HOSTEL MANAGEMENT SYSTEM

BY

Sujal Sthapit

TU Registration No: 7-2-410-124

TU Symbol No:14866

Prime College

A Project Report Submitted to

Faculty of Management, Tribhuvan University

In partial fulfillment of the requirements for the degree of

Bachelor of Information Management

Khusibu, Kathmandu

November, 2025

STUDENT DECLARATION

This is to certify that I have completed the Project entitled “Hostel Management System” under the guidance of “**Er. Niraj Khadka**” in partial fulfillment of the requirements for the degree of Bachelor of Information Management at the Faculty of Management, Tribhuvan University. This is my original work, and I have not submitted it elsewhere.

Date:

Signature: _________________
Name: Sujal Sthapit
(14866)

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the project entitled “Hostel Management System" is an academic work done by "Sujal Sthapit" submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Management (BIM), Faculty of Management, Tribhuvan University, under my guidance and supervision. To the best of my knowledge, the information presented in the project report by him has not been submitted earlier.

____

Signature of the Supervisor
Name: Er Niraj Khadka
Designation: Supervisor
Date:

APPROVAL SHEET

This is to certify that the project titled “Hostel Management System” submitted by “**Sujal Sthapit**" has been examined and approved. In our opinion, it meets the required scope and quality standards for a project submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Management (BIM).

Approval Panel:

S.No.	**Name**	**Designation**	**Signature**
1	**Er. Niraj Khadka**	**Project Supervisor**	
2	**Mr. Sailesh Karmacharya**	**Research Coordinator**	
3		**Internal Examiner**	
4		**External Examiner**	
5		**External Examiner**	

Date of Defense:
Department: Faculty of Management
Faculty: Bachelor of Information Management (BIM)

ACKNOWLEDGEMENT

I, **Sujal Sthapit**, would like to express my heartfelt gratitude to all those who have contributed to the successful completion of my project, "Hostel Management System." First and foremost, I owe my deepest appreciation to my project supervisor, **Er. Niraj Khadka**, for their constant guidance, valuable suggestions, and encouragement throughout the development of this system. Their expertise and thoughtful feedback have been instrumental in shaping this project both technically and conceptually, and their unwavering support was a source of great strength.

I am grateful to the Head of Department and all the faculty members of the BIM at PRIME COLLEGE for providing a conducive learning environment and the necessary infrastructure that made this work possible.

I would also like to extend my sincere thanks to my friends and classmates for their cooperation and encouragement throughout this work. Whether it was exchanging ideas, testing different features, or simply offering moral support, their involvement has been a great help.

Above all, I am deeply indebted to my family for their endless patience, love, and unwavering belief in me. They have been my source of strength, providing me with the right environment to focus and inspiring me to keep moving forward.

Completing this project has been a valuable learning experience, and I am thankful to everyone who has contributed, directly or indirectly, to its success. The knowledge and skills I have gained will remain a part of me as I continue my professional journey.

Sujal Sthapit

ABSTRACT

The Hostel Management System is a comprehensive web-based application designed to streamline the complex and often manual processes of managing student accommodation in educational institutions. Traditional, paper-based administrative methods are prone to errors and inefficiencies, which this project addresses by providing a centralized, automated platform.

The system is built using PHP for robust server-side scripting and MySQL for secure database management, ensuring optimal data integrity and performance. It is structured to serve three distinct user roles: Students, Hostel Managers, and System Administrators, providing a tailored, intuitive experience for each.

Key features include a secure user registration with email verification, an easy-to-use portal for students to submit applications, and a real-time dashboard for hostel managers to review applications, allocate rooms, and manage residency records. Furthermore, an integrated messaging module fosters clear communication between students and management.

The successful implementation of this system demonstrates a significant improvement over manual methods. It reduces administrative overhead, minimizes allocation conflicts, and enhances transparency, contributing to a robust, scalable, and effective solution for modernizing student accommodation services.

KEYWORDS: Automate, Centralized, Comprehensive, Database, Integrated, PHP, Real-time.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CRUD	Create, Read, Update, Delete
CSS3	Cascading Style Sheets, version 3
ER	Entity-Relationship
FK	Foreign Keys
HTML5	Hyper Text Markup Language, version 5
JS	JavaScript
MVC	Model-View-Controller
MySQL	Structured Query Language variant
PHP	Hypertext Preprocessor
PK	Primary Keys
ROI	Return on Investment - implied in 'high ROI'
SMART	Specific, Measurable, Achievable, Relevant, Time-bound
SQL	Structured Query Language - implied in 'SQL injection'
SMS	Short Message Service
TU	Tribhuvan University
UI	User Interface
UAT	User Acceptance Testing
UX	User Experience
XAMPP	Cross-platform, Apache, MySQL, PHP, Perl - implied in technology stack
XSS	Cross-Site Scripting

CHAPTER I
INTRODUCTION

1.1 Background of the Project

The Hostel Management System is a comprehensive web-based application designed to modernize and streamline the administration of student housing within educational institutions. Historically, managing student residency, room allocation, fee collection, and record-keeping has relied on manual, paper-based processes, which are inherently inefficient, time-consuming, and prone to error. This project was initiated to address the growing need for a digital transformation in student accommodation services. The system's primary target audience includes Students (for application and communication), Hostel Managers (for operational oversight and allocation), and System Administrators (for global configuration). The application is built upon a robust technology stack, utilizing PHP for the backend scripting logic and MySQL for secure and scalable database management. This combination ensures a reliable, efficient, and responsive user experience.

1.2 Problem Statement

The traditional, manual management of hostels creates significant inefficiencies, resulting in delayed application processing, frequent room allocation conflicts, and a lack of real-time visibility into occupancy and student status. This system aims to solve these pain points by replacing disparate paper records with a single, centralized digital platform. Specifically, the automation of application validation and room assignment will drastically cut down on human error and processing time. This shift is essential to reduce administrative overhead, ensure optimal resource utilization, and provide a more transparent and satisfactory service experience for both the administration and the students. The digital platform will also enable instantaneous reporting, allowing management to make data-driven decisions about future resource planning.

1.3 Project Objectives

The primary goal of the Hostel Management System is to deliver a robust and scalable solution that achieves the following SMART objectives:

a) To develop and implement a fully functional, centralized web application using PHP and MySQL within four months, establishing a single source of truth for all student accommodation data.

b) To create and deploy an intuitive dashboard that enables Hostel Managers to review, approve, and allocate rooms to student applicants in real-time upon system deployment.

c) To automate key processes including student application, user registration, and residency status tracking, aiming to reduce administrative workload by at least 50%.

d) To integrate a secure, internal messaging module between students and managers, ensuring clear and documented communication regarding housing issues and updates.

1.4 Review of Related Work and Literature

The E-Based Hostel Management System is a comprehensive web-based application developed to automate manual hostel operations across various domains. It features modules for efficient room allotment, real-time attendance tracking, and digital complaint registration. The system utilizes PHP for backend logic and MySQL for secure database management, aiming to eliminate paper-based record-keeping, which is prone to error and data loss. By centralizing data and providing real-time information, the system enhances security, transparency, and administrative efficiency within the institution. However, the system's reliance on digital infrastructure presents challenges. Successful implementation requires reliable network connectivity, and the system is subject to the risk of technical glitches and maintenance issues, which can disrupt daily operations if not continuously monitored and managed by technical staff (Diyaolu et al., 2024).

The Web-based Hostel Reservation and Location Identification System addresses the inefficiencies students face when searching for suitable accommodation, which traditionally involves numerous physical visits and manual calls. The system acts as a complete web-based solution that supports scalable online reservation capabilities and includes location identification features. It allows students to remotely browse, compare different hostel options, view available services (such as food, internet, and laundry), and perform online bookings, effectively eliminating the need for physical visits. Key functionalities include streamlined information management and acceptance of online applications, improving speed and efficiency for users (Aqeel et al., 2024). A noted limitation is the potential scope constraint; while initially intended for local or single-city use, scaling the system nationwide requires significant infrastructure investment and widespread adoption by all hostel owners to achieve maximum effectiveness.

The Web-Based Booking System for Dormitories and Mess Services focuses on deeply integrating two traditionally separate functions: room assignment and meal service management. This system simplifies the booking process for both dormitory rooms and associated mess services, significantly increasing data accuracy and reducing the administrative burden. Features include real-time room availability, automated mess scheduling, and centralized data management using web technologies. By automating processes such as tracking mess attendance and calculating dining fees, the system enhances transparency in billing and ensures that administrative staff can focus on service delivery rather than tedious calculations (Darwin et al., 2024). Despite these benefits, a primary challenge remains the requirement for initial staff training to effectively operate the system, and ensuring seamless data integration between the accommodation and mess modules to prevent inconsistencies in billing and record keeping.

1.5 Development Methodology

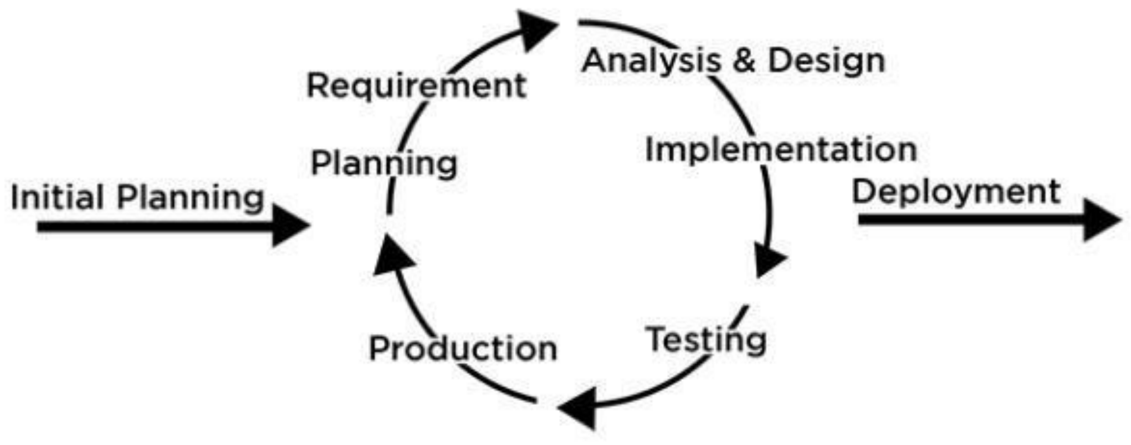

Figure 1.1: Iterative and Incremental Model
Source:(https://www.researchgate.net/figure/terative-Incremental-Model)

The Hostel Management System was developed using the Iterative and Incremental Model, which allowed for continuous refinement and adaptation. The project was broken down into five main phases:

a) **Requirement Analysis and Planning**

Defined the project scope and core functional needs for Students, Managers, and Admins, establishing the application workflow.

b) System Design

Created the architectural design, including the MySQL relational database and a modular PHP application structure (utilizing shared components like config.inc.php).

c) Implementation and Incremental Development

The System was built in four key increments:

i. Core User Foundation (registration/authentication).
ii. Hostel/Room Management (admin interface).
iii. Application/Allocation Workflow (student application/manager allocation).
iv. Communication and Profile Management.

d) Testing and Integration

Performed unit testing on each increment, followed by integration testing to ensure smooth and correct interaction between all modules.

e) Deployment and Maintenance

Deployed the final application on an Apache server via XAMPP. Maintenance includes running necessary scripts and implementing future enhancements.

1.6 Project Scope and Limitations

1.6.1 Scope

The system is designed to be a comprehensive platform for managing hostel accommodations, encompassing the following core features:

a) **Multi-Role Access:** The platform provides distinct interfaces and defined functionalities tailored for three primary user roles: Student, Hostel Manager, and Administrator.
b) **Secure User Authentication:** The system ensures secure registration and login processes for all users, including mandatory email verification for students.
c) **Centralized Core Management:** Administrators are provided with tools to add, manage, and maintain centralized records for all hostels and individual rooms within the system.
d) **Streamlined Application Workflow:** The platform facilitates the entire student application process, from submission by the student to the manual room allocation (reviewing, approving, or rejecting applications) by Hostel Managers.
e) **Integrated Communication:** An internal messaging module is included to support direct and documented communication between students and their assigned managers regarding housing matters.

1.6.2 Limitations

While the system effectively automates core management tasks, it currently operates with the following design limitations:

a) **Exclusion of Online Payment:** Financial transactions, such as the collection of fees and deposits, must be handled externally, as the system does not include an integrated payment gateway.
b) **Manual Room Allocation:** The room assignment process lacks automation; it is performed entirely manually by managers and does not support features like student room selection or preference matching.
c) **Basic Reporting Capabilities:** The platform has limited built-in reporting functionality and does not offer advanced analytics, such as automated occupancy rates or detailed financial summaries.
d) **Absence of Service Requests:** The system does not support the submission of maintenance requests or formal complaint lodging by residents.
e) **Limited Notifications:** Notifications are restricted to internal messaging and initial email communications, lacking support for real-time alerts or SMS notifications.

CHAPTER II
SYSTEM DEVELOPMENT PROCESS

2.1 Analysis

The analysis phase is the foundation of the system development process, focusing on gathering and examining requirements to ensure the final product meets its intended goals. The process focused on understanding existing manual processes and the specific needs of the three primary user groups: Students, Hostel Managers, and System Administrators.

2.1.1 Requirements

Table 2.1: Functional Requirements

FR	Functional Requirements	Descriptions
FR.1	User Authentication & Access	Provide secure, role-based login/registration for all users. Must include email verification for students.
FR.2	Manager Account Administration	Allow Administrators to manage (add, edit, remove) Manager accounts.
FR.3	Student Application Workflow	Enable students to submit detailed applications, view their application status, and check their allocated room details.
FR.4	Application Review & Decision	Allow managers to view, approve, or reject applications.
FR.5	Room Allocation & Monitoring	Allow managers to manually allocate specific rooms and monitor real-time room occupancy and student details for their hostel.
FR.6	Hostel Details Management	Grant administrators the ability to manage (add, edit, remove) all hostel details (e.g., room information, capacity).
FR.7	Integrated Communication	Include an integrated messaging feature for direct communication between students and their managers.

Table 2.2 Non Functional Requirement

NFR	Non-Functional Requirements	Descriptions
NFR.1	Application Security	Ensure robust protection against common web vulnerabilities (SQL injection, XSS).
NFR.2	User Data Protection	All passwords must be securely hashed, and user sessions must be managed securely.
NFR.3	System Availability	The system must maintain 24/7 availability.
NFR.4	Data Integrity	Data consistency and integrity must be ensured using database constraints (e.g., foreign keys).
NFR.5	User Interface (UI) Usability	The UI must be intuitive, clean, and provide clear, unambiguous user feedback.
NFR.6	Browser Compatibility	The application must be compatible and fully functional across major modern browsers (Chrome, Firefox, Edge).
NFR.7	Page Loading Performance	Optimize database queries and front-end assets to achieve target page loading times of 3-5 seconds.
NFR.8	System Scalability	The underlying system architecture must be scalable to accommodate future increases in user load and data volume without performance degradation.

2.2 Feasibility Study

A feasibility study was conducted to assess the viability of the project from technical, economic, and operational perspectives.

2.2.1 Technical Feasibility

The project is technically feasible as it relies on mature, well-supported technologies (PHP, MySQL, HTML/CSS/JS) and open-source resources, requiring only standard development expertise. These technologies have robust communities, extensive documentation, and readily available talent, which mitigates the risk of encountering unsolvable technical roadblocks. Furthermore, utilizing this established stack ensures the final system will be inherently stable, scalable, and easy to maintain over its lifecycle.

2.2.2 Economical Feasibility

The project is economically feasible. Costs are low (due to open-source software and minimal hardware), and benefits are high, including a significant reduction in administrative time, paper costs, and improved efficiency and service quality (high ROI). The initial investment is primarily focused on development and will be quickly offset by the continuous savings from process automation and the elimination of physical documentation. This strong cost-benefit ratio makes the Hostel Management

System a financially sound investment with a high probability of delivering long-term fiscal value.

2.2.3 Operational Feasibility

The project is operationally feasible due to high predicted user acceptance (simplifies student application, automates manager tasks), easy integration with existing workflows, and minimal training requirements. By automating tedious, repetitive tasks, the system directly addresses pain points experienced by both staff and students, ensuring a quick and positive adoption rate. The intuitive design and familiar web interface will also allow users to transition to the new system quickly, minimizing disruption to daily operations.

2.2.4. Legal Feasibility

Legal feasibility ensures that the system complies with applicable laws, regulations, and industry standards. This includes data protection laws, intellectual property rights, online transaction regulations, and other legal requirements relevant to the system. A system is legally feasible if it does not violate any legal or regulatory obligations.

2.2.4 Schedule Feasibility

To ensure the timely and systematic completion of the Hostel Management System, the development process is organized into a series of scheduled phases. Each phase is focused on specific, measurable tasks, including detailed requirement analysis, secure system design, full-stack coding, rigorous testing, and comprehensive documentation. The use of a structured phased approach, such as a modified Waterfall or Agile methodology, allows for close monitoring of progress against deadlines and facilitates early identification of potential scheduling delays. This systematic planning ensures that the project remains on track and is delivered within the expected timeframe.

Table 2.3: Schedule Table

Activities	Start Date	End Date	Days
Sprint 1: Planning & Setup	June 9, 2025	June 20, 2025	11
Sprint 2: Requirements & Design	June 23, 2025	July 11, 2025	18
Sprint 3: Development - Core Features	July 14, 2025	September 2, 2025	50
Sprint 4: Development - Admin & Integration	September 3, 2025	October 22, 2025	49
Sprint 5: Development - Communication & Refinement	October 23, 2025	October 28, 2025	5

Sprint 6: Quality Assurance & Testing	October 29, 2025	November 3, 2025	5
Sprint 7: Final Documentation & Deployment	November 4, 2025	November 6, 2025	2
Project Completion	November 6, 2025	-	**140**

Gantt Chart

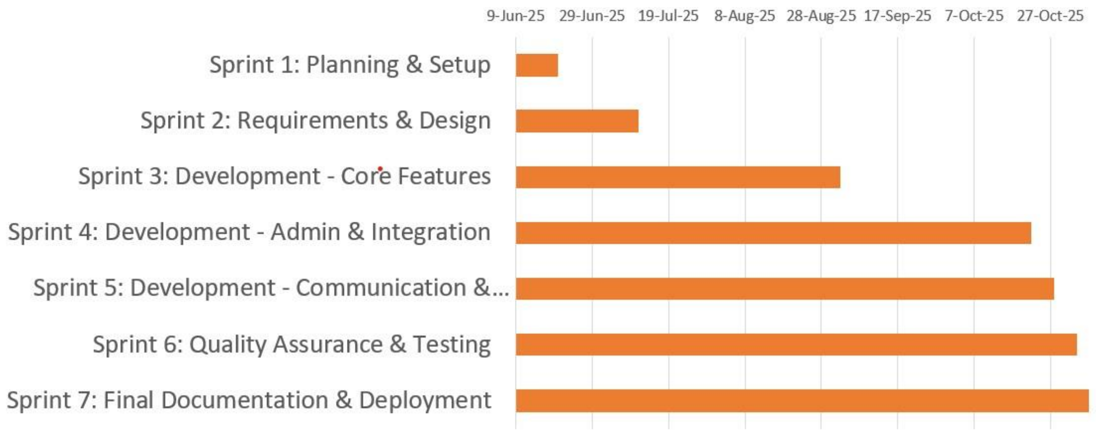

Figure 2.1: Gantt Chart

2.3 Structured Oriented Modeling

Structured Oriented Modeling (SOM) is a traditional engineering approach primarily focused on the internal functions and processes of a system. It utilizes a top-down decomposition strategy, breaking the entire system down into manageable modules and tracking the movement of data between them, typically visualized using Data Flow Diagrams (DFDs). SOM prioritizes defining the logical structure and internal workings the *how* of the system before implementation, often leading to a stable and maintainable code base when followed rigorously.

2.3.1 Use Case Diagram:

The Hostel Management System is driven by three key actors: the Student, the Manager, and the Admin, each with a distinct set of functionalities. The Student acts as the primary end-user, whose core functions revolve around securing and managing accommodation, starting with Register Account and User Login. Once logged in, students can Apply for Room, subsequently View Application status, and Send/Receive Message for communication with staff. The Manager is responsible for daily operations, performing tasks like User Login to access the management area, View Student Records. Finally, the Admin maintains system-wide control, using User Login to access high-level functions like Add/Edit Hostel Details and Manage Manager Accounts, ensuring the system's foundational configuration and staff access are correctly maintained.

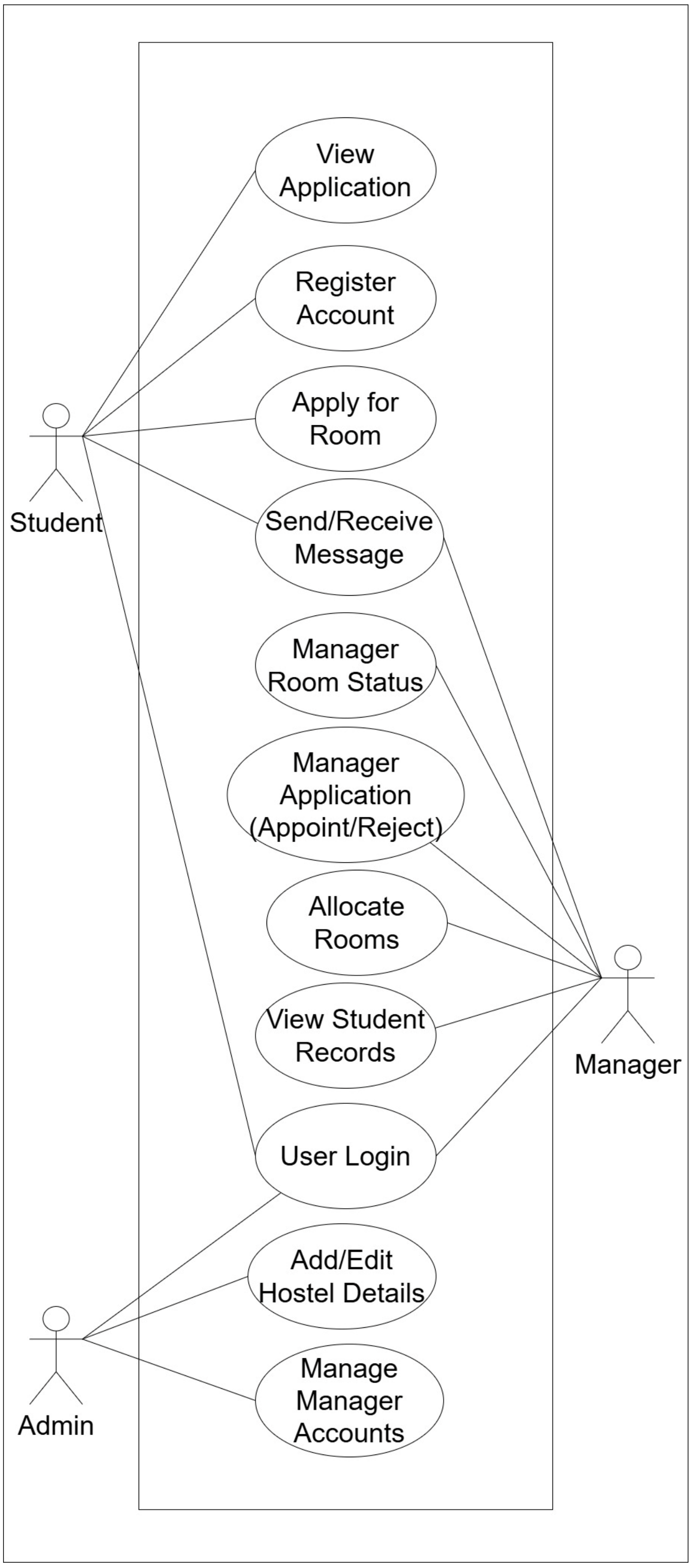

Figure 2.2: Use Case

2.3.2 Activity Diagram:

The Hostel Management System's operational activities, accessible from the dashboard, are divided into four main parallel flows. The Complaint Submission and Assignment flow begins when a complaint is Submitted; the system then uses NLP Analysis to categorize the content before attempting to Auto-Assign Manager. If the Assignment is Successful (Yes), the flow proceeds to Notify Manager, otherwise (No), it branches to Notify Admin for manual intervention. The Complaint Review and Resolution flow allows staff to View Complaints, Check Details, Update Status (e.g., to 'resolved'), and Message Student to communicate updates. The Classification and User Management flow enables staff to Review Classifications made by the system, using a Decision Node (Correct?) to either Approve the classification or Override Classification for manual correction, and also includes the independent activity to Manage Users. Finally, the Reporting and Statistics flow focuses on monitoring, where the system Calculate Stats, allowing staff to View Resolution Rates and **Generate Reports** based on collected performance data.

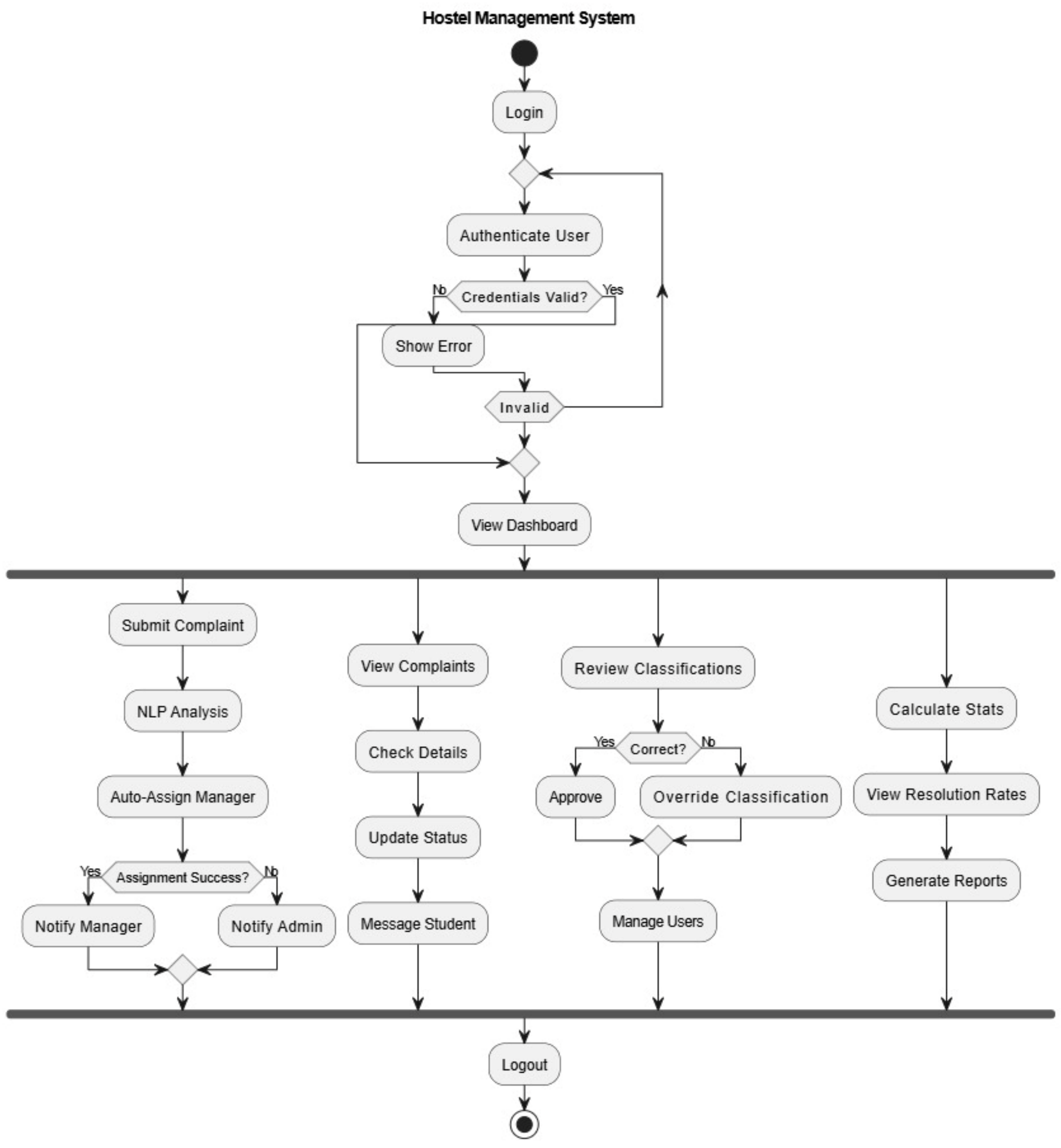

Figure 2.3: Activity Diagram

2.3.3 Sequence Diagram:

The Hostel Manager Management Sequence Diagram details the interaction flow between the Manager, Portal, Handler, and Database. The sequence starts with Authentication & Login, where the Handler verifies credentials against the Database. After successful login, the Manager can View & Filter Complaints and Update Complaint Status, both requiring the Handler to fetch or modify data in the Database. The Manager can also View Hostel Analytics (calculated by the Handler using Database data) before ultimately concluding the session with a Logout.

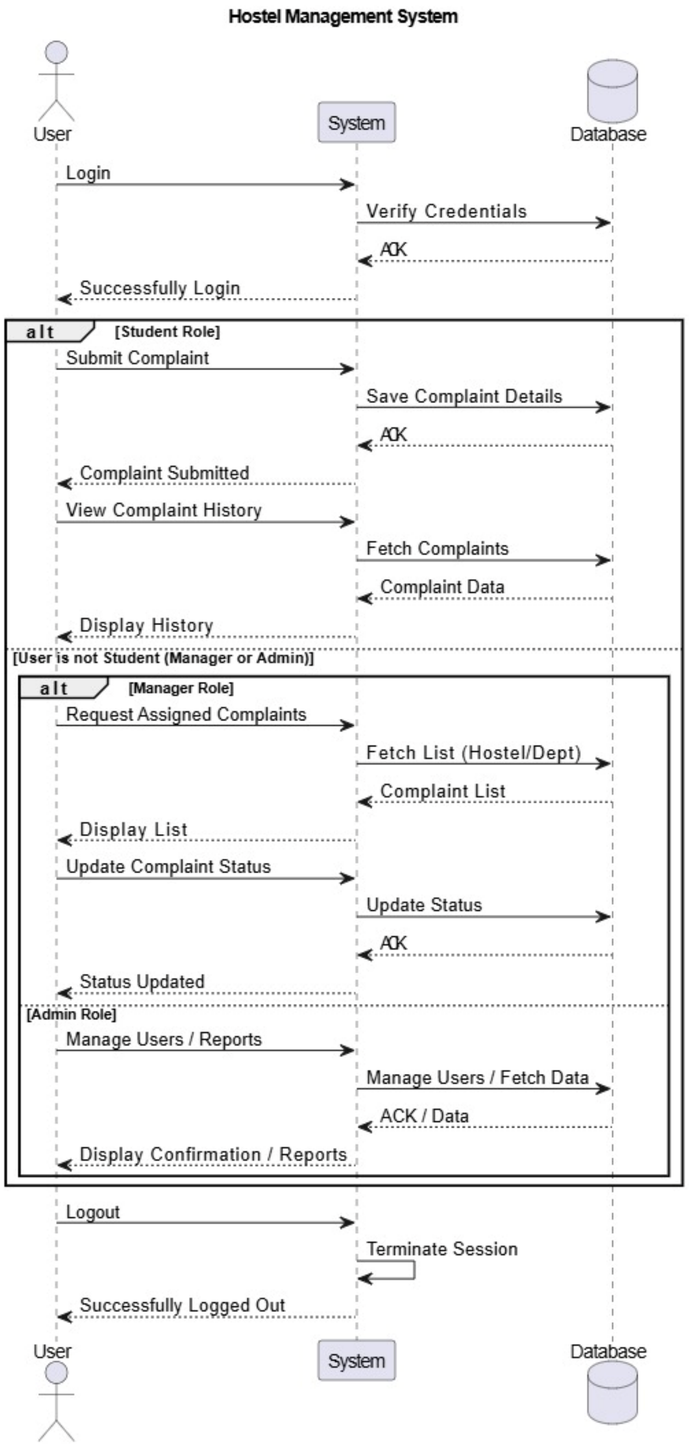

Figure 2.4: Sequence Manager Diagram

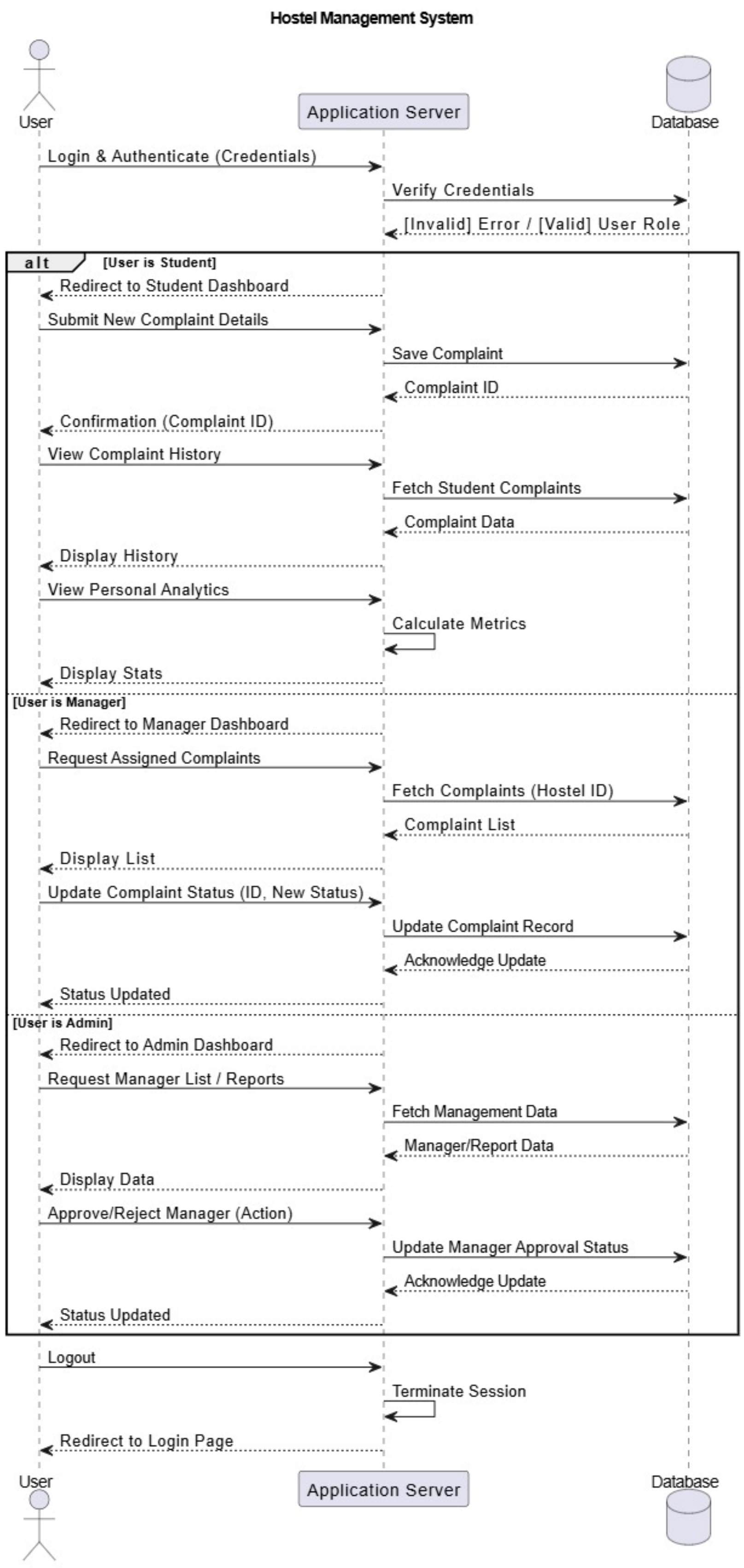

Figure 2.5: Sequence Diagram

2.3.4 System Architecture:

The Hostel Management System utilizes a simple two-tier architecture where the three primary actors Student, Manager, and Admin all interact with the system via the Web Application hosted on the Apache WebServer. This server functions as the Application and Presentation Layer, handling the user interface, business logic, and communication protocols, allowing the actors to perform their various tasks, such as room application, allocation, and account management. The Web Application then connects to the MySQL DatabaseServer, which forms the Data Layer and manages the Database component. This server is responsible for persistent storage, securely holding all critical system information including student records, room status, and credentials, thereby maintaining a clear separation of concerns between application processing and data management.

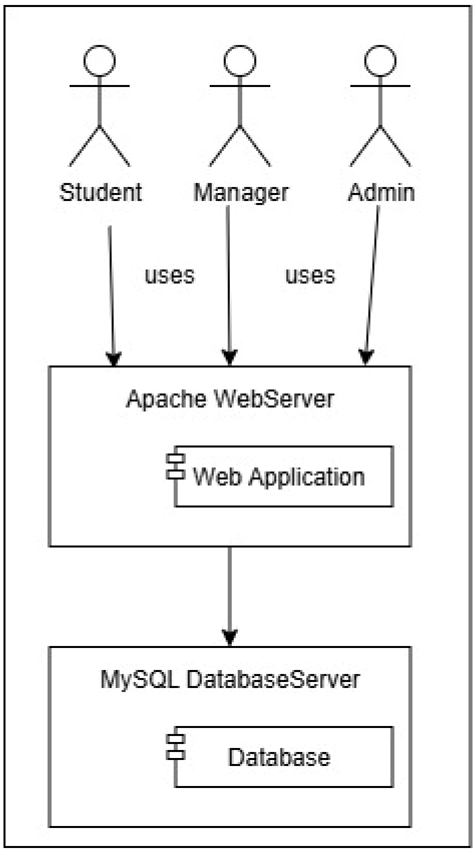

Figure 2.6: System Architecture Diagram

2.4 Design

2.4.1User Interface (UI)

The user interface is designed for professionalism, clarity, and role-based efficiency. The core objective is to deliver a predictable and cohesive user experience (UX) that minimizes effort for each user group.

Key Design Principles:

a) **Clarity & Consistency:** A clean layout with a clear hierarchy of information ensures that users can quickly find what they need. Consistent elements, such as shared navigation and footers (user_header.php, manager_header.php), provide a cohesive and predictable user journey.

b) **Role-Based Simplicity:** The interface utilizes role-specific dashboards (Public, Student, Manager/Admin). This approach is highly effective, presenting each user with only the necessary information and controls, thereby preventing confusion.

c) **Data-Centric Management:** The Manager/Admin dashboard is optimized for efficiency, relying on tables, lists, and clear action buttons ("Approve," "Reject") for efficient record management and application processing.

d) **Responsiveness:** The design incorporates responsive elements (likely from Bootstrap) to ensure optimal usability across various screen sizes (desktop, tablet, and mobile).

Low Fidelity

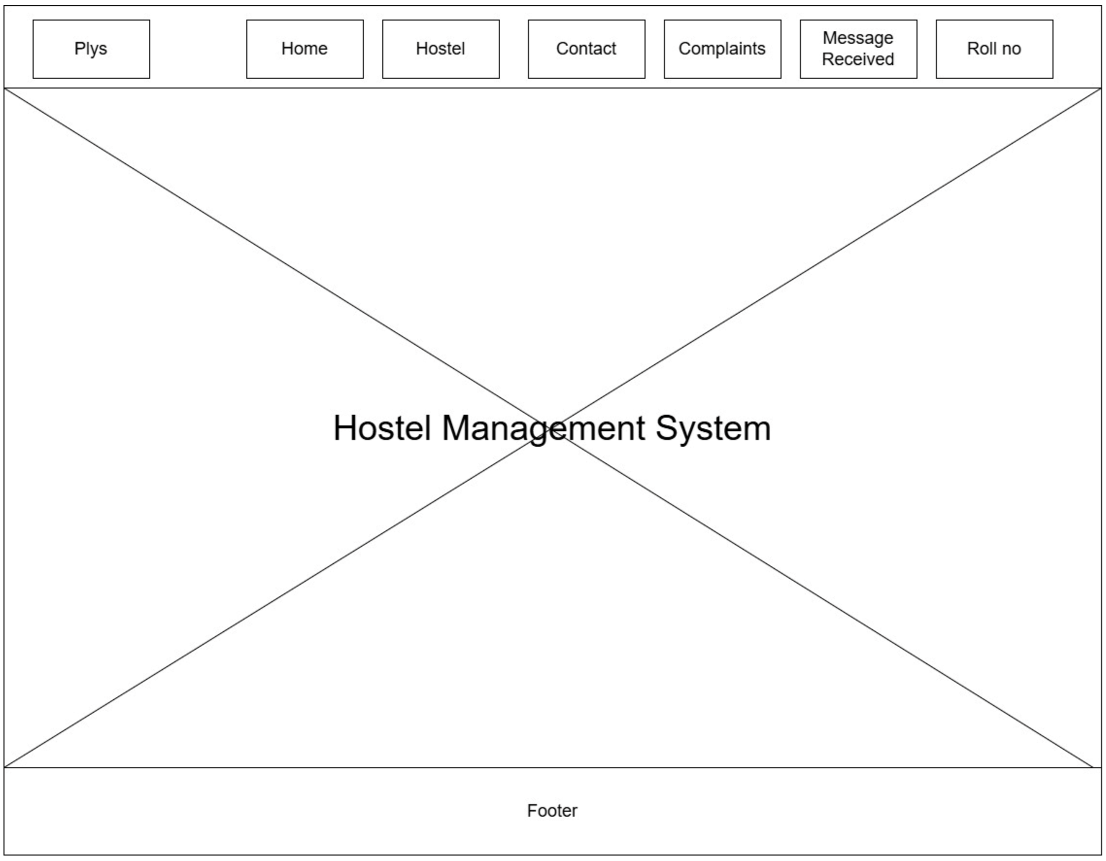

Figure 2.7: Low Fidelity UI Design

High Fidelity

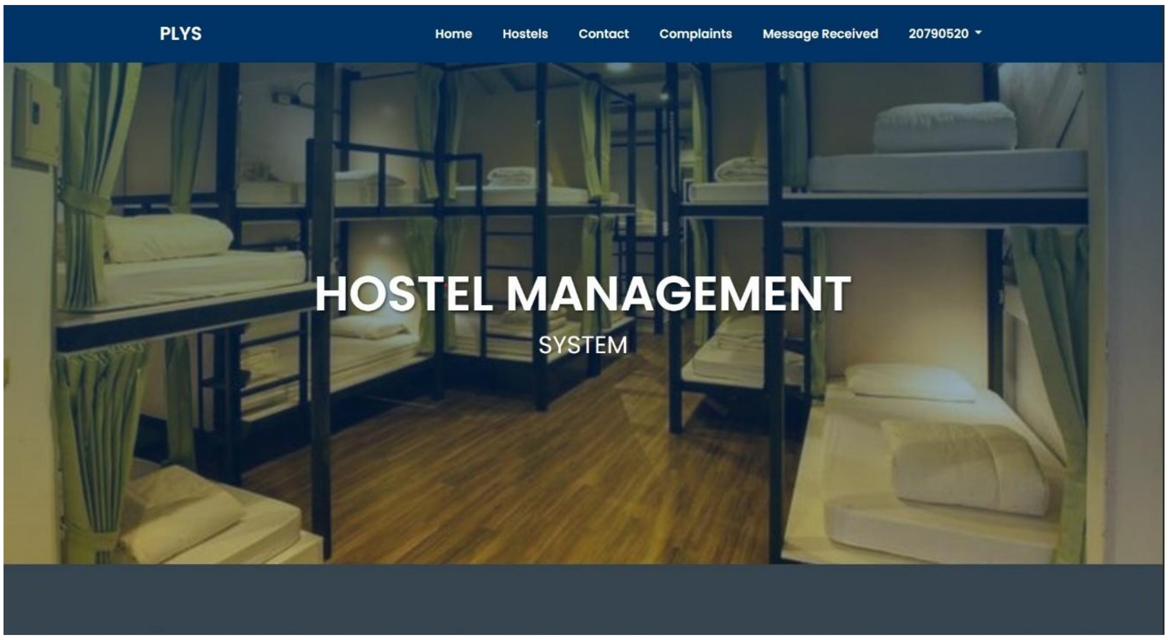

Figure 2.8: High Fidelity UI Design

2.4.2 Major Interfaces

The Hostel Management System is structured around several major interfaces designed to facilitate efficient operations for both students and hostel management staff. Each interface is tailored to specific user roles and functions, ensuring security, clarity, and ease of task completion. The major interfaces are:

a) **Student Application Interface:**
This interface is the primary entry point for students seeking accommodation. It presents a formal application form where students submit all necessary personal and academic details. The purpose is to collect the data required for an application, which is then passed to the backend system for review and processing by the hostel manager.

b) **Room Allocation Interface:**
This is a manager-level interface critical for the core function of the system. It enables hostel managers to review the list of approved or pending student applications and assign an available room to a selected student. The interface provides the manager with the necessary tools to pair an applicant with a specific room ID, confirming the allocation in the system.

c) **Room Change Interface:**
Dedicated to post-allocation management, this interface allows hostel managers to execute room transfers for students who are already residing in the hostel. It handles the process of changing a student's assigned room, ensuring the system accurately updates the room status (making the old room vacant and the new room occupied) and the student's record.

d) **Student Profile Interface:**
This interface serves as the personalized dashboard for the logged-in student. It displays their comprehensive personal information, including contact details and academic status. Most importantly, it clearly shows the student's currently allocated

room details (if assigned), providing them with immediate access to their residential information.

e) **Hostel Manager Creation Interface:**
This interface is an exclusive, administrator-level tool designed for system management. Its sole purpose is to create and register new hostel manager accounts. The admin uses this interface to input manager credentials and assign the new manager to a specific hostel, thus controlling access and managerial oversight within the system.

f) **Student Messages/Contact Report Interface:**
This interface serves as a comprehensive Student Messages/Contact Report, dedicated to facilitating communication management for hostel managers. It allows managers to effectively view, track, and respond to all inquiries, issues, and feedback submitted by students. By centralizing student communication, this report ensures effective tracking and timely resolution of student concerns.

g) **Empty Rooms Report Interface:**
This interface is crucial for Hostel Managers as it generates an Empty Rooms Report, listing all rooms that are currently vacant and available for immediate allocation within the system. The primary goal is to provide a real-time view of room inventory, enabling efficient assignment of new students. A key feature is the integrated search function, which allows the manager to quickly check the occupancy status of any specific room number.

2.5 Database Design and Data Integrity

Database design structures data using **normalization** to organize information into related tables and minimize redundancy. **Data integrity** refers to ensuring the accuracy, completeness, and consistency of this data throughout its lifecycle.

2.5.1 ER Diagram

The ER Diagram illustrates how entities interact, using rectangles for entities, ovals for attributes (like Fname, Mob_no, Hostel_id), and diamonds for relationships (like Belongs to, Allocated to, Submits). Key relationships include a Hostel having Rooms, a Student occupying a Room for a defined duration (start_date, end_date), and a Student submitting an application. Separate entities exist for communication (Message) and staff (Hostel_manager), linking the overall management components.

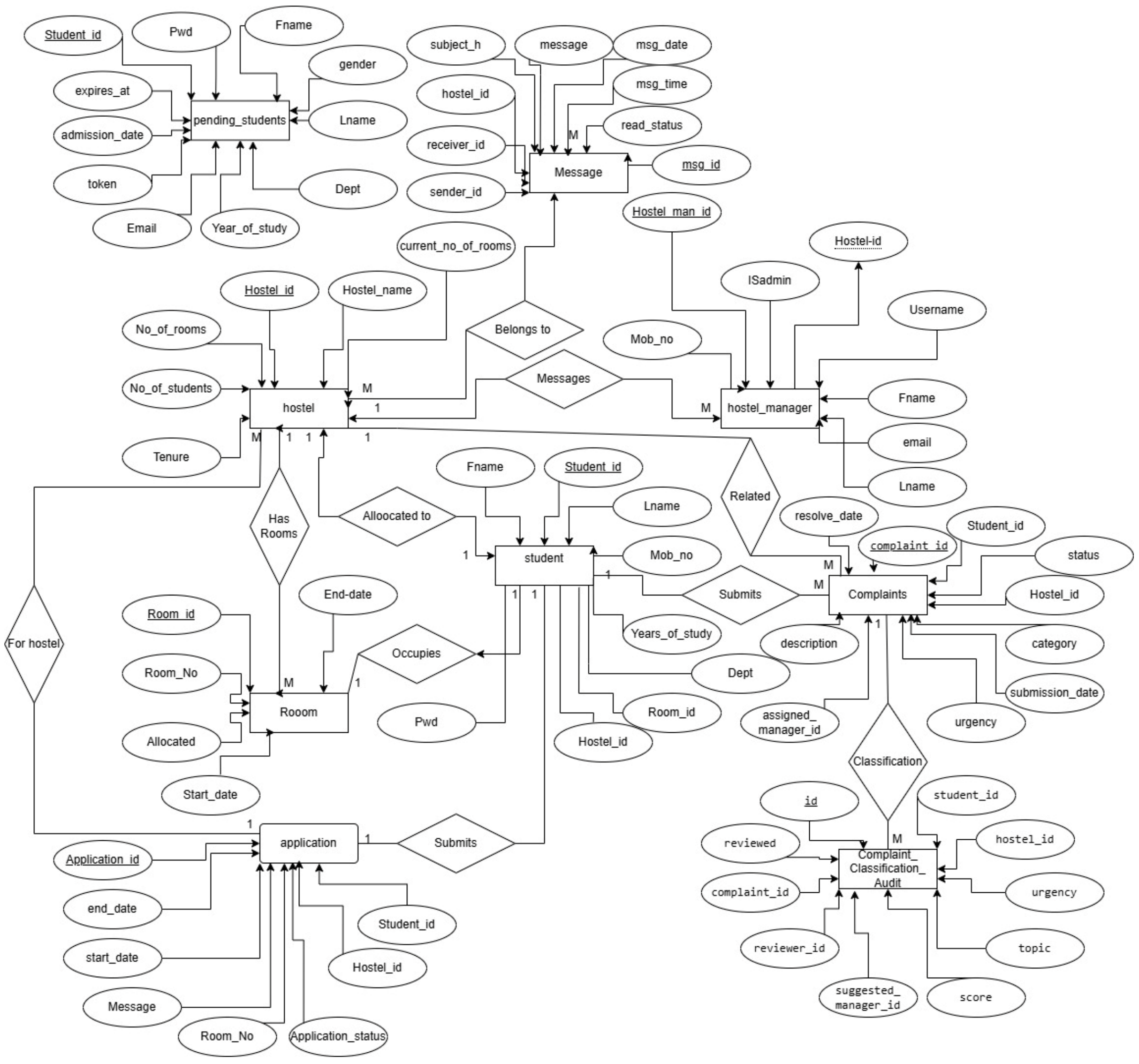

Figure 2.9: ER Diagram

2.5.2 Database Design

The Hostel Management System Database Diagram is built on a highly normalized relational architecture in MySQL, centralizing data integrity and performance. The system manages core entities across several key tables: the hostel table defines physical locations, which are linked to individual room entries via the One-to-Many (1:M) relationship defined by room.Hostel_id. User management is handled by the student and hostel_manager tables, both of which are directly assigned to a specific hostel using the Hostel_id foreign key, while also linking students to their assigned room via Room_id. Operational workflows are covered by the application table (linking students to hostel applications), the complaints table (tracking issues and assigning them to a manager via assigned_manager_id), and the message table for all internal communications. This entire structure is rigorously interconnected using Foreign Keys, ensuring that every record, from a student's complaint to a room allocation, is consistent and accurately associated across the system.

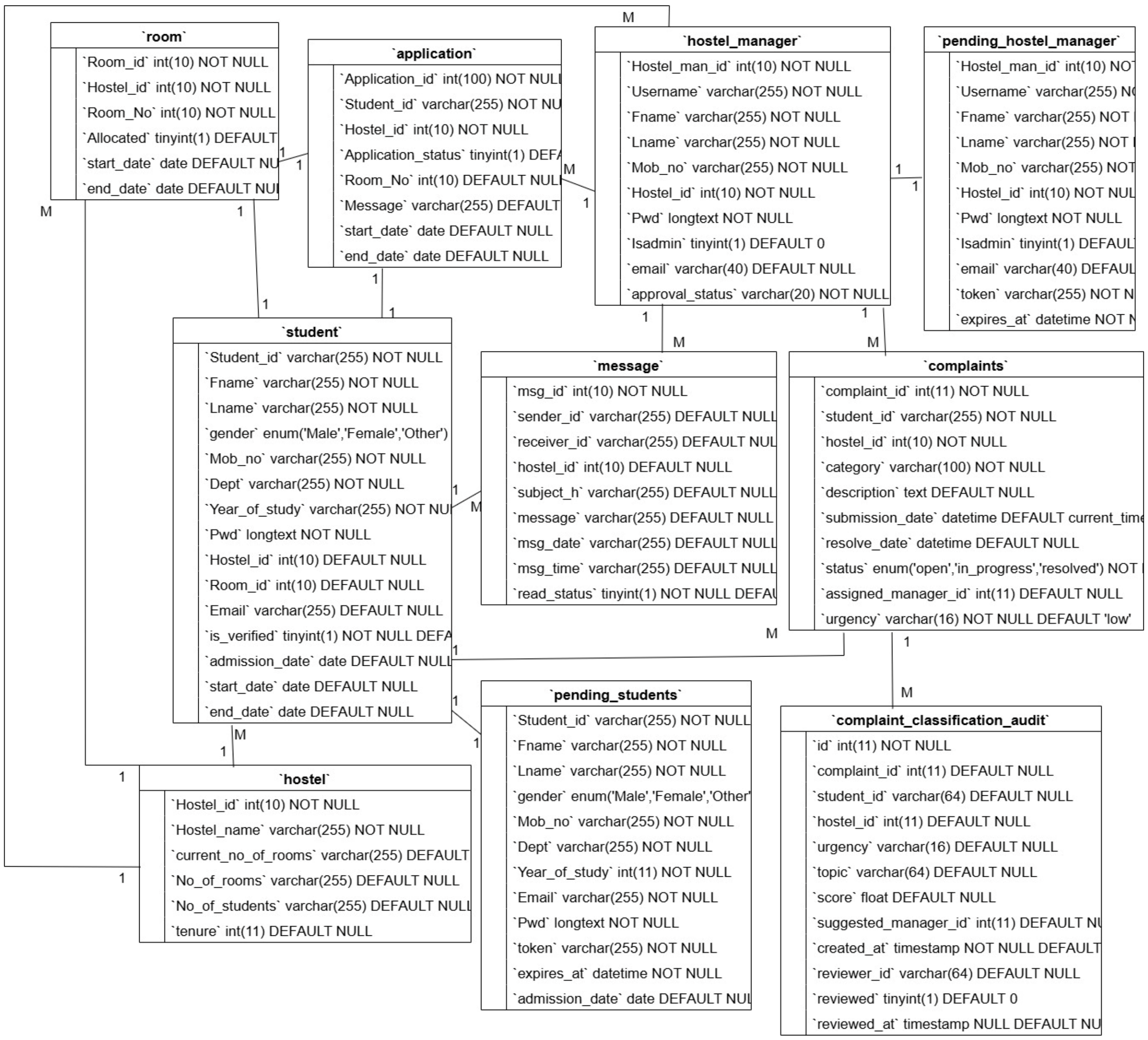
`room`
`Room_id` int(10) NOT NULL
`Hostel_id` int(10) NOT NULL
`Room_No` int(10) NOT NULL
`Allocated` tinyint(1) DEFAULT
`start_date` date DEFAULT NU
`end_date` date DEFAULT NUI
`application`
`Application_id` int(100) NOT NULL
`Student_id` varchar(255) NOT NU
`Hostel_id` int(10) NOT NULL
`Application_status` tinyint(1) DEF/
`Room_No` int(10) DEFAULT NULL
`Message` varchar(255) DEFAULT
`start_date` date DEFAULT NULL
`end_date` date DEFAULT NULL
`hostel_manager`
`Hostel_man_id` int(10) NOT NULL
`Username` varchar(255) NOT NULL
`Fname` varchar(255) NOT NULL
`Lname` varchar(255) NOT NULL
`Mob_no` varchar(255) NOT NULL
`Hostel_id` int(10) NOT NULL
`Pwd` longtext NOT NULL
`Isadmin` tinyint(1) DEFAULT 0
`email` varchar(40) DEFAULT NULL
`approval_status` varchar(20) NOT NULL
`pending_hostel_manager`
`Hostel_man_id` int(10) NO
`Username` varchar(255) N
`Fname` varchar(255) NOT
`Lname` varchar(255) NOT
`Mob_no` varchar(255) NOT
`Hostel_id` int(10) NOT NUL
`Pwd` longtext NOT NULL
`Isadmin` tinyint(1) DEFAUL
`email` varchar(40) DEFAUL
`token` varchar(255) NOT N
`expires_at` datetime NOT N
`student`
`Student_id` varchar(255) NOT NULL
`Fname` varchar(255) NOT NULL
`Lname` varchar(255) NOT NULL
`gender` enum('Male','Female','Other')
`Mob_no` varchar(255) NOT NULL
`Dept` varchar(255) NOT NULL
`Year_of_study` varchar(255) NOT NU
`Pwd` longtext NOT NULL
`Hostel_id` int(10) DEFAULT NULL
`Room_id` int(10) DEFAULT NULL
`Email` varchar(255) DEFAULT NULL
`is_verified` tinyint(1) NOT NULL DEFA
`admission_date` date DEFAULT NULL
`start_date` date DEFAULT NULL
`end_date` date DEFAULT NULL
`message`
`msg_id` int(10) NOT NULL
`sender_id` varchar(255) DEFAULT NULL
`receiver_id` varchar(255) DEFAULT NUL
`hostel_id` int(10) DEFAULT NULL
`subject_h` varchar(255) DEFAULT NULL
`message` varchar(255) DEFAULT NULL
`msg_date` varchar(255) DEFAULT NULL
`msg_time` varchar(255) DEFAULT NULL
`read_status` tinyint(1) NOT NULL DEFAU
`complaints`
`complaint_id` int(11) NOT NULL
`student_id` varchar(255) NOT NULL
`hostel_id` int(10) NOT NULL
`category` varchar(100) NOT NULL
`description` text DEFAULT NULL
`submission_date` datetime DEFAULT current_tim
`resolve_date` datetime DEFAULT NULL
`status` enum('open','in_progress','resolved') NOT
`assigned_manager_id` int(11) DEFAULT NULL
`urgency` varchar(16) NOT NULL DEFAULT 'low'
`pending_students`
`Student_id` varchar(255) NOT NULL
`Fname` varchar(255) NOT NULL
`Lname` varchar(255) NOT NULL
`gender` enum('Male','Female','Other'
`Mob_no` varchar(255) NOT NULL
`Dept` varchar(255) NOT NULL
`Year_of_study` int(11) NOT NULL
`Email` varchar(255) NOT NULL
`Pwd` longtext NOT NULL
`token` varchar(255) NOT NULL
`expires_at` datetime NOT NULL
`admission_date` date DEFAULT NUI
`complaint_classification_audit`
`id` int(11) NOT NULL
`complaint_id` int(11) DEFAULT NULL
`student_id` varchar(64) DEFAULT NULL
`hostel_id` int(11) DEFAULT NULL
`urgency` varchar(16) DEFAULT NULL
`topic` varchar(64) DEFAULT NULL
`score` float DEFAULT NULL
`suggested_manager_id` int(11) DEFAULT NI
`created_at` timestamp NOT NULL DEFAULT
`reviewer_id` varchar(64) DEFAULT NULL
`reviewed` tinyint(1) DEFAULT 0
`reviewed_at` timestamp NULL DEFAULT NU
`hostel`
`Hostel_id` int(10) NOT NULL
`Hostel_name` varchar(255) NOT NULL
`current_no_of_rooms` varchar(255) DEFAULT
`No_of_rooms` varchar(255) DEFAULT NULL
`No_of_students` varchar(255) DEFAULT NULL
`tenure` int(11) DEFAULT NULL
M
1
M

Figure 2.10: Database Design

2.6 Implementation

Table 2.4: Tools and Technologies

Category	Technology	Purpose
Development Environment	**XAMPP** (Apache, MySQL, PHP)	Provided an integrated local server and database environment.
	Visual Studio Code/Sublime Text	Used as the standard source-code editor.
Backend	**PHP (v8.0)**	Handled all server-side logic, including business processes and strong password hashing for authentication.
	MySQL (MariaDB)	The robust relational database for all application data storage.

Category	Technology	Purpose
	PHP Mailer	Third-party library for managing email-based account verification and notifications.
Frontend	**HTML5**	Structured the content and web pages.
	CSS3 & Bootstrap	Provided modern, professional styling and ensured a **responsive** user interface.
	JavaScript	Enabled dynamic and interactive client-side user experiences.

2.7 Modular System Description

The system was organized into five core functional modules:

a) **User Authentication Module:** This module forms the secure core of the system by controlling access and verifying user identity. This module manages the registration of new users, ensuring data is stored securely, followed by mandatory email-based account verification to confirm user validity before they can gain access. Finally, it handles login credential validation and maintains a secure session throughout the user's activity, which assigns and enforces the appropriate roles and permissions (e.g., student or admin).

b) **Student Module:** This Module is the core functional area designed for the student user, focusing on managing their personal records and hostel application process. This interface allows students to efficiently maintain their profile management details, ensuring their personal information is current. Crucially, it facilitates the detailed room application submission, guiding the student through the process of formally applying for hostel accommodation. Finally, it provides immediate visibility into the application process by allowing students to view their application status, enabling them to track the progress of their request from submission to final room allocation.

c) **Hostel Manager Module:** This Module provides the essential tools for hostel staff to oversee and manage the entire lifecycle of student accommodation. This module enables staff to efficiently review and process pending applications, perform mandatory room allocation for accepted students, and continuously monitor occupied rooms to maintain real-time inventory and occupancy status. Furthermore, it includes functions to manage the formal process of student departures, ensuring rooms are properly vacated and marked as available for new allocations.

d) **Administrator Module:** This Module functions as the top-level supervisory control panel for the entire system, granting system administrators overarching authority. Its primary role is to manage the next level of system hierarchy by facilitating the creation and management of Hostel Manager accounts, thereby defining which staff members can oversee day-to-day hostel operations. Additionally, this module allows for the editing of core hostel information, ensuring all foundational data and system-wide settings remain accurate and up-to-date across the application.

e) **Communication Module:** This Module is a straightforward internal messaging system designed to facilitate direct and private message exchange within the application. Its core function is to bridge the communication gap by allowing students to easily send inquiries, requests, or information to their respective hostel managers. This focused channel ensures that communication regarding accommodation and administrative issues remains organized, documented, and delivered directly to the appropriate staff member responsible for the student's area.

f) System & Maintenance Module: This Module is the dedicated, non-user-facing component responsible for preserving the system's accuracy, efficiency, and structural health. This module executes crucial background tasks, such as automatically checking for and flagging expired room allocations to ensure room availability data is always current. It also runs periodic routines to update hostel statistics and counts, which feeds real-time data to managerial dashboards. Finally, this module includes processes for managing database backups, guaranteeing system resilience and the ability to restore data integrity in case of any failure.

2.8 Testing

Testing is a critical phase in software development designed to ensure the system is free of errors, meets its specified requirements, and performs reliably under various conditions. It involves systematically executing the software with the intent of finding defects (bugs), validating the core functions (like application submission and room allocation), and verifying the security and data integrity mechanisms. The goal is to reduce risk and confirm that the final system is suitable for deployment and use by students and managers.

Table 2.5 Test case: Successful New Student Registration

Test Case ID: TC001
Module Name: User Authentication
Test Title: Successful New Student Registration
Description: Test for a new student successfully creating an account.
Pre-condition: The user must not have an existing account. The user is on the registration page.
Post-condition: A new student record is created in the database. The user is redirected to the login page or a success page.
Test steps 1. Navigate to http://localhost/project/signup.php.

2. Fill in all required fields (First Name, Last Name, Student ID, Mobile Number, Gender, Email, Password, Confirm Password).
3. Click the "SIGNUP" button.

Test case	Test Scenario	Test Data	Expected Results	Actual Results	Status
TC001	User Authentication	Rollno:20790520 Password:@sujal#2003	The system registers the user, and a success message is displayed. The user can now log in with the new credentials.	Same as expected	Pass

Table 2.6 Test case: Incomplete Room Application Submission

Test Case ID: TC002
Module Name: Student
Test Title: Incomplete Room Application Submission
Description: Test for submitting a room application form with missing required fields.
Pre-condition: The user must be logged in as a student and must not have an active room allocation.
Post-condition: The application is not submitted. The system displays an error message indicating the missing fields.
Test steps 1. Log in as a student. 2. Navigate to http://localhost/project/application_form.php. 3. Fill in some, but not all, of the required fields (e.g., leave the 'Guardian Name' field blank). 4. Click the "Submit" button.

Test case	Test Scenario	Test Data	Expected Results	Actual Results	Status
TC002	Submit application with missing data	Guardian Name: (empty) All other fields filled correctly.	The system should prevent form submission and display an error message like "Please fill in all required fields."	Same as expected	Pass

Table 2.7 Test case: Successful Room Allocation

Test Case ID: TC003					
Module Name: Hostel Manager					
Test Title: Successful Room Allocation					
Description: Test for a Hostel Manager successfully allocating a room to a student with a pending application.					
Pre-condition: A student has submitted a room application. The Hostel Manager is logged in. There is an available room.					
Post-condition: The student is allocated to a room. The room's status is updated to 'Occupied'. The student's application status is updated to 'Allocated'.					
Test steps 1. Log in as a Hostel Manager. 2. Navigate to the student applications page. 3. Find a pending application and click the "Allocate Room" button/link. 4. Select an available room from the list on http://localhost/project/allocate_room.php. 5. Set the allocation duration and click "Submit".					
Test case	Test Scenario	Test Data	Expected Results	Actual Results	Status
TC003	Allocate a room to a student	Student ID: 20790520 (from TC001) Room Number: 101	The room is successfully allocated. The system shows a confirmation message. The student's profile now shows the allocated room details.	Same as expected	Pass

Table 2.8 Test case: Create Hostel Manager with Duplicate Email

Test Case ID: TC004
Module Name: Administrator
Test Title: Create Hostel Manager with Duplicate Email
Description: Test for preventing the creation of a new Hostel Manager account with an email address that already exists in the system.
Pre-condition: An administrator is logged in. A Hostel Manager account with a specific email already exists.
Post-condition: The new Hostel Manager account is not created. An error message is displayed indicating the email is already in use.
Test steps 1. Log in as an Administrator. 2. Navigate to http://localhost/project/admin/create_hm.php.

3. Fill in the form with details for a new manager, but use an existing manager's email address. 4. Click the "Create" or "Submit" button.					
Test case	Test Scenario	Test Data	Expected Results	Actual Results	Status
TC004	Attempt to create HM with existing email	Username: Ram Email: RamA@example.com Password: password123	The system should reject the submission and display an error message such as "Email address already exists."	Same as expected	Pass

Table 2.9 Test case: Student Sends Message to Hostel Manager

Test Case ID: TC005					
Module Name: Communication					
Test Title: Student Sends Message to Hostel Manager					
Description: Test for a student successfully sending a message to their assigned Hostel Manager.					
Pre-condition: The student is logged in and has been allocated a room by a Hostel Manager.					
Post-condition: The message is stored in the database. The Hostel Manager can view the message in their inbox.					
Test steps 1. Log in as the student (e.g., sujal@example.com). 2. Navigate to the "Message Hostel Manager" page. 3. Enter a subject and a message in the provided text fields. 4. Click the "Send" button.					
Test case	Test Scenario	Test Data	Expected Results	Actual Results	Status
TC005	Student sends a message	Subject: Change room Message: Hello, I want my room (101) to be changed Please can you chnage it?	The message is sent successfully, and a confirmation is displayed. The message appears in the Hostel Manager's message list.	Same as expected	Pass

Table 2.10: Testing Phases

Testing Level	**Objective**	**Key Focus/Example**
Unit Testing	To verify that individual functions and scripts work correctly in isolation.	Testing the database connection script (config.inc.php) for successful connection, and rigorous checks on form processing scripts using valid, invalid, and empty inputs to confirm data validation.
Integration Testing	To ensure that combined modules interact correctly and that data flows accurately between them.	Testing the seamless flow between the User Authentication Module and the Student Module (e.g., a new user successfully logging in and accessing their dashboard). A critical case was the flow from a student applying for their appearance on the manager's dashboard.
System Testing	To test the entire application end-to-end to verify all modules work together as a cohesive system, validating complete user workflows.	Executing comprehensive workflows, such as: Student Lifecycle (registration -> email verification -> room application -> viewing allocation details) and Manager/Admin Operations (manager logging in, approving application, allocating room; administrator creating a new manager account).

CHAPTER III
CONCLUSION AND RECOMMENDATION

3.1 Project Summary

The Hostel Management System is a functional, web-based application developed using procedural PHP and MySQL for efficiently managing student accommodations. The system centralizes all hostel operations, replacing manual paperwork and significantly enhancing administrative efficiency. It supports three distinct user roles with clearly defined responsibilities:

a) Students: Can manage their profiles, view available hostels and room capacity, and submit formal room applications directly through the system.
b) Hostel Managers: Are responsible for operational tasks, including reviewing and acting upon student applications, allocating specific rooms, and overseeing the day-to-day welfare of students within their assigned hostel.
c) Administrators: Maintain high-level system oversight, manage core hostel data, and handle the creation and maintenance of Hostel Manager accounts.

The application uses the external PHP-Mailer library for automated email notifications (e.g., application status updates and account verification) and employs the Bootstrap framework for its responsive, front-end interface, ensuring a modern and accessible user experience across various devices.

3.2 Conclusion

The Hostel Management System currently serves as a highly successful proof-of-concept, demonstrably meeting all primary functional requirements for student application, room allocation, and managerial oversight. Its foundational structure, developed using procedural PHP and MySQL, provides a solid operational base. By implementing the recommended architectural and security enhancements, specifically transitioning to a more structured framework and applying advanced security protocols, the project can be quickly transformed into a robust, scalable, and modern platform that fully adheres to contemporary development standards and ensures long-term viability and data integrity. This enhancement will significantly improve performance and maintainability, ensuring the system can handle future growth in student populations.

3.2 Key Recommendations

To ensure the system's long-term viability, maintainability, and security, a strategic evolution of the codebase and architecture is recommended:

Codebase & Architecture Modernization

a) **Adopt OOP/MVC:**

i. **Object-Oriented Programming:** Refactor the existing procedural code into an Object-Oriented Programming (OOP) architecture and apply the Model-View-Controller (MVC) pattern. This will improve code organization, reusability, and long-term scalability.

b) **Security Enhancements**

i. **SQL Injection Prevention:** Enforce the consistent use of prepared statements (e.g., using PDO) for *all* database queries to eliminate SQL injection vulnerabilities.

ii. **XSS Prevention:** Ensure all user-generated content is properly escaped using functions like htmlspecialchars() before rendering to mitigate Cross-Site Scripting (XSS) risks.

c) **User Experience (UX)**

i. **Dynamic Interactions:** Implement AJAX for key actions, such as form submissions and availability checks, to provide a smoother, non-page-reloading user experience.

ii. **Front-End Framework:** Consider integrating a modern JavaScript framework (e.g., React or Vue.js) to build a more dynamic and interactive UI.

d) **System Automation**

i. Implement automated processes (via cron jobs or scheduled tasks) for essential maintenance, such as de-allocating rooms post-expiry dates and systematically cleaning up old application requests.

REFERENCES

Adeboye, M. A., Ajao, A. O., & Okeyinka, O. E. (2020). Development of Web-Based Hostel Management System. *International Journal of Scientific and Engineering Research, 11*(9), 137–143.

Islam, M. S., & Hossain, M. A. (2018). *Hostel management system* (Undergraduate thesis, East West University). Retrieved from http://dspace.ewubd.edu/handle/123456789/3055

Jain, A., Sharma, A., Singh, A., & Gupta, A. (2021). Understanding The Intricacies of Hostel Management Platform (HMP) Through Modern Technologies. *International Journal of Innovative Research in Computer Science & Technology, 9*(6), 18–22.

Kumar, P. S. A. P., Sairam, K. V. S. S. S. S., Karthikeya, G. S., Murthy, K. L. N., & Murthy, K. S. S. P. V. S. L. N. (2022). Automated Hostel Management System. *International Journal of Scientific Research in Engineering and Technology, 11*(5), 103–111.

Oyeleke, O. A., Omidiora, E. O., & Alowolodu, S. A. (2019). Development of an E-Based Hostel Management System. *International Journal of Computer Applications, 177*(45), 22–28.

Shinde, S. S., Patil, S. S., Kolekar, S. R., & Kumbhar, P. P. (2022). Hostel Management System. *International Journal of Research and Presentation, 3*(10), 1–5.

Suganya, S., Kowsalya, S., Priyadharshini, S., & Sangeetha, R. (2019). Hostel Management System. *International Journal of Trend in Scientific Research and Development, 3*(4), 1016–1018.

Diyaolu, A. S., Adebisi, A. A., & Adebola, E. A. (2024). Development of an E-Based Hostel Management System. *International Journal of Innovative Science and Research Technology*, *9*(6), 1184–1191.

Darwin, E. J., P., & Jha. (2024). Hostel Management System. *International Journal of Advanced and Innovative Technology in Engineering*, *10*(2), 99-103.

Aqeel, B., Alshahrani, T., Alasmari, R., & Alshehri, F. (2024). Web-based Hostel Reservation and Location Identification System Using IOT. *International Journal of Advanced Engineering and Management Research*, *10*(2).