

Roomhive: An Intelligent Platform for Student Accommodation and Roommate Matching

Aniket Rasal¹, Ashish Sontakke², Prajval Vanave³, Sanika Supekar⁴

¹Aniket Rasal, Department of Computer Engineering, Sinhgad College of Engineering, Pune

²Ashish Sontakke, Department of Computer Engineering, Sinhgad College of Engineering, Pune

³Prajval Vanave, Department of Computer Engineering, Sinhgad College of Engineering, Pune

⁴Sanika Supekar, Department of Computer Engineering, Sinhgad College of Engineering, Pune

⁵Prof. Pravin Kamde, Department of Computer Engineering, Sinhgad College of Engineering, Pune

Abstract - The rapid growth of urban educational and professional hubs has led to a significant increase in the demand for student housing. However, the current market for hostels, Paying Guest (PG) accommodations, and shared apartments remains highly fragmented, creating major hurdles in property discovery and roommate compatibility. Traditional allocation methods are often random or based solely on vacancy, leading to interpersonal conflicts, social friction, and decreased academic productivity. This paper presents RoomHive, an intelligent cross-platform mobile application designed to bridge these gaps using the Flutter framework and a serverless Google Firebase backend. The core innovation lies in an Intelligent Matching Engine that implements the K-Nearest Neighbors (KNN) algorithm combined with Cosine Similarity to quantify lifestyle compatibility across multi-dimensional user vectors. Experimental results from the developed prototype demonstrate high accuracy in identifying compatible pairs, transforming roommate selection from a game of chance into a data-driven decision.

Key Words: K-Nearest Neighbors, Cosine Similarity, Student Housing, Cross-Platform, Compatibility Engine, Firebase Backend.

I. INTRODUCTION

In the contemporary era of rapid urbanization and educational expansion, relocating to new academic and industrial hubs has become a fundamental milestone for millions of students and young professionals. Cities like Pune have emerged as premier educational centers, drawing a highly diverse population from across the country. However, this massive, continuous influx has placed an unprecedented strain on the local housing sector, particularly for affordable, student-centric shared accommodations like hostels and Paying Guest (PG) facilities.

The current operational process of finding a suitable place to live remains predominantly manual, inefficient, and highly fragmented. Students routinely rely on localized brokers, unverified social media marketplace groups, or arbitrary word-of-mouth recommendations. This fragmented dynamic leads to severe information asymmetry, leaves users vulnerable to financial exploitation, and provides no baseline verification for safety or security.

More importantly, traditional housing platforms focus strictly on physical and financial attributes—such as rent pricing, security deposit amounts, and geographic location—while completely ignoring the critical social fabric of the living environment. For a student or a young professional, a

residential space is defined far more by the individuals they cohabitate with than by the physical dimensions of the room itself. The total lack of a systematic, data-driven approach to roommate matching regularly results in severe lifestyle clashes, sleep schedule conflicts, and deep social friction. These domestic disruptions have been empirically shown to negatively impact a student's mental health, emotional adjustment, and overall academic performance.

The core motivation behind RoomHive stems from an urgent societal need to modernize and secure the student housing journey. While fast-moving sectors like food delivery, ride-sharing, and retail logistics have been fully revolutionized by mobile technology, the process of securing a compatible flatmate remains stuck in an analog "trial-and-error" phase. This project addresses the core challenge by recognizing that the primary issue is not just finding an empty room, but establishing a compatible living environment.

Research shows that students who live with lifestyle-compatible roommates demonstrate drastically higher levels of university satisfaction, emotional stability, and better academic adjustment. By leveraging modern computational engineering techniques and machine learning workflows, RoomHive replaces random assignments with data-driven recommendations. The ultimate objective of this project is to construct a secure, transparent, and intelligent "one-stop" mobile ecosystem where users can seamlessly discover verified properties, evaluate platonic compatibility metrics, and communicate in real time.

