



International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Smart House Rental Web Application with AI Chatbot: A Three-Tier Platform for Verified Listings, Intent-Based Property Recommendation and Secure Tenant–Owner Interaction

Jallela Praveen¹, Dr. V. Harsha Shastri²

¹PG Scholar, Dept. of Computer Applications, School of Informatics, Aurora Higher Education and Research Academy Deemed-to-be-University, Uppal, Hyderabad, Telangana, India

²Associate Professor, Dept. of Computer Applications, School of Informatics, Aurora Higher Education and Research Academy Deemed-to-be-University, Uppal, Hyderabad, Telangana, India

Emails: Praveen.jallela@aurora.edu.in¹, harshasastry@aurora.edu.in²

ABSTRACT

Finding a rental home remains slow and trust-poor. Tenants depend on manual enquiries and portals that offer little more than text search, while owners cannot easily prove that a listing is genuine. Most existing rental systems lack intelligent assistance, multi-criteria filtering, personalised recommendation, private messaging and visit scheduling. This paper presents the Smart House Rental Web Application with AI Chatbot, a full-stack platform built with React.js, Java Spring Boot and MySQL that brings tenants, property owners and an administrator onto one system. The design provides role-based access secured by JSON Web Tokens (JWT) and BCrypt hashing, owner document verification, advanced search by location, budget, BHK type and amenities, an intent-based chatbot that returns a conversational reply with ranked properties, private per-property messaging, visit scheduling and Google Maps location display. The system is specified through use case, class, sequence and activity diagrams, a nine-table normalised schema and data flow diagrams. The prototype currently implements the complete data model, the authentication and authorisation module, and a working React shell with registration, login and protected routing; the remaining modules are scaffolded. Performance indicators are therefore not reported, and the paper presents design and implementation-status results rather than experimental accuracy. The work demonstrates a practical, modular and extensible route to an integrated, AI-assisted rental platform.

Keywords: House Rental System, AI Chatbot, Property Recommendation, Natural Language Processing, Spring Boot, React, JWT Authentication, MySQL

1. INTRODUCTION

Rapid urbanisation has increased demand for accessible and trustworthy rental housing. Traditional renting relies on physical visits, manual enquiries and referrals, which are inadequate for technology-savvy tenants and owners. Web portals have partly addressed this by letting owners post listings and tenants browse them, but most remain passive directories with basic text search.

For tenants, the main difficulties are finding properties that match location, budget, BHK type and amenities, getting quick answers to routine questions, and trusting that a listing is genuine. Owners struggle to manage enquiries and coordinate visits, and administrators need a structured way to verify owners and moderate listings.

To address this problem, we developed the Smart House Rental Web Application with AI Chatbot — a centralised web platform on which tenants search and filter properties, converse with an AI chatbot, receive recommendations, message owners privately and schedule visits, while owners list verified properties and an administrator oversees the platform. The main features of the proposed system are:

- Role-based workflows for tenants, property owners and administrators
- Stateless JWT authentication with BCrypt password hashing
- Document-based owner verification managed by the administrator
- Advanced multi-criteria search (location, budget, BHK type, amenities)
- An intent-based AI chatbot returning a reply and a ranked property list
- Private, per-property tenant–owner messaging and visit scheduling

- Google Maps location display on the property details page

2. LITERATURE SURVEY

2.1 Voumick et al. (2021)

Voumick et al. proposed “Development of Online Based Smart House Renting Web Application.” The study built a rental portal for tenants and owners in Bangladesh using PHP, Laravel, MySQL, AJAX and the Google Maps API, with verified owner accounts, listings and a conversation box. However, it offers no AI chatbot, only basic text search and no recommendations, and its conversation box is public rather than private per listing. [1]

2.2 Ikuomola and Asefon (2020)

Ikuomola and Asefon proposed “A Secured Mobile Cloud-Based House Rental Management System” using Flutter, Flask and SQL Server, with encrypted cloud records and a wallet-based payment engine. The work improves record security for landlords and tenants. However, it is mobile-only and includes no AI features, geolocation mapping or smart search filters. [2]

2.3 Gommans, Njiru and Owange (2014)

Gommans et al. presented “Rental House Management System,” a classic three-tier system in HTML, PHP and MySQL covering tenant registration, property listing, landlord contact and admin control. It established the standard modules of rental systems. However, it has no maps, images, real-time messaging, mobile responsiveness or AI components. [3]

