AI IN REAL ESTATE RECOMMENDATION SYSTEMS

EstateHub: An Integrated Real Estate and Accommodation Management Platform

Mayureshwar Mane, Atharva Gaikwad, Yashraj Kondekar, Pareetosh Chougule, S. S. Chougule

Department of Computer Science and Engineering, Sanjay Ghodawat Institute, Kolhapur, Maharashtra, India

Published November 2026

Abstract

Many different platforms are needed to search for properties (buy), rental accommodation (rent), pay per guest (PG), hostels, short term accommodation and roommates. This leads to issues with comparing, communication, tracking and management. This paper outlines EstateHub which allows users to search for property in a single web application. It has been developed in Python using the Flask framework and a relational database with MySQL, however it can also use a local database (SQLite). The application will allow users to search for property, book properties and ask questions. It will also allow users to search for pay per guest/hostel accommodations, resorts/lodges/villas. Users will be able to match with roommates. They will also be able to shortlist properties for families and receive notifications. Different dashboards will be available depending on the user role. An AI Advisor is also included. It will reduce the need to switch between multiple disjointed applications to assist users by allowing specific features based on their role as a buyer, renter, owner, agent, accommodation host, and administrator. This paper outlines the issue, aims, architecture, key modules and development approach of the application. However no controlled usability test or quantitative results were provided so no claims can be made to increase speed, accuracy or user satisfaction. The application design has been outlined.

Keywords: accommodation management; Flask; property marketplace; roommate matching; web application

1 Introduction

There are many different types of accommodation and property searches. This may include finding residential properties, pay per guest, hostels, resorts, lodges, villas and more. Users may wish to compare properties, contact providers, save options, book visits, manage bookings and co-ordinate with family. This can become quite a tedious task when multiple platforms or manual approaches are taken.

EstateHub proposes a single platform to manage all of the above. This will allow users to search for property, however it is designed to keep the specific details of each type of accommodation. It will also allow users to be identified by their role. Buyers/renters can search for options, owners/agents can manage property listings and questions, and hosts can manage their accommodation.

The main contribution this work makes is designing and developing a single role-based application that allows users to search for property and temporary accommodation with shared living. The application also contains extra features like Family Hub to allow users to shortlist properties with their family and receive notifications. The AI Advisor provides users with a specific interface to receive assistance from AI. This will provide users with the capabilities of the application however the real world benefits need to be formally tested.

1.1 Problem statement

When searching for property/accommodation users may need to switch between multiple platforms and manual approaches. This can cause issues with keeping a consistent view of different types of properties. EstateHub allows users to search for multiple property/accommodation options within a single rolebased web application.

1.2 Aims

- Provide a single platform to buy, rent and search property listings.
- Allow users to search for multiple types of accommodation (pay per guest, hostels, resorts, lodges, villas).
- Allow users to match with roommates.
- Allow users to ask questions, book visits, save favourite options, receive notifications and view dashboards based on their role.
- Provide the ability to shortlist properties for families and receive notifications.
- Provide an AI Advisor interface whilst keeping the AI separate from verified legal, financial and property information.

1.3 Scope and contributions

This will be a web-based prototype. The paper will outline the architecture, organisation of features, data management and a repeatable test plan. It cannot be claimed EstateHub replaces property registration, legal checks, identity verification, payments and professional advice. These all require their own integrations, controls and validation.

2 Related work and motivation

Typically digital property marketplaces will provide users with the ability to list properties, search with filters and find properties based on location. Accommodation providers will provide availability, detail about their accommodation and booking options. Shared living platforms will provide users with the ability to find roommates based on their preferences. Overall there is overlap with the need for structured property profiles, search options, trusted information and tracking of transactions.

The motivation for EstateHub is to provide users with a single account and role-based application to manage multiple property/accommodation options. The unique aspect of this design is that multiple property/accommodation options can be found within a single application. Users can search for long term property options, temporary accommodation and shared living. Owners, agents, hosts can see the relevant management options. Family shortlisting and notifications have been added to allow users to coordinate with their family.

A full academic comparison should be made by researching some of the existing platforms to see how they compare based on their published documentation. As there was no verified competitor research provided within the project materials, this draft will not make any specific claims about the shortcomings of specific commercial platforms. Recent peer reviewed work on online real estate information systems, recommender systems, shared living matching, usability and secure web application design should be added.

3 System requirements and methodology

3.1 Development methodology

An iterative approach has been taken to develop the web application. This has involved identifying the user roles and use cases, organising the data relating to listings and transactions, developing the routes and templates on the server side. Finally the forms have been connected to the persistence methods and each module has been reviewed using functional tests. The iterative approach has been important as there are common fields across the different listing types however specific attributes are required.

