DEVELOPMENT OF A PROPERTY RENTAL WEBSITE USING REACT FRAMEWORK WITH CONTENT-BASED AND COLLABORATIVE FILTERING RECOMMENDATION TECHNIQUE

BY

CHAN Zi Bin

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology
(Kampar Campus)

FEB 2025

ACKNOWLEDGEMENTS

I would like to sincerely express my thanks and appreciation to my supervisors, Ms Tseu Kwan Lee who has given me this opportunity to take part in this project. This act as my first step to full-stack application development integrating with recommender models. A million thanks to you.

COPYRIGHT STATEMENT

This Final Year Project is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ABSTRACT

The rapid digitalization of the property rental market has highlighted significant limitations in existing platforms, particularly in their ability to provide personalized property recommendations and streamline the rental search process. This project proposes the development of a modern property rental website that implements a hybrid recommendation system combining content-based and collaborative filtering techniques. The platform aims to address the limitations by providing personalized property suggestions based on user preferences, behavior patterns, and property attributes. This project is built using Next.js, the backend infrastructure utilizes Supabase for database management and storage solutions. The hybrid recommender system analyzes both property attributes and user interaction data to generate relevant recommendations, improving the property discovery process for potential tenants. The project's preliminary implementation has established core functionalities, while future development will focus on implementing the recommendation engine and administrative features.

Area of study: Web development, recommender systems

Keywords: Property rental platform, React framework, Next.js, Content-based filtering, Collaborative Filtering

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
ERD	Entity Relationship Diagram
JS	JavaScript
OAuth	Open Authorization
SMTP	Simple Mail Transfer Protocol
UI	User interface
SQL	Structured Query Language
NCF	Neural Collaborative Filtering
NDCG	Normalized Discounted Cumulative Gain
HR	Hit rate

CHAPTER 1 INTRODUCTION

1.1 Problem Statement and Motivation

In recent years, the digitalization of property rental markets has seen significant growth, yet many existing platforms still employ traditional, static approaches to property listings that fail to meet modern user expectations. While property listing websites have become common place, many continue to operate with basic functionality that simply displays identical listings to all users without personalization. Yang et al. [1] emphasized this disconnect in their research on web house page listing cues, revealing that both high-task-relevant cues (HTRCs) like property details and low-task-relevant cues (LTRCs) such as website design significantly influence users' decision-making processes, with LTRCs surprisingly showing greater impact on purchase intention.

Despite the availability of various property rental platforms, current systems face several critical limitations. The lack of personalization in property discovery forces users to navigate through extensive listings irrelevant to their preferences, resulting in inefficient search experiences and low conversion rates. Users logging in with different accounts see identical property listings, regardless of their unique preferences and search history. This limitation mirrors the challenges observed in other e-commerce sectors that have since evolved to embrace more sophisticated technological solutions.

Modern customers increasingly expect personalized experiences in their online interactions, similar to what they encounter on other digital platforms like social media or e-commerce platform. This kind of trend motivates me to build this project. This project proposes the development of a modern property rental website using the React framework, incorporating both content-based and collaborative filtering techniques. The solution will track user interactions, analyze property attributes, and leverage machine learning algorithms to create personalized user experiences. By implementing these advanced features, the platform aims to streamline the property search process and provide valuable insights for pricing optimization, creating a more efficient and user-friendly rental marketplace.

1.2 Objectives

The first objective is to **analyze and implement a hybrid recommendation system that combines content-based and collaborative filtering techniques**. This recommendation system aims to improve content discovery and delivery by analyzing both property attributes and user behavior patterns. Content-based filtering will match properties based on specific features and characteristics, while collaborative filtering will leverage user interaction data to identify patterns and preferences across similar users. By combining these approaches, the system can provide more accurate and personalized property suggestions, helping users discover relevant listings more efficiently and potentially reducing the time needed to find suitable properties.

The second objective is to **develop a comprehensive rental property website with essential features using the React framework**. This involves creating a modern, responsive web application that provides core functionalities such as user authentication, property listing management, and interactive search capabilities. By leveraging React's component-based architecture and server-side rendering capabilities through Next.js, the platform aims to deliver a seamless user experience across different devices while maintaining optimal performance and scalability. The system will include integrated property management tools for landlords to create and manage listings and features for users to track saved properties and viewing history.

The third objective is to **evaluate the effectiveness of the proposed hybrid recommendation system in improving property discovery**. This involves conducting systematic testing and analysis to measure the system's performance in terms of recommendation accuracy and efficiency in property matching. The evaluation will include quantitative metrics such as precision and recall rates of property recommendations. This assessment will validate the effectiveness of combining content-based and collaborative filtering approaches in the context of property rental recommendations.

1.3 Project Scope and Direction

The deliverable of this project is a modern property rental website that implements a hybrid recommendation system combining content-based and collaborative filtering techniques. This platform is developed to address the limitations of traditional property

rental websites, which often display identical listings to all users without personalization. The system aims to streamline the property search process by providing personalized recommendations and enhancing the user experience through intelligent filtering and sorting capabilities.

This project implements functionalities including property listing management, user authentication, profile management and a recommender model. Key features include property search, interactive map integration, wishlist functionality and a multi-role user system supporting regular users, landlords and administrators. The platform will implement a hybrid recommender system that analyzes both property characteristics and user behavior patterns to generate personalized recommendations.

The system will be built using the Next.js framework, with Supabase handling the backend infrastructure for database management and storage solutions. The primary users of this platform will be both property seekers and landlord, with a focus on creating an efficient and user-friendly rental marketplace that matches properties with potential tenants based on their preferences and behavior patterns.

There are several features and functionalities fall outside the scope of this project to maintain focus on the core recommendation and listing capabilities. The project will not implement transaction-related features such as online payment processing, rental contract generation or booking management systems. Internal direct messaging or chat functionality between users and landlords will not be included. Instead, contact information will be provided for external communication like WhatsApp or email. The platform will not handle rental agreement processing or legal documentation. While basic user authentication and authorization will be implemented, extensive security features such as two-factor authentication or advanced encryption are not included. The recommendation system will focus on basic user preferences and behaviors, excluding more complex factors such as socioeconomic analysis or long-term market predictions.

1.4 Contributions

From a technical perspective, this project improves system scalability through efficient recommendation algorithms. Traditional search methods with linear complexity O(n) become inefficient as data volume grows. By implementing content-based and collaborative filtering techniques, the system can deliver relevant properties directly to

users, reducing search time and server load while maintaining performance as the database expands.

From a user perspective, this project enhances the user experience by addressing the common frustration of scrolling through irrelevant listings. The personalized recommendation system helps users find suitable properties more efficiently.

From a landlord perspective, this project provides them a platform and encourage them to list their properties online. With the recommender system, it can enhance the visibility of the property when its attributes match a prospective tenant's preferences.

Recommendation systems have proven highly effective in various domains, such as e-commerce to content streaming services, demonstrating their potential to enhance user engagement and satisfaction. By applying these technologies to the property rental sector, platforms can better match properties with potential tenants based on both explicit preferences and implicit behavior patterns, thereby improving the efficiency of the rental market.

1.5 Background Information

The development of online rental market is one of the biggest changes that have taken place in the last years regarding searching for and renting accommodation. These platforms provide landlords the opportunity to advertise vacant spaces in units, while also providing tenants with the ability to search for units to rent by comparing different options [2]. Popular websites like Craigslist, Zillow, and Apartments.com hold common features including searchable listings, photo galleries, virtual tours and direct messaging between tenants and landlords or agents [2]. Thus, the transition to online platform has broadened renters' informational opportunities and options as compared with offline methods.

Furthermore, the necessity for dedicated online platforms to promote rental properties has been underscored by research. Monteverde et al. note that homeowners require specialized websites to effectively market their rental houses, as many existing platforms primarily cater to hotel listings [3]. This gap in the market has led to the development of innovative solutions, such as the Web-Based Rental House Smart Finder, which streamlines the process of searching for rental properties and enhances communication between landlords and potential tenants [3]. Such advancements not

only improve user experience but also contribute to a more efficient rental market. The quality of rental property websites is also a significant factor influencing user engagement and satisfaction. Research has shown that the effectiveness of these platforms can directly impact the time and effort required for users to find suitable rental options [4]. As such, ongoing assessments of website quality are essential for enhancing user experience and ensuring that these platforms meet the evolving needs of the rental market.

Moreover, integration of recommendation systems into e-commerce and digital service platforms has become increasingly prevalent across various industries. Recommendation systems utilize data analysis and machine learning techniques to suggest relevant items or services to users based on their preferences, behavior patterns, and historical interactions. In the context of e-commerce, recommender systems can lead to increased sales by suggesting products that align closely with user interests, thus fostering a more personalized shopping experience [5]. In the context of property rentals, these systems can significantly enhance the user experience by helping tenants find suitable properties that match their specific requirements, such as location preferences, budget constraints, and desired amenities.

The evolution of online rental platforms, combined with advances in recommendation systems and web technologies, has created new opportunities for improving the property rental market. The following chapter will examine current market solutions and identify areas where innovative features like recommendation systems can address existing limitations and improve the property rental experience for both tenants and landlords.

CHAPTER 2 LITERATURE REVIEW

2.1 Existing Property Rental Websites

This section discusses the current existing similar websites including comparison of the strength and weakness for each of existing similar websites in Malaysia.

2.1.1 iBilik.my [6]

iBilik.my is one of the most popular websites for housing room rental and property listing in Malaysia. Commencing in 2004 and has since then proved to be a useful guide for user and landlords in the Malaysian rental market. This case study reviews the key features and limitations of iBilik.my.

On iBilik.my, like other traditional rental platforms, users begin their search by selecting a location, property name or cities, and then refine their search using several search filters. The platform provides personalized filter options, allowing users to specify preferences based on nationality, race, occupation and lease term. The search and filter options are presented in the Figure 2.1.

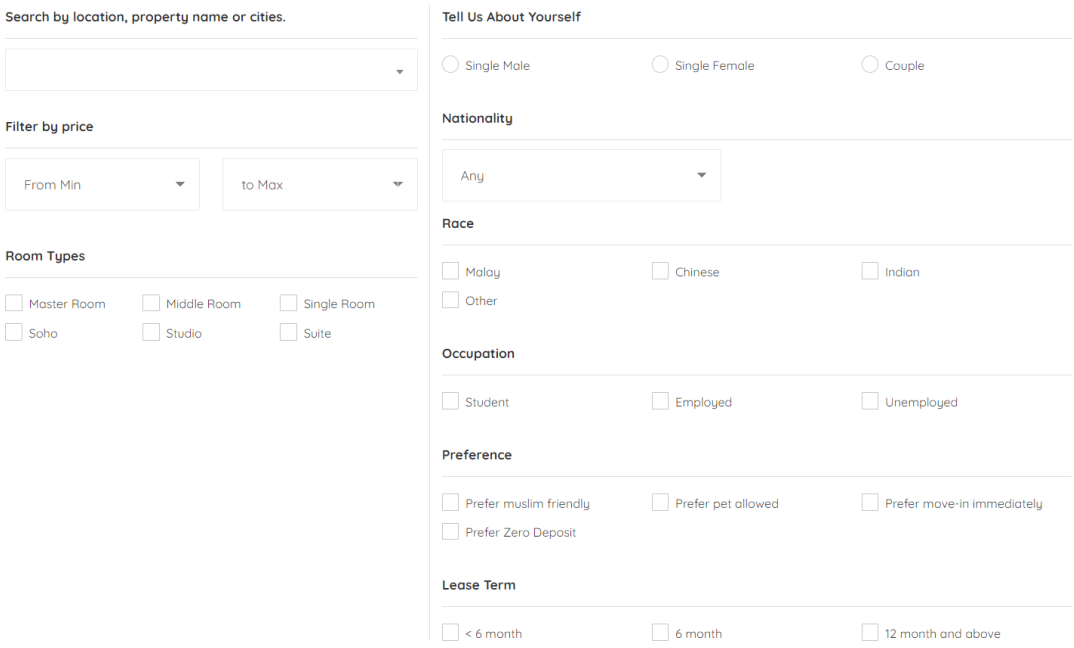


Figure 2.1 Screenshot of iBilik.my's search interface with filter options

The search results are shown in a grid view, presenting essential details such as the rental price, location as well as the primary photo of the property. Each listing card also includes icons indicating available amenities and the total views of the property at the top right corner, providing a quick overview of the property's features. Users are able

to access more information about a specific listing by clicking on the listing This includes additional information, images, property description, available facilities and rules and regulations. Figure 2.2 illustrates the grid view of rental listings on iBilik.my.

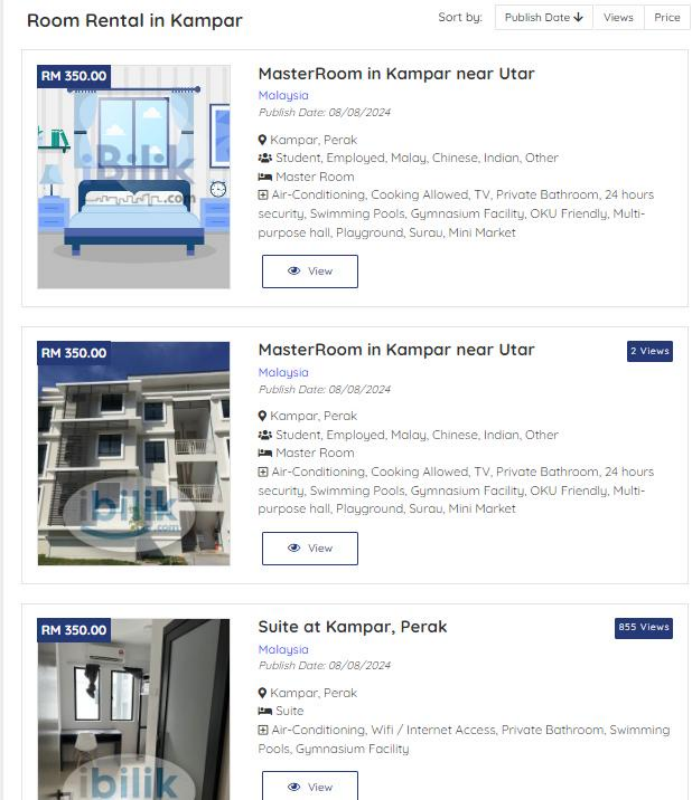


Figure 2.2 Grid view of rental listings on iBilik.my

iBilik.my integrates a map feature where users can find out the location of the listings and other related facilities. It is quite useful for the users who are not familiar with the areas. The map feature of the platform is shown in Figure 2.3.

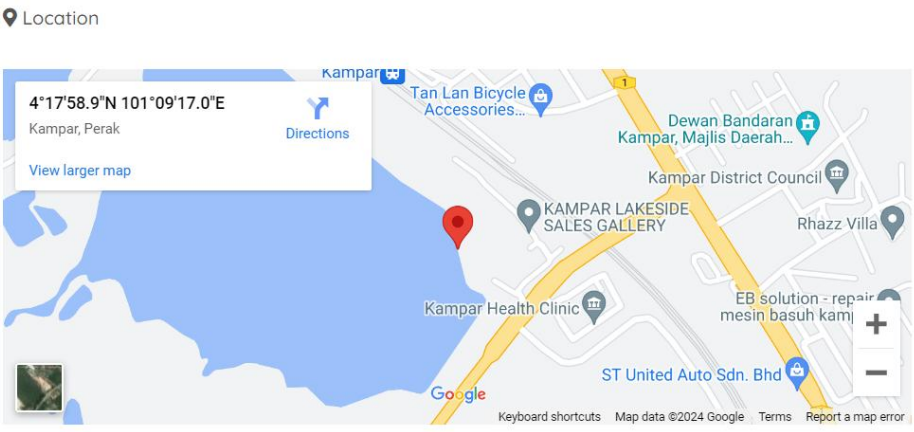


Figure 2.3 Map view of listings on iBilik.my

The overall experience of browsing iBilik.my is generally satisfactory, with notable features such as property view counters and a simple and intuitive user interface. However, several limitations have been identified in the current website. Firstly, despite being a room rental platform, the website lacks proper categorization of property types, which may cause confusion for users searching for specific type of accommodations. The small text size of the website also affects the readability and user experience. Furthermore, after testing all the features, it appears that no recommender system has been implemented. This is evident as the website displays same property listings on the first page of the listings view regardless of user behavior, even when accessing the platform through different accounts, performing various searches, or revisiting the website.

2.1.2 PropertyGuru Malaysia [7]

PropertyGuru.com.my is a Malaysia's leading property site for users to search properties for investing, buying or looking a place to rent. Similar to other rental listing websites, PropertyGuru.com.my features a search bar with three filter options on its landing page, as shown in Figure 2.4. Users are required to enter a location to begin the search. The 'All Residential' option refers to the property type such as landed, apartment, condo and others. 'Any Price' refers to the range of price and bedroom refers to the number of bedrooms of the property.

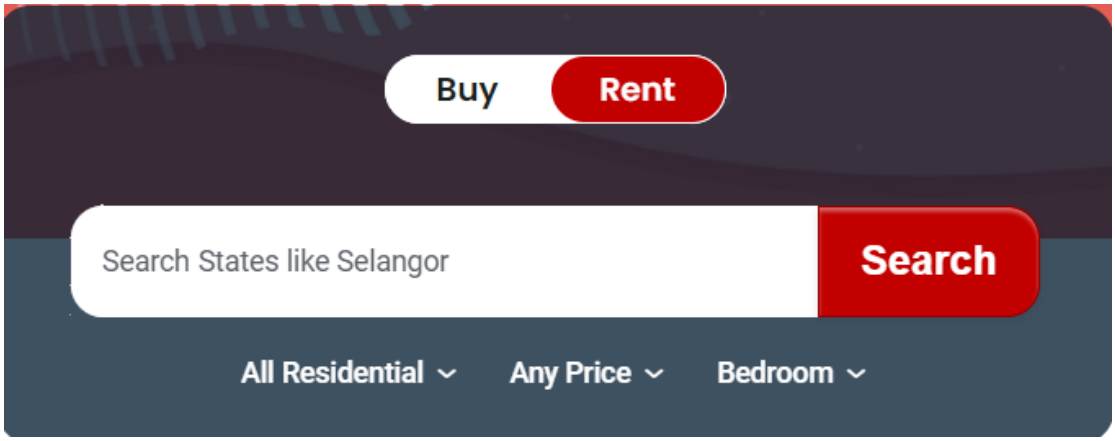


Figure 2.4 PropertyGuru.com.my search interface

Similar to iBilik.my, the search results are presented in a grid view and the main information includes the rental price, location and the primary photo of the property. The listings can be sorted by publish date, price, size and price per square foot (PSF).

Each listing card also includes tags and icons indicating available number of beds, size, PSF and furnishing status, providing an overview of the property. Users can get more information about a specific listing by clicking on the listing, and get more information, images, description of the property, the facilities that are available and other rules and regulations.

One key feature of PropertyGuru is the ‘Show Map’ function. When users trigger the feature, the search results will be categorized and pinpointed on the map as shown in Figure 2.5. This allows users to have an overview of each property’s location and the distances among themselves.

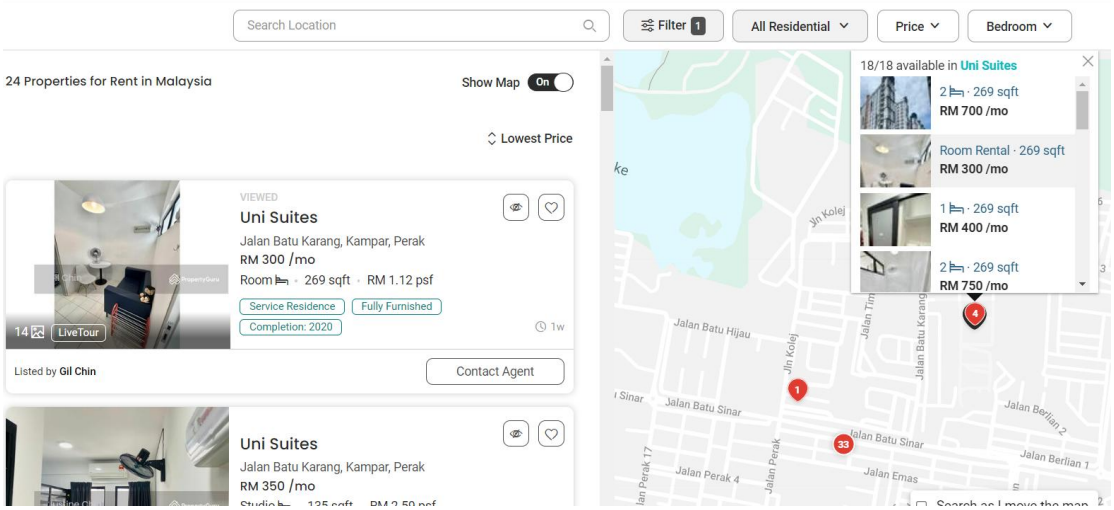


Figure 2.5 PropertyGuru Malaysia Show Map feature

PropertyGuru Malaysia presents an impressive user interface with a comprehensive and well-structured property categorization system. The platform notably features a two-level property type classification, such as its Apartment/Condo/Service Residence category, which is further subdivided into specific types including Condominium, Apartment, Flat, Penthouse, Service Residence, Studio, Duplex, and Townhouse Condo. However, several limitations such as the platform suffers from information overload, presenting users with an overwhelming number of property listings alongside additional features such as loan services, property guides, find agents and budget calculators. While these supplementary tools are beneficial, they may distract users from their primary goal of finding rental properties. Although PropertyGuru implements a preference system in user profiles that collects information about lease terms, occupancy requirements, and budget, its functionality appears limited to basic notification purposes rather than intelligent recommendation. Testing reveals that the

platform likely lacks a sophisticated recommender system, as property listings remain consistently sorted by publication date regardless of user preferences or account differences.

2.1.3 EdgeProp [8]

EdgeProp is another online property platform in Malaysia, designed to facilitate property transactions and empower consumers with essential real estate information. Similar to others website, users begin their search by utilizing filters to view the property listings. A unique feature of its filter is it includes MRT/LRT lines options. However, the purpose of this option appears unclear, no noticeable differences observed in the search results when selecting the options in MRT/LRT lines. Figure 2.6 shows the search interface of EdegeProp.my.

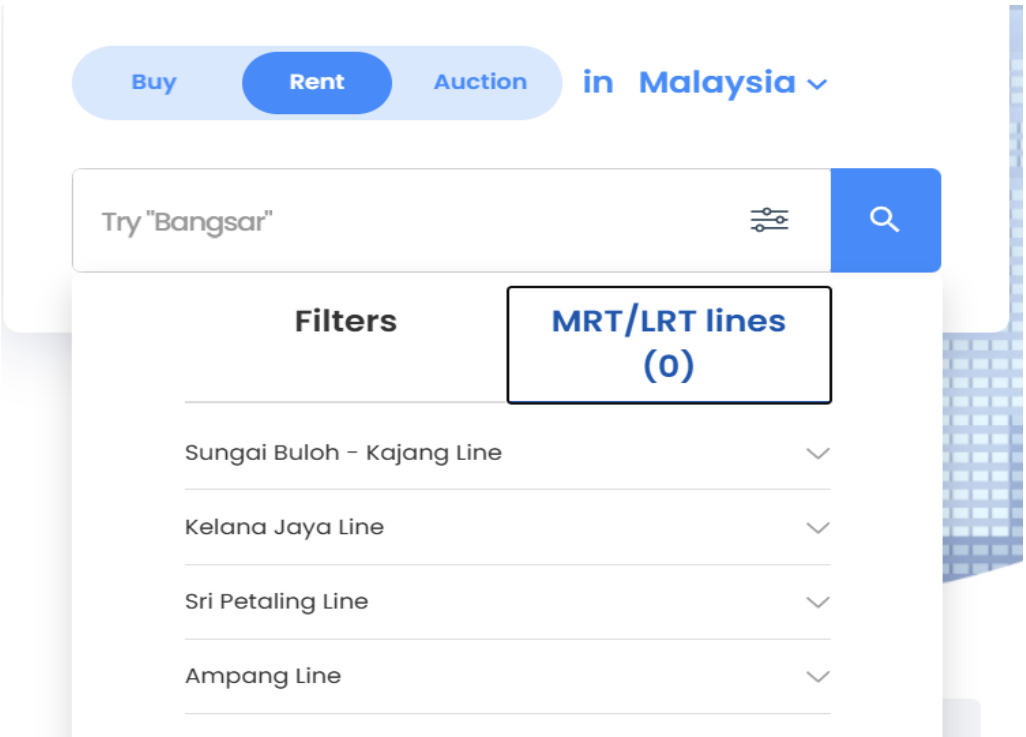


Figure 2.6 Search interface of EdgeProp.my

A key feature of this website is the application of chatbot. This website implements a question and answering chatbot named Buddy to assist users on transactions, market value and listing problems. Figure 2.7 Shows the example of conversation result when the user asks the chatbot to “Show me listings for rent in Kampar”.

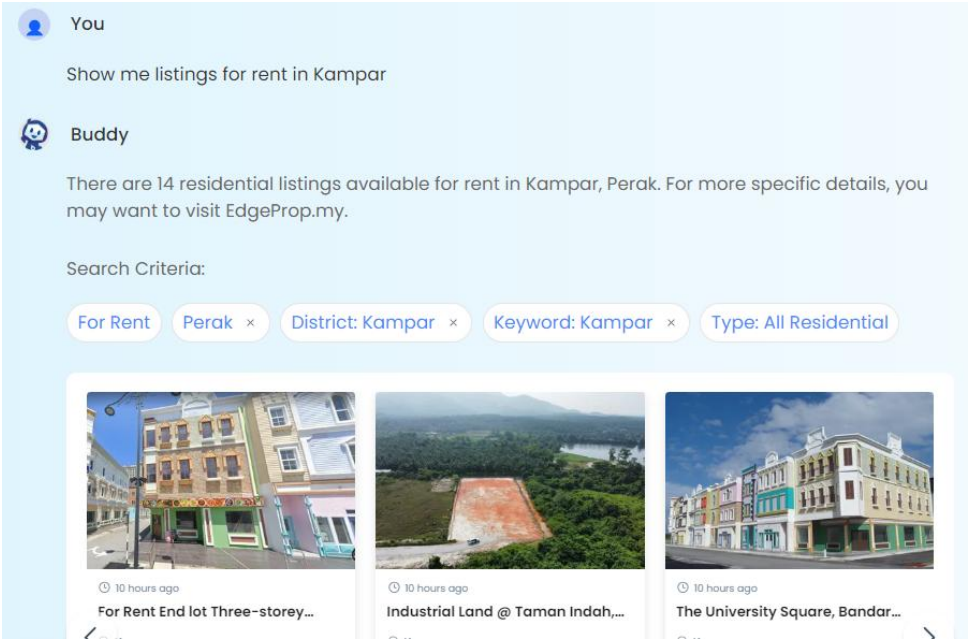


Figure 2.7 Chatbot interface of Buddy

In addition to that, EdgeProp has implemented a new feature called 'Ask EdgeProp' that provides detailed information about specific locations, such as who lives here (demographic data based on income level), what are the typical sizes of homes here and other relevant statistics. The system presents statistical data along with corresponding analysis for each location. Currently the feature only support locations in Kuala Lumpur, Selangor, Johor and Penang. Figure 2.8 shows the result of a location in Johor.

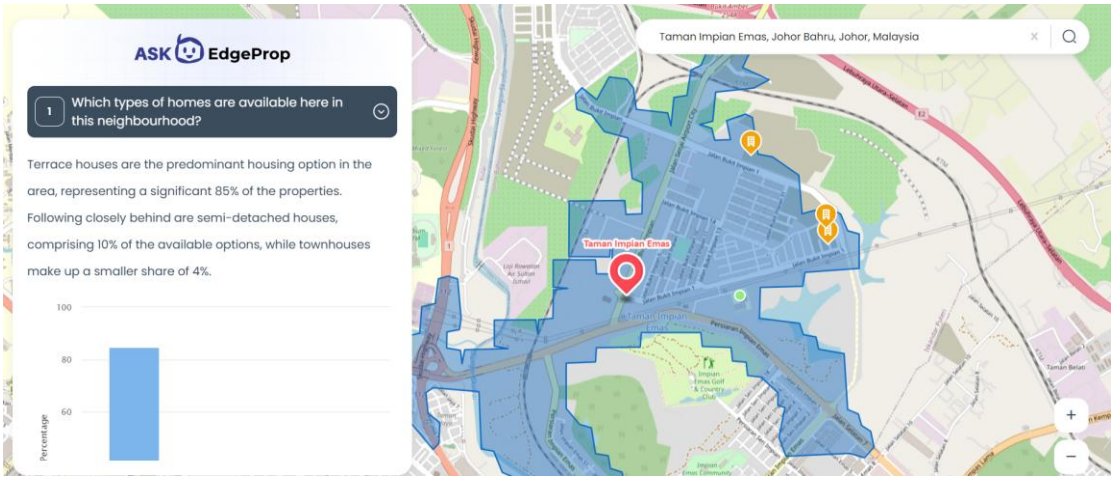


Figure 2.8 Ask EdgeProp interface

The overall experience of using EdgeProp demonstrates a strong focus on providing informative content, featuring relevant property news on its landing page, innovative

features such as the Buddy chatbot and the 'Ask EdgeProp' location analysis tool, which improve the user experience and providing additional insights and assistance. However, several limitations have been identified on the website. The most significant drawback is the slow loading times of the website, including property listing and chatbot responses. Furthermore, while the website remembers user search history and provides auto-fill functionality for previous search terms, it lacks a recommender system as well. This is evident in the landing page's property listings, it shows the properties by published date rather than leveraging user preferences and search history to display relevant properties.

2.1.4 Comparison Between Similar Existing Websites

Table 2.1 Comparison between property rental websites

Functionalities	iBilik.my	PropertyGuru Malaysia	EdgeProp
Search Filters	✓	✓	✓
Map view	✓	✓	✓
Register and login	✓	✓	✓
Wishlist		✓	✓
Search history			✓
Communication between renters and landlords	Whatsapp/Phone	Whatsapp/Phone	Phone/Email
User profile	✓	✓	✓
Property view counter	✓		
Chatbot			✓
Location analysis tool			✓
Newsletter		✓	✓
Alert for new property		✓	✓
Recommender system			
Create property listing	✓	✓	✓
Rating/review system			

Based on Table 2.1, the comparison reveals several common features and differences in the current Malaysian property rental websites. All three platforms share fundamental features such as search filters, map view functionality, user registration, and property listing capabilities. However, some differences exist in their approach to user engagement and property management. EdgeProp stands out with its modern technological integration, being the only platform offering a chatbot and location analysis tool. PropertyGuru and EdgeProp have implemented features like newsletters and property alerts to keep users engaged. A significant observation is that none of the platforms currently implement a recommender system, indicating a clear opportunity for innovation in the market. Furthermore, the communication mechanism between renters and landlords relies heavily on external tools (WhatsApp, phone, email) rather than integrated messaging systems, suggesting a potential area for improvement. The property listing creation process also varies significantly, with PropertyGuru only allows verified agents to create property listing, while iBilik.my and EdgeProp offer more flexible listing options for both agents and individual property owners. These findings suggest that there is substantial room for improvement in terms of implementing features like recommender systems and integrated communication platforms, which could significantly enhance the user experience in the property rental market.

2.2 Frameworks

2.2.1 React

React is a JavaScript library designed for building user interfaces, particularly for single-page applications. It was developed by Facebook and has gained immense popularity due to its component-based architecture, which allows developers to create reusable UI components. This modular approach enhances code maintainability and facilitates the development of complex user interfaces that can efficiently manage dynamic data [9]. React operates on a virtual DOM, which optimizes rendering performance by minimizing direct manipulations of the actual DOM, thus improving application responsiveness.

One of the key features of React is its declarative programming style, which simplifies the process of designing interactive UIs. Instead of detailing the step-by-step procedures for UI updates, developers describe what the UI should look like at any

given point in time, allowing React to handle the underlying updates. This approach reduces the amount of code required and enhances readability and maintainability, making it easier for teams to collaborate on large projects [10].

