Pg Management System

Akanksha C. Tambe[1] Akanksha S. Chavan,[2] Manali R. Pol[3], Latika J. Mahadik[4], Prof. Santosh P. Patange[5]
[1-2-3-4-5]*Shree Santkrupa Institute of Engineering & Technology, Karad*

***Abstract*—In today's fast-paced urban environments, managing Paying Guest (PG) accommodations through manual processes often leads to inefficiencies, errors, and poor tenant experience. This paper proposes the development of a web-based PG Management System using Java and the Spring Boot framework. The system enables PG owners to manage rooms, tenants, rent payments, and complaints efficiently through a centralized platform. It also allows tenants to register, view room details, pay rent, and raise issues. The application uses a layered architecture consisting of the presentation, service, and data access layers, ensuring modularity and ease of maintenance. MySQL serves as the backend database for storing structured data. The proposed system is scalable, user-friendly, and secure, providing a digital transformation to traditional PG operations. It also lays a foundation for future integrations such as online payments and mobile app extensions.**

***Index Terms*—Paying Guest (PG), Java, Spring Boot, PG Management System, Web Application, MySQL, Tenant Management, Room Allocation, Rent Tracking, Complaint Handling, REST API, Full Stack Development.**

I. INTRODUCTION

In today's rapidly urbanizing world, migration to cities for education, employment, and business opportunities has increased significantly. As a result, the demand for temporary residential arrangements such as Paying Guest (PG) accommodations has seen exponential growth. PG facilities provide a cost-effective and convenient alternative to traditional rentals, particularly for students, professionals, and single migrants. Despite their popularity, many PG establishments continue to rely on manual record-keeping methods and traditional communication practices to manage day-to-day operations. These include physical registers for tenant records, paper receipts for rent collection, and verbal or informal complaint handling systems. Such outdated practices are not only inefficient but also prone to data loss, miscommunication, and lack of accountability.

To address these challenges, the development of a digital PG Management System has become essential. A well-designed, centralized software solution can significantly improve operational efficiency, transparency, and user satisfaction. This research paper presents the design and implementation of a PG Management System using Java and the Spring Boot framework, built to automate and digitize the core functionalities of PG operations. The system provides features such as tenant registration, room allocation, rent tracking, complaint management, and role-based access controls for PG owners and tenants.

II. RELATED WORK

In recent years, the demand for digital solutions in the accommodation sector has increased, prompting the development of various systems aimed at managing hostels, rental apartments, and PG accommodations. This section discusses related projects, tools, and research studies that laid the groundwork for the proposed PG Management System.

1 Traditional Management Systems

Historically, most PG owners have relied on manual methods such as maintaining physical registers or using Excel spreadsheets for managing room availability, tenant records, and rent collection. While simple, these approaches are highly error-prone, lack real-time updates, and are not scalable. These systems are also prone to data loss and make it difficult to maintain historical records or generate reports.

2 Hostel and Rental Management Systems

Several digital solutions have been proposed and implemented for hostel and rental property management. For example:

- Hostel Management Systems developed using PHP and MySQL typically offer features like student registration, room allocation, and mess management. However, these are primarily designed for institutional use and may not suit the business needs of private PG owners.
- Rental Property Portals like Housing.com or NestAway provide large-scale real estate services, but they do not offer dedicated backend systems for PG owners to manage individual tenants or track operational issues like complaints or rent defaulters.

3 Academic Research Projects

Academic institutions have proposed student hostel management applications using various technologies. Some of these include:

- A Java-based Hostel Management System by engineering students where tenants can check room status and admins can update room availability. However, many such projects lack scalability, user-role separation, or integration with modern frameworks like Spring Boot.
- Android-based PG Finder Apps, which help users locate PGs in specific areas. These systems often focus on the discovery phase but do not provide backend functionality for PG owners to manage internal operations.

