E-ISSN: 2664-8784
P-ISSN: 2664-8776
Impact Factor: RJIF 8.26
IJRE 2026; 8(1): 79-83

www.engineeringpaper.net
Received: 08-01-2026
Accepted: 14-02-2026

Dhanshri Khande
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Achal Rokade
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Sejal Ingle
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Dhanashri Panchbudhe
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Bhagyashri Koturwar
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Dr. VS Gangwani
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

Corresponding Author:
Dhanshri Khande
Department of Information Technology, Sant Gadge Baba Amravati University, Amravati, Maharashtra, India

PGLIFE: A full-stack web-based PG accommodation management system

Dhanshri Khande, Achal Rokade, Sejal Ingle, Dhanashri Panchbudhe, Bhagyashri Koturwar and VS Gangwani

DOI: https://www.doi.org/10.33545/26648776.2026.v8.i1a.168

Abstract
Student housing in urban academic environments remains a persistent logistical challenge, often linked to manual search processes, unverified listings, and the absence of structured booking and communication mechanisms. A web-based Paying Guest accommodation management system was developed to address these limitations. The system supports three distinct roles: students, PG owners, and administrators, each operating within a dedicated workflow. Students can discover PGs by city, apply filters, mark favorites, book rooms, cancel bookings, chat with owners, submit reviews, and report issues. Owners can register, submit PG listings for administrative review, and communicate with prospective tenants. Administrators verify and approve listings before publication, ensuring platform reliability. The backend is implemented in PHP with a MySQL relational database, while the frontend uses HTML, CSS, Bootstrap, and JavaScript. Email notifications are delivered through PHPMailer integrated with Gmail SMTP. A rule-based natural language search assistant is included to support budget- and preference-based PG discovery. The system demonstrated consistent end-to-end functionality across all major workflows including booking, cancellation with refund computation, multi-role authentication, and moderation. This platform provides a practical foundation for digitizing the PG accommodation ecosystem in student-dense urban settings.

Keywords: PG accommodation management, web-based booking system, multi-role platform, PHP MySQL, student housing, admin moderation, natural language search

1. Introduction
Urban educational centers across India host millions of students who rely on Paying Guest (PG) accommodations as their primary housing option. Despite the scale of this demand, the process of searching, verifying, and securing PG accommodations remains largely informal. Students typically depend on personal referrals, unverified online listings, or direct property visits to evaluate available options, which often leads to significant time consumption and information asymmetry between students and property owners [1].
computer-based platforms have increasingly been adopted in hospitality and rental markets, demonstrating that the digitization of booking and communication workflows can significantly improve both efficiency and user experience [2]. At the same time, full-stack web application frameworks have evolved to a stage where multi-role platforms with real-time interaction, modular authentication, and database-backed dynamic content can be developed at a modest infrastructure cost [3]. Previous studies on accommodation management systems have explored property listing portals, hotel booking engines, and hostel management applications; however, only a limited number of these systems address the specific lifecycle requirements of the PG sector, such as owner verification, admin moderation, and post-booking communication [4].
This work presents PGLIFE, a web-based PG accommodation management system designed to bridge these practical gaps. Rather than serving as a simple listing directory, the system manages the complete accommodation lifecycle, including discovery, shortlisting, booking, cancellation, reviews, issue reporting, and owner-student communication. A three-role architecture clearly separates student, owner, and administrator access, while an admin-moderated listing approval workflow ensures that only verified properties are visible to prospective tenants.
This architecture helps maintain reliability and trust across all platform interactions while remaining easy to use for non-technical users.

By separating responsibilities across user roles and implementing modular PHP components for each workflow, the system minimizes functional overlap and improves maintainability. The addition of a rule-based natural language search assistant further enhances accessibility for students who may not be familiar with structured filtering interfaces. The platform is designed as a deployable prototype suitable for academic use and early-stage real-world adoption, with clearly defined pathways for future enhancement.

2. Materials and Methods

2.1 System Overview

PGLIFE is a full-stack web application developed using a monolithic server-side rendering architecture, designed to provide a seamless and centralized platform for PG accommodation management. The system integrates all major functionalities-user interaction, business logic, and database communication-within a unified server-side framework, allowing efficient processing and simplified deployment. This architectural choice was made to ensure ease of maintenance, faster development, and better suitability for academic and prototype-level deployment environments.