Furthermore, React's ecosystem is enriched by a variety of tools and libraries that complement its functionality. For instance, state management libraries like Redux and MobX are commonly used alongside React to manage application state in a predictable manner [9]. Additionally, the integration of React with modern JavaScript features, such as ES6 modules, has further solidified its position as a leading framework in the realm of web development [10].

2.2.2 Next.js

Next.js is a React-based framework developed by Vercel that enables developers to build server-rendered applications and static websites with ease. Next.js enhances the capabilities of React by providing features such as server-side rendering (SSR), static site generation (SSG), and API routes, which streamline the development process and improve performance. This framework allows developers to create applications that are not only fast but also optimized for search engines, as server-side rendering provides fully rendered HTML pages to search engine crawlers, improving discoverability [11].

One of the standout features of Next.js is its support for hybrid applications, where developers can choose between server-side rendering and static generation on a per-page basis. This flexibility allows for the optimization of performance based on the specific needs of each page within an application. For instance, pages that require real-time data can benefit from SSR, while static content can be pre-rendered, resulting in faster load times and a better user experience. Additionally, Next.js simplifies routing through a file-based routing system, where the file structure of the application directly corresponds to the routes, making it intuitive for developers to manage navigation [11].

Next.js also integrates seamlessly with various data-fetching methods, allowing developers to fetch data at build time, request time, or even on the client side. This capability enhances the framework's versatility, enabling the creation of dynamic applications that can efficiently handle data from various sources. Furthermore, the framework supports API routes, which allow developers to create backend endpoints directly within the Next.js application, simplifying the architecture by reducing the need for a separate backend server [11].

2.3 Recommendation Systems in Similar Platforms

2.3.1 Airbnb [12]

Airbnb, established in 2008, has revolutionized the accommodation rental market by creating a platform where property owners can list their spaces for short-term rentals. Like the proposed property rental platform, Airbnb operates on a Consumer-to-Consumer (C2C) business model where property owners directly interact with potential tenants. While Airbnb focuses on short-term stays and vacation rentals, its property discovery and user interaction patterns share similarities with long-term rental platforms, making it a relevant case study for property recommendation approaches.

The platform seems like employing a location-centric approach to property discovery rather than a personalized recommendation system. Upon accessing Airbnb's homepage, users are immediately presented with property listings based on their IP address location. The platform organizes properties into categorical sections such as "Popular destinations," "Trending," and property type categories like "Amazing pools" or "Beachfront." This categorization appears to be static and doesn't adapt to user preferences or browsing patterns. To verify the platform's recommendation approach, multiple tests were conducted using different user accounts and search queries. When accessing the platform with different accounts but the same IP location, the homepage consistently displayed identical property listings, indicating no personalization based on user profiles or browsing history. However, when accessing the platform using different IP locations (through VPN), the homepage displayed entirely different properties based on the new geographic location. This confirms that Airbnb's primary discovery mechanism relies on location data rather than user preferences or behavioral patterns. Figure 2.9 and Figure 2.10 shows the results of Airbnb homepage accessed from different IP locations.

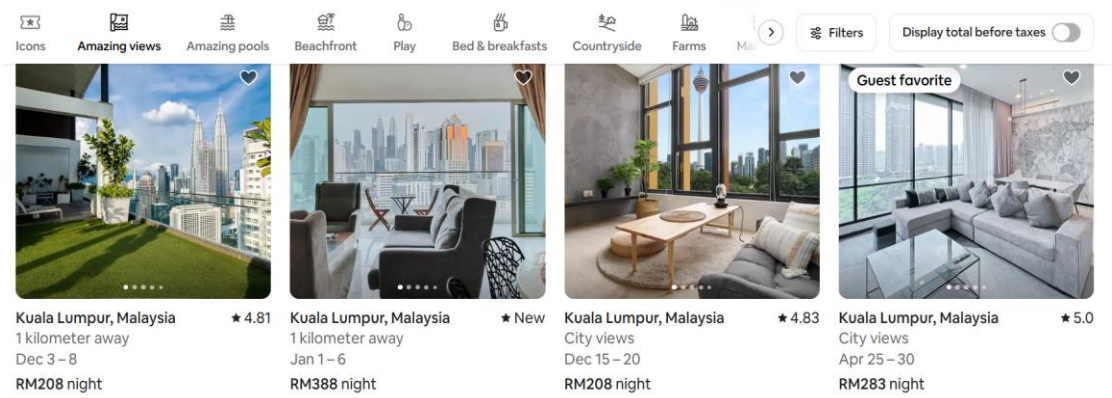


Figure 2.9 Airbnb homepage accessed from Malaysia IP address

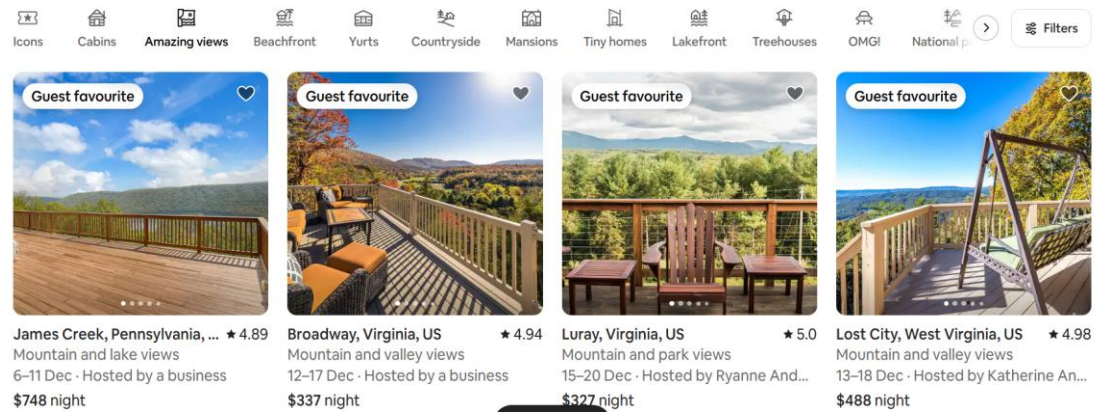


Figure 2.10 Airbnb homepage accessed using United States VPN

The current approach employed by Airbnb exhibits several limitations in its property discovery system. Despite having access to extensive user interaction data, the platform shows no evidence of implementing personalization features, resulting in identical property recommendations for both new and returning users. Furthermore, property detail pages lack a "Similar Properties" section, missing an opportunity to help users discover relevant alternatives. These identified limitations highlight the need for enhancement in the proposed property rental platform. By implementing more sophisticated recommendation algorithms, the platform can offer a more personalized and efficient property discovery experience.

2.3.2 Lazada [13]

Lazada, a prominent e-commerce platform in Southeast Asia, demonstrates a sophisticated recommendation system that actively personalizes the shopping experience. While operating in the e-commerce sector, Lazada's C2C marketplace

model and recommendation implementation offer valuable insights for property rental platforms, particularly in how it handles product discovery and user personalization.

Lazada employs a comprehensive recommendation approach across different sections of its platform. On the homepage, users encounter multiple recommendation modules that adapt based on their interactions. The "Just For You" section prominently displays personalized product recommendations that differ between user accounts, demonstrating effective use of user browsing history and preferences. Figure 2.11 shows the relevant section.

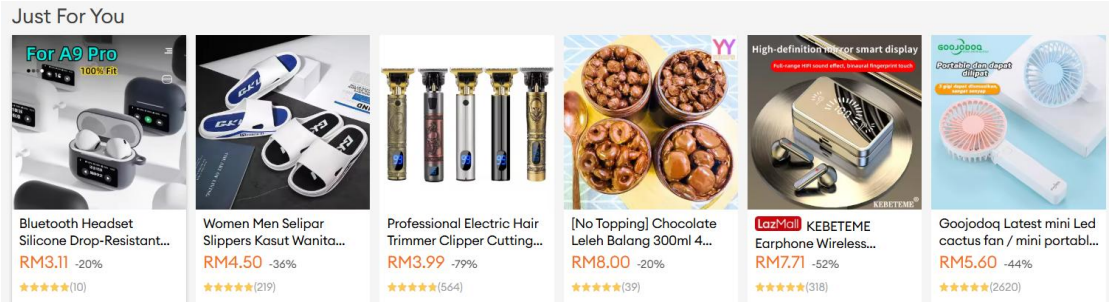


Figure 2.11 Lazada “Just For You” recommendation section

Testing with multiple accounts reveals Lazada's dynamic recommendation system in action. When accessing the platform with different accounts, the "Just For You" section displays distinctly different products based on each account's browsing history. For example, an account that frequently views electronics receives more technology-related recommendations, while another account browsing fashion items sees more clothing suggestions. On product detail pages, the platform shows both "Similar Products" and "You May Also Like" sections to suggest users other products. Figure 2.12 shows the relevant section.

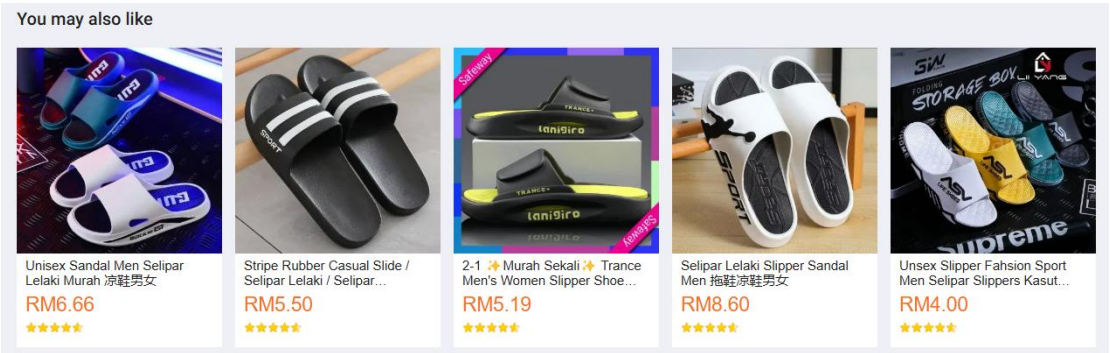


Figure 2.12 Lazada “You may also like” recommendation section

However, Lazada's recommendation system also faces certain challenges in its implementation. The recommendations sometimes appear overly sensitive to recent browsing history, causing rapid shifts in suggested products that might not reflect long-term user preferences. The system's handling of new users relies heavily on displaying popular items, which may not align with the user's actual interests. Additionally, the platform occasionally shows recommendations from vastly different categories, potentially reducing the relevance of suggestions. In the context of property rentals, these limitations underscore the importance of balancing recent user activity with stable preference patterns and maintaining recommendation relevance within the property domain.

2.3.3 Comparison between Recommendation Approaches

Table 2.2 Comparison between Recommendation Approaches

Aspect	Airbnb	Lazada
Approach	Location-based only. Same recommendations for all users with same IP	User behavior based. Personalized "Just For You" recommendations
Detail Page Recommendations	No "Similar Properties" suggestions	Implements "Similar Products" and "You May Also Like" sections
New User Experience	Shows location-based listings only	Shows popular items; gradually builds personalization
Strengths	Clear location-based relevance	Personalized experience.
Key Limitations	Lacks personalization. No adaptation to user preferences	Over-reactive to recent activity. Category relevance issues

Table 2.2, the comparison between Airbnb and Lazada reveals significant insights for implementing recommendation systems in property rental platforms. While both platforms operate on C2C business models, their approaches to recommendations are different. Airbnb's location-centric approach ensures geographical relevance but misses opportunities for personalization, despite having access to user interaction data. In contrast, Lazada demonstrates how effective personalization can enhance user experience through dynamic recommendations based on user behavior and preferences.

These findings suggest several key considerations for the proposed property rental platform. First, there is a clear opportunity to combine the strengths of both approaches – maintaining location relevance while implementing personalized recommendations. Second, the implementation of detail page recommendations, as demonstrated by Lazada, can significantly enhance property discovery.

The analysis highlights the importance of addressing both platforms' limitations. While Airbnb's lack of personalization could frustrate users seeking specific property types, Lazada's over-sensitivity to recent browsing history shows the need for balanced recommendation algorithms. These observations suggest that property rental platforms require an effective recommendation approach that can balance both personalization and relevance.

2.4 Recommender System Algorithms

Recommender systems have become integral components of online platforms, helping users discover relevant items from large datasets. These systems analyze user preferences and behaviors to suggest items that users might find interesting or relevant. This section explores two primary approaches to recommender systems: content-based filtering and collaborative filtering, each offering unique strengths for property recommendations.

2.4.1 Content-based Filtering

Content-based filtering is a prominent recommendation algorithm that focuses on analyzing the attributes of items to suggest relevant content to users based on their preferences. This method operates by constructing a user profile that reflects the user's interests, which is then matched against the characteristics of available items to generate recommendations. The fundamental principle behind content-based filtering is that if a user liked a particular item, they are likely to appreciate other items with similar attributes or features [14].

In content-based filtering systems, the recommendation process begins with the collection of data regarding the items that a user has previously interacted with, such as ratings or preferences. This data is used to create a profile that encapsulates the user's interests. For instance, in the context of textual content, this might involve analyzing the keywords or topics associated with the items the user has liked. The system then

compares these user profiles against the features of other items in the database to identify those that align closely with the user's interests. This approach is particularly effective in domains where item characteristics can be clearly defined, such as in music, books, or movies [15] which is similar to property features like location, size and price range.

One of the key advantages of content-based filtering is its ability to recommend items that are novel or less popular, which collaborative filtering methods might overlook due to their reliance on historical user data [15]. Moreover, content-based systems do not require data from other users, making them suitable for scenarios where user interaction data is sparse or unavailable. However, this method also has limitations, such as the potential for over-specialization, where the system only recommends items similar to those already liked, thereby reducing the diversity of recommendations [16].

2.4.2 Collaborative Filtering

Collaborative filtering is another widely utilized recommendation algorithm that predicts a user's preferences by leveraging the historical behavior and preferences of other users. The fundamental premise of collaborative filtering is based on the assumption that users who have agreed in the past will continue to agree in the future, thereby allowing the system to recommend items that similar users have liked [17]. This approach can be categorized into two main types: user-based collaborative filtering and item-based collaborative filtering.

User-based collaborative filtering identifies users with similar preferences and recommends items that those similar users have rated highly. For instance, if User A and User B have similar tastes in movies, and User A enjoys a particular film that User B has not yet seen, the system will recommend that film to User B [18]. Conversely, item-based collaborative filtering focuses on the relationships between items rather than users. It suggests items that are similar to those the user has liked in the past, based on the ratings given by all users [18]. In this project, the user behaviors may include viewing patterns, search history and wishlisted items. Collaborative filtering systems can be further enhanced through the use of matrix factorization techniques, which decompose the user-item interaction matrix into lower-dimensional representations. This allows for the identification of latent factors that explain observed ratings, improving the accuracy of predictions.

Despite its effectiveness, collaborative filtering faces challenges, particularly in terms of data sparsity and scalability. As the number of users and items increases, the system may struggle to find sufficient overlap in preferences to make accurate recommendations. Techniques such as incorporating social network information and trust metrics have been proposed to mitigate these issues and enhance the robustness of collaborative filtering systems [19].

2.4.3 Comparison of Recommender Techniques

Table 2.3 Comparison of Recommender Techniques

Aspect	Content-based Filtering	Collaborative Filtering
Input Data	-Property features -Property type	-User rental history -User browsing behavior -Wishlisted properties
Strengths	-Works well for new properties -Can explain recommendations based on features -Effective for specific property preferences	-Discovers unexpected recommendations -Can identify trending properties -Better for personalization
Limitations	-Over-specialization -Requires detailed property attributes	-Cold-start problem -Requires substantial user data -Limited for new users

After analyzing both recommendation techniques, this project proposes to implement a hybrid approach that combines content-based and collaborative filtering methods to overcome their individual limitations while leveraging their respective strengths. Content-based filtering excels at matching properties based on specific features that users prefer, such as location, price range, and amenities, which is crucial for initial property recommendations. However, it may lead to over-specialization and miss important social trends in the rental market. Conversely, collaborative filtering can capture community preferences and identify trending properties but struggles with new listings and users who haven't established a browsing history. By implementing a hybrid system, the property rental website can provide more comprehensive and accurate recommendations by considering both the objective property characteristics and user behavior patterns.

2.5 Proposed Solutions

This project proposes a property rental website that implements a hybrid recommender system combining content-based and collaborative filtering techniques to enhance the property discovery process. The platform will be developed using the Next.js framework, which provides efficient server-side rendering capabilities and optimized performance for web applications. The hybrid recommender system will analyze both property attributes and user interaction patterns to generate personalized property suggestions, requiring significant effort in model training and testing to ensure accurate recommendations.

The backend infrastructure will utilize Supabase as the primary database solution, offering several advantages: secure data storage, real-time updates and efficient file storage for property images. This solution provides essential features including user management, property listing storage, and interaction tracking capabilities. The platform will also integrate external services such as Google Maps API for location visualization and OAuth for secure authentication.

CHAPTER 3 SYSTEM METHODOLOGY/APPROACH

3.1 System Development Methodology

This project adopts the Agile methodology rather than traditional Waterfall approach, as its iterative and incremental development process is well-suited for projects involving complex features like recommendation systems. Unlike Waterfall's linear progression where each phase must be completed before moving to the next, Agile methodology allows for continuous refinement and adaptation throughout the development lifecycle. This flexibility is crucial when building a complex web application that needs to integrate various components like user authentication, property management, and recommendation features.

The development process follows the core principles of Agile methodology as illustrated in Figure 3.1, moving through cycles of planning, design, development, testing, review, and deployment. Each iteration begins with planning and design phases where specific features and components are identified for implementation. For instance, early iterations focus on establishing the core web infrastructure using React and Next.js, setting up the database schema, and implementing essential user interfaces. The development phase then focuses on implementing these features, followed by testing to ensure functionality, performance and user experience meet requirements. The review phase allows for evaluation of completed work and identification of necessary adjustments, while the deployment phase integrates new features into the main system.

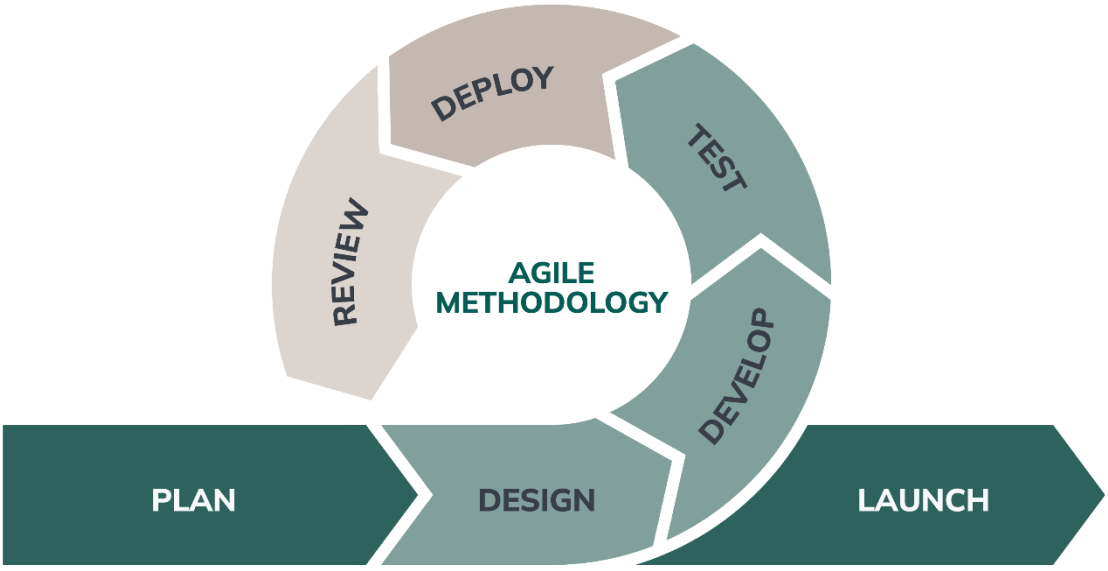


Figure 3.1 Agile methodology [20]

This iterative approach benefits both the frontend and backend development of the project. For the frontend, it allows for continuous refinement of user interfaces based on usability testing. Backend components, including the database structure, API endpoints and recommendation system can be developed and optimized incrementally. This incremental progression ensures that each component like basic CRUD operations and personalized recommendations is thoroughly tested and integrated effectively into the system. The flexibility of Agile methodology also supports effective risk management through early identification and resolution of technical challenges. By breaking down development into smaller, manageable iterations, potential issues in areas such as system architecture, data management and user interface design can be addressed before they impact the overall project timeline.

3.2 System Design Diagram

3.2.1 System Architecture Diagram

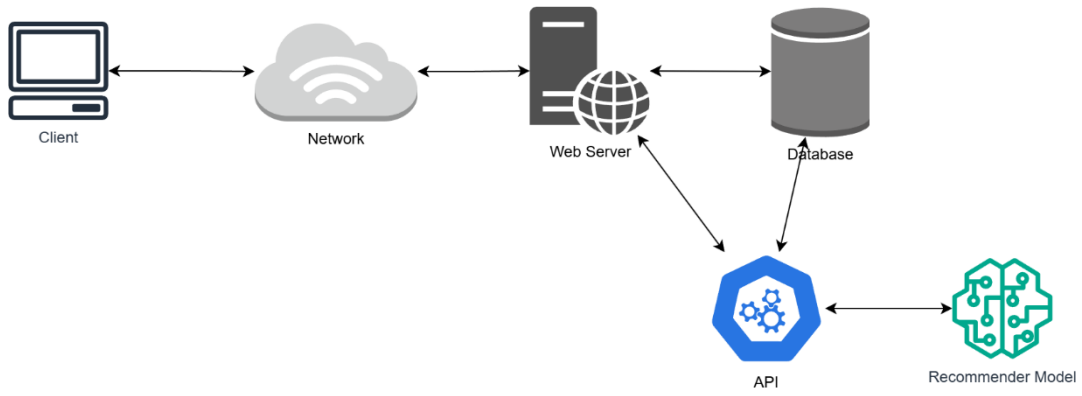


Figure 3.2 Overview system architecture diagram

The proposed property rental website follows the three-tier architecture. As illustrated in Figure 3.2, the system is divided into distinct layers that separate presentation, application logic, and data management. The presentation tier (client) consists of the user interface built with React and Next.js framework, providing users a responsive and interactive interface for browsing properties and managing their accounts.

The second layer is the application layer which contains the web server and API components. This tier handles business logic, user authentication, property management operations and hosts the recommendation service. The API in this layer

connects the front-end with the database and includes the Recommender Model that uses content-based and collaborative filtering to suggest properties to users.

The third layer, data tier consists of a Supabase database system that manages storage of user data, property listings and interaction histories. This three-tier architecture ensures the system more maintainable, scalable and secure. Each tier can be modified or scaled independently without affecting the others, providing flexibility for future enhancements and maintenance.

3.2.2 Entity Relationship Diagram (ERD)

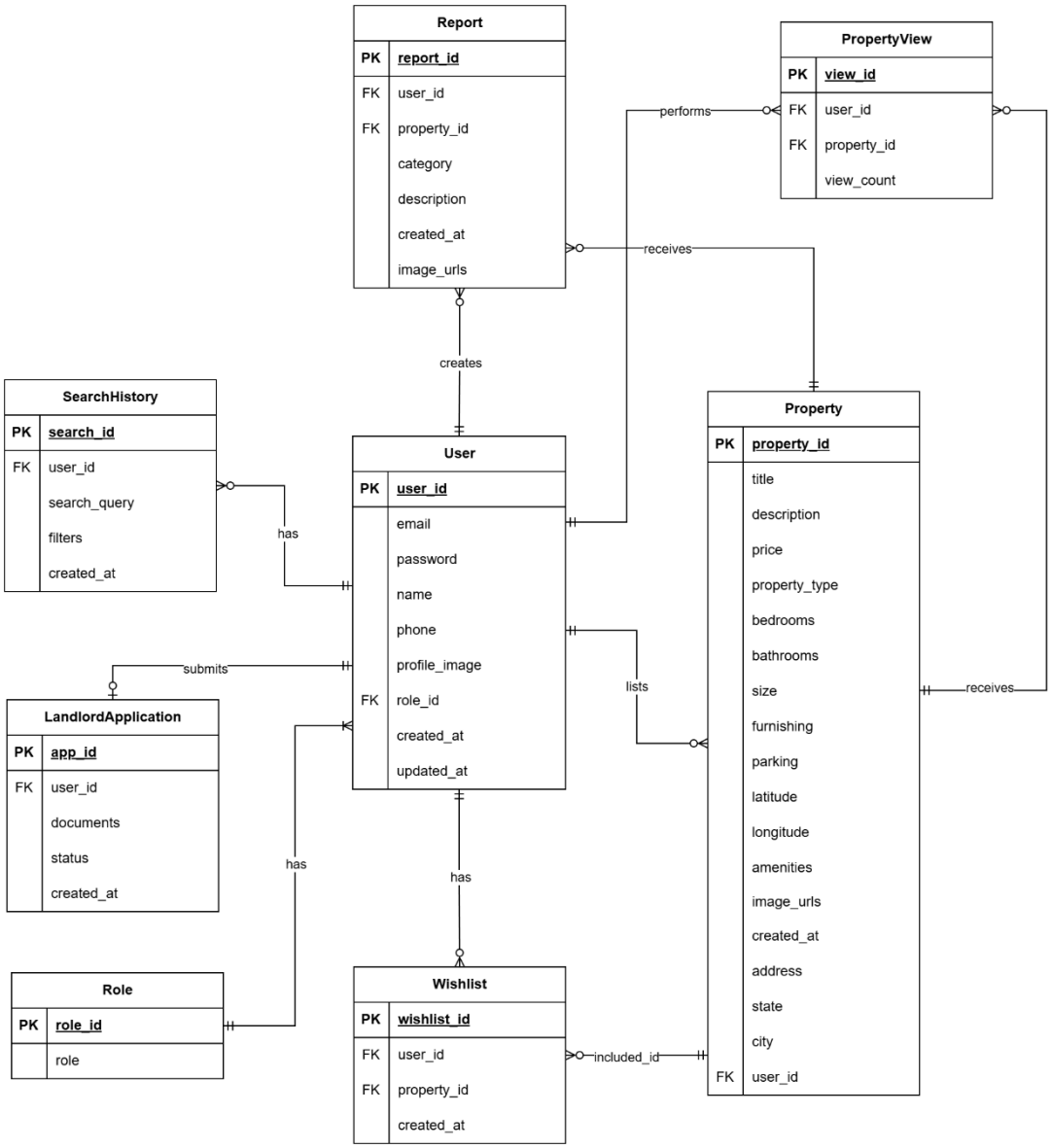


Figure 3.3 ERD of database

3.2.3 Activity Diagrams

3.2.3.1 Activity Diagrams of User and Landlords Associated Use Case

Sign up

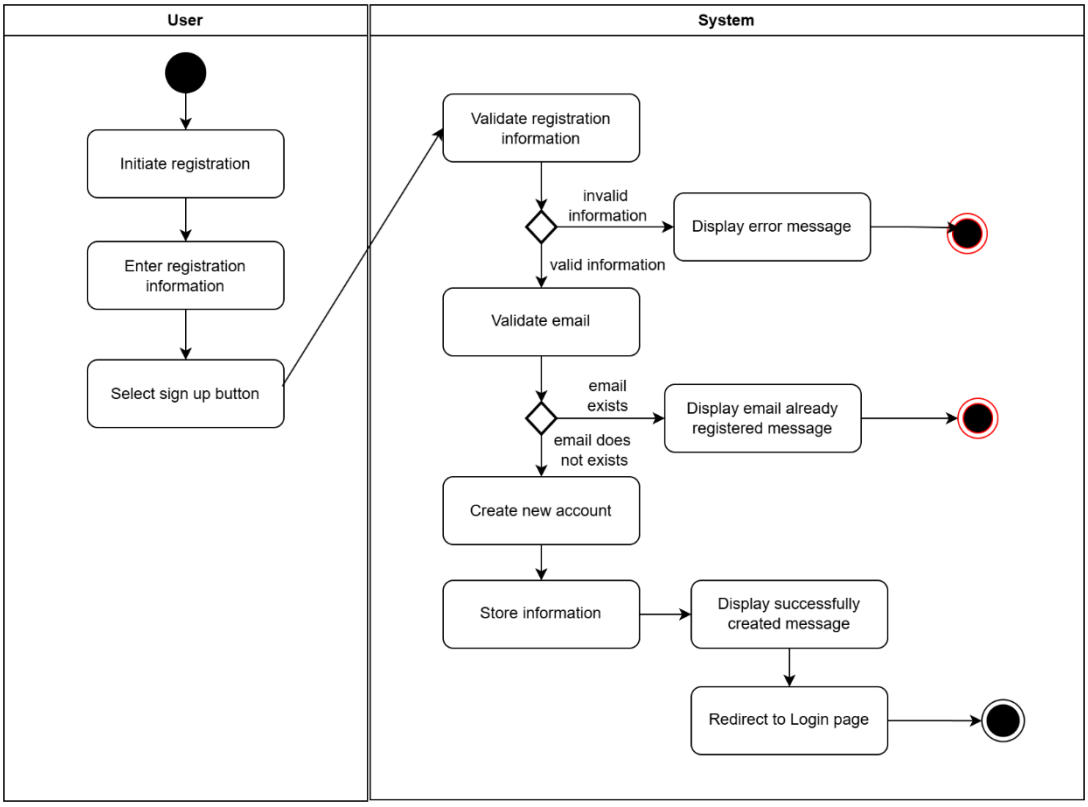


Figure 3.4 Activity diagram of sign up use case

Figure 3.4 illustrates the registration flow for the property rental website. The process begins when a user initiates registration and enters their required before selecting the sign up button. Once submitted, the system performs a series of validation checks. First, it validates the registration information, if any data is invalid, the system displays an error message to the user. If the information passes initial validation, the system then checks if the provided email already exists in the database. In case of a duplicate email, the user receives a message indicating the email is already registered. If the email is unique, the system creates a new account and store the user's information. Upon successful account creation, the system displays a confirmation message and automatically redirects the user to the login page.

Login

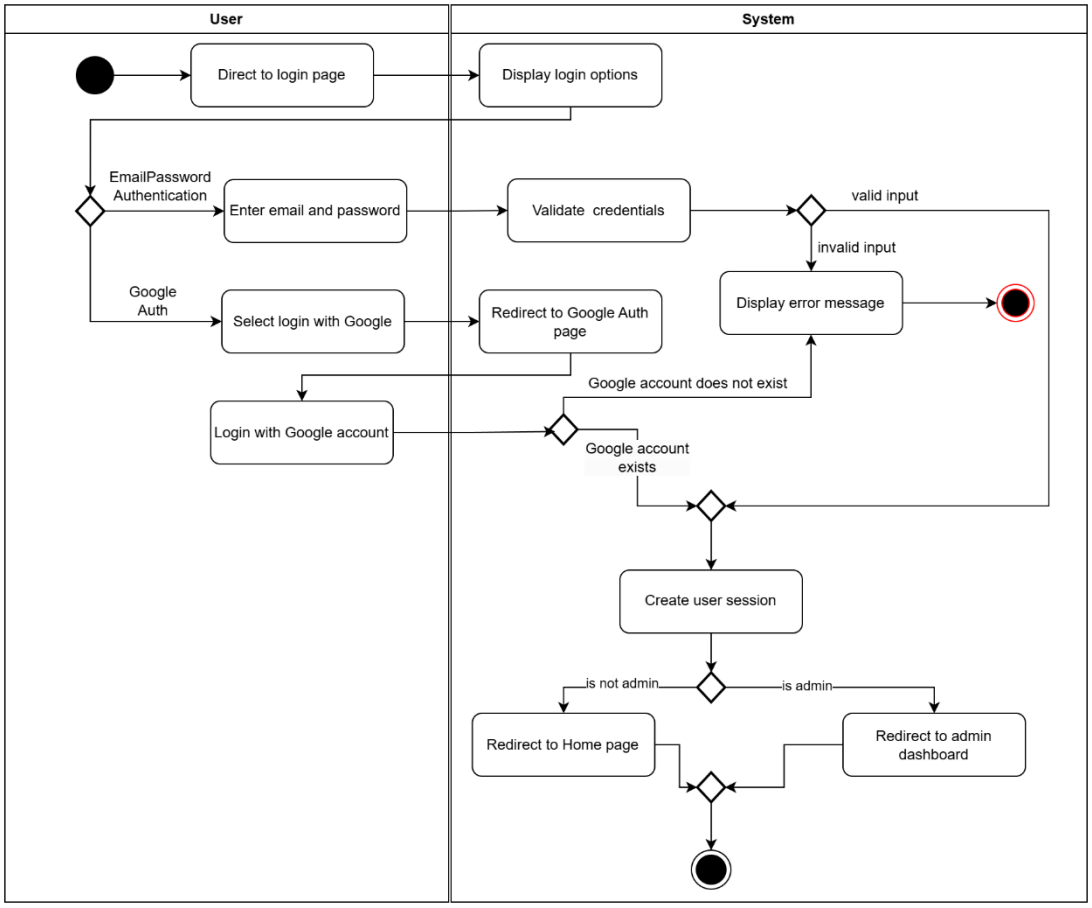


Figure 3.5 Activity diagram of login use case

Figure 3.5 illustrates the login flow for the property rental website. The process begins when a user accesses the login page, where the system presents two authentication options: email/password login or Google authentication. For email/password authentication, users enter their credentials. If the credentials are invalid, an error message is displayed. For Google authentication, users select the "Login with Google" option, which redirects them to Google's authentication page. The system verifies if the Google account exists. Upon successful authentication through either method, the system creates a user session. The system then checks the user's role - if the user is an administrator, they are redirected to the admin dashboard while regular users are directed to the home page.

