Software Requirements Specification
(Hostel Management System)

1.1 1. Introduction

1.1.1 1.1 Purpose

The purpose of this document is to provide a detailed description of the requirements for the Hostel Management System (HMS). This system will automate hostel operations such as student registration, room allocation, fee management, complaint handling, and attendance tracking. The HMS will simplify administrative tasks, improve efficiency, and ensure a seamless experience for administrators and students.

1.1.2 1.2 Scope

The **Hostel Management System (HMS)** is a web-based platform designed to manage hostel activities efficiently. The application will include functionalities for:

- Student registration, including maintaining student profiles and hostel details.
- Room allocation, enabling dynamic room assignments based on availability and preferences.
- Fee management for tracking hostel fees, payments, and generating reports.
- Complaint management to allow students to submit and track complaints online.
- Attendance tracking to maintain a record of students' in-time and out-time details.

The system will utilize HTML, CSS, and JavaScript for the front-end, Python with Flask for the back-end, and MySQL as the database.

1.1.3 1.3 Definitions, Acronyms, and Abbreviations

- **CRUD**: Create, Read, Update, Delete operations for data management.
- **MySQL**: Relational database management system for secure and efficient data storage.
- **Flask**: Python-based web framework for developing the back-end.

1.1.4 1.4 References

- Flask documentation: Flask
- MySQL documentation: MySQL

1.2 2. Overall Description

1.2.1 2.1 Product Perspective

The he **HMS** is a standalone, web-based system accessible via web browsers. The front-end will ensure a responsive and user-friendly experience, while the back-end will manage interactions with the MySQL database for secure data handling.

1.2.1 2.2 Product Functions

The core functionalities of the HMS include:

- **Student Registration**: Register and manage student details (e.g., name, hostel ID, room number).
- **Room Allocation**: Assign and manage hostel room availability dynamically.
- **Fee Management**: Record payments, generate invoices, and track payment statuses.
- **Complaint Management**: Enable students to log complaints and allow administrators to resolve them.
- **Attendance Tracking**: Record and manage daily student attendance details.

1.2.2 2.3 User Classes and Characteristics

- **Admin**: Responsible for managing all hostel operations, including student registrations, room allocations, fee updates, and resolving complaints.
- **Student**: Can register, check room allocation, pay hostel fees, log complaints, and view attendance records.

1.2.3 2.4 Operating Environment

- **Server**: The application will run on a server with Python and Flask installed.
- **Database**: MySQL will be used for data storage.

1.2.4 2.5 Design and Implementation Constraints

- The application must be developed using Flask and MySQL.
- The application should follow the MVC (Model-View-Controller) architecture.

1.2.5 2.6 Assumptions and Dependencies

- The user has a web browser to access the application.
- The server environment supports Python and Flask.

1.3 3. Specific Requirements

1.3.1 3.1 Functional Requirements

3.1.1 Student Registration

Description: Allow admins to register students with details like name, hostel ID, email, room number, and fee status.

Inputs: Name, email, room preference, and contact details.

Outputs: Confirmation message and a registered student list.

3.1.2 Room Allocation

Description: Allocate rooms to students based on availability and preferences.

Inputs: Student ID, room preferences.

Outputs: Assigned room number.

3.1.3 Fee Management

Description: Track payments, generate invoices, and update payment statuses.

Inputs: Student ID, amount paid.

Outputs: Receipt and payment confirmation.

3.1.4 Complaint Management

Description: Allow students to submit complaints and track their status.

Inputs: Student ID, complaint details.

Outputs: Complaint ID and resolution status.

3.1.5 Attendance Tracking

Description: Maintain attendance records for students.

Inputs: Student ID, date, in-time, out-time.

Outputs: Daily attendance log.

1.3.2 3.2 Non-Functional Requirements

3.2.1 Performance Requirements

- The system should respond to user actions within **3 seconds**.

3.2.2 Security Requirements

- Use hashed passwords for secure authentication.
- Prevent SQL injection by validating inputs.

3.2.3 Usability Requirements

- The system should have a user-friendly interface for both admins and students.

3.3 Database Requirements

Tables:

- **students**: Store student details (ID, name, contact, room number, fee status).
- **rooms**: Store room details (room number, capacity, status).
- **fees**: Store fee payment details (student ID, amount, due date, payment status).
- **complaints**: Store complaints (complaint ID, student ID, description, status).
- **attendance**: Store attendance records (student ID, date, in-time, out-time).

3.3 4. External Interface Requirements

3.3.1 4.1 User Interfaces

- A login page for admins and students.
- A dashboard for admins to manage all operations.
- A student portal for viewing personal details, complaints, and attendance records.

4.2 Hardware Interfaces

- Standard server hardware for hosting the application.

4.3 Software Interfaces

- Interaction with the MySQL database for all data operations.

4.4 Communications Interfaces:

- The system will use HTTP/HTTPS for secure communication between the client and server.