# Software Requirements Specification

# For

# Hostel Management System



**Centurion University Of Technology And Management**

**School of Engineering and Technology**

**Bhubaneswar**

**Submitted By:**

Name:

Regd No:

Branch:

Date:

**Submitted to:**

**Table of Contents**

**CHAPTER 1: Introduction1.1 Purpose1.2 Scope1.3 ObjectivesCHAPTER 2: Overall Description2.1 Product Perspective2.2 User Roles2.3 System ArchitectureFigure 1: System Architecture DiagramCHAPTER 3: System Requirements3.1 Hardware Requirements3.2 Software RequirementsCHAPTER 4: System Features (Functional Requirements)4.1 Administrator Functions4.2 Student FunctionsCHAPTER 5: System Model5.1 Use Case DiagramFigure 2: Detailed Use Case Diagram5.2 Student Application Sequence DiagramFigure 3: Student Application Sequence Diagram5.3 Room Allocation Activity DiagramFigure 4: Room Allocation Activity DiagramCHAPTER 6: Data Requirements6.1 Entity-Relationship (ER) Diagram6.2 Data DictionaryCHAPTER 7: Non-Functional Requirements7.1 Performance7.2 Security7.3 Usability7.4 ReliabilityREFERNCE**

**CHAPTER-1**

## 1. Introduction

This document outlines the requirements for the Hostel Management System (HMS), a web-based application designed to simplify and automate hostel administration.

### 1.1 Purpose

The main purpose of this project is to replace the current manual, paper-based system for managing hostel activities. The existing process is slow, prone to errors, and requires significant manual effort. This software will provide a centralized and efficient platform for both students and administrators, making tasks like room allocation, student registration, and record-keeping much easier.

### 1.2 Scope

The project focuses on developing a web application for managing hostel operations within a university or school. The key features include:

* **Admin Module:** For hostel staff to manage student applications, allocate rooms, update student records, and post notices.
* **Student Module:** For students to apply for rooms, check their application status, and view notices.
* **Core Functions:** The system will handle student registration, room allocation and vacating, visitor logging, and basic furniture tracking. [1, 1]

### 1.3 Objectives

* To create a user-friendly web application for hostel management.
* To reduce manual paperwork and the possibility of human error.

**CHAPTER-2**

## 2. Overall Description

### 2.1 Product Perspective

The Hostel Management System is a self-contained web application. It is designed to be used by two main types of users: Hostel Administrators and Students. The system will provide separate interfaces and functionalities tailored to the roles of each user group.