View profile, edit profile

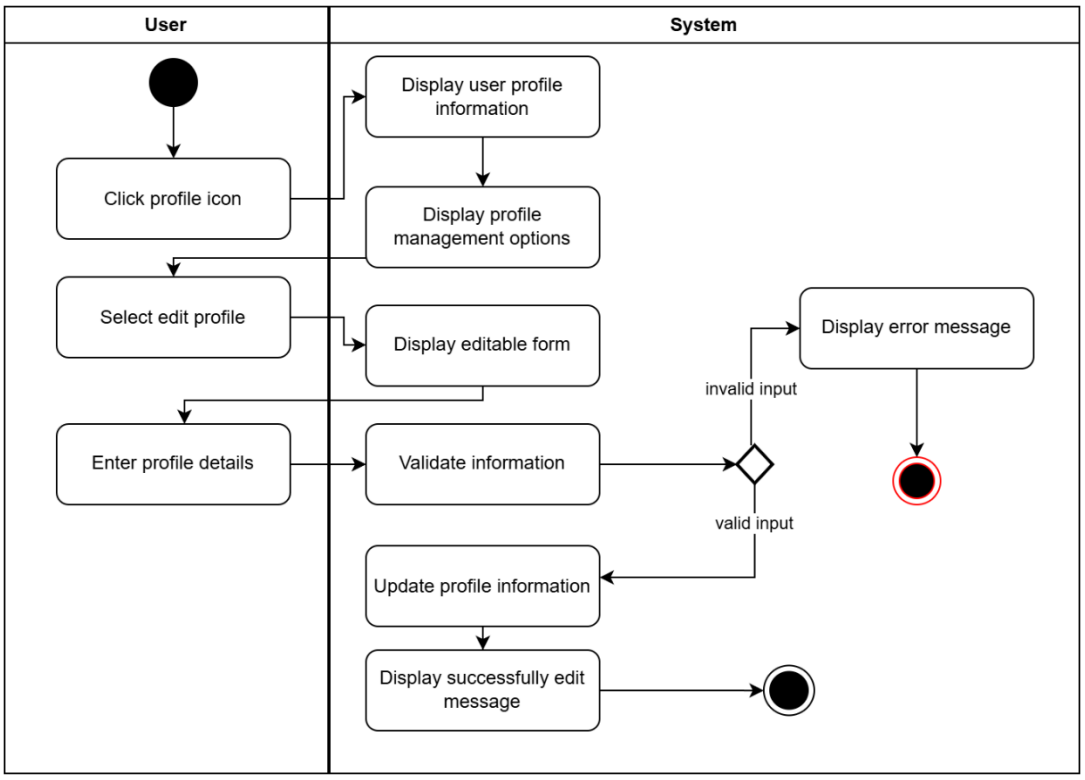


Figure 3.6 Activity diagram of view profile and edit profile use case

Figure 3.6 illustrates the view and edit profile flow for the property rental website. The process begins when a user clicks on their profile icon and redirect to profile page. The system displays their profile information along with available profile management options. When the user selects the edit profile option, the system presents an editable form containing the current profile information. The user can then modify their profile details as needed. Upon submission, the system validates the entered information. If any input is invalid, an error message is displayed to the user. Else, the system updates the profile information in the database and displays a success message to confirm the changes.

Delete account

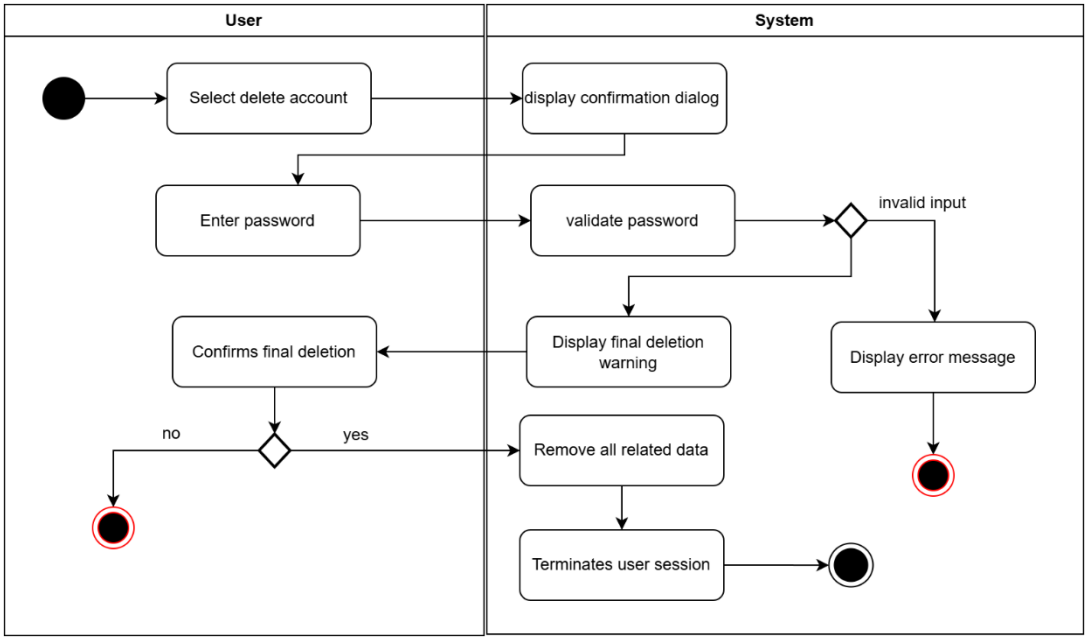


Figure 3.7 Activity diagram of delete account use case

Figure 3.7 illustrates the account deletion flow for the property rental website. The process begins when a user selects the delete account option, the system displays a confirmation dialog. To proceed with deletion, the user must enter their password for security verification. If the password is invalid, the system displays an error message. With valid password entry, the system shows a final deletion warning that requires user confirmation. If the user declines at this stage, the process terminates. However, if the user confirms deletion, the system proceeds to remove all associated user data from the database and terminates the user session.

Search property

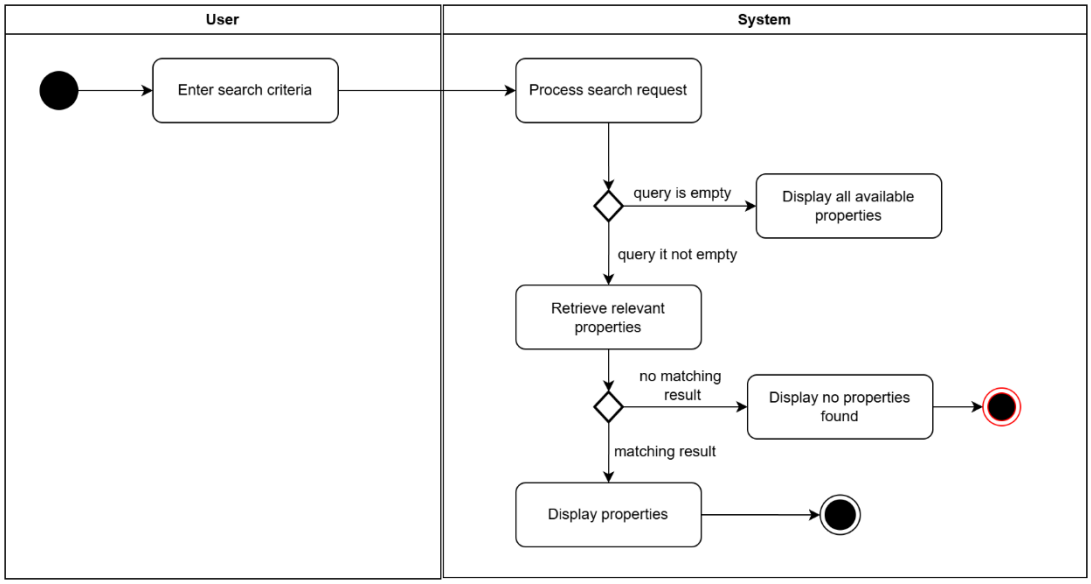


Figure 3.8 Activity diagram of search property use case

Figure 3.8 illustrates the property search flow for the property rental website. The process begins when a user enters their search criteria such as location, property type, price range, or other filters. The system processes the search request and checks if any search parameters were provided. If the search query is empty, the system displays all available properties. However, if search criteria are specified, the system retrieves relevant properties matching those criteria. If no properties match the search parameters, the system displays a "no properties found" message. When matching properties are found, the system displays them to the user.

View property details, report listing, add to wishlist

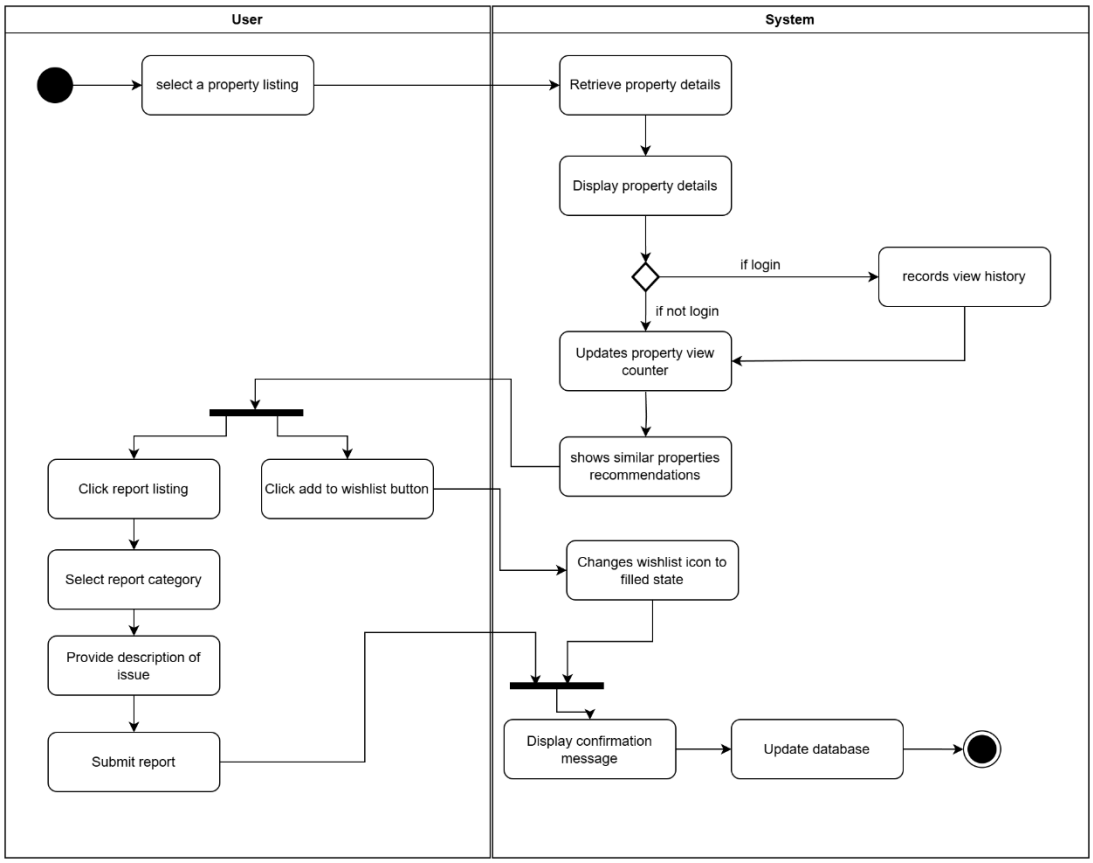


Figure 3.9 Activity diagram of view property details, report listing, add to wishlist use cases

Figure 3.9 illustrates the property details viewing flow along with reporting and wishlist functionality. The process begins when a user selects a specific property listing, the systems retrieve and display property details. If the user is logged in, the system records their view history. Whether logged in or not, the system updates the property's view counter and displays similar property recommendations. From this point, users have two main interaction paths. Report Listing Path: Users can report inappropriate listings by clicking the report button, selecting a report category, providing a description of the issue, and submitting the report. Wishlist Path: Users can add properties to their wishlist by clicking the wishlist button, which changes the wishlist icon to a filled state indicating the property has been saved. Both actions conclude with the system displaying a confirmation message and updating the database accordingly.

View wishlist items, remove wishlist items

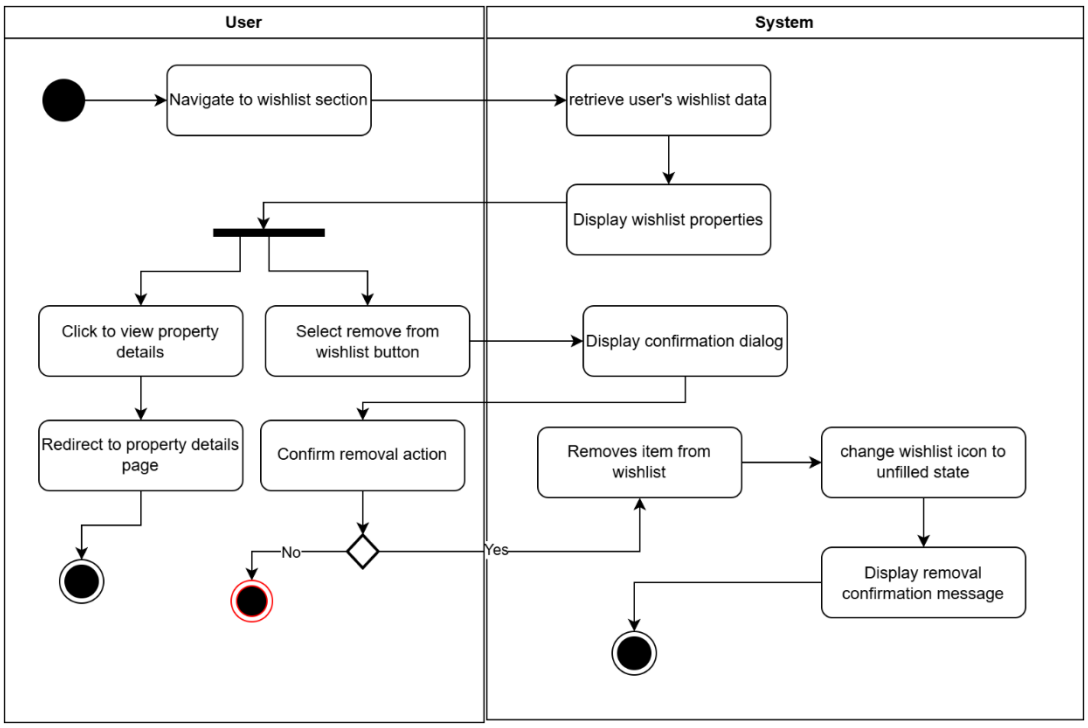


Figure 3.10 Activity diagram of view wishlist items, remove wishlist items use cases

Figure 3.10 illustrates the wishlist management flow for the property rental website. When a user navigates to the wishlist section, the system retrieves and displays all properties saved in their wishlist. Users have two main interaction paths. View Property Details: Users can click on any wishlist property to view its details, which redirects them to the property details page. Remove from Wishlist: Users can remove properties by selecting the remove button, which triggers a confirmation dialog. If the user confirms removal, the system removes the item from the wishlist, changes the wishlist icon to an unfilled state, and displays a removal confirmation message. If the user declines removal, the process terminates with no changes made.

Apply for landlord status, track application status

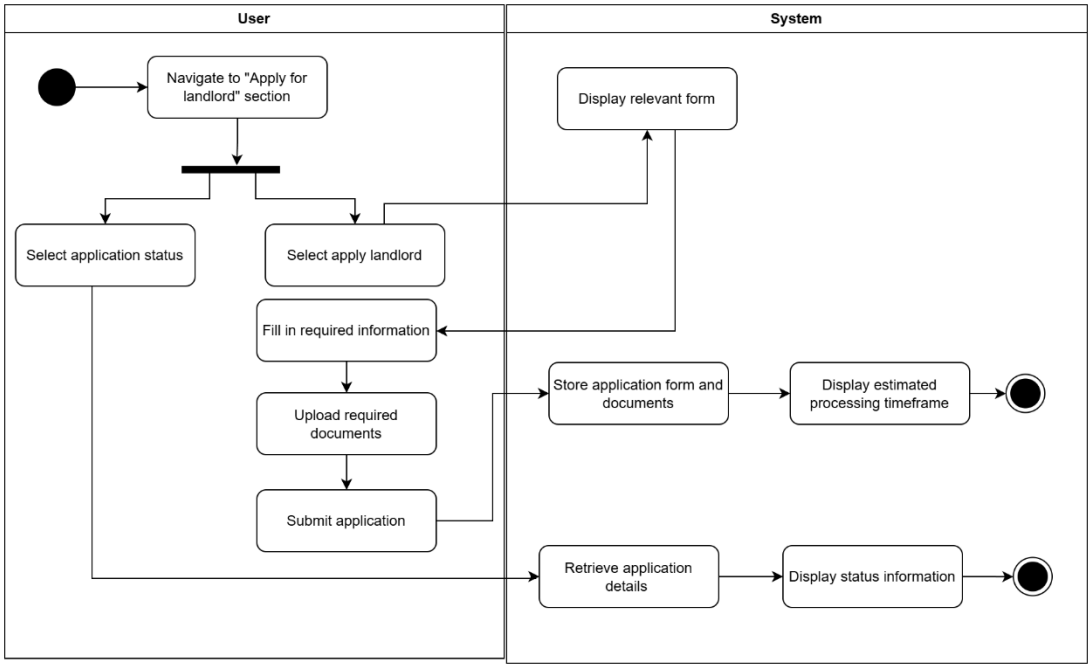


Figure 3.11 Activity diagram of apply for landlord status, track application status use cases

Figure 3.11 illustrates the landlord application and status tracking flow for the property rental website. When users navigate to the "Apply for Landlord" section, they have two main paths. Apply for Landlord Status: Users can select the apply option, which prompts the system to display the application form. Users must fill in required information and upload necessary documents before submitting their application. Once submitted, the system stores the application data and documents, then displays an estimated processing timeframe. Track Application Status: Users can select to view their application status, prompting the system to retrieve and display their current application details and status information.

Create listing

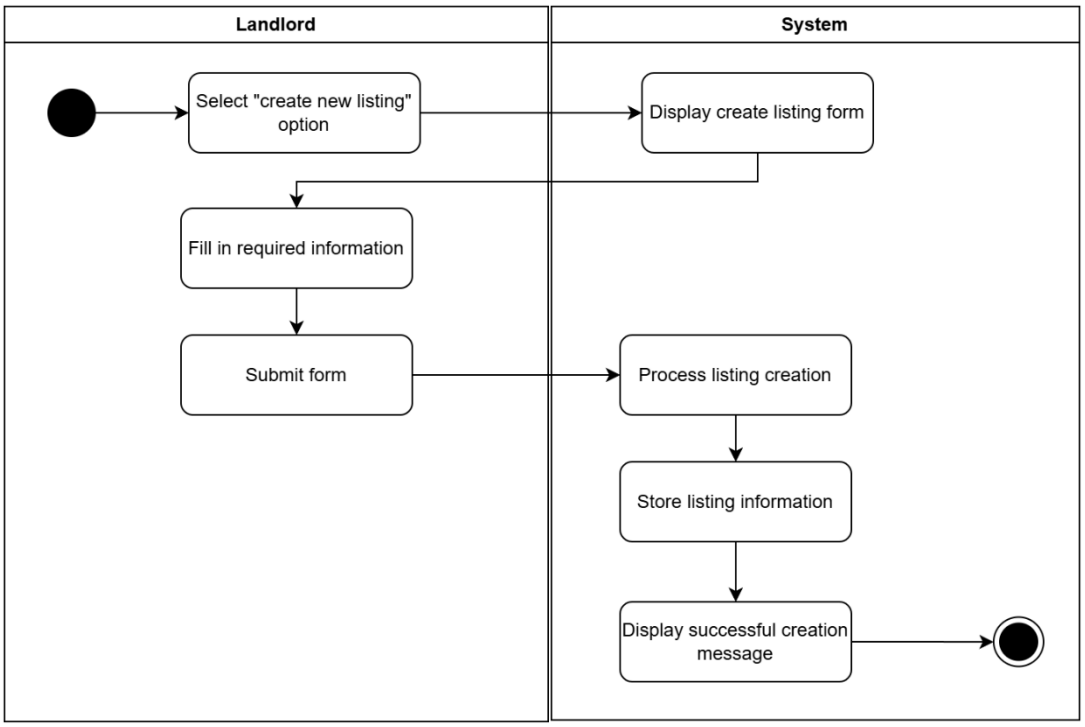


Figure 3.12 Activity diagram of create listing use case

Figure 3.12 illustrates the flow of create property listing. When landlord select the create new listing option, system displays a form. The landlords then fill in the required information, including upload property images. Upon submit the form, the system stores the listing information in database and display successful creation message to the users.

Edit listing details

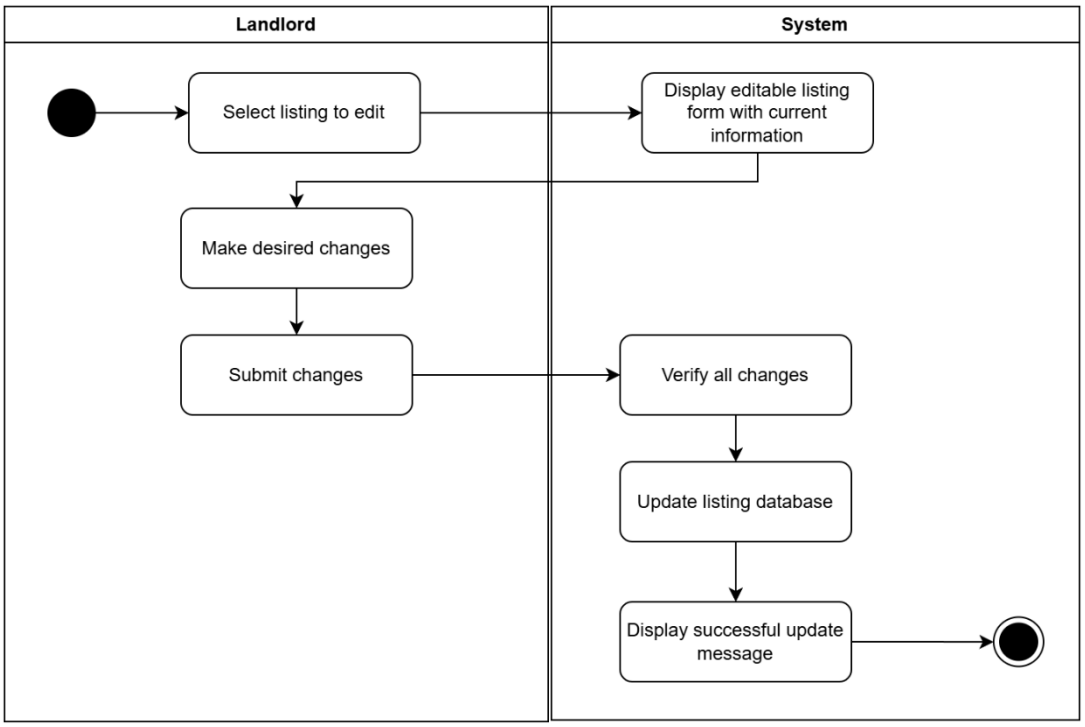


Figure 3.13 Activity diagram of edit listing details use case

Figure 3.13 illustrates the process for landlords to edit their property listings. The flow begins when a landlord selects a specific listing to edit, the system displays an editable form with the current property information. The landlord can then make desired changes to any aspect of the listing including property details, pricing, images, or amenities. After making modifications, the landlord submits these changes. The system verifies all modifications, updates the listing information in the database, and displays a success message confirming the changes have been saved.

Delete listing

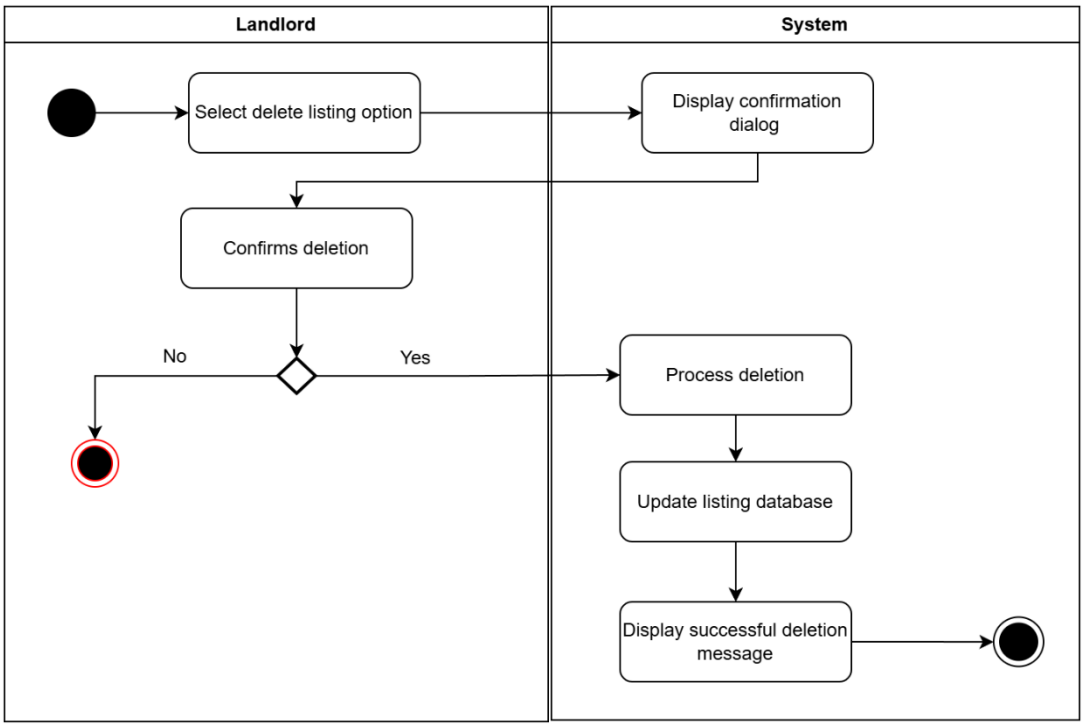


Figure 3.14 Activity diagram of create listing use case

Figure 3.14 illustrates the property listing deletion flow for landlords. The process begins when a landlord selects the delete option for a specific listing, the system displays a confirmation dialog. If the landlord cancels at this stage, the process terminates with no changes made. However, if the landlord confirms the deletion, the system processes the request by removing the listing data, updates the database to reflect the deletion and displays a success message confirming the listing has been removed.

3.2.3.2 Activity Diagrams of Admin associated use case

Verify landlord application

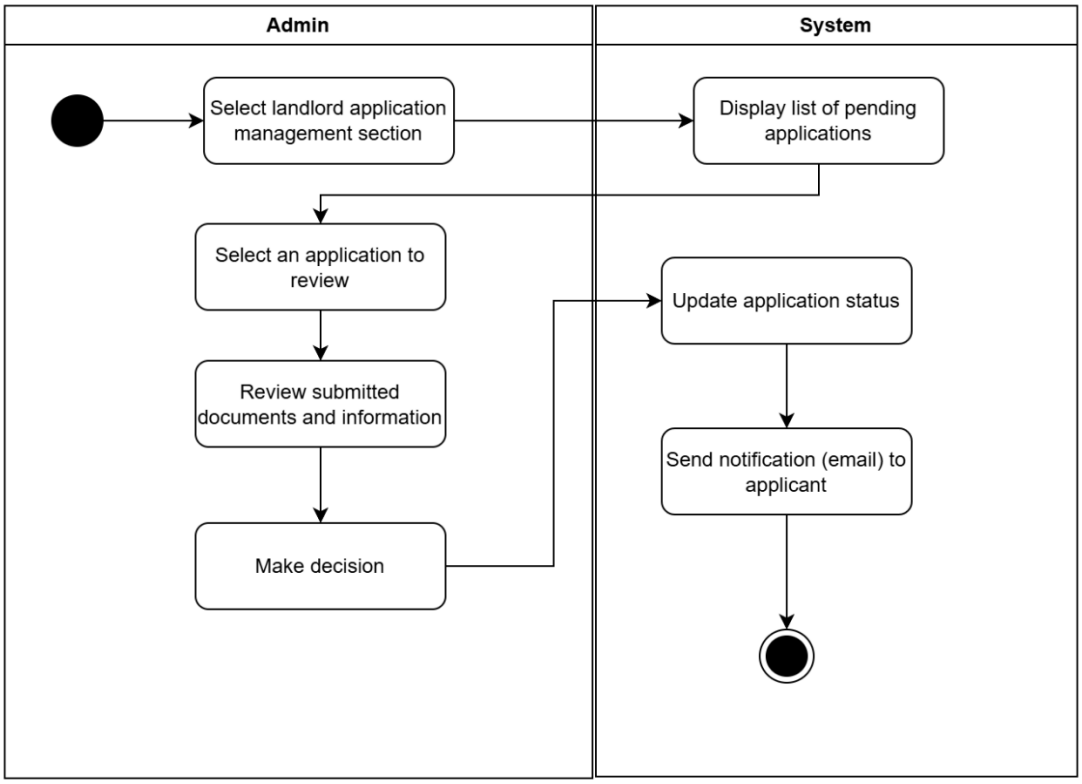


Figure 3.15 Activity diagram of verify landlord application use case

Figure 3.15 illustrates the landlord application verification flow for administrators. The process begins when an admin accesses the landlord application management section, where the system displays a list of pending applications. The admin selects a specific application to review and examines the submitted documents and information for verification. After reviewing the materials, the admin makes a decision on whether to approve or reject the application. Upon decision, the system updates the application status in the database and sends an automated email notification to the applicant informing them of the outcome.

