'>
          {listing.description}
        </p>
        <p className='text-slate-500 mt-2 font-semibold '>
          Rs
          {listing.offer
            ? listing.discountPrice.toLocaleString('en-US')
            : listing.regularPrice.toLocaleString('en-US')}
          {listing.type === 'rent' && ' / month'}
        </p>
        <div className='text-slate-700 flex gap-4'>
          <div className='font-bold text-xs'>
            {listing.bedrooms > 1
              ? `${listing.bedrooms} beds `
              : `${listing.bedrooms} bed `}
          </div>
          <div className='font-bold text-xs'>
            {listing.bathrooms > 1
              ? `${listing.bathrooms} baths `
              : `${listing.bathrooms} bath `}
          </div>
        </div>
      </div>
    </Link>
  </div>
  );
}
```

OAuth.jsx

```
import { GoogleAuthProvider, getAuth, signInWithPopup } from 'firebase/auth';
import { app } from '../firebase';
import { useDispatch } from 'react-redux';
import { signInSuccess } from '../redux/user/userSlice';
import { useNavigate } from 'react-router-dom';

export default function OAuth() {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const handleGoogleClick = async () => {
    try {
      const provider = new GoogleAuthProvider();
      const auth = getAuth(app);

      const result = await signInWithPopup(auth, provider);