2.4 The Rental Zone (2022)

“The Rental Zone — House Renting Website” (IRJMETS) combined PHP, MySQL, Bootstrap and the Google Maps API to give a responsive, location-aware rental interface. However, it provides no chatbot or recommendation engine, no price-range filter and no owner verification, and communication relies on basic forms. [4]

2.5 MERN Rental Platform (2026)

“Home Rental Finding Platform Using MERN Stack” (IJISRT) built a full-stack rental platform with image upload through Cloudinary and a chatbot that displays properties from user input. However, the chatbot is rule-based and limited to listing display, with no NLP, price sliders, geolocation filters or owner verification. [5]

2.6 NLP for Customer Service Chatbots (2024)

This IJISAE study compared NLP techniques such as intent detection, entity recognition and sentiment analysis across industry chatbot deployments, finding better first-contact resolution than rule-based bots. However, the findings are domain-general and were not validated on rental or housing data. [6]

2.7 Intent Matching Chatbot (2022)

The arXiv study “Intent Matching Based Customer Services Chatbot with Natural Language Understanding” used Dialogflow for intent recognition and reported a 30% cost reduction in customer service. However, it depends on a manually maintained intent library and does not adapt to property-specific terminology without retraining. [7]

2.8 Dash et al. (2024)

Dash et al. developed an NLP chatbot with tokenisation, intent recognition and sentiment scoring integrated with a CRM, reducing manual customer handling by about 40%. However, it depends heavily on training data quality and was not tested on property or housing datasets. [8]

2.9 Giri, Yadav and Rakhde (2024)

Giri et al. systematically reviewed NLP chatbot implementations across banking, e-commerce and healthcare, finding that transformer-based chatbots outperform rule-based systems by 25–35% in intent accuracy. However, the review is cross-industry and gives no housing-specific evaluation metrics. [9]

2.10 Context-Aware LLM Chatbots (2025)

The arXiv study “Enhancing Smart Environments with Context-Aware Chatbots Using Large Language Models” combined UWB localisation with an LLM to improve assistance relevance in smart offices. However, it depends on specialised hardware not applicable to a web rental platform, and LLM API costs may be prohibitive at small scale. [10]

2.11 Housing Rental Recommendation by Data Mining (2023)

This ICMLCA paper proposed a personalised rental recommender based on grey relational analysis and a user-interest model, improving accuracy by 32% over a baseline. However, it is computationally expensive at scale, suffers from cold start for new users, and uses a China-specific dataset. [11]

2.12 Rental Recommendation by Deep Learning (2023)

This work combined a TextCNN for property description features with content-based filtering and reached an F1-score of 0.87. However, quality depends on listing descriptions, the model needs periodic retraining, and no end-user interface was demonstrated. [12]

2.13 LLM Hotel Recommendations (2024)

The arXiv study “Enhancing Hotel Recommendations with AI” applied retrieval-augmented generation to summarise reviews and answer queries, saving about 60% of reading time. However, it is hotel-focused, carries a hallucination risk for factual details and needs vector-database maintenance. [13]

2.14 Renting Price Prediction (2023)

This MSIEID paper compared linear regression, decision tree, Random Forest and XGBoost for rent prediction, with Random Forest reaching R^2 of 0.89 and location proximity as the strongest predictor. However, the dataset covers one region and does not consider seasonal changes. [14]

2.15 Rental Demand and Supply Analysis (2024)

A Rochester Institute of Technology thesis applied SVM, KNN, Random Forest and ANN to rental price and demand analysis, finding that geospatial features improved accuracy by 18%. However, ANN models are computationally heavy and the dataset is US-centric. [15]

2.16 AI in Real Estate Operations (2025)

A Taylor & Francis systematic review of 87 papers found AI can substantially improve appraisal precision and cut operating costs by 20–30%, and identified chatbot integration as the most underexplored area. However, most reviewed systems are proprietary and benchmark datasets are lacking. [16]

2.17 AI in Real Estate Literature Review (2024)

An IJAEMR review of 45 papers reported that chatbots reduce tenant enquiry handling time by 55% and explicitly identified a chatbot gap in rental platforms. However, it is descriptive, has no experimental validation, and does not cover the Indian rental market. [17]