Manage user roles

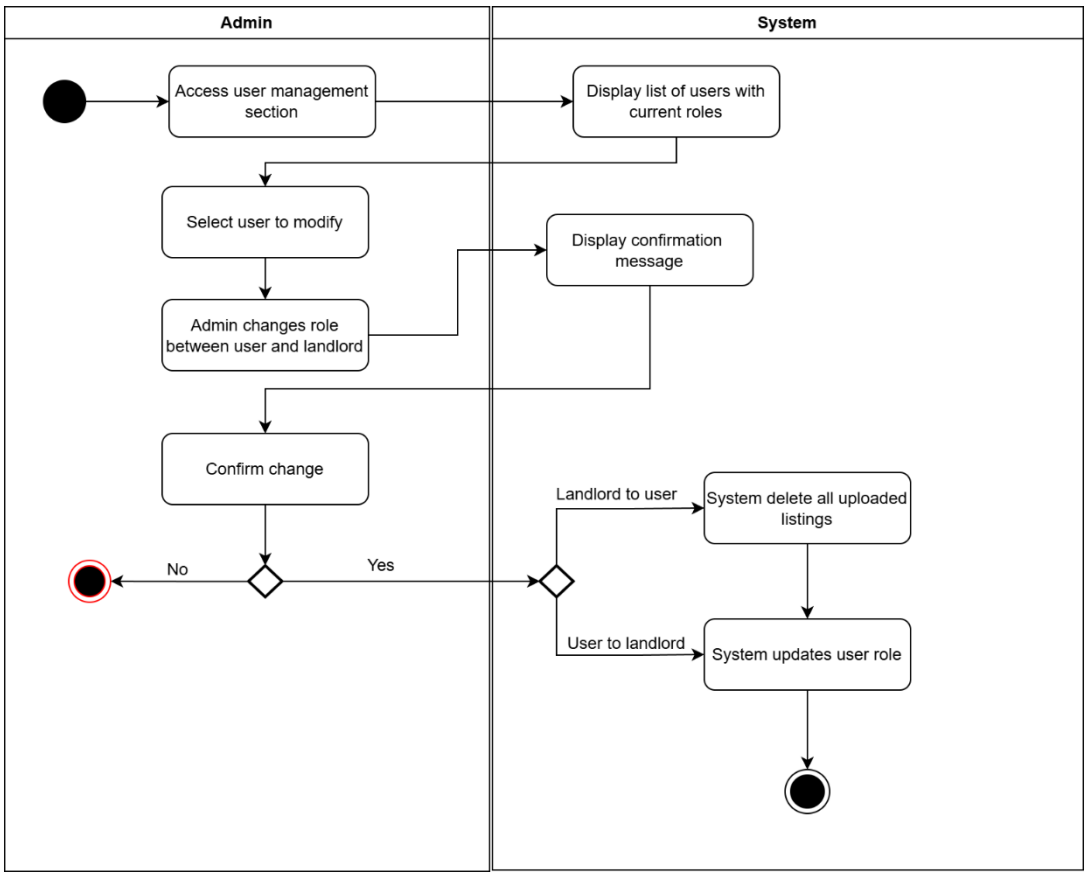


Figure 3.16 Activity diagram of manage user roles use case

Figure 3.16 illustrates the user role management flow for administrators. The process begins when an admin accesses the user management section, where the system displays a list of all users with their current roles. The admin can select a specific user to modify and change their role between regular user and landlord status. The system displays a confirmation message for the role change, requiring admin confirmation before proceeding. If the admin declines, the process terminates with no changes. Upon confirmation, the system follows one of two paths. Landlord to User: If changing from landlord to regular user, the system first deletes all property listings associated with that account before updating the user role. User to Landlord: If upgrading to landlord status, the system directly updates the user role.

View user reports, remove inappropriate listing

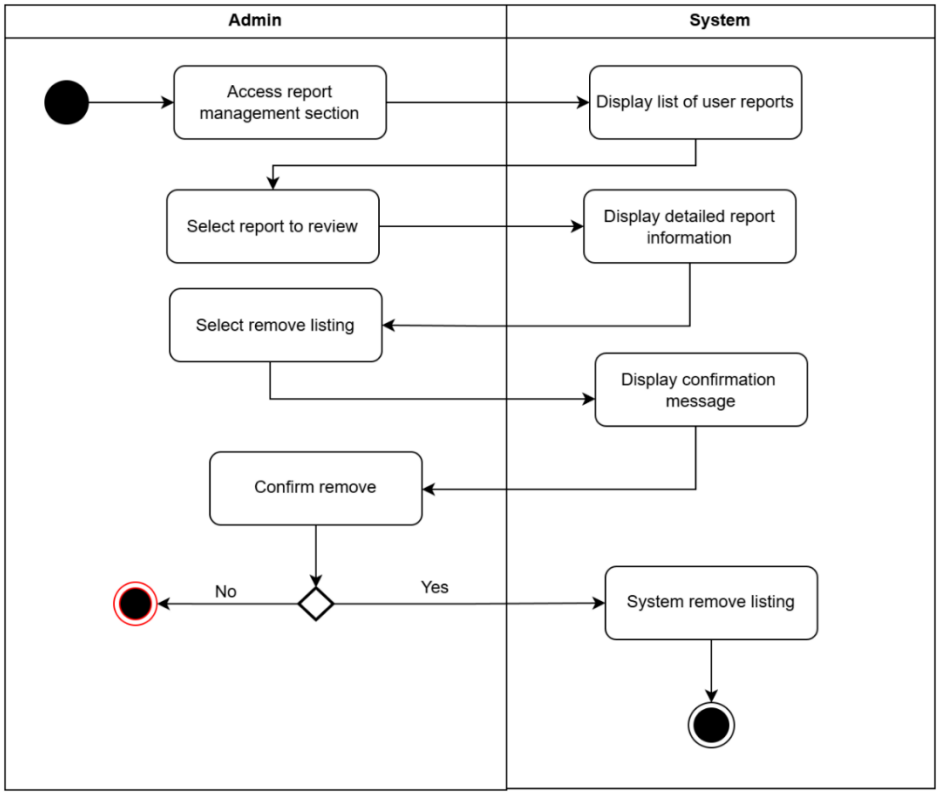


Figure 3.17 Activity diagram of view user reports, remove inappropriate listings use cases

Figure 3.17 illustrates the report handling and listing removal flow for administrators. The process begins when an admin accesses the report management section, where the system displays a list of user-submitted reports. The admin can select a specific report to review, prompting the system to display detailed information about the reported issue. After reviewing the report, if the admin determines the listing violates platform guidelines, they can select to remove the listing. The system displays a confirmation message to prevent accidental deletions. If the admin declines, the process terminates with no changes. However, if the admin confirms removal, the system proceeds to remove the listing from the platform.

3.2.4 Use Case Diagram

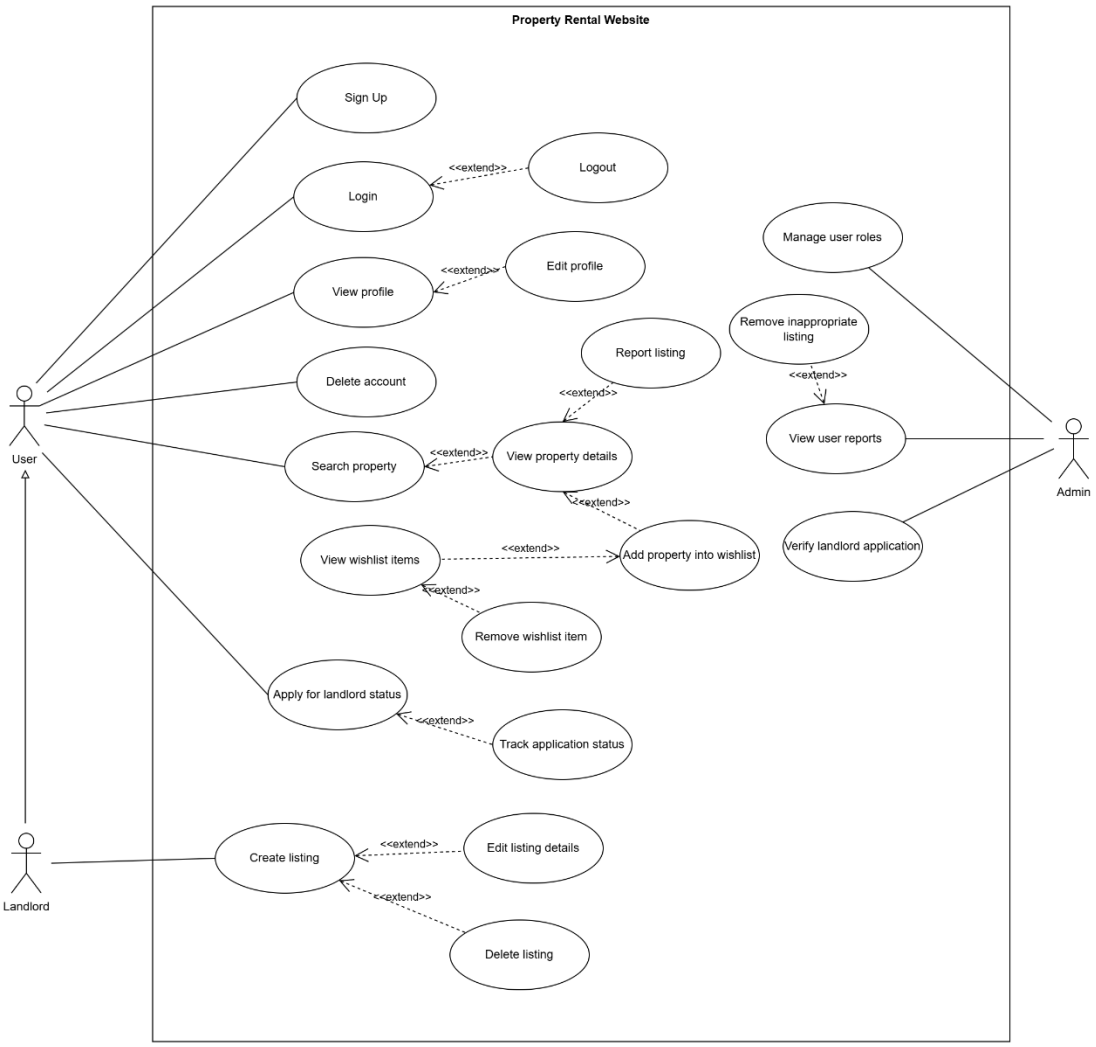


Figure 3.18 Use case diagram

Figure 3.18 shows the use-case diagram of the property rental website with three main actors: Users, Landlords, and Administrators. Regular users can perform account-related actions (sign up, login, view/edit profile, delete account), search and view properties, manage their wishlist by adding or removing properties and apply for landlord status. Landlords, who are specialized users with additional privileges, can create new property listings, edit listing details, and delete their listings. Administrators have oversight capabilities including managing user roles, verifying landlord applications, viewing user reports, and removing inappropriate listings to maintain platform quality. The use case description for each use case is included in APPENDIX - Appendix A: Use Case Descriptions.

3.3 System Requirements

3.3.1 Functional Requirements

User Perspective:

- Users shall be able to register and login using email/password or Google OAuth.
- Users shall be able to search for rental properties with filters.
- Users shall be able to view detailed property information including images, descriptions and location on map.
- Users shall be able to save properties to favorites and view their search history.
- Users shall be able to compare multiple properties side by side.
- Users shall be able to report inappropriate listings or suspicious activities.
- Users shall be able to view their profile and edit personal information.
- Users shall be able to apply for landlord/agent status.
- Users shall be able to track their landlord/agent application status.
- Users shall be able to receive personalized property recommendations.
- Users shall be able to delete their account.

Landlord Perspective:

- Landlords shall be able to create and manage property listings once verified.
- Landlords shall be able to upload and manage property images.
- Landlords shall be able to input and edit property details including price, location and amenities.
- Landlords shall be able to track views on their listings.
- Landlords shall be able to remove their listings.

Admin Perspective:

- Administrators shall be able to review and verify landlord/agent applications.
- Administrators shall have a dedicated dashboard.
- Administrators shall be able to manage user accounts and roles.
- Administrators shall be able to suspend or ban problematic users.
- Administrators shall be able to remove inappropriate content.
- Administrators shall be able to handle user reports and complaints.

System Perspective:

- The system shall provide secure user authentication and authorization.
- The system shall maintain role-based access control for different user types.
- The system shall process and validate property data before storage.
- The system shall optimize and store property images.
- The system shall maintain search terms for efficient property discovery.
- The system shall handle concurrent user sessions.
- The system shall validate user inputs and provide appropriate error messages.
- The system shall integrate with external services (maps, authentication) securely.
- The system shall ensure data consistency across all operations.
- The system shall provide accurate search results based on user queries.
- The system shall analyze user behavior patterns including view history, wishlist items and interaction data to generate collaborative filtering recommendations

3.3.2 Non-functional Requirements

Performance:

- Website pages should load within 3 seconds on standard internet connections
- Search results should display within 2 seconds
- Property images should be automatically compressed for faster loading

Security:

- User passwords must be securely hashed
- User authentication is required for accessing personal features
- Users can only edit or delete their own listings

Usability:

- Website should be mobile-responsive (works on desktop and mobile devices)
- Interface should be easy to navigate with clear labels and buttons
- Form validation with clear error messages
- Compatible with common browsers (Chrome, Firefox, Safari)

Recommender System:

- Precision rate of at least 70% for property recommendations

- Recall rate of at least 65% for relevant properties

3.4 Timeline

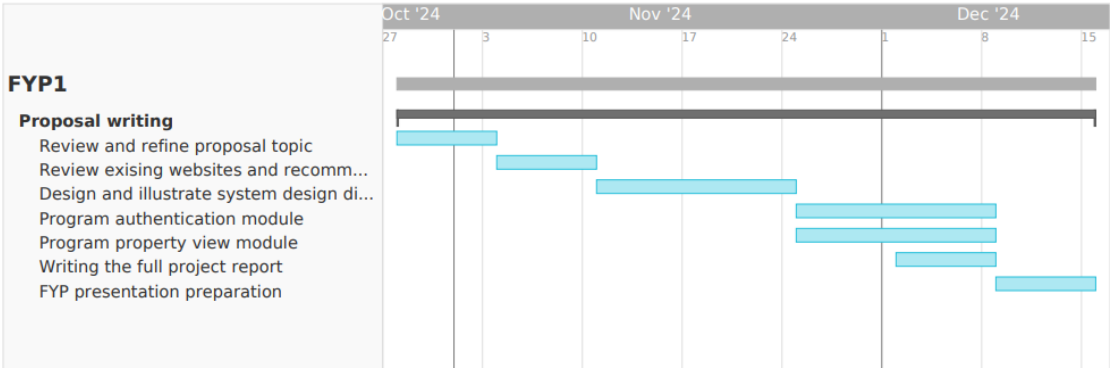


Figure 3.19 Gantt Chart of Final Year Project 1

FYP begins in Week 1 with a review and refinement of the proposal. Week 2 is dedicated to conducting research on existing property rental websites and analyzing various recommendation techniques. Weeks 3 and 4 focus on system design and documentation, including the creation of ERD, use-case diagrams, activity diagrams, and low-fidelity UI diagrams. During Weeks 5 and 6, the preliminary development focuses on the implementation of authentication module and property view module, which includes database setup and integration of external APIs. Week 6 also involves compiling and writing the complete project report. Week 7 is allocated for preparing presentation materials, including the project poster and presentation slides for the FYP1.

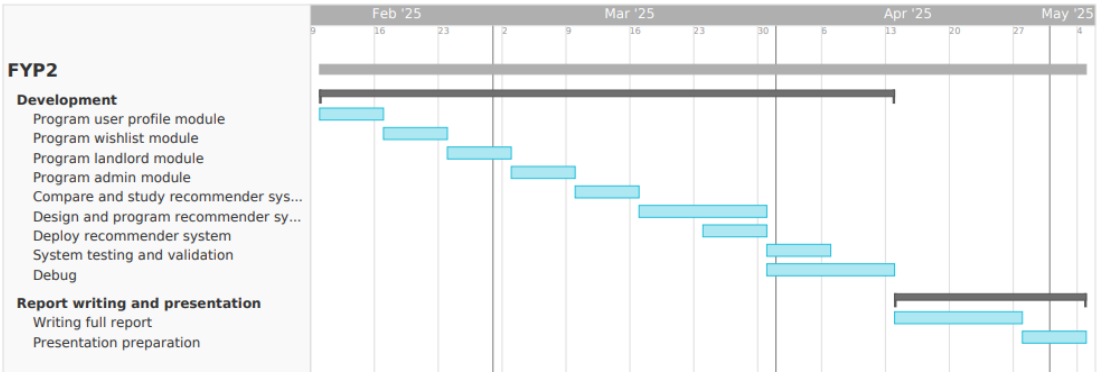


Figure 3.20 Gantt Chart of Final Year Project 2

The development phase begins in Week 1 with the implementation of the user profile module, establishing core user management functionality. Week 2 is to develop the wishlist module to enable property saving and tracking features. In Week 3, the landlord module is implemented, incorporating property listing and management capabilities. Week 4 focuses on the admin module development including a dedicated dashboard for admin. Weeks 5 through 8 are dedicated to the recommender system, with Week 5 allocated for comparative study of recommendation techniques, followed by Weeks 6 and 7 for designing and programming the hybrid recommender system that combines content-based and collaborative filtering approaches. Week 8 focuses on deploying the recommender system into the platform. Week 9 is dedicated to comprehensive system testing and validation, while Weeks 10 and 11 are reserved for debugging and system optimization to ensure robust performance.

The final phase, Weeks 12 through 14, is documentation and presentation preparation. Weeks 12 and 13 focus on writing the complete project report, while Week 14 is dedicated to preparing materials for the final project presentation.

CHAPTER 4 SYSTEM DESIGN

4.1 System Block Diagram

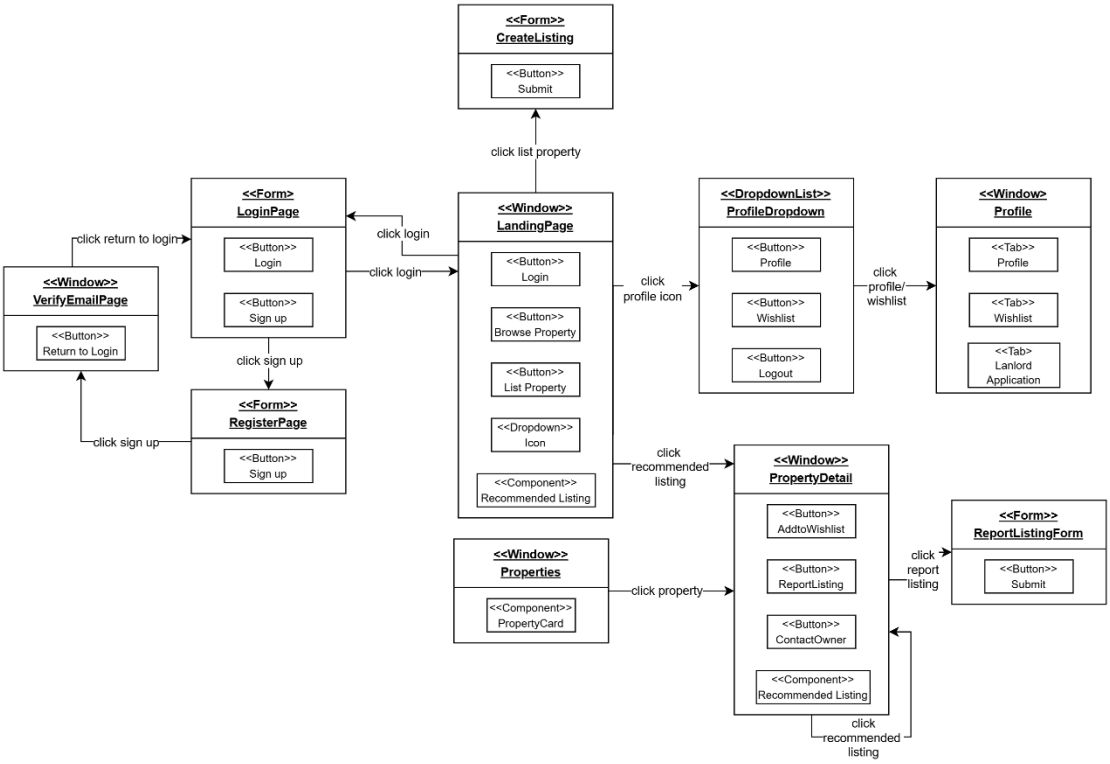


Figure 4.1 Window Navigation Diagram (User/Landlord)

Figure 4.1 illustrates the user interface structure and interaction flow of the rental property website. Starting from the LandingPage as the central access point, users can navigate to various sections through dedicated buttons and controls. The authentication flow is represented on the left side, showing the sequence from LoginPage to RegisterPage and VerifyEmailPage. The center path demonstrates property browsing functionality, where users can view property listings and access detailed information about specific properties. On the right shows the profile management, including tabs for wishlist and landlord application features.

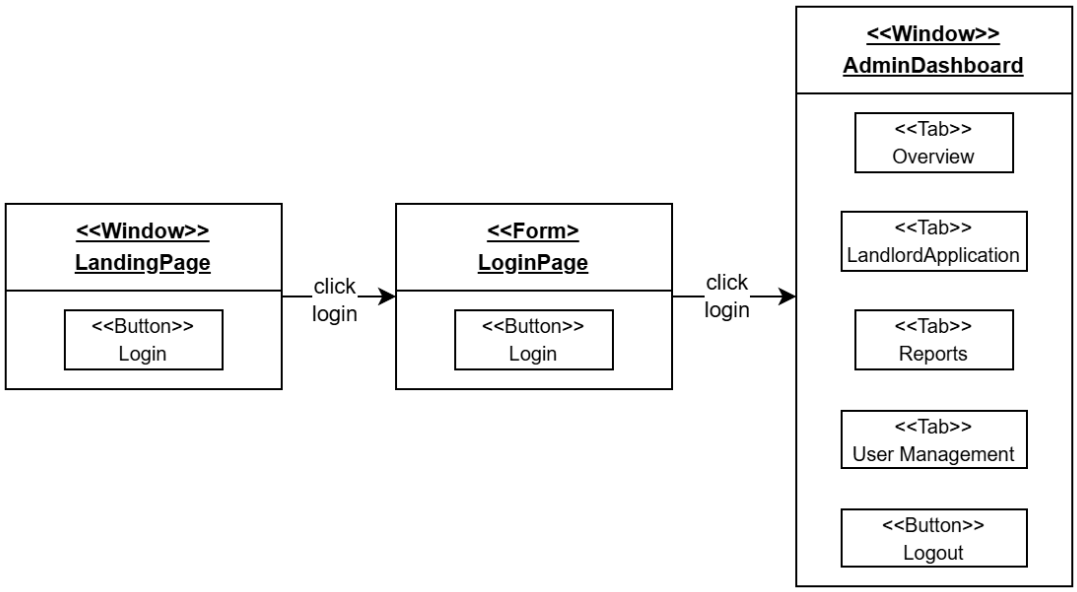


Figure 4.2 Window Navigation Diagram (Admin)

Figure 4.2 illustrates the structure and navigation of the administrator role of the rental property website. It shows the pathway from the initial LandingPage through the LoginPage to access the AdminDashboard. Once authenticated, administrators gain access to a comprehensive dashboard featuring four specialized tabs: Overview for monitoring key metrics, LandlordApplication for reviewing and approving landlord verification requests, Reports for managing property listing reports and user complaints, and User Management for administering user accounts and permissions.

4.2 System Components Specifications

Landing Page

The Landing Page serves as the entry point to the rental property website. It features a search section with a background image of property listings. Users can perform initial property searches by entering location, property type, price range, and number of bedrooms. The page includes a navigation bar for site-wide access and showcases a section of recommended property listings to engage users.

Login Page

The Login Page provides user authentication functionality through a form interface. It contains email and password input fields with credential validation and a submit button. The page implements JWT token-based authentication and stores user session

information securely. Error handling displays appropriate messages for invalid credentials.

Register Page

The Register Page allows new users to create accounts using a form that collects essential user information. It includes fields for name, email and password. The registration process incorporates email verification through a token-based system to ensure authentic user accounts. Client-side validation checks for proper email formatting, password complexity, and field completion before submission.

Email Verification Page

The Email Verification Page confirms user email addresses through a token validation system. When users click verification links sent to their email, this page processes the token, updates the user's verified status in the database, and provides visual feedback on the verification result.

Properties View Page

The Properties View page displays available rental properties with filtering and sorting capabilities. It offers a grid view showing property cards in a responsive layout. The filtering system allows users to refine properties by type, price range, location, amenities and other attributes. This component implements pagination to manage large result sets.

Property Details Page

The Property Details Page presents comprehensive information about individual rental properties. It features an image gallery with property photos, detailed descriptions, key specifications, amenities list, location map and pricing information. The page includes contact options for reaching the landlord and buttons for saving to wishlist or reporting inappropriate listings. A recommendation section displays similar properties based on the current property's attributes and the user's browsing history.

Create Listing Page

The create listing page enables landlords to list new rental properties. It contains multiple sections for inputting property details, including title, description, property type, address with Google Maps integration for geocoding, pricing, specifications, amenities and photo uploads. The form implements step-by-step validation to ensure

all required information is provided before submission. Image uploads are handled through Supabase cloud storage service with preview functionality.

Profile Page

The Profile Page provides users with account management capabilities through a tabbed interface. The main profile tab allows users to edit personal information and update their profile image. The Wishlist tab displays saved properties for easy access. The Landlord Application tab enables users to apply for landlord status by uploading verification documents. For existing landlords, this page also provides access to manage their property listings. The interface adapts based on user role permissions and verification status.

Admin Dashboard

The Admin Dashboard offers administrative controls for platform management, accessible only to users with admin privileges. It features four main tabs: Overview tab displaying key metrics and platform statistics, Landlord Applications for reviewing and approving landlord verification requests, Reports for handling property listing reports and user complaints and User Management for managing user accounts.

4.3 Recommender Model Architecture

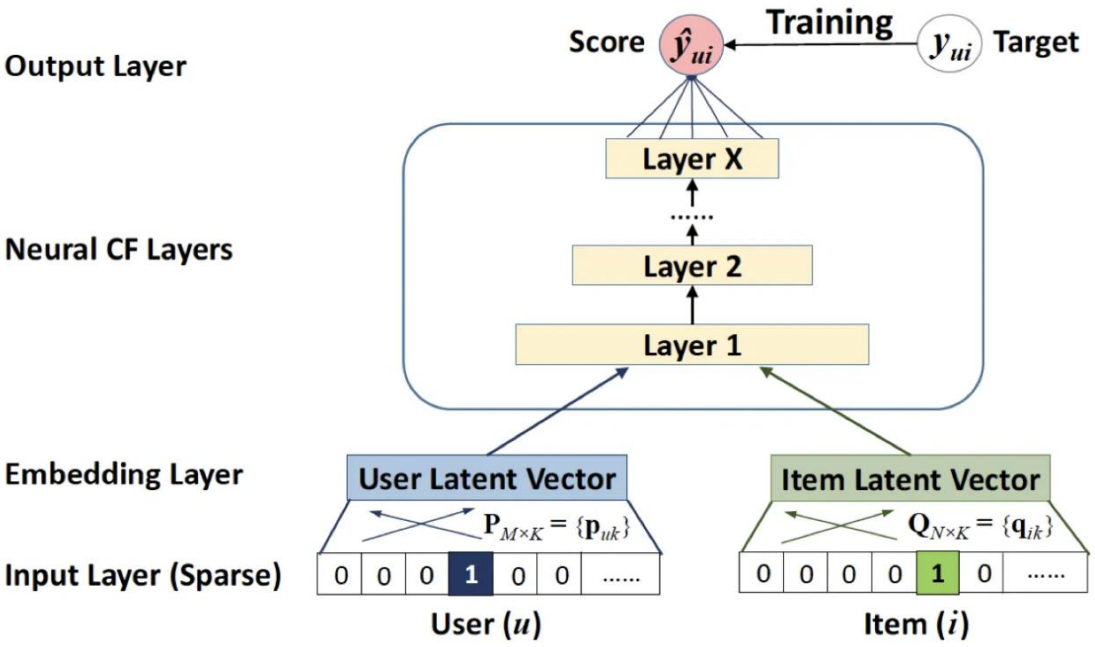


Figure 4.3 NCF Architecture Diagram [21]

The recommendation system employs Neural Collaborative Filtering (NCF) architecture as shown in Figure 4.3.

Input Layer receives sparse representations where users and properties are identified by one-hot encoded vectors. These sparse vectors are transformed in the Embedding Layer into dense User Latent Vectors (P) and Item Latent Vectors (Q), capturing the essential characteristics of both entities in a lower-dimensional space. For properties, these embeddings encode attributes such as location, price range, and property type, while user embeddings represent behavior patterns and preferences.

Neural CF Layers consists of multiple fully connected layers that learn complex non-linear relationships between users and properties. These hidden layers effectively address the cold-start problem through its dual pathway learning approach. For new users with no interaction history, the model can still make recommendations based on the property embeddings alone, essentially defaulting to content-based filtering by identifying properties similar to generally popular ones or those matching common preference patterns. Similarly, for new property listings with no viewing history, the model leverages the property's content features encoded in its latent vector to position it correctly in the recommendation space relative to other properties with established interaction patterns. This is possible because the Neural CF layers learn to recognize patterns between property characteristics and user preferences even without direct interaction data for specific user-property pairs.

The Output Layer produces the final recommendation score through a sigmoid activation function, yielding a probability that represents the likelihood of a user's interest in a specific property.

4.4 Recommender System Pipeline

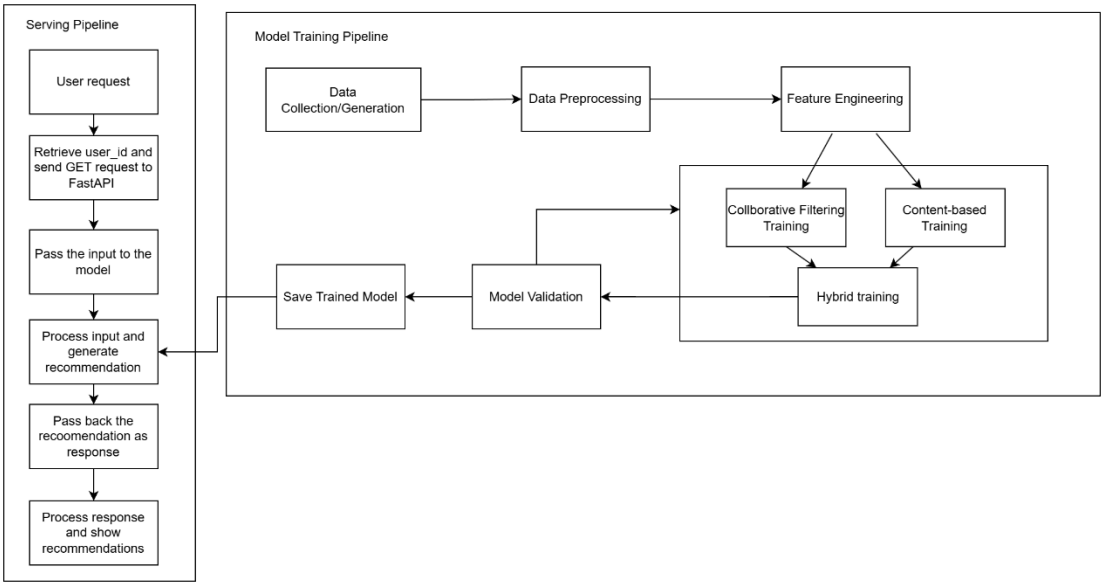


Figure 4.4 Recommender System Block Diagram

The recommender system consists of two main pipelines, the training pipeline during model training and the serving pipeline when integrate with the website. Figure 4.4 illustrates how these components interact to provide property recommendations to users.

4.4.1 Data Collection/Generation

Local property rental data was scraped from iProperty.my to build the foundation of the dataset. To train a recommendation model, both property listings and user interaction data were needed. A real property rental dataset from Kaggle was located that included both property information and user interaction data. However, this reference dataset posed a challenge as it contained Japanese property listings with significantly different price ranges and location characteristics compared to the local Malaysia data.

Several statistical transformation methods were applied to address the distributional mismatch between local and foreign property prices. The uniform quantile transformation was particularly effective. This statistical method transformed the skewed local price distribution into a uniform distribution that closely matched the foreign data pattern. Min-max scaling was then applied after transformation to ensure price ranges aligned while preserving the adjusted distribution.

After the property data was transformed, synthetic user interaction data was generated to simulate realistic user behavior on the platform. The generation process involved

creating three primary types of user interactions: views, wishlists, and inquiries. The distribution of these interaction types was modeled after patterns observed in the reference dataset, with approximately 34% views, 34% wishlists, and 32% inquiries. To ensure realistic behavioral patterns, logical constraints were implemented in the data generation algorithm. For instance, users would typically view a property before adding it to their wishlist, and properties in a user's wishlist had a higher probability (70%) of receiving an inquiry. The algorithm also incorporated geographical preferences, where users were more likely (85%) to interact with properties located in their own state. This approach resulted in a synthetic dataset that closely resembled natural user behavior patterns.