The platform supports three primary user roles: students, PG owners, and administrators. Students use the system to search for accommodations, compare available options, shortlist suitable properties, and complete booking-related activities. PG owners are provided with dedicated modules to list their properties, update room details, manage availability, respond to student inquiries, and monitor booking requests. Administrators act as the central moderation authority, reviewing newly submitted listings, verifying property details, handling issue reports, and ensuring that platform policies are consistently enforced.

All three roles interact through a unified web interface that is designed to be simple, responsive, and easy to navigate, even for users with limited technical knowledge. The platform relies on a shared relational database to maintain consistency across user records, property listings, booking information, reviews, and communication logs. This centralized data model helps preserve data integrity while enabling smooth interaction between different workflows.

Since the application follows a browser-based approach, it is accessible through any standard web browser without requiring any client-side software installation. This improves usability and ensures that students and property owners can access the platform conveniently from desktops, laptops, or mobile devices. The overall design emphasizes reliability, role-based separation, and real-world usability, making it suitable as a deployable prototype for practical PG accommodation management scenarios.

Table 1: Technology Stack Summary

Component	Technology
Frontend	HTML, CSS, Bootstrap, JavaScript, jQuery
Backend	PHP
Database	MySQL
Email Service	PHPMailer with Gmail SMTP
Session Management	PHP Sessions
Async Interactions	fetch () and AJAX
Search Assistant	Rule-based NLP (PHP)

2.2 System Architecture: The overall system architecture of PGLIFE follows a structured three-layer design, comprising the presentation layer, business logic layer, and data layer, as illustrated in Fig. 1. This layered approach was intentionally adopted to ensure modularity, maintainability, and a clear separation of responsibilities among system components. By organizing the platform into independent layers, the architecture reduces code complexity and supports easier debugging, future upgrades, and feature expansion.

The presentation layer is responsible for all user-facing interactions and visual rendering of the platform. It consists primarily of PHP pages that dynamically generate HTML content based on user requests and database responses. CSS files are used to manage layout styling, typography, and interface consistency, while the Bootstrap framework provides responsive design support to ensure smooth accessibility across desktops, tablets, and mobile devices. JavaScript enhances the interactivity of this layer by handling modal windows, asynchronous chat refresh mechanisms, client-side filtering operations, form validation, and API calls to backend endpoints. These features collectively improve usability and create a more interactive and user-friendly experience.

The business logic layer is implemented entirely in PHP and acts as the core processing engine of the application. This layer contains separate modular scripts for each major workflow, including login and signup processing, property listing submission, booking creation and cancellation, review and issue reporting, owner-side property management, and administrator approval operations. The modular design of these scripts minimizes functional overlap and makes the application easier to maintain. In addition, PHP session management is used to preserve authenticated user states across multiple page requests, enabling secure and role-specific access for students, PG owners, and administrators.

The data layer is powered by a MySQL relational database that stores all persistent application data in a structured and well-organized format. This layer serves as the foundation of the entire PGLIFE platform, ensuring that all user actions and system events are reliably captured and preserved. The database maintains essential entities such as user account details, PG property listings, room availability information, booking transactions, cancellation records, user reviews, reported issues, chat messages, waiting list entries, and administrator activity logs. By organizing these entities into normalized relational tables, the system minimizes data redundancy while preserving logical relationships across multiple workflows.

A relational database model was selected because it provides strong support for data consistency, integrity, and efficient query execution, all of which are critical for an accommodation management platform. Relationships between tables-such as users and bookings, owners and listings, or properties and reviews-are maintained through primary and foreign key constraints. This relationship mapping allows the platform to retrieve connected data efficiently, such as displaying all bookings made by a specific student, showing reviews associated with a property, or tracing ownership details for a submitted listing. Such structured associations are particularly important in multi-role systems where multiple entities interact within the same transaction lifecycle.

The database design also supports transactional reliability,

especially for sensitive workflows such as booking creation, cancellation processing, and waiting list updates. For example, when a student books a room, the system can simultaneously create a booking record, reduce room availability, and update the waiting list status if necessary. Using a relational model helps ensure that these dependent operations remain synchronized and prevents inconsistent states within the platform.

To maintain uniformity in database operations, connectivity is centralized through a shared configuration file that manages connection parameters and reusable query access settings. This centralized connectivity layer ensures consistent communication between PHP modules and the database while reducing repetitive connection code throughout the application. As a result, maintenance becomes simpler, debugging database-related issues is more efficient, and updates to connection credentials or database settings can be performed from a single location.

From a scalability perspective, the structured schema design also allows future expansion of the platform with minimal disruption. Additional modules such as payment integration, recommendation systems, occupancy analytics, or AI-based pricing optimization can be introduced by extending the existing relational structure. This makes the data layer not only reliable for the current prototype but also flexible enough to support future production-scale enhancements.

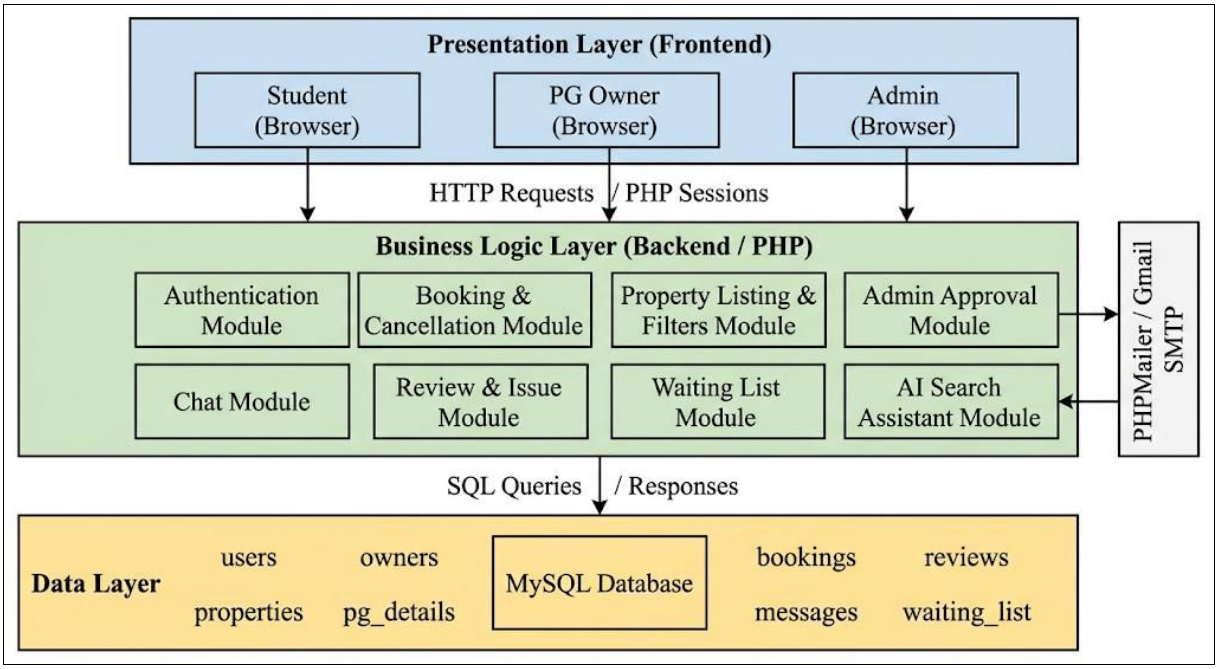

Fig 1: Three-layer system architecture of PGLIFE showing presentation, business logic, and data layers with PHPMailer integration for external notifications

2.3 Database Design

The system operates on user-generated data rather than a pre-existing dataset. Property records are submitted by registered owners and stored in a staging table pending administrative review. Upon approval, listings are migrated to the main properties table and made discoverable to students. City data and initial property seed records are pre-populated during deployment.

The database schema encompasses eighteen primary tables supporting all platform workflows. Core relational constraints include one-to-many relationships between cities and properties, properties and amenities, users and bookings, and chats and messages.

Table 2: Primary Database Tables and Purposes

Table	Purpose
Users	Student account records
Owners	PG owner account records
Properties	Approved and active PG listings
Pg_Details	Owner-submitted listings pending admin approval
Student_Registration	Booking records and tenant information
Reviews	Property reviews submitted by booked students
Issues	Issue reports submitted by booked students
Chats	Chat session records linking student, owner, and property
Messages	Individual chat messages per session
Waiting_List	Waitlist registrations for fully occupied properties
App_Reviews	Platform-level feedback from users

2.4 Module Design

The system is organized into seventeen functional modules, each implemented as a discrete set of PHP pages and API endpoints.