2.18 GAO Report (2025)

The U.S. Government Accountability Office analysed AI tools reshaping home buying and renting, reporting that conversational search reduces time-to-shortlist by 45% while raising fairness and bias concerns. However, its focus is the US market and policy context. [18]

2.19 Bafna et al. (2022)

Bafna et al. reviewed the MERN stack for web development, noting the benefits of a single language and strong React UI performance, and the risk of data inconsistency from schema-less storage. [19]

2.20 Yousif and Ibrahim (2023)

Yousif and Ibrahim integrated the Google Maps API with a PHP/MySQL backend to show database-driven location markers in real time. However, API costs and free-tier quotas limit scale, and no address auto-complete or proximity search was shown. [20]

RESEARCH GAP

The reviewed studies demonstrate significant progress in rental portals, NLP chatbots, property recommendation, rent price prediction and mapping. However, these capabilities are largely addressed as separate point solutions. Existing rental platforms rely on manual owner–tenant messaging, offer only basic location search, rarely include recommendation, use shallow or no owner verification, provide public or no messaging, and lack visit scheduling. Limited work combines an AI chatbot, multi-criteria search, personalised recommendation, verified listings, private messaging and visit scheduling within a single platform.

The proposed Smart House Rental Web Application with AI Chatbot addresses this integration gap by bringing these functions together for tenants, owners and administrators.

3. SYSTEM ARCHITECTURE

The system follows a layered three-tier client–server architecture, chosen for separation of concerns, independent testability and suitability for a defined-scope academic project. The three tiers are:

- **Presentation Layer** – React.js renders the tenant, owner and admin interfaces, listing pages, dashboards and the chatbot widget. It communicates with the backend only through HTTPS REST calls and holds no business logic.
- **Business Logic Layer** – Java 17 with Spring Boot exposes REST controllers for authentication, property, search, chatbot, messaging, booking and admin modules. A service layer implements business rules, an AI/NLP engine interprets chatbot queries and ranks results, and a security module enforces JWT authentication and role-based authorisation on every request.
- **Data Layer** – MySQL 8 stores users, properties, bookings, messages, chat sessions and recommendations through Spring Data JPA / Hibernate. Property images are kept in file storage with only the file reference in the database.

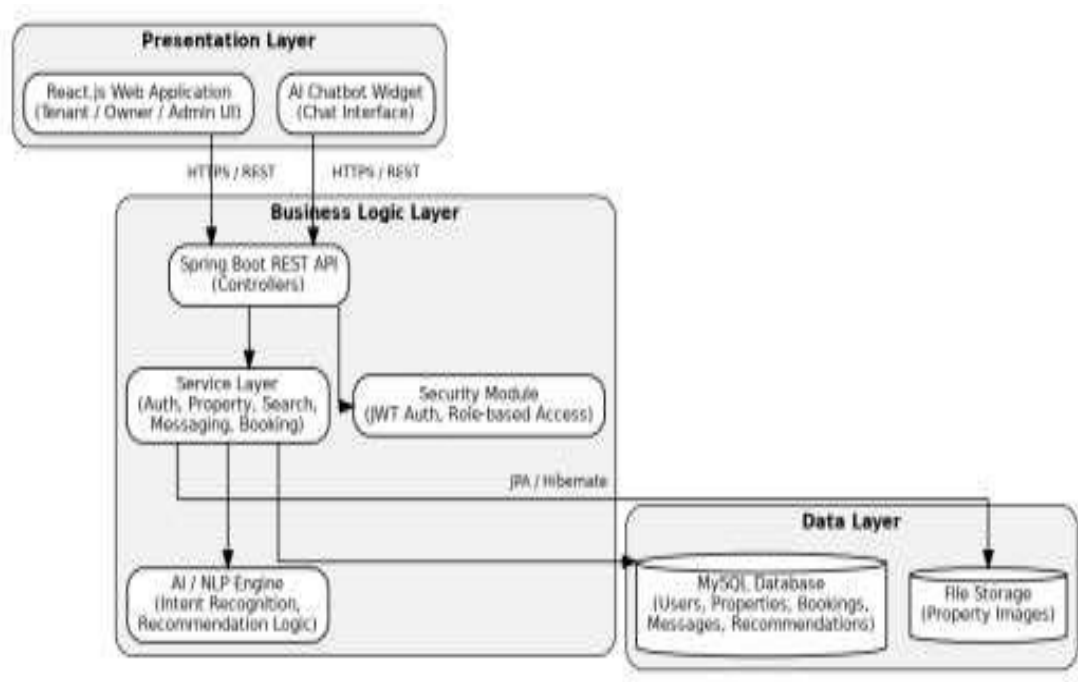


Figure 3.1 – High-Level System Architecture (Three-Tier)

The functional modules are: (1) User Registration and Authentication, (2) Property Listing Management, (3) Advanced Search and Filtering, (4) AI Chatbot and Recommendation, (5) Messaging and Visit Scheduling, (6) Admin Management and (7) Google Maps Integration. Modules communicate only through service classes and own their own tables, keeping them loosely coupled.

