Mid-Progress Review

MINOR PROJECT

Intelligent Ranking and Chat System for Property Rental

(ES-451: Minor Project - Dissertation)

Submitted in partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

Computer Science & Engineering

Supervisor(s):

Ms. Apurva Jain (Assistant Professor)

submitted by:

Akshat Jain (Roll No. 00115607223)

Khushal (Roll No. 00715607223)

Pankaj Singh (Roll No. 04315602722)

Raghav Dwivedi (Roll No. 01015602722)

Department of Computer Science and Engineering

Dr. Akhilesh Das Gupta Institute of Professional Studies

(Formerly ADGITM) FC-26, SHASTRI PARK, NEW DELHI

Affiliated to

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

Sector - 16C Dwarka, Delhi - 110075, India

2022-26

Declaration

We, the undersigned, hereby declare that the proposed work for the Minor Project titled: "**Intelligent Ranking and Chat System for Property Rentals**", submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (B.Tech.) in Computer Science and Engineering, is based on our original work.

We further declare that this work has not been submitted, either in part or full, to this or any other university/institute for the award of any degree or diploma.

All sources of information and content from other works have been duly acknowledged and referenced wherever applicable.

Student Details

Sl.	Name	Enrollment No.	Mobile No	Signature
1.	Akshat Jain	00115607223	9810684778	
2.	Khushal	00715607223	8287532142	
3.	Pankaj Singh	04315602722	7703818600	
4.	Raghav Dwivedi	01015602722	7065999677	

Date: October 15th, 2025

Place: New Delhi

Table of Contents

Room-Bridge: AI-Powered Room Rental Platform

1. Introduction

Finding suitable rental rooms in large metropolitan cities is a complex challenge influenced by numerous social, economic, and technological factors. The rapid pace of urbanization has created a significant surge in rental housing demand, especially among students, young professionals, and people migrating for employment opportunities. Despite the presence of numerous property listing websites, the process of finding an ideal room still remains cumbersome, time-consuming, and inefficient.

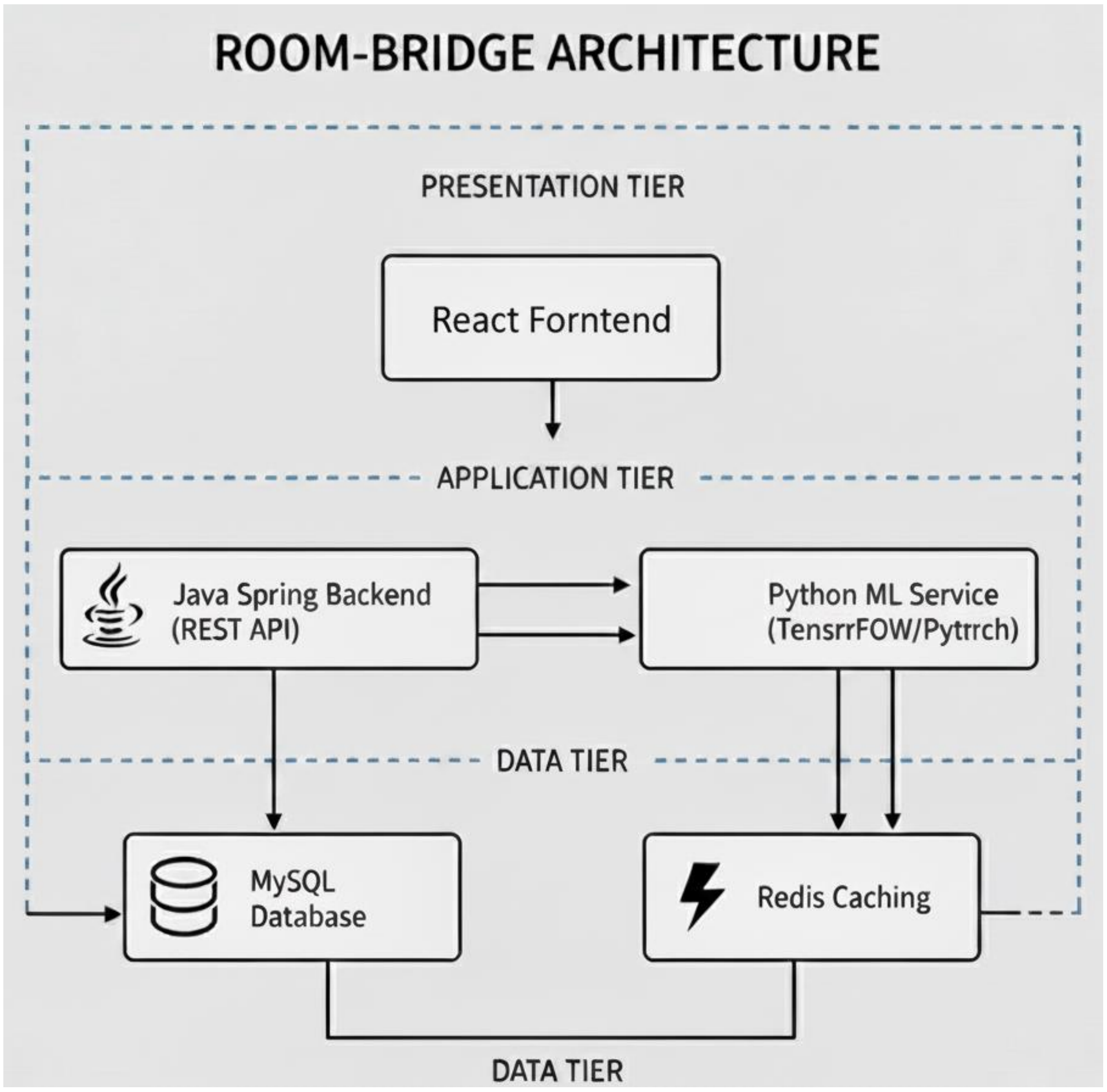
Traditional platforms typically provide basic search filters such as price range, location, and property type. However, they fail to capture the nuanced needs and lifestyle preferences of individuals. For example, a student might prioritize proximity to public transportation and universities, while a working professional may value amenities such as Wi-Fi, parking, or gym access. These platforms rarely incorporate adaptive algorithms that evolve with user interactions, making their recommendations static and often irrelevant over time.

Another major challenge lies in data reliability. Many property listings are either outdated, incomplete, or unverifiable, leading to user frustration and distrust. Tenants often need to manually contact landlords, cross-check information, and verify property conditions — resulting in significant delays. Similarly, property owners struggle to reach the right audience due to platform inefficiencies and lack of intelligent targeting.

The **Room-Bridge** project introduces a transformative approach to this problem by leveraging Artificial Intelligence (AI) and Machine Learning (ML) to deliver personalized and dynamic room recommendations. The system analyzes multiple dimensions — including user preferences, location proximity, price sensitivity, amenities, feedback ratings, and user behavior patterns — to rank listings intelligently.

Unlike conventional systems, **Room-Bridge** integrates a **feedback and rating mechanism**, which ensures data credibility and enhances transparency. By employing **adaptive ranking algorithms** and **predictive analytics**, the platform continually refines its recommendations to reflect real-time trends and changing user interests.

Ultimately, the project seeks to **bridge the gap between static search mechanisms and intelligent recommendation frameworks**. Through data-driven insights, it provides users with accurate, time-efficient, and reliable options while empowering landlords with meaningful engagement and analytics about their listings. This results in a more **user-centric, trustworthy, and scalable rental ecosystem**.



2. Objectives and Scope of the Project

The **primary objective** of this project is to design and develop an AI-driven web application that simplifies the process of discovering and booking rental rooms through intelligent ranking and personalized recommendations.

Objectives

1. **Develop a responsive web application** using React.js that allows seamless browsing, filtering, and visualization of room listings.
2. **Integrate a machine learning ranking model** within the backend to provide personalized recommendations based on user preferences and interactions.
3. **Enable data visualization** features to display real-time trends, such as popular localities, pricing distributions, and room availability over time.
4. **Implement robust security mechanisms** — including secure authentication, encrypted data storage, and access control — ensuring safety and privacy.
5. **Establish a transparent review system**, where users can provide feedback and ratings that influence future recommendations and property rankings.
6. **Ensure platform scalability and maintainability**, using microservices architecture and modular design principles.

Scope

The **target audience** includes:

- Students relocating for education.
- Working professionals seeking affordable, reliable accommodations.
- Landlords and property managers who want to reach the right tenants efficiently.

The **current phase** focuses on:

- Requirement gathering and analysis.
- Conceptual design of architecture, data models, and APIs.

- Development of an initial prototype demonstrating the platform’s user interface and preliminary ranking features.

The **future scope** includes:

- Integration of advanced AI ranking and real-time recommendation systems.
- Implementation of **NLP-based chatbots** for conversational property search.
- Expansion to multi-city listings and integration with mapping APIs.
- Continuous feedback-driven model improvement and real-time analytics dashboards for landlords.

3. Work Completed So Far

3.1 Literature Survey / Background Study

A thorough survey of existing room rental platforms such as **MagicBricks, 99acres, and NoBroker** reveals that most rely on **static search filters**. These systems provide limited personalization, as they fail to adapt to individual user behavior or changing market dynamics. Moreover, their recommendation logic is typically rule-based rather than data-driven.

Research in **recommender systems** emphasizes two major approaches:

- **Collaborative Filtering**, which relies on user-user or item-item similarity based on interaction data.
- **Content-Based Filtering**, which matches user preferences with item attributes.

While both are widely used, they lack adaptability and do not account for contextual information such as **time-dependent demand, neighborhood popularity**, or **real-time availability**.

This project builds upon existing recommender system theories but introduces a **hybrid adaptive ranking model** that combines multiple methods with continuous feedback integration. Prior academic works such as *Adomavicius & Tuzhilin (IEEE)* and *MDPI's AI-Driven Recommendations Review* informed the design of the ranking algorithms and feedback mechanisms.

3.2 Analysis and Conceptualization

Surveys conducted among students and professionals identified several critical user needs:

- Accuracy in ranking based on personal preferences.
- Verified and trustworthy property listings.

- Visual representation of locations on maps.
- Secure communication channels with property owners.

Based on these findings, a conceptual framework was designed integrating **AI-driven ranking**, **secure data management**, and **intuitive UI/UX**. The framework emphasizes a **modular microservices architecture**, allowing independent scaling and faster development cycles.

3.3 Initial Design / Methodology

3.3.1 System Architecture & Design Approach

The proposed architecture consists of the following major components:

- **Frontend(React.js):**
  Provides user-facing interfaces for property browsing, search filters, map visualizations, and authentication. Features include interactive maps (via Google Maps API), responsive dashboards, and real-time data rendering.
- **Spring Security:**
  Acts as a unified entry point for client requests. Handles authentication (JWT tokens), routing, validation, and service communication.
- **Backend in Java:**
  - *Listings Service:* Manages CRUD operations for rental properties.
  - *User Service:* Handles user profiles, preferences, and feedback.
  - *Analytics Service:* Generates insights such as trending localities and average rent per area.
- **Machine Learning Service (Python/Scikit-learn):**
  Implements the AI ranking engine using regression and ranking algorithms (e.g., XGBoost Ranker, LambdaMART). Handles feature extraction, normalization, and continuous learning from feedback.
- **Database Layer (MySQL):**
  Stores structured data such as listings, user profiles, transaction records, and metadata.

- **Caching and Messaging Systems (Redis):**
  Used for asynchronous communication, caching frequently accessed data, and real-time updates.

Design Principles: modularity, scalability, high availability, security, observability, and maintainability through continuous integration and deployment (CI/CD).

3.3.2 Planned Methodology for Implementation and Testing

The project follows the **Agile (Scrum)** methodology, enabling iterative development and continuous improvement.
Each sprint (2 weeks) delivers tangible functionality increments — from basic CRUD features to intelligent ranking integration.

Feature Implementation Order:

1. CRUD and authentication
2. Search and filter integration
3. Baseline ranking model
4. Advanced feature engineering
5. Model deployment and serving
6. Feedback-based model retraining

Testing Strategy:

- **Unit Testing:** JUnit and PyTest for backend and ML components.
- **Integration Testing:** Ensures smooth data flow across services.
- **Model Evaluation:** Metrics like NDCG@k, Precision@k, and MAP to validate ranking quality.
- **Load Testing:** Simulates concurrent user activity to test scalability.
- **Security Testing:** Validates encryption, access controls, and vulnerability protection.

Deployment:

Containerized deployment using **Docker** and automated CI/CD pipelines via **GitHub Actions**.
Canary testing and rolling updates ensure minimal downtime.
Monitoring pipelines detect model drift and trigger retraining workflows.

3.4 Prototype / Initial Implementation

A functional prototype has been developed using React.js for the frontend and Node.js for backend experimentation. It demonstrates:

- Login and authentication workflows.
- Dynamic listings fetched from sample datasets.
- Basic ranking and filtering functionality.

Early testing revealed the importance of **data normalization**, as inconsistent listing data affected model performance. User feedback led to UI enhancements for better responsiveness and accessibility.

3.5 Challenges / Issues Faced

- **Data Inconsistency:** Variability in formats and missing attributes from third-party sources.
- **Integration Complexity:** Communication between Java microservices and Python-based ML components.
- **Limited Training Data:** Lack of sufficient real-world data for accurate ranking model training.

Mitigation Strategies:

- Creation of synthetic datasets using generative techniques.
- Modular API design with well-defined endpoints for cross-language interaction.
- Continuous integration tests and mocking frameworks for API validation.

4. Tools, Platforms, and Methodology

- **Frontend:** React.js for UI components, TailwindCSS for styling, Axios for API calls.
- **Backend:** Java (Spring Boot) for business logic and RESTful APIs.
- **Database:** MySQL for structured data storage.
- **AI/ML:** Python, Pandas, Scikit-learn, and XGBoost for training ranking models.
- **DevOps:** Docker, GitHub Actions, Prometheus, Grafana for CI/CD and monitoring.
- **Testing:** Postman for API testing, JMeter for load testing.
- **Version Control:** GitHub with feature branching workflow.

The chosen methodology — **Agile (Scrum)** — allows flexibility, incremental releases, and continuous user feedback integration.

5. Preliminary Design / Implementation

Preliminary diagrams (DFDs, UML, ERD) visualize how user requests travel across system layers.
The prototype demonstrates:

- Secure authentication.
- Real-time listing retrieval via REST APIs.
- Responsive UI capable of adapting to devices.
- Early-stage ranking algorithm functioning on sample data.

Design prioritizes **scalability**, **fault tolerance**, and **clean separation of concerns** to ensure future extensibility.

6. Challenges / Issues Faced

1. **Integration Complexity:** Managing communication between AI microservices and real-time backend APIs.
2. **Data Synchronization:** Handling dynamic property updates while ensuring consistency across databases.
3. **Security Concerns:** Implementing JWT-based authentication, HTTPS, and data encryption.
4. **Performance Bottlenecks:** Reducing inference latency of the ranking model.
5. **Technology Dependencies:** Ensuring compatibility across versions of React, Java, MySQL, and Python.

Mitigation:

- Use of caching (Redis) for faster query retrieval.
- CI/CD pipelines for automated testing and deployment.
- Regular code reviews and documentation updates.

7. Next Steps / Future Plan

1. Train the ranking model on real and synthetic datasets to improve prediction quality.
2. Integrate real-time recommendation APIs using hybrid filtering methods.
3. Enhance UI/UX with map-based visualizations and advanced search filters.
4. Conduct extensive user testing and collect feedback for iterative improvements.
5. Deploy application to cloud environments such as AWS or Azure using container orchestration (Kubernetes).
6. Finalize documentation, performance evaluation, and publish research findings on AI ranking for property rentals.

8. Significance of the Project

The **Room-Bridge** project represents a major step toward transforming online property rentals through the power of Artificial Intelligence. It bridges the gap between **static listings** and **adaptive recommender systems**, providing an intelligent, user-centered platform.

Key contributions include:

- **Personalized Recommendations:** Dynamic suggestions aligned with user lifestyle, preferences, and behavior.
- **Enhanced Credibility:** Verified listings, feedback loops, and transparent ranking metrics.
- **Social Impact:** Reduces housing search stress and prevents fraud by verifying and filtering unreliable data.
- **Technological Innovation:** Demonstrates practical integration of AI/ML, full-stack web technologies, and scalable cloud deployment.
- **Academic Value:** Provides a comprehensive case study on intelligent recommender systems applied to the real estate sector.

Ultimately, Room-Bridge not only simplifies the process of finding rooms but also contributes to **trust, transparency, and data-driven decision-making** in the rental market ecosystem.

References

1. **Airbnb Engineering Blog** – Articles on search ranking, trust, and verification. Available at: https://medium.com/airbnb-engineering
2. **MagicBricks** – Real estate listings and property information. Available at: https://www.magicbricks.com/
3. **99acres –** Property portal for buying, selling, and renting properties. Available at: https://www.99acres.com/
4. **Trulia Engineering –** Research on recommendations in rental/property platforms. Available at: https://www.trulia.com/research/
5. **ISO/IEC 25010:2011 –** Systems and software engineering – Systems and software quality models (for system verification & usability).
6. Scalable and Secure Real-Time Chat Application Development Using MERN Stack and Socket.io for Enhanced Performance. Available at: https://www.macawpublications.com/Journals/index.php/FCR/article/view/51
7. **Socket.io Documentation** – Real-time chat implementation. Available at: https://socket.io/docs
8. **MySQL Documentation –** Handling property listings and relational data. Available at: https://www.mysql.org/docs