Below are the data dictionaries for the training data. There are a total of 884 records of property data and 1000 records of user interaction data.

Table 4.1 Data dictionary for property data

Field	**Data Type**	**Description**
property_id	String	Unique identifier for each property
title	String	Title of the property listing
state	String	State where property is located
city	String	City where property is located
address	String	Detailed address of the property
size	Integer	Size of the property in square units
price	Float	Rental price
beds	Integer	Number of bedrooms
bathrooms	Integer	Number of bathrooms
carparks	Integer	Number of parking spaces
furnishing	String	Furnishing status (e.g., Fully Furnished)

Table 4. 2 Data dictionary for user interaction data

Field	Data Type	Description
user_id	String	Unique identifier for each user
property_id	String	Unique identifier for each property
event	String	Type of interaction (view, wishlist, inquiry)

4.4.2 Data Preprocessing

The datasets were examined for missing values, inconsistencies and outliers to ensure high data quality. For property data, missing values in categorical features like 'furnishing' and 'property_type' were filled with the most frequent values, while numerical features such as 'price' and 'size' were processed with median imputation to avoid the influence of extreme values. Text descriptions of properties underwent cleaning through the removal of special characters, conversion to lowercase and elimination of stopwords to improve the quality of text-based features.

For the user interaction data, transformation involved converting the different interaction types (view, wishlist, inquiry) into numerical weights to reflect their relative importance. Views were assigned a base weight of 1.0, wishlists received a weight of 2.0 to indicate a stronger level of interest, and inquiries were given the highest weight of 3.0 to represent the strongest signal of user interest. This weighting scheme allowed the recommendation models to differentiate between casual browsing and serious interest. The preprocessed data was then split into training and testing sets using an 80:20 ratio. Additionally, features were normalized where appropriate to ensure that no single feature dominated the modeling process due to its scale.

4.4.3 Feature Engineering

The feature engineering phase focused on transforming preprocessed data into meaningful representations for the recommendation models. For the content-based filtering approach, property features were extracted and converted into numerical vectors. Textual information, such as property descriptions, titles, and amenities, was processed using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This technique weighted terms based on their importance within each

property listing and their rarity across the entire dataset. Categorical features like property type, furnishing status, and location were one-hot encoded to create binary feature vectors. Numerical attributes including price, size, number of bedrooms, and bathrooms were normalized and incorporated into the feature set. For geographical data, state and city information was encoded and weighted to allow the model to recognize location preferences.

For the collaborative filtering approach, user-item interaction matrices that capture the frequency and type of interactions were constructed based on the processed user behavior data. The interaction weights provided the values for this matrix, with missing entries indicating no interaction.

4.4.4 Model Training

The model training phase involved developing two complementary recommendation approaches: content-based filtering and neural collaborative filtering. For the content-based filtering component, the TF-IDF vectors generated during feature engineering were used to calculate similarity scores between properties. The cosine similarity metric was employed to measure the angle between property feature vectors, with values closer to 1 indicating higher similarity.

For collaborative-based filtering, Neural Collaborative Filtering (NCF) model was implemented. The architecture consisted of several key components: embedding layers to map user and item IDs to dense vectors, element-wise product layers to capture linear interactions and deep neural network layers to model complex non-linear relationships. User and property embeddings were initialized with 128 dimensions to capture sufficient latent factors. The model employed a multi-layer perceptron (MLP) with three hidden layers of decreasing sizes (128, 64, and 32 neurons), each followed by Dropout (0.6) to prevent overfitting. These layers were activated using the ReLU function.

After training both models separately, a hybrid approach was developed that combined their predictions. The final recommendation score was calculated as a weighted average of the content-based similarity score and the NCF prediction.

4.4.5 Model Validation

The recommendation models were evaluated using several metrics to assess performance accuracy. It includes NDCG, hit rate, precision and recall. Additionally, hyperparameter tuning was conducted using grid search to optimize model architecture and training parameters, including learning rates, batch sizes, dropout rates and embedding dimensions. After the best model is trained it is saved and made ready for deployment in the serving pipeline.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1 Hardware Setup

Tabel 5. 1 Hardware Specifications

Operating System	Windows 11 Home
Processor	AMD Ryzen 5 5600H @3.3GHz
RAM	16GB
Graphic	NVIDIA RTX3050 Laptop

5.2 Software Setup

Tabel 5. 2 Software Specifications

Software	Description
Next.js	Next.js is used as the primary framework for building the full-stack web application, providing server-side rendering, API routes, and optimized performance features. Version 15.0.0
Supabase	Supabase serves as the backend platform, providing cloud-based PostgreSQL database, storage. Version 2.45.6
Tailwind CSS	Tailwind CSS is utilized for styling the application using utility-first CSS framework, enabling rapid UI development. Version 3.4.15
FastAPI	FastAPI is used to build the recommendation system API, handling property recommendations and machine learning model serving. Version 0.109.0
VS Code	Visual Studio Code serves as the primary integrated development environment for coding, debugging and version control integration. Latest stable version
Git	Git is used for version control and collaborative development of the project. Version 2.43.0
Draw.io	Draw.io is used to illustrate several diagrams such as ERD, use case diagram, activity diagrams and wireframes
Vercel	Vercel serves as the deployment and hosting platform for the Next.js application, providing continuous deployment and severless function. Latest stable version

Render	Render is used to deploy and host the FastAPI recommendation service. Latest stable version

5.3 Setting and Configuration

The development environment setup began with installing Node.js and npm, followed by initializing a Next.js application with TypeScript support for data type safety during development. Additionally, Git was initialized and used for version control, with the repository hosted on GitHub to facilitate collaboration and code management. A PostgreSQL database hosted on Supabase serves as the primary data storage solution, with the database schema following the ERD outlined in section 3.2.2.

For the frontend and web implementation, the project leverages Tailwind CSS for styling, supplemented by the Shadcn UI components library to ensure consistency and accessibility across the application. Authentication is implemented using NextAuth.js with custom email token verification via SMTP using Gmail service. For geographical features, Google Maps API was integrated to provide property location visualization. Vercel was chosen as the hosting platform for this Next.js project due to its strong Next.js support and free tier offerings.

The hybrid recommender system implementation required training a recommender model and setting up a separate FastAPI service deployed on Render. This service provides both content-based and collaborative filtering recommendations via dedicated API endpoints, which are ready to be called by the Next.js application to generate personalized property recommendations for users.

5.4 Key Code Implementation

This section focuses on discussing the core implementation of the key components of the rental property website. The complete source code of this project is available in the GitHub repository [22]

5.4.1 Hybrid Recommender System Implementation

```python
@app.get("/hybrid-recommendations/")
async def get_hybrid_recommendations(user_id: str, property_id: str, top_n: int = 10, alpha: float = 0.6, beta: float = 0.4):
    try:
        # Check if user exists in the model
        if user_id not in user_encoder.classes_:
            # If user not in model, fall back to content-based only
            print(f"User {user_id} not found in trained model, using content-based only")
            similar_properties = await get_similar_properties(property_id=property_id, top_n=top_n)
            return similar_properties

        # Find the reference property
        reference_property = properties_df[properties_df['property_id'] == property_id]
```

Figure 5. 1 Code snippets for recommendation system endpoint

The implementation leverages FastAPI to serve recommendations through dedicated endpoints. The hybrid-recommendations endpoint accepts parameters including user ID, property ID and weighting factors for the hybrid approach. When a request is received, the system first attempts to generate collaborative recommendations using the Neural Collaborative Filtering model. For new users that not present in the training data, the system falls back to content-based recommendations by analyzing property attributes.

```python
# Get collaborative filtering recommendations for user
user_recommendations = await get_recommendations(user_id=user_id, top_n=top_n*2, alpha=0.7)
user_rec_ids = [item['property_id'] for item in user_recommendations]

# Get content-based similar properties
similar_properties = await get_similar_properties(property_id=property_id, top_n=top_n*2)
similar_prop_ids = [item['property_id'] for item in similar_properties]

# Combine both recommendation sets and score
all_properties = properties_df.copy()

# Add collaboration score
all_properties['cf_score'] = 0.0
for i, prop_id in enumerate(user_rec_ids):
    idx = all_properties[all_properties['property_id'] == prop_id].index
    if len(idx) > 0:
        # Reverse the position to create a score (higher = better)
        all_properties.loc[idx, 'cf_score'] = 1.0 - (i / len(user_rec_ids))

# Add content similarity score
all_properties['cb_score'] = 0.0
for i, prop_id in enumerate(similar_prop_ids):
    idx = all_properties[all_properties['property_id'] == prop_id].index
    if len(idx) > 0:
        # Reverse the position to create a score (higher = better)
        all_properties.loc[idx, 'cb_score'] = 1.0 - (i / len(similar_prop_ids))

# Calculate hybrid score
all_properties['hybrid_score'] = (alpha * all_properties['cf_score']) + (beta * all_properties['cb_score'])

# Exclude the reference property
all_properties = all_properties[all_properties['property_id'] != property_id]

# Sort by hybrid score and get top N
hybrid_recommendations = all_properties.sort_values(by='hybrid_score', ascending=False).head(top_n)
```

Figure 5. 2 Code snippets for hybrid recommendation algorithms

The hybrid scoring is calculated by combining NCF predictions with content similarity scores using configurable alpha and beta weights, allowing fine-tuning of the recommendation balance.

5.4.2 Authentication System

```
export const authService = {
  async signUpWithEmail(userData: {
    email: string;
    password: string;
    name: string;
  }): Promise<UserProfile> {
    // Check if user already exists
    const { data: existingUser } = await supabase
      .from('user')
      .select('email')
      .eq('email', userData.email)
      .single();

    if (existingUser) {
      throw new Error('Email already registered');
    }

    // Generate verification token
    const verificationToken = crypto.randomBytes(32).toString('hex');
    const tokenExpires = new Date();
    tokenExpires.setHours(tokenExpires.getHours() + 24); // Token expires in 24 hours

    // Hash password
    const hashedPassword = await bcrypt.hash(userData.password, 10);
```

Figure 5. 3 Code snippets for email registration

The authentication system was custom-built to handle both traditional email-password authentication and Google OAuth integration. The email verification workflow generates secure tokens with 24-hour expiration periods. The system employs bcrypt for password hashing, ensuring security for stored credentials.

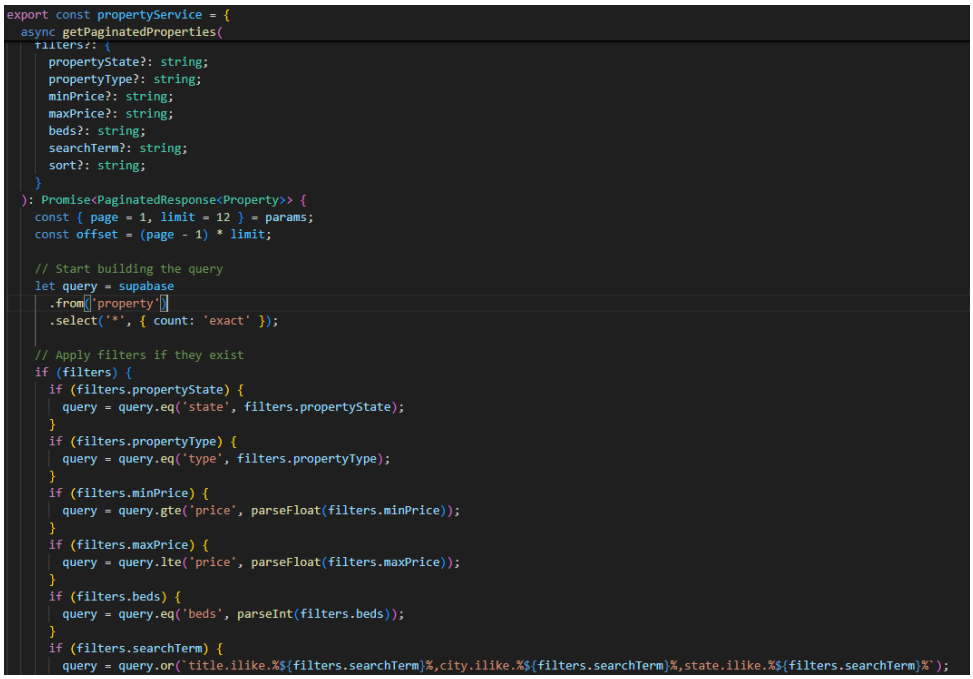
```
  async session({ session, token }: { session: Session; token: any }) {
    if (session?.user) {
      // Get user from database
      const { data: user } = await supabase
        .from('user')
        .select('*')
        .or(`email.eq.${session.user.email},google_id.eq.${token.sub}`)
        .single();

      if (user) {
        session.user.id = user.user_id;
        session.user.role = user.role;
      }
    }
    return session;
  },
  async jwt({ token, user, account }) {
    if (account && user) {
      token.sub = user.id;
    }
    return token;
  },
```

Figure 5. 4 Code snippets for NextAuth

NextAuth.js manages session handling with JWT tokens, and custom middleware protects routes based on user roles.

5.4.3 Property Search and Filtering



```
export const propertyService = {
  async getPaginatedProperties(
    filters?: {
      propertyState?: string;
      propertyType?: string;
      minPrice?: string;
      maxPrice?: string;
      beds?: string;
      searchTerm?: string;
      sort?: string;
    }
  ): Promise<PaginatedResponse<Property>> {
    const { page = 1, limit = 12 } = params;
    const offset = (page - 1) * limit;

    // Start building the query
    let query = supabase
      .from('property')
      .select('*', { count: 'exact' });

    // Apply filters if they exist
    if (filters) {
      if (filters.propertyState) {
        query = query.eq('state', filters.propertyState);
      }
      if (filters.propertyType) {
        query = query.eq('type', filters.propertyType);
      }
      if (filters.minPrice) {
        query = query.gte('price', parseFloat(filters.minPrice));
      }
      if (filters.maxPrice) {
        query = query.lte('price', parseFloat(filters.maxPrice));
      }
      if (filters.beds) {
        query = query.eq('beds', parseInt(filters.beds));
      }
      if (filters.searchTerm) {
        query = query.or(`title.ilike.%${filters.searchTerm}%,city.ilike.%${filters.searchTerm}%,state.ilike.%${filters.searchTerm}%`);
```

Figure 5. 5 Code snippets for properties search and filtering

The property search functionality implements a multi-criterion filtering system that processes user queries. The algorithm accepts multiple filter parameters including location, price range, property type and other features. The implementation utilizes Supabase's query builder with conditional chaining to construct dynamic SQL queries based on provided filters. The system employs pagination to handle large result sets, with configurable items per page and sort options.

5.5 System Operation

In this section, only important parts of the system operation will be discussed. These components represent the core functionality of the rental property platform and showcase how users interact with the system.

Register

Newcomers or first-time users can register via “Sign Up” in the navigation bar.

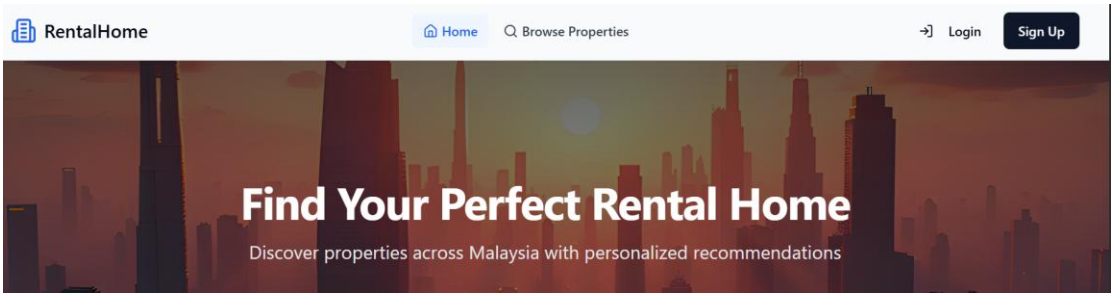


Figure 5. 6 Home Page and Navigation Bar

User will be redirected to the register page and a registration form will be displayed. Users must fill up all the required fields. The form has implemented form validation feature that will display error messages like shown in Figure 5. 7, Figure 5. 8, Figure 5. 9 if the user's input is incorrect. After every field is filled up, users can click sign up to proceed. There is an alternative login option when users click the "Continue with Google". In this example, only email registration processes will be shown, Google login will not be discussed.

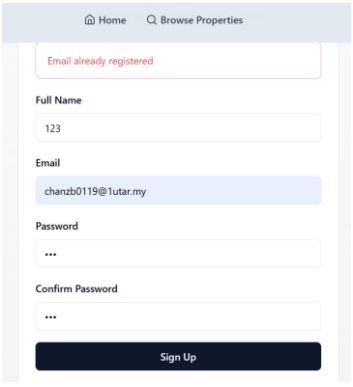


Figure 5. 7 Email being registered message

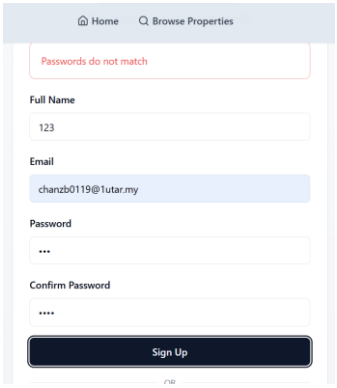


Figure 5. 8 Password do not match message

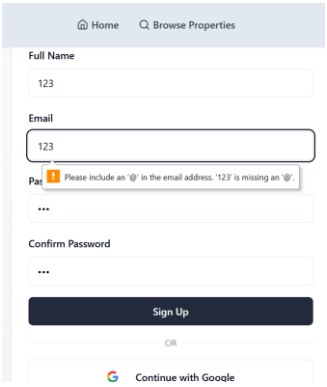


Figure 5. 9 Email format error message

Users are required to verify their account through their email before login to the website as shown in Figure 5. 10. Once users confirm their email verification, users can click "return the login" button and they will be redirected to the login page

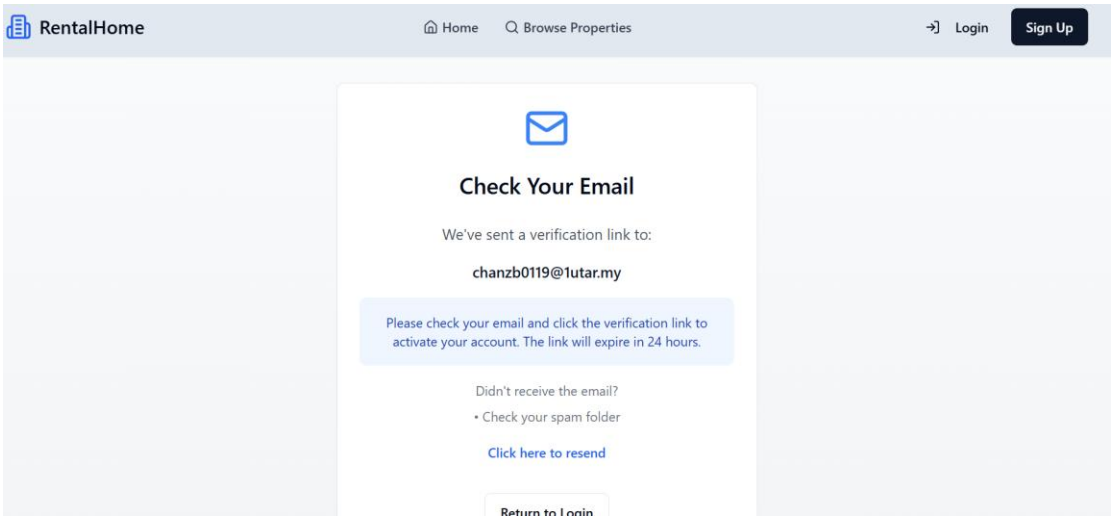


Figure 5. 10 Email verification page

Login

Registered user can click the "login" button in the navigation bar and proceed to login their account. They will be redirected to login page and they must fill up the login form. Similar to the registration form, login form does implement the validation feature that will display error messages to the users if their input is incorrect. Figure 5. 11 and Figure 5. 12 illustrates the error messages.

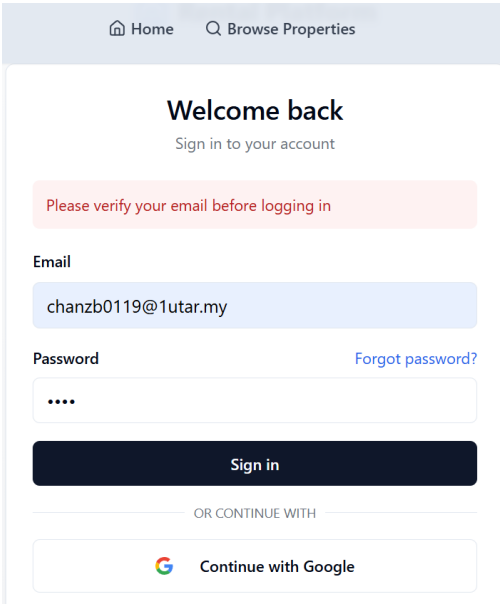


Figure 5. 11 Verification error message

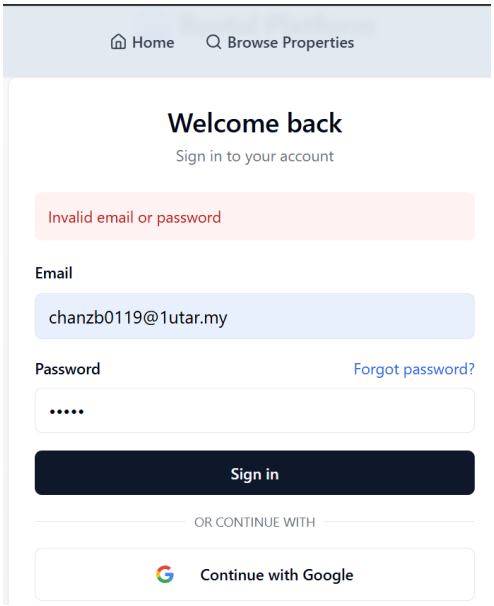


Figure 5. 12 Invalid credentials message

Once successfully logged in, users will be redirected to the landing page and their profile picture, or first username will be shown in the navigation bar as shown in Figure 5. 13.

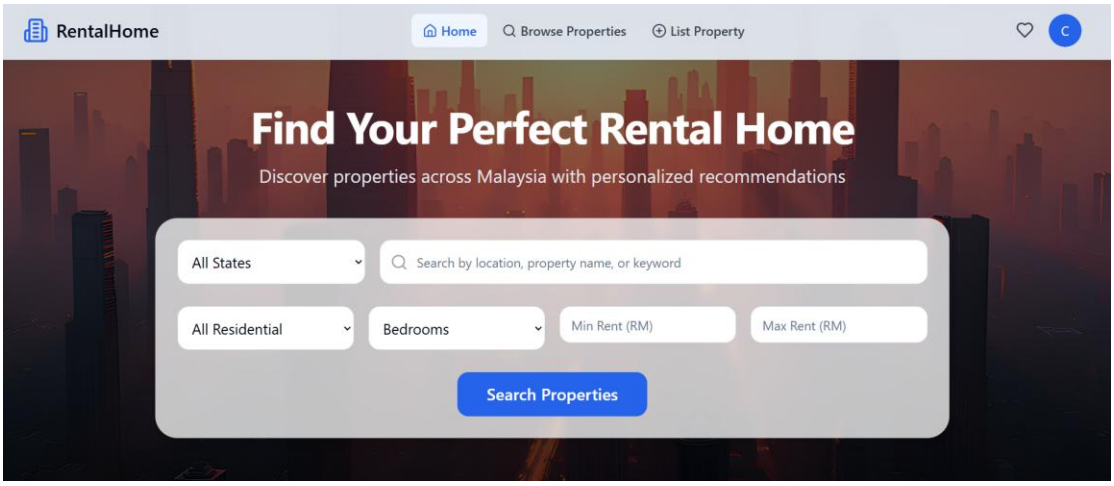


Figure 5. 13 Landing page with user icon being shown

For an administrator case, users with admin role will be redirected to admin dashboard instead of the landing page as shown in Figure 5. 14.

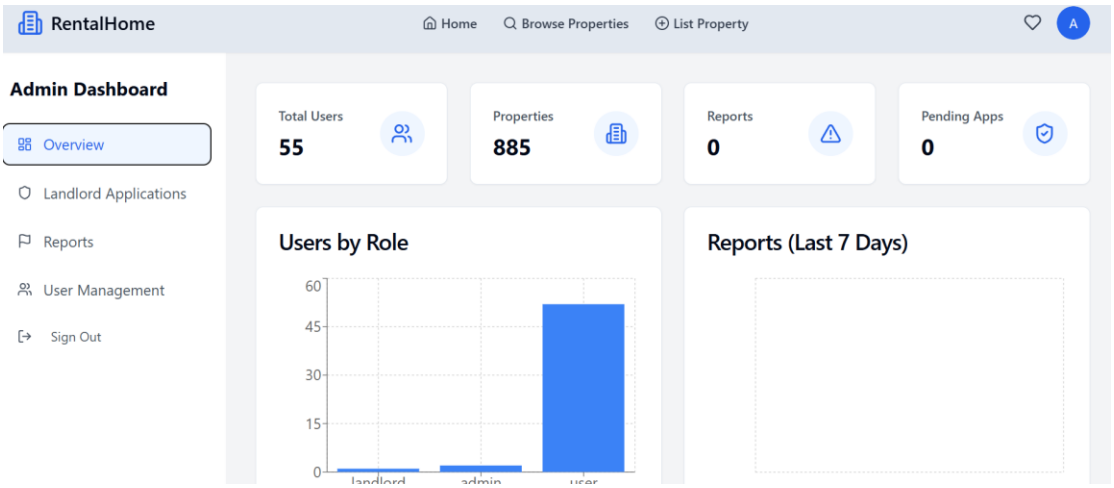


Figure 5. 14 Admin dashboard

Profile

Users can click the icon at the top right corner of the navigation bar. It will display a dropdown list with profile and logout options as shown in Figure 5. 15. Click “My profile” to proceed to profile page.

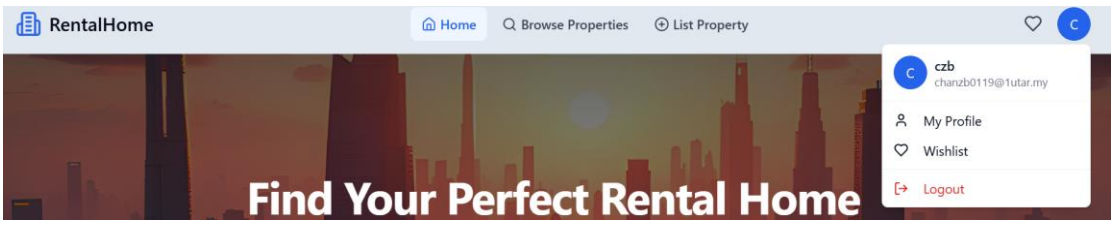


Figure 5. 15 Dropdown list for profile

There are three tabs on the profile page. Profile tab to manage the user information including changing profile icon, display name and phone number. Wishlist tab to show users' favorite properties. Become a landlord tab for landlord application for those non-landlord users. Figure 5. 16, Figure 5. 17, Figure 5. 18 illustrate the tabs of profile page.

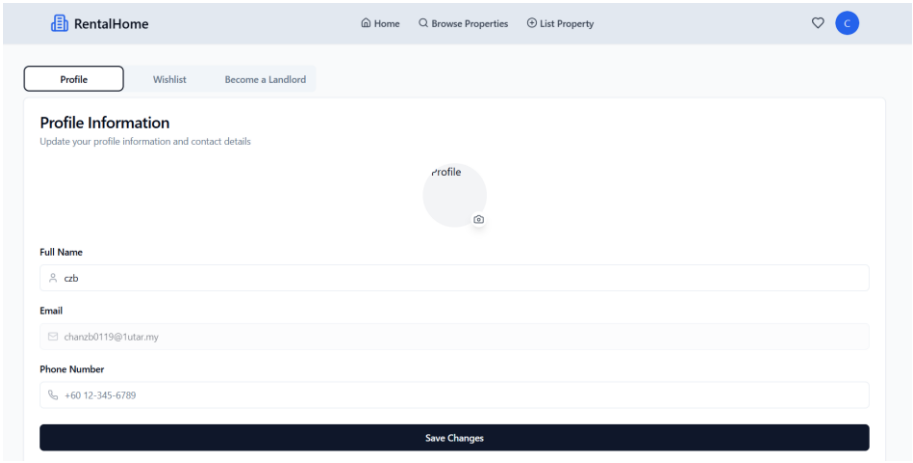


Figure 5. 16 Profile tab

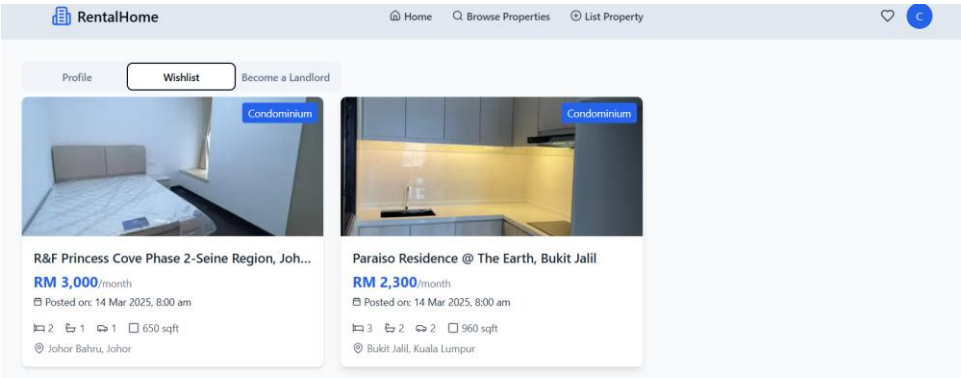


Figure 5. 17 Wishlist tab

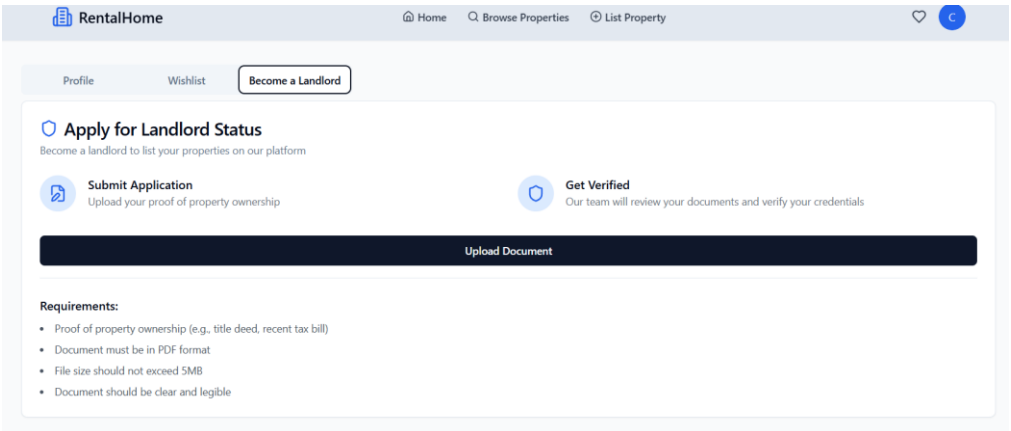


Figure 5. 18 Become a landlord tab

Apply for landlord status

For news user to upload their properties, first they must upload relevant document in the landlord application tab. After successfully uploading the document, they have to wait for the admin to approve their application as shown in Figure 5. 19.