4. METHODOLOGY

An Agile, module-wise approach was followed: each module is designed, implemented and tested independently before integration, which suits the review-driven capstone schedule. Authentication was built first because every other module depends on it. The tenant interaction pipeline is:

1. Tenant registers or logs in and receives a JWT
2. Tenant searches with filters or asks the chatbot in natural language

3. The chatbot detects intent and extracts filter criteria
4. Matching properties are retrieved from MySQL and ranked
5. A combined JSON response (reply plus property list) is shown as a chat message with property cards
6. Tenant views details and map location, then messages the owner or requests a visit
7. Owner confirms, reschedules or declines the visit

4.1 UML Modelling

The use case diagram identifies three actors (Tenant, Property Owner, Admin). The class diagram specialises User into Owner, Tenant and Admin, associates Property with VisitBooking and Message, and has a ChatbotEngine use a RecommendationEngine to rank properties. A sequence diagram traces the chatbot property search from the React UI through the Spring Boot API and NLP engine to MySQL and back (Figure 4.1), and an activity diagram models the tenant search-and-visit journey.

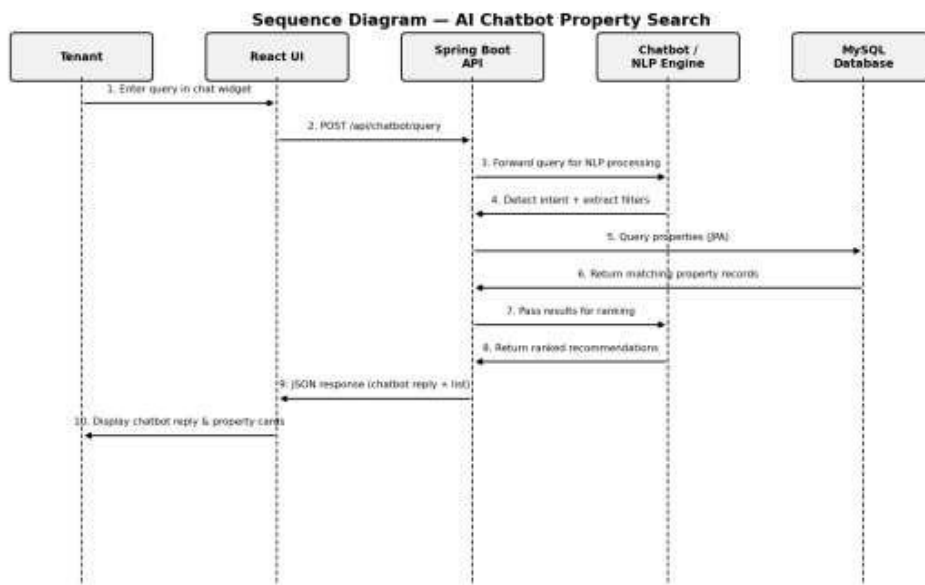


Figure 4.1 – Sequence Diagram (AI Chatbot Property Search)

4.2 AI Chatbot and Recommendation Engine

The chatbot is designed as an intent-detection engine using keyword and pattern matching to classify a message as SEARCH_PROPERTY, FAQ or GENERAL, extract filters such as location, budget and BHK type, and rank matching properties with a scoring function. Each interaction is logged to the CHAT_SESSION table so later recommendations can use accumulated preferences. A single engine powers both chatbot replies and recommendations, avoiding duplicated logic.

4.3 Database Design

The schema is a normalised MySQL design of nine tables: user, owner, tenant, property, property_image, visit_booking, message, chat_session and recommendation. USER is the parent of OWNER and TENANT in a one-to-one specialisation keyed on the shared user_id. A property belongs to one owner and can have many images and visit bookings (Figure 4.2, Table 4.1).

