FAR WESTERN UNIVERSITY

SCHOOL OF ENGINEERING

MAHENDRANAGAR, NEPAL

A MINOR PROJECT ON

A REAL ESTATE WEB APPLICATION - GHAREADY

SUBMITTED BY

BHAWANA CHAND (EG-2021-1-1-3605)

BHUWAN CHAND (EG-2021-1-1-3606)

ISHWOR NATH (EG-2021-1-1-3615)

PANKAJ JOSHI (EG-2021-1-1-3629)

RAJENDRA PD JOSHI (EG-2021-1-1-3636)

SUBMITTED TO

DEPARTMENT OF COMPUTER ENGINEERING

FAR WESTERN UNIVERSITY, CENTRAL CAMPUS

MAHENDRANAGAR

SUBMISSION DATE 2081/07/23

ACKNOWLEDGEMENT

We would like to extend our heartfelt gratitude to the Head of Department (HoD), Er. Harendra Raj Kalauni, and esteemed faculty members Er. Rohit Bist, Er. Kamal Lekhak, Er. Rishi K. Marseni and Er. Sanjay Thakurathi, for their unwavering support, expert guidance, and invaluable feedback throughout this project. Their insightful advice and encouragement have been instrumental in shaping this work, and we are deeply grateful for their continuous mentorship and belief in our efforts.

We also wish to express our deepest respect and remembrance for the late Er. Bhim Pun, who served as our project supervisor during the initial stages. His guidance and dedication provided us with a strong foundation, and his absence has been a profound loss. His memory continues to inspire us, and we honor his invaluable contributions to this project.

We are deeply thankful to the esteemed teachers and faculty members whose valuable feedback, encouragement, and scholarly insights have enriched this project. Their dedication to academic excellence and commitment to fostering knowledge have been a source of inspiration. Their collective guidance has immensely contributed to the refinement and improvement of this project.

ABSTRACT

Ghaready is a comprehensive real estate application designed to simplify property searching, listing, and user management, offering an intuitive user experience. The platform includes secure and efficient sign-in and sign-up processes, allowing users to register and log in seamlessly. It also features a customizable profile page, complete with image upload functionality, enabling users to personalize their profiles.

The application provides full control over user profiles with options to update or delete personal information. Users can create, edit, and delete property listings, each with image uploads to enhance visual appeal and user engagement. Additionally, the platform ensures privacy by making profile pages accessible only to authorized users.

A powerful search functionality is integrated into the header and a dedicated search page, allowing users to explore available listings effectively. The “Contact Landlord” feature further enhances user interaction by enabling direct communication with property owners. The platform also supports a “Show More Listings” functionality for efficient browsing of extensive property catalogs.

With an emphasis on privacy, user control, and seamless navigation, Ghaready offers a complete solution for real estate enthusiasts, simplifying the process of finding and managing property listings.

Keywords: RealEstate, Platform, Listings, User, Profile, Customization, Authentication, Vite, React, Firebase, Privacy, Security, Search, Upload, Interaction, Contact, Landlords, Images, Listings, Filtering, Map, Location, UI, Database, Notifications, Optimization, Dashboard, Interface, Backend, Frontend, Responsive, Access, Permissions, Navigation, Storage, Query

TABLE OF CONTENTS

ACKNOWLEDGEMENT **2**
ABSTRACT **1**
TABLE OF CONTENTS **2**
LIST OF FIGURES **1**
LIST OF ABBREVIATIONS **1**
1. INTRODUCTION **1**
1.1 Background Study **1**
1.2 Problem Definition **1**
1.3 Objectives **1**
1.4 Scope of Project **2**
2. LITERATURE REVIEW **3**
2.1. Historical Evolution **3**
2.2 Algorithms and Models **3**
2.2.1 Search-Based Filtering 4
2.2.2 Content-Based Filtering 5
2.2.3 Feature-Based Filtering 5
2.2.4 User Profile Matching 5
2.2.5 Geolocation-Based Recommendation 5
2.2.6 Hybrid Model 6
2.3 Data Pre-Processing and Feature Engineering **6**
2.4 Gaps and Challenges **7**
3. METHODOLOGY **8**
3.1 Block Diagram 8
3.2 Tools and Techniques 8
3.2.1 Frontend Development 8
3.2.2 Backend Development 9
3.2.3 Database and Data Management 9
3.2.4 Development Environment and Version Control 9
3.2.5 Additional Tools 9
3.3 Data Management 10
3.3.2 User Data 10
3.3.3 Property Listings 10
3.3.4 Interactions and Communication 10
3.3.5 Database Structure 11
3.3.6 External Data Integration 11
3.4 System Design 12
3.4.1 USE CASE DIAGRAM 12
3.4.1.1 Use Case Diagram Description 13
3.4.2 CLASS DIAGRAM 14
3.4.2.1 Class Diagram Description 15
4. IMPLEMENTATION **17**
4.1 System Architecture 17
4.2 Database design 17
4.3 Backend development **17**

4.4 Frontend development.....18
5. RESULT.....21
6. CONCLUSION.....22
7. LIMITATION.....23
8. FUTURE ENHANCEMENTS.....24
REFERENCES.....26
APPENDIX.....27

LIST OF FIGURES

Figure 3.1: Block diagram 8
Figure 3.4: System design 12
Figure 3.4.1: Use case diagram 13
Figure 3.4.2: Class diagram 15
Figure 4.I1: API Testing by Insomnia 19
Figure 4.I2: Uer controller API 19
Figure A1: Sign up page 27
Figure A2: Sign in page 27
Figure A3: Home page 28
Figure A4: Listings 28
Figure A5: Search page 29
Figure A6: Profile page 29
Figure A7: Create listing 30

LIST OF ABBREVIATIONS

UI	User Interface
UX	User Experience
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
CRUD	Create, Read, Update, Delete
ID	Identifier
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
CSS	Cascading Style Sheets
JS	JavaScript
DB	Database
OTP	One-Time Password
MVC	Model-View-Controller
SEO	Search Engine Optimization
DNS	Domain Name System
URL	Uniform Resource Locator

1. INTRODUCTION

1.1 Background Study

The real estate industry has seen significant digital transformation over the past decade. With the advent of online property platforms, users can now explore, compare, and engage with property listings from the comfort of their homes. However, despite these advancements, many users face difficulties in managing listings, filtering relevant properties, and securely interacting with property owners or landlords. This background study explores the need for streamlined, feature-rich platforms that address these user challenges while also ensuring data security and ease of use.

1.2 Problem Definition

The abundance of property listings online can make it challenging for users to efficiently locate options tailored to their needs. Traditional platforms often lack intuitive filters, profile personalization, and user-friendly listing management capabilities. Users also require secure methods to sign in, sign out, and manage their profiles and interactions with landlords or agents. This project aims to solve these issues by developing a comprehensive real estate application that simplifies the user experience, enhances accessibility, and improves engagement with personalized listing recommendations..

1.3 Objectives

- To develop a real estate web application that allows users to create and manage listings seamlessly.
- To provide a secure and efficient user authentication system, including functionalities for signing up, signing in, and signing out.
- To create an intuitive search functionality that filters property listings based on user preferences.
- To implement a user-friendly interface that enhances navigation and interaction with the platform.

1.4 Scope of Project

The project is limited to the development of a web-based platform for real estate listing

management, focusing primarily on usability and user experience. It covers secure user authentication, listing creation and management, profile customization, and landlord interaction features. However, the project does not include complex recommendation algorithms or real-time property value estimations. Future enhancements may expand to include AI-driven recommendations and price trend analysis to further support user decision-making.

2. LITERATURE REVIEW

2.1. Historical Evolution

The concept of digital real estate platforms has evolved alongside advancements in internet technology and user expectations. Initially, property transactions were largely conducted offline, requiring buyers and sellers to rely on intermediaries like real estate agents, who held market knowledge and facilitated transactions. However, with the advent of Web 1.0 in the 1990s, basic property listing websites emerged, allowing agencies to display property information online. These early platforms served as virtual classifieds, displaying limited data about properties without much user interaction or filtering options.

By the early 2000s, the growth of Web 2.0 enabled greater user interaction, laying the foundation for today's real estate platforms. Websites like Zillow and Trulia incorporated advanced search filters, mortgage calculators, and interactive maps, which improved user engagement and experience. More recently, with the integration of artificial intelligence (AI) and machine learning (ML), real estate platforms can now provide personalized recommendations, property value estimations, and predictive analytics based on user behavior. These advancements have shifted real estate platforms from mere listing sites to comprehensive digital tools, assisting users throughout their property-buying or rental journey.

Today's platforms leverage complex data models and user-centered designs to improve the quality of property recommendations, user navigation, and data security. Our project draws on these historical advancements to offer an intuitive, personalized experience that meets modern user needs, including secure authentication, user-friendly listing management, and a streamlined interface for interacting with landlords.

2.2 Algorithms and Models

The Ghaready real estate platform is built on carefully implemented algorithms and models designed to offer a secure, efficient, and user-friendly experience. The following key areas highlight the core functionalities that drive the platform's operations.