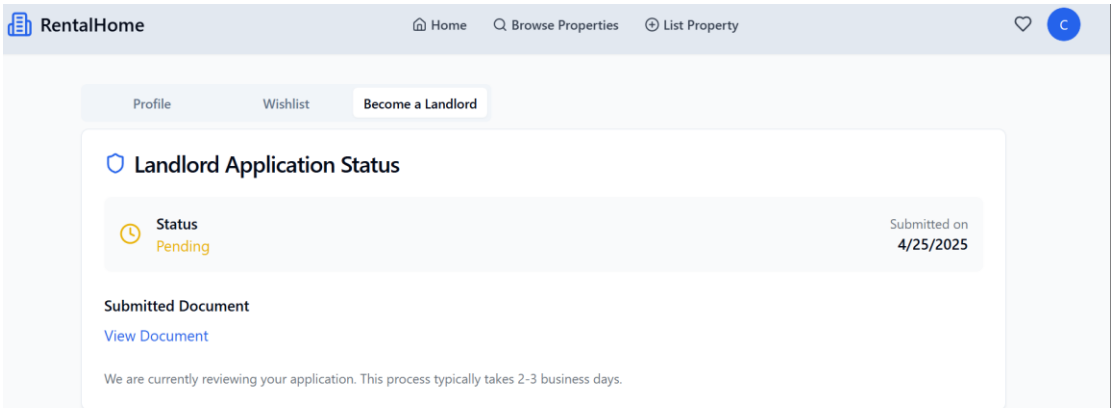


Figure 5. 19 Pending landlord application

Approve/reject landlord application by admin

Admin can review the landlord applications in “landlord applications’ tab on the admin dashboard as shown in Figure 5. 20.

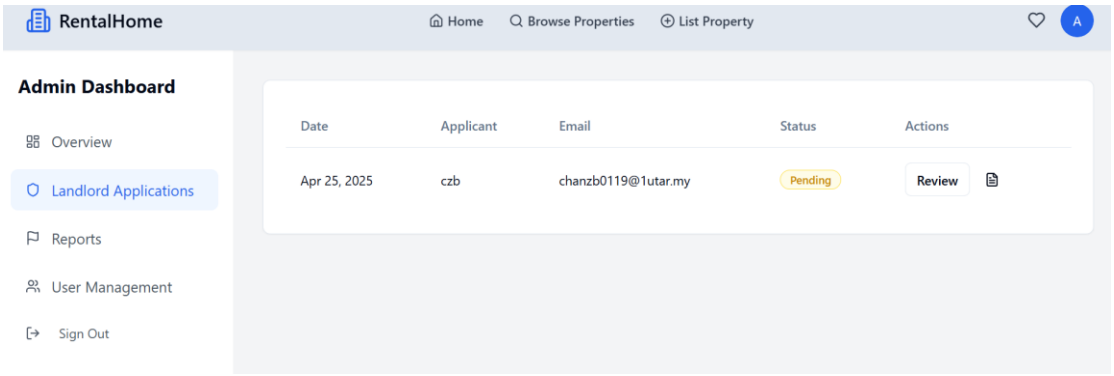


Figure 5. 20 Admin dashboard with landlord application tab

Click on “review” button and the system will display a dialog window for admin to approve or reject the application as shown in Figure 5. 21. Once the application is approved, the user’s role will be updated to “landlord” status.

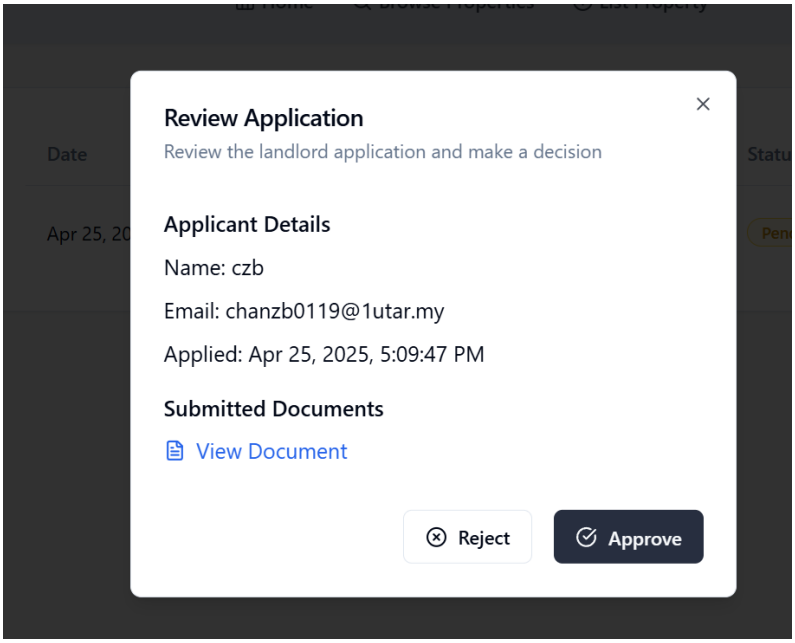


Figure 5. 21 Dialog window for reviewing landlord application

Create a new property listing

Users can upload their property through the "list property" in navigation bar. After redirected to the create property page, the system will check whether the user is a verified landlord and display error message to those non-landlord users as shown in Figure 5. 22.

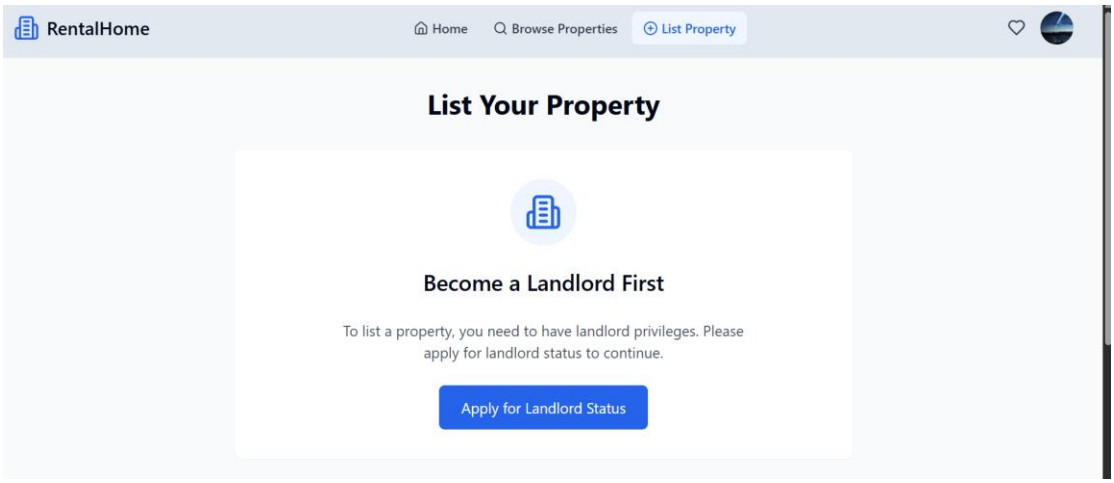


Figure 5. 22 Error message for non-landlord user

For landlord user, a multi-section form will be displayed to them. The property listing form is implemented as a multi-section interface that guides users through the process of creating a comprehensive property listing. Basic information captures fundamental

property details including the title and property type. Property details section collects specific attributes of the property. Figure 5. 23 shows the relevant sections.

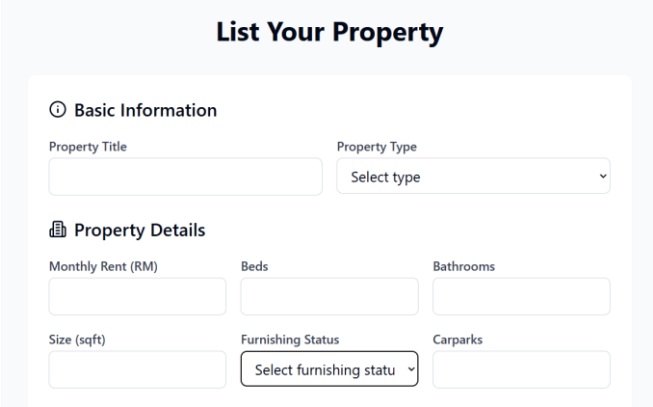


Figure 5. 23 Property listing form 1

The location information section implements Google Maps interface with address autocomplete functionality. This feature allows users to search for specific addresses, pinpoint the exact location on the interactive map and automatically extract state and city information. Figure 5. 24 illustrates the location components of the listing form.

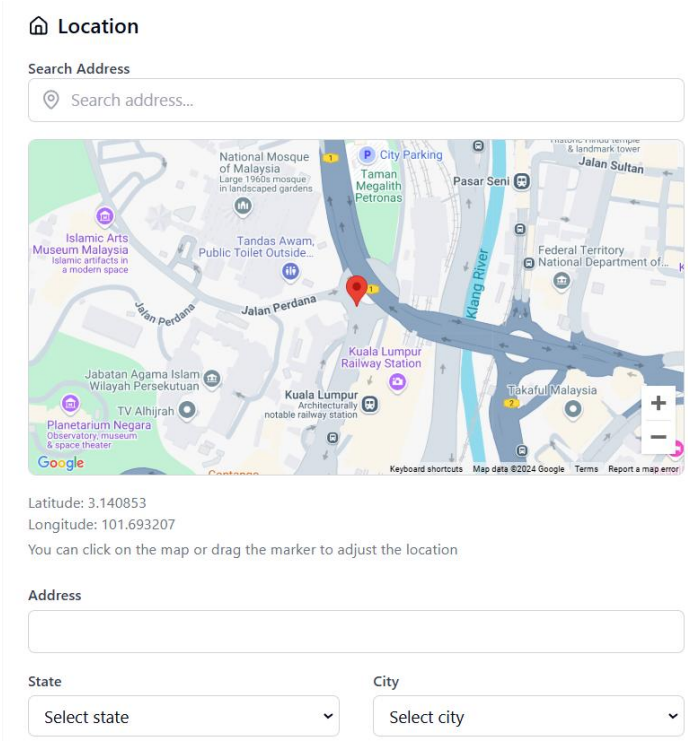


Figure 5. 24 Property listing form 2

Next, the amenities selection provides a checklist of common property amenities and the description section allows owners to describe their listing.

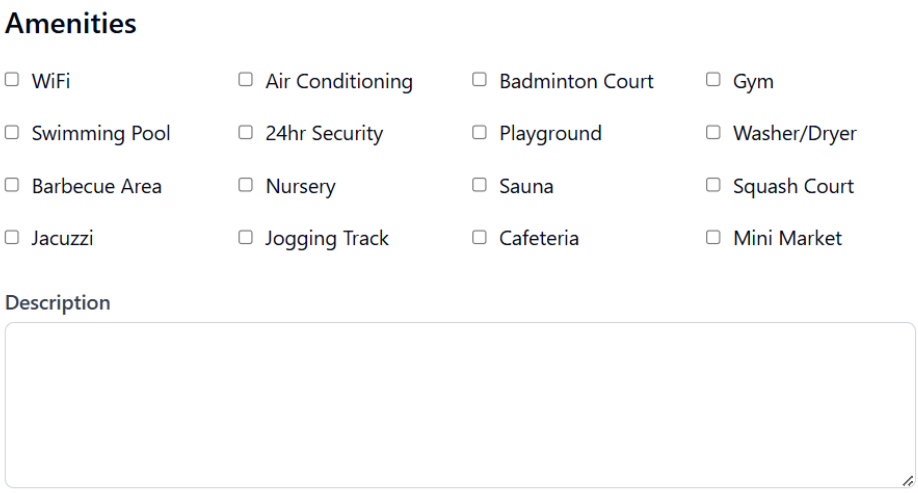


Figure 5. 25 Property listing form 3

Lastly, image upload components allow users to upload properties images and the images can be previewed. Figure 5. 26 illustrates the upload image component.

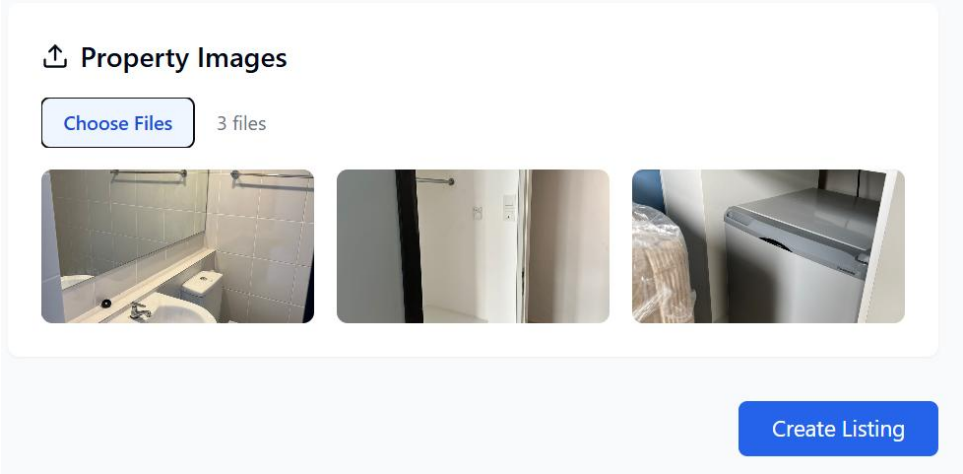


Figure 5. 26 Property listing form 4

View properties

Users can view all the properties which had been created by the landlord. All the properties are dynamically fetched from the database. Users can click on the property card as shown in Figure 5. 27 to view a single property detail page.

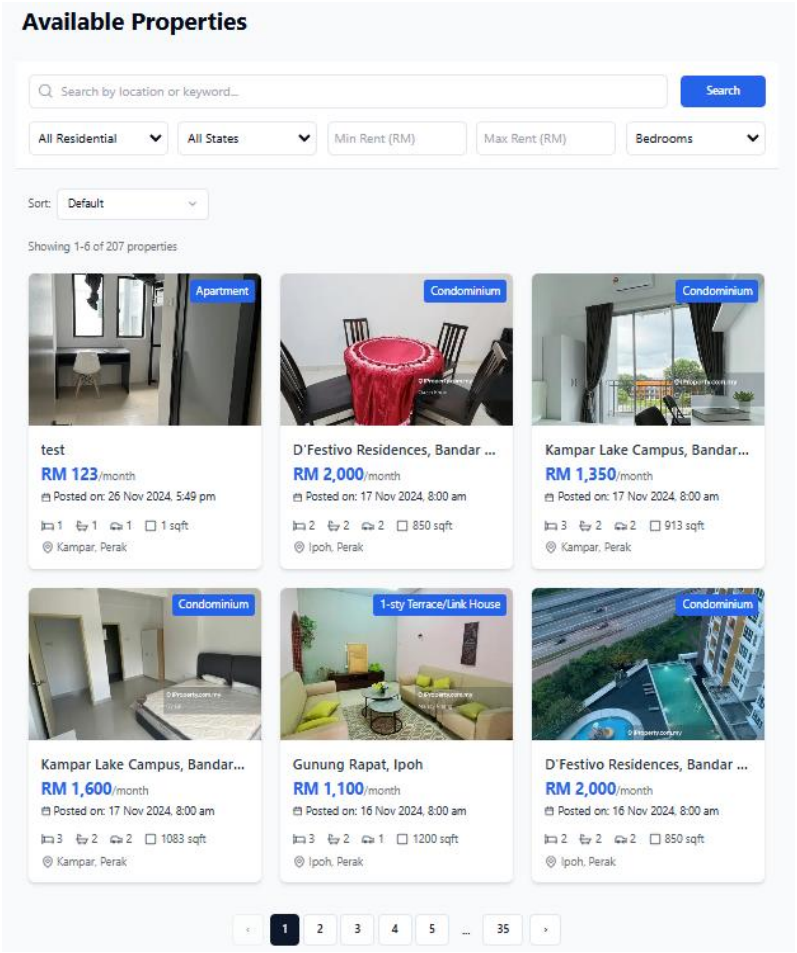


Figure 5. 27 Property view page

After users click a property card, they will be redirected to the property details page which provides the details of the selected property. It contains an image gallery for property images, a sticky container that keeps essential actions such as wishlist, contact landlord, report listing accessible while scrolling and sections displaying key details like price, bedrooms, amenities, and property specifications. An embedded Google Maps component shows the property's location. The page concludes with a recommendation section that leverages the hybrid filtering algorithm to suggest similar properties based on the user's preferences. Figure 5. 28 and Figure 5. 29 show the property details page.

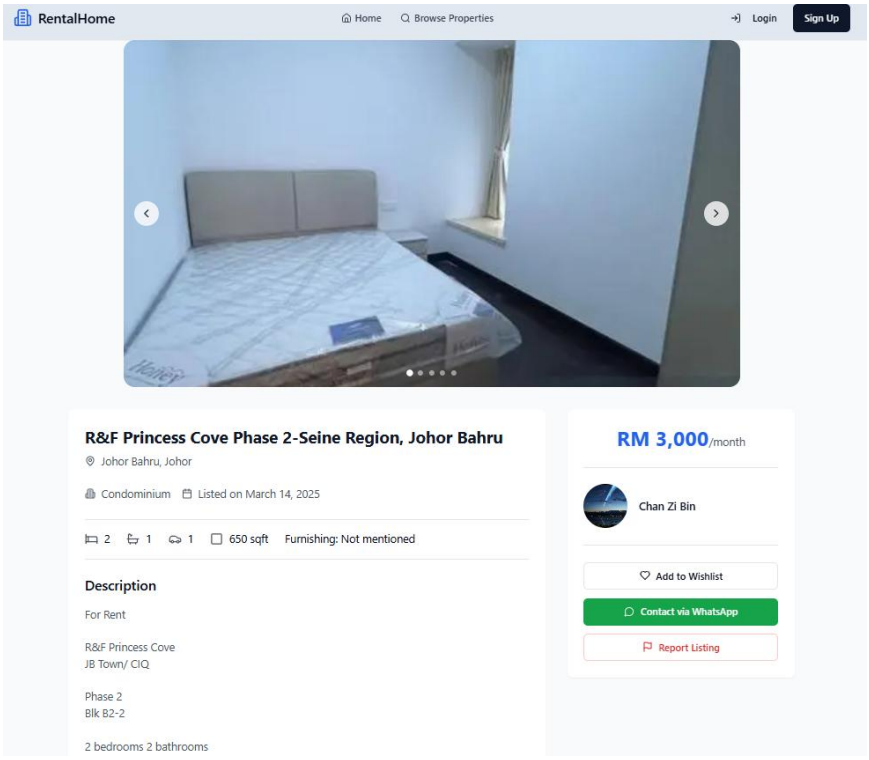


Figure 5. 28 Property details page1

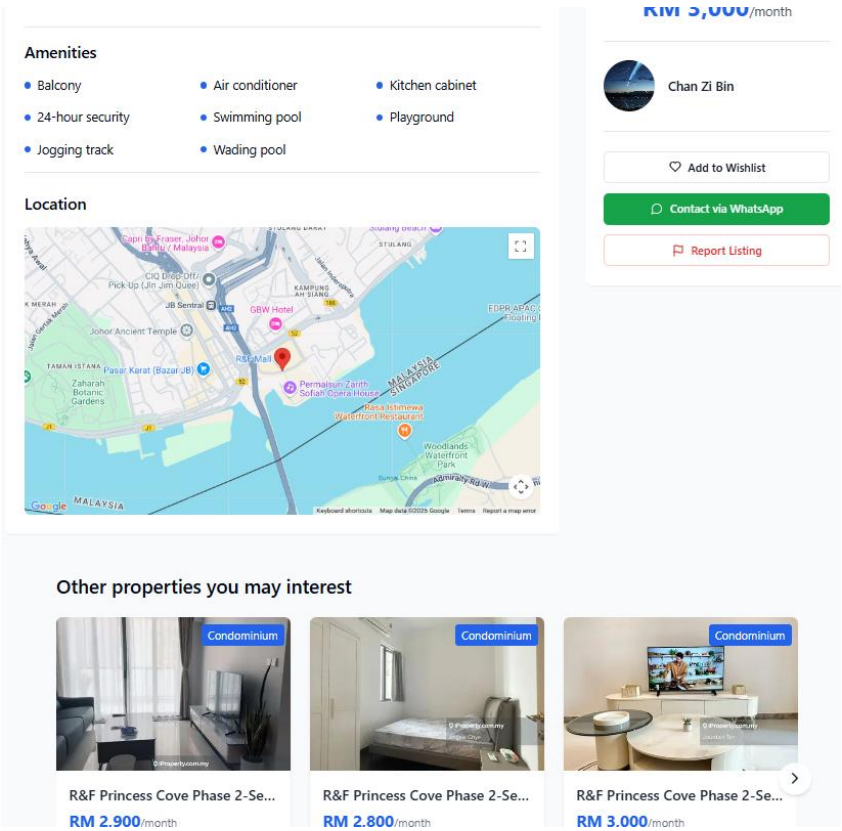


Figure 5. 29 Property details page 2

Recommendation system

The website implements a hybrid personalized recommendation system, primarily on the landing page and property details page. For logged-in users, the system retrieves and recommends relevant properties based on their viewing history, wishlist additions, and search patterns (collaborative filtering), while also considering property attributes like location, price range, and amenities (content-based filtering). For new or non-logged-in users, recommendations are based on popular properties and content similarities. Figure 5. 30 and Figure 5. 31 show different recommended properties on the landing page for different users. On the property details page, the "Other properties you may interest" section showcases similar properties based on the currently viewed listing.

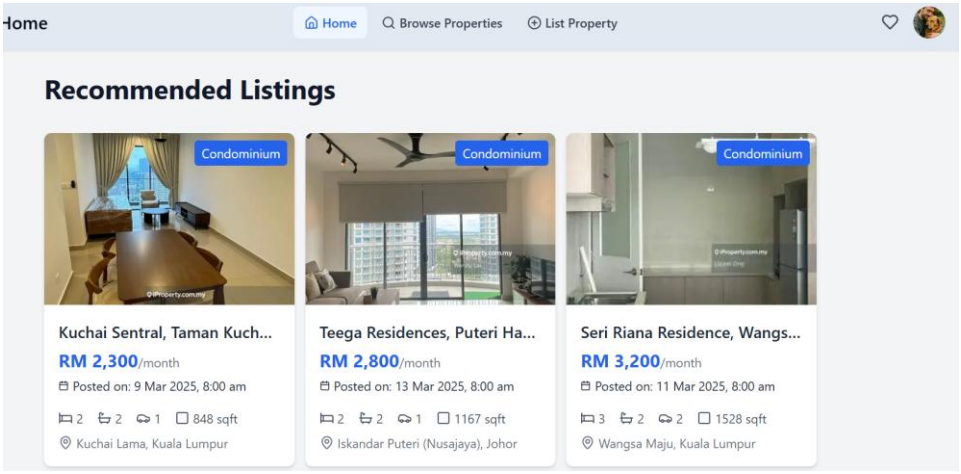


Figure 5. 30 Recommendation section for user1

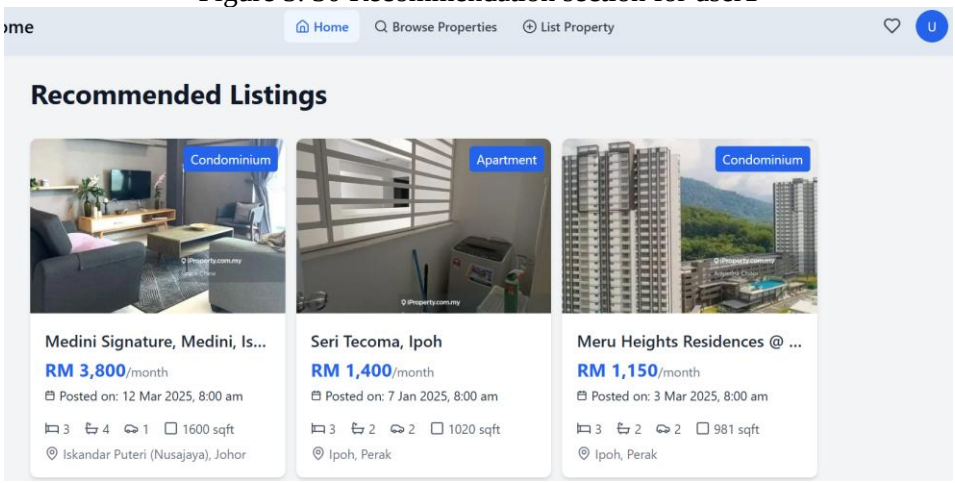


Figure 5. 31 Recommendation section for user2

Mobile responsive

This project is a mobile responsive website where the UI adapts depending on the user's device. Figure 5. 32 is screenshots of the mobile views which demonstrate how the

interface elements resize, reflow and reorganize to create an optimal viewing experience on smaller screens.

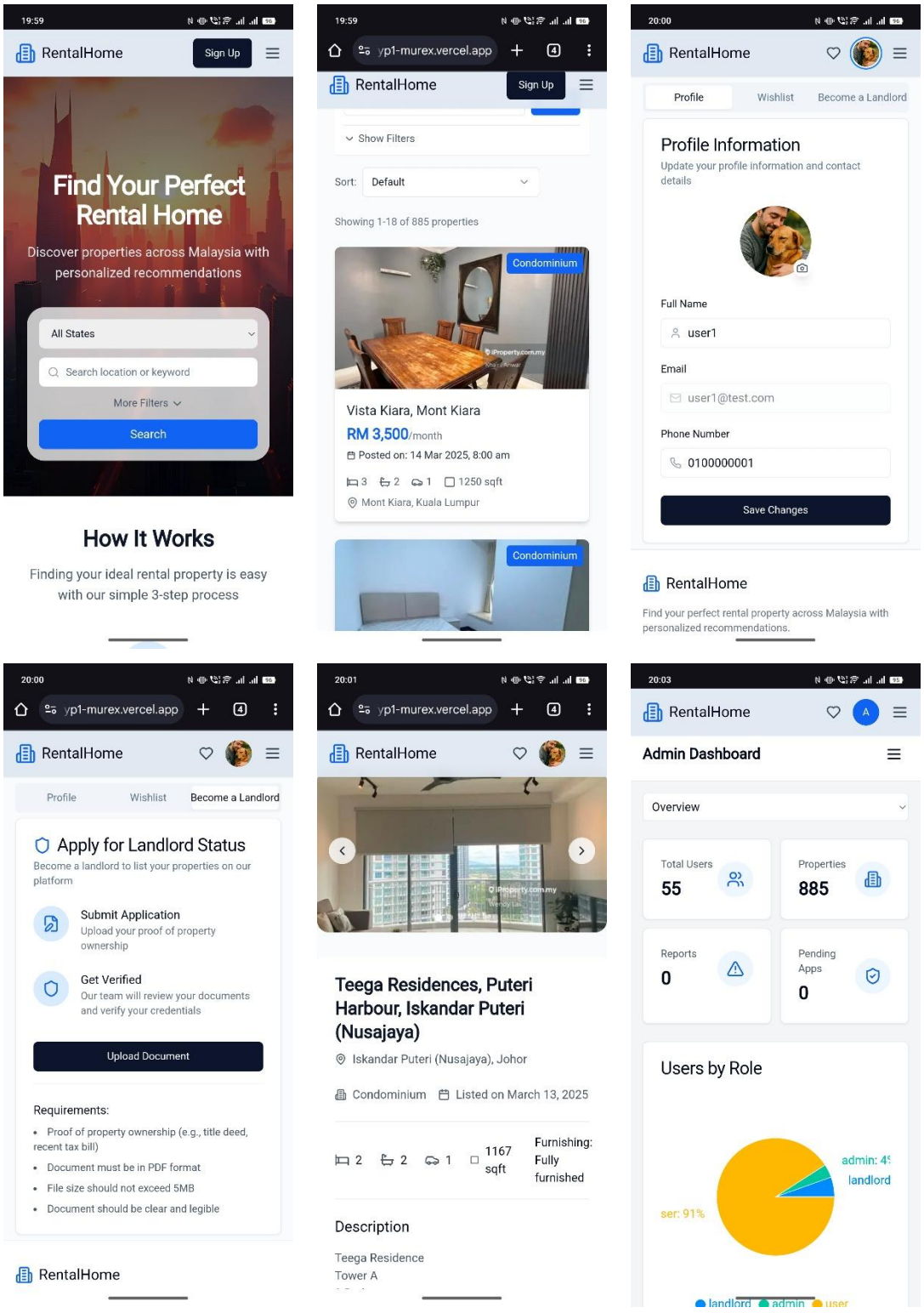


Figure 5. 32 Screenshots of mobile views

5.6 Implementation Issues and Challenges

During the development of this rental property website, several challenges emerged. One of the most challenging issues was insufficient historical data to train a robust collaborative filtering model. To address this challenge, data transformation techniques were applied to real user interaction datasets obtained from other property websites and sources. This approach allowed for the creation of a synthetic but representative dataset that preserved interaction patterns while adapting them to the current property data.

Next, the learning curve associated with Next.js and React component architecture posed another challenge during the initial project development. Despite having prior experience with JavaScript, the transition to React's component-based and Next.js's server-side rendering framework requires some adaptation time. Understanding key concepts such as state management, props passing, component lifecycle methods and the differences between client-side and server-side rendering required extensive self-study and experimentation. Particularly challenging aspects included mastering React hooks, context API for state management and Next.js routing system. This learning process extended the initial development timeline but ultimately resulted in a more structured, maintainable codebase as familiarity with these technologies improved throughout the project lifecycle.

Other than that, Integration with various external APIs raises another challenge. Different APIs returned data in inconsistent structures and formats, requiring the development of adapter layers to normalize responses before they could be used within the application. The Google Maps API integration was especially challenging, with issues arising around geocoding accuracy that needed extensive testing and refinement to ensure reliable property location services.

Additionally, code that functioned flawlessly in the local development environment often encountered unexpected issues in the production environment after deployed. Environment-specific bugs related to API key configurations, CORS policies and database connections required debugging. These deployment issues required more thorough testing procedures and better error tracking to efficiently find and fix problems that only appeared in the production environment.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

6.1 System Testing and Performance Metrics

This section focuses on discussing the performance of the recommendation system.

6.1.1 Testing Methodology

The recommender system was implemented using a hybrid approach that combines collaborative filtering with content-based techniques. NCF was selected as the primary model due to its effectiveness in capturing complex user-item interaction patterns. For evaluation purposes, the dataset was split into training (80%) and testing (20%) sets.

Testing was conducted under three different conditions:

1. Content-based approach alone
2. Basic integration with NCF
3. Integration with NCF after hyperparameter fine-tuning

6.1.2 Performance Metrics

Table 6. 1 Recommendation system performance metrics

Condition	NDCG@5	HR@5	Precision@5	Recall@5
Content-based alone	0.5897	0.0233	0.0047	0.0078
Integration with NCF	0.3594	0.6939	0.1755	0.2578
Integration with Fine-tuned NCF	0.6093	0.9592	0.3796	0.5388

Table 6. 1 presents the performance metrics for the property recommendation system under three different implementation approaches. The content-based approach alone, which relies solely on TF-IDF vectorization and cosine similarity, achieved an NDCG@5 score of 0.5897, indicating moderate ranking quality. However, this approach demonstrated notably weak performance in other metrics with a hit rate (HR@5) of only 0.0233, precision of 0.0047, and recall of 0.0078, suggesting limited effectiveness in identifying relevant properties for users.

When integrating the basic Neural Collaborative Filtering (NCF) model, a significant improvement was observed in hit rate, precision, and recall metrics. The HR@5 increased dramatically to 0.693. Similarly, precision and recall showed substantial increases, reaching 0.1755 and 0.2578 respectively. However, the NDCG@5 score

decreased to 0.3594, suggesting that while more relevant properties were being recommended but their ranking order was not optimal.

The most impressive results were achieved with the fine-tuned NCF integration, which employed hyperparameter optimization through grid search. This approach delivered balanced improvements across all metrics, with NDCG@5 increasing to 0.6093, HR@5 reaching 0.9592, precision improving to 0.3796, and recall improving to 0.5388. These results indicate that nearly 96% of users received at least one relevant property in their top 5 recommendations, with more than half of all relevant properties being successfully recommended.

6.2 Testing Setup and Result

This section focuses on the test cases for each use case. The relevant screenshots of results that have not been shown in Chapter 5.4 System Operation will be included in Appendix B: Test Cases Results.