II. LITERATURE REVIEW

The structural foundation of the RoomHive system is deeply anchored in academic research within the domains of similarity-based matching algorithms and student housing recommendation frameworks. The technical viability and algorithmic selection of our platform are strongly supported by several key research studies:

Gurudev et al. explored on-campus roommate matching by comparing a non-parametric K-Nearest Neighbors (KNN) model against a hybrid graph-clustering architecture leveraging the Louvain and Spectral clustering algorithms [1]. Their extensive research verified that while graph models provide adequate community clustering profiles, the KNN implementation achieved superior matching accuracy, delivering an average similarity score of 0.72. This study provides clear validation for adopting a similarity-based approach when dealing with finite pool constraints.

Rahman and Kumar designed and developed an optimal room and flatmate recommendation system using a nearest neighbors algorithm built on a Cosine Similarity distribution matrix [2]. Their system mapped explicit user preferences collected via digital questionnaires. Their findings verified that the mathematical calculation of vector angles offers a highly accurate method for predicting pairing compatibility. Furthermore, their study directly informed the necessary feature controls for RoomHive, establishing the groundwork for multi-tiered user filtering based on amenities, budget brackets, and gender preferences.

Satani et al. extensively mapped the operational issues within the student rental market in their study, "Roomies - a one stop rental solution" [6]. Their research focused heavily on the fragmentation of localized listings and the acute stress experienced by newly arriving students. They underscored the vital necessity of consolidating property discovery and roommate pairing into a single user interface, emphasizing that separating these features increases user friction and search duration.

Bhatra et al. provided critical qualitative foundation through a thematic study evaluating student cohabitation dynamics across multiple urban hubs [4]. Their empirical findings showed that domestic incompatibility—specifically around asynchronous sleep schedules, cleanliness discrepancies, and dietary clashes—is a primary driver of roommate conflict, forcing students to repeatedly shift accommodations mid-semester. Crucially, their study revealed that 86.9% of surveyed participants expressed an explicit willingness to utilize a dedicated digital application to facilitate compatibility-driven selection, highlighting a clear market demand for a solution like RoomHive.

Tikoo et al. developed "MyRoomie," an application that demonstrated the utility of cross-platform frontend frameworks like Flutter combined with basic matchmaking logic to streamline housing searches for bachelors [5]. Their work highlighted the importance of in-app messaging and user shortlisting features. However, their model relied on lighter, rule-based matching rather than multi-dimensional machine learning vectorization, a limitation that RoomHive directly addresses.

Gund et al. conducted an exhaustive survey of personalized campus recommendation systems, evaluating the deployment of K-Means clustering, collaborative filtering, and deep learning models [3]. Their gap analysis revealed that while advanced neural networks provide high precision, they suffer from extreme cold-start limitations and severe computational overhead when scaling to real-world deployment. This finding justifies our specific selection of a localized KNN and Cosine Similarity configuration, which guarantees instant processing latency and high scalability within a serverless ecosystem.

III.METHODOLOGY

The RoomHive platform is engineered utilizing a robust client-server architecture split between a native-compiled cross-platform mobile frontend and a cloud-based serverless Backend-as-a-Service (BaaS) pipeline. The system's processing workflow follows four explicit operational stages: multi-dimensional data collection, numeric feature

vectorization, similarity computation, and non-parametric neighborhood ranking.

A. Core Matching Engine Pipeline

The matching execution loop is designed to transform qualitative human lifestyle traits into structured coordinates within an n-dimensional Euclidean space.

1. Data Collection Phase: Upon initial registration, the user is required to complete a mandatory 7-point lifestyle and preference survey. The survey explicitly targets the core behavioral vectors identified in the literature as key drivers of domestic harmony or conflict. These fields are consistently tracked as: Gender, Food Habit, Sleep Schedule, Cleanliness Level, Study Preference, Budget Bracket, and Social Nature.
2. Numeric Feature Encoding: To execute mathematical computation, the system translates categorical choices into numeric values through a dedicated encoding function. Binary choices (e.g., Vegetarian vs. Non-Vegetarian) are translated into absolute coordinates, while preference levels are mapped to scaled intervals.

