**Contents**

[**1. Introduction 2**](#_Toc152789720)

[**1.1 Purpose 2**](#_Toc152789721)

[**1.2 Document conventions 3**](#_Toc152789722)

[**1.3 Project Scope 3**](#_Toc152789723)

[**1.4 References 4**](#_Toc152789724)

[**2. Overall Description 5**](#_Toc152789725)

[**2.1 Product perspective 5**](#_Toc152789726)

[**2.2 User classes and characteristics 6**](#_Toc152789727)

[**2.3 Operating environment 6**](#_Toc152789728)

[**2.4 Design and implementation constraints 7**](#_Toc152789729)

[**2.5 Assumptions and dependencies 8**](#_Toc152789730)

[**3. System features 9**](#_Toc152789731)

[**3.1 Add Incident 9**](#_Toc152789732)

[**3.2 Allocate Room 10**](#_Toc152789733)

[**3.3 Add Feedback 10**](#_Toc152789734)

[**3.4 Manage Staff 11**](#_Toc152789735)

[**3.5 Maintain Mess 11**](#_Toc152789736)

[**3.6 Submit Fee 12**](#_Toc152789737)

[**3.7 Register 12**](#_Toc152789738)

[**3.8 Login/Logout 12**](#_Toc152789739)

[**4. Data Requirements 13**](#_Toc152789740)

[**4.1 Logical Data Model 13**](#_Toc152789741)

[**4.2 Data Dictionary 14**](#_Toc152789742)

[**4.3 Reports 15**](#_Toc152789743)

[**4.4 Data Acquisition, Integrity, Retention, and Disposal 16**](#_Toc152789744)

[**5. External Interface Requirements 17**](#_Toc152789745)

[**5.1 User Interfaces 17**](#_Toc152789746)

[**5.2 Software Interfaces 18**](#_Toc152789747)

[**5.4 Communications Interfaces 19**](#_Toc152789748)

[**6. Quality attributes 19**](#_Toc152789749)

[**6.1 Availability: 19**](#_Toc152789750)

[**6.2 Correctness: 19**](#_Toc152789751)

[**6.3 Flexibility: 19**](#_Toc152789752)

[**6.4 Portability: 20**](#_Toc152789753)

[**6.5 Reliability: 20**](#_Toc152789754)

[**6.6 Reusability: 20**](#_Toc152789755)

[**6.7 Robustness: 20**](#_Toc152789756)

[**6.8 Testability: 20**](#_Toc152789757)

[**7. Other requirements 21**](#_Toc152789758)

[**Appendix A: Glossary 21**](#_Toc152789759)

[**Appendix B: Analysis models 21**](#_Toc152789760)

[**Activity diagram 21**](#_Toc152789761)

# **1. Introduction**

## **1.1 Purpose**

This Software Requirements Specification (SRS) document pertains to the Hostel Management System (HMS), version 1.0. The HMS is a comprehensive software solution designed to streamline and enhance the management of hostel operations within a university environment. The scope of this SRS encompasses the entire HMS, including its modules for incident management, room allocation, feedback, staff management, mess operations, fee submission, and user authentication. The document outlines the functional and non-functional requirements necessary for the development and successful deployment of the complete HMS system. It serves as a guide for developers, stakeholders, and end-users to understand the features, constraints, and quality attributes of the Hostel Management System.Top of Form

## **1.2 Document conventions**

**1.2.1 Overview** The document follows industry-standard conventions to ensure clarity, consistency, and ease of understanding for all stakeholders involved in the development and deployment of the Hostel Management System (HMS).

**1.2.2 Conventions Used**

1. **Naming Conventions:**
   * CamelCase is used for naming variables and functions in the code.
   * Underscore (\_) is used to separate words in file names.
2. **Formatting Conventions:**
   * Times New Roman font is used for the entire document.
   * Headings are in bold for easy identification.
   * Bullet points and numbering follow a consistent pattern for readability.
3. **Document Structure:**
   * Functional requirements are presented in Section 3, while non-functional requirements are detailed in Section 6.
4. **Version Control:**
   * Version numbers follow the semantic versioning format (e.g., 1.0, 1.1, 2.0).

### **1.3 Project Scope**

**1.3.1 Overview** The project scope outlines the boundaries and limitations of the Hostel Management System (HMS), defining the extent of its functionalities and features.

**1.3.2 Scope Inclusions**

1. Incident Management Module:
   * Recording and tracking incidents within the hostel.
2. Room Allocation Module:
   * Efficient allocation of rooms to residents.
3. Feedback Module:
   * Facilitating resident feedback on hostel services.
4. Staff Management Module:
   * Administration of staff details and roles.
5. Mess Management Module:
   * Planning and maintenance of hostel mess operations.
6. Fee Submission Module:
   * Management and tracking of hostel fees.
7. Authentication Module:
   * User registration, login, and logout functionalities.

**1.3.3 Scope Exclusions**

1. Financial Systems Integration:
   * Integration with external financial systems is not within the scope of this project.
2. Advanced Analytics:
   * In-depth data analytics beyond standard reporting is not included in the initial scope.

### **1.4 References**

**1.4.1 Relevant Documents** The development and deployment of the Hostel Management System (HMS) refer to the following documents for guidance and standardization:

1. IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.
2. University Data Privacy and Security Policies.
3. University Technology Stack Guidelines.

**1.4.2 External References** External tools, frameworks, and technologies referenced in the development of the HMS include:

1. Database Management System (DBMS): MySQL version 8.0.
2. Web Development Framework: Django version 3.0.
3. Web Browsers: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.

# 2. Overall Description

## **2.1 Product perspective**

The Hostel Management System (HMS) is a standalone software designed to efficiently manage and streamline various aspects of hostel operations. It is not part of a larger system but is intended to be an independent solution catering specifically to the needs of hostel management.

*System Components:*

The major components of the Hostel Management System include:

1. User Interface: The interface allows users, including administrators, staff, and residents, to interact with the system seamlessly.
2. Database: The system relies on a robust database to store and manage information related to residents, incidents, room allocations, staff details, mess management, fees, and user accounts.
3. Functional Modules:
   * Incident Management Module: Facilitates the recording and tracking of incidents within the hostel.
   * Room Allocation Module: Manages the allocation of rooms to residents efficiently.
   * Feedback Module: Enables residents to provide feedback on hostel services.
   * Staff Management Module: Administers the details and roles of hostel staff.
   * Mess Management Module: Handles the planning and maintenance of hostel mess operations.
   * Fee Submission Module: Manages the submission and tracking of hostel fees.
   * Authentication Module: Responsible for user registration, login, and logout functionalities.

*Subsystem Interconnections:*

The subsystems are interconnected to ensure a smooth flow of information and processes. For example:

* The Incident Management Module may interact with the User Interface to display incident reports.
* The Room Allocation Module may communicate with the Database to update resident room assignments.
* The Authentication Module is linked to all other modules, ensuring secure access to relevant functionalities.

*External Interfaces:*

The Hostel Management System may interact with external systems for supplementary functionalities, such as financial systems for fee processing.

## **User classes and characteristics**

1. **Administrator:**

**Characteristics:** University staff with high privileges and technical expertise.

**Functions:** Comprehensive access for overall hostel management.

1. **Staff:**

**Characteristics:** University employees managing hostel operations.

**Functions:** Access to specific modules aligned with their roles.

1. **Residents (University Students):**

**Characteristics:** Students with varying technical skills and limited access.

**Functions:** Personal modules for feedback, room details, and fee submission.

1. **System Guests:**

**Characteristics:** Visitors with basic technical proficiency and minimal access.

**Functions:** Limited access for viewing general hostel information.

1. **Support/IT Personnel:**

**Characteristics:** Technical support staff ensuring system functionality.

**Functions:** High-level access for maintenance, troubleshooting, and support.

## **2.3 Operating environment**

The Hostel Management System (HMS) is tailored for operation within the university environment, specifically catering to the needs of students and hostel administrators. In terms of hardware, the system requires a robust server hosted by the university to manage centralized data and access control efficiently. This server should run a compatible server operating system such as Windows Server, Linux (e.g., Ubuntu Server, CentOS), or another suitable server OS.

End-users, including residents, staff, and administrators, can access the HMS through various client devices. Access to the HMS is facilitated through modern web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

Moreover, the HMS may need to integrate with other university systems, such as student databases or financial systems, to ensure seamless data flow and accuracy. Regular updates and compatibility checks will be conducted to maintain a versatile and secure operating environment, aligning with the dynamic nature of university systems and technologies.

## **2.4 Design and implementation constraints**

Several items and issues will shape the design and implementation of the Hostel Management System (HMS), imposing specific constraints on the development process. These constraints include:

1. **Regulatory Compliance:**

The HMS must adhere to relevant regulatory policies and standards governing data privacy and security, particularly as it involves sensitive information related to university students and staff.

1. **Hardware Limitations:**

The design and implementation must consider hardware limitations, including timing and memory requirements. The system should be optimized for efficient performance on a variety of devices, especially those used by students and staff.

1. **Database Technology:**

The selection of a specific database management system (DBMS) is a constraint, requiring compatibility with university standards and existing infrastructure. The chosen DBMS (e.g., MySQL, PostgreSQL) should align with the university's technology stack.

1. **Integration with Existing Systems:**

Integration with other university systems, such as student databases or financial systems, may be constrained by existing technologies and protocols. The HMS development must align with these systems to ensure seamless data flow.

1. **Design Conventions and Programming Standards:**

The development team must follow established design conventions and programming standards. If the customer's organization will be responsible for maintaining the software, adherence to their coding practices is essential.

1. **Parallel Operations:**

The development and deployment of the HMS must be conducted with minimal disruption to ongoing hostel operations. This constraint necessitates careful planning and coordination to avoid conflicts during system integration.

1. **Language Requirements:**

The system must support multiple languages to accommodate a diverse user base. Language localization should be a consideration during the design and implementation phases.

These constraints provide a framework for the development team, ensuring that the HMS aligns with university policies, technological infrastructure, and operational requirements while delivering a secure and effective hostel management solution.

## **2.5 Assumptions and dependencies**

Several assumptions and dependencies influence the requirements and successful implementation of the Hostel Management System (HMS):

**Assumptions:**

1. Third-Party Components: It is assumed that third-party components, such as database management systems, will function as expected and integrate seamlessly with the HMS. Any changes or issues with these components may impact development timelines.
2. Regulatory Compliance: Assumption that the university's regulatory environment remains stable, and any changes in data privacy or security regulations will be communicated promptly to ensure compliance.
3. Hardware Stability: It is assumed that the university's hardware infrastructure remains stable and can support the requirements of the HMS, including server capacities and network capabilities.
4. User Availability: Assumption that users, including administrators, staff, and students, have reliable internet access and compatible devices to interact with the HMS.
5. Availability of Development Resources: Assuming that necessary development resources, including skilled personnel and development tools, are consistently available throughout the project lifecycle.

**Dependencies:**

1. External System Integration: The HMS is dependent on successful integration with existing university systems, such as student databases and financial systems. Delays or changes in these external systems may impact the project timeline.
2. Web Browser Compatibility: Dependencies on web browser compatibility for user access. Any changes in browser standards or updates may affect the user experience.
3. University Policies: The project depends on adherence to university policies, including security protocols, coding standards, and technology preferences. Changes in these policies may require adjustments to the HMS.
4. External Software Updates: Dependencies on updates or changes in third-party software components (e.g., web frameworks, databases). Changes to these components may require adjustments to the HMS.

These assumptions and dependencies underscore the importance of ongoing communication, monitoring, and flexibility throughout the development process to address any unforeseen changes or challenges.

# **3.** **System features**

### **3.1 Add Incident**

**3.1.1 Description** This feature allows authorized users to record and track incidents that occur within the hostel. Incidents may include maintenance issues, rule violations, or any other noteworthy occurrences.

**3.1.2 Functional Requirements**

1. **User Authentication:**
   * The system should authenticate users before allowing them to add incidents.
   * Only authorized staff members and administrators can access this functionality.
2. **Incident Recording:**
   * Users should be able to enter details of the incident, including date, time, location, and a description.
   * Optionally, users can attach relevant files or images to provide additional information.
3. **Incident Tracking:**
   * The system should maintain a log of all incidents for future reference.
   * Staff and administrators should have the ability to view, edit, or close incidents.

### **3.2 Allocate Room**

**3.2.1 Description** This feature manages the allocation of rooms to residents efficiently, ensuring a smooth and organized hostel living experience.

**3.2.2 Functional Requirements**

1. **Room Availability Check:**
   * The system should display the availability of rooms based on various criteria (e.g., occupancy, room type).
2. **Room Allocation:**
   * Staff members should be able to allocate rooms to residents, considering preferences and availability.
   * The system should prevent over-allocation or conflicting room assignments.
3. **Room Transfer:**
   * Residents may request room transfers, and staff should have the ability to approve or deny such requests.
   * Transfer history and reasons should be logged for reference.

### **3.3 Add Feedback**

**3.3.1 Description** This feature enables residents to provide feedback on hostel services, contributing to continuous improvement.

**3.3.2 Functional Requirements**

1. **User Interface:**
   * The system should provide an intuitive interface for residents to submit feedback.
2. **Feedback Categories:**
   * Residents should be able to select from predefined categories (e.g., facilities, cleanliness, staff behavior) when submitting feedback.
3. **Anonymous Feedback:**
   * Optionally, residents can choose to submit feedback anonymously to encourage honest opinions.

### **3.4 Manage Staff**

**3.4.1 Description** This feature is designed for administrators to oversee and manage details and roles of hostel staff.

**3.4.2 Functional Requirements**

1. **Staff Information:**
   * The system should store details of hostel staff, including personal information and roles/responsibilities.
2. **Role Assignment:**
   * Administrators should have the ability to assign roles and permissions to staff members.
3. **Staff Performance Tracking:**
   * Optionally, the system may include features to track staff performance and activities within the hostel.

### **3.5 Maintain Mess**

**3.5.1 Description** This feature handles the planning and maintenance of hostel mess operations, including menu planning and food quality management.

**3.5.2 Functional Requirements**

1. **Menu Planning:**
   * Staff should be able to plan daily, weekly, or monthly menus.
   * The system should consider dietary preferences and restrictions.
2. **Mess Attendance:**
   * Track resident mess attendance for billing purposes.
3. **Feedback Integration:**
   * Integrate resident feedback on mess services for continuous improvement.

### **3.6 Submit Fee**

**3.6.1 Description** This feature manages the submission and tracking of hostel fees by residents.

**3.6.2 Functional Requirements**

1. **Fee Details:**
   * Residents should be able to view details of their hostel fees, including due dates and amounts.
2. **Payment Options:**
   * Provide multiple payment options such as online transactions, bank transfers, or manual submission.
3. **Late Fee Calculation:**
   * Automatically calculate and apply late fees for overdue payments.

### **3.7 Register**

**3.7.1 Description** This feature allows users to register for the Hostel Management System.

**3.7.2 Functional Requirements**

1. **User Registration Form:**
   * Users should fill out a registration form with necessary details.
2. **Verification Process:**
   * Implement a verification process to ensure the authenticity of user registration.

### **3.8 Login/Logout**

**3.8.1 Description** This feature enables users to log in and out of the Hostel Management System securely.

**3.8.2 Functional Requirements**

1. **Secure Authentication:**
   * Use secure authentication methods such as username/password or two-factor authentication.
2. **Session Management:**
   * Automatically log out users after a period of inactivity for security purpose.

# **4. Data Requirements**

## **4.1 Logical Data Model**

**4.1.1 Overview** The logical data model for the Hostel Management System (HMS) represents the structure and relationships of data entities essential for effective hostel management.

**4.1.2 Entities and Relationships**

1. **User Entities:**
   * Administrator
   * Staff
   * Residents
   * System Guests
   * Support/IT Personnel
2. **Functional Modules:**
   * Incident Management
   * Room Allocation
   * Feedback
   * Staff Management
   * Mess Management
   * Fee Submission
   * Authentication
3. **Common Entities:**
   * Room
   * Maintenance
   * Visitor
   * Event
   * Notification
   * Security Logs
4. **Attributes and Relationships**
   * User entities have attributes like username, password, name, contact details, and roles.
   * Room entity has attributes like room number, type, occupancy status, and related maintenance records.
   * Incident entity includes details such as incident ID, date, time, description, and associated user information.
   * Feedback entity contains feedback ID, category, description, and user details.

## **4.2 Data Dictionary**

**4.2.1 Overview** The data dictionary provides a comprehensive reference for data elements used in the HMS, specifying their definitions, formats, and relationