# Software Requirements Specification

for

# Hostel Management

Version 1.0

Prepared by

Group Name: Zen

| | | |
|---|---|---|
| Yelavarthi Lalitya | SE22UARI188 | se22uari188@mahindrauniversity.edu.in |
| Tikkisetti Sri Dhruti | SE22UARI175 | se22uari175@mahindrauniversity.edu.in |
| Monisha Kollipara | SE22UARI098 | se22uari098@mahindrauniversity.edu.in |
| Mullangi Guna Shritha | SE22UARI102 | se22uari102@mahindrauniversity.edu.in |
| Cherith Reddy Yerabolu | SE22UARI036 | se22uari036@mahindrauniversity.edu.in |

| | |
|---|---|
| Instructor: | Dr. Avinash Arun Chauhan |
| Course: | Software Engineering |
| Lab Section: | Wednesday 10:35am – 12:30pm |
| Teaching Assistant: | *Yasaswi Vanarasi, Surya Prakash, Surya Phani Teja* |

**Date:** **10th March 2025**

## Contents

## Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| 1.0 | Yelavarthi Lalitya, Monisha Kollipara, | Added detailed Sequence Diagram for complaint submission, AI chatbot, and food request. | 10/03/2025 |

| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| | Sri Dhruti Tikkisetti, Cherith Reddy Yerablolu, Mullangi Guna Shritha | Updated Use Case Diagrams and included Alternative Flows for database failures and admin rejections. | |

# 1 Introduction

## 1.1 Document Purpose

This document specifies the **Software Requirements Specification (SRS)** for the **Hostel Management System (HMS)**, version **1.0**. It outlines the system's functionality, constraints, and intended users. This document serves as a reference for developers, testers, and stakeholders to ensure clarity in system requirements.

The HMS is designed to **streamline hostel-related activities**, including **room allocation, complaints management, lost and found services, room cleaning schedules, washing machine bookings, and AI-powered roommate matching**. This document focuses on defining the **functional and non-functional requirements** of the system.

## 1.2 Product Scope

The **Hostel Management System (HMS)** is a **mobile** platform aimed at improving hostel administration and resident experience. The system will provide:

- **Smart Roommate Matching**: Uses AI-based questionnaires to match students based on preferences.
- **Complaint Management**: Allows students to submit and track complaints.
- **Lost and Found Service**: Maintains a database of lost and found items.
- **Room Cleaning & Washing Machine Scheduling**: Provides automated booking systems.
- **AI Chatbot**: NLP-powered chatbot answers hostel-related queries.
- **Call for food when sick:** Allows students to call for food from mess when sick.

The HMS enhances efficiency, reduces administrative workload, and improves student satisfaction.

## 1.3 Intended Audience and Document Overview

This document is intended for:

- **Developers**: To understand system architecture and implementation.
- **Project Managers & Professors**: To oversee project progress and evaluate feasibility.
- **Testers**: To verify system functionality.
- **Users (Hostel Residents & Admins)**: To understand system features and usability.

The document includes:

- Section **2**: System overview and design constraints.

- Section **3**: Functional and non-functional requirements.
- Section **4**: System models and use case diagrams.

## 1.4 Definitions, Acronyms and Abbreviations

This document uses several abbreviations and technical terms relevant to the **Hostel Management System (HMS)**. HMS refers to the **Hostel Management System**, which is the software being developed. **AI (Artificial Intelligence)** is used for features like **smart roommate matching** and the **NLP (Natural Language Processing) chatbot** that answers student queries. **API (Application Programming Interface)** enables communication between the backend and frontend components of the system. The **DBMS (Database Management System)** stores and manages hostel-related data, including complaints, schedules, and lost and found items. **FAQ (Frequently Asked Questions)** refers to the chatbot's ability to provide automated responses to common queries from hostel residents.

## 1.5 Document Conventions

- **Font**: Arial, size **11**
- **Headings**: Bold, numbered sections.
- **Code snippets**: Monospace font.
- **Abbreviations**: Defined in Section **1.4**.

## 1.6 References and Acknowledgments

This document follows the **IEEE 830-1998 Software Requirements Specification (SRS) standard** for structuring software requirements. The development of the **Hostel Management System (HMS)** is based on requirements gathered from **hostel administrators and students** to ensure the system meets real-world needs. Additionally, references include industry best practices for **AI-powered chatbots, scheduling systems, and complaint management platforms**.

# 2 Overall Description

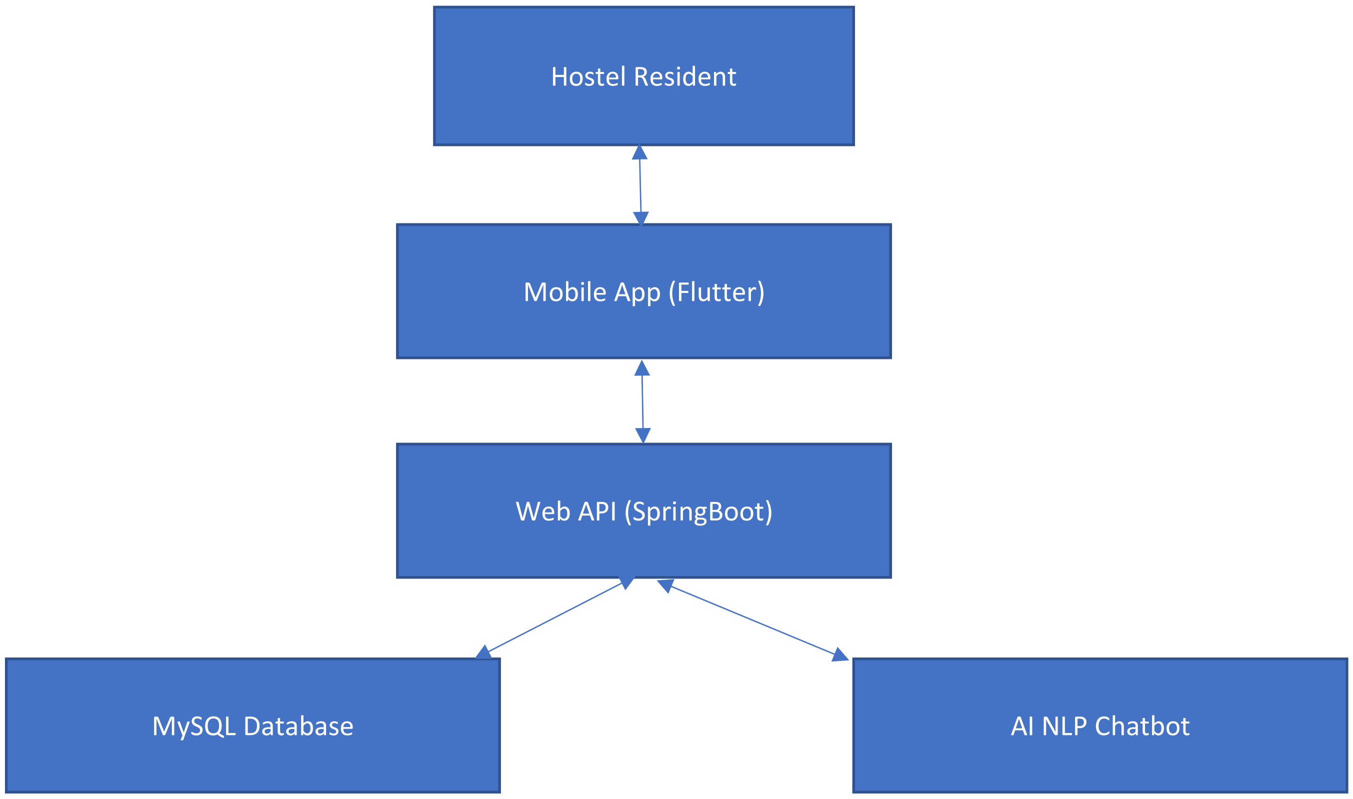
## 2.1 Product Overview

The **Hostel Management System (HMS)** is designed to assist students and hostel administrators in managing hostel-related activities efficiently. It replaces manual processes such as paper-based complaint submissions, lost-and-found tracking, and information requests.

The system consists of a **Flutter-based mobile application** that interacts with a **Spring Boot backend** using **REST APIs**. It includes an **AI-powered chatbot** and an **AI powered Roommate Matching System that provides automated responses to common hostel-related queries. Data is stored in a MySQL database**, ensuring secure and efficient record-keeping.

### 2.1.1 System Interaction Overview

The following diagram illustrates how HMS interacts with its users and components:



Students will use the **Flutter mobile app** to interact with the system, and the **backend** will process requests, store data, and provide responses.

## 2.2 Product Functionality

The **Hostel Management System** provides the following key functionalities:

- **Complaint Submission** – Students can **submit complaints** related to hostel facilities and maintenance.
- **Lost and Found System** – A dedicated section to **report lost items and check for found belongings** in the hostel.
- **Room Cleaning Schedule –** Students can **view** the scheduled cleaning timings for their respective rooms and hostel areas.
- **Washing Machine Booking** – Allows students to **check and book available slots** for washing machines.
- **AI Chatbot (NLP-Based)** – A chatbot that provides **automated responses to hostel-related queries** such as rules, facilities, and procedures.
- **Call for Food When Sick** – Allows students to **request food delivery to their room** when they are unwell.
- AI Based Roommate Matching – Matches students according to the responses given in the questionnaire.

These features aim to **improve communication, reduce administrative workload, and help students access services more conveniently**.

## 2.3 Design and Implementation Constraints

The **Hostel Management System** will be built using a **specific technology stack and design constraints** to ensure performance, security, and usability.

- **Technology Stack**
  - **Backend:** Spring Boot (REST APIs)
  - **Frontend:** Flutter (Mobile App)
  - **Database:** MySQL
  - **AI Chatbot:** NLP-based system using **spaCy** for FAQ handling
- **Performance Constraints**
  - The system must be **scalable**, allowing multiple users to access services **simultaneously**.
  - Chatbot responses should be **generated in less than 2 seconds** for optimal user experience.
- **Security Considerations**
  - Users will have **restricted access** based on their role (e.g., students vs. administrators).
- **Software Design Standards**
  - The system will follow **industry best practices** for software architecture and API development.
  - **Unified Modeling Language (UML)** diagrams will be used for system modeling and documentation.

These constraints ensure that the system remains **efficient, secure, and easy to maintain** while delivering the required functionalities.

## 2.4 Assumptions and Dependencies

Several **assumptions and dependencies** may affect the system’s development and operation.

### 2.4.1 Assumptions

- Students will have **smartphones with internet access** to use the mobile application.
- The hostel administration will **regularly update the database** with student information and facility details.
- The AI chatbot will be **trained on predefined FAQs** but can be expanded with additional data over time.

### 2.4.2 Dependencies

- **Database Availability** – A stable **MySQL database** is required for storing complaints, schedules, and chatbot responses.
- **Third-Party Libraries** – The NLP chatbot depends on **spaCy**, which must be properly configured and maintained.
- **Server Uptime –** The backend must be **hosted on a reliable server** to provide real-time responses to students.

These factors must be considered to ensure **smooth development, deployment, and operation** of the system.

# 3 Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The **Hostel Management System (HMS) mobile application** will serve as the primary interface for students and administrators to interact with the system.

- **Main Interface**: The mobile app will feature a **dashboard** that provides quick access to complaints, lost & found, cleaning schedules, washing machine bookings, and the AI chatbot.
- **Navigation**: A **bottom navigation bar** will allow users to switch between key sections easily.
- **Chatbot Interface**: The chatbot will have a **text input field** where users can type questions and receive automated responses.
- **Accessibility Features**: The interface will include **dark mode, large text support**

### 3.1.2 Hardware Interfaces

The **HMS mobile application** will interact with the following hardware components:

- **Smartphones & Tablets** (Android & iOS) with **Flutter support**
- **Backend Server** running **Spring Boot** to process API requests
- **MySQL Database** for storing user interactions and system data

### 3.1.3 Software Interfaces

The **HMS system** will interact with the following software components:

- **Flutter Mobile App**: The app will send and receive data from the backend using **REST APIs**.
- **Spring Boot Backend**: The backend will handle **business logic, chatbot processing, and database management**.
- **MySQL Database**: The database will store information related to **complaints, lost & found, schedules, and chatbot FAQs**.
- **NLP Chatbot (spaCy-based)**: The chatbot will process **user queries** and return relevant responses using **predefined FAQ datasets**.

## 3.2 Functional Requirements

The **Hostel Management System (HMS)** must perform the following functions to facilitate hostel operations efficiently:

### 3.2.1 Complaint Management

- **F1:** The system shall allow students to **submit complaints** related to hostel facilities.
- **F2:** The system shall store complaints in a **MySQL database** for future reference.

### 3.2.2 Lost and Found System

- **F3:** The system shall allow students to **report lost items** and check for found belongings.

### 3.2.3 Room Cleaning Schedule

- **F4:** The system shall display **room cleaning schedules** for students to view.

### 3.2.4 Washing Machine Booking

- **F5:** The system shall allow students to **book washing machine slots** based on availability.

### 3.2.5 AI Chatbot (NLP-Based)

- **F6:** The system shall provide an **NLP-powered chatbot** to answer hostel-related queries.
- **F7:** The chatbot shall retrieve responses based on **predefined hostel FAQs**.

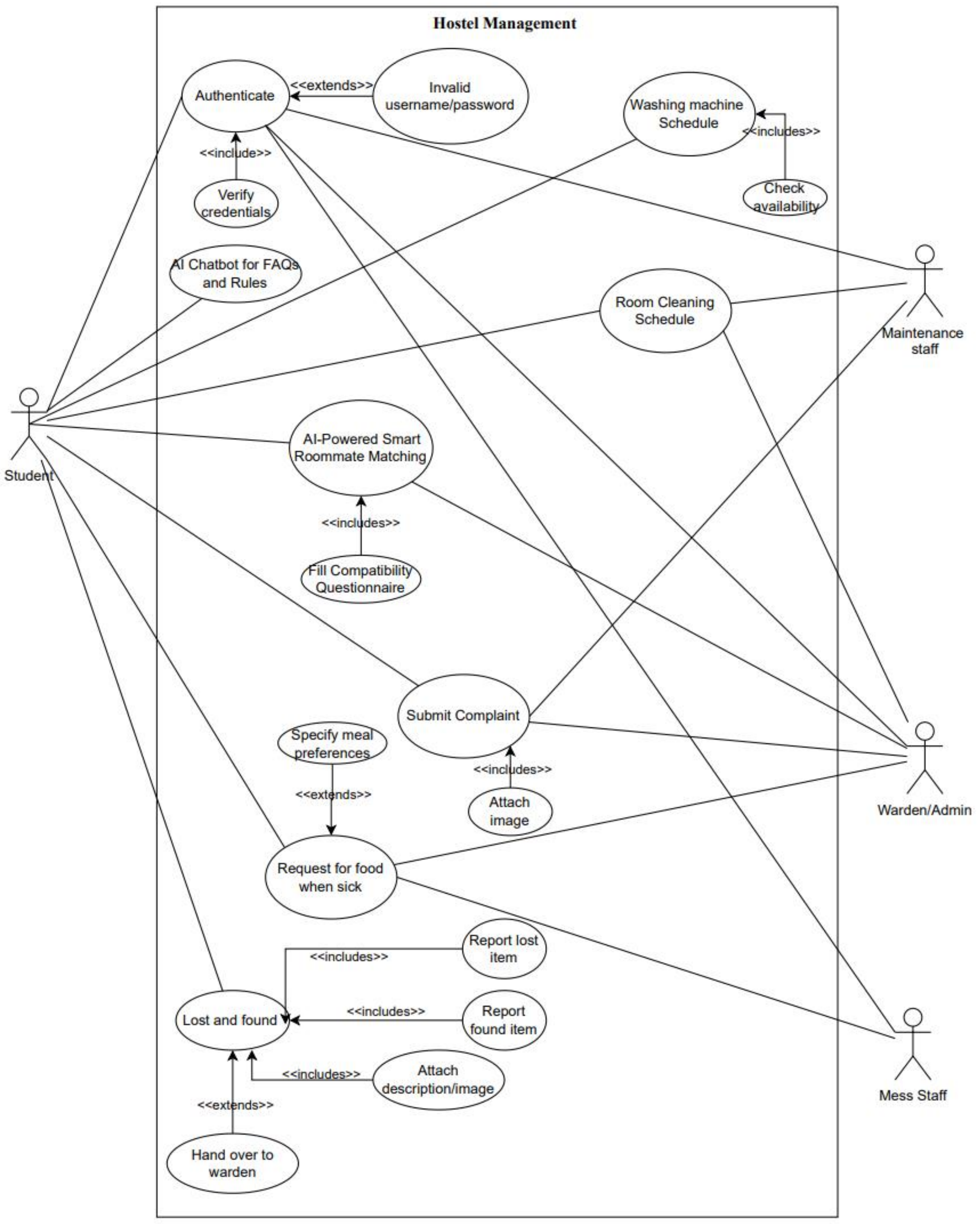
### 3.2.6 Call for Food When Sick

- **F8:** The system shall allow students to **request food delivery** when they are sick.

### 3.2.7 AI-Powered Smart Roommate Matching

- **F9:** The system shall use an AI-based algorithm to match students with compatible roommates based on their preferences.

## 3.3 Use Case Model



### 3.3.1 Use Case: AI Chatbot Query (U1)

**Author:** Lalitya

**Purpose:** The objective of this use case is to allow students to interact with an AI-powered chatbot that provides instant responses to hostel-related queries. The chatbot processes user input, retrieves relevant information, and displays appropriate answers based on predefined FAQs.

**Requirements Traceability:** Linked to F6 & F7.

**Priority:** High. This use case is crucial for providing students with quick access to hostel-related information and reducing the workload on hostel administration.

**Preconditions:** The chatbot must be trained with hostel FAQs, and the user must have access to the mobile application.

**Postconditions:** The chatbot retrieves and displays a relevant response based on the user's query.

**Actors:** Student (asks the question), Chatbot System (processes and responds).

**Extends:** None.

**Flow of Events**

**Basic Flow:**

1. The student opens the chatbot interface in the mobile app.
2. The student types a question related to hostel facilities or rules.
3. The chatbot processes the query using NLP techniques.
4. The chatbot searches the FAQ database for a matching answer.
5. The chatbot displays the most relevant response to the student.

**Alternative Flow:** If the chatbot cannot find an exact match, it provides a general response indicating that the information is unavailable.

**Exceptions:** If the server is down, the chatbot displays an error message stating, "Service unavailable. Please try again later."

**Includes:** None.

**Notes/Issues:** Additional training data may be required to improve chatbot accuracy over time.

### 3.3.2 Use Case: Submit Complaint (U2)

**Author:** Lalitya

**Purpose:** The objective of this use case is to allow students to submit complaints regarding hostel facilities. The system processes the complaint and stores it in the database for administrators to review.

**Requirements Traceability:** Linked to F1 & F2.

**Priority:** High. This use case ensures that students can report hostel-related issues efficiently.

**Preconditions:** The student must be logged into the mobile application.

**Postconditions:** The complaint is stored in the database and marked as "New."

**Actors:** Student (submits the complaint), System (stores and processes the complaint).

**Extends:** None.

**Flow of Events**

**Basic Flow:**

1. The student selects "Submit Complaint" from the app.
2. The system presents a complaint submission form.
3. The student enters complaint details and submits the form.
4. The system stores the complaint in the database and marks it as "New."
5. The system confirms the successful submission.

**Alternative Flow:** If the student submits incomplete details, the system prompts the user to fill in the missing fields before submission.

**Exceptions:** If the system is down, the complaint submission fails, and an error message is displayed.

**Includes:** None.

**Notes/Issues:** The system does not provide complaint tracking or updating functionalities.

### 3.3.3 Use Case: Call for Food When Sick (U3)

**Author:** Lalitya

**Purpose:** The objective of this use case is to allow students to request food delivery to their room when they are unwell. The system processes the request and updates its status in the database.

**Requirements Traceability:** Linked to F8.

**Priority:** Medium. This feature is important for student well-being but does not impact core hostel functionalities.

**Preconditions:** The student must be logged into the mobile application, and food services must be available at the hostel.

**Postconditions:** The food request is stored in the database and marked as "Pending."

**Actors:** Student (requests food), System (processes and stores the request).

**Extends:** None.

**Flow of Events**

**Basic Flow:**

1. The student selects "Request Food When Sick" from the app.
2. The system prompts the student to enter their room number and illness details.
3. The student submits the request.
4. The system stores the request and marks it as "Pending."
5. The system confirms that the request has been received.

**Alternative Flow:** If the student submits multiple requests in a short time, the system prevents duplicate submissions.

**Exceptions:** If food services are unavailable, the system displays an error message stating, "Service not available at the moment."

**Includes:** None.

**Notes/Issues:** The system does not track food delivery status beyond marking it as "Pending."

### 3.3.4 Use Case: AI-Powered Smart Roommate Matching (U4)

**Author:** Lalitya

**Purpose:** The objective of this use case is to help students find compatible roommates based on their lifestyle preferences using an AI-based matching system. The system processes student responses and provides recommendations based on compatibility scores.

**Requirements Traceability:** Linked to F9.

**Priority:** High. This feature significantly improves student satisfaction and hostel harmony.

**Preconditions:** The student must fill out the roommate preference questionnaire, and a database of student preferences must be available.

**Postconditions:** The system generates and displays roommate match recommendations.

**Actors:** Student (completes the questionnaire), AI Matching System (processes responses and generates matches).

**Extends:** None.

**Flow of Events**

**Basic Flow:**

1. The student selects "Find a Roommate" from the app.
2. The system presents a questionnaire about lifestyle preferences (e.g., sleep schedule, cleanliness, noise tolerance).
3. The student completes and submits the questionnaire.
4. The system processes the responses using an AI algorithm.
5. The system displays the top three most compatible roommates along with a compatibility score.

**Alternative Flow:** If no matching roommates are found, the system suggests, "No matches available, please try later."

**Exceptions:** If the server is down, the request fails, and an error message is displayed.

**Includes:** None.

**Notes/Issues:** The accuracy of the roommate matching system depends on the quality of the questionnaire responses.

### 3.3.5 Use Case: Lost and Found(U5)

**Author:** Monisha

**Purpose:** The objective of this use case is to enable students and wardens to report lost or found items efficiently through a digital dashboard. Users can upload images and descriptions of lost or found items, allowing easy identification and retrieval. Once an item is claimed, users can mark it as retrieved to update the dashboard accordingly.

**Requirements Traceability:** Linked to F3.

**Priority:** High. This use case is crucial for streamlining the lost and found process, reducing manual tracking efforts, and ensuring students can quickly recover their belongings.

**Preconditions:**

- The user must have access to the mobile application.
- The dashboard must be operational with storage for images and descriptions.

**Postconditions:**

- The lost or found item is successfully posted on the dashboard.
- Once an item is retrieved, it is marked as resolved and removed from the active list.

**Actors:**

- **Student** (reports a lost item, checks the dashboard for found items, marks an item as retrieved).
- **Warden** (posts found items, verifies retrieved items).
- **Lost and Found System** (stores and displays posts).

**Extends: None.**

**Flow of Events**

**Basic Flow:**

1. The student or warden opens the Lost and Found dashboard in the mobile app.
2. The user clicks on the "Report Lost Item" or "Report Found Item" button.
3. The user uploads an image and provides a brief description (e.g., item details, location, date).
4. The system processes and adds the post to the Lost and Found dashboard.
5. Other users can browse the dashboard to check for lost or found items.
6. If a student finds their lost item, they click the "Mark as Retrieved" checkbox.
7. The system updates the item’s status, and the post is moved to the "Resolved" section.

**Alternative Flow:**

- If an item is posted but has insufficient details, the system prompts the user to provide more information before submission.
- If a found item remains unclaimed for a certain period, the warden may take action based on hostel policies.

**Exceptions:**

- If the server is down, the system displays an error message: **"Service unavailable. Please try again later."**
- If an image upload fails, the user receives a notification: **"Image upload unsuccessful. Please try again."**

**Includes: None.**

**Notes/Issues:**

- A notification system can be implemented to alert students when a new item is posted.
- Additional filtering options may be required to help users search for items efficiently.
- Future enhancements may include AI-based image recognition for automatic item categorization.

### 3.3.6 Use Case: Room cleaning schedule (U6)

**Author:** Monisha

**Purpose:**
The objective of this use case is to allow students to book room cleaning slots based on their availability and active hours. The system ensures that users can check for available slots before booking, preventing conflicts and optimizing cleaning schedules for efficiency.

**Requirements Traceability:** Linked to F4.

**Priority:** High. This use case helps maintain hygiene and cleanliness in hostels while giving students control over scheduling based on their convenience.

**Preconditions:**

- The student must have access to the mobile application.
- The cleaning schedule system must be operational with real-time slot availability tracking.

**Postconditions:**

- A student successfully books a cleaning slot.
- The system updates availability and prevents double booking.
- The warden/cleaning staff is notified of the scheduled cleaning.

**Actors:**

- **Student** (checks slot availability, books a cleaning slot, modifies/cancels bookings).
- **Cleaning Staff** (views scheduled bookings, marks cleaning as completed).
- **Room Cleaning System** (manages bookings, prevents conflicts, sends notifications).

**Extends: None.**

**Flow of Events**

**Basic Flow:**

1. The student opens the **Room Cleaning Schedule** section in the mobile app.
2. The system displays available slots for the selected date.
3. The student selects a preferred slot based on availability.
4. The system confirms whether the slot is still open.
5. If available, the student books the slot.
6. The system updates availability and sends a confirmation notification.
7. The cleaning staff receives the booking details.
8. After cleaning, the staff marks the slot as **"Completed"** in the system.

**Alternative Flow:**

- If a student tries to book a slot that is already taken, the system displays an error message: **"Slot unavailable. Please select a different time."**
- If a student wants to cancel or reschedule, they can modify their booking before a cutoff time.
- If cleaning staff are unavailable due to unforeseen circumstances, the system notifies affected students and offers rescheduling.

**Exceptions:**

- If the server is down, the system displays an error message: **"Service unavailable. Please try again later."**
- If the student doesn't select a valid slot, the system prompts: **"Please choose a valid date and time."**

**Includes: None.**

**Notes/Issues:**

- A reminder notification can be sent to students before their booked slot.
- Future enhancements may include an **auto-scheduling** feature based on past booking patterns.
- Cleaning staff can have an **admin panel** to manage and verify completed cleanings.

### 3.3.7 Use Case: Room cleaning schedule (U7)

**Author:** Monisha

**Purpose:**
The objective of this use case is to allow students to book a washing machine based on real-time availability detected by a sensor. The system ensures that students can check machine status before booking and log the type of clothes they are washing for maintenance and efficiency tracking.

**Requirements Traceability:** Linked to F5.

**Priority:** High. This use case optimizes laundry scheduling, reduces conflicts, and improves washing machine maintenance by tracking usage data.

**Preconditions:**

- The student must have access to the mobile application.
- The washing machine must be equipped with a **sensor** to detect real-time availability.
- The system must track and update washing machine usage.

**Postconditions:**

- A student successfully books a washing machine.
- The system updates real-time availability to prevent conflicts.
- The washing machine logs the type of clothes being washed for maintenance records.

**Actors:**

- **Student** (checks availability, books a slot, enters the type of clothes).
- **Washing Machine System** (detects availability, updates booking status, logs washing details).

- **Maintenance Staff** (views washing records, schedules maintenance if needed).

**Extends: None.**

**Flow of Events**

**Basic Flow:**

1. The student opens the **Washing Machine Schedule** section in the mobile app.
2. The system checks real-time availability using the **sensor**.
3. The student selects an available washing machine and preferred time slot.
4. The student enters the **type of clothes** being washed (e.g., cotton, wool, delicates).
5. The system **confirms the booking** and updates the machine's availability.
6. The student receives a **confirmation notification** with washing details.
7. After use, the machine logs washing details, and the system updates its availability.

**Alternative Flow:**

- If all machines are occupied, the system notifies the student: **"No machines available. Please try again later."**
- If the student does not select a clothing type, the system prompts them to do so before confirming the booking.
- If the washing machine is out of order, the system displays: **"Machine under maintenance. Please choose another."**

**Exceptions:**

- If the server is down, the system displays an error: **"Service unavailable. Please try again later."**
- If the sensor fails to detect real-time availability, the system alerts the maintenance staff.

**Includes: None.**

**Notes/Issues:**

- The system can send **reminder notifications** before the booked slot.
- Future enhancements may include **automated detergent recommendations** based on fabric type.
- Maintenance staff can access **usage logs** for scheduling repairs efficiently.

# 4 Other Non-functional Requirements

## 4.1 Performance Requirements

- P1: The system should process roommate matching results within 10-15 seconds.
- P2: Service booking confirmation should be instant.
- P3: Complaint submission should be acknowledged within 5-10 **seconds**.
- P4: The AI chatbot should respond to user queries within **3-4 seconds**.
- P6: The bulletin board should update announcements within 5 **seconds** of posting.
- P7: Lost and found notifications should be sent within 10 **seconds** after an item is reported

## 4.2 Safety and Security Requirements

#### *4.2.1.1 Data Security and Privacy*

- Students can view and update their profile securely, including contact information and room details.
- Privacy settings allow students to control what details, such as contact information, are visible to their roommates.
- Login authentication requires university email to prevent unauthorized access.
- Users can check their last login details to detect any suspicious activity.

#### *4.2.1.2 Mobile and Web Security*

- Login is available through university email authentication.
- If inactive, the system sends a pop-up warning before logging the user out.

#### *4.2.1.3 Complaint Management Security*

- Students can track the status of their submitted complaints.
- Complaints can be marked as urgent for issues like broken locks or security threats, ensuring faster resolution.

#### *4.2.1.4 Lost and Found System Security*

- Students can browse a list of found items with descriptions and images to check if their lost item has been reported.
- If an item matching their lost report is found, they receive a notification.
- Students must verify their identity with their hostel ID before claiming lost items.

#### *4.2.1.5 Room Cleaning & Washing Machine Scheduling Security*

- Booking confirmation is provided through a notification.

- Washing machine bookings are limited to once per day per student to ensure fair use.

***Account Deletion and Data Removal***

- If a student leaves the hostel, they can request the deletion of their account and personal data.

## 4.3 Software Quality Attributes

### 4.3.1 Reliability

- The system shall maintain 90-95**% uptime** to ensure hostel services always remain accessible.

### 4.3.2 Adaptability

- It should be capable of adding new hostel facilities and services without requiring significant code modifications.
- The UI shall be responsive and compatible across **mobile (Android/iOS) and web browsers**.