2.2.1 User Authentication and Profile Management

User authentication and profile management form the backbone of secure access and data

handling in Ghaready. Built on Firebase's authentication services, the system employs encryption for user credentials and session management for login and logout processes. The profile management functionality enables users to create, update, or delete their profiles, with access controlled by role-based permissions. This ensures only authorized users can access sensitive features like property listing management, maintaining privacy and user data security.

2.2.2 CRUD Operations and Property Listings

Create, Read, Update, and Delete (CRUD) operations enable seamless interactions with both property listings and user profiles. These operations follow RESTful API principles to maintain a standardized data exchange protocol, ensuring robust and efficient data management. Users can add, view, modify, and delete property listings as well as their own profiles. Additionally, the platform allows listing images to be uploaded with Firebase Storage, utilizing compression and asynchronous storage to optimize quality and loading speed.

2.2.3 Search and Filtering Functionalities

The platform's search and filtering capabilities help users find relevant property listings quickly and efficiently. A keyword-based search algorithm enables users to retrieve listings matching specific terms, while additional filtering options allow users to refine results by parameters such as location, price, and property type. This model prioritizes user experience by delivering highly relevant results in an intuitive, easy-to-navigate format, making it easier for users to identify suitable properties.

2.2.4 Communication and Data Security

Data privacy and security are crucial components of Ghaready's design. Firebase's secure data handling and encryption protect sensitive information, with security protocols implemented to restrict unauthorized data access. Additionally, the "Contact Landlord" feature facilitates secure communication between potential renters or buyers and property owners while ensuring user data privacy. These safeguards are in place to comply with industry standards for data protection, delivering a safe environment for users to interact and engage.

2.3 Data Pre-Processing and Feature Engineering

In data-driven applications, especially recommendation systems, data pre-processing and feature engineering are critical. Data gathered from user interactions, such as clicks and search histories, requires thorough cleaning, normalization, and transformation before it can be used in models. Feature engineering involves selecting and crafting meaningful features, such as price range and location preference, to enhance recommendation relevance. For our project, we focus on pre-processing listing data and user interaction data to maintain data quality and accuracy.

2.4 Gaps and Challenges

Despite technological advancements, there are notable gaps and challenges in real estate recommendation systems. Issues like data privacy, user trust, and recommendation transparency remain significant. While machine learning enhances personalization, models may unintentionally reinforce biases or provide recommendations that lack interpretability. Our project aims to mitigate these gaps by prioritizing transparency and user control, balancing personalization with ethical consideration.

3. METHODOLOGY

3.1 Block Diagram

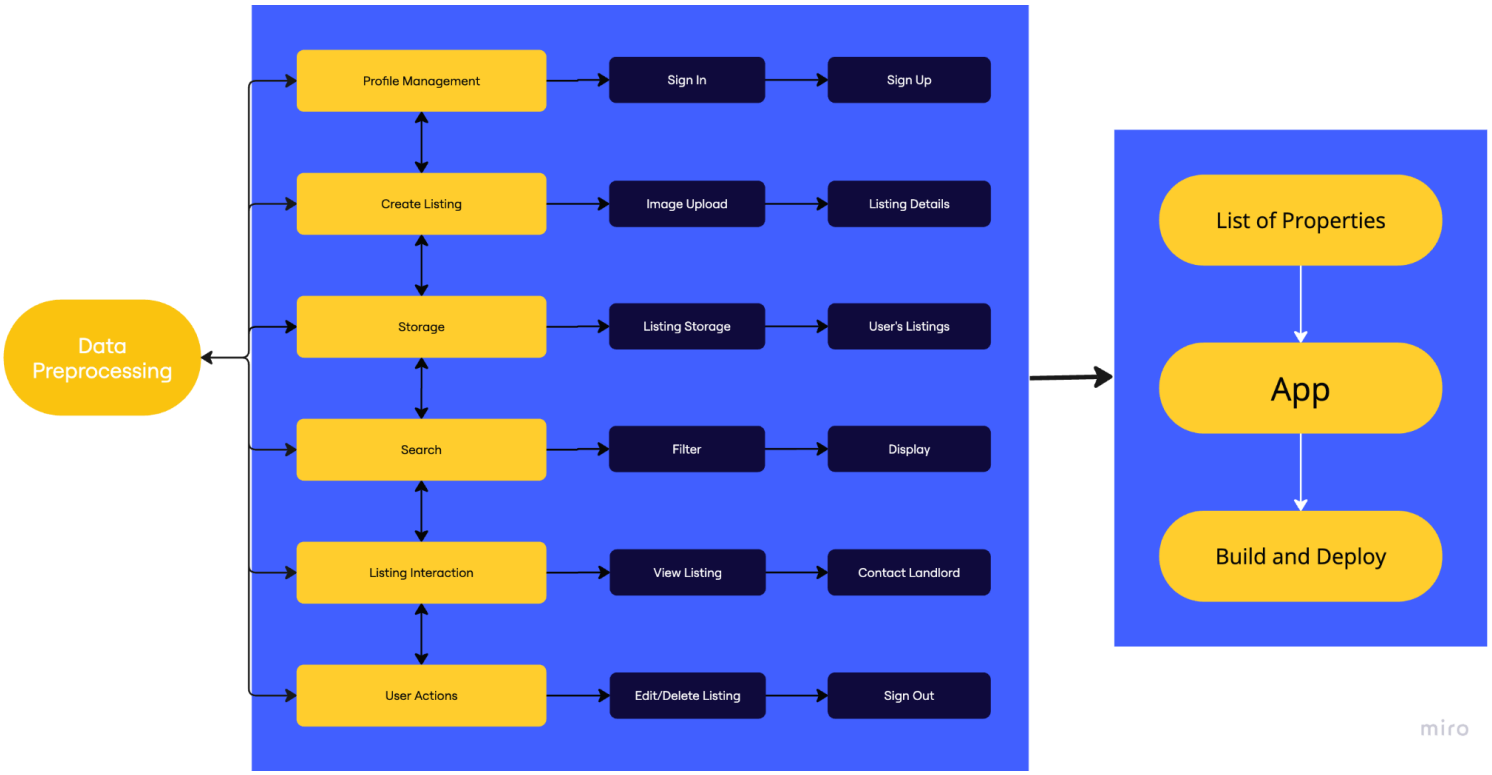

Figure 3. 1 Block Diagram

3.2 Tools and Techniques

3.2.1 Frontend Development

- **HTML, CSS, JavaScript**: These foundational languages were used to create the structure, style, and interactivity of the web application.
- **React**: React was selected as the primary frontend framework due to its component-based architecture and efficient state management, enabling a responsive and dynamic user interface.
- **Vite**: Used as a modern, fast build tool and development server, Vite enhanced the development experience, allowing for rapid iteration and optimization.
- **Tailwind CSS**: This utility-first CSS framework facilitated rapid and consistent styling, enabling a clean, modern design and faster prototyping.

3.2.2 Backend Development

- **Node.js**: The backend was developed using Node.js, a lightweight JavaScript runtime, allowing for asynchronous event-driven programming.
- **Express.js**: Express was used to set up the server and manage API routing, providing a solid foundation for handling user requests and interacting with the database.
- **Insomnia**: This API testing tool enabled the team to test HTTP requests during the backend development process, ensuring endpoints functioned as expected.

3.2.3 Database and Data Management

- **MongoDB**: MongoDB, a NoSQL database, was chosen for its scalability and flexibility, allowing for efficient storage of user profiles, listings, and associated data in a JSON-like format.

3.2.4 Development Environment and Version Control

- **Visual Studio Code (VSCode)**: VSCode was the primary IDE used, providing syntax highlighting, debugging tools, and extensions that streamlined the coding process.

- **Git and GitHub**: Git was used for version control, while GitHub facilitated collaboration, allowing team members to manage code versions, track changes, and collaborate effectively.

3.2.5 Additional Tools

- **Postman**: As a complement to Insomnia, Postman provided API testing capabilities, enabling thorough validation of all backend endpoints.
- **JWT (JSON Web Tokens)**: JWT was used to manage user authentication, ensuring secure sessions across the application.
- **Bcrypt**: This hashing algorithm was implemented for securely storing user passwords.
- **Google Docs**: Documentation, planning, and report writing were managed through Google Docs, providing a centralized platform for collaboration.

- **Miro**: Miro was utilized for creating project flowcharts, brainstorming ideas, and visualizing workflows, fostering effective planning and team collaboration.

3.3 Data Management

3.3.2 User Data

User Profiles: Each user has a unique profile containing information like username, email, password (securely hashed), and other personal details. This data is crucial for authentication, personalized recommendations, and secure interactions.

User Preferences and Activity: Interaction history, saved listings, and search preferences are stored and used to provide customized experiences, making it easier for users to find relevant listings based on their past activities.

3.3.3 Property Listings

Listing Details: Each property listing contains information such as property type, price, location, number of rooms, and amenities. This core data is essential for providing users with detailed property information.

Images and Media: Property images, uploaded by the listing creator, help users visualize properties. These images are stored and managed to ensure quick loading and high-quality presentation on listing pages.

Location Data: Geographic location (address, city, region) is stored to help users search properties by area. This information is also used to implement map functionalities, allowing users to view listings based on location.

3.3.4 Interactions and Communication

Contact Records: When users contact landlords or inquire about a property, these interaction records are stored. This data is essential for tracking user engagement and ensuring that landlords can manage potential leads efficiently.

Search History: To provide better recommendations and streamline the user experience, the application stores recent search queries, allowing for quicker access to relevant property searches.

3.3.5 Database Structure

MongoDB Collections: The application uses MongoDB collections to manage various data categories. For instance:

Users: Stores user profiles, preferences, and security details.

Listings: Contains property details, images, and related metadata.

Messages: Manages records of communications between users and landlords.

Data Models and Schemas: Each collection follows a defined schema, ensuring data consistency and making it easy to validate and retrieve information when needed.

3.3.6 External Data Integration

Geolocation API: An external geolocation API is integrated to support map functionalities and improve location-based searching, allowing users to visualize properties on a map interface.

Image Storage: Property images are stored in cloud storage for secure, scalable, and efficient image retrieval, contributing to faster loading times and a better user experience.

3.4 System Design

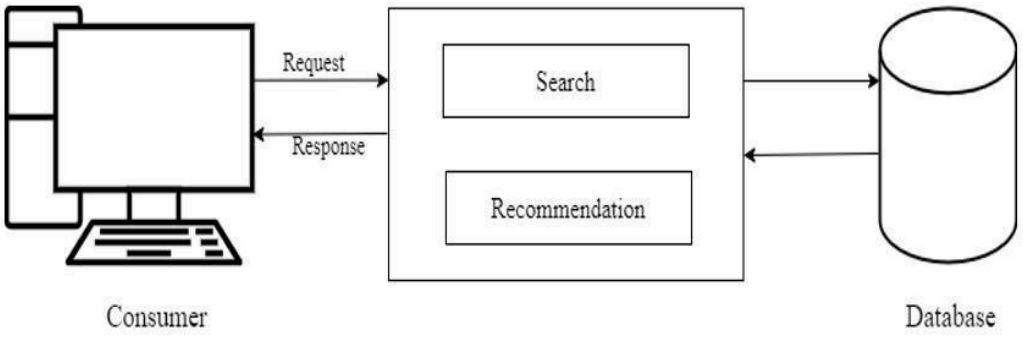

Figure 3.4: system design

3.4.1 Use Case Diagram

A use case diagram provides a visual representation of how users interact with a system, illustrating the various actions or tasks users can perform and their relationships with the system components.

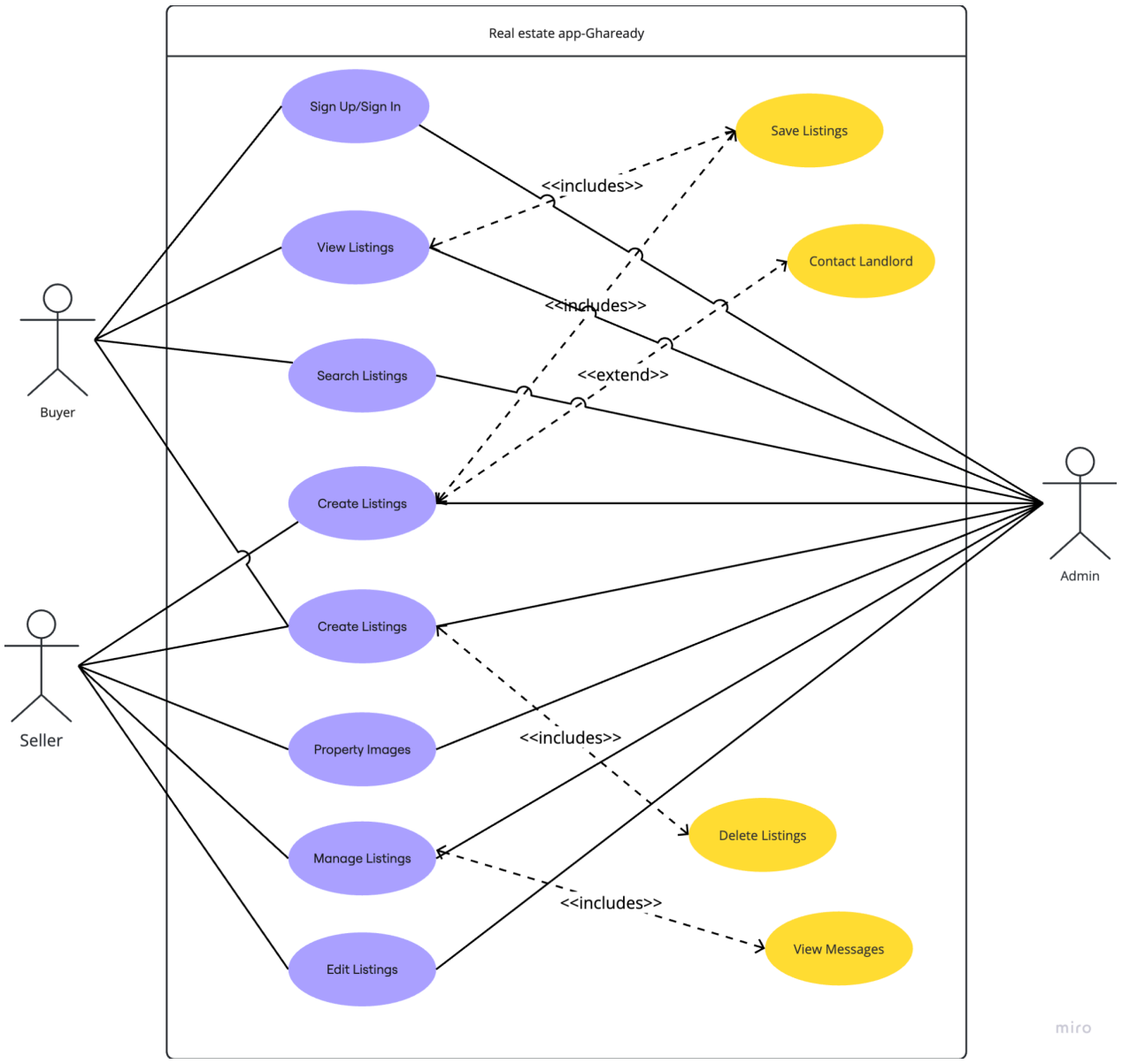

Figure 3.4.1: Use case diagram

3.4.1.1 Use Case Diagram Description

User Registration

In this use case, a new user registers by providing their name, email, and password. The system validates and stores these details securely. Upon successful registration, the user gains access to key features, including creating listings and saving preferred properties.

User Login

A registered user logs in by entering their email and password. The system verifies credentials,

granting access if correct. Successful login allows users to access personalized features, such as saved listings and user profiles.

Listing Creation

Here, a logged-in user can create a new property listing by entering details like location, price, and type. The user also uploads images to represent the property. Once submitted, the listing becomes available for others to view on the platform.

Contact Landlord

A user interested in a property can contact the landlord directly via the property page. After clicking "Contact Landlord" they can message or get contact information, facilitating direct communication regarding the listing.

3.4.2 Class Diagram

Class diagrams are the backbone of virtually each object-oriented methodology as well as UML. They describe the static structure of a system. Categories represent associate degree abstraction of entities with common characteristics. Associations represent the relationships between categories.

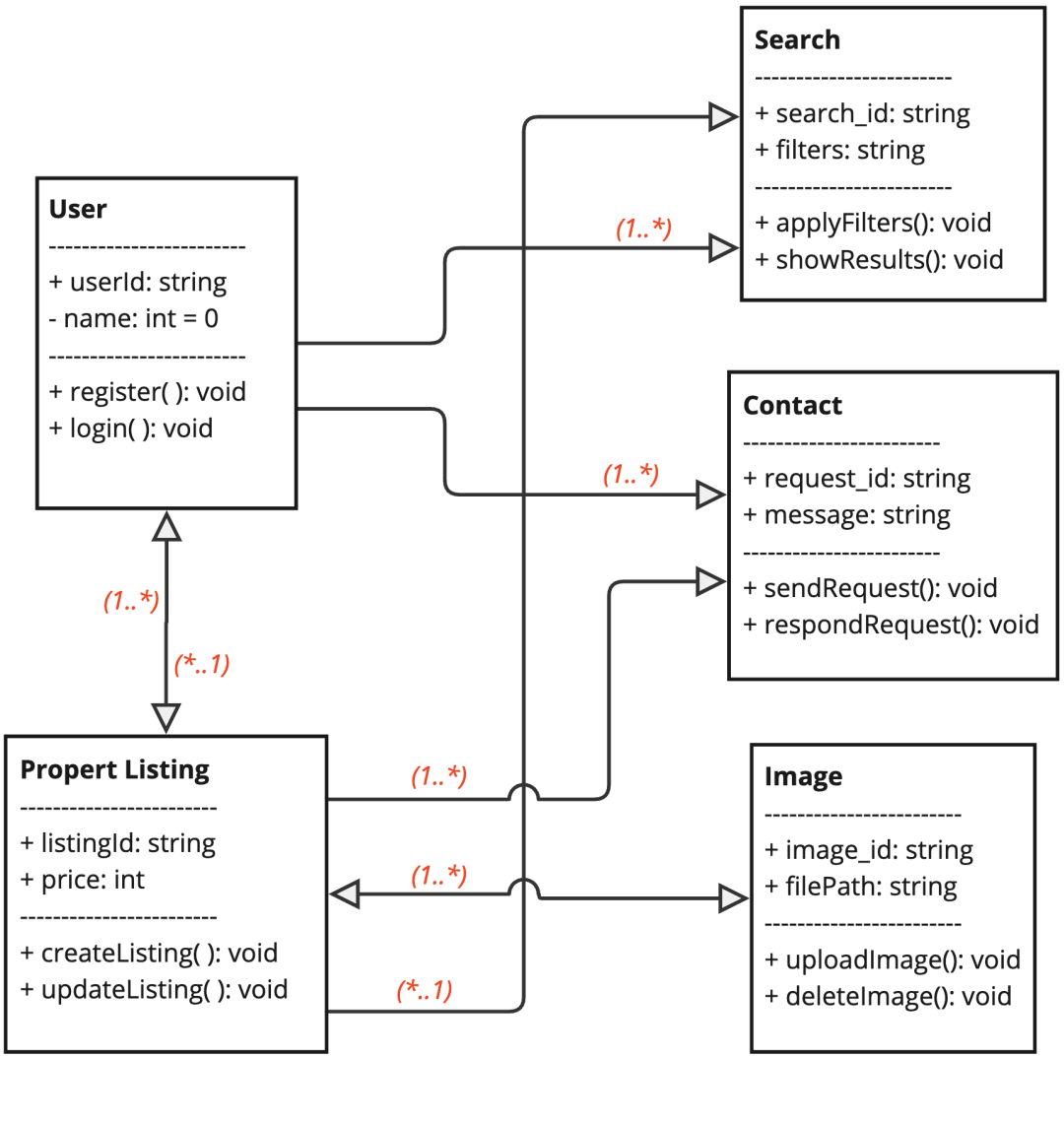

Figure 3. 4: Class diagram

3.4.2.1 Class Diagram Description

The class diagram for the real estate system includes five main classes: User, Listing, Image, Inquiry, and Search. The User class represents individuals interacting with the system, having attributes like `userID` and `name`.

A User can create multiple Listings, submit multiple Inquiries, and perform multiple Searches, forming one-to-many relationships with Listing, Inquiry, and Search.

The Listing class represents properties available for sale or rent, with attributes like `listingID`, `price`, and `location`. Each Listing belongs to one User, and can have multiple Inquiries and Images, creating many-to-one relationships with User, Inquiry, and Image, and a many-to-one relationship with Search.

The Image class handles images linked to listings, with attributes like `imageID` and `url`. Multiple images can belong to a single listing, forming a many-to-one relationship with Listing.

The Inquiry class represents user inquiries about listings, containing attributes like `inquiryID` and `message`. Each Inquiry is associated with one User and one Listing, forming many-to-one relationships with both.

The Search class enables users to search for properties, based on filters like price and location. Each Search can return multiple Listings, forming a one-to-many relationship between Search and Listing, and a many-to-one relationship with User.

4. IMPLEMENTATION

The implementation phase of the real estate system involves translating the design into a working application. This section details the core components, technologies, and strategies used to develop the system. The following sub-headings describe the implementation of various modules, from system architecture to user interface, backend functionality, and integration.

4.1 System Architecture

The real estate system follows a client-server architecture, where the client-side (frontend) interacts with the server-side (backend) to fetch data and manage user interactions. The frontend is built using **React** and **Vite**, ensuring a responsive, dynamic user interface. The backend is powered by **Node.js** with **Express** for routing and handling API requests, while the data is managed in a **MongoDB** database, ensuring scalability and performance. The system architecture is designed to support key features such as property listings, user management, image uploading, and inquiries.

4.2 Database design

The database schema is designed to support the core entities of the system, including Users, Listings, Inquiries, Images, and Searches. Each table is optimized for the relationships between these entities:

- **User**: Stores details of users such as user ID, name, and contact information.
- **Listing**: Contains property details, including listing ID, price, location, and owner (user).
- **Inquiry**: Stores user inquiries related to listings, with attributes such as inquiry ID, message, and user ID.
- **Image**: Holds image data associated with property listings, including image ID and URL.
- **Search**: Tracks search queries performed by users, including filters like price range, location, and property type.

4.3 Backend development

The backend of the real estate system is built with Node.js and Express. The main features of the

backend include:

- **User Authentication**: Using JWT (JSON Web Tokens) for secure user login and registration.
- **Listing Management**: Allows users to create, update, and delete property listings.
- **Inquiry Handling**: Users can send inquiries regarding properties, and property owners can respond.
- **Search Functionality**: Allows users to filter and search for properties based on various criteria such as price, location, and type

4.4 Frontend development

The frontend is developed using React and Vite for a fast, dynamic user experience. The key features of the frontend include:

- **User Interface**: Clean, modern UI with easy navigation for browsing listings, submitting inquiries, and managing user accounts.
- **Responsive Design**: The app is fully responsive, ensuring a seamless experience across devices (desktop, tablet, mobile).
- **Image Upload**: Users can upload images for their property listings, with progress indicators and error handling.

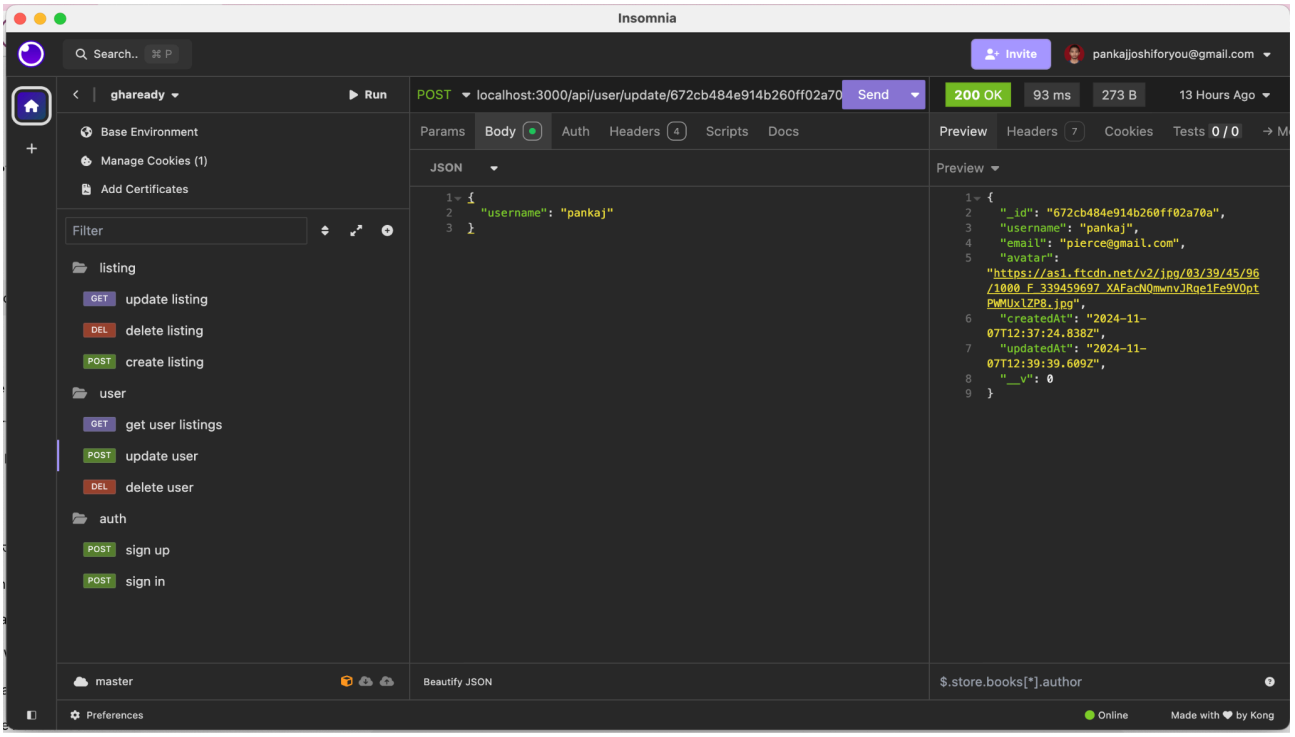

Figure 4.I1: API Testing with Insomnia

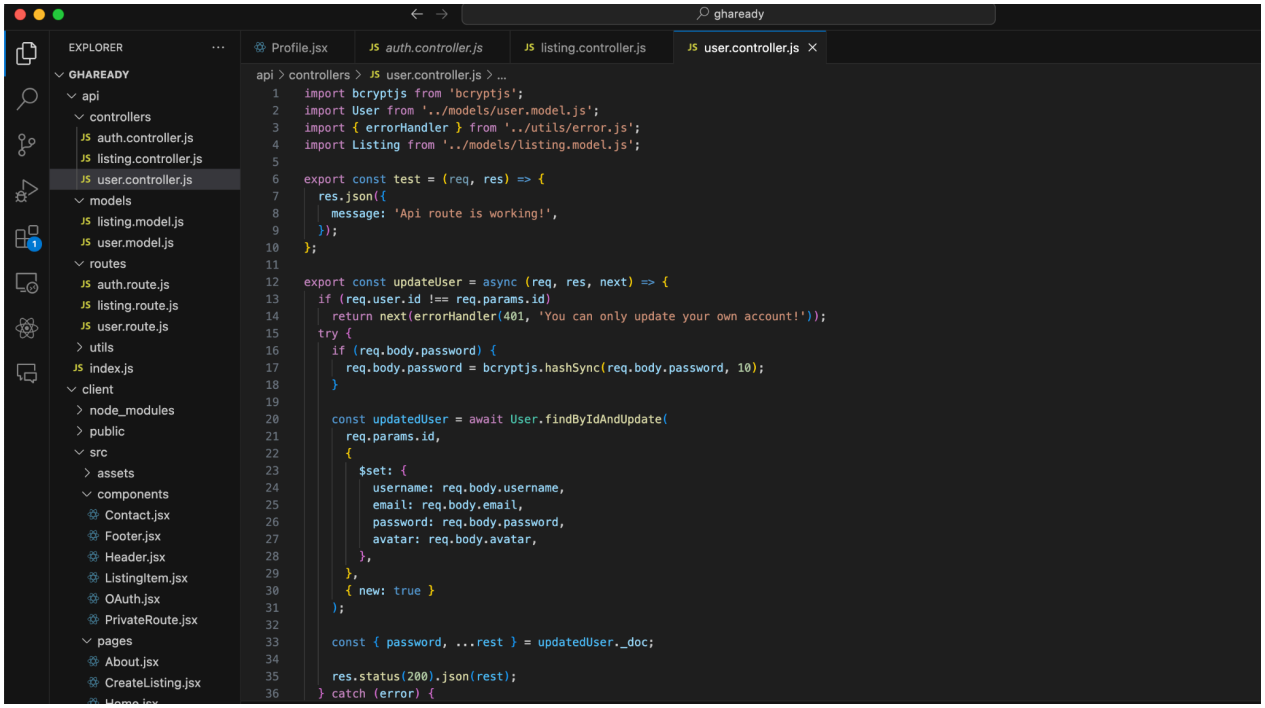

Figure 4.I2: User Controller API

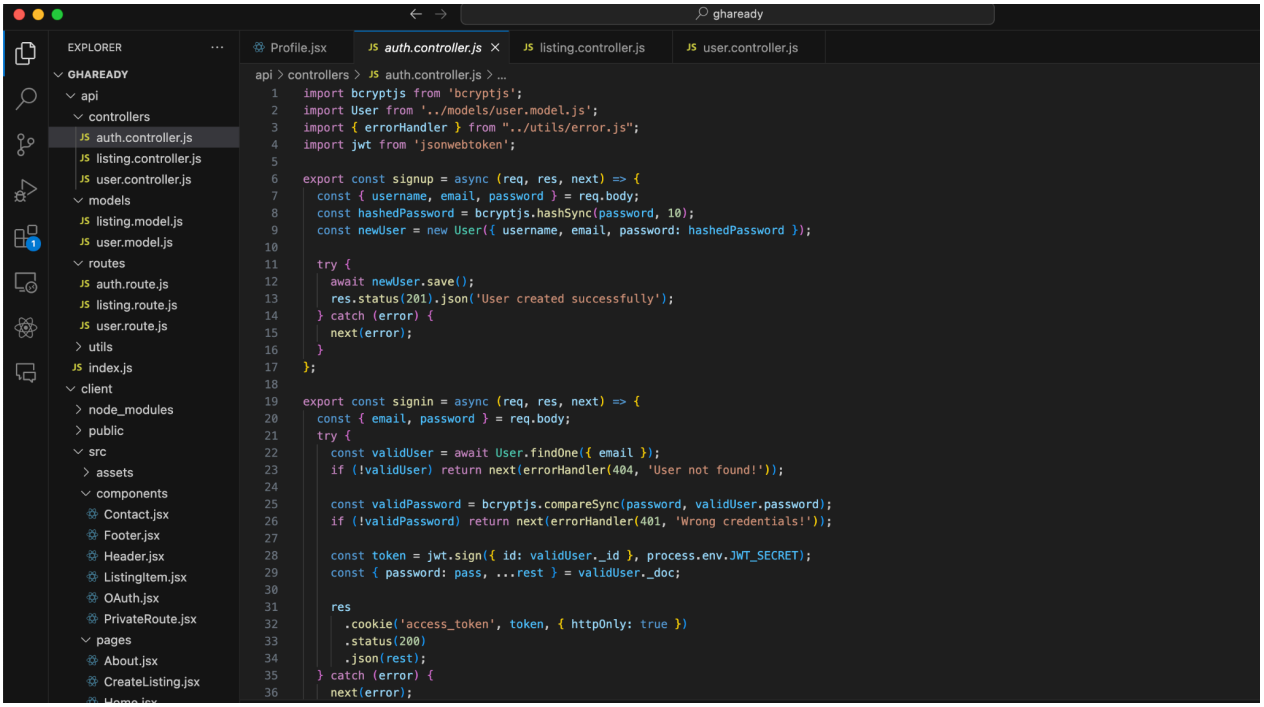

Figure 4.I3: Authentication controller API

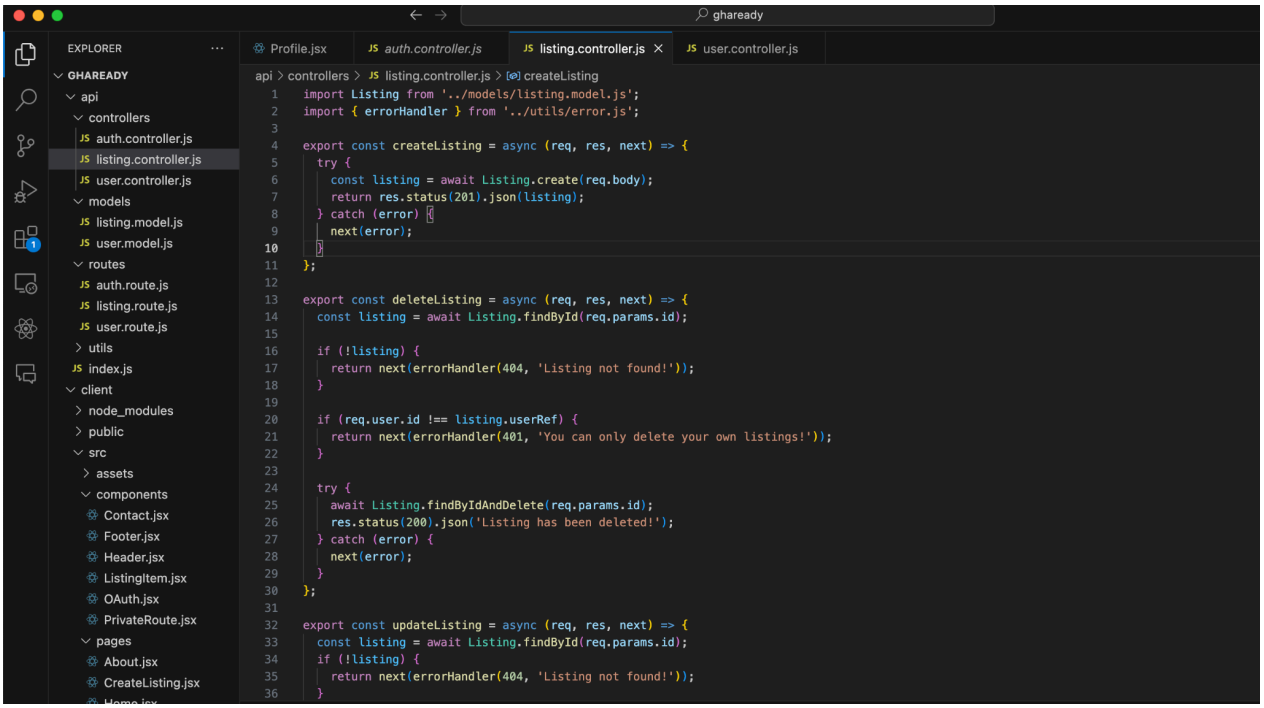

Figure 4.I4: Listing controller API

5. RESULT

The main goal of the real estate system is to offer users an intuitive platform for property searches, listing management, and seamless interactions. Users can filter properties by criteria such as price, location, and type, receiving relevant results dynamically. Each listing includes detailed information like price, description, images, and location, aiding informed decision-making.

The system also allows users to submit inquiries, enabling direct communication with property owners or agents. Property owners can manage their listings by creating, updating, or removing entries and images, ensuring accuracy and relevancy.

With secure user authentication, personalized sessions, and real-time updates on listings and inquiries, the app provides a comprehensive and efficient property search experience. Overall, the system meets its objectives by delivering dynamic search results, easy listing management, and fostering communication between users and property owners.

6. CONCLUSION

This project, Real Estate App, was developed as part of the B.E. Computer Engineering ‘Minor Project’ at Far Western University, SoE. It successfully provides a comprehensive platform for property searches, listing management, and user interaction. Using React, Node.js, and MongoDB, the app allows users to easily filter and search for properties, while property owners can manage their listings and respond to inquiries. The use of JWT for secure authentication and Multer for image handling ensures a seamless and secure user experience.

Leveraging modern tools and technologies like HTML, CSS, Tailwind CSS, Express.js, Vite, Git/GitHub, and VSCode, the project was developed efficiently. Tools like Miro, Insomnia, and Google Docs also contributed to effective planning and execution.

While there were challenges in optimizing search features and managing real-time data, the system meets its objectives and provides a functional, user-friendly real estate platform. This project has been a valuable learning experience, successfully fulfilling the requirements of the minor project in the course.

7. LIMITATION

While the Ghaready - Real Estate App offers a comprehensive platform for property searches, listings management, and user interactions, there are some limitations that should be noted:

Limited Search Filters: Although users can filter properties by price, location, and type, additional filters such as property size, amenities, and proximity to important landmarks could enhance the user experience.

- Search Performance: With a large number of property listings, the search functionality may experience performance issues, especially with complex queries. Optimizing search algorithms and implementing caching mechanisms would help address this limitation.

- Real-Time Updates: The app relies on periodic refreshes for updating property listings and inquiries. Real-time updates for new listings or changes to existing ones could improve user engagement and responsiveness.

- Scalability: As the number of users and listings grows, the app may face challenges in maintaining performance and scalability. Future enhancements could include optimized database queries and more robust infrastructure.

8. FUTURE ENHANCEMENTS

Comprehensive Search Filters, Real-Time Updates, and Image Handling Improvements:
To address limitations in search functionality, **advanced search filters** can be integrated, allowing users to search based on property size, amenities, and other parameters like school proximity, parking availability, or pet-friendliness. Real-time updates could ensure that users are immediately notified of new listings, price changes, or inquiries, improving engagement and reducing the reliance on periodic refreshes. Furthermore, the image upload system can be enhanced to handle higher resolution images and larger file sizes without compromising the app's performance.

Banking Service Integration & Payment Transactions:
Adding banking service integrations for direct **payment transactions** would allow users to securely make payments for properties, deposits, or rents through the app. Features such as **EMI (Equated Monthly Installment) calculations** would assist users in assessing financing options based on their budget, streamlining the buying or renting process.

Blockchain Technology for Smart Contract Signing:
Integrating **blockchain technology** would allow for secure, transparent, and tamper-proof **smart contracts** for property transactions. This would eliminate the need for intermediaries, enabling automatic execution of agreements once terms are met, making property deals faster, more secure, and more reliable.

Virtual Reality (VR) Property Tours:
By adding **virtual reality (VR)** property tours, users could explore properties from anywhere in a fully immersive experience. With the integration of VR or AR, users could virtually walk through properties, inspecting every corner before making a decision. This feature would be particularly useful for users who are unable to visit properties in person, saving time and effort while providing a more engaging way to view listings.

AI-Powered Property Valuation:
Implementing **AI-powered property valuation** would allow users to receive accurate and data-driven price assessments based on various factors like market trends, location, and property features. This could help users make better investment decisions and provide sellers with

realistic pricing guidance, enhancing the overall user experience.

REFERENCES

[1] K. Verma, A. Saxena, and S. Agarwal, "Survey on Real Estate Data Management and Property Recommendation Systems," International Journal of Computer Science and Engineering, 2020.

[2] A. B. Gupta, S. S. Mehta, and R. K. Sharma, "Real Estate Application Framework for Secure Transactions and EMI Calculations," International Journal of Digital Transactions and Real Estate, Nov 2022.

[3] R. K. Jain, P. R. Sharma, and M. N. Singh, "Integrating Blockchain for Secure Property Transactions in Real Estate Apps," International Conference on Digital Transformation and Real Estate Technology, 2021.

[4] J. Das, S. P. Singh, and S. S. Tyagi, "Artificial Intelligence in Real Estate: Market Prediction and Property Valuation Models," Journal of Real Estate Finance, May 2022.

[5] T. R. Lee, C. T. Huang, and D. Y. Wu, "Augmented Reality for Virtual Property Tours in Real Estate Applications," International Journal of Virtual Reality and Real Estate, Jan 2023.

[6] M. R. Shah, A. M. Desai, and R. S. Patil, "Real-Time Property Search and Advanced Filtering in Real Estate Platforms," International Journal of Web and Mobile Development, Mar 2023.

APPENDIX

Appendix A

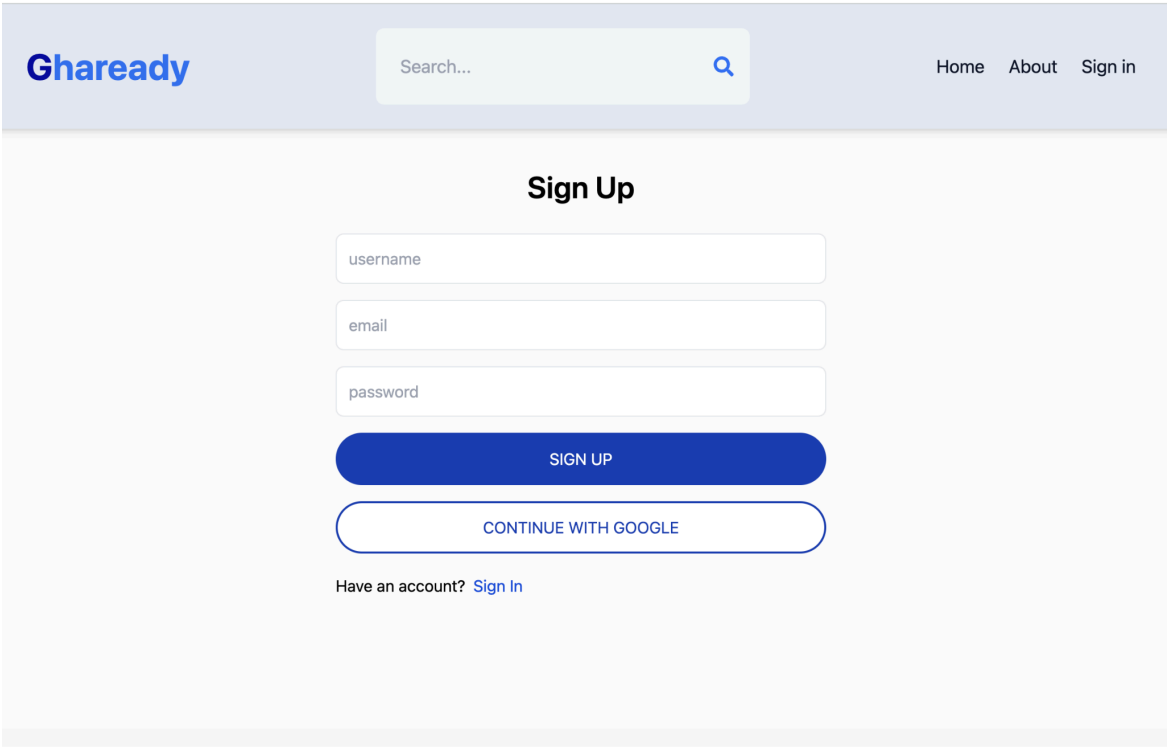

Figure A1: Sign Up page

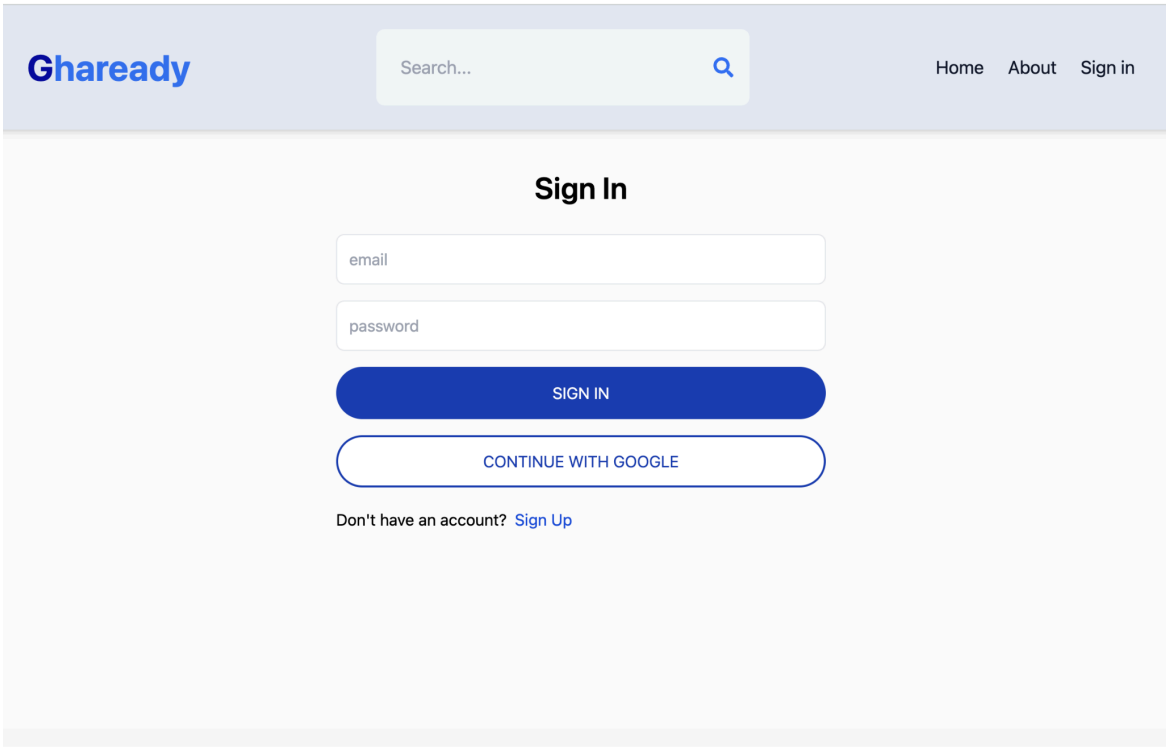

Figure A2: Sign in page

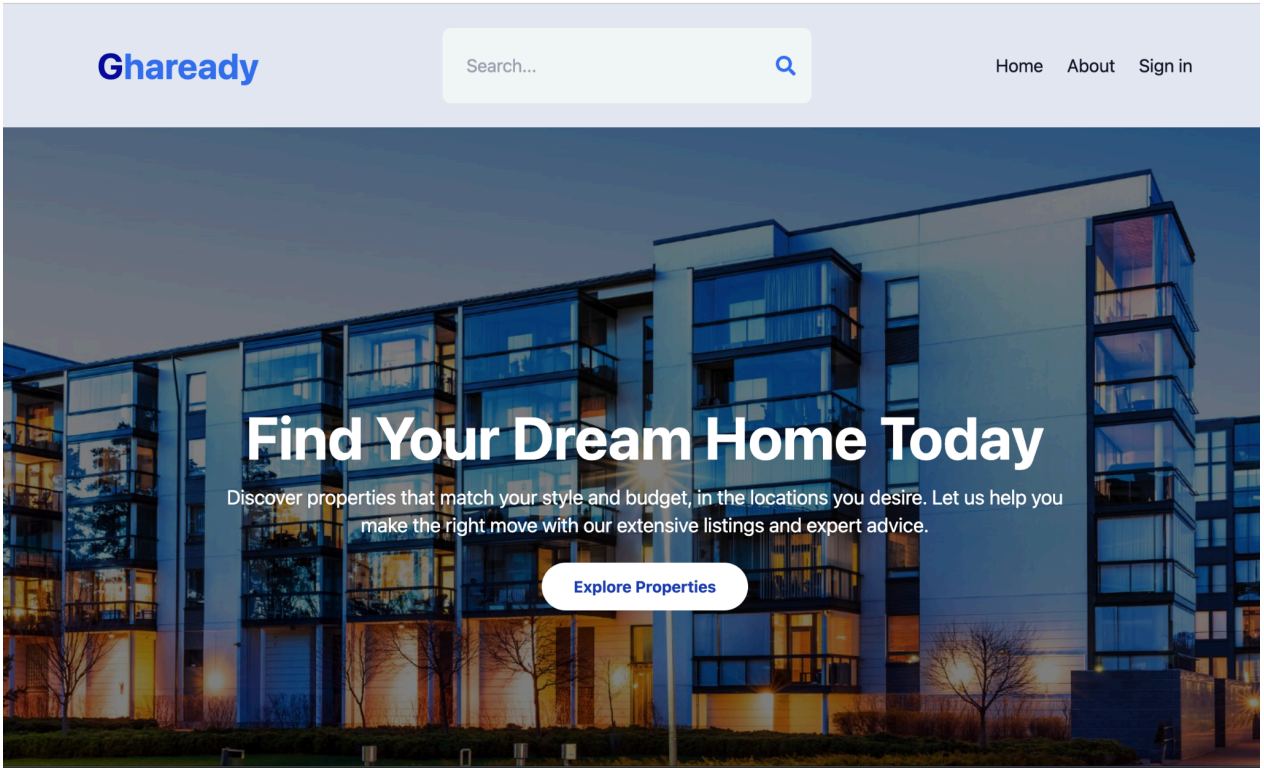

Figure A3: Home page

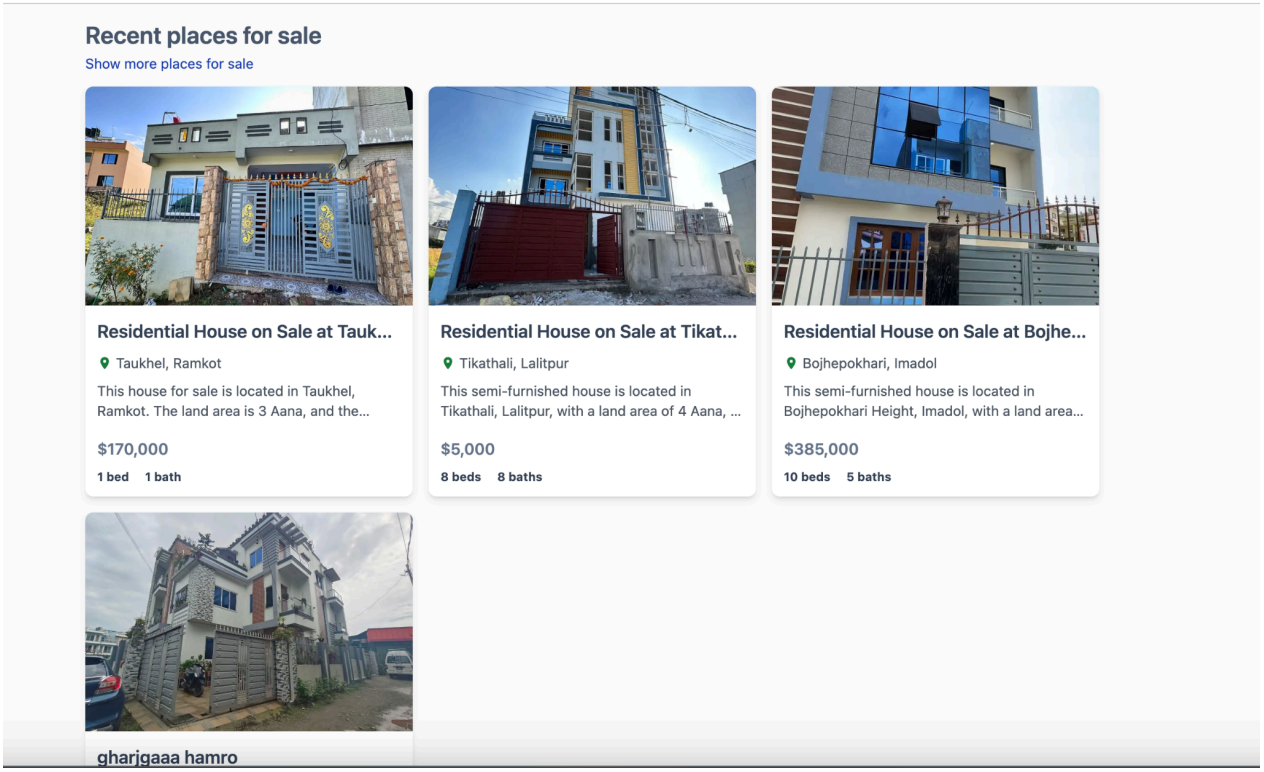