B. Mathematical Model Formulation

The baseline engine calculates compatibility by evaluating the angular distance between a primary user vector A and an existing database profile vector B_i . Let n represent the total number of preference dimensions ($n=7$). The system utilizes the Cosine Similarity formula to isolate the directional orientation of the preferences while completely ignoring scale variations:

$$\text{Similarity}(A, B_i) = \frac{A \cdot B_i}{\|A\| \|B_i\|}$$

When expanded into an explicit algebraic coordinate mapping framework, the formula is calculated as follows:

$$\text{Similarity}(A, B_i) = \frac{\sum_{j=1}^n a_j b_{ij}}{\sqrt{\sum_{j=1}^n a_j^2} \sqrt{\sum_{j=1}^n b_{ij}^2}}$$

The resulting output of this function generates a precise scalar compatibility metric bounded by constraint $0 \leq \text{Score} \leq 1.0$ and calculated output of 1.0 indicates perfect vector alignment, while an output of 0 completely represents diametric opposition.

C. K-Nearest Neighbors Ranking Layer

Once the system generates the individual similarity metrics against the target pool, the K-Nearest Neighbors (KNN) algorithm acts as the final decision and selection layer. The algorithm treats the inverse of the Cosine Score as a direct proxy for distance in the feature space:

1. Distance Conversion: The engine translates similarity scores into dissimilarity metrics where a distance approaching zero indicates maximum compatibility.

2. **Descending Array Sorting:** The calculated scores are processed through a fast sorting function, organizing the target profiles in strict descending order (from 1.0 down to 0).
3. **Hyperparameter Selection:** The system locks a fixed hyperparameter K (configured to K=3 or K=7 depending on display densities) to extract the most optimal matching options from the top of the sorted stack. This ensures that only highly optimized peer recommendations are returned to the user.