### 2.2 User Roles

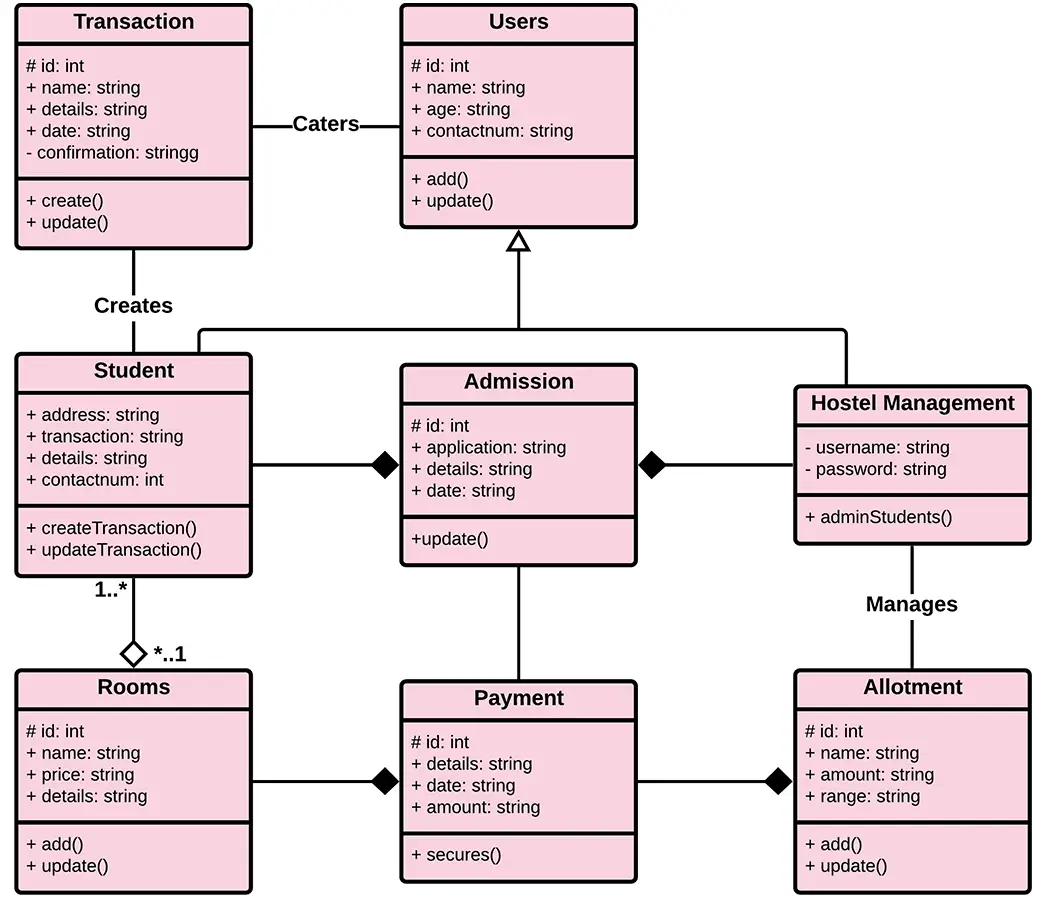
* **Administrator:** A staff member responsible for managing the hostel. The admin will have full access to all features, including managing student data, allocating rooms, and handling system settings. 1
* **Student:** An enrolled student of the institution who is applying for or residing in the hostel. Students can use the system to submit applications, view their status, and receive updates. 1

### 2.3 System Architecture

The system will be built using a modern three-tier architecture, which separates the application into the following layers:

1. **Presentation Layer (Frontend):** Developed using **Angular**. This is the user interface that runs in the client's web browser. It will be responsible for displaying information and capturing user input.
2. **Application Layer (Backend):** Developed using **Java** and the **Spring Boot** framework. This layer will contain the core business logic, handle user requests, and expose data through REST APIs.
3. **Data Layer (Database):** A **MySQL** database will be used to store and manage all the data for the application, such as student records, room details, and application information. 5

**The diagram below illustrates this architecture:**



***Figure 1: System Architecture Diagram***

**CHAPTER-3**

## 3. System Requirements

### 3.1 Hardware Requirements

* **Processor:** Intel Pentium 4 or higher
* **RAM:** 512 MB minimum
* **Hard Disk:** 10 GB of free space

### 3.2 Software Requirements

* **Client-Side:**
  + **Operating System:** Windows, macOS, or Linux
  + **Web Browser:** Google Chrome, Mozilla Firefox, or Microsoft Edge (latest versions)
* **Server-Side:**
  + **Backend Framework:** Spring Boot
  + **Programming Language:** Java
  + **Database:** MySQL 5
  + **Web Server:** Apache Tomcat (embedded with Spring Boot)

**CHAPTER-4**

## 4. System Features (Functional Requirements)

This section describes the key functions the system will perform for each user role.

### 4.1 Administrator Functions

The Administrator will be able to:

* **Manage Student Applications:** Review applications submitted by students, verify their details, and approve or reject them.
* **Allocate Rooms:** Assign rooms to students with approved applications based on availability.
* **Vacate Rooms:** Update the system when a student leaves the hostel, making the room available for others.
* **Manage Student Records:** View, create, update, and delete student profiles in the system.
* **Post Notices:** Create and publish announcements for all students to see on their dashboard.
* **Log Visitors:** Record details of visitors for security purposes.
* **Track Furniture:** Maintain a basic inventory of furniture in each room.