Table 6. 2 User registration test case

Test Case Name: User Registration			ID: TC01	
Main Flow	Test Steps		Expected Result	Result
	1	User navigates to the signup page	The system displays the registration form with name, email, password fields	PASS
	2	User enters valid details	The form accepts the input	PASS
	3	User submits the form	The system creates a new account and redirects to verification reminder page	PASS
Alternative Flows	2a	User enters an email that already exists	System shows "Email already registered" error message	PASS
	2b	User enters mismatched passwords	System shows "Passwords do not match" error	PASS
	2c	User enters an invalid email format	System shows email validation error	PASS

Table 6. 3 User login test case

Test Case Name: User Login			ID: TC02	
Main Flow	Test Steps		Expected Result	Result

	1	User navigates to the login page	The system displays the login form	PASS
	2	User enters valid email and password	The form accepts the input	PASS
	3	User submits the form	The system authenticates the user and redirects to home page	PASS
Alternative Flows	2a	User enters incorrect credentials	System shows "Invalid email or password" error message	PASS
	2b	User logins with unverified email	System shows "Please verify your email" message	PASS

Table 6. 4 Profile management test case

Test Case Name: Profile Management			ID: TC03	
Main Flow	Test Steps		Expected Result	Result
	1	User navigates to profile page	The system displays user profile information	PASS
	2	User updates profile information (name, phone)	The form accepts the changes	PASS
	3	User uploads a profile image	The system accepts and displays the new image	PASS
	4	User saves changes	The system updates the profile and shows success message	PASS

Table 6. 5 Property search test case

Test Case Name: Property Search			ID: TC04	
Main Flow	Test Steps		Expected Result	Result
	1	User enters search term in search field	The system displays the input in search field	PASS
	2	User selects filters options	The filters show as selected	PASS
	3	User clicks "Search" button	The system displays properties matching the criteria	PASS

Alternative Flows	3a	No properties match search criteria	System shows "No properties found" message	PASS

Table 6. 6 Property sorting test case

Test Case Name: Property Sorting			ID: TC05	
Main Flow	Test Steps		Expected Result	Result
	1	User views property listing page	The system displays properties in default order	PASS
	2	User selects sort option (price low to high)	The dropdown shows selected option	PASS
	3	System applies the sorting	Properties are displayed in ascending price order	PASS

Table 6. 7 Property listing creation test case

Test Case Name: Profile Listing Creation			ID: TC06	
Main Flow	Test Steps		Expected Result	Result
	1	Landlord navigates to create listing page	The system displays the property listing form	PASS
	2	Landlord enters property details	The form accepts all inputs	PASS
	3	Landlord uploads property images	The system displays image previews	PASS
	4	Landlord submits the form	The system updates the profile and shows success message	PASS
Alternative Flows	1a	User is not a landlord	System redirects to landlord application page	PASS
	2b	Landlord leaves required fields empty	System highlights missing fields	PASS

Table 6. 8 Landlord application test case

Test Case Name: Landlord Application			ID: TC07	
Main Flow	Test Steps		Expected Result	Result
	1	User navigates to landlord application	The system displays application form	PASS
	2	User uploads required documents	The system accepts the documents	PASS

	3	User submits application	The system creates application with "pending" status	PASS
Alternative Flows	2a	User uploads invalid document format	System shows file format error	PASS
	3a	User already has a pending application	System shows current application status	PASS

Table 6. 9 Property wishlist test case

Test Case Name: Property Wishlist			ID: TC08	
Main Flow	Test Steps		Expected Result	Result
	1	User views a property listing	The system displays property details	PASS
	2	User clicks "Add to Wishlist"	The system adds property to user's wishlist	PASS
	3	User navigates to wishlist page	The system displays all saved properties	PASS
Alternative Flows	2a	User removes property from wishlist	System removes the property from wishlist	PASS

Table 6. 10 Landing page recommendations test case

Test Case Name: Landing Page Recommendations			ID: TC09	
Main Flow	Test Steps		Expected Result	Result
	1	Logged-in users visit homepage	The system displays personalized property recommendations	PASS
Alternative Flows	1a	User is not logged in or new user with no history	System shows random popular properties	PASS

Table 6. 11 Property details recommendations test case

Test Case Name: Property Details Recommendations			ID: TC010	
Main Flow	Test Steps		Expected Result	Result
	1	User views property details page	The system displays similar properties section	PASS
	2	User scrolls to similar properties	Properties with similar attributes are shown	PASS

Alternative Flows	2a	User is not logged in	System shows content-based recommendations only	PASS

Table 6. 12 Admin dashboard test case

Test Case Name: Admin Dashboard			ID: TC011	
Main Flow	Test Steps		Expected Result	Result
	1	Admin logs into the system	The system authenticates and identifies admin role	PASS
	2	Admin navigates to dashboard	The system displays admin dashboard with statistics	PASS
	3	Admin reviews different dashboard sections	Dashboard shows users, properties, and reports data	PASS
Alternative Flows	1a	Non-admin user attempts to access	System redirects to home page	PASS

Table 6. 13 Landlord application review

Test Case Name: Landlord Application Review			ID: TC012	
Main Flow	Test Steps		Expected Result	Result
	1	Admin navigates to landlord applications	The system displays pending applications	PASS
	2	Admin reviews application documents	The system allows document viewing	PASS
	3	Admin approves application	User role is updated to landlord	PASS
Alternative Flows	3a	Admin rejects application	Application status is updated to rejected	PASS

Table 6. 14 Property Report Handling

Test Case Name: Property Report Handling			ID: TC013	
Main Flow	Test Steps		Expected Result	Result
	1	Admin navigates to property reports	The system displays reported properties	PASS
	2	Admin reviews report details	The system shows report category and description	PASS
	3	Admin takes action (dismiss/delete)	The system processes the admin decision	PASS

6.3 Objectives Evaluation

The implementation and testing of the rental property website has provided evidence for evaluating the fulfillment of the project's objectives. Regarding the first objective of analyzing and implementing a hybrid recommendation system, the project successfully delivered a recommendation engine that combines content-based and collaborative filtering techniques. The system effectively analyzes property attributes such as location, price range, and amenities while also incorporating user behavior patterns including viewing history and wishlist items. Test cases demonstrated that the recommendation algorithm consistently delivers personalized property suggestions that align with user preferences.

The second objective of developing a comprehensive rental property website with essential features using the React framework has been thoroughly achieved. The implementation resulted in a fully functional platform built on Next.js with React, featuring user authentication, responsive property listing management, and interactive search capabilities. The test results confirm the successful implementation of core functionalities, with all test cases from TC01 through TC08 validating key features such as user registration, property searching, listing creation and wishlist management. The administrative capabilities, landlord verification system, and integrated property management tools were all successfully implemented and validated through test cases TC11, TC12, and TC06 respectively.

Evaluating the third objective regarding the effectiveness of the hybrid recommendation system has shown promising results. Test case TC9 specifically validated that the recommendation system successfully provides personalized suggestions based on user behavior, while TC10 ensured that content-based similar property recommendations function correctly. The system is able to handle different user scenarios, including new users with no history and users with diverse browsing patterns. Although comprehensive quantitative metrics such as precision and recall rates would require larger-scale user testing over an extended period, the initial test results indicate that the hybrid approach effectively improves property discovery by balancing specific property attribute matching with collaborative insights from user interaction patterns. The implemented recommendation system has satisfied the core

requirement of enhancing content discovery by providing contextually relevant property suggestions tailored to individual user preferences and behavior.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

7.1 Conclusion

In summary, this project successfully developed a modern rental property website using Next.js and React framework, integrated with a hybrid recommendation system combining content-based and collaborative filtering techniques. The project successfully implemented essential features including detailed property search with multiple filters, interactive map integration, wishlist management and an administrative dashboard for platform oversight. These features create a comprehensive property rental ecosystem that serves the needs of both property seekers and landlords. Additionally, testing results confirmed the successful implementation of all core objectives, with most test cases passing validation across user authentication, property management and recommendation system functionality. The overall project demonstrates a successful application of modern web development techniques and recommendation algorithms in creating a practical solution for the property rental market.

7.2 Recommendation

Based on the development experience and testing results, several recommendations for future work and improvements can be made. First, for the recommendation engine, implementing attention-based Neural Collaborative Filtering would capture the relative importance of different property features for individual users, potentially improving recommendation accuracy. The current static NCF model could be upgraded with online stochastic gradient descent for incremental updates without full retraining, reducing computational requirements while keeping recommendations fresh. Deep feature extraction from property images using pre-trained convolutional neural networks (ResNet or EfficientNet) could capture visual aspects like interior style and spatial arrangements that text descriptions miss, improving recommendation relevance visually driven users.

Next, adding a payment system or gateway such as integrating Stripe or PayPal API for landlords to list their properties would meet real-world use case scenarios. This would create a sustainable business model where landlords pay a fee to list their properties on the platform. Such a payment integration would not only provide a revenue stream but

also potentially increase the quality of listings as landlords would be more invested in creating complete and accurate property information when there's a financial commitment involved.

REFERENCES

[1] Z. Yang, W. Zhiruo, C. Chang, D. Xiaoyi, W. Chengliang, and N. Yanjie, "Impact of Web Page House Listing Cues on Internet Rental," *Applied Mathematics and Nonlinear Sciences*, vol. 6, no. 2, pp. 483–498, Jul. 2021, doi: 10.2478/amns.2021.2.00021.

[2] G. Boeing, M. Besbris, A. Schachter, and J. Kuk, "Housing Search in the Age of Big Data: Smarter Cities or the Same Old Blind Spots?," *Hous Policy Debate*, vol. 31, no. 1, pp. 112–126, 2021, doi: 10.1080/10511482.2019.1684336.

[3] A. L. Monteverde, J. J. S. Maderazo, K. C. M. Cruz, and Ni. A. Magnaye, "Web-Based Rental House Smart Finder using Rapid Application Development basis for Evaluation of ISO 205010," *International Journal of Metaverse*, vol. 1, no. 1, pp. 1–4, Apr. 2023, doi: 10.54536/ijm.v1i1.1464.

[4] "Website Quality and Intention to Use Real Estate Website in Housing Market," *European Journal of Business and Management*, Jun. 2021, doi: 10.7176/ejbm/13-11-09.

[5] M. Kompan, P. Gaspar, J. MacIna, M. Cimerman, and M. Bielikova, "Exploring Customer Price Preference and Product Profit Role in Recommender Systems," *IEEE Intell Syst*, vol. 37, no. 1, pp. 89–98, 2022, doi: 10.1109/MIS.2021.3092768.

[6] "iBilik." Accessed: Aug. 08, 2024. [Online]. Available: https://www.ibilik.my/

[7] "PropertyGuru Malaysia." Accessed: Aug. 23, 2024. [Online]. Available: https://www.propertyguru.com.my/

[8] "EdgeProp." Accessed: Aug. 08, 2024. [Online]. Available: https://edgeprop.my/

[9] G. T. Kurniaji, Y. S. Nugroho, and S. Islam, "A preliminary empirical study of react library related questions shared on stack overflow," *Computer Science and Information Technologies*, vol. 4, no. 1, pp. 14–23, Mar. 2023, doi: 10.11591/csit.v4i1.pp14-23.

[10] A. Shukla, “Modern JavaScript Frameworks and JavaScript’s Future as a FullStack Programming Language,” *Journal of Artificial Intelligence & Cloud Computing*, pp. 1–5, Dec. 2023, doi: 10.47363/JAICC/2023(2)144.

[11] A. Aurelia, W. Wasino, D. Chandra, and T. B. Jap, “Developing Website-Based Information System Applications to Map PT. XYZ’s Properties Using Next.JS Framework with Haversine Method,” *International Journal of Application on Sciences, Technology and Engineering*, vol. 1, no. 1, pp. 59–64, Feb. 2023, doi: 10.24912/ijaste.v1.i1.59-64.

[12] “Airbnb.” Accessed: Dec. 03, 2024. [Online]. Available: https://www.airbnb.com/

[13] “Lazada”, Accessed: Dec. 03, 2024. [Online]. Available: https://www.lazada.com.my/

[14] H. Papadakis, A. Papagrigoriou, E. Kosmas, C. Panagiotakis, S. Markaki, and P. Fragopoulou, “Content-Based Recommender Systems Taxonomy,” *Foundations of Computing and Decision Sciences*, vol. 48, no. 2, pp. 211–241, Jun. 2023, doi: 10.2478/fcds-2023-0009.

[15] B. McFee, L. Barrington, and G. Lanckriet, “Learning content similarity for music recommendation,” *IEEE Trans Audio Speech Lang Process*, vol. 20, no. 8, pp. 2207–2218, 2012, doi: 10.1109/TASL.2012.2199109.

[16] A. Verma, “A Comparative Study of AI-based Recommender Systems,” *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 11, no. 1, pp. 827–830, Apr. 2020, doi: 10.17762/turcomat.v11i1.13564.

[17] T. Ma *et al.*, “Social network and tag sources based augmenting collaborative recommender system,” Apr. 01, 2015, *Maruzen Co., Ltd.* doi: 10.1587/transinf.2014EDP7283.

[18] X. Wu, “Review of collaborative filtering recommendation systems,” *Applied and Computational Engineering*, vol. 43, no. 1, pp. 76–82, Feb. 2024, doi: 10.54254/2755-2721/43/20230811.

[19] A. SANTIAGO, B. JESÚS, O. FERNANDO, and M. RICARDO, “Robust Model-Based Reliability Approach to Tackle Shilling Attacks in Collaborative

Filtering Recommender Systems," *IEEE Access*, vol. 7, pp. 41782–41798, Feb. 2019, doi: https://doi.org/10.1109/access.2019.2905862.

[20] Pierre-Jean Lajoie Michaud, "Agile software development: everything you need to know." Accessed: Dec. 04, 2024. [Online]. Available: https://www.nexapp.ca/en/blog/agile-software-development

[21] Kishalaya, "Introduction to NCF architecture in Recommender systems." Accessed: May 05, 2025. [Online]. Available: https://medium.com/@kumarkishalaya/introduction-to-ncf-architecture-in-recommender-systems-ad08da1a8705

[22] Z. Bin Chan, "fyp." Accessed: Apr. 27, 2025. [Online]. Available: https://github.com/chanzb0119/fyp

APPDENDIX

Appendix A: Use Case Descriptions

Sign Up

Use Case Name: Sign Up	**ID:** 1	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to create an account to access platform features		
Brief Description: This use case describes the process of new user registration on the property rental platform		
Trigger: User wants to create a new account on the platform **Type:** External		
Relationship: Association : User Include : Create profile Extend : - Generalization : -		
Preconditions: • User is not registered in the system • User has a valid email address		
Postconditions: • User account is registered and verified • User can log in using registered credentials		
Normal Flow of Events: 1. User initiates registration by clicking Sign Up button 2. User enters registration information (username, email, password, and confirmation password) 3. User selects Sign Up button 4. System validates registration information 5. System sends verification email to user 6. User verifies email address by clicking verification link 7. System creates new user account 8. System redirects user to the Login page		
Sub Flows: Not applicable		
Alternate Flow: 2a. Invalid input information: system displays error messages and prompts user to enter correct details 3a. Email already exists: system displays message "Email already registered" and prompts user to use different email		

Login

Use Case Name: Login	**ID:** 2	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to access their account and platform features		
Brief Description: This use case describes how a registered user logs into their account on the property rental website.		
Trigger: User wants to access their account **Type:** External		
Relationship: Association : User Include : - Extend : Logout Generalization : -		
Preconditions: • User has registered an account • User is not currently logged in		
Postconditions: • User is successfully authenticated • User session is created		
Normal Flow of Events: 1. User clicks on Login button 2. System displays login form 3. User enters credentials (email and password) 4. User clicks Submit button 5. System validates user credentials 6. System authenticates user 7. System creates user session 8. System redirects user to homepage with personalized recommendation		
Sub Flows: Not applicable		
Alternate Flow: 3a. User selects "Login with Google" option: 1. System redirects to Google authentication page 2. User authenticates with Google credentials 3. System creates account session using Google authentication 5a. Invalid credentials: 1. System displays "Invalid email or password" message 2. System prompts user to retry login		

View Profile

Use Case Name: View Profile	**ID:** 3	**Importance Level:** Medium

Primary Actor: User	**Use Case Type:** Details, Essential
Stakeholders and Interest: Users – wants to view and manage their profile information	
Brief Description: This use case describes how a user can view their profile information and access profile management options.	
Trigger: User wants to access their profile information **Type:** External	
Relationship: Association : User Include : - Extend : Edit profile Generalization : -	
Preconditions: • User is logged into the platform • User has a valid session	
Postconditions: • User can view their profile information • System displays all relevant profile sections	
Normal Flow of Events: 1. User clicks on Profile icon/button 2. System retrieves user profile information from database 3. System displays profile page with user information 4. System shows available profile management options	
Sub Flows: Not applicable	
Alternate Flow: Not applicable	

Delete Account

Use Case Name: Delete Account	**ID:** 4	**Importance Level:** Low
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to remove their account and data		
Brief Description: This use case describes how a user can permanently delete their account and all associated data from the system.		
Trigger: User wants to delete their account **Type:** External		
Relationship: Association : User Include : - Extend : - Generalization : -		
Preconditions:		

• User is logged into the platform • User has a valid session
Postconditions: • User account and associated data are permanently deleted • User session is terminated • User can no longer access the system with deleted credentials
Normal Flow of Events: 1. User navigates to account settings 2. User selects "Delete Account" option 3. System displays confirmation dialog with warnings about data loss 4. User enters password to confirm deletion 5. System validates password 6. User confirms final deletion warning 7. System remove all related property listing and user data 8. System terminates user session
Sub Flows: Not applicable
Alternate Flow: 5a. Invalid password: 1. System displays error message 2. System prompts user to retry password entry

Search Property

Use Case Name: Search Property	**ID:** 5	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to find suitable rental properties		
Brief Description: This use case describes how users can search and filter rental properties based on their preferences and requirements.		
Trigger: User wants to find rental properties **Type:** External		
Relationship: Association : User Include : - Extend : View property details Generalization : -		
Preconditions: • User has access to the platform (login optional)		
Postconditions: • System displays relevant search results • User can view property listings matching their criteria		

• Search history is recorded (if user is logged in)
Normal Flow of Events: 1. User accesses search interface 2. User enters search criteria 3. System processes search request 4. System queries database for matching properties 5. System displays search results in grid/list view 6. System shows total number of matching properties 7. System provides pagination if results exceed display limit
Sub Flows: Not applicable
Alternate Flow: 3a. No search criteria entered: 1. System displays all available properties 2. System applies default sorting 4a. No matching properties: 1. System displays "No properties found" message

View Property Details

Use Case Name: View Property Details	**ID:** 6	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to view detailed information about a specific property		
Brief Description: This use case describes how users can view detailed information about a specific rental property.		
Trigger: User clicks on a property listing **Type:** External		
Relationship: Association : User Include : - Extend : Report listing, Add property into wishlist Generalization : -		
Preconditions: • Property listing exists in the system • User has access to the platform (login optional)		
Postconditions: • User views complete property information • System records property view • Property view count is updated		
Normal Flow of Events: 1. User selects a property listing 2. System retrieves property details from database		

3. System displays comprehensive property information: 4. System records view history (if login) 5. System updates property view counter 6. System shows similar properties recommendations (if applicable) 7. System displays available actions (contact landlord, save to wishlist, report)
Sub Flows: Not applicable
Alternate Flow: Not applicable

Report Listing

Use Case Name: Report Listing	**ID:** 7	**Importance Level:** Low
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to report inappropriate or suspicious property listings		
Brief Description: This use case describes how users can report problematic property listings for review by system administrators.		
Trigger: User identifies a suspicious or inappropriate property listing **Type:** External		
Relationship: Association : User Include : - Extend : View property details Generalization : -		
Preconditions: • User is logged into the system • User is viewing a property listing		
Postconditions: • Report is submitted and stored in system		
Normal Flow of Events: 1. User clicks "Report Listing" button on property details page 2. System displays report form with several categories 3. User selects report category 4. User provides detailed description of issue 5. User can upload supporting evidence (optional) 6. User submits report 7. System stores report in database		
Sub Flows: Not applicable		
Alternate Flow: Not applicable		

Add Property into Wishlist

Use Case Name: Add Property into Wishlist	**ID:** 8	**Importance Level:** Medium
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to save and track interesting properties		
Brief Description: This use case describes how users can save properties of interest to their wishlist.		
Trigger: User wants to save a property to their wishlist **Type:** External		
Relationship: Association : User Include : - Extend : View property details, View wishlist items Generalization : -		
Preconditions: • User is logged into the system • User is viewing a property listing		
Postconditions: • Property is added to user's wishlist		
Normal Flow of Events: 1. User clicks "Add to Wishlist" button/icon on property listing 2. System adds property to user's wishlist database 3. System updates wishlist counter 4. System changes wishlist icon to filled state 5. System displays confirmation message		
Sub Flows: 1. Remove from Wishlist: a. User clicks filled wishlist icon b. System removes property from wishlist c. System displays removal confirmation		
Alternate Flow: Not applicable		

View Wishlist Items

Use Case Name: View Wishlist Items	**ID:** 9	**Importance Level:** Medium
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to view and manage saved properties		
Brief Description: This use case describes how users can view and manage their saved properties in the wishlist.		
Trigger: User wants to access their saved properties **Type:** External		

Relationship: Association : User Include : - Extend : Remove wishlist item, Add property to wishlist Generalization : -
Preconditions: • User is logged into the system
Postconditions: • User can view all saved properties • User can manage wishlist items
Normal Flow of Events: 1. User navigates to Wishlist section 2. System retrieves user's wishlist data 3. System displays wishlist properties in grid/list view 4. System displays total number of saved properties
Sub Flows: Not applicable
Alternate Flow: 2a. Empty wishlist: 1. System displays "No saved properties" message 2. System suggests popular properties

Remove Wishlist Item

Use Case Name: Remove Wishlist Item	**ID:** 10	**Importance Level:** Medium
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to remove properties from their wishlist		
Brief Description: This use case describes how users can remove properties from their wishlist.		
Trigger: User wants to remove a property from wishlist **Type:** External		
Relationship: Association : User Include : - Extend : View wishlist items Generalization : -		
Preconditions: • User is logged into the system		
Postconditions: • Property is removed from wishlist • System records are updated accordingly		
Normal Flow of Events:		

1. User clicks "Remove from Wishlist" button/icon 2. System displays confirmation dialog 3. User confirms removal action 4. System removes property from user's wishlist 5. System changes wishlist icon to unfilled state (if on property page) 6. System displays removal confirmation message
Sub Flows: Not applicable
Alternate Flow:

Apply for Landlord Status

Use Case Name: Apply for Landlord Status	**ID:** 11	**Importance Level:** Medium
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to become a verified landlord to list properties		
Brief Description: This use case describes how users can apply to become verified landlords on the platform.		
Trigger: User wants to apply account to landlord status **Type:** External		
Relationship: Association : User Include : - Extend : Track application status Generalization : -		
Preconditions: • User is logged into the system		
Postconditions: • Landlord application is submitted • User can track application status		
Normal Flow of Events: 1. User navigates to "Become a Landlord" section 2. User fills required information 3. User uploads required documents 4. User submits application 5. System stores application and documents securely 6. System confirms application submission to user 7. System provides estimated processing timeframe		
Sub Flows: Not applicable		
Alternate Flow: 2a. Incomplete information: 1. System highlights missing fields		

Track Application Status

Use Case Name: Track Application Status	**ID:** 12	**Importance Level:** Medium
Primary Actor: User	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Users – wants to monitor landlord application progress		
Brief Description: This use case describes how users can track the status of their landlord application.		
Trigger: User wants to check landlord application status **Type:** External		
Relationship: Association : User Include : - Extend : Apply for landlord status Generalization : -		
Preconditions: • User is logged into the system • User has submitted a landlord application		
Postconditions: • User views current application status • User can access application history		
Normal Flow of Events: 1. User navigates to "Application Status" section 2. System retrieves application details using application ID 3. System displays comprehensive status information		
Sub Flows: Not applicable		
Alternate Flow: Not applicable		

Create Listing

Use Case Name: Create Listing	**ID:** 13	**Importance Level:** High
Primary Actor: Landlord	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Landlord – wants to list their property for rent		
Brief Description: This use case describes how verified landlords can create new property listings on the platform.		
Trigger: Landlord wants to create a new property listing **Type:** External		
Relationship: Association : Landlord Include : Submit verification documents Extend : Edit listing details, Delete listing		

Generalization : -
Preconditions: • Landlord is logged into the system • Landlord has verified status
Postconditions: • Wait admin for approved • System updates property database
Normal Flow of Events: 1. Landlord selects "Create New Listing" option 2. System displays create listing form 3. Landlord fills required information 4. Landlord submits listing form 5. System processes listing creation and store listing information 6. System displays successful creation message
Sub Flows: Not applicable
Alternate Flow: Not applicable

Edit Listing Details

Use Case Name: Edit Listing Details	**ID:** 14	**Importance Level:** High
Primary Actor: Landlord	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Landlord – wants to update property listing information		
Brief Description: This use case describes how landlords can modify existing property listing information.		
Trigger: Landlord wants to update property listing details **Type:** External		
Relationship: Association : Landlord Include : - Extend : Create listing Generalization : -		
Preconditions: • Landlord is logged into the system • Listing exists in the system • Landlord owns the listing		
Postconditions: • Listing information is updated • Users see updated information		
Normal Flow of Events: 1. Landlord selects listing to edit		

2. System displays editable listing form with current information 3. Landlord makes desired changes 4. Landlord submits changes 5. System verifies all changes 6. System updates listing database 7. System notifies landlord of successful update
Sub Flows: Not applicable
Alternate Flow: Not applicable

Delete Listing

Use Case Name: Delete Listing	**ID:** 15	**Importance Level:** High
Primary Actor: Landlord	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Landlord – wants to remove property listing		
Brief Description: This use case describes how landlords can remove their property listings from the platform.		
Trigger: Landlord wants to delete a property listing **Type:** External		
Relationship: Association : Landlord Include : - Extend : Create listing Generalization : -		
Preconditions: • Landlord is logged into the system • Listing exists in the system		
Postconditions: • Listing is removed from active listings		
Normal Flow of Events: 1. Landlord selects "Delete Listing" option 2. System displays confirmation dialog 3. Landlord confirms deletion 4. System processes deletion 5. System confirms deletion to landlord		
Sub Flows: Not applicable		
Alternate Flow: Not applicable		

Verify Landlord Application

Use Case Name: Verify Landlord Application	**ID:** 16	**Importance Level:** Medium

Primary Actor: Admin	**Use Case Type:** Details, Essential
Stakeholders and Interest: Admin – needs to verify legitimacy of landlord applications	
Brief Description: This use case describes how administrators review and verify applications from users wanting to become landlords on the platform.	
Trigger: New landlord application is submitted **Type:** External	
Relationship: Association : Admin Include : - Extend : - Generalization : -	
Preconditions: • Admin is logged into the system • There are pending landlord applications to review	
Postconditions: • Application status is updated (approved/rejected) • Landlord is notified of the decision • System records are updated accordingly	
Normal Flow of Events: 1. Admin accesses landlord application management section 2. System displays list of pending applications 3. Admin selects an application to review 4. Admin reviews submitted documents and information 5. Admin verifies authenticity of submitted documents 6. Admin makes decision (approve/reject) 7. System updates application status 8. System sends notification (email) to applicant	
Sub Flows: Not applicable	
Alternate Flow: Not applicable	

Manage User Role

Use Case Name: Manage User Roles	**ID: 17**	**Importance Level:** Medium
Primary Actor: Admin	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Admin – needs to control who has landlord access		
Brief Description: This use case describes how administrators manage user roles by assigning or revoking landlord status for users on the platform.		
Trigger: Landlord application is approved/rejected **Type:** External		
Relationship:		

Association : Admin Include : - Extend : - Generalization : -
Preconditions: • Admin is logged into the system • User account exists in the system • Landlord verification process is completed (if granting landlord status)
Postconditions: • User role is updated (normal user or landlord) • User is notified of role change
Normal Flow of Events: 1. Admin accesses user management section 2. System displays list of users with current roles 3. Admin selects user to modify 4. System shows current role (normal user/landlord) 5. Admin changes role between normal user and landlord 6. System prompts for confirmation 7. Admin confirms change 8. System updates user role
Sub Flows: Not applicable
Alternate Flow: 5a. If revoking landlord status: 1. System removes all relevant listing 2. User reverts to normal user status

View User Reports

Use Case Name: View User Reports	**ID: 18**	**Importance Level:** Medium
Primary Actor: Admin	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Admin – needs to resolve user complaints		
Brief Description: This use case describes how administrators review and handle user reports regarding suspicious listings.		
Trigger: User submits a report about inappropriate listing **Type:** External		
Relationship: Association : Admin Include : - Extend : Remove inappropriate listing Generalization : -		
Preconditions:		

• Admin is logged into the system • Reports exist in the system to be reviewed
Postconditions: • Reports are reviewed and marked with appropriate status • Action taken on valid reports (listing removed)
Normal Flow of Events: 1. Admin accesses report management section 2. System displays list of user reports sorted by priority/date/category 3. Admin selects report to review 4. System shows detailed report information 5. Admin determines appropriate action
Sub Flows: Not applicable
Alternate Flow: Not applicable

Remove Inappropriate

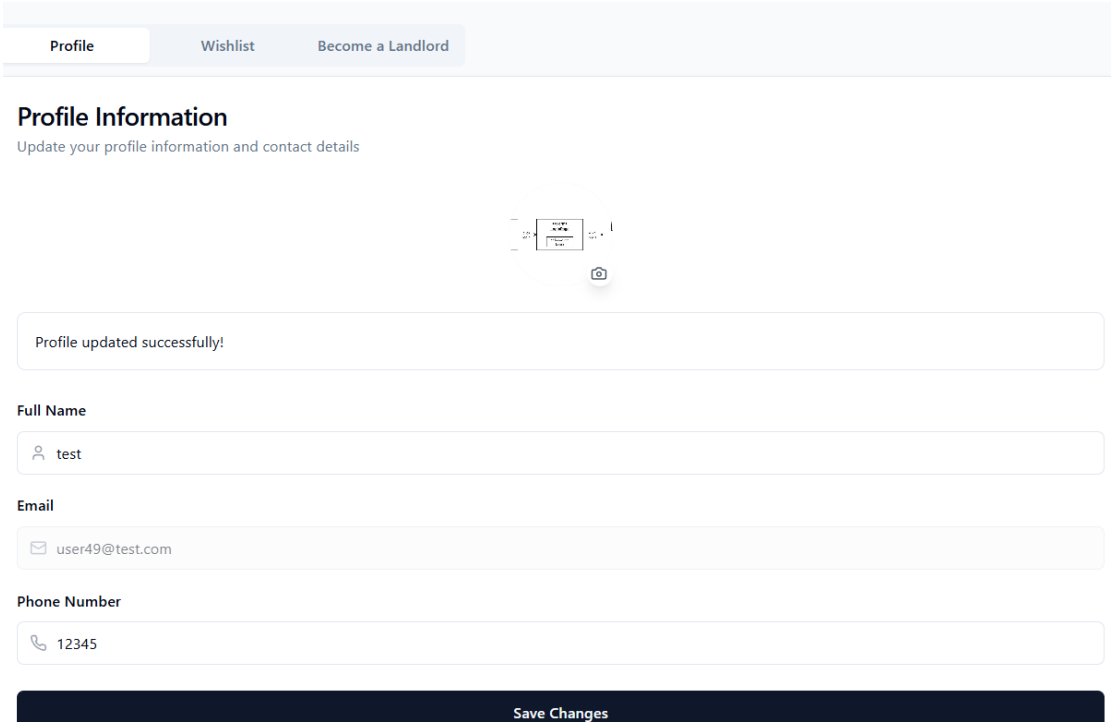
Use Case Name: Remove Inappropriate Listing	**ID: 19**	**Importance Level:** Medium
Primary Actor: Admin	**Use Case Type:** Details, Essential	
Stakeholders and Interest: Admin – maintain platform content standards Users - expect safe and appropriate content		
Brief Description: This use case describes how administrators remove content that violates platform guidelines.		
Trigger: User reports inappropriate content **Type:** External		
Relationship: Association : Admin Include : - Extend : View user reports Generalization : -		
Preconditions: • Admin is logged into the system • Inappropriate content has been reported		
Postconditions: • Inappropriate content is removed • Content creator is notified		
Normal Flow of Events: 1. Admin accesses reported listing 2. System displays violation type 3. Admin reviews content against guidelines 4. Admin select remove listing 5. System prompts for confirmation		

6. Admin confirms removal 7. System removes listing
Sub Flows: Not applicable
Alternate Flow: Not applicable

Appendix B: Test Case Result

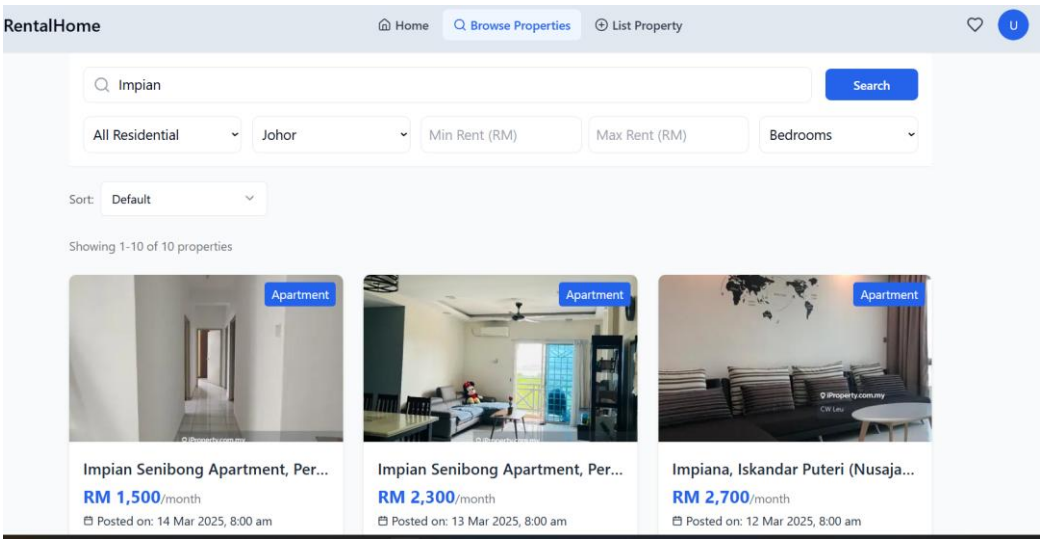
TC03 Profile Management

Update new profile picture, full name, phone number.