- **Home and Search Module:** The entry point provides city-based search input and quick links to major cities.

Students initiate the discovery workflow from this page.

- **Property Listing Module:** Filtered property listings are displayed based on city, area, gender preference, and rent range. Filter conditions are applied server-side through dynamically constructed SQL queries.
- **Property Detail Module:** A consolidated property profile page presents images, amenities, ratings, student reviews, booking controls, owner chat initiation, issue reporting, and waiting list options in a single interface.
- **Authentication Module:** Separate registration and login workflows are implemented for students and owners. Student passwords are hashed using PHP's native password_hash() function and verified with password_verify(). Owner accounts follow a parallel authentication flow. Admin login is managed through a dedicated interface.
- **Booking Module:** The booking workflow collects student personal, academic, and payment information through a structured form. A unique booking identifier is generated in the format PG + year + random digits. Booking records are stored in the student_registration table and confirmations are dispatched by email.
- **Booking Cancellation and Refund Module:** Students may cancel active bookings through a dedicated interface. Refund computation applies a fixed deduction of ₹200, with the remainder refunded. Cancellation confirmation is sent by email.
- **Review and Issue Reporting Module:** Review submission is restricted to students with confirmed bookings for the target property. Reviews capture five rating dimensions: overall, cleanliness, food, safety, and textual feedback. Issue reporting follows the same eligibility restriction.
- **Student-Owner Chat Module:** A messaging interface allows students to initiate conversations with property owners directly from the listing page. Chat records are created on first contact, messages are stored in the messages table, and the chat window auto-refreshes every second using a polling mechanism. Unread message counts are surfaced on student and owner dashboards. The module also incorporates a rule-based auto-response layer that generates owner-style replies based on keyword matching for common queries regarding location, rent, and facilities.
- **Owner Module:** Owners access a separate dashboard to manage their profile, submit new PG listings, view listing approval status, and respond to student inquiries.
- **Admin Approval Module:** Submitted PG listings are held in a pending state until reviewed by an administrator. Upon approval, listing data is transferred from the staging table to the main properties table and amenity associations are replicated. Rejected listings are marked accordingly and not published.
- **Waiting List Module:** When a property reaches full capacity, students may register on a waiting list. Email notifications are dispatched to waitlisted students when rooms become available.
- **AI Search Assistant Module:** A lightweight rule-based assistant embedded in the site footer accepts natural language input such as "PG for girls under 8000 in Pune." The assistant extracts city, budget, and gender parameters, constructs a SQL query, and returns the best matching result. This component functions as a structured query generator rather than a generative AI model.
- **App Feedback Module:** A platform-level review mechanism allows users to submit feedback on the application itself, separate from property-specific reviews.

2.5 Notification Handling

Email notifications are dispatched at several lifecycle points: student registration confirmation, booking confirmation, booking cancellation with refund details, and waiting list availability alerts. All email delivery is handled through PHPMailer configured with Gmail SMTP. Notification triggers are embedded within the respective PHP processing scripts for each workflow.

2.6 Security Mechanisms

Password storage uses PHP's native hashing functions for all student and owner accounts. SQL prepared statements are applied in selected modules to mitigate injection risks. Session-based authentication restricts access to role-specific pages. Booking eligibility checks are enforced before review and issue submission to prevent unauthorized entries. Duplicate checks prevent multiple bookings or repeated reviews from the same user for the same property.

3. Results

The system was evaluated across all primary workflows using a local development environment with MySQL and PHP. All three user roles were tested through their complete interaction sequences. student search and filtering correctly returned property results based on city, rent range, gender filter, and area. The booking workflow successfully created records, generated unique identifiers, and dispatched confirmation emails. Cancellation correctly applied the fixed deduction and issued refund notification emails. Chat message delivery and auto-refresh operated as expected, with unread counts updating on dashboards following new messages.

The admin approval workflow correctly migrated approved listings from the staging table to the active properties table, with amenity associations replicated accurately. The waiting list notification mechanism dispatched emails to registered students upon room availability trigger. Review and issue submissions were correctly restricted to users with confirmed bookings, preventing unauthorized entries. Duplicate review checks functioned as intended, blocking repeated submissions from the same account for the same property.

The rule-based search assistant correctly parsed structured natural language inputs and returned matching property results based on extracted parameters. Responses were consistent across repeated queries with the same input structure.

4. Discussion

The results suggest that a modular, role-separated PHP architecture is practically effective for managing the complexity of a multi-stakeholder accommodation platform. By assigning each workflow to a dedicated set of PHP scripts and maintaining clear session-based role separation, the system avoids the functional overlap that commonly arises in monolithic portal designs. The admin moderation layer adds a meaningful layer of trust enforcement,

distinguishing PGLIFE from simple listing directories where unverified content is published immediately [3].

The dual-layer booking model, comprising an active booking record and a separate cancellation workflow with refund logic, reflects the real-world requirements of PG accommodation agreements more accurately than systems that treat booking as a single atomic transaction. The fixed deduction approach is a simplified implementation of cancellation policy enforcement; more sophisticated time-based deduction schedules would better represent commercial PG practices in a production setting.

The student-owner chat module introduces meaningful interaction capability, though the current polling-based refresh mechanism has inherent limitations in responsiveness compared to WebSocket-based real-time messaging [2]. The inclusion of a rule-based auto-response layer within the same chat pipeline is a pragmatic feature for owners who may not respond immediately, though it should be clearly disclosed to users to avoid confusion regarding the source of replies.

The rule-based natural language search assistant provides a useful entry point for students unfamiliar with structured filter interfaces. However, it performs parameter extraction and SQL query construction rather than semantic understanding, and its performance degrades outside the patterns it was designed to match. Extending it to handle ambiguous or compositional queries would require a more robust natural language processing approach.

One limitation worth acknowledging is that several security practices remain incomplete. SQL string interpolation is used in portions of the codebase that have not been updated to use prepared statements. SMTP credentials are embedded in source files rather than environment configuration, which presents a risk in shared or version-controlled deployments. Admin authentication uses plain-text password comparison, bypassing the hashing protections applied to student and owner accounts. These gaps are appropriate to address before any production deployment and represent the primary distinction between the current implementation as a functional academic prototype and a hardened commercial system.

The absence of a real payment gateway is a further constraint. The current implementation simulates payment collection through form fields, which is adequate for demonstrating the booking workflow in an academic context but insufficient for live financial transactions. Integration with a regulated payment service provider would be a prerequisite for any commercial deployment [4].

5. Conclusion

This paper described PGLIFE, a full-stack web-based Paying Guest accommodation management system serving student, owner, and administrator roles within a unified platform. The core architectural contribution is the separation of role-specific workflows through modular PHP components, a shared relational MySQL database, and an admin-moderated listing approval pipeline. This design ensures that only verified properties reach students while providing owners with structured tools for listing management and tenant communication.

The booking and post-booking lifecycle, encompassing confirmation, cancellation, refund computation, reviews, issue reporting, and waiting list management, extends the platform beyond simple directory functionality. Multi-role authentication with password hashing, session management, and booking eligibility enforcement provides a baseline of access control appropriate for a prototype deployment. Email notification integration through PHPMailer supports key lifecycle events across all roles.

The results demonstrate consistent end-to-end functionality across all major workflows under test conditions. The system represents a practical, deployable prototype for automating the PG accommodation ecosystem and provides a structured foundation for further development toward a production-grade multi-site accommodation management platform.

6. Acknowledgments

The authors wish to acknowledge the open-source contributors responsible for the Bootstrap framework, PHPMailer library, and jQuery toolkit used in this implementation. The authors also thank the faculty of the Department of Information Technology, [University Name], [City], India, for their guidance and support during this work. The authors declare no conflict of interest related to this work. No external funding was received for this research.

References

1. Bhatia S, Sharma P, Burman R, Hazela B, Srivastava A. A review on various aspects of PG accommodation portals. Int J Comput Sci Eng. 2019;7(5):422-427.
2. Shklar L, Rosen R. Web application architecture: principles, protocols and practices. 2nd ed. Hoboken: Wiley; 2009.
3. Pressman RS. Software engineering: a practitioner's approach. 8th ed. New York: McGraw-Hill; 2014.
4. Kumar A, Singh R. Design and implementation of an online hostel management system. Int J Adv Res Comput Sci. 2018;9(2):112-117.
5. Podunavac I, Crnjac Milić D, Nenadić K. Proposal for a web portal managing registration for student accommodation in a dormitory. Tehnički glasnik. 2019 Mar 23;13(1):75-80.