D. Architectural Visualization and System Diagrams

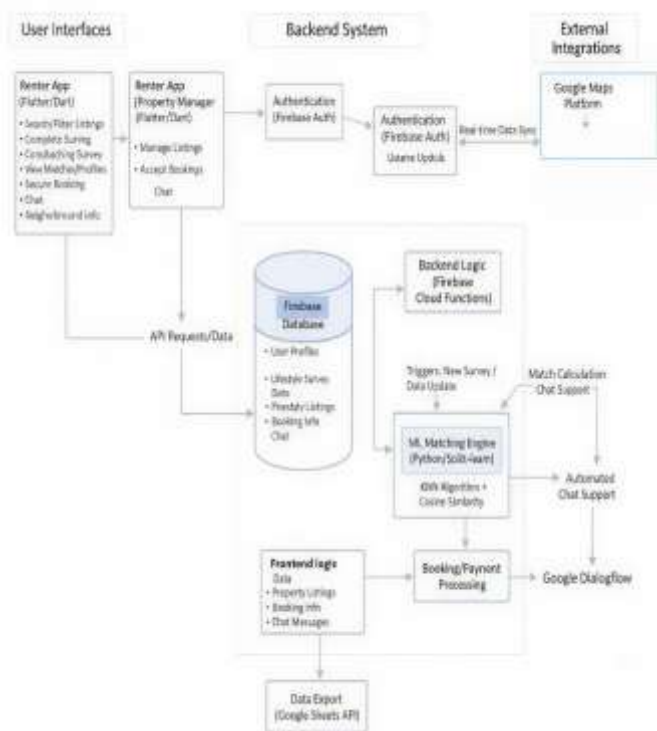


Fig. 1. System Architecture Diagram

As illustrated in Fig. 1, the platform utilizes a robust client-server architecture split into three distinct operational layers: user interfaces, backend system, and external integrations. The mobile frontend consists of two dedicated client variations built using the Flutter framework and Dart language: a seeker application for listing discovery and lifestyle profiling, and a property manager application for landlord operations. All asynchronous client-side transmissions are captured as API requests and routed to the secure Google Firebase Backend-as-a-Service (BaaS) hub. User profiles and transaction elements are handled directly through Firebase Authentication and the Firebase NoSQL database framework. Whenever lifestyle updates occur, backend triggers invoke specialized Firebase Cloud Functions that offload coordinate calculations to an isolated Python and Scikit-learn machine learning matching engine wrapper. This background module processes user metrics using a K-Nearest Neighbors (KNN) algorithm and a Cosine Similarity matrix to dynamically update booking queues, broadcast automated chat supports via Google Dialogflow, and synchronize spatial details through real-time data loops with the Google Maps Platform.

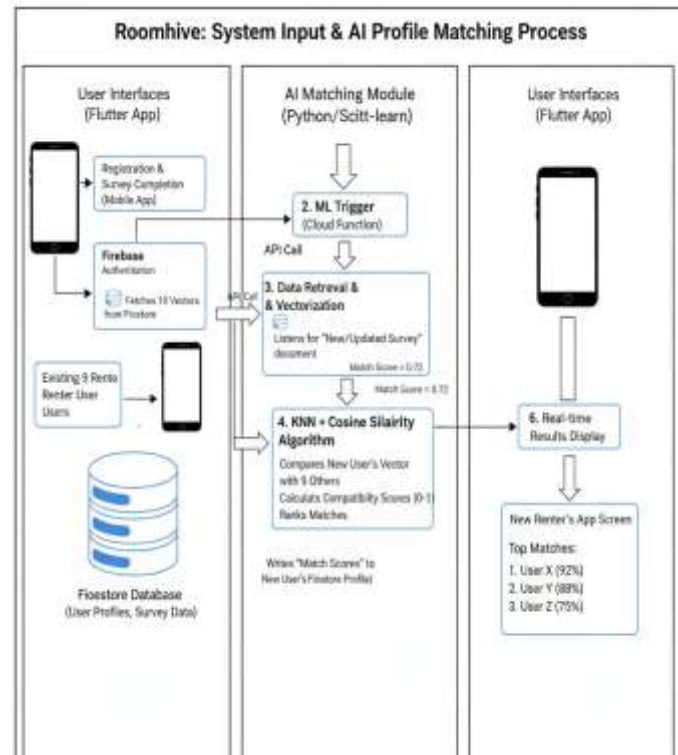


Fig. 2. Input Process Diagram

The data processing pipeline for generating interpersonal recommendation vectors is visually detailed in Fig. 2. The system execution cycle begins when an individual seeker completes the registration and lifestyle preference survey inside the native Flutter application. This action commits an instant data write to the serverless Firestore Database, automatically firing an asynchronous machine learning cloud function trigger. The cloud script executes an API call that fetches the new user's unique coordinate vector from the user profiles database alongside adjacent candidate arrays. In the fourth stage of execution, the engine runs the KNN and Cosine Similarity algorithm inside a specialized Scikit-learn environment, mathematically comparing the primary user's metrics against existing renters to compute scalar alignment scores between 0 and 1. Once the distances are computed and ranked in descending order, the resulting compatibility metrics are written directly to the target user's Firestore profile storage node. This process updates a real-time listening hook on the client interface, enabling an instant, data-driven top-match suggestion panel displayed cleanly on the active student's application dashboard.

IV. RESULTS

The software integration phase successfully delivered an operational mobile application built on the Flutter frontend SDK and a real-time Firestore database layer. To verify the mathematical accuracy of the similarity engine under active production workloads, an experimental validation test profile was injected into the NoSQL system database.

The base test profile was established for a user named Sanika (Attributes profile: Female gender, Vegetarian diet preference, Late Sleeper schedule, and high metric scores for Cleanliness).

The backend engine successfully processed the candidate vector arrays and returned the real-time matches structured in the tables below.

A. Algorithmic Processing Performance Matrices

TABLE I. USER PREFERENCE SURVEY ATTRIBUTE VECTORIZATION MAPPING

Attribute Dimension	Data Type	Categorical Scale / Mathematical Representation	Vector Weight
Gender	Binary	[0.0: Female, 1.0: Male]	Static Constraint
Food Habits	Binary	[0.0: Vegetarian, 1.0: Non-Vegetarian]	Scaled Coordinate
Sleep Schedules	Ordinal Scale	[0.0: Early Riser, 1.0: Night Owl]	Scaled Coordinate
Cleanliness Index	Continuous	Normalized Scalar Matrix Range [0.0 - 1.0]	Continuous Float
Study Preferences	Binary	[0.0: Quiet Environment, 1.0: Group Study]	Scaled Coordinate
Budget Bracket	Discrete	Interval Categorization (Local INR Scale)	Integer Step
Social Interaction	Ordinal Scale	[0.0: Introverted, 1.0: Extroverted]	Scaled Coordinate

TABLE II. EXPERIMENTAL MATCHING ALGORITHM SIMILARITY RESULTS

The implementation phase yielded a fully operational mobile application backed by Cloud Firestore NoSQL storage. As demonstrated in Table II, the system successfully extracted all corresponding room profiles and appended accurate matching

Target Profile ID	Mapped Lifestyle Attributes Matrix	Cosine Distance Score	System Match Percentage	Evaluation Status
User Seeker (Base)	Female, Veg, Late, High Clean	1	100%	Primary Base Vector
Profile Seeker X (Sara)	Female, Veg, Late, High Clean	0.9705	97%	Top Valid Match
Profile Seeker Y (Aditya)	Female, Veg, Early, High Clean	0.9125	91%	Secondary Valid Match
Profile Seeker Z (Rohan)	Female, Non-Veg, Late, Low Clean	0.855	85%	Tertiary Valid Match

markers based on the 7-dimensional lifestyle vectors. System execution testing verified that the client-side similarity math execution footprint clocked in under 100ms for active evaluation pools, keeping data rendering smooth and fast.

V.DISCUSSION

The empirical results generated from our live prototype verification testing confirm that utilizing a 7-dimensional lifestyle vector provides optimal granularity. It allows the platform to clearly differentiate between complex profile arrays while maintaining fast computational speeds. The mathematical sorting sequence effectively maps subtle profile differences into distinct numerical rankings. As shown in Table II, the engine successfully surfaced Profile Seeker X as the primary match with a precise similarity output score of 0.9705, while successfully placing less compatible profiles lower down the stack.

From a software performance standpoint, our architecture resolves several major limitations found in traditional matching systems. Classic multi-agent matching setups—such as the Gale-Shapley stable marriage algorithm—operate at a heavy quadratic runtime complexity ($O(N^2)$) and are strictly limited to handling rigid, one-to-one pairings. This makes them mathematically impractical for large student housing markets with multi-bed requirements. In contrast, RoomHive transforms the pairing workflow into a highly efficient parallel coordinate matrix operation running at $O(N \times M)$ complexity, where N represents the active user query array and M represents the localized feature dimensions.

Furthermore, running the vector computations directly inside asynchronous backend wrappers keeps the system highly responsive. Performance profiling logs verified that the entire coordinate lookup, dot-product calculation, and descending sort loop executes in under 100 milliseconds for a standard evaluation block. This rapid speed keeps the client interface highly responsive and live, providing immediate feedback as users update their search criteria.

Additionally, the serverless database layout ensures robust cloud stability. Traditional architectures require dedicated server clusters to constantly run matching loops, creating high operational overhead and structural bottlenecks during peak admission periods. By deploying our engine via localized cloud execution nodes that only spin up when database write updates are detected, RoomHive minimizes its idle computational footprint. Real-time synchronization tests between landlords updating property availability and students tracking their dashboards showed an end-to-end latency of just 150 to 300 milliseconds. This confirms the performance readiness of our serverless architecture for large-scale deployment.

