Software Requirements Specification

for

Hostel Management System

Version 0.1

Prepared by

Group Number: 03

Kasireddy Koti Reddy	B230373CS	CS01
Katta Abhiram	B230669CS	CS01
Amit Kumar	B230801CS	CS01
Dumpuru Sridhar	B230093CS	CS01

Instructor:	Dr. Pranesh Das
Course:	Database Management System
Lab Section:	--
Teaching Assistant:	--
Date:	28 / 02 / 2025

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Kasireddy Koti Reddy	This is the first version of SRS document for Hostel Management System (MHS). It gives basic idea of what HMS is for and how it is implemented.	28/02/2025

1 Introduction

1.1 Document Purpose

This **Software Requirements Specification (SRS)** document outlines the requirements for the **Hostel Management System (HMS)**. The system is designed to **streamline hostel administration** by automating key operations such as **room allocation, fee management, visitor tracking, complaint handling, maintenance requests, and staff management**. It ensures efficient data handling, retrieval, and modification through a **MySQL database**, with a **web-based interface** for both students and administrators.

This document specifically focuses on **the database and operational aspects** of the system, ensuring that all functionalities related to hostel management are well-defined. The system will have **two primary interfaces**:

- **User Interface for students:** To check room allocation, pay fees, submit complaints, and track visitors.
- **Admin Interface for hostel staff:** To manage room assignments, fee payments, complaints, and staff records.

1.2 Product Scope

The **Hostel Management System (HMS)** is designed to **digitize and streamline hostel operations**, reducing manual work and improving efficiency. The system aims to provide a **centralized platform** for managing hostel-related tasks such as **room allocation, fee management, visitor tracking, and complaint handling**. By using **a web-based interface with a MySQL database**, the system ensures **secure, real-time access to hostel records** for both students and administrators.

The **key benefits** of the system include:

- **Automation** of routine hostel management tasks, minimizing paperwork.
- **Transparency** in fee payments, room allocations, and complaint tracking.
- **Convenience** for students to access hostel services anytime.
- **Efficient monitoring** of visitor entries and staff activities.

1.3 Intended Audience and Document Overview

This **Software Requirements Specification (SRS)** document is intended for:

- **Developers** – To understand the system requirements and implement the database and web-based interface.
- **Clients (Hostel Administrators & Management)** – To review the system’s functionality and ensure it meets their needs.
- **Professors/Evaluators** – To assess the project’s scope, requirements, and implementation approach.
- **Testers** – To verify that the system functions as expected and meets all specified requirements.

1.4 Definitions, Acronyms and Abbreviations

- DBMS – Database Management System
- HMS – Hostel Management System
- GUI – Graphical User Interface
- MySQL – Structured Query Language-based relational database management system
- PHP – Hypertext Preprocessor, a scripting language used for web development
- SQL – Structured Query Language, used for database operations
- UI – User Interface
- UX – User Experience
- Admin – Administrator responsible for managing hostel operations
- User – A student or staff member interacting with the system
- HTTP – Hypertext Transfer Protocol, used for communication between the web browser and the server
- HTTPS – Secure Hypertext Transfer Protocol, an encrypted version of HTTP
- XAMPP – A local server software package that includes Apache, MySQL, PHP, and Perl
- CRUD – Create, Read, Update, Delete (basic database operations)

1.5 Document Conventions

1. Formatting
    - Font: Arial MT, size 12.
    - Headings: Bold, numbered (e.g., 1, 1.1, 1.1.1).
    - Line spacing: 1.5, margins: 1 inch.
2. Numbering
    - Sections, figures, and tables are numbered sequentially (e.g., **Figure 1, Table 2**).
3. Language & Clarity
    - Formal, clear, and concise language is used.
    - Slang, contractions, and ambiguity are avoided.

1.6 References and Acknowledgments

References:

These are documents or web pages that provide additional details about standards, technologies, or guidelines followed in this project:

1. **MySQL Documentation** – Official MySQL reference for database design and queries.
    a. https://dev.mysql.com/doc/
2. **PHP Official Documentation** – Guidelines on PHP scripting for web development.
    a. https://www.php.net/manual/en/
3. **XAMPP Documentation** – Instructions for setting up a local development server.
    a. https://www.apachefriends.org/
4. **IEEE SRS Standard (IEEE 830-1998)** – Guidelines followed for writing this document.
    a. https://ieeexplore.ieee.org/document/720574

Acknowledgments:

We would like to acknowledge:

- **Professors and Mentors** – For providing guidance on database and web development.
- **Online Learning Platforms** (such as W3Schools, GeeksforGeeks) – For tutorials and explanations related to PHP, MySQL, and web technologies.
- **Open-Source Developers and Communities** – For providing valuable insights into hostel management systems and database design best practices.

2 Overall Description

2.1 Product Overview

The Hostel Management System (HMS) is a new, self-contained software solution designed to streamline and automate hostel-related operations in schools, universities, and other institutions. The system digitizes and simplifies tasks such as room allocation, fee management, visitor tracking, complaints, maintenance requests, staff management, and leave tracking.

This system replaces the traditional manual record-keeping process, which is prone to errors, inefficiencies, and delays. By using a centralized database (MySQL) and a web-based interface (PHP, HTML, CSS, JavaScript), the HMS ensures efficient data storage, retrieval, and management.

The system consists of two main interfaces:

1. Admin Interface – Used by hostel administrators to allocate rooms, manage payments, handle maintenance requests, and oversee staff.
2. User Interface (Students & Staff) – Allows students to check room details, report issues, and track complaints, while staff members can submit leave requests and monitor hostel operations.

The HMS is a standalone system that can be hosted on a local server (XAMPP) or a remote web server. It interacts with:

- Users (Students, Wardens, Staff, and Admins)
- Database (MySQL) for data storage and retrieval
- Web Technologies (PHP, JavaScript) for front-end and back-end processing

2.2 Product Functionality:

The **Hostel Management System (HMS)** is designed to automate and streamline various hostel operations, improving efficiency and reducing manual efforts. The system includes several key functionalities:

1. **Room Allocation**: Automates room assignments based on student preferences and room availability, with manual override options for admins.

2. **Hostel Fee Payment**: Allows students to make online fee payments, with reminders, payment tracking, and receipts.

3. **Visitor Tracking**: Records and tracks visitors entering the hostel, including their details, approval, and duration of stay.

4. **Complaint Management**: Facilitates the filing and tracking of complaints by students and staff, with a ticketing system for issue resolution.

5. **Maintenance Requests**: Enables students and staff to request repairs or maintenance, with status tracking and updates on resolution.

6. **Room Occupancy Report**: Generates real-time and historical reports on room

occupancy, helping admin track vacant and occupied rooms.

7. **Wardens and Staff Management**: Manages staff profiles, assignments, attendance, and performance, streamlining staff operations.
8. **Staff Leave Register**: Tracks leave requests from hostel staff, with approval, leave balance management, and leave history.
9. **Student Leave Register**: Allows students to apply for leave, with admin approval, leave balance tracking, and status notifications.

Overall, the HMS automates hostel tasks, improving operational efficiency, enhancing user experience, and ensuring seamless management of the hostel environment.

2.3 Design and Implementation Constraints

The Hostel Management System (HMS) has specific design and implementation constraints that must be adhered to during development. These constraints include hardware limitations, software and technology requirements, programming standards, and security considerations.

Hardware Limitations:

To ensure optimal performance and reliability, the system requires the following minimum hardware specifications:

- **Server Requirements**:
  - **Processor**: Dual-core processor (Intel i3 or equivalent)
  - **Memory**: 4 GB RAM or higher
  - **Storage**: 100 GB SSD or higher
  - **Network**: Minimum **100 Mbps bandwidth**
- **Local Development Environment**:
  - **XAMPP/WAMP** for backend testing
  - **MySQL** for database management

These specifications ensure smooth system operation and efficient database transactions, even with multiple simultaneous users.

2.1.1.1 2.3.2 Software and Technology Constraints

- **Design Methodology**: The **COMET (Collaborative Object Modeling and Enterprise Transformation)** method will be used to ensure a structured and scalable software design approach.

    - **Reference**: Gomaa, H. (2005). *Designing Software Product Lines with UML*. Addison-Wesley.
- **Modeling Language**: The **Unified Modeling Language (UML**) will be utilized to design system workflows, including use case diagrams, class diagrams, and sequence diagrams.
    - **Reference**: Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide (2nd Edition)*. Addison-Wesley.
- **Programming Languages & Tools**:
    - **Frontend**: HTML, CSS, JavaScript (React.js for enhanced UI)
    - **Backend**: PHP (for request processing and database interaction)
    - **Database**: MySQL (for storing hostel and user data)
    - **Development Tools**:
        - **IDE**: VS Code / PhpStorm
        - **Version Control**: Git (GitHub)
        - **API Standards**: RESTful API (for future scalability)

Security Considerations:

To protect sensitive user and administrative data, the system will implement the following security measures:

- OWASP Security Guidelines: The system will adhere to OWASP best practices to mitigate threats such as **SQL injection**, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).
- Data Encryption: User passwords will be hashed using bcrypt or MD5 to prevent unauthorized access.
- Role-Based Access Control (RBAC): Access to administrative features will be restricted based on user roles to prevent unauthorized modifications.

2.4 Assumptions and Dependencies

This section outlines the key assumptions and dependencies that could impact the requirements of the **Hostel Management System (HMS)**.

2.4.1 Assumptions:

1. **Stable Internet Connectivity:** It is assumed that end users will have reliable internet.

2. **Consistent User Roles and Permissions:** The system assumes a predefined set of user roles (**Administrator, Warden, Student, Maintenance Staff**) with fixed permissions.

3. **Hosting and Server Infrastructure:** It is assumed that the HMS will be deployed on a cloud-based infrastructure (AWS, Azure, or Google Cloud).

4. **Third-Party Service Availability:**

    - **Payment gateways** (e.g., Stripe, PayPal) for hostel fee transactions.

5. **Regulatory Compliance Requirements Remain Unchanged:**
    - The system is designed to comply with **GDPR** and **local hostel management regulations** regarding data privacy and record-keeping.
    - Any changes in legal or regulatory requirements may necessitate modifications in system functionality.

2.4.2 Dependencies:

1. **External Software Components:**
    - The project depends on existing software components, including:
        - **Database Management System:** PostgreSQL/MySQL
        - **Authentication Frameworks:** OAuth 2.0 for user authentication
        - **Frontend Libraries:** React.js

2. **Development Frameworks and Tools:**
    - The system will be developed using PHP and development tool XAMPP) for the backend.
    - Any major changes in these frameworks (e.g., deprecations, security vulnerabilities) may impact development.

3. **Hardware and Network Infrastructure:**
    - The project relies on an adequate **server environment** with sufficient CPU, memory, and storage.
    - Any issues with hardware procurement or hosting service availability could delay deployment.

4. **Client-Side Device Compatibility:**
    - The system assumes that users will access the platform using **modern browsers and up-to-date mobile or computer devices**.
    - Users with outdated hardware or software may face compatibility issues.

These **assumptions and dependencies** establish the conditions under which the **Hostel Management System** will be developed and deployed.

3 Specific Requirements

3.1 External Interface Requirements

The Hostel Management System (HMS) will provide two types of user interfaces:

Admin Interface:

Purpose: Used by hostel administrators to manage rooms, fees, complaints, and staff.

Access Method: Web-based interface (accessible via a browser).

Login: Administrators will log in using their credentials.

Functionalities:

- Manage room details and allocation status.
- Process hostel fee payments.
- Handle maintenance requests and complaints.
- Track visitor entries.
- Manage staff records.

Navigation: Menu-based navigation with options for **Room Management, Fee Processing, Complaints, Staff Management, and Visitor Tracking**.

Student/User Interface

Purpose: Used by students to check room allocation, pay fees, submit complaints, and track visitors.

Access Method: Web-based interface (accessible via a browser).

Login: Students will sign up and log in using their email and password.

Functionalities:

- View room details and allocation status.
- Pay hostel fees online.
- Submit maintenance requests and complaints.
- Track visitor entries.
- View personal profile and hostel rules.

Navigation: Menu-based navigation with options for **Room Status, Fee Payment, Complaints, and Visitors**.

3.1.1 Hardware Interfaces

User Requirements:

- **Device**: Desktop, laptop, tablet, or smartphone with a modern web browser.
- **Processor**: Minimum Intel i3 or equivalent.
- **RAM**: 2 GB (students); 4 GB (administrators).
- **Network**: Active internet connection for accessing the web application.

Server Requirements:

- **Processor**: Dual-core CPU (e.g., Intel i5 or equivalent).
- **RAM**: 4 GB (minimum).
- **Storage**: 10 GB or more for database and logs.
- **Network**: Reliable internet connection for hosting and user access.

3.1.2 Software Interfaces

The system requires the following software components:

Frontend Technologies: HTML, CSS, JavaScript
Backend Technologies: PHP
Database: MySQL
Web Server: Apache (included in XAMPP)
Development Tools: XAMPP, VS Code, phpMyAdmin

- **Frontend Interface**:
  The web application will provide two primary interfaces:
  - **Student Interface**: Allows students to check room allocations, pay fees, submit complaints, and track visitors.
  - **Admin Interface**: Enables hostel staff to manage room assignments, fee payments, complaints, and staff records.

  The user interfaces will be accessible through a modern web browser (e.g., Chrome, Firefox, Edge).

- **Backend Interface**:
  - The backend will handle all business logic and database interactions.
  - RESTful APIs will facilitate communication between the frontend and the backend.

- **Database Interface**:
  - The system will use a MySQL database for data storage and management.
  - Database queries will be executed to handle operations such as room allocation, fee management, and tracking complaints.

The system will utilize a web server to host the backend logic and serve the frontend interface.

3.2 Functional Requirements

The system meets the following functional requirements:

1. **Room Allocation**:
    a. Allocate rooms to students based on predefined eligibility criteria.
    b. Maintain records of allocated and available rooms.
2. **Hostel Fee Management**:
    a. Allow students to view their fee status.
    b. Enable online hostel fee payments and update the payment status in real time.
3. **Visitor Tracking**:
    a. Record details of visitors, including their name, contact information, and purpose of visit.
    b. Associate visitor logs with student records for easy tracking.
4. **Complaint Management**:
    a. Provide students with an interface to lodge complaints.
    b. Allow administrators to view, update, and resolve complaints.
5. **Maintenance Requests**:
    a. Accept maintenance requests from students and staff.
    b. Update the status of maintenance requests (e.g., pending, in progress, resolved).
6. **Room Occupancy Report**:
    a. Generate reports on room occupancy, including details of empty and occupied rooms.
    b. Provide summary statistics for administrators.
7. **Warden and Staff Management**:
    a. Maintain records of wardens and other hostel staff, including their personal details, roles, and responsibilities.

8. **Leave Management**:

- The **Student Leave Register** functionality can include:
    - Submission of leave requests by students, indicating dates and reasons.
    - Approval or rejection of requests by the administrator.
    - Leave history tracking for each student.
- The **Staff Leave Register** functionality should similarly allow:
    - Staff to request leave and specify the reason.
    - Wardens or higher authorities to manage these requests.

9. **Authentication and User Roles**:
    a. Provide a secure login system with different roles (e.g., student, administrator).
    b. Ensure role-based access to system functionalities.
10. **Database Operations**:

a. Perform CRUD (Create, Read, Update, Delete) operations on all entities (e.g., rooms, students, visitors, complaints).
b. Ensure data consistency and integrity in the MySQL database.

3.3 Use Case Model

3.3.1 Use Case #1: Room Allocation (U1)

Author: K ABHIRAM

Purpose:
The **Room Allocation** use case is designed to assign hostel rooms to students based on availability, preferences, and eligibility criteria. It ensures proper record-keeping and prevents conflicts in room assignments.

Requirements Traceability:

- RQ1: The system must allow wardens/admins to allocate rooms.
- RQ2: Students should be able to request room changes.
- RQ3: The system should check room availability before allocation.
- RQ4: Only eligible students (based on hostel rules) should be assigned rooms.

Priority: High

Preconditions:

- The student must be registered in the system.
- Available rooms must be listed in the database.
- The student must have cleared the necessary hostel fees.

Postconditions:

- The student is assigned a room.
- The system updates the room's status in the database.
- The student receives a confirmation notification.

Actors:

- **Primary Actor:** Warden/Admin
- **Secondary Actor:** Student (Requester)

Extends:

- This use case extends **Student Profile Management (U2)** and **Hostel Fee Payment (U3)**, as room allocation is only possible after verification.

Flow of Events:

Basic Flow (Normal Flow):

1. Warden/Admin logs into the system.
2. Selects "Room Allocation" from the menu.
3. Searches for the student using Student ID.
4. Selects an available room from the list.
5. Confirms room assignment.
6. System updates the room's status to "Occupied."
7. Student receives an email/SMS confirmation.

Alternative Flow:

- If no rooms are available, the system suggests waitlisting the student.
- If the student wants a specific room, the system checks its availability and either allocates it or suggests alternatives.

Exceptions:

- **E1:** The student has not paid the hostel fee → System prevents allocation and displays a message.
- **E2:** The room is already occupied → System prompts the admin to choose another room.
- **E3:** The student ID is invalid → System displays an error and asks for correct input.

Includes:

- **U4: Notification System** (to send room confirmation messages).
- **U5: Student Profile Management** (to verify student details).

Notes/Issues:

- Consider implementing an **automated room allocation algorithm** based on hostel rules.
- Ensure that room assignments comply with gender-specific hostel regulations.

Use Case #2: Hostel Fee Payment (U3):

Author: AMIT KUMAR

Purpose:
To enable students to pay hostel fees online, ensuring that only fee-cleared students are eligible for room allocation and hostel services.

Requirements Traceability:

- RQ5: The system must support multiple payment methods.
- RQ6: The payment system must generate invoices and receipts.
- RQ7: Students cannot be allocated rooms until payment is completed.

Priority: High

Preconditions:

- The student must be registered in the hostel management system.
- Payment gateway must be active and configured.

Postconditions:

- The system marks the student as "Fee Paid."
- The student receives a payment confirmation receipt.
- The system updates the financial records.

Actors:

- **Primary Actor:** Student
- **Secondary Actor:** Admin (for verification)
- **External Actor:** Payment Gateway

Extends:

- Extends **U1: Room Allocation** (as room allocation is dependent on payment).

Flow of Events:

1. Basic Flow:

1. Student logs into the system.
2. Selects "Hostel Fee Payment" option.
3. Chooses a payment method (Credit Card, UPI, Net Banking).
4. Enters payment details and confirms payment.
5. System verifies payment with the payment gateway.
6. If successful, the system updates the fee status and sends a confirmation message.
7. Student can now proceed with room allocation.

2. Alternative Flow:

- If payment fails, the system allows the student to retry with a different payment method.
- If the payment gateway is down, the system suggests manual payment at the hostel

office.

3. Exceptions:

- **E1:** Payment fails due to insufficient funds → System notifies the student and asks for an alternative payment method.
- **E2:** Payment gateway timeout → System retries or suggests manual payment.
- **E3:** Student ID not found → System prompts re-entry of valid credentials.

Includes:

- **U6: Payment Gateway Integration** (to handle online transactions).
- **U4: Notification System** (to send payment confirmation).

Notes/Issues:

- Consider enabling **installment-based payments** for students.
- Ensure **secure payment transactions** by implementing **encryption**.

3.1.3 Use Case #3: Student Leave Register (U5)

Author: K. KOTI REDDY
Purpose:

- To allow students to apply for leave and obtain approval from the warden.
- To keep track of students leaving and returning to the hostel for security purposes.

Requirements Traceability:

- System should record leave applications, approvals, and rejections.
- The warden must verify and approve/reject leave requests.
- Leave records should be accessible for security checks.

Priority: High

Preconditions:

- Student must be a hostel resident.
- Leave request must be submitted within the allowed timeframe.

Postconditions:

- Approved leave is logged in the system.
- If rejected, the student receives a notification.

Actors:

- **Student** (Applies for leave)
- **Warden** (Approves or rejects leave)
- **Security Guard** (Verifies leave status when the student exits/returns)

Flow of Events:

1. **Basic Flow:**
    a. Student submits a leave request with dates and reason.
    b. Warden reviews and approves/rejects the request.
    c. If approved, the system logs the leave, and the security guard is notified.
    d. Student checks out at the hostel gate.
    e. Upon return, student checks in, and the system updates the status.
2. **Alternative Flow:**
    a. If the student needs an urgent leave, an instant approval request is sent to the warden.
    b. If the warden does not respond within a set time, leave is escalated to a senior staff member.
3. **Exceptions:**
    a. If the student has unpaid fees or disciplinary issues, leave may be denied.
    b. If the student does not return on time, an alert is sent to the warden and security.

Use Case #4: Complaint Management (U7)

Author: D. SRIDHAR

Purpose:
To allow students to report issues such as maintenance requests, room complaints, or service-related grievances.

Requirements Traceability:

- RQ8: The system must allow students to submit complaints.
- RQ9: Admins/wardens must be able to view, track, and resolve complaints.
- RQ10: Students should receive updates on complaint resolution.

Priority: Medium

Preconditions:

- The student must be logged into the system.

Postconditions:

- The complaint is logged into the system.
- The responsible admin/staff is notified.
- The student receives updates on complaint progress.

Actors:

- **Primary Actor:** Student
- **Secondary Actor:** Admin, Maintenance Staff

Flow of Events:

1. Basic Flow:

1. Student logs into the system.
2. Selects “Submit Complaint” option.
3. Fills out complaint details and submits it.
4. The system assigns a complaint ID and notifies the admin.
5. Admin/warden reviews and assigns it to maintenance staff.
6. The complaint is resolved, and the system updates the student.

2. Alternative Flow:

- If the complaint is **urgent**, it is marked as "High Priority" for immediate action.
- If the complaint is **misdirected**, the admin can reassign it to another department.

3. Exceptions:

- **E1:** Student submits an **incomplete complaint form** → System prompts for missing details.
- **E2:** Maintenance staff is unavailable → System escalates the complaint.

Includes:

- **U8: Notification System** (to inform students and staff about complaint updates).

Notes/Issues:

- Ensure a **feedback system** for students to rate complaint resolutions.

4 Other Non-functional Requirements

4.1 Performance Requirements:

Performance Requirements for the Hostel Management System (HMS)

The performance requirements for the **Hostel Management System (HMS)** ensure that the system operates efficiently, providing a seamless user experience even during high usage or peak demand periods. These requirements are based on the functionality of the system and consider both the load on the system and the response time needed for specific tasks. Below are the key performance requirements for the HMS:

1. Room Allocation

- **P1**: The system should allocate a room within **3 seconds** for students when requested during off-peak times.
- **P2**: In peak times (e.g., during the beginning of a semester), the system should allocate rooms within **5 seconds** to prevent delays and maintain a smooth user experience.
- **Rationale**: Quick allocation of rooms is essential for students to confirm their bookings efficiently. The system should handle requests promptly, especially during high-demand periods.

2. Fee Payment

- **P3**: Fee payment transactions must be processed within **2 seconds** of receiving the payment request from the user.
- **P4**: If the payment gateway experiences delays, the system should provide feedback to the user within **10 seconds** and offer options to retry or contact support.
- **Rationale**: Users expect a fast, seamless fee payment process. Delays could frustrate users and potentially lead to failed transactions, which can affect user trust and hostel operations.

3. Visitor Tracking

- **P5**: Visitor check-in and check-out records should be processed and updated in the system within **5 seconds**.
- **Rationale**: Real-time visitor tracking is essential for hostel security and smooth operation. Delays in updating visitor records could lead to security issues or confusion regarding visitor permissions.

4. Complaint Management

- **P6**: Complaints submitted by students or staff should be acknowledged and entered into the system within **10 seconds** of submission.
- **P7**: Admin responses to complaints should be visible to the user within **20 seconds** to ensure transparency and effective issue resolution.
- **Rationale**: Fast processing and resolution of complaints are critical for maintaining a positive environment. Users should be kept informed promptly about the status of their complaints.

5. Maintenance Requests

- **P8**: Maintenance requests should be logged and categorized within **5 seconds** of submission.
- **P9**: For urgent maintenance issues, the system should trigger an alert to the relevant staff within **10 seconds**.
- **Rationale**: Maintenance issues require prompt attention to ensure the hostel's functionality. Immediate logging and alerting can reduce response time to critical issues.

6. Room Occupancy Report

- **P10**: Room occupancy reports should be generated and displayed in **less than 2 seconds** upon request from an admin or staff user.
- **P11**: The system should handle up to **10,000 concurrent room occupancy queries** without degradation in performance.
- **Rationale**: The system must generate room occupancy data quickly to aid in decision-making. Performance during high query loads must be optimal to avoid delays in management operations.

7. Wardens and Staff Management

- **P12**: Changes to staff schedules or assignments should be reflected in the system within **3 seconds**.
- **Rationale**: Ensuring that staff assignments and changes are updated quickly is crucial for operational efficiency, especially when managing shifts and responsibilities.

8. Leave Management (Staff and Student)

- **P13**: Leave requests should be processed and stored in the system within **5 seconds** of submission.
- **P14**: Leave balances should be recalculated and displayed within **5 seconds** of an admin approving or rejecting a leave request.
- **Rationale**: Prompt leave request processing ensures smooth tracking of absences and avoids confusion or scheduling conflicts.

9. Multi-User Load and Scalability

- **P15**: The system should support at least **500 concurrent users** (students, staff, and administrators) without degradation in performance.
- **P16**: During peak load times, such as when students are logging in at the start of a semester, the system should maintain performance with no more than a **10% increase in response time**.
- **Rationale**: The system must be capable of handling high user loads, particularly during peak times, such as the beginning of terms when room allocations, fee payments, and leave requests are more frequent.

10. System Availability and Uptime

- **P17**: The system must maintain an **uptime of 99.9%**, ensuring it is available for use at all times except during scheduled maintenance windows.
- **Rationale**: Hostel management is a 24/7 operation, and system downtime can significantly impact hostel operations. High availability ensures smooth, continuous functioning.

11. Backup and Recovery

- **P18**: The system should perform a full backup of all data at least once every **24 hours**, and restore data to its latest state within **30 minutes** in case of a failure.
- **Rationale**: Regular backups and quick recovery are essential to prevent data loss and minimize downtime in the event of system failure.

4.2 Safety and Security Requirements

The Hostel Management System (HMS) must incorporate safety and security measures to prevent unauthorized access, data breaches, and potential misuse of sensitive information. The following requirements must be met:

Data Security & Privacy:

- **User Authentication:**
  - The system must implement **role-based access control (RBAC)** to restrict access based on user roles (e.g., **students, wardens, administrators**).
  - Multi-factor authentication (MFA) should be implemented for **admin-level access**.
  - All passwords must be **hashed and stored securely** using **bcrypt**.
- **Data Protection & Encryption:**

    - Personal details such as student ID, contact information, and fee payment records must be **encrypted using AES-256**.
    - Sensitive data must be transmitted securely using **HTTPS with SSL/TLS encryption**.
    - Backup databases must be **regularly updated and stored securely** to prevent data loss.

System Safety & Threat Prevention:

- **SQL Injection & XSS Prevention:**
    - The system must use prepared statements and parameterized queries to prevent SQL injection attacks.
    - Input fields must be validated and sanitized to prevent cross-site scripting (XSS) attacks.
- **Session Management & Access Logs:**
    - User sessions must **automatically expire after 10 minutes of inactivity** to prevent unauthorized access.
    - A detailed logging system must record login attempts, failed logins, and data modifications for security audits.
- **Mobile Connection Security:**
    - The mobile access interface must be secured with **OAuth 2.0 authentication** to prevent **unauthorized API access**.
    - Any **mobile-based interactions** (e.g., **fee payments or room applications**) must **require OTP verification**.

Compliance with Safety Regulations

- The system must comply with **General Data Protection Regulation (GDPR)** to ensure the protection of **personal user data**.
- **System administrators** must ensure **regular security updates** and conduct **penetration testing** to identify vulnerabilities.
- The **software development team** must follow **OWASP security guidelines** to prevent common cyber threats.

4.3 Software Quality Attributes

The **Hostel Management System (HMS)** must meet several quality attributes to ensure efficient, reliable, and scalable operation:

1. **Reliability**: The system should maintain **99.9% uptime** and handle error rates of less than **1%** during peak usage, with **10-second recovery** from common failures.

2. **Usability**: The system must be **easy to use**, with first-time users completing tasks in under **5 minutes** and a consistent, accessible interface that adheres to **WCAG 2.1 Level AA** standards.

3. **Maintainability**: The system should be **modular**, with at least **80% of code** in reusable modules and **90% code documentation**. Automated testing should ensure **80% test coverage** for smooth updates.

4. **Scalability**: It must support **concurrent users** and handle **data** without performance degradation, ensuring smooth operation as the system grows.

5. **Interoperability**: The system should integrate with **3 third-party services** and support **JSON** and **XML** for data exchange to work seamlessly with other systems.

6. **Adaptability**: The system should be designed for easy **module integration**, allowing changes like adding new features (e.g., for additional facilities) and enabling **easy configuration management** through the admin dashboard.