### 4.2 Student Functions

The Student will be able to:

* **Register and Login:** Create an account and log in to the system securely.
* **Submit Application:** Fill out and submit the hostel application form online.
* **View Application Status:** Check the current status of their application (Pending, Approved, or Rejected).
* **View Notices:** See important announcements and updates from the administrator on their dashboard. uu
* **Submit Vacate Request:** Formally request to vacate their room through the system.

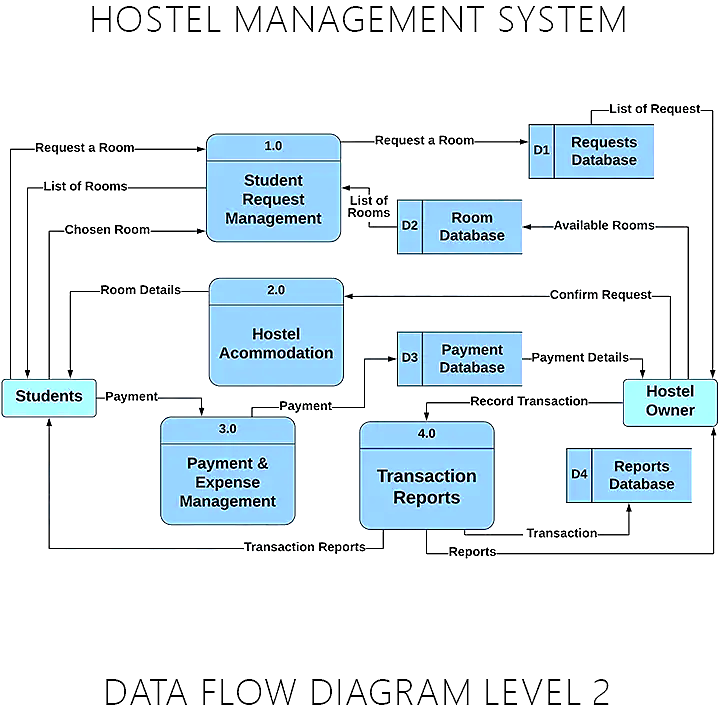
**CHAPTER-5**

## 5. System Model

This section provides detailed diagrams to visualize the system's processes and user interactions.

### 5.1 Use Case Diagram

The Use Case diagram below illustrates the various actions taken by the Student and Administrator actors. The actors are presented as stick figures, while the use cases are presented in ovals.



***Figure 2: Detailed Use Case Diagram***

**A brief description is created for each use case:**

* **Apply for Hostel:** This use case is started by the student. It allows them to fill out and submit their application for hostel accommodation.
* **View Application Status:** This allows the student to track the real-time status of their submitted application (e.g., Pending, Approved, Rejected).
* **Manage Applications:** This use case is started by the administrator. It provides the functionality to review, verify, approve, or reject student applications.
* **Allocate & Vacate Rooms:** This allows the administrator to assign rooms to approved students and process room vacation requests from departing students.

### 5.2 Student Application Sequence Diagram

The Sequence Diagram below shows the sequential steps involved when a student applies for the hostel and an administrator approves the application. It details the interactions between the user, the web application, the backend server, and the database.

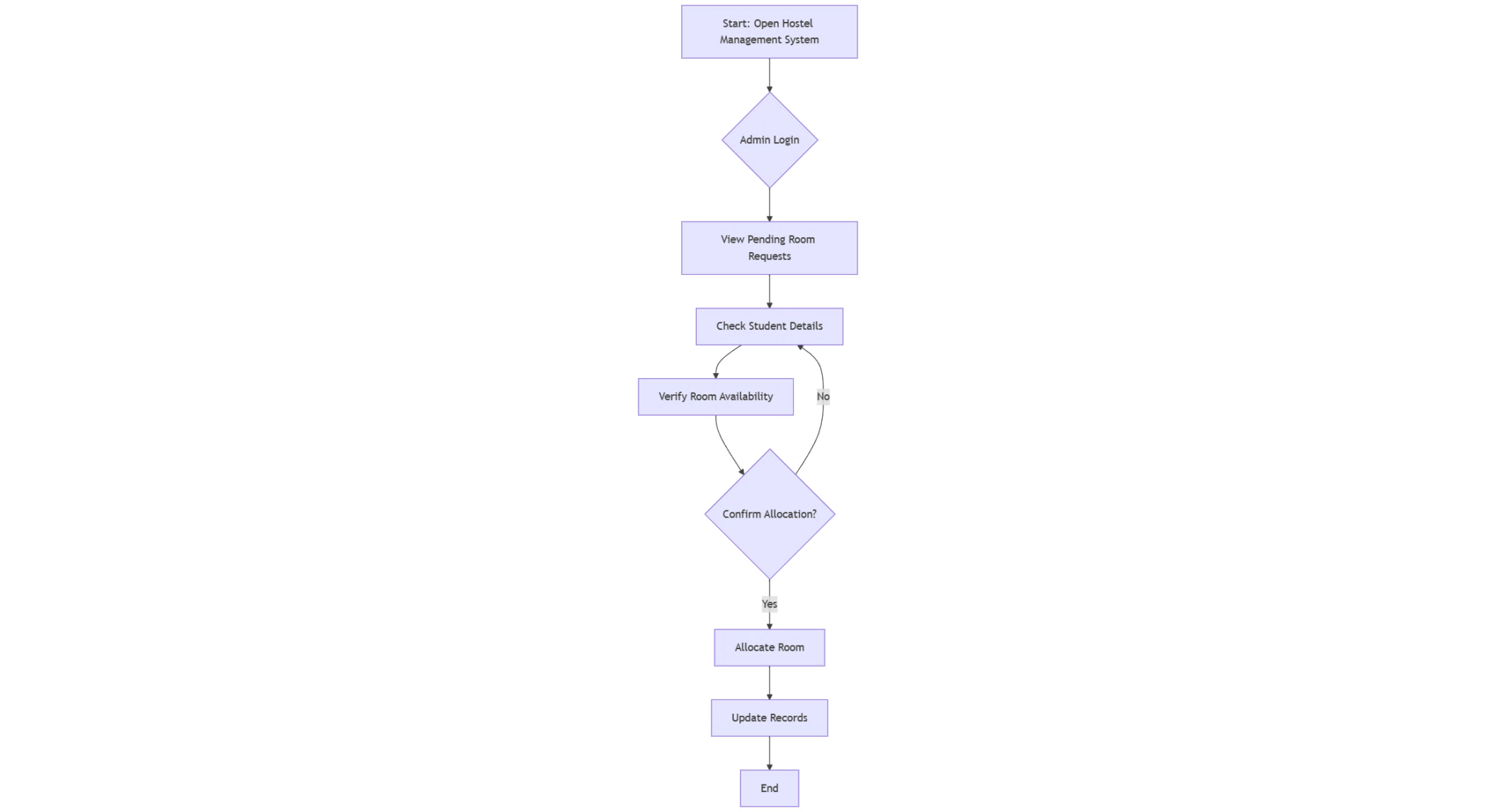
***Figure 3: Student Application Sequence Diagram***

**The steps involved in this sequence are detailed below:**

1. The student fills out the application form on the Web App and submits it.
2. The Web App sends the application data to the Backend server.
3. The Backend validates the data and saves the new application record in the Database.
4. The Backend confirms to the Web App that the application was created successfully.
5. The Web App displays a success message to the student.
6. Later, the Administrator logs in and requests a list of pending applications.
7. The Backend retrieves the list from the Database and sends it to the Web App.
8. The Web App displays the list of pending applications to the Administrator.
9. The Administrator approves an application.
10. The Backend updates the application's status to 'Approved' in the Database.
11. The Web App displays a confirmation message to the Administrator.

### 5.3 Room Allocation Activity Diagram

The Activity Diagram below shows the flow of events for the room allocation process, which is performed by the Administrator. This diagram models the workflow from logging in to successfully allocating a room.



***Figure 4: Room Allocation Activity Diagram***

This activity diagram shows the logical flow of the room allocation task. The administrator logs in, selects an approved student and an available room, and confirms the choice. The system then updates the relevant records in the database to finalize the allocation.

s

**CHAPTER-6**

## 6. Data Requirements

The system's data will be stored in a relational database (MySQL). The core entities and their relationships are shown in the ER diagram below.

### 6.1 Entity-Relationship (ER) Diagram



### 6.2 Data Dictionary

The following tables define the key data entities and their attributes.

**Entity: Hostel**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| Hostel\_ID | INT (PK) | Unique ID for the hostel |
| Hostel\_name | VARCHAR(100) | Name of the hostel |
| No\_of\_rooms | INT | Total number of rooms |

**Entity: Administrator**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| Admin\_ID | INT (PK) | Unique ID for the administrator |
| Fname | VARCHAR(50) | First name |
| Lname | VARCHAR(50) | Last name |
| Mob\_No | VARCHAR(15) | Mobile number |
| Hostel\_ID | INT (FK) | ID of the hostel they manage |

**Entity: Student**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| Student\_ID | INT (PK) | Unique ID for the student |
| Fname | VARCHAR(50) | First name |
| Lname | VARCHAR(50) | Last name |
| Mob\_No | VARCHAR(15) | Mobile number |
| Dept | VARCHAR(50) | Academic department |
| Year\_of\_study | INT | Current year of study |
| Room\_ID | INT (FK) | ID of the assigned room |

**Entity: Room**

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Description |
| Room\_ID | INT (PK) | Unique ID for the room |
| Hostel\_ID | INT (FK) | ID of the hostel the room is in |
| Status | VARCHAR(20) | e.g., 'Available', 'Occupied' |

**CHAPTER-7**

## 7. Non-Functional Requirements

### 7.1 Performance

* The system should be able to handle at least 50 users at the same time without a noticeable drop in speed.
* Web pages and search results should load in under 3 seconds.
* The database should efficiently manage up to 1000 student records.

### 7.2 Security

* All users must log in with a valid username and password.
* The system will have two access levels: Administrator and Student. Students will not be able to access administrative functions.
* Passwords will be stored in an encrypted format in the database.

### 7.3 Usability

* The user interface should be simple, intuitive, and easy to navigate for users with basic computer skills.
* The system should be responsive and work well on standard desktop web browsers.
* Error messages should be clear and help the user understand what went wrong.

### 7.4 Reliability

* The system should be available for use 24/7 with minimal downtime.
* Regular backups of the database must be performed to prevent data loss in case of a system failure.

**REFERNCE**

1. HOSTEL MANAGEMENT SYSTEM | Judiciary, accessed September 25, 2025, <https://judiciary.go.ug/files/Hostel%20system%20management(1).pdf>
2. (PDF) Hostel Management System - ResearchGate, accessed September 25, 2025, <https://www.researchgate.net/publication/391482640_Hostel_Management_System>
3. BRMehta/hostel-management-frontend - GitHub, accessed September 25, 2025, <https://github.com/BRMehta/hostel-management-frontend>
4. Hostel Management System (HMS) - IJRASET, accessed September 25, 2025, <https://www.ijraset.com/research-paper/hostel-management-system-hms>
5. Understanding Use Case Diagram for Hostel Room Allocation Management System, accessed September 25, 2025,

<https://www.mymap.ai/blog/hostel-room-allocation-management-use-case>