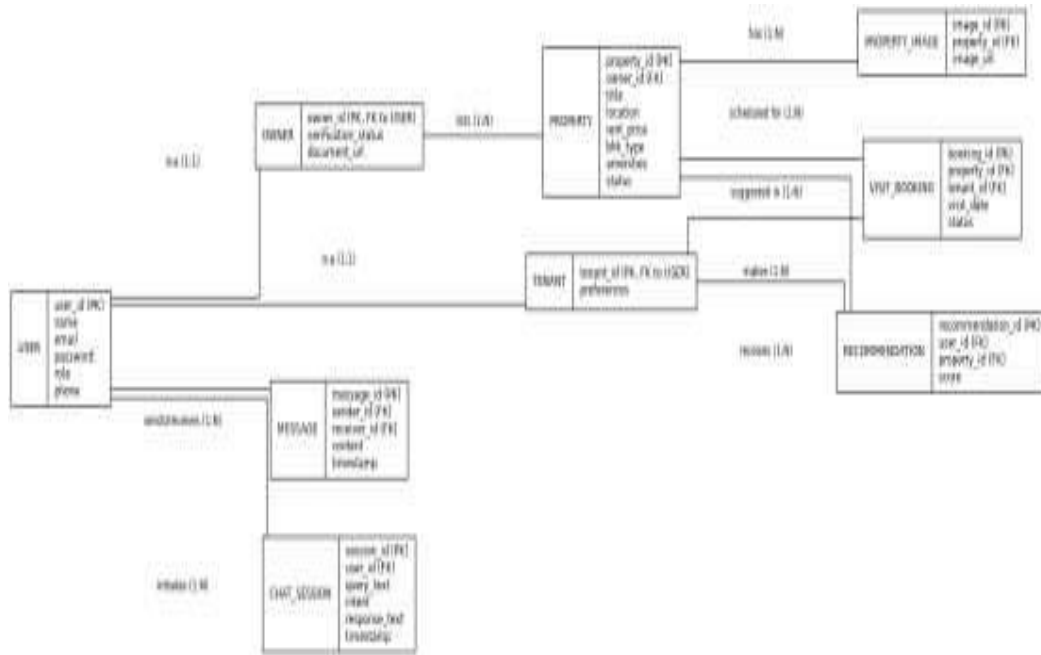


Figure 4.2 – Entity–Relationship Diagram

Table 4.1 – Key Table Structures

Table	Key fields
USER	user_id (PK), name, email, BCrypt password, role, phone
PROPERTY	property_id, owner_id (FK), title, location, rent_price, bhk_type, amenities, status
VISIT_BOOKING	booking_id, property_id, tenant_id, visit_date, status
MESSAGE	message_id, sender_id, receiver_id, content, timestamp
CHAT_SESSION	session_id, user_id, query_text, intent, response_text, timestamp

5. SYSTEM WORKFLOW

After login or registration, users are routed by role. The workflow of each role is:

- Tenant – search properties, view details and map, chat with the AI assistant, receive recommendations, message owners and schedule visits.
- Property Owner – register and upload verification documents, list and manage properties after approval, respond to enquiries and manage visit requests.
- Administrator – verify owner documents, activate or suspend accounts, moderate listings and generate reports.

Data flows through four stores: User DB, Property DB, Message/Booking DB and Chat/Recommendation DB, accessed by six processes: User Authentication, Property Management, Search and Filtering, AI Chatbot and Recommendation, Messaging and Visit Scheduling, and Admin Management (Figure 5.1).

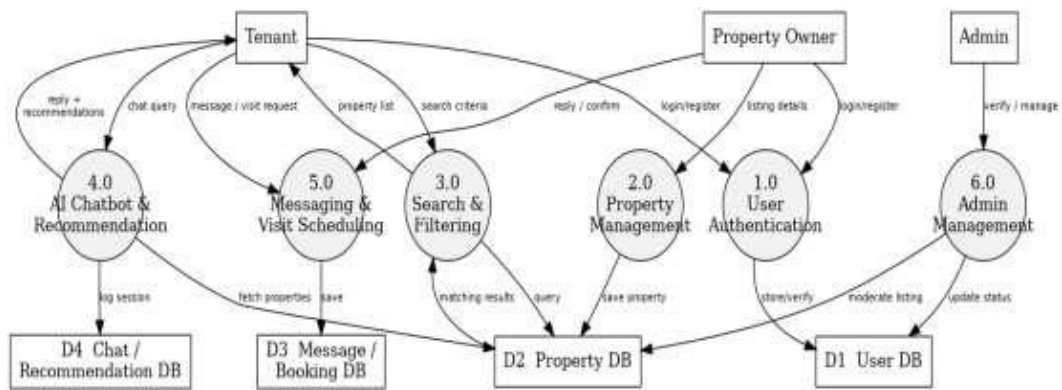


Figure 5.1 – DFD Level 1 (Detailed System Processes)

