

Hostel Information System: A Comprehensive Hostel Management Application

Submitted in partial fulfillment for the award of the degree of
Bachelor of Technology
in
Computer Science and Engineering
Under the guidance of

Dr. Siddhartha Sankar Satapathy
Professor
Department of Computer Science and Engineering
Tezpur University



Submitted By

Sakil Aziz
Roll No: CSB22012

Department of Computer Science and Engineering
Tezpur University, Tezpur (784028), Assam



Department of Computer Science and Engineering
Tezpur University

CERTIFICATE FROM PROJECT GUIDE

This is to certify that the project report entitled “**Hostel Information System: A Comprehensive Hostel Management Application**”, submitted to the Department of Computer Science and Engineering, Tezpur University, in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bona fide work carried out by **Mr. Sakil Aziz**, Roll No. **CSB22012**, under my supervision and guidance.

All assistance received from various sources has been duly acknowledged.

Date: 30th May 2025
Place: Tezpur University

Dr. Siddhartha Sankar Satapathy
Professor, Department of CSE
Tezpur University



Department of Computer Science and Engineering
Tezpur University

CERTIFICATE

This is to certify that the report entitled “**Hostel Information System: A Web Application for Hostel Management**” is a bona fide record of the project completed by **Sakil Aziz (CSB22012)** and submitted in fulfillment of the requirements and regulations of the mini project for the 3rd year of the 4-year Bachelor of Technology course in Computer Science and Engineering at Tezpur University. He has carried out the project work under the supervision of **Dr. Siddhartha Sankar Satapathy**, Professor, Department of Computer Science and Engineering, Tezpur University.

This certification does not necessarily endorse or accept every statement made, opinion expressed, or conclusion drawn in the report. It only signifies the acceptance of this report for the purpose for which it is submitted.

Date: 30th May 2025
Place: Tezpur University

Dr. Sarat Saharia
Head of Department, CSE



Department of Computer Science and Engineering
Tezpur University

DECLARATION

I hereby declare that the Project Report titled “**Hostel Information System: A Web Application for Hostel Management**”, submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering during the academic session of Spring 2025, is a result of my independent work. All the information and content presented in this report is based on my original research, analysis, and implementation. Any references and sources of information used have been duly acknowledged and cited.

I further declare that this project report has not been submitted to any other academic institution or published in any form.

Date: 30th May 2025
Place: Tezpur University

Sakil Aziz
CSB22012

ACKNOWLEDGMENT

I would like to take this opportunity to convey my sincere gratitude to **Dr. Siddhartha Sankar Satapathy**, who served as my project supervisor and guide, for his exceptional guidance, constant support, and valuable mentorship during the course of this project. The web application *Hostel Information System* has greatly benefited from his in-depth expertise, insightful suggestions, and professional guidance. I am truly grateful for his patience, encouragement, and the countless hours he dedicated to reviewing and refining the project. His unwavering encouragement and faith in my abilities have been a great source of inspiration.

Additionally, I would like to express my profound gratitude to **Dr. Sarat Saharia**, Head of the Department of Computer Science and Engineering at Tezpur University. His inspiring leadership, encouragement, and support have been instrumental in fostering an academic environment conducive to the success of innovative initiatives like this project. His advice and commitment to nurturing software development talent have been invaluable.

I would also like to thank my friends for their contributions to this project through brainstorming sessions, teamwork, and constructive feedback. Their combined efforts and diverse perspectives have enriched the software development process and contributed significantly to producing a robust software solution.

Finally, I am deeply thankful to my family for their unwavering support, patience, and inspiration throughout this journey. Their confidence in me and continuous encouragement have been a driving force behind my accomplishments.

Contents

1	Introduction	1
1.1	Background	1
1.2	Limitations of Existing Systems	1
1.3	Objectives	1
1.4	Technology Overview	1
1.5	Document Structure	1
2	Initial System Study	2
2.1	Drawbacks of the Existing Manual System	2
2.2	Problem Statement	2
2.3	Proposed Solution	2
2.4	Scope	2
3	Feasibility Report	3
3.1	Technical Feasibility	3
3.2	Operational Feasibility	3
3.3	Economic Feasibility	3
3.4	Conclusion	3
4	Overall Description	4
4.1	Product Perspective	4
4.2	Product Functions	4
4.3	Operating Environment	4
4.4	Design and Implementation Constraints	5
4.5	Assumptions and Dependencies	5
5	System Features and Functional Requirements	6
5.1	Detailed Feature Descriptions	6
5.1.1	Complaint Management System	6
5.1.2	Boarders List Management	7
5.1.3	Notification Module	7
5.1.4	Wing Assignment and Role Management	8
6	Other Non-functional Requirements	9
6.1	Performance Requirements	9
6.2	Safety and Security Requirements	9
7	System Design	10
7.1	System Architecture	10
7.2	Data Flow Diagram (DFD)	10
7.3	Database Design	13
8	System Implementation	19
8.1	Tech Stack	19
8.2	Hosting Service	19
8.3	User Interface Design	19
8.3.1	Student Raises a Complaint	20
8.3.2	Warden/Admin Login	20
8.3.3	Wing Representative Dashboard	21
8.3.4	Prefect Dashboard	21

8.3.5	Warden Dashboard and Logs	21
8.3.6	Final Complaint Resolution by Prefect	22
8.4	Room and Boarder Management	23
8.5	Wing Management	24
8.6	Notice Upload	24
9	System Testing	26
9.1	Unit Testing	26
9.2	Validation Testing	26
9.3	Functional Testing	26
9.4	Database Testing	26
10	Conclusion and Future Work	28
10.1	Conclusion	28
10.2	Future Work	28

1. Introduction

1.1. Background

Hostel management in residential universities often suffers from inefficiencies due to manual processes. Students typically report issues like broken fixtures or poor sanitation verbally or through handwritten logs, leading to delays, miscommunication, and unresolved complaints. Additionally, room allotment and change requests are managed through paper-based systems, which are error-prone and lack transparency.

1.2. Limitations of Existing Systems

The absence of a centralized digital system makes it difficult for hostel staff to track complaints, monitor assets, or communicate updates effectively. There is no proper mechanism for recording complaint statuses, managing furniture and fixtures, or ensuring that students are informed about hostel notices or complaint resolutions.

1.3. Objectives

The main goal of this project is to modernize hostel operations by improving transparency, accountability, and responsiveness. It seeks to ensure faster complaint resolution, better resource monitoring, and real-time updates for both students and administrators.

1.4. Technology Overview

The system will be developed using modern web technologies to ensure responsiveness, security, and ease of use. The backend will integrate with a MySQL database to maintain reliable data storage and enable efficient retrieval across modules.

1.5. Document Structure

This document outlines the system's objectives, features, architecture, and implementation strategy. It serves as a guide for developers, testers, and stakeholders involved in building and deploying the Online Hostel Information System.

2. Initial System Study

2.1. Drawbacks of the Existing Manual System

The current manual system at Tezpur University lacks structure and efficiency in handling key hostel operations. Complaint resolution, room allotment, and asset tracking are informal and paper-based, resulting in:

- **Delayed Complaint Resolution:** Issues like broken fans or toilets are reported verbally, often remaining unaddressed.
- **No Tracking Mechanism:** Complaints cannot be monitored, prioritized, or historically referenced.
- **Manual Room Allotment:** Allotment and change requests are handled without transparency, leading to errors and bias.
- **No Asset Monitoring:** Resources like furniture and fixtures are not systematically managed.
- **Lack of Notifications:** Students receive no real-time updates on complaints or hostel announcements.
- **Unstructured Lost & Found:** Lost items have minimal chances of recovery due to the absence of a formal reporting system.

These issues contribute to poor student satisfaction and administrative inefficiency.

2.2. Problem Statement

The manual management of hostel operations at Tezpur University has resulted in delays, lack of accountability, and poor communication. Without a centralized digital system, processes like complaint resolution, room allotment, and resource tracking remain inefficient and opaque. There is a clear need for an integrated solution that automates these tasks and improves visibility and communication for all stakeholders.

2.3. Proposed Solution

To address these issues, we propose an Online Hostel Information System that digitizes key administrative functions. The system will include modules for complaint tracking with escalation, room allotment and change requests, asset management, a lost and found registry, and a digital notice board. It aims to centralize information and streamline communication between students and hostel staff.

2.4. Scope

This Software Requirements Specification (SRS) outlines the design of a Hostel Information System to streamline:

- Complaint management with escalation, tracking, and status updates
- Room allotment and change requests with occupancy visualization
- Hostel asset monitoring and maintenance tracking
- Digital notices and real-time notifications

The system aims to modernize hostel administration, ensuring better efficiency, transparency, and responsiveness.

3. Feasibility Report

3.1. Technical Feasibility

The Online Hostel Information System is technically feasible using current hardware, software tools, and available developer skills. It will be built as a web-based platform using Node.js (backend), MySQL (database), and React (frontend)—all open-source, scalable, and widely adopted technologies.

Existing University-hostel infrastructure with basic internet access and computing devices is sufficient for deployment.

3.2. Operational Feasibility

The system fits well with current hostel workflows and requires minimal training for staff and students. It replaces manual systems with a digital platform that's intuitive and accessible from any browser-enabled device.

It addresses major pain points like:

- Inefficient complaint resolution processes.
- Lack of centralized complaint and maintenance tracking.
- Limited transparency in room allotments and updates.
- Gaps in communication between admin and boarders.

Digital notifications and status updates will boost transparency, reduce response time, and enhance user satisfaction.

3.3. Economic Feasibility

Economically, the system is cost-effective due to its reliance on open-source tools. Primary costs involve:

- Development resources (developers, testers).
- Minimal cloud or server hosting.
- Occasional updates and maintenance.

No major hardware investments are needed unless future IoT or RFID features are added. Over time, the system will lower administrative costs, reduce paper use, and improve efficiency.

3.4. Conclusion

Considering the technical, operational, and economic factors, the project is feasible and aligned with institutional needs. It promises improved efficiency, better communication, and a more structured hostel management experience for all users.

4. Overall Description

4.1. Product Perspective

The Online Hostel Information System is conceived as a standalone, web-based platform that seeks to replace the currently manual and informal processes associated with hostel management. The existing method relies heavily on physical registers, verbal communication, and ad hoc spreadsheets, all of which contribute to inefficiencies and inconsistencies in data handling. The proposed system will standardize operations such as complaint management, room allotment, and asset tracking through an intuitive digital interface.

While the initial design assumes an independent deployment, the system architecture is flexible enough to allow for integration with any existing hostel or institutional databases, if they exist. The platform will be accessible from any device with a web browser—whether desktop, laptop, tablet, or smartphone—ensuring ease of use for both students and administrative personnel. Furthermore, the system is designed to support real-time communication and status updates, enhancing the responsiveness of hostel administration.

4.2. Product Functions

The system will offer a comprehensive set of functionalities aimed at automating and improving the following core hostel management tasks:

- **Complaint Management:** Students and designated representatives (e.g., wing captains, prefects) will be able to lodge complaints regarding maintenance issues, such as broken fans, faulty lighting, sanitation problems, and other infrastructural concerns. Each complaint will be categorized (e.g., electrical, plumbing, general), prioritized (e.g., urgent, normal), and tracked through various resolution stages (Pending, Approved, In Progress, Resolved).
- **Real-Time Tracking:** The system will maintain a historical log of all complaints, allowing students and administrators to view the status and progress of each issue. Timely updates will be reflected on the user dashboard, providing transparency and accountability.
- **Digital Notifications:** Important updates—such as complaint approvals, resolution progress, and general announcements—will be communicated through a digital notice board interface. Notifications will also be delivered to users in real time to ensure prompt awareness and action.

Future versions of the system may include extended features such as biometric access control, integration with student ID systems, and analytics dashboards for administrative planning.

4.3. Operating Environment

The application will be deployed in a client-server architecture, where the backend services (APIs, databases, authentication mechanisms) are hosted on a centralized server. Users will interact with the system through a responsive web interface built using modern frontend technologies (such as React.js or Vue.js), while the backend will rely on robust technologies like Node.js and MySQL for data processing and storage.

The system is expected to operate reliably across all major web browsers (Chrome, Firefox, Safari, Edge) and mobile platforms (iOS, Android). Access will be secured through user authentication and role-based permissions to ensure data integrity and privacy.

4.4. Design and Implementation Constraints

Several constraints must be acknowledged during the development and deployment of the system:

- **Browser and OS Compatibility:** The frontend must be fully responsive and compatible with modern web browsers and mobile operating systems.
- **Data Privacy and Security:** As the system handles sensitive personal data and complaint logs, it must comply with institutional data protection policies and general IT security practices (e.g., HTTPS, encrypted credentials).
- **Scalability:** The system architecture should support an increasing number of users and database entries, especially during peak times such as semester starts or hostel admissions.

4.5. Assumptions and Dependencies

The successful implementation of the system depends on several assumptions and external dependencies:

- **Reliable Internet Connectivity:** Real-time functionalities such as status updates and notifications require stable and continuous network access for both clients and the server.
- **User Familiarity with Technology:** The system assumes a baseline level of digital literacy among students and staff. Basic navigation skills (clicking, form-filling, etc.) are expected.
- **Administrative Support:** Timely and accurate data entry (e.g., room assignments, asset lists) depends on active cooperation from hostel administration.
- **Third-Party Integrations:** Any plans to integrate with existing hostel databases or authentication systems (e.g., university login portals) depend on the openness and structure of those systems.

5. System Features and Functional Requirements

5.1. Detailed Feature Descriptions

5.1.1. Complaint Management System

- **Functionality:**

- Allows students to raise complaints without logging in , but by entering their roll number; the system fetches associated room details automatically.
- Complaints are categorized into predefined types such as Electrical, Plumbing, Internet, Furniture, and Sanitation.
- Each complaint is assigned a priority level (Low, Medium, High, Critical) based on severity and urgency.
- Complaints follow a structured workflow:
 1. **Pending:** Complaint submitted, awaiting Wing Representative's approval.
 2. **In Progress:** Approved by Wing Representative and visible to Hostel Admin for resolution.
 3. **Escalated:** Admin is unable to resolve and escalates the complaint to the Warden.
 4. **Resolved:** Issue has been fixed and the complaint is closed.
- Warden and Admin can communicate with each other and the student through complaint logs.
- Students receive status updates via automated notifications.

- **Data Elements:**

- **Complaint ID:** A unique identifier for each complaint.
- **Date/Time:** Timestamp of complaint submission.
- **Roll Number:** Used to identify the student and fetch room details.
- **Room Number:** Auto-fetched using the roll number.
- **Category:** Type of complaint.
- **Description:** Detailed explanation of the issue.
- **Priority Level:** Urgency of the complaint.
- **Current Status:** Stage in the complaint lifecycle.
- **Resolution Notes / Logs:** Admin or Warden comments and updates.

- **User Roles:**

- **Student:** Submits complaints using roll number.
- **Wing Representative:** Reviews and approves or rejects complaints.
- **Hostel Administration:** Resolves approved complaints or escalates them if needed.
- **Warden:** Handles escalated complaints and communicates progress.

- **Use Case Scenarios:**

- A student reports a non-functioning fan under the “Electrical” category.
- The Wing Representative approves it, changing the status to *In Progress*.

- The Hostel Admin attempts to resolve the issue. If unsuccessful, the complaint is escalated to the Warden.
- The Warden provides progress updates through logs and coordinates resolution.
- Once resolved, the status is updated to *Resolved*.
- For a leaking tap complaint, the Admin resolves it directly after Wing Rep approval without escalation, and updates the status to *Resolved*.

5.1.2. Boarders List Management

- **Functionality:**

- Allows Hostel Admin to upload and update the boarders list via spreadsheet or manual entry.
- Roll numbers are mapped to room numbers to track student accommodation.
- System can generate vacancy reports to help assess room availability.
- Reports can be shared with the Dean of Student Welfare (DSW) when needed.

- **Data Elements:**

- **Roll Number:** Unique identifier for students.
- **Name:** Student's full name.
- **Room Number:** Assigned room number.
- **Wing/Floor:** Student's floor/wing location.
- **Admission Year:** Year the student joined (can be derived from roll no.).

- **Use Case Scenarios:**

- Hostel Admin uploads a spreadsheet containing new boarders at the start of a semester.
- A student is moved to a different room; the change is updated manually in the system by the Admin.
- A boarder completes their stay and is marked as "Left"; the room is marked as vacant.
- A vacancy report is generated and shared with the DSW upon request.

5.1.3. Notification Module

- **Functionality:**

- The Warden and Hostel Admin can create and push notices or announcements.
- Notifications may contain a title, description, and optionally a PDF attachment (e.g., rulebooks, event posters).
- Students can view all posted notifications on their dashboard.

- **Data Elements:**

- **Notification ID:** Unique identifier for each notification.
- **Title:** Short heading or subject of the notification.
- **Description:** Detailed message body or notice content.
- **PDF Path (optional):** File path to an attached PDF document (if any).

- **Created By:** User ID of the warden/admin who created the notice (foreign key to `users.user_id`).
- **Created At:** Timestamp indicating when the notification was published.

- **Use Case Scenarios:**

- A warden posts a general notice about hostel maintenance; students can view or download the notice as a PDF.
- An admin uploads the updated hostel rules; a notification with a clickable link to the PDF appears on all dashboards.
- A student is assigned to a new room; they get notified with room details.

5.1.4. Wing Assignment and Role Management

- **Functionality:**

- Warden can assign wings to rooms based on room numbers and floor mapping.
- Warden can appoint or update Wing Representatives for each designated wing.
- Ensures that each wing has a single responsible student representative for initial complaint handling.
- Admin and Warden can view and manage the list of Wing Representatives.

- **Data Elements:**

- **Wing ID:** Unique identifier for each wing.
- **Wing Name/Code:** Label used to identify the wing.
- **Room Range:** List or range of room numbers associated with a wing.
- **Wing Representative Roll Number:** Student assigned as Wing Rep.

- **Use Case Scenarios:**

- Warden maps rooms 101–110 to Wing A and assigns a Wing Representative for Wing A.
- A Wing Representative graduates or leaves the hostel; the Warden updates the role with a new student.
- During complaint review, the system identifies the Wing Representative responsible based on the complainant's room.

6. Other Non-functional Requirements

6.1. Performance Requirements

- **System Scalability:** The system should be capable of handling at least 500 concurrent users without experiencing significant latency. The server infrastructure should support horizontal and vertical scaling to accommodate increasing loads.
- **Response Time:** Complaint logging, room change requests, and asset updates should be processed within 2 seconds to ensure a seamless user experience. This requires optimized database queries, caching strategies, and efficient backend processing.
- **Notification Delivery:** Notifications should be delivered within few seconds of status updates to keep users informed in real-time.
- **Database Query Performance:** Queries related to student data, complaints, and room allocations should return results in under few seconds. Database indexing, query optimization, and caching mechanisms should be used to enhance performance.
- **Concurrent Processing:** The backend should support asynchronous and parallel processing for handling tasks such as complaint verification and asset updates without blocking other critical operations.

6.2. Safety and Security Requirements

- **Authentication and Authorization:** The system must enforce a secure login mechanism using JWT(JSON Web Token)-based authentication, with bcrypt for password hashing. Authentication tokens should have expiration policies to minimize security risks.
- **Data Encryption:** Sensitive data, including student personal details and complaints, must be encrypted using AES-256 encryption both at rest and in transit. Secure transport protocols such as TLS should be used for all communications between the client and server.
- **Role-Based Access Control (RBAC):** The system should implement RBAC to restrict access based on user roles. Students should only be able to view and submit complaints, while wing captains can escalate issues. Administrators should have the ability to approve complaints, assign room changes, and update asset records. Unauthorized users must be prevented from accessing restricted functionalities.
- **Protection Against Cyber Threats:** The system should implement security measures against common cyber threats such as SQL injection, cross-site scripting (XSS), and distributed denial-of-service (DDoS) attacks. Regular security audits and vulnerability assessments should be conducted to identify and mitigate potential risks.

7. System Design

7.1. System Architecture

The system architecture of the Hostel Information System is designed with a layered approach to promote separation of concerns, modularity, and scalability. It ensures seamless interaction among various system components while maintaining clarity in data flow and functionality.

Three-Tier Architecture

The application follows a three-tier architectural model, consisting of the Presentation Layer, Application Layer, and Data Layer. Each layer performs distinct roles to collectively deliver a robust and maintainable system.

- **Presentation Layer:** This layer is responsible for interfacing with the users. It provides a dynamic and intuitive user experience for various stakeholders including students, wing representatives, hostel administrators, prefects, and the warden. It handles input validation and communicates with the backend via structured data requests.
- **Application Layer:** The application layer serves as the core of the system's business logic. It handles processing of user inputs, complaint routing workflows, role-based access control, escalation logic, and communication between the user interface and the database.
- **Data Layer:** This layer is responsible for persistent storage and retrieval of data. It includes entities such as students, rooms, complaints, logs, and notifications. It ensures data integrity, consistency, and efficient query handling to support system operations.

7.2. Data Flow Diagram (DFD)

This section presents the context-level (Level 0) Data Flow Diagram of the Hostel Information System. It illustrates how data flows between external entities (such as students, wing representatives, hostel admins, and the warden) and the system. The system is represented as a single process that handles all the operations internally.

The diagram shows major data flows such as:

- Students submit complaints and receive notifications.
- Wing Representatives review and approve or reject complaints.
- Hostel Admin updates complaint statuses, escalates unresolved cases, and manages boarders.
- The Warden handles escalated complaints and adds resolution progress.

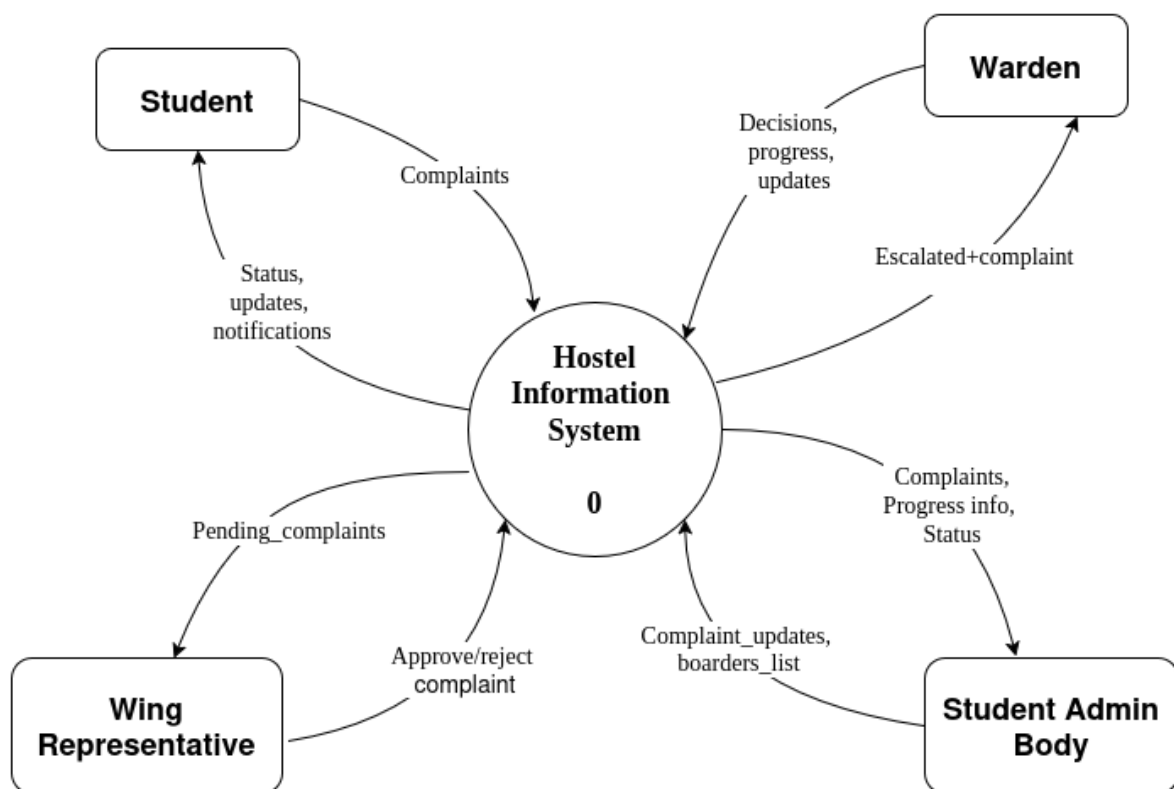


Figure 1: Level 0 Data Flow Diagram

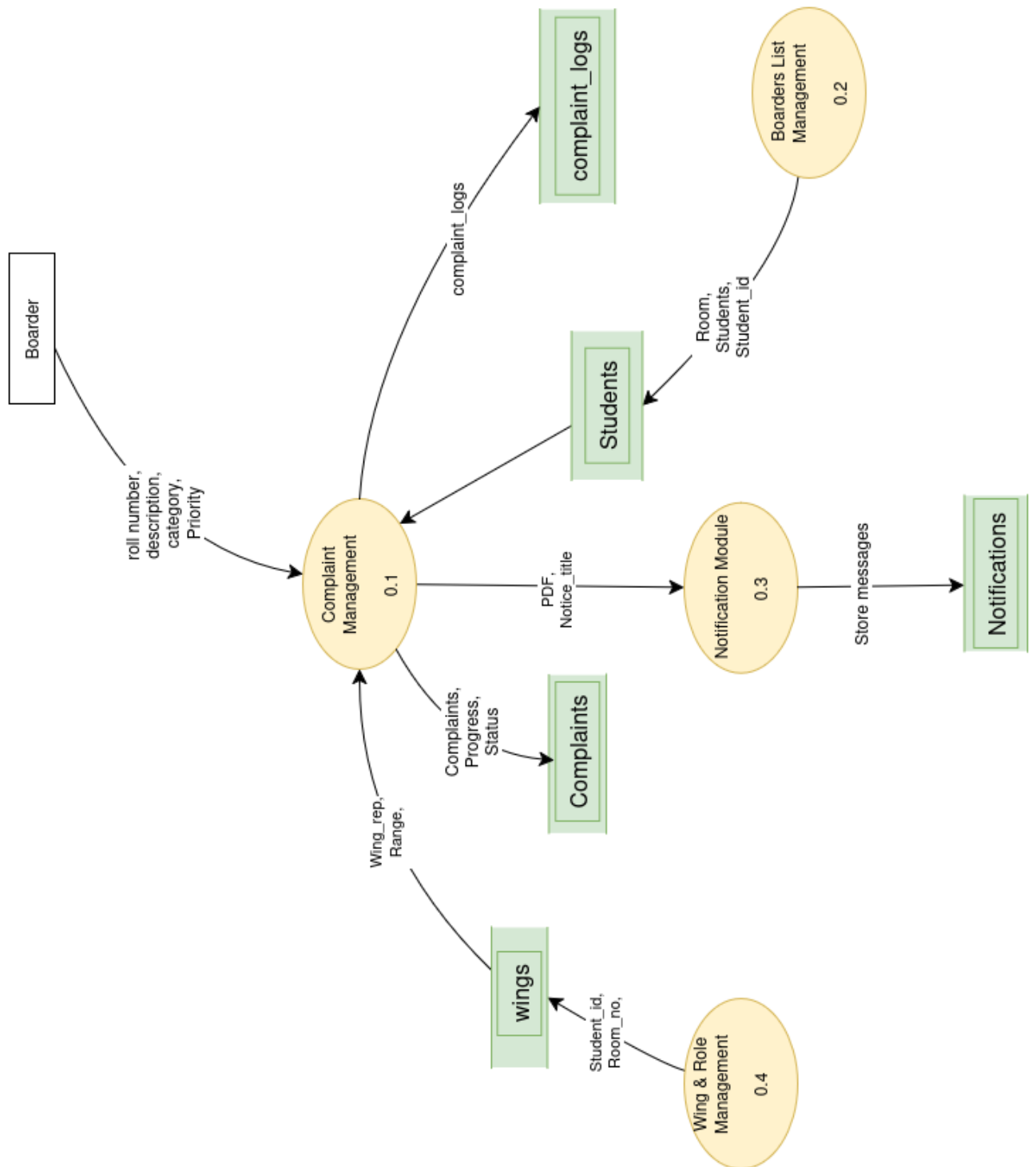


Figure 2: Level 1 Data Flow Diagram

7.3. Database Design

The system schema includes key tables to support hostel operations, including students, rooms, complaints, escalations, and user roles. The schema has been visualized using an ER diagram and a structured data dictionary.

- The DB Flow Diagram illustrates relationships between key entities such as students, rooms, complaints, and users , which is shown latter.
- A detailed data dictionary outlines each table, column types, constraints, and descriptions are shown below:

Table: rooms

Field	Type	Description
room_id	INT, AUTO INCREMENT, PRIMARY KEY	Unique identifier for each room.
room_number	VARCHAR(10), UNIQUE	Unique room number.
capacity	INT	Maximum capacity of the room.
current_occupants	INT, DEFAULT 0	Number of students currently in the room.

Table: students

Field	Type	Description
student_id	VARCHAR(20), PRI- MARY KEY	Student roll number as ID.
name	VARCHAR(100)	Full name of the student.
email	VARCHAR(100), UNIQUE	Unique email address.
contact_number	VARCHAR(20)	Contact phone number.
room_number	VARCHAR(10), FOR- EIGN KEY	Room assigned to the student.

Table: complaints

Field	Type	Description
complaint_id	INT, AUTO_INCREMENT, PRIMARY KEY	Unique ID for each complaint.
student_id	VARCHAR(20), FOREIGN KEY	Student who submitted the complaint.
category	ENUM(...)	Complaint category.
description	TEXT	Description of the issue.
submitted_at	DATETIME	Timestamp when complaint was submitted.
status	ENUM(...)	Complaint resolution status.
priority	ENUM(...)	Severity of the complaint.
approval_status	ENUM(...)	Approval status for escalation.
assigned_to	VARCHAR(20), FOREIGN KEY	Person responsible for handling.

Table: assets (to be used in the future)

Field	Type	Description
asset_id	INT, AUTO_INCREMENT, PRIMARY KEY	Unique asset ID.
type	VARCHAR(50)	Type/category of the asset.
location	VARCHAR(100)	Where the asset is located.
current_condition	ENUM(...)	Current operational status.
purchase_date	DATE	When the asset was purchased.
replacement_date	DATE	Expected replacement date.

Table: maintenance_logs (to be used in future)

Field	Type	Description
log_id	INT, AUTO_INCREMENT, PRIMARY KEY	Log entry ID.
asset_id	INT, FOREIGN KEY	Associated asset.
requested_by	VARCHAR(20), FOREIGN KEY	Student who requested maintenance.
request_date	DATETIME	Date and time of request.
status	ENUM(...)	Maintenance status.
notes	TEXT	Additional notes or updates.

Table: lost_items (to be used in the future)

Field	Type	Description
lost_item_id	INT, AUTO_INCREMENT, PRIMARY KEY	Unique lost item ID.
student_id	VARCHAR(20), FOREIGN KEY	Student who reported the item.
item_description	VARCHAR(255)	Description of the lost item.
incident_date	DATETIME	When the item was lost.
status	ENUM(...)	Current status of the item.

Table: notifications

Field	Type	Description
notification_id	INT, AUTO_INCREMENT, PRIMARY KEY	Unique notification ID.
title	VARCHAR(255), NOT NULL	Title of the notification.
description	TEXT, NOT NULL	Description or body content of the notification.
pdf_path	VARCHAR(255)	File path to an optional attached PDF.
created_by	INT, FOREIGN KEY	Warden (users.user_id) who created the notification.
created_at	DATETIME, DEFAULT CURRENT_TIMESTAMP	Time when the notification was posted.

Table: users

Field	Type	Description
user_id	INT, AUTO_INCREMENT, PRIMARY KEY	Unique user ID.
name	VARCHAR(100)	Full name of the user.
email	VARCHAR(100), UNIQUE	Login email.
password	VARCHAR(255)	Hashed password.
role	ENUM(...)	Role such as warden, prefect, etc.

Table: wings

Field	Type	Description
wing_id	INT, AUTO_INCREMENT, PRIMARY KEY	Unique wing ID.
wing_name	VARCHAR(50), UNIQUE	Name of the wing.
representative_id	VARCHAR(20), FOREIGN KEY	Wing representative.
room_start	VARCHAR(10)	Starting room number.
room_end	VARCHAR(10)	Ending room number.

Table: warden_complaints

Field	Type	Description
warden_complaint_id	INT, AUTO_INCREMENT, PRIMARY KEY	Unique ID for escalated complaints.
complaint_id	INT, FOREIGN KEY	Linked complaint.
student_id	VARCHAR(20), FOREIGN KEY	Student who filed the original complaint.
category	ENUM(...)	Complaint type.
description	TEXT	Complaint description.
submitted_at	DATETIME	When it was submitted.
status	ENUM(...)	Resolution status.
priority	ENUM(...)	Complaint priority.
action_taken	TEXT	Actions taken by warden.

Table: complaint_logs

Field	Type	Description
log_id	INT, AUTO_INCREMENT, PRIMARY KEY	Unique log ID.
warden_complaint_id	INT, FOREIGN KEY	Associated warden complaint.
update_text	TEXT	Progress update text.
updated_at	TIMESTAMP	When the update was logged.



Figure 4: DataBase Tables

8. System Implementation

8.1. Tech Stack

The online Hostel Information System leverages a robust and versatile technology stack, carefully selected to ensure optimal performance, scalability, and reliability.

- **Frontend:**

The frontend is developed using **React**, a popular JavaScript library for building interactive and responsive user interfaces. React enables the creation of reusable components, facilitating efficient development and maintenance of the user-facing application.

- **Backend:**

The backend is built with **Node.js** and **Express**, providing a fast, scalable, and event-driven server environment which provides excellent performance for handling multiple simultaneous requests. To handle HTTP requests and perform API calls, **Axios** is used. For managing file uploads, such as documents or images related to complaints or boarders, **Multer** middleware is integrated seamlessly with Express.

- **Database:**

MySQL is chosen as the relational database management system (RDBMS) to store and manage data efficiently, mainly due to its **ACID** compliance — which stands for:

- **Atomicity:** ensures that each transaction is all-or-nothing.
- **Consistency:** ensures that a transaction brings the database from one valid state to another.
- **Isolation:** ensures that concurrent transactions do not interfere with each other.
- **Durability:** ensures that once a transaction is committed, it remains so, even in the case of a system failure.

These properties make MySQL an ideal choice for handling the complex data structures associated with hostel management, including complaints, user information, and administrative records.

8.2. Hosting Service

The system will be deployed on the university’s internal server infrastructure, where the current hostel website is also hosted. Hosting on the university server ensures greater control, security, and integration with existing campus systems, while avoiding additional external hosting costs.

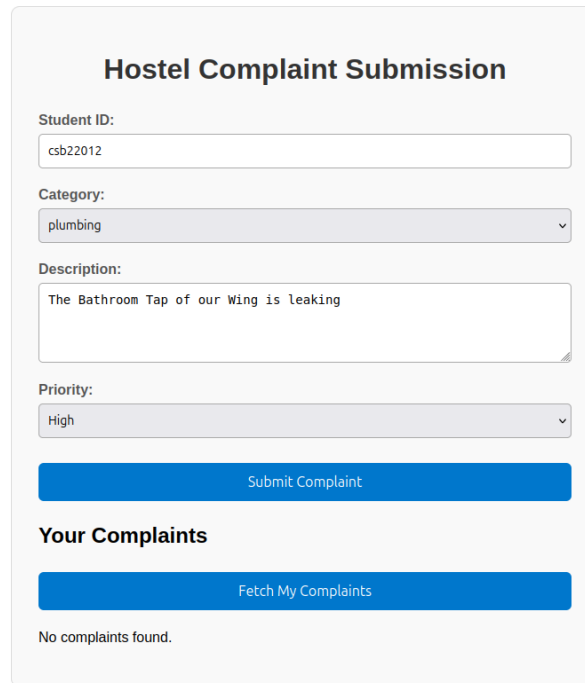
8.3. User Interface Design

The user interface design follows a user-centric approach, catering to the diverse needs of students, wing representatives, prefects, hostel administrators, and the warden. This design philosophy is aimed at enhancing the overall user experience and ensuring that the application is accessible, intuitive, and easy to navigate for all user roles.

This section showcases the Complaint Management System workflow through a series of UI snapshots, arranged in the order of typical user interactions.

8.3.1. Student Raises a Complaint

The student begins by entering their roll number and complaint details. Once submitted, they can view the current status of their complaint on the same page.



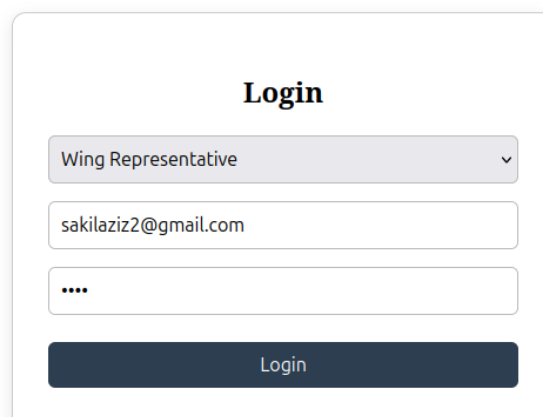
The form is titled "Hostel Complaint Submission". It contains the following fields and controls:

- Student ID:** A text input field containing "csb22012".
- Category:** A dropdown menu with "plumbing" selected.
- Description:** A text area containing "The Bathroom Tap of our Wing is Leaking".
- Priority:** A dropdown menu with "High" selected.
- Submit Complaint:** A blue button.
- Your Complaints:** A section header.
- Fetch My Complaints:** A blue button.
- No complaints found.** A message displayed below the button.

Figure 5: Student Complaint Submission and Status View

8.3.2. Warden/Admin Login

Hostel authorities log in through a role-selection interface. The warden or admin must choose their role and provide valid credentials to proceed.



The form is titled "Login". It contains the following fields and controls:

- Role:** A dropdown menu with "Wing Representative" selected.
- Email:** A text input field containing "sakilaziz2@gmail.com".
- Password:** A text input field with masked characters "....".
- Login:** A dark blue button.

Figure 6: Warden/Admin Role-Based Login Page

8.3.3. Wing Representative Dashboard

The wing representative views all newly submitted complaints and decides whether to approve or reject them.

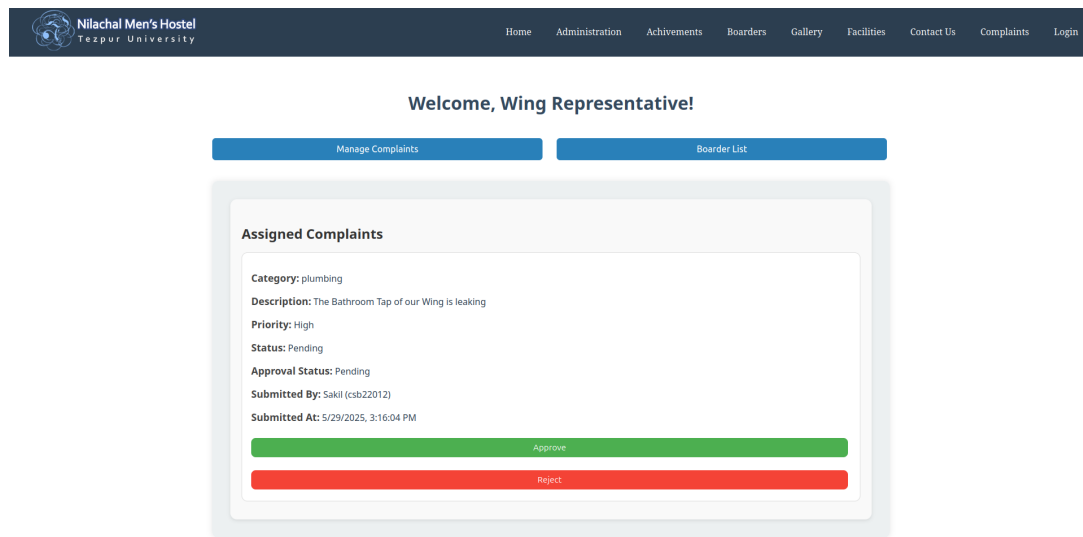


Figure 7: Wing Representative Dashboard - Complaint Approval

8.3.4. Prefect Dashboard

Once a complaint is approved by the wing rep, it reaches the prefect. The prefect can either resolve the issue directly or escalate it to the warden if needed.

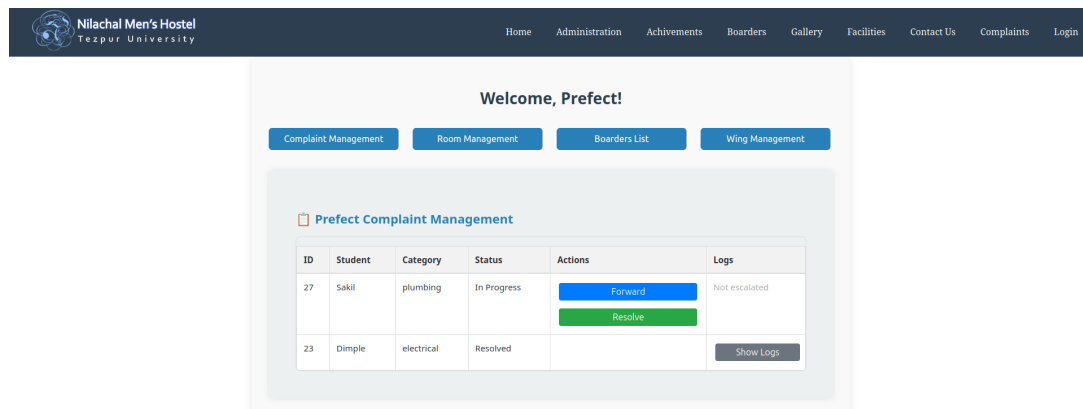


Figure 8: Prefect Dashboard - Resolve or Escalate Complaint

8.3.5. Warden Dashboard and Logs

For escalated complaints, the warden reviews the issue, performs necessary actions, and updates the status through logs visible to the prefect.

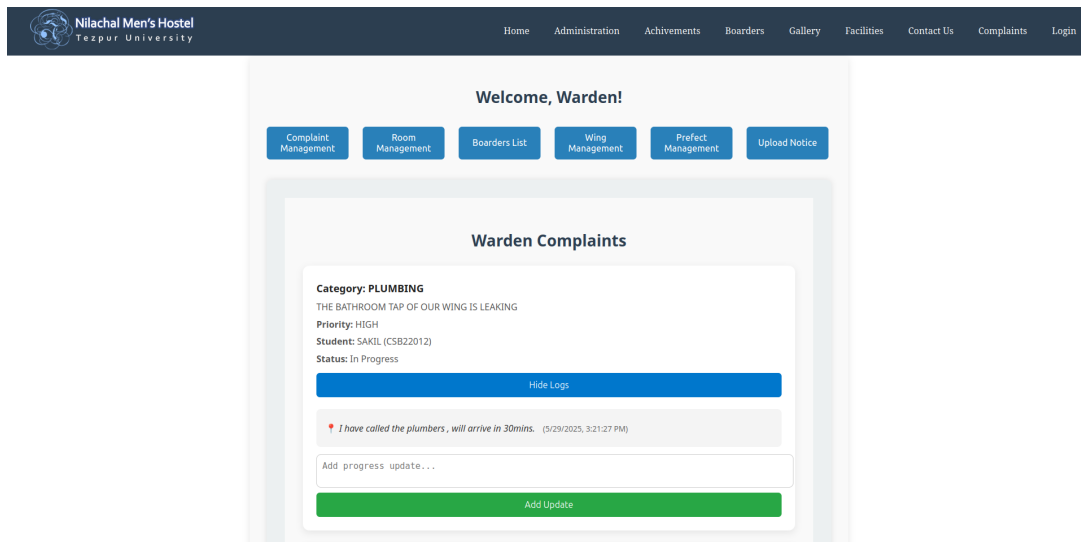


Figure 9: Warden Dashboard with Complaint Logs

8.3.6. Final Complaint Resolution by Prefect

After the warden has addressed the complaint, the prefect verifies and marks the complaint as resolved, closing the loop.

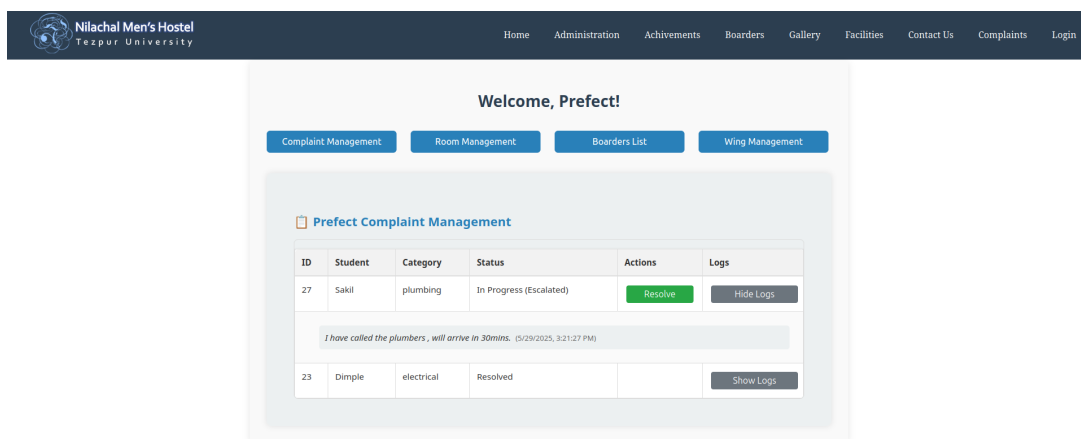


Figure 10: Prefect Marks Complaint as Resolved

8.4. Room and Boarder Management

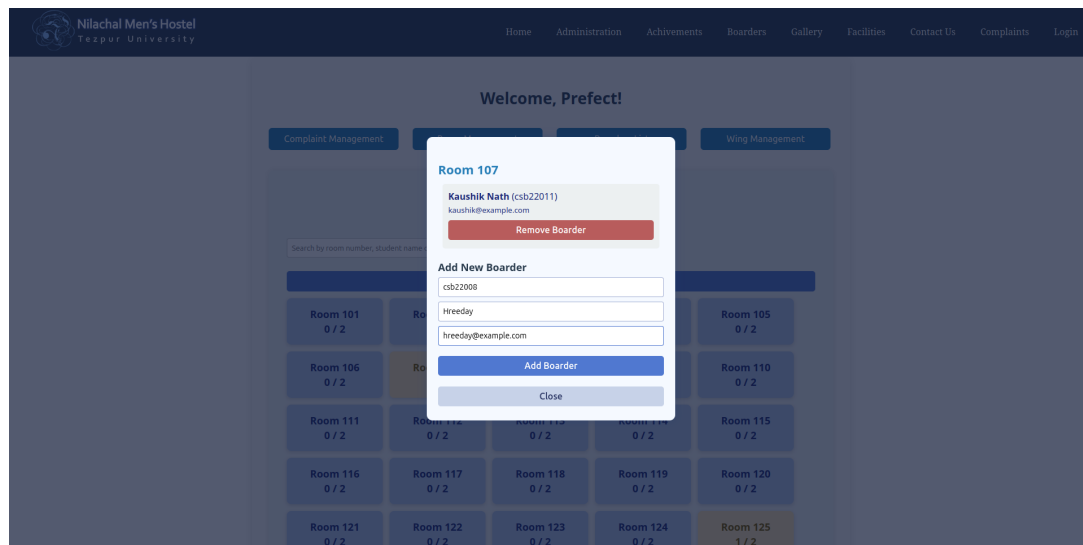


Figure 11: Room Management Actions: On selecting a room, a modal opens showing current boarders along with options to add or remove boarders. This interface is accessible to both prefects and the warden.

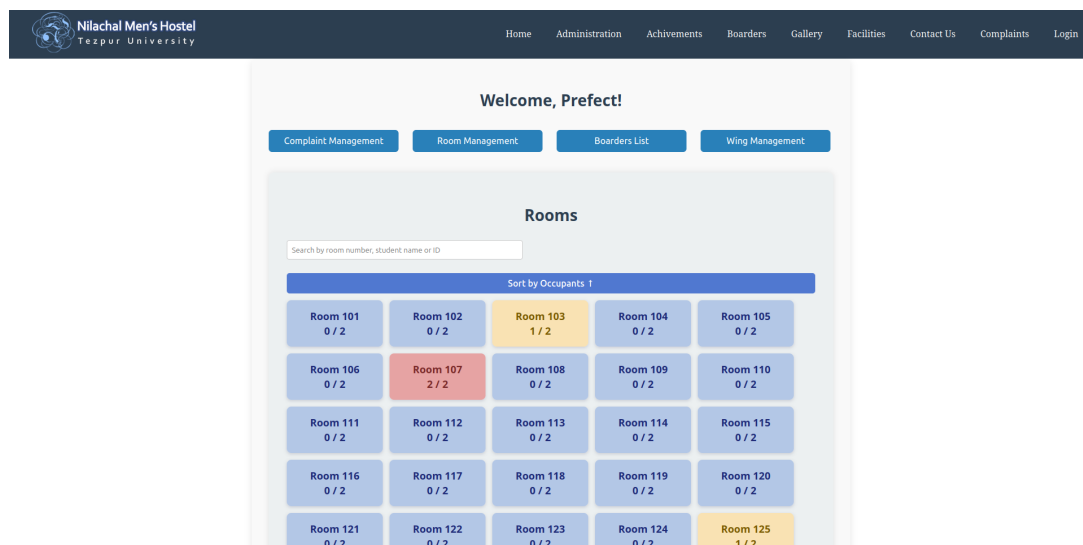


Figure 12: Room Status View: Rooms are color-coded based on availability—blue for 2 vacant seats, yellow for 1 vacant seat, and red for no vacancy. Prefects and the warden can click on any room to manage boarders and also sort the rooms based on availability.

8.5. Wing Management

The screenshot shows the 'Wing Management' section of the Nilachal Men's Hostel web application. The header includes the university logo and navigation links. A 'Welcome, Warden!' message is displayed above a row of buttons: Complaint Management, Room Management, Boarders List, Wing Management, Prefect Management, and Upload Notice. The 'Wing Management' form contains input fields for Wing Name, Representative Student_ID, Start Room, and End Room, followed by an 'Add Wing' button. Below this, two existing wings are listed: 'Wing: Deewan' with Rep ID: csb22001 and Rooms: 237 - 272, and 'Wing: Sukapha Wing' with Rep ID: csb22015 and Rooms: 101 - 136. Each entry has an 'Edit' button.

Figure 13: Wing Creation and Assignment: The prefect or warden can add new wings and assign a Wing Representative. Each wing is associated with a range of room numbers. Once a wing is added, its corresponding rooms become active and manageable in the system. After a wing is added, the rooms in its assigned range are unlocked and visible in the boarder management view, enabling boarder allocation and complaint tracking.

8.6. Notice Upload

The screenshot shows the 'Upload Notice' form within the Nilachal Men's Hostel web application. The header and navigation are consistent with the previous figure. The 'Upload Notice' form has three text input fields: 'Admin Body 2025-26 Selection', 'Interview for the selection of Admin Body 2025-26 schedule', and a file upload field with a 'Browse...' button and the filename 'Admin Body Notification_NMH.pdf'. An 'Upload' button is at the bottom of the form.

Figure 14: Notice Upload by Warden: The warden can upload a new notice by providing a title, a short description, and a PDF file. This ensures that important announcements such as maintenance schedules, rules, and events are formally documented and communicated.

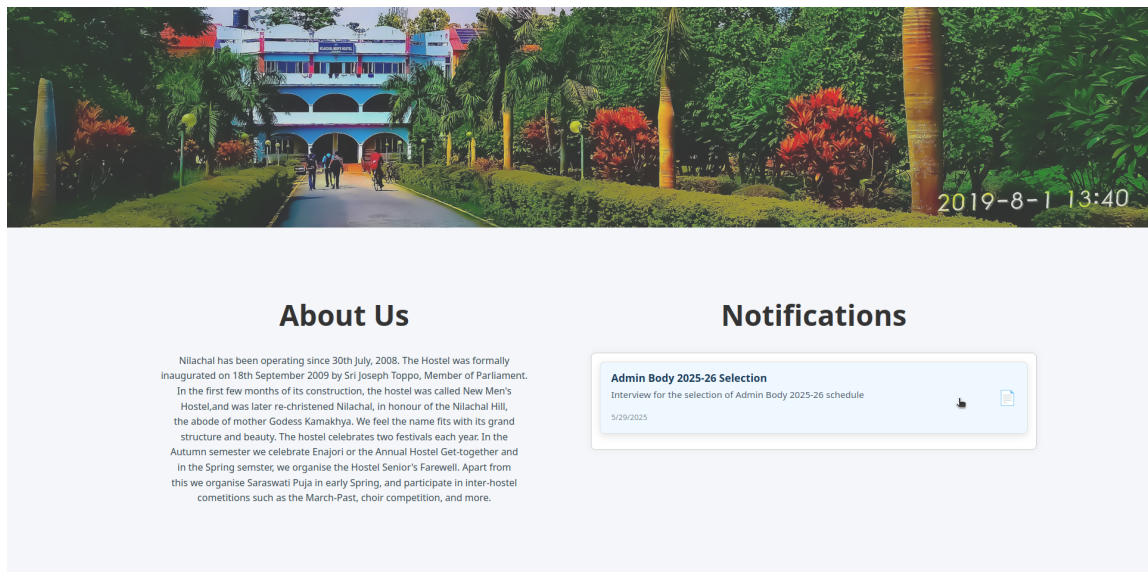


Figure 15: Public Notice Board on Homepage: All uploaded notices are displayed on the hostel's public landing page, where students and visitors can view or download the PDF files. Each notice includes its title and a brief description for quick readability.

9. System Testing

In the testing phase of the Hostel Information System's development, priority was placed on ensuring reliability, functionality, and performance across all modules. This chapter outlines the comprehensive testing strategy adopted to verify the robustness of the system, including unit testing, validation testing, functional testing, and database testing. These evaluations were essential in ensuring the integrity of individual components, seamless system behavior, stable data management, and responsiveness under real-world usage scenarios.

9.1. Unit Testing

Unit testing involved the detailed examination of individual program components, such as controllers, services, and utility functions. Each unit was tested using predefined inputs and expected outputs to ensure correct logic execution. For example, validation functions for complaint submission, user role-based access, and file handling utilities were tested in isolation to ensure robust and predictable behavior. This step promoted code reliability and minimized the chances of runtime errors.

9.2. Validation Testing

Validation testing focused on verifying that the system adhered to the specified requirements and met user expectations. Throughout development, features were regularly reviewed and tested against client needs. The complaint escalation workflow (from wing representative to prefect and then to warden), boarder management system, and notice upload functionality were all validated to ensure they aligned with the project's intended objectives. Results demonstrated that the system fulfilled its core purposes effectively.

9.3. Functional Testing

Functional testing ensured that each system module performed its intended operations correctly. The following modules were tested with both valid and invalid input scenarios:

- Student complaint submission and multi-level status tracking
- Approval and escalation logic for wing representatives, prefects, and wardens
- Admin panel for managing boarders and uploading notices
- Public notice board display on the hostel landing page
- Room availability view with color-coded capacity indicators

All functions were verified for expected behavior, proper error handling, and user feedback mechanisms.

9.4. Database Testing

Database testing validated the system's interaction with MySQL. Operations tested included data creation, retrieval, updating, and deletion across major tables (students, complaints, logs, notices, and rooms). Dummy data sets were used to simulate real hostel scenarios. Tests ensured:

- Consistent data integrity during approval and escalation
- Accurate complaint log updates with timestamps

- File metadata storage for notices and proper retrieval
- Referential integrity between boarders, rooms, and complaints

The database schema and queries were optimized for efficiency and reliability.

The system sustained acceptable response times and remained stable during tests.

10. Conclusion and Future Work

The development of the Hostel Information System successfully addressed critical aspects of hostel administration, offering a centralized platform for managing complaints, boarder information, room occupancy, and notice dissemination. The system streamlines communication between students, wing representatives, prefects, and wardens, ensuring transparent and efficient workflows. Its modular design, role-based access, and integration of file upload and tracking features significantly enhance hostel operations and accountability.

10.1. Conclusion

The Hostel Information System proved to be a robust and effective solution for simplifying hostel management tasks. It enabled stakeholders at all levels—students, representatives, and wardens—to participate in a structured workflow that enhances issue resolution, room assignment visibility, and notice sharing. The system’s responsive interface and strong backend integration helped deliver a smooth user experience and reliable functionality during testing and pilot usage.

10.2. Future Work

While the current system meets core requirements, several enhancements are planned to improve functionality and user convenience. Some supporting database tables have already been created but were not integrated due to time constraints.

- **Asset Tracking and Maintenance Logging:** The `assets` and `maintenance_logs` tables were provisioned to manage hostel assets and log maintenance requests. These features will be implemented in future versions to support preventive maintenance and inventory tracking.
- **Interconnected Lost and Found Module:** A centralized system to report and track lost items across hostels is planned. Hostel admins can moderate entries while students can view reports from all hostels.
Note: The `lost_items` table exists but the module was not implemented due to time constraints.
- **Room Change Request System:** An online interface to request room changes with automated approval flows involving hostel staff.
- **Integrated Payment Gateway:** A secure payment system for mess dues using UPI, cards, or net banking, with instant receipt generation.
- **Analytics Dashboard:** A visual dashboard showing real-time stats on complaints, occupancy, notices, and activity logs.

These additions will help the Hostel Information System evolve into a more scalable and feature-rich platform for student accommodation management.

Bibliography

- MySQL 8.0 Reference Manual, Oracle Corporation,
<https://dev.mysql.com/doc/refman/8.0/en/>
- Node.js Documentation, OpenJS Foundation,
<https://nodejs.org/en/docs>
- Node.js - Introduction, GeeksforGeeks,
<https://www.geeksforgeeks.org/nodejs/>
- React – A JavaScript Library for Building User Interfaces, Meta (Facebook), <https://react.dev/>
- HTML, CSS, and JavaScript – MDN Web Docs, Mozilla Developer Network,
<https://developer.mozilla.org/>
- What is DFD (Data Flow Diagram)?, GeeksforGeeks,
<https://www.geeksforgeeks.org/what-is-dfddata-flow-diagram/>
- Software Testing Basics, GeeksforGeeks,
<https://www.geeksforgeeks.org/software-testing-basics/>
- LaTeX Project: Official Documentation,
<https://www.latex-project.org/help/documentation/>
- Rajib Mall, *Fundamentals of Software Engineering*, PHI Learning, 5th Edition, 2018.
- Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, Pearson, 7th Edition, 2016.