Figure A4: Listings

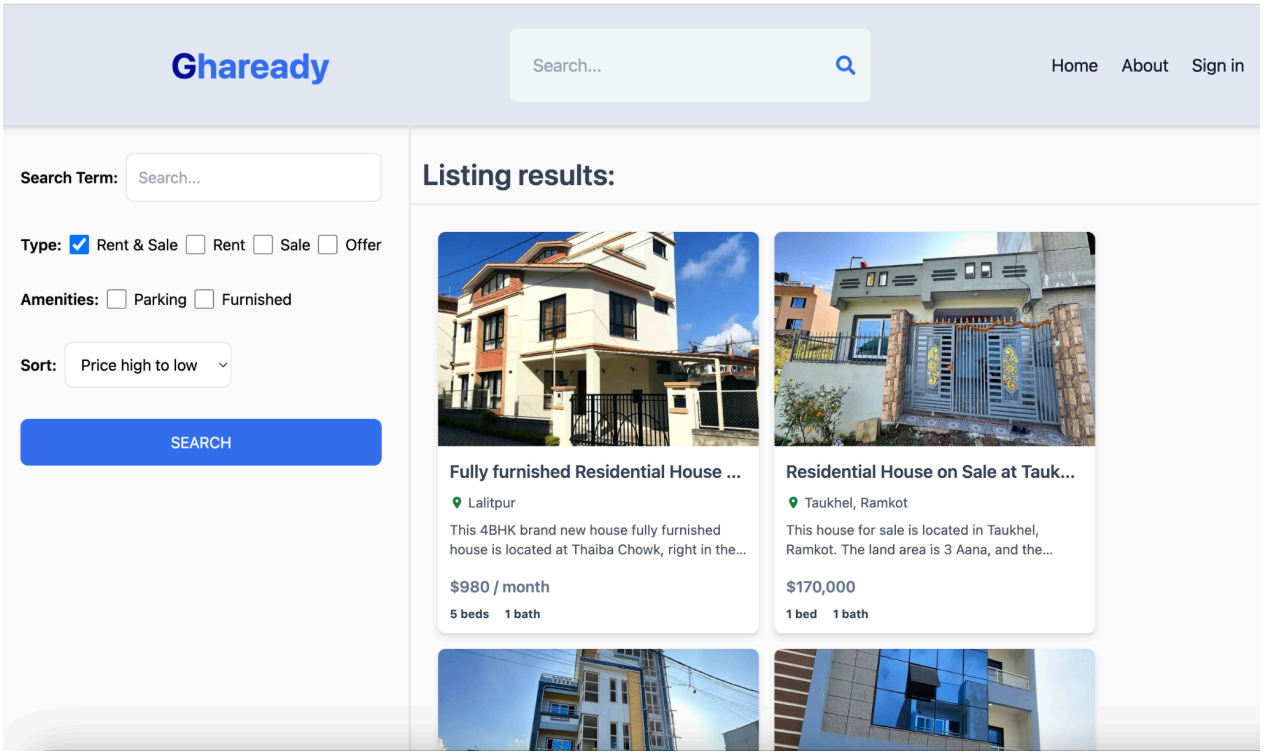

Figure A5: Search page

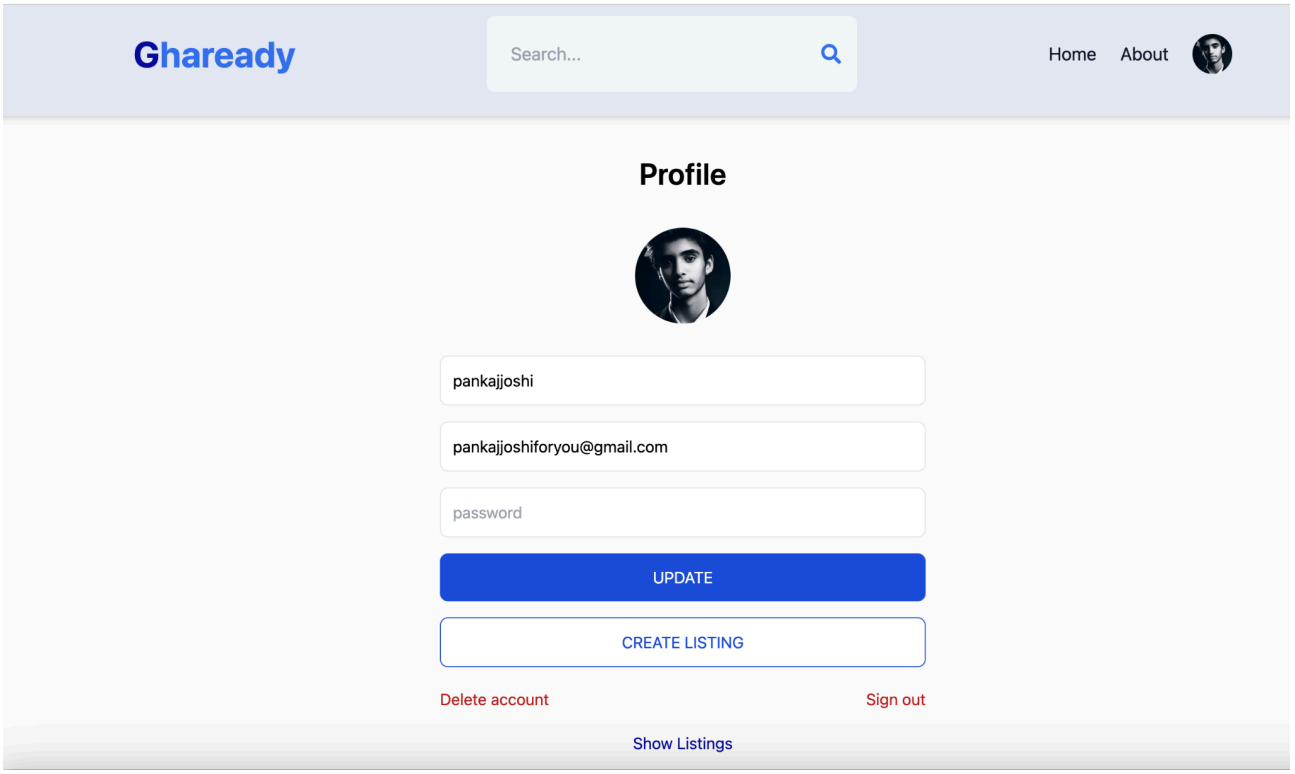

Figure A6: Profile page

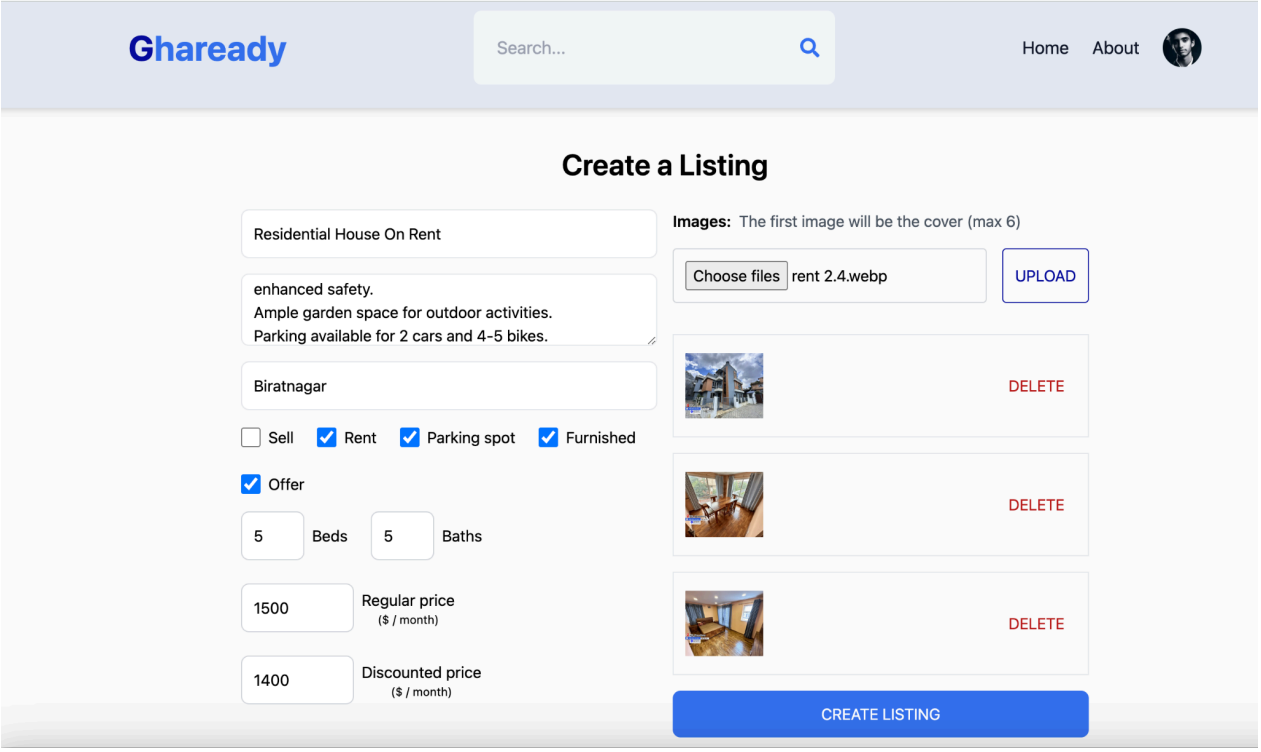

Figure A7: Create listing