6. IMPLEMENTATION

6.1 Frontend

The frontend uses React 18 with Vite, React Router for client-side navigation and Axios for HTTP. It is organised into an api layer, an AuthContext for global login state, reusable components (Header, ProtectedRoute) and one page file per screen. An Axios interceptor attaches the JWT to every request and clears it on a 401 response.

6.2 Backend

The backend uses Java 17 and Spring Boot 3.3 with Spring Web, Spring Data JPA / Hibernate and Spring Security, organised into entity, repository, security, config, service, controller, dto and exception packages. Owner and Tenant are mapped as one-to-one specialisations of User using JPA `@MapsId` so their primary keys always equal the `user_id`, mirroring the ER diagram exactly.

6.3 Authentication and Security

On registration the password is hashed with BCrypt, and the User row and the matching Owner or Tenant row are saved in a single @Transactional method so a user never exists without its role-specific row. On login, Spring Security's AuthenticationManager verifies the credentials and a signed JWT carrying the user id, email and role is returned. A custom JwtAuthFilter validates the token on every request so role rules (Owner-only property writes, Admin-only management) apply consistently with stateless sessions. The signing key is read from an environment variable. Additional protections include HTTPS transport, Bean Validation on request DTOs, parameterised JPA queries against SQL injection, restricted CORS origins, and a GlobalExceptionHandler returning uniform JSON errors without stack traces.

6.4 Development Environment

The backend runs on port 8080 (Maven) and the frontend on port 5173 (Vite), with the API base URL externalised through `VITE_API_BASE_URL`. Tools include Visual Studio Code, IntelliJ IDEA, Postman, MySQL Workbench and Git.

7. EXPERIMENTAL RESULTS

7.1 Module Implementation Status

No accuracy, latency or user-study figures are claimed because the AI modules are not yet implemented. Table 7.1 reports what has actually been built.

Table 7.1 – Module Implementation Status

Module	Status
1. Authentication	Implemented end-to-end
2. Property Listing	Scaffolded (entities, repository)
3. Search and Filtering	Scaffolded (repository query)

Module	Status
4. AI Chatbot and Recommendation	Scaffolded (entities, repositories)
5. Messaging and Visits	Scaffolded (entities, thread query)
6. Admin Management	Scaffolded (verification field)
7. Google Maps	Not started

7.2 Home Page

The home page presents navigation and a build-status panel describing the current state of the system.

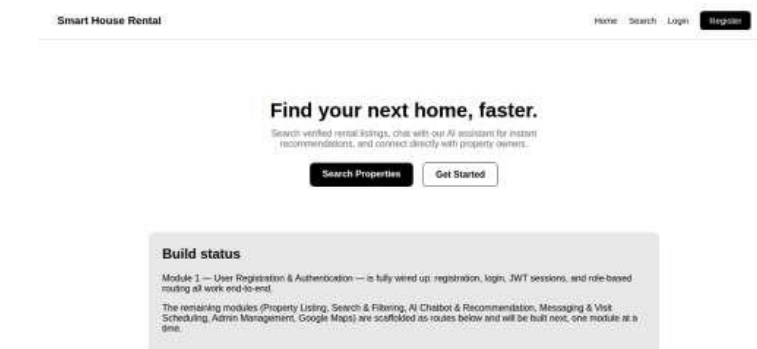


Figure 7.1 – Home page with navigation and build status

7.3 Registration and Login

The registration page supports Tenant and Owner roles and informs owners that admin verification is required before listing. The login page authenticates against the live backend and receives a JWT.

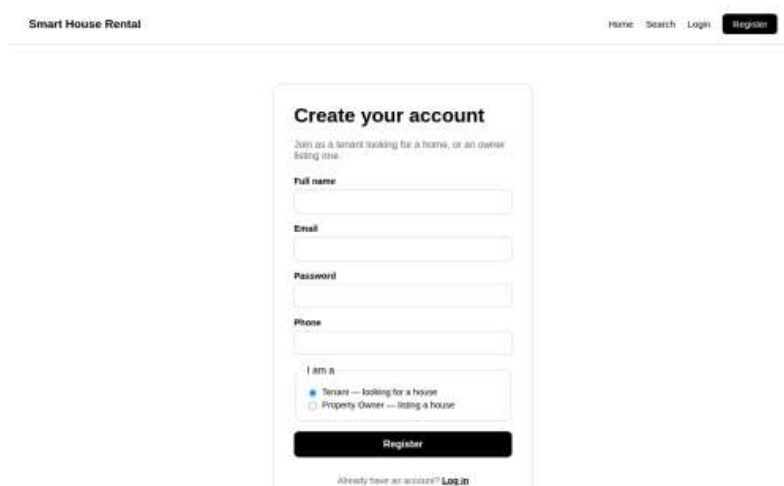


Figure 7.2 – Registration page

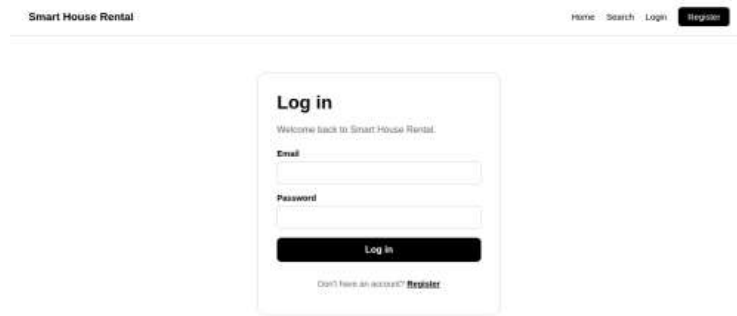


Figure 7.3 – Login page

7.4 Routed Placeholders

Every unbuilt module is already routed behind its correct role restriction and shows a labelled placeholder, so navigation for the whole application is in place and each later module replaces one placeholder.

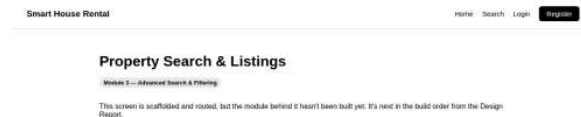


Figure 7.4 – Search page placeholder

7.5 Verification Status

The frontend was installed, built (npm run build) and run (npm run dev), and the screens above were captured from the running instance. The backend was written to standard Spring Boot 3.3 conventions in an environment without access to Maven Central, so compilation and run verification in a connected environment remains a required follow-up. No formal scenario-based test suite has yet been executed.

8. COMPARISON WITH OTHER METHODS

Table 8.1 – Comparison with Existing Approaches

Feature	Existing rental systems [1]–[5]	Proposed System
Owner verification	Manual or none	Document-based, admin-managed
Google Maps	Some systems	Yes (planned module)
AI chatbot	None / rule-based listing display	Intent-based with ranked results
Price and amenity filters	No	Yes
Recommendation	No	Yes
Private messaging	Public or none	Per-property private

Feature	Existing rental systems [1]–[5]	Proposed System
Visit scheduling	No	Yes
Role-based JWT security	Not reported	Yes

The proposed system combines AI assistance, recommendation, verified listings, private messaging and scheduling in one multi-role platform. The comparison is based on features reported in the cited studies; chatbot, search, messaging, scheduling and mapping entries describe the design, while authentication and the data model are the implemented components.

9. DISCUSSION

The survey showed that rental platforms and AI techniques have advanced largely in isolation. The proposed design attempts integration: one NLP engine serves both conversation and recommendation, one security model covers all modules, and one schema supports the planned features without structural rework.

The design choices are deliberately conservative. A rule-and-intent-based chatbot is transparent, needs no training data or paid LLM API, and is feasible for a prototype, but will be less flexible than the transformer-based approaches reviewed in [9] and needs a maintained intent library as noted in [7]. The cold-start problem noted in [11] is mitigated by starting from explicit filters and chat-extracted preferences. Fairness concerns raised in [18] mean ranking logic should stay inspectable.

The main limitation of the current work is that only the authentication foundation is complete and no empirical evaluation has been run. The paper therefore claims design validity and implementation status, not performance.