### 4.3.3 Maintainability

- Code shall be **well-documented**, and API versioning shall be maintained for backward compatibility.
- An admin dashboard shall provide logs and system performance reports for maintenance tracking.

### 4.3.4 Performance Efficiency

- The system shall be optimized to handle **at least 1000 concurrent users** without experiencing delays.
- AI-driven roommate matching shall be processed within **10-15 seconds** even under peak loads.

# Appendix A – Data Dictionary

| Variable / Constant | Type | Description | Possible State/Values | Related Variables | Functional Requirement |
|---|---|---|---|---|---|
| user_id | Integer | Unique identifier for each user | Auto-incremented | User login, complaint submission, roommate matching | F1, F9 |
| user_role | String | Defines user type | "Student", "Admin" | Role-based access | F1, F3, F6 |
| complaint_id | Integer | Unique ID for complaints | Auto-incremented | Complaint submission | F1, F2 |
| complaint_status | String | Status of a complaint | “New”, ” Closed” | Complaint submission | F1, F2 |
| lost_item_id | Integer | Unique ID for lost items | Auto-incremented | Lost item report | F3 |
| item_status | String | Status of a lost item | “lost”,”found” | Lost & found operations | F3 |

| | | | | | |
|---|---|---|---|---|---|
| schedule_id | Integer | Unique ID for cleaning schedules | Auto-incremented | Viewing schedules | F4 |
| cleaning_time | Timestamp | Time for cleaning service | Datetime format | Display cleaning schedule | F4 |
| washing_machine_id | Integer | Unique ID for washing machines | Auto-incremented | Machine booking | F5 |
| machine_status | String | Status of a washing machine | “Available” ,” Booked” | Booking operation | F5 |
| chatbot_query | String | Input message for chatbot | User input text | Query processing | F6 |
| chatbot_response | String | Chatbot output message | Auto-generated text | Display response | F6, F7 |
| food_request_id | Integer | Unique ID for food requests | Auto-incremented | Request submission | F8 |
| food_status | String | Status of food request | "Pending", "Delivered" | Request processing | F8 |
| roommate_pref_id | Integer | Unique ID for roommate preference entry | Auto-incremented | Roommate matching | F9 |
| roommate_match_score | Float | AI-generated compatibility score | 0.0 - 1.0 | Matching algorithm | F9 |

# Appendix B - Group Log

*<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document>*

***Minutes of Meeting***

**Meeting 1: Project Planning and Feature Finalization**

**Date:** 10th February 2025
**Attendees:** Guna Shritha, Monisha, Lalitya, Dhruti, Cherith
**Agenda:**

Finalizing project scope and key features.
Assigning responsibilities among team members.
Discussing the technology stack (Spring Boot, Flutter, MySQL).

**Discussion Points:**

Decided on the following core features: Complaint Management, AI Chatbot for FAQs, Washing Machine Booking, Lost and Found, Room Cleaning Schedule, Call for Food When Sick, and AI-Powered Roommate Matching.

Decided to use REST APIs for mobile app integration.
Database schema and entities were roughly outlined.

**Action Items:**

Team members to research respective feature implementations.
Database schema to be finalized in the next meeting.

**Meeting 2: Non-Functional Requirements Discussion**

**Date:** 15th February 2025
**Attendees:** Guna Shritha, Monisha, Lalitya, Dhruti, Cherith
**Agenda:**

Discussing system performance, security, and usability requirements.

**Discussion Points:**

Security and data privacy measures were reviewed, ensuring only registered users can access complaint submissions and food requests.
Login criteria discussed, considering role-based access control for students and administrators.
Chatbot response time should not exceed **2 seconds** for seamless user experience.
The system must handle at least **100 concurrent users** without performance degradation.

**Action Items:**

Implement basic authentication and access control mechanisms.
Optimize API response times to ensure smooth operation.

**Meeting 3: App Flow and UI Design Discussion**

**Date:** 1st March 2025
**Attendees:** Guna Shritha, Monisha, Lalitya, Dhruti, Cherith
**Agenda:**

Finalizing the UI flow and system interactions.

**Discussion Points:**

Basic **UI layout and navigation** were discussed for the Flutter app.
The **bottom navigation bar** will provide quick access to key sections (Complaints, Chatbot, Lost & Found, etc.).
Chatbot interface to have a **text input field with a send button** for user queries.
Complaint submission will require **student name, room number, and complaint details**.
Food request feature will prompt for **room number and illness details** before submission.

**Action Items:**

UI wireframes to be created and reviewed in the next meeting.
Ensure API endpoints align with frontend requirements.