3.2 Technologies used

- Language: Python
- Framework: Flask
- Database: MySQL 8 will be used as the primary relational database, however a local database (SQLite) can also be used.
- UI: HTML, CSS and JavaScript (Exact libraries and versions to be used in the final code.)
- Tools: Antigravity and Google AI Studio were used.
- Exact versions and hosting details should be included in the final paper. Whilst it is convenient to have the local database, it needs to be noted in the logs and in the configuration. This will ensure users do not get a different data set if they unexpectedly lose their MySQL connection.

3.3 Functional requirements

- User authentication and role based access.
- Create, view, update and manage property/accommodation listings.
- Search listings.
- Ask questions, visit properties and book.
- Save favourites/family shortlist and receive notifications.
- Create roommate profiles and match based on available profiles.
- Management options for owners, agents, hosts and administrators.
- AI Advisor interface.

3.4 Non-functional requirements

- Secure: Passwords are hashed, secure sessions, role based authorisation, inputs are validated and secrets are kept secure.
- Reliable: Consistent database transactions, clear error handling, back ups and predictable database choice.
- Usable: Consistent navigation, easy to use forms, clear status messages and responsive design.
- Maintainable: Modular routes, re-usable templates, documented database relationships and automated testing.
- Private: Only the required information will be requested for profiles. Personal, booking and family information will be restricted.

4 System architecture and modules

EstateHub can be seen as a layered application. The UI layer will provide the pages/forms. The Flask application will handle the routing, authentication, validation, authorisation and business logic. The data access layer will handle the database operations. The relational database will store users, listings, accommodation, bookings, questions, notifications, profiles and related information. External AI services can be accessed if they have been configured.

4.1 Property search and listings

The property module provides users with the ability to search for property within the marketplace. The information will be stored and shown to users who wish to buy/rent. Owners/agents with permission can manage their listing. Structured information can be used to compare options. Server side checks should be used to ensure only the correct users can access the listings.

4.2 Accommodation and stays

EstateHub has expanded the listing functionality to include pay per guest, hostels, resorts, lodges and villas. This may require more detailed information such as availability of rooms/units, how many people are staying, what facilities are available, how long they are staying and information about the host. The data model should contain a property/establishment and individual units.

4.3 Roommate matching

The roommate module provides users with the ability to create profiles to find roommates. The matching can be based on the preferences they provide such as budget, location, lifestyle and move in date. Any scoring should be explained as this is just a recommendation. Users should have control over what information they provide within their profile.

4.4 Questions, visits and bookings

Users can ask questions and visit properties. This gives them the ability to log their interest. Bookings can be made to link them to the relevant property/accommodation. This can also provide a history of their status. Validation should be implemented to ensure they are available. If the business logic says they can only be used by one person, there should be no conflicting bookings. No live payments or legally binding reservations have been assumed.

4.5 Family hub, favourites and notifications

Family Hub allows users to work together to evaluate their property options. This can be done through shortlists and notifications. In this scope notifications have been added however no chat functionality is required. Controls should be in place to ensure only authorised users can see the shortlisted information.

4.6 Dashboard and administration

Different dashboards have been provided based on the user role. Administration provides oversight of the platform. Every sensitive action should be verified on the server. This includes modifying listings, managing users and accessing private information.

4.7 AI advisor

Provides users with an interface to ask questions using AI. The responses should be given as information. If they have not been implemented and verified then they should not be presented as verified legal title checks, document verification, guaranteed fraud detection or professional financial advice. The generated content should be clearly separated from the data within the platform. User documents and personal information should not be sent to external services without the correct consents in place.

5 Database design

The relational data model has been organised into different sections. The development database has several entities including users, properties, property images, amenities, property to amenity relationships, accommodation listings, accommodation units, bookings, questions, question messages, notifications, favourites, family groups, group members, group shortlists, roommates, agent profiles, host profiles, property visits, reviews, audit logs and workflow information. The final deployed database should be checked to ensure the correct tables are included as there may be differences between the local and MySQL database.

Where possible primary and foreign keys should be used. Listings should be linked to the correct user/organisation. Bookings should be linked to the correct user and property/accommodation unit. Family groups should be clearly defined. Notifications should be linked to the correct user and event. Constraints, indexes on searched fields and transactional updates will aid consistency.

A key consideration when implementing the database is ensuring it is correctly configured. If there is an issue connecting to MySQL and the application automatically switches to SQLite then users may see a different data set. The application should tell the user what database has been chosen. It should also be clear that the local database is being used during development. It should fail safely if it cannot connect to the required database.

6 Security, privacy and responsible use

As this platform may hold personal information, contact details, family shortlists, questions and bookings it is important to ensure security and privacy are considered as key requirements. Passwords should be hashed using a suitable password hashing function. Secrets should be provided using environment variables or secret manager. Database accounts should follow the principle of least privilege. All inputs should be validated on the server. Queries to the database should use parameterised queries or a safe ORM.

All secured routes and objects should have authorisation. Users should not be able to view/edit another user's private information by changing the ID in the URL/form.

HTTPS should be used when deploying to production. Secure cookie settings should be used for sessions. CSRF protections should be in place for forms that change state. Rate limiting should be in place for sensitive endpoints. Logs should not include unnecessary personal information. There should be a documented back up and recovery plan.

Recommendations from the AI may be incomplete or inaccurate. The interface should explain the limitations. No claims should be made about legal status or safety of a property. Users should have a

way to verify important information. If documents are uploaded then the retention, access, processing and deletion should be specified.

7 Testing plan and evaluation plan

A realistic testing plan should be provided. The below table shows the proposed testing plan. No claims can be made that these have been passed.

Table 1. Proposed testing plan.

Test area	Example test	Evidence to record
Authentication	Valid and invalid sign-in; session expiry	Pass/fail, error handling, session behavior
Authorization	Attempt crossuser and crossrole access	Unauthorized access blocked
Listings	Create, edit, view, and remove permitted listings	Record consistency and correct permissions
Search	Filter listings using supported fields	Correct results and response time
Bookings	Create booking and test conflicting availability	Correct status and conflict prevention
Notifications	Trigger an event and verify recipient	Delivery, ownership, and read state
Family Hub	Share shortlist and test groupmember boundaries	Only authorized members can access it
Roommate profiles	Create/update profile and test matching inputs	Correct filtering and privacy controls
Database reliability	Run with MySQL and separately test SQLite fallback	Startup database identity and persistence
Security	Test validation, CSRF controls, and common web risks	Documented findings and remediation
Usability	Ask representative users to complete common tasks	Completion rate, time, errors, user feedback

7.1 Metrics

It is recommended to have quantitative metrics such as completion rate, median time to complete tasks, search time, booking success rate, authorisation pass rate and number and severity of defects. Recommendations for roommates should ensure the returned matches meet the specific constraints. No claims should be made that the algorithm scores prove compatibility. Usability testing should include sample size, selection, tasks, consent and analysis.

7.2 Results and findings

Based on the project context it can be described that a multi module Flask application has been created along with a relational data model. As no verified testing has been provided there are no numerical results to claim. The main design outcome of the system has been to provide users with a single application that can search for property, multiple types of accommodation, shared living, role based management and family shortlists/notifications. This should be verified against the final code. Screenshots/test evidence should be provided to show this.

8 Discussion

The main benefit of the design has been to provide users with a single workflow. This can reduce the need to switch between different types of accommodation as everything is in one place. However, having a wide range of features has caused some challenges. The listing types need to have a consistent yet flexible data model. Authorisation needs to be consistent across modules. Notifications need to be reliable. The database fallback shows how important it is to have visibility of how the application is operating.

The real value of the platform cannot be determined purely by the number of features. A comparison against existing services should be made to ensure it provides users with the ability to complete tasks easily, has good quality information, security/safety measures in place, is accessible and is reliable. A small user study and repeatable functional tests would help determine if the single workflow provides users with a better experience.

9 Limitations and future work

This work has outlined the implementation of a project and proposed testing. No controlled experiment has been performed. The features and production status should be verified against the final application. No claims can be made that there has been an improvement in user satisfaction, search efficiency, matching quality, security or business outcomes as this has not been independently measured. The application may also depend on the database configuration and external AI services.

- Perform a structured comparison against selected real estate/accommodation platforms using current evidence.
- Automated unit, integration, end to end, authorisation and database migration tests.
- Usability testing with buyers, renters, owners, agents, hosts and users looking for shared accommodation.
- Transparent verification processes for listings. Clear distinction between user provided and verified information.
- Improved search and roommate recommendations using explainable criteria.
- Privacy aware evaluation.
- Improved visibility, back up, database migration and deployment controls.
- Ensure the application is accessible, mobile friendly and performs well.
- Transparent data retention and consent for AI related features and user documents.

10 Conclusion

This paper has outlined EstateHub which is a web based application that allows users to search for property, pay per guest/hostel, resorts, lodges, villas, roommates, book and ask questions, dashboards based on their role, family shortlists, receive notifications and an AI Advisor. It uses a flask based application and relational database to provide structured data. The main contribution has been the ability to provide users with a single application to manage multiple related housing/accommodation options. Future testing should be performed to ensure the final features are provided, authorisation and data consistency, performance and usability with representative users. This will allow claims to be made about the effectiveness of the application.

Acknowledgments

The authors thank their guide, S. S. Chougule, and the Department of Computer Science and Engineering, Sanjay Ghodawat Institute, Kolhapur, for guidance and support throughout this work.

References

[1] Gharahighehi A, Pliakos K, Vens C. Recommender systems in the real estate market—a survey. *Applied Sciences*. 2021;11(16):7502. https://doi.org/10.3390/app11167502