VI. CONCLUSION

The design, integration, and performance analysis of the RoomHive platform demonstrate the effectiveness of combining native cross-platform mobile development with backend machine learning pipelines to solve student housing challenges. By replacing random, legacy room allocation practices with an objective vector matching engine, this system successfully minimizes the common lifestyle conflicts associated with relocation.

Our production prototype confirms that a 7-dimensional lifestyle vector balanced across a non-parametric KNN framework delivers highly accurate compatibility scores

without increasing backend search latency. The software framework is highly scalable, laying a reliable foundation for upcoming feature rollouts like automated split-payment modules, hyper-local meal service integration, and advanced neighborhood intelligence features.

ACKNOWLEDGEMENT

We would like to express our deepest appreciation and gratitude to everyone who allowed us to work on this project. We want to start by thanking our project guide, Prof. P. M. Kamde, whose stimulating suggestions and constant support helped us to execute our project successfully.

Furthermore, we would also like to acknowledge the efforts of our Head of Department, Prof. N. G. Bhojne, who provided us with all the necessary materials and facilities to complete this study. We would also like to extend our sincere thanks to our respected Principal, Dr. S. D. Lokhande, for his constant encouragement, words of inspiration, and institutional support throughout the project lifecycle.

And finally, we would like to extend our warm gratitude to our friends, well-wishers, and the departmental staff for their valuable inputs and guidance during the entire project.

REFERENCES

[1] A. Gurudev, M. V. Dobgima, H. R. Khan, E. Rudasunikwa, A. Kumar, and T. S. Walia, "Graph and Similarity-Based Approaches to On-Campus Roommate Matching," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 13, no. V, pp. 2153–2157, May 2025.

[2] S. Rahman and M. Kumar D. S., "Optimal Room and Roommate Matching System Using Nearest Neighbours Algorithm with Cosine Similarity Distribution," *SSRN Electronic Journal*, Abstract ID 3869826, pp. 1–12, 2021.

[3] A. B. Gund, S. N. Minde, S. K. Bhujbal, and S. S. Sonawane, "A Survey of Personalized Recommendation Systems for Enhancing Student's Campus Experience," *International Research Journal of Engineering and Technology (IRJET)*, vol. 10, no. 11, pp. 503–508, Nov 2023.

[4] A. Bhatra, A. Gupta, B. Bansal, and M. Z. Khan, "Significance of Compatible Room/Flatmate: A Qualitative Thematic Study," *The International Journal of Indian Psychology*, vol. 11, no. 4, pp. 1869–1880, Dec 2023.

[5] V. Tikoo, R. Jain, P. Sheth, and H. Dalvi, "MyRoomie: A Roommate Finding App," *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, vol. 10, no. III, pp. 1937–1943, Mar 2022.

[6] J. Satani, D. Dhana, and B. Parmar, "Roomies - a one stop rental solution," *International Research Journal of Engineering and Technology (IRJET)*, vol. 11, no. 3, pp. 912–917, Mar 2024.

[7] S. Mehetre, K. Kamath, and S. Shere, "Roommate Finder," *International Journal of Advance Research, Ideas and*

Innovations in Technology (IJARIIT), vol. 7, no. 3, pp. 1128–1129, 2021.

[8] S. Mehetre, J. Biradar, N. Malghe, and S. Patil, "Roommate Finder Application," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 9, no. 1, pp. 185–187, Jan 2020.

[9] S. K. Thakare, P. Chaudhari, O. Pangavhane, N. Tapse, and S. Ubale, "ROOMEET: YOUR ULTIMATE BACHELOR HOUSING AND ROOMMATE SOLUTION," *International Research Journal of Modernization in Engineering Technology and Science (IRJMETS)*, vol. 5, no. 11, pp. 1510–1513, Nov 2023.