      const res = await fetch('/api/auth/google', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          name: result.user.displayName,
          email: result.user.email,
          photo: result.user.photoURL,
        }),
      });
      const data = await res.json();
      dispatch(signInSuccess(data));
      navigate('/');
    } catch (error) {
      console.log('could not sign in with google', error);
    }
  };
  return (
    <button
      onClick={handleGoogleClick}
      type='button'
      className='bg-red-700 text-white p-3 rounded-lg uppercase hover:opacity-95'
    >
      Continue with google
    </button>
  );
}
```

Chapter 7: Testing (Test Plan/Cases/Result)

7.1 Test Plan

A comprehensive test plan is crucial to ensure the reliability and functionality of RentEase. The test plan outlines the testing strategy, objectives, scope, and resources required for testing. It includes various types of testing such as unit testing, integration testing, system testing, and user acceptance testing (UAT). The objectives of the test plan are to verify that all functionalities work as intended, identify and fix any bugs or issues, ensure compatibility across different devices and browsers, and validate the overall user experience. The scope of testing covers all features and functionalities of RentEase, including user authentication, property listings, search and filtering, messaging system, and user dashboard. The resources required for testing include testing tools, devices, and personnel responsible for conducting and documenting the tests.

7.2 Test Cases

Test cases are detailed descriptions of specific scenarios and inputs to be tested, along with expected outcomes and actual results. Each test case covers a particular aspect or functionality of RentEase and includes steps to execute the test, input data, expected results, and actual results observed during testing. For example, a test case for user authentication would include steps to register a new user, input valid credentials for login, and verify that the user is successfully logged in. Test cases are designed to validate the correctness and robustness of RentEase and ensure that it meets the specified requirements and user expectations.

7.3 Test Results

After executing the test cases, the results of testing are documented and summarized. This includes a summary of the testing outcomes, highlighting any issues or defects encountered during testing, such as bugs, errors, or inconsistencies. The test results also include details of any successful tests and confirmation that the functionalities work as intended. Additionally, any deviations from the expected results are noted, along with the steps taken to address them.

7.4 Issues and Resolutions

This section provides a detailed discussion of significant issues encountered during testing and the steps taken to resolve them. It includes descriptions of the issues identified, their root causes, and the actions taken to fix them. Additionally, it may include discussions on any challenges faced during testing, such as compatibility issues, performance issues, or usability issues, and the strategies employed to overcome them.

Chapter 8: Conclusion

8.1 Project Limitations

Despite the successful development of RentEase, there may be certain limitations encountered during the project. These limitations could include technical constraints, such as limitations of the chosen technologies or frameworks, resource limitations, such as budget or time constraints, or scope changes that may have affected the project deliverables. It is essential to acknowledge these limitations to provide a realistic assessment of the project's achievements and to identify areas for improvement in future projects.

8.2 Future Scope

While RentEase fulfills its primary objectives, there is always room for improvement and future enhancements. This section discusses potential areas for future development and expansion of the platform. This could include features such as mobile application development to reach a broader audience, integration with other services or platforms to enhance functionality, and the implementation of advanced recommendation algorithms to personalize the user experience further. By outlining the future scope of RentEase, this section provides a roadmap for ongoing development and improvement of the platform to meet the evolving needs of its users.

Chapter 9: References

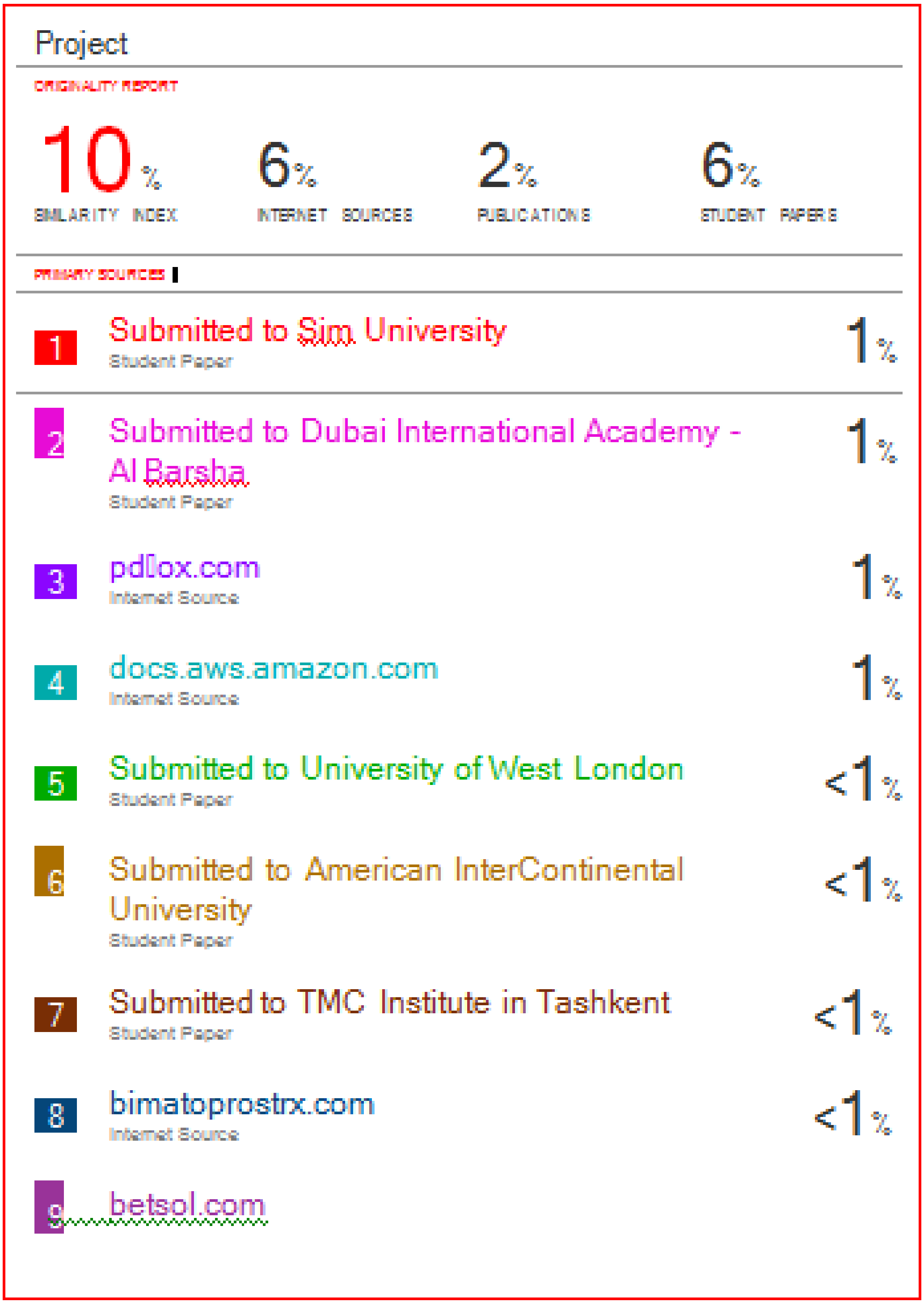
MERN Documentation:

MongoDB. - MongoDB Documentation. Retrieved from [https://www.mongodb.com/docs/]

Express.js. - Express.js Documentation. Retrieved from [https://expressjs.com/en/5x/api.html]

React.js. - React.js Documentation. Retrieved from [https://legacy.reactjs.org/docs/getting-started.html]

Node.js. - Node.js Documentation. Retrieved from [https://nodejs.org/docs/latest/api/]



Project

ORIGINALITY REPORT

10%	6%	2%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Sim University Student Paper	1%
2	Submitted to Dubai International Academy - Al Barsha Student Paper	1%
3	pdfox.com Internet Source	1%
4	docs.aws.amazon.com Internet Source	1%
5	Submitted to University of West London Student Paper	<1%
6	Submitted to American InterContinental University Student Paper	<1%
7	Submitted to TMC Institute in Tashkent Student Paper	<1%
8	bimatoprostrx.com Internet Source	<1%
9	betsol.com	