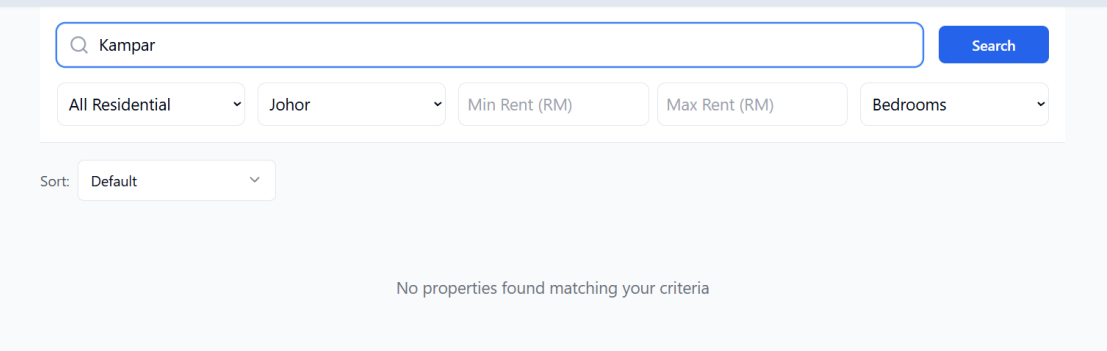


TC04 Property Search

Select “Johor” as state, enter “Impian” as search query. The system displays 10 matched properties.

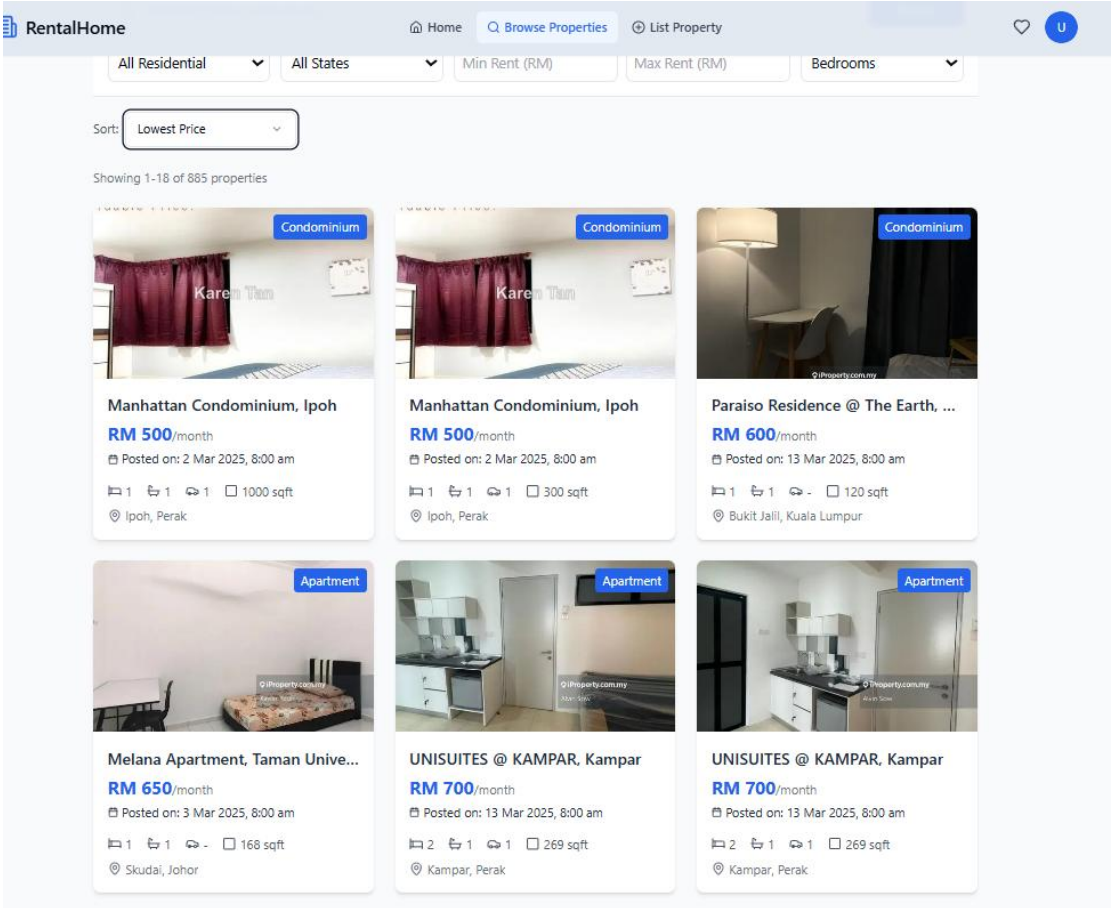


Select “Johor” as state, enter “Kampar” as search query. The system displays no properties found.



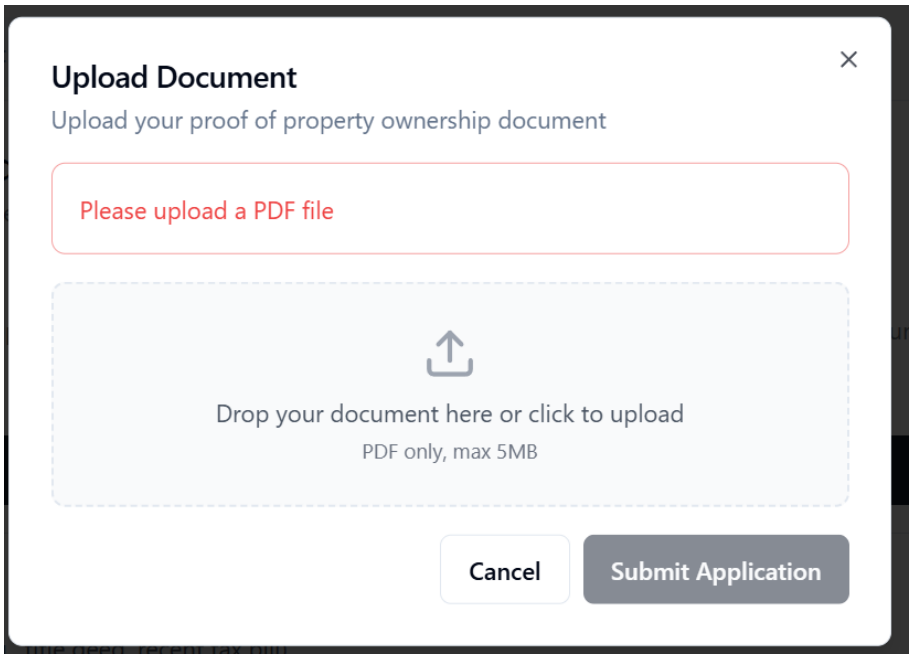
TC05 Property Sorting

Sort properties by price

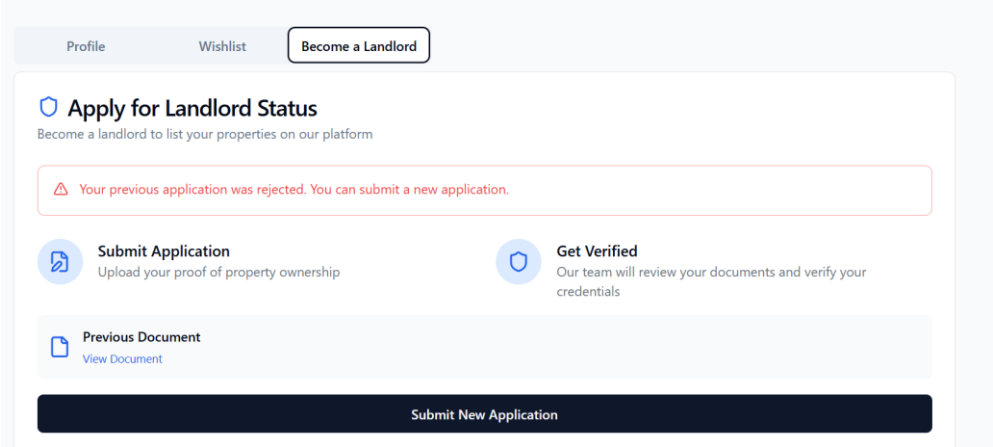


TC07 Landlord application

Upload invalid file format

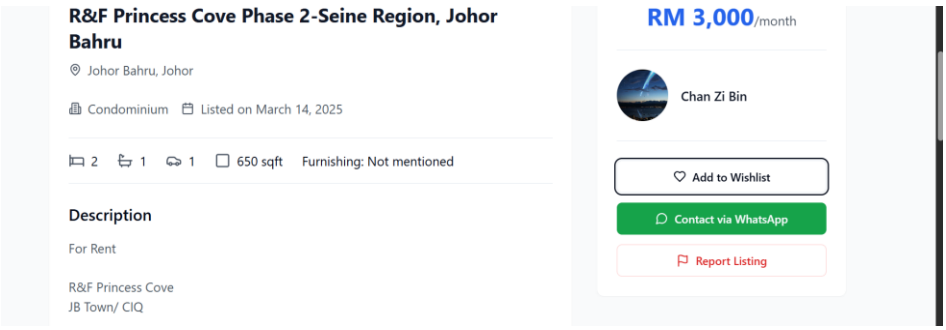


Rejected application

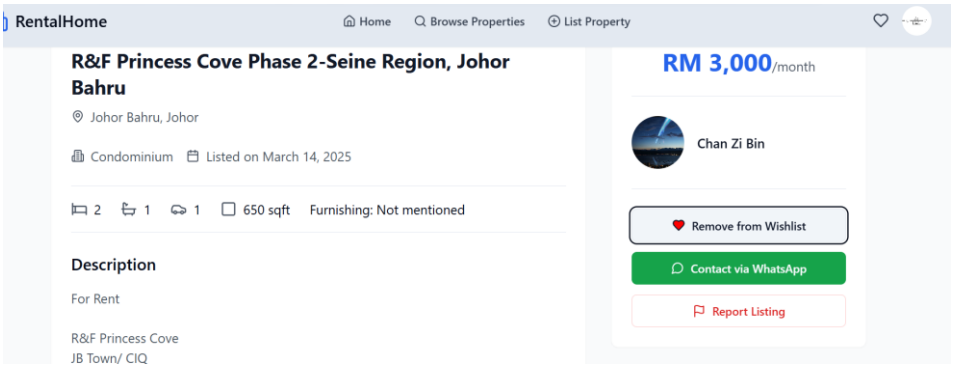


TC08 Property Wishlist

Add to wishlist

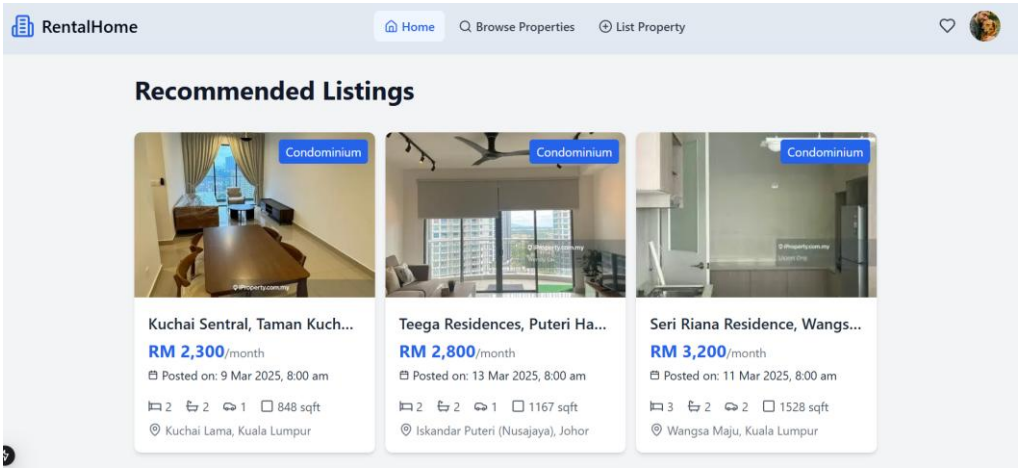


Remove from wishlist

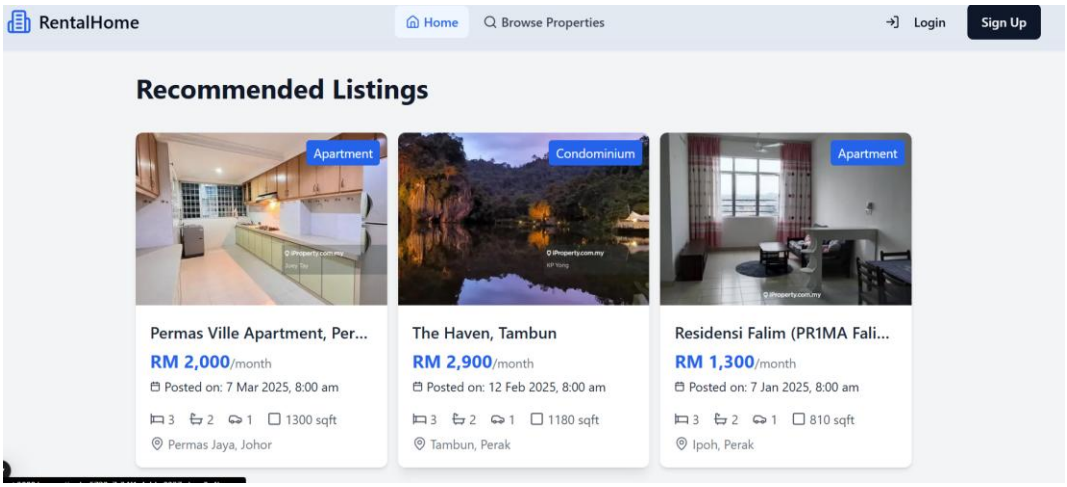


TC09 Landing Page Recommendation

Logged-in users get personalized recommendations from the system

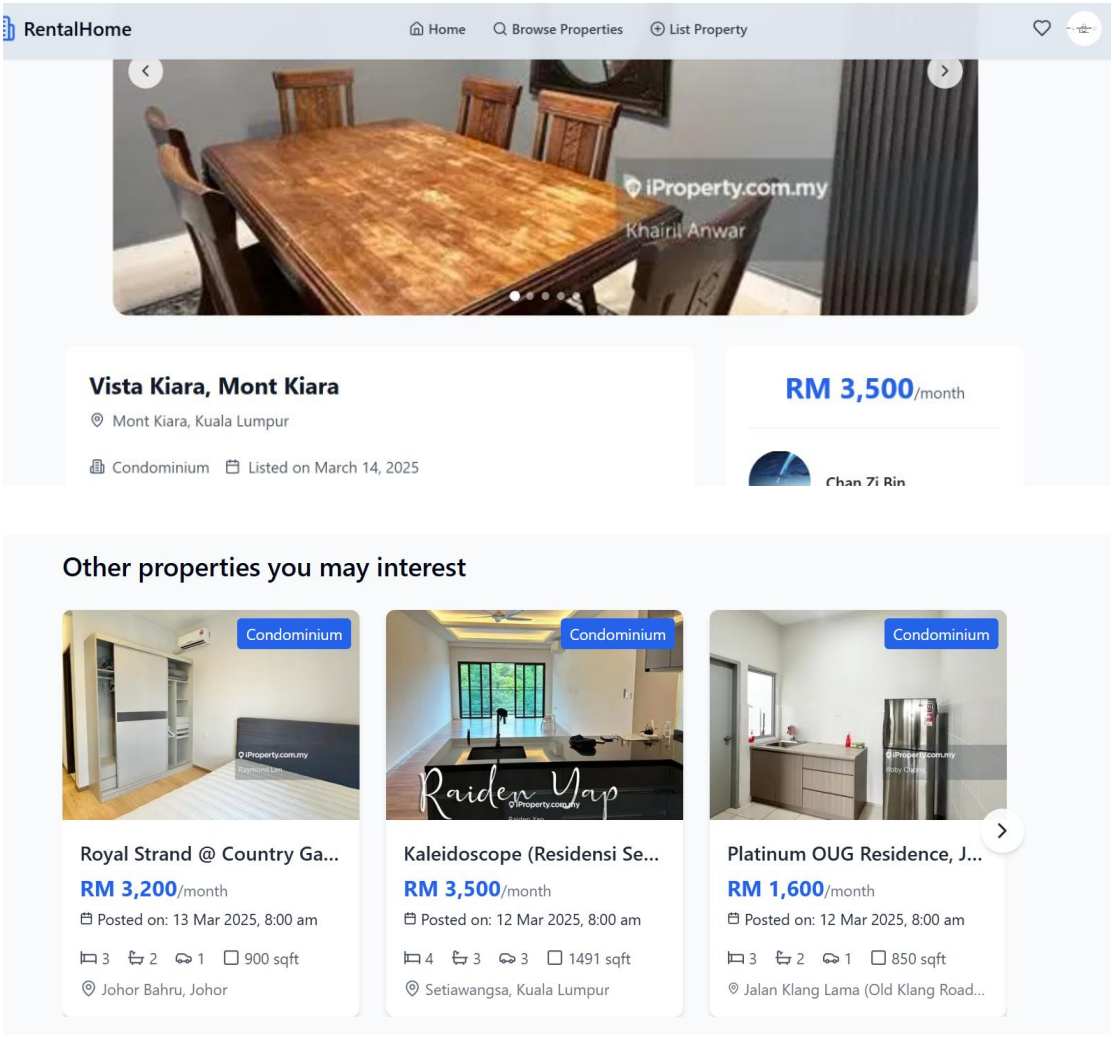


Non-logged in users or new user with no view history get random popular recommended properties



TC10 Property Details Recommendations

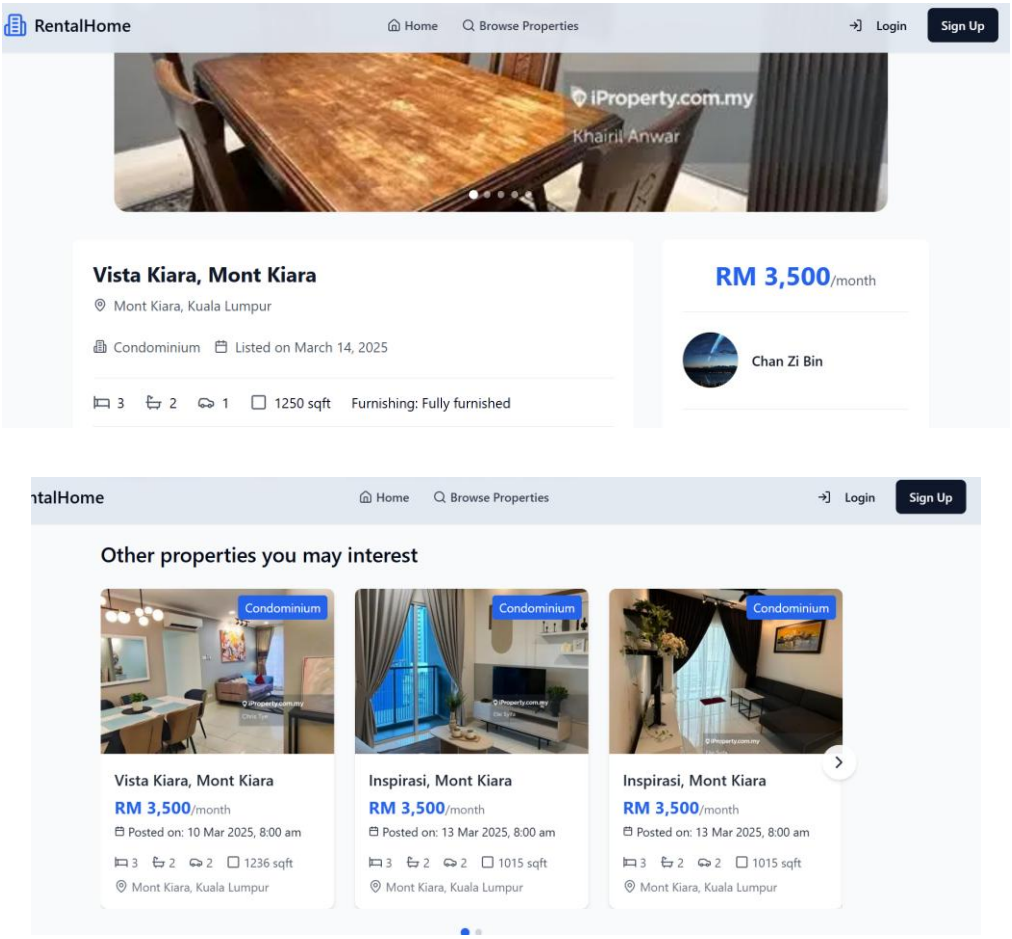
Logged-in user visit "Vista Kiara, Mont Kiara" property page. Scrolling to the recommendation section.



Passing user_id and property_id to the deployed API server, the server returns a list of recommended properties based on based on user wishlist items and view history as well as the visited property's features (hybrid recommendation approach).

```
Apr 26 03:26:52 PM  kqfc5  INFO:     14.192.245.245:0 - "GET /hybrid-recommendations/?property_id=42aaf53c-2d4b-42bd-8395-6a6a6bcce831&user_
id=373a5ef7-9540-48d9-9ce5-f5215feb40ef&top_n=6&alpha=0.6&beta=0.4 HTTP/1.1" 200 OK
```

Non-logged-in users visit the same page. Scrolling to the recommendation section.

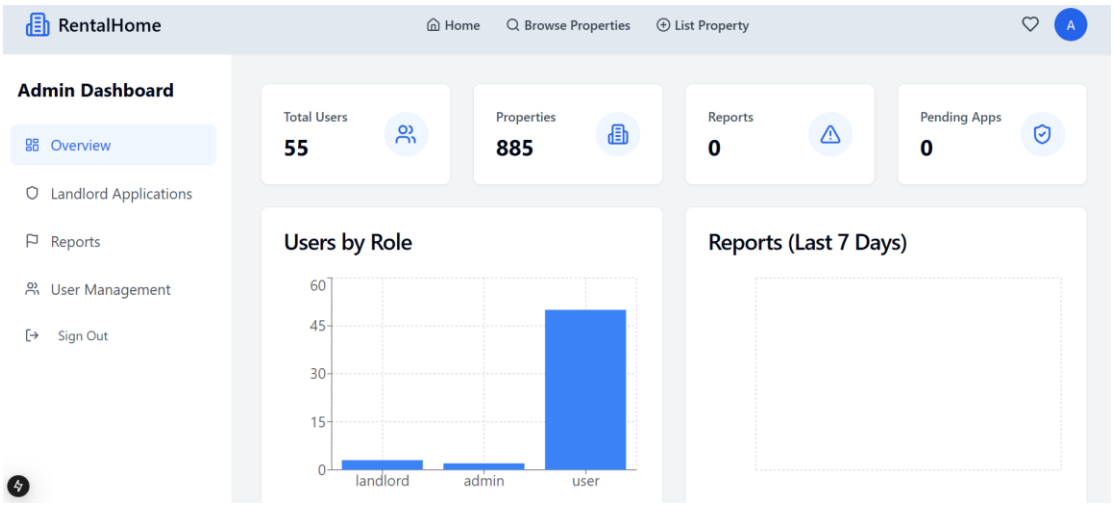


The system recommends properties only based on the visited property's features (content-based filtering).

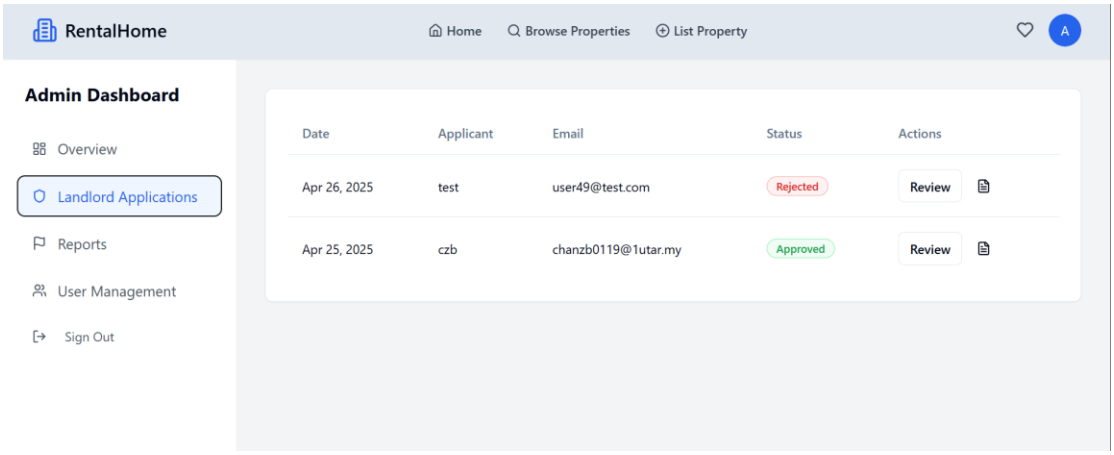
```
Apr 26 03:08:25 PM  4ccw8  INFO:     14.192.245.245:0 - "GET /similar-properties/?property_id=f4a2c33d-fb5e-40ea-821c-14b3f378c289&top_n=6 H
TTP/1.1" 200 OK
```

TC11 Admin Dashboard

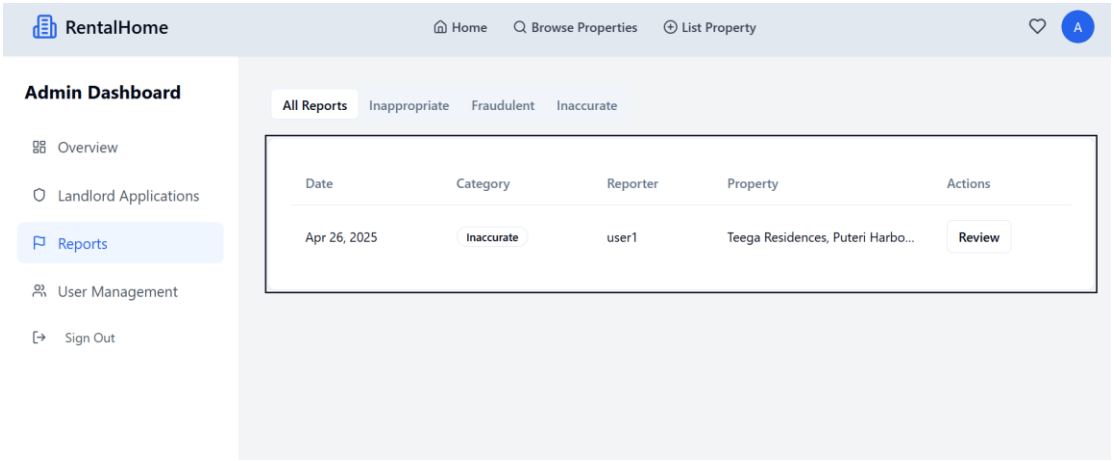
Overview tab



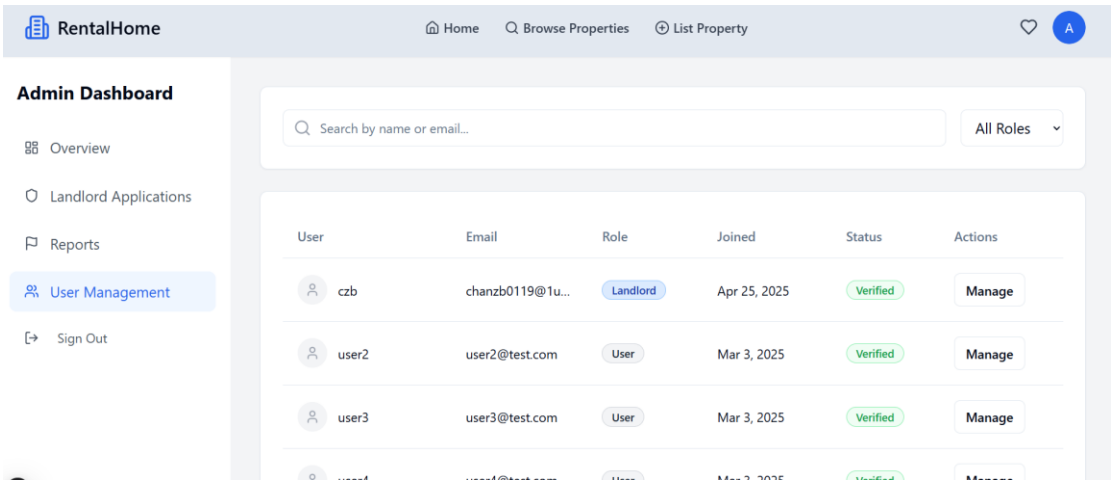
Landlord application tab



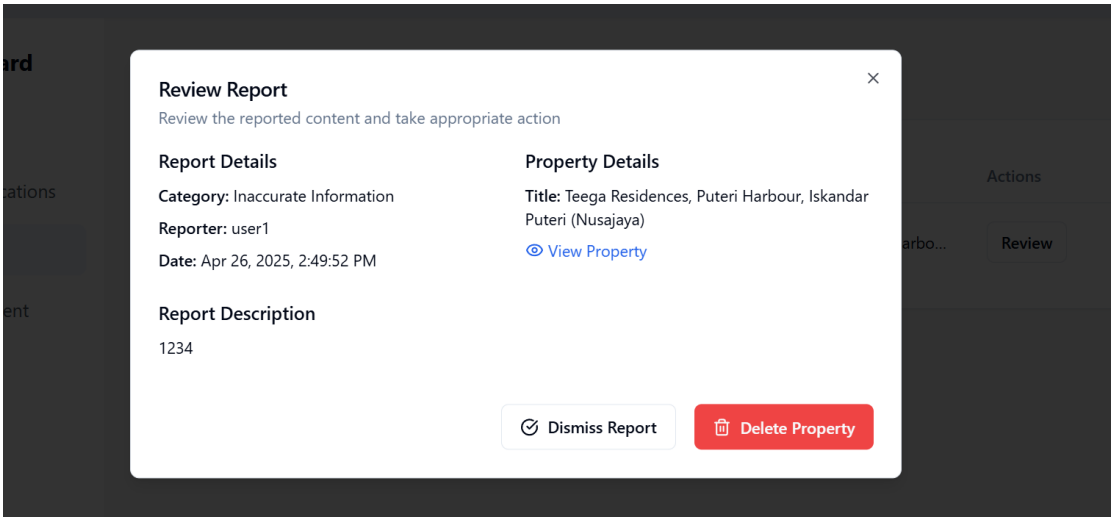
Reports



User management



TC013 Property Report Handling



Appendix C: Poster