10. CONCLUSION

The Smart House Rental Web Application with AI Chatbot was designed to make renting more transparent, trusted and efficient for tenants, owners and administrators. The work provides a three-tier design with UML models, a nine-table schema, seven modules and a security design, together with a working implementation of the data model, JWT-secured authentication and a React frontend shell. The remaining modules are scaffolded with entities, repositories and routes already in place, giving a solid and extensible foundation for the chatbot, search, messaging and scheduling features.

11. FUTURE SCOPE

- Complete the property listing, search, chatbot, messaging, scheduling, admin and maps modules
- Compile and verify the backend in a connected environment and add automated tests
- Evaluate chatbot intent accuracy and recommendation quality on real data and with user studies
- Replace rule-based intent detection with transformer or LLM-based NLU
- Add machine-learning rent price prediction (e.g., Random Forest)
- Online rent payment and Aadhaar-based owner verification
- Native mobile application sharing the same backend
- Deep-learning recommendation trained on chat and booking history
- Horizontal scaling and MySQL read replicas

ACKNOWLEDGEMENT

The author sincerely thanks the project guide, Dr. V. Harsha Shastri, Associate Professor, Department of Computer Applications, for guidance and support throughout this project, and is grateful to the Department and Aurora Higher Education and Research Academy for the resources and encouragement provided.

REFERENCES

- [1] D. Voumick, P. Deb, S. Sutradhar, and M. M. Khan, "Development of online based smart house renting web application," J. Software Eng. Appl., vol. 14, pp. 312–328, 2021, doi: 10.4236/jsea.2021.147019.
- [2] A. J. Ikuomola and M. P. Asefon, "A secured mobile cloud-based house rental management system," Ondo State Univ. of Science and Technology, 2020.

-
- [3] H. P. Gommans, G. M. Njiru, and A. N. Owange, "Rental house management system," *Int. J. Sci. Res. Publications*, vol. 4, no. 11, 2014.
- [4] "The Rental Zone – House renting website," *IRJMETS*, vol. 4, no. 8, Aug. 2022.
- [5] "Home rental finding platform using MERN stack," *IJISRT*, Jan. 2026.
- [6] "Natural language processing for customer service chatbots: Enhancing customer experience," *IJISAE*, 2024.
- [7] "Intent matching based customer services chatbot with natural language understanding," *arXiv:2202.00480*, 2022.
- [8] C. Dash et al., "Developing a chatbot system utilizing natural language processing for improved customer service and marketing automation," in *Proc. IEEE CISCT*, 2024, doi: 10.1109/CISCT62494.2024.11134178.
- [9] R. Giri, L. Yadav, and V. Rakhde, "NLP in chatbot customer service," *IJRASET*, vol. 13, no. 5, 2024.
- [10] "Enhancing smart environments with context-aware chatbots using large language models," *arXiv:2502.14469*, 2025.
- [11] "Research on personalized recommendation algorithm for housing rental based on data mining," in *Proc. ICMLCA, ACM*, 2023, doi: 10.1145/3650215.3650310.
- [12] "Research on house rental recommendation algorithm based on deep learning," *Adv. Econ., Bus. Manag. Res.*, vol. 233, pp. 604–613, 2023.
- [13] "Enhancing hotel recommendations with AI: LLM-based review summarization and query-driven insights," *arXiv:2510.18277*, 2024.
- [14] "Research on renting price prediction based on machine learning," in *Proc. MSIEID 2023, EAI*, doi: 10.4108/eai.8-12-2023.2344718.
- [15] "Rental property demand and supply analysis using machine learning," M.S. thesis, Rochester Inst. Technol., 2024.
- [16] "A systematic literature review on artificial intelligence based techniques for nurturing operations in real estate sector," Taylor & Francis, 2025, doi: 10.1080/09277544.2025.2466370.
- [17] "A literature review on application of artificial intelligence in real estate," *IJAEMR*, vol. 9, no. 1, 2024.
- [18] U.S. Government Accountability Office, "AI is changing home buying and renting – but not always for the better," Dec. 2025.
- [19] S. A. Bafna et al., "Review on study and usage of MERN stack for web development," *IJRASET*, vol. 10, no. 2, 2022, doi: 10.22214/IJRASET.2022.40209.
- [20] A. N. Yousif and H. M. Ibrahim, "Online destinations map using Google Maps API based on the private database," *IJITC*, vol. 3, no. 5, 2023.