4Web Applications using Spring Boot

Spring Boot has become a popular choice for building RESTful web services due to its ease of configuration, embedded server support, and integration with databases via Spring Data JPA. Projects like Employee Management Systems, Library Management Systems, and E-commerce Backends using Spring Boot and MySQL have demonstrated the framework's robustness and scalability. However, despite the availability of frameworks and tools, there is a noticeable gap in systems specifically tailored for PG Management, which cater to the daily administrative tasks of PG owners and the service-related needs of tenants. Most existing applications are either too generic or lack key modules such as complaint management, rent tracking, or tenant history management.

III. EXISTING ANALYSIS

The existing systems used for managing PG (Paying Guest) accommodations are largely manual or semi-digital in nature. Most small to mid-sized PG owners still rely on traditional approaches such as physical registers, Excel spreadsheets, or informal verbal communication for managing daily operations. These methods, while simple to use, have significant limitations in terms of accuracy, scalability, and efficiency.

1 Manual Record-Keeping

Most PG owners maintain tenant data manually, noting down details such as name, contact number, room number, rent status, and date of entry in registers or notebooks. Room availability, rent collection, and service requests are managed using physical files, which makes data retrieval time-consuming and error-prone.

Limitations:

- Prone to human error and data inconsistency.
- Difficult to maintain historical data and generate reports.
- No data backup in case of physical damage or loss.
- Time-consuming to update or search for records.

2 Excel-Based Systems

Some PG operators use Excel sheets to store tenant and rent information. While this is an improvement over manual registers, it still lacks automation and system intelligence. Excel files are vulnerable to accidental deletions and often lack data validation or security.

Limitations:

- No real-time access or multi-user collaboration.
- Manual updates needed; no automated alerts for rent due or complaints.
- Poor data security and no user authentication.
- Not suitable for growing PG businesses with multiple properties.

3 Third-Party Rental Listing Platforms

Websites like MagicBricks, 99acres, or NestAway allow PG listings, helping owners advertise their rooms. However, these platforms do not offer internal management systems. They only assist in lead generation and cannot help PG owners manage room allocation, track rent payments, or resolve tenant complaints after onboarding.

Limitations:

- Limited to listing and discovery, not post-booking operations.
- No owner dashboard for tenant tracking or rent management.
- Tenants must rely on offline communication after booking.

4 Absence of Centralized Management

Without a centralized digital solution, it becomes difficult for PG owners to manage:

- Multiple properties or branches.
- Service requests and complaints from tenants.
- Automatic rent tracking or report generation.
- Notifications or alerts for tenants (rent reminders, room updates, etc.)

IV. IMPLEMENTATION DETAILS

The PG Management System was implemented using a full-stack approach, with Java and Spring Boot forming the backend foundation, and MySQL used for persistent storage. The goal of the implementation was to develop a robust, scalable, and maintainable application that automates the core activities of managing a Paying Guest accommodation.

1. User Management

- Users can register as either PG Owner or Tenant.
- Login functionality provided using a simple authentication mechanism or Spring Security.
- Role-based access ensures tenants and owners see different dashboards.

2. Room Management

- PG owner can add, update, delete, and view room details.
- Rooms include information such as room number, type (single/double/shared), rent, availability status, and occupant.
- Tenants can view available rooms in a searchable list.

3. Tenant Management

- Tenant registration includes name, contact info, chosen room, and joining date.
- Admin/owner can view tenant profiles, check-in/out dates, and assigned rooms.
- CRUD operations supported for managing tenant records.

4. Rent Management

- Owners can record rent received from each tenant monthly.
- The system tracks due dates and notifies (manually or through logs) about pending payments.
- Rent payment history per tenant can be viewed or exported.

5. Complaint Handling System

- Tenants can raise complaints (e.g., plumbing, electricity).
- Complaints are logged with status: Open, In Progress, Resolved.

PG owner/admin can assign, update, and close issues through their dashboard.

V. METHODOLOGY

The methodology adopted for building the PG Management System follows the Software Development Life Cycle (SDLC), specifically the Agile Model, which supports iterative development, continuous feedback, and incremental progress. The system was developed using Java, Spring Boot, and MySQL, with modular design principles to ensure maintainability, scalability, and security.

1 Requirement Gathering and Analysis

The first phase involved identifying key functional and non-functional requirements by:

- Conducting informal surveys with PG owners and tenants.
- Studying existing manual and Excel-based systems.
- Analyzing pain points such as delayed rent tracking, inefficient complaint resolution, and lack of transparency.

Key Requirements Identified:

- Secure login and user role management (tenant/owner).
- Room availability tracking.
- Tenant onboarding and management.

- Rent collection records and alerts.
- Complaint registration and resolution.

2 System Design

Based on the requirements, a system architecture was designed using:

- UML Diagrams (Use Case, Class, Sequence diagrams).
- Entity-Relationship (ER) Diagrams for database modeling.
- Layered architecture separating the presentation, service, and data access layers.

VI. CONCLUSION

The PG Management System developed in this research effectively addresses the challenges faced by hostel and PG accommodation providers in managing daily operations. By leveraging Java and the Spring Boot framework, the system provides a secure, scalable, and user-friendly platform for automating key functions such as room allocation, tenant registration, payment tracking, grievance handling, and reporting.

This project significantly reduces manual errors and administrative workload, while enhancing transparency and operational efficiency. It offers real-time data management, robust authentication, and a responsive interface suitable for both desktop and mobile platforms. The implementation of RESTful APIs and Spring Security ensures extensibility and data protection, making the system ideal for future integration with mobile apps or third-party platforms.

Through the development and deployment of this system, we demonstrate how modern web technologies can streamline traditional property and hostel management processes. The PG Management System stands as a comprehensive solution that can be adopted by private PG owners, institutions, or rental agencies to digitize and manage their accommodation services effectively.

REFERENCES

[1] Sharma, A., & Verma, R. (2020). Design and Implementation of Hostel Management System Using Web Technologies. International Journal of Computer Applications, 176(19), 25–29. DOI:10.5120/ijca2020920204

[2] Ramesh, S., & Srinivasan, P. (2019). Automation of Student Hostel Management System Using Java and MySQL. International Journal of Recent Technology and Engineering (IJRTE), 8(4), 7637–7641.

[3] Johnson, D. (2018). Spring Boot in Action. Manning Publications. ISBN: 978-1617292545

[4] Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall. ISBN: 978-0132350884

[5] Gupta, M., & Bansal, S. (2021). A Web-Based Application for Hostel Management. International Journal of Advanced Research in Computer Science, 12(1), 35–40.

[6] Oracle. (n.d.). Java Platform, Standard Edition Documentation. Retrieved from: https://docs.oracle.com/javase/

[7] Pivotal Software. (n.d.). Spring Boot Reference Documentation. Retrieved from: https://docs.spring.io/spring-boot/docs/current/reference/html/

[8] Kumar, P., & Sharma, A. (2019). Web Application Development for PG Accommodation Services Using MVC Framework. Journal of Emerging Technologies and Innovative Research, 6(5), 522–526.

[9] Rajput, S., & Jain, V. (2017). Development of Web-Based Hostel Management System. International Journal of Engineering Development and Research (IJEDR), 5(2), 102–106.

[10] Al-Ameen, M. N., & Rahman, M. M. (2020). A Cloud-Based Smart Hostel Management System. International Journal of Advanced Computer Science and Applications (IJACSA), 11(1), 560–567. https://doi.org/10.14569/IJACSA.2020.0110169

[11] Shah, M., & Patel, A. (2019). Development of a Web-Based Hostel Allocation System Using Spring Boot and MySQL. International Journal of Computer Science and Mobile Computing (IJCSMC), 8(3), 132–138.

[12] Mishra, S., & Kumar, R. (2020). Implementation of Student Hostel Management System Using Java Frameworks. International Journal of Scientific & Engineering Research, 11(5), 301–306.