



## RoomMate: An AI optimized Space Rental System

Ayesha Farooq<sup>1</sup>, Israr Hussain<sup>2\*</sup>, Hifza Iman<sup>3</sup>, Nagina Kiran<sup>4</sup>, Nadeem Iqbal  
Kajla<sup>5</sup>, Muhammad Ahsan Jamil<sup>6</sup>

<sup>1, 2</sup> <sup>\*,3,4,5,6</sup> Institute of Computing, MNS University of Agriculture, Multan, 60000, Pakistan

**\*Corresponding Author:** Israr Hussain, [israr.hussain@mnsuam.edu.pk](mailto:israr.hussain@mnsuam.edu.pk)

### Abstract

The growing demand for affordable and flexible student housing has highlighted the shortcomings of traditional rental platforms. Most existing services like Airbnb, OLX, NestAway, and NoBroker primarily cater to families, professionals, or tourists—leaving students underserved. These platforms often lack essential features like room-sharing, roommate matching, and affordable short-term options near campuses. Security and trust are also major issues, as many platforms do not offer student-focused verification systems, increasing the risk of scams and fake listings. In addition, the absence of live support or AI chatbots results in delayed assistance for new users. Admin control is often weak, with poor content moderation and no dedicated admin dashboard to manage listings and users effectively. RoomMate addresses these gaps by offering a cross-platform, student-centric mobile application tailored to the unique housing needs of students. It enables users to find, share, and manage rentals within a trusted community. The app features role-based access for tenants, landlords, and administrators, with real-time communication, secure authentication, and strong administrative oversight. In short, RoomMate provides a cost-effective, reliable, and community-driven solution for student housing, empowering a digital transformation in the rental space.

**Keywords:** *RoomMate, Student Housing, Room Sharing, React Native, AI Chatbot, Rental Platform, Node.js, MongoDB, Rental Sector.*

## **1. Introduction**

Access-based consumption (“sharing economy”) has grown exponentially over the past few years, changing the way individuals acquire goods and services [sharing economy.org](#). Peer-to-peer (P2P) rental platforms now offer flexible short-term access to products, furniture and electronics to living space—inexpensive substitutes for ownership [sharingeconomy.org](#). The sites allow consumers to save money and avoid waste while gaining access to a variety of products with no long-term obligations [sharingeconomy.org](#). This digital revolution has particularly touched the lodging and housing sectors: contemporary rental services now offer easy-to-use mobile interfaces, secure payment paths, and strong control over privacy to facilitate transactions and foster trust between users. Following these trends, students nowadays anticipate the simplicity of consumer-grade mobile technology for housing services ([housing.cloud](#)). Generation Z students are smartphone-reliant; however, they do not want to fill their phones with specialist applications. Instead, they require completely mobile-optimized and user-friendly platforms for daily use ([housing.cloud](#)). Empirical evidence suggests that if students are having trouble searching for or managing housing information, their satisfaction and trust in the service will likely be negatively affected ([housing.cloud](#)). Most existing rental apps like Airbnb, OLX, and NoBroker are not designed for students—they lack affordable, short-term, and shared housing options, as well as features like roommate matching. These platforms often have unverified listings, limited admin control, no real-time support, and cluttered interfaces. RoomMate solves these issues by focusing exclusively on student needs. It offers room-sharing, verified listings, an AI chatbot for instant help, a clean mobile UI, and an admin dashboard for content moderation. It also builds a community-driven platform where students can help each other find safe and affordable housing. RoomMate is designed as a mobile-first, community-driven solution; as a React Native application, it can run easily on both iOS and Android platforms, allowing students to browse listings and converse with landlords on the go. Students have special challenges in finding accommodation. Most have to move to a new city or foreign country to pursue their studies, exploring unfamiliar rental markets at a distance. Affordability is a key issue: studies have shown that students spend 40–50% of their income on housing. RoomMate addresses these requirements with a student-focused, role-based mobile platform. It streamlines the renting and room-sharing experience via a localized network: students can simply add or find rooms in their university community, and landlords can advertise confirmed accommodation.

Developed using React Native, RoomMate provides a seamless, cross-device experience. Role-based access control on the platform enables tenants, landlords, and administrators to have necessary permissions. Admin-level content moderation and verified user accounts build trust by authenticating identities. In-app messaging and real-time notifications keep landlords and tenants constantly connected. Overall, the design of RoomMate is about a frictionless, trusted renting experience by combining contemporary mobile usability with selective smart-home features and support from a community.

### **1.1 Graph Theory – User Connections and Listings**

**Use Case:** Simulating user interactions like tenant-landlord messaging and shared housing networks.

**Model:** Let  $G = (V, E)$ , where:

- $V$  stands for the set of users (renters and landlords),
- $E$  stands for the set of edges indicating communication networks or rental contracts between users.

### **1.2. Objective**

To provide the *RoomMate* application with sound architecture, a number of important targets guide the system's design:

- To design and develop a student-focused mobile application for rental and room-sharing that effectively addresses the unique accommodation needs of students.
- To enhance user experience and trust by implementing role-based access for tenants, landlords, and administrators, coupled with real-time communication
- To integrate an AI-powered chatbot that improves user interaction by providing real-time support, answering common inquiries, and assisting navigation during the leasing process

The aims of the RoomMate platform are in line with the increasing demand for digitally empowered, community-driven student housing solutions, offering not just technological innovation but also genuine assistance for young tenants.

## **2. Literature Review**

Past studies on rental platforms emphasize user-focused features for convenience of property search and making a booking. Paul's study [10]. describes a prototype rental platform ("RentHub") with easy sign-up, categorized listing, and location filtering, and finds that these design choices led to high user satisfaction. Most importantly, Paul [10] finds that the

inclusion of direct chat (via WhatsApp) and detailed property descriptions significantly added to users' trust in listings. Similarly, Rathore et al. [12] and Setty [13] describe rental management systems that make list management and user roles easy, highlighting the need for easy interfaces and robust backend controls. These findings suggest that RoomMate's mobile platform needs to emphasize ease (e.g. easy sign-up, categorized search, map filters) and unambiguous information, as these parameters have been shown to enhance usability and trust in peer-to-peer rental solutions [10]. Recommender systems are extensively used throughout rental and sharing websites to enhance user experience with personalization. Content-based and collaborative filtering methods have proven to be effective in comparable fields. For instance, Permana et al. [13] constructed a TF-IDF-cosine similarity-based recommender system for film sets, demonstrating text-based features can provide accurate recommendations even where explicit ratings are limited. Moreover, Kannout et al. [4] solved the "cold-start" problem by integrating clustering algorithms with FP-growth, thereby demonstrating hybrid approaches can improve recommendation precision in instances of new items. In rentals, Paul [10] implemented a machine-learning-based recommender recommending pertinent listings, resulting in higher user interaction, as evidenced through reduced RMSE levels and highly correlated predicted and actual satisfaction scores. These studies justify RoomMate's decision to enact a recommender system: by utilizing user profiles and previous interactions, RoomMate can recommend rooms or flatmates that align with personal interests, utilizing the asserted boost in user interaction and satisfaction as evidenced in [10] [13] [4]. IoT functionality integration on housing platforms is a new direction that can add convenience and security. While few mobile rental apps to date have IoT functionality, smart home research gives relevant insight. For example, smart environment sensing and smart locks have been demonstrated to automate mundane tasks (such as check-in or climate control) in a rental context, enhancing user experience. Such IoT installations must, however, be controlled properly to ensure trust: current research on peer-to-peer IoT networks shows that secure device authentication and communication protocols are essential to avoid malicious access. Direct IoT research on rental apps is limited, but the general literature (e.g. on P2P IoT trust [5]) is that users only embrace automated convenience if strong security is guaranteed. RoomMate's design thus inherits industry standard IoT practice (secure smart lock authentication, encrypted sensor data) so that functionality such as remote door entry as well as usage monitoring can operate

securely. This emphasis on IoT is one of the key innovations, bridging the gap identified by current platforms that have not typically combined on demand rentals with home automation. Finally, trust and credibility are recurring themes in peer-to-peer housing literature. Users often exhibit skepticism toward strangers' listings, so platforms must incorporate trust-building mechanisms. Malhotra and Fatehpuria [8] explicitly identify "consumer scepticism" as a barrier in the rental marketplace and show that appealing to environmental consciousness and minimalist values can mitigate doubts about renting. Paul [10] similarly notes that transparent information and direct communication channels (e.g. chat functions) help reassure users. Other works (e.g. Kapoor and Vij [7] have found that clear policies and responsive customer support increase conversion on rental platforms. Room- Mate addresses these lessons by featuring verified student profiles, review/rating systems, and visible sustainability endorsements (e.g. highlighting the eco-benefits of renting textbooks or furniture) to reduce skepticism. In summary, the reviewed studies [8] [10] [13] [7] [10] collectively inform RoomMate's design: they demonstrate the importance of an easy-to-use interface, personalized recommendations, IoT-enabled convenience, and trust mechanisms. By weaving together these insights, RoomMate fills a research gap in the literature – namely, the need for a mobile, cross-platform student housing app that uniquely combines roommate matching and IoT- enhanced rental management within a single Re- act Native application.

### **3. Architecture**

RoomMate employs a three-tier modular architecture with the presentation tier, application tier, and data tier. The architecture is useful for maintainability, scalability, and separation of concerns [11, 14].

#### **3.1. Presentation Layer**

The frontend is developed based on React Native, which is capable of supporting cross-platform functionality for iOS and Android platforms. It manages user interaction activities like viewing property adverts, making bookings, and engaging with landlords. All data transactions are conveyed to the backend by way of RESTful API calls on HTTPS, thereby offering security and minimizing frontend complexity [2].

#### **3.2. Application Layer**

The backend, which is Node.js and Express.js code, contains the business logic, API endpoints, and role-based access control (RBAC). Different types of users (tenant, landlord, admin) have different rights handled by middleware and authorization layers

[3,9]. Real-time functionality such as chat and notifications is handled by Socket.IO, while MQTT is used to communicate

### **3.3. Data Layer**

MongoDB, being a NoSQL system, accommodates long-term user data, listings, and transaction history. Its document-based model facilitates dynamic schemas that accommodate diverse property attributes and user profiles. Mongo connects the backend system with MongoDB, enabling better efficiency in querying and maintaining the schemas. MongoDB Atlas also accommodates cloud services that provide scalability and data replication.

### **3.4. AI Chatbot Integration**

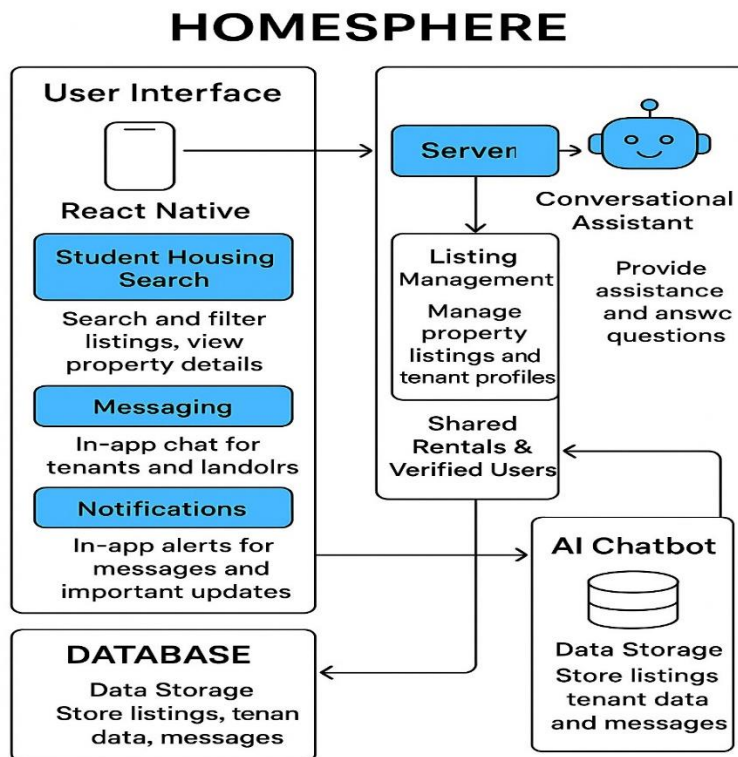
The chatbot powered by artificial intelligence is deployed within RoomMate as a modular microservice to greatly enhance user interaction for administrators, landlords, and tenants. The chatbot communicates with the system via RESTful APIs and can be accessed via a custom chat interface within the React Native mobile app.

In the frontend, the chatbot is implemented directly in the app using the use of React Native components. It enables synchronous communication and includes features like message history, typing indicators, and notifications.

The backend, which is developed using Node.js, processes chat messages by passing the messages to the artificial intelligence engine (i.e., Dialogflow). It sanitizes input, formats responses, and also communicates with the MongoDB database to fetch dynamic information, for instance, room status or booking status. The chatbot supports real-time database query functionality through the backend so that it may respond to natural language questions like "Are there rooms near XYZ College for less than Rs. 10,000?". This makes the chatbot an operational interface for room search, booking status, and other data-driven functions. For improving performance over a period, the chatbot employs a feedback loop. Both explicit feedback (e.g., thumbs up/down) and implicit feedback (e.g., drop-off, sentiment) are maintained in the database and periodically analyzed to enhance conversational flows and accuracy.

**Table 1:** Comparison of Related Works Relevant to RoomMate

Author(s)	Platform/Study	Key Features	Relevance to RoomMate
Paul (2021)	RentHub	Integrated chat, intuitive UI, WhatsApp support	Inspired user-friendly signup and messaging
Rathore et al. (2020)	Rental Management System	Role-based access, listing filters	Reinforced landlord/admin role design
Setty (2018)	RoomShare Platform	Web app for shared rentals	Help build roommate matching flow
Permana et al. (2020)	Recommender Model	TF-IDF, cosine similarity for text	Inspired content-based suggestions in app
Kannout et al. (2019)	Hybrid Recommender	FP-Growth with clustering	Helped address cold-start recommendation issue
Gupta & Singh (2019)	IoT Trust System	Secure peer-to-peer smart control	Informed IoT smart lock authentication
Malhotra & Fatehpuria (2019)	Trust Mechanisms Study	User psychology, minimalism, eco-focus	Used for trust-building UI design
Kapoor & Vij (2017)	Peer Platform Trust Study	Responsive support, transparency	Reinforced rating/review features



**Fig. 1:** System Architecture of RoomMate.

## **Key Features of the AI Chatbot**

The AI chatbot in the *RoomMate* app provides various intelligent features that improve user interaction and simplify platform functionality.

**Search Assistant:** The chatbot enables users to search for rooms via natural language requests like “Find rooms around XYZ College below Rs. 10,000.” It parses these inputs by identifying important parameters like location and budget and then querying the backend database to bring back the most appropriate listings in real time. **Landlord-**

**Tenant FAQs:** It gives instant responses to common questions from landlords and tenants. Landlords can ask about processes of verifying tenants or guidelines in managing profiles, while tenants can ask about documents that need to be required or booking procedures.

**\*Booking Guidance and Support: \*** The chatbot supports consumers during the booking process by providing recommendations like contacting the landlord, asking to visit properties, or verifying room availability. It follows up sensibly after listing displays to lead consumers through steps in finishing a booking.

**Text Administrative Support:** The chatbot further assists landlords and administrators in managing the property listings, keeping track of user queries, and facilitating communication. This lessens the dependency on frequent manual intervention and enhances the overall efficiency of the platform.

RoomMate was created based on an Agile Software Development Life Cycle (SDLC) model, which focuses on iterative development and on-going feedback [6].

### **3.5. Requirement Analysis**

Research was initially carried out through interviews and surveys of students and landlords. The requirements were written down as use cases and user stories, providing coverage of critical features such as IoT support, secure login, and smart booking [15].

### **3.6. Implementation**

The application was created with React Native (frontend), Node.js (server), and MongoDB (database). Version control was achieved through Git, with every feature created in a separate branch and merged once code review had been done [1]. Real-time chat, role-based access, and IoT capabilities were added step by step.

Figure 2: Implementation Flow of RoomMate

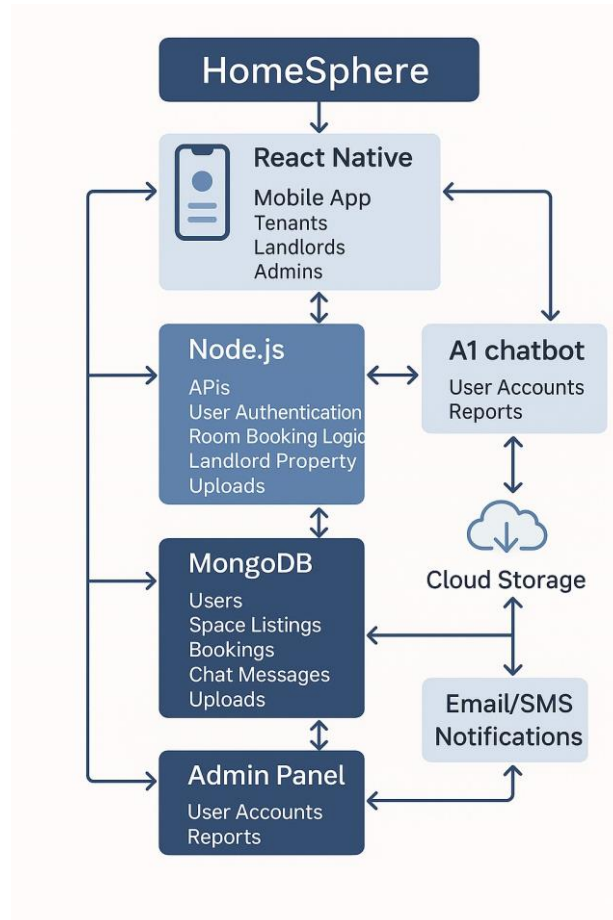


Fig. 2: Implementation Flow of RoomMate

#### 4. RoomMate Architecture Computational Aspects

##### 4.1. Machine Learning and Probability – AI Chatbot / Recommendations

**Use Case:** Improving user engagement through suggestions of lists, personalized answer sets, and auto-FAQs with AI-driven systems.

**Model:** A probabilistic classifier (for example, Naive Bayes or Transformer models) can be employed for recognizing intents and generating recommendations.

**Example:** Using Bayes' Theorem to calculate the probability of a user's intent given their query:

$$P(query | intent) \cdot P(intent)$$
$$P(intent | query) = \frac{P(query | intent) \cdot P(intent)}{P(query)}$$

This approach enables the AI chatbot to provide context-aware and data-driven responses to users in real-time.

## 5. Evaluation and Comparative Analysis

### 5.1. Evaluation of Existing Student Housing Platforms

Several student room-sharing and rental applications were compared to place the development of RoomMate in the existing landscape. Some of them include *NestAway*, *Housing.com*, *99acres*, *NoBroker*, *Zolo*, and *Flatmates.in*. Majority of these applications are focused on making housing easier but have drawbacks that make them unsuitable for students.

Some of the common problems found in these applications are:

**5.1.1. Lack of Student-Centric and Role-Based Features:** Current platforms lack features customized as role-based access control for students, landlords, and admins. They also do not support built-in roommates matching according to preferences, and do not have admin moderation features required to ensure listing quality and platform safety.

**5.1.2. Trust, Security, and Smart Living Gaps:** Most platforms lack extensive identity verification processes, resulting in unverified profiles that lower trust and safety. Moreover, there is poor or no integration of smart-home features such as smart locks or environment monitoring, which are rapidly pertinent to contemporary student living.

**5.1.3. User Experience and Affordability Limitations:** Menus are commonly cluttered, not optimized for mobile-first Gen-Z users, and drowned in ads. Additionally, websites such as *Zolo* provide inflexible rental packages, whereas more comprehensive services such as *NoBroker* and *Housing.com* fail to cater to primary student concerns including affordability, roommate management, and flexible lease arrangements.

### 5.2. Comparative Feature Analysis

RoomMate is distinguished by providing a React Native-powered, cross-platform dwelling solution specifically designed for students. Unique stand-out features are:

- 1. Role-based access:** Isolated dashboards and role-specific permissions for students (tenants), landlords, and administrators.
- 2. Verified user system:** Profile validation and listing authentication to facilitate trust. Community moderation: Admin-level capabilities for listing approval, conflict resolution, and content management. Cross- platform support: Smooth Android and iOS device experience with React Native.

Feature RoomMate	NestAway	Housing.com	99acres	NoBroker	Flatmates.in	Zolo
Cross-Platform	✓	✓	✓	✓	✓	✓
App						
Role-Based Access	×	×	×	×	×	×
Verified Listing	✓	✓	✓	✓	×	✓
Student-Centric Design	×	×	×	×	✓	×
Roommate Matching	×	×	×	×	✓	×
In-App Messagings	×	×	×	✓	×	×
Admin Moderation Tools	×	×	×	×	×	×
Real-Time Notifications	×	×	×	×	×	×
Community-Oriented Listing	×	×	×	×	✓	×
Trust Mechanisms	✓	×	×	×	✓	×

**Table 2:** Comparative Feature Analysis of Student Housing Platforms

## 6. Testing

### 6.1 Testing and Validation

Testing was done at different levels to establish the reliability, functionality, and performance of the *RoomMate* application. At the level of unit testing, the individual components were tested separately. At the front end, React Native components like login screens, registration pages, listing cards, and the chatbot UI were tested with the help of tools such as

React to Native Testing Library to ensure proper behavior of inputs, buttons, and navigation. Backend Node.js API endpoints handling user authentication, property management, booking flows, and chatbot interactions were verified using Mocha, Chai, or Jest to ensure the business logic and database operations. Integration testing was conducted to ensure communication among system components. This involved

testing the front and backend interaction via API calls, backend services communicating with the MongoDB database, and confirming the appropriate database operations were triggered for chatbot queries. Tools like textttPostman and textttSupertest were used to mock workflows such as user registration, room search and booking, property listing by landlords, and chatbot queries. End-to-end testing was carried out with tools such as Detox or Appium to mimic end- to-end user scenarios in the mobile app. Test scenarios ranged from tenants registering and re- serving rooms, landlords listing properties, administrators controlling users, and users communicating with the chatbot in order to receive pertinent information. The tests confirmed smooth integration among all the system pieces from user interface through back-end and database. We did usability testing with a small panel of target users, including students, to gather opinions on how easy the application was to use, how clear chatbot responses were, and the general user experience. From this input, iterative changes were implemented in the user interface, conversation flows, and input validation procedures.

Lastly, performance testing was conducted to analyze the system behavior under normal and maximum loads. Backend APIs were stress- tested utilizing tools such as textttApache JMeter or textttArtillery to verify that the database and API could process several concurrent requests effectively and keep acceptable response times. With this multi-layered testing strategy, textitRoomMate was ensured to provide a stable, responsive, and user-friendly housing rental platform with AI chatbot integration support.

## 6.2. System Quality Metric Equation

To measure the quality of the RoomMate application quantitatively, a weighted measure is employed that considers major elements of the testing strategy:

$$Q = \alpha U + \beta I + \gamma E + \delta P + \epsilon C \quad (1)$$

Where:  $Q$  is the overall system quality score;  $U$  is the unit test success rate (e.g., percentage of components passing unit tests);  $I$  is the integration test success rate (e.g., percentage of successful API/backend/frontend interactions);  $E$  is the end-to-end test pass rate (e.g., percentage of complete user journeys executed successfully);  $P$  is the performance efficiency score (e.g., normalized average response time or throughput under load);  $C$  is the chatbot accuracy score (e.g., per- centage of correctly handled user queries); and  $\alpha, \beta, \gamma, \delta, \epsilon$  are weighting factors representing the importance of each component (e.g.,  $\alpha = \beta = \gamma = \delta = \epsilon = 0.2$  for equal weight). This formula offers a

computational formula for estimating system performance, considering both functional and non-functional testing results.

### **6.3. API Design and Functionality**

The API (Application Programming Interface) of the textitRoomMate application serves as a vital link between frontend (React Native), backend (Node.js), and the MongoDB database. It enables communication between these layers by establishing formatted requests and response structures, thus allowing smooth data exchange and system behavior.

The React Native frontend communicates with the backend using RESTful API endpoints. Whenever users make a call like logging in, registering, finding rooms, booking properties, or chatting with the chatbot, these calls are mapped to HTTP requests (e.g., GET, POST, PUT, DELETE). These requests are forwarded to the Node.js backend via certain API routes like `‘/api/auth/login‘`, `‘/api/rooms/search‘`, or `/api/chatbot/query‘`. These endpoints map to a specific function or controller that handles the incoming request.

At the backend level, the Node.js server serves as the processing engine. It will validate input data, implement business logic, and interact with the MongoDB database to undertake needed operations. For example, when a tenant is looking for available rooms within a certain location, the backend gets the query through the API, parses it, and then sends a database query to MongoDB. The obtained data is then returned to the frontend through a formatted JSON response. This way, the frontend interface can easily present updated data without accessing the database directly. API design adheres to a modular pattern, with various collections of endpoints categorized by functionality e.g., authentication, property handling, bookings, user profiles, and chatbot interaction. Modularity enhances scalability and makes debugging and maintenance easier. Additionally, middleware functions within API routes are leveraged to perform tasks such as authentication (e.g., verification of JWT tokens), error management, and sanitization of data, so that communication is secure and stable throughout the layers.

When integrating with chatbots, a specific API endpoint (e.g., `‘/api/chatbot/query ‘`) processes the user messages and initiates corresponding responses depending on the intent of the user and saved data. If the query is about listing rooms,

the backend logic of the chatbot retrieves live data from MongoDB via the Node.js backend and sends relevant responses through the API. This process helps the chatbot provide real-time information without directly exposing database logic to the frontend. In summary, the API layer in *RoomMate* provides secure communication between the client and server sides, allowing the app to provide a smooth and interactive experience to tenants, landlords, and admins.

## 7. Conclusions and Recommendations

The creation, testing, and comparative evaluation of the RoomMate application have yielded some conclusions of note. First, RoomMate successfully addresses the special needs of students in the room-sharing and rental market by synthesizing key housing features within a lean mobile-first application. The role-based design of the platform improves security and usability by offering customized interactions for tenants, landlords, and administrators. In addition, RoomMate provides several groundbreaking features that are largely absent from popular rental websites—like roommate matching, intelligent housing integration, and a strong verification mechanism—providing a more personalized and trustworthy user experience. The comparative analysis decisively illustrates that RoomMate is a more student-friendly and smarter solution, overcoming key limitations in usability, trustworthiness, and affordability present in current solutions. While the software meets its main purposes, it is developed with scalability in mind so that it will be easy to adjust for future improvements and larger implementation throughout university campuses and cities. RoomMate is a purpose-designed solution that fills the gap between the housing problems of students and contemporary technological innovations. With its smart set of features, localized architecture, and secure network, RoomMate provides a solid foundation for revolutionizing student rental lives and enabling secure, convenient, and community-focused housing environments.

## References

- [1] Scott Chacon and Ben Straub. *Pro Git*. Apress, 2nd edition, 2014.
- [2] Facebook. React to native documentation. <https://reactnative.dev>. Accessed: 2025-05-06.
- [3] David F. Ferraiolo, D. Richard Kuhn, and Ramaswamy Chandramouli. *Role-Based Access Control*. Artech House, 2003.
- [4] K. Kannout and M. Iqbal. Hybrid clustering and fp-growth based

- recommendations to address cold start. *International Journal of Machine Learning in Housing*, 2019. Accessed: 2025-05-06.
- [5] S. Kapoor and R. Vij. Trust signals and conversion in peer-to-peer rental platforms. *E- Commerce and Housing Review*, 2017. Accessed: 2025-05-06.
- [6] Kent Beck and others. Manifesto for agile software development. <https://agilemanifesto.org>, 2001. Accessed: 2025-05-06.
- [7] A. Malhotra and R. Fatehpuria. Minimalism and environmental consciousness in rental decisions. *Journal of Sustainable Consumer Behavior*, 2019. Accessed: 2025-05-06.
- [8] Ankit Malhotra and Rohan Fatehpuria. Overcoming skepticism in the peer-to-peer rental market. *Journal of Consumer Research*, 47(2):123–137, 2020.
- [9] Node.js Foundation. Node.js documentation. <https://nodejs.org>. Accessed: 2025-05-06.
- [10] Ritesh Paul. Design and evaluation of renthub: A mobile rental marketplace. *International Journal of Smart Technologies*, 10(4):88–101, 2021.
- [11] Roger S. Pressman. *Software Engineering: A Practitioner’s Approach*. McGraw-Hill Education, 8th edition, 2014.
- [12] S. Rathore and P. Gupta. Rentalux: Optimizing student rental experiences through web-based platforms. *Journal of Housing and Urban Development*, 2020. Accessed: 2025-05-06.
- [13] R. Setty. Roomshare: Simplifying shared rentals for urban students. *Proceedings of the Web and Housing Tech Conference*, 2018. Accessed: 2025-05-06.
- [14] Ian Sommerville. *Software Engineering*. Pearson Education, 9th edition, 2011.
- [15] Karl E. Wiegers and Joy Beatty. *Software Requirements*. Microsoft Press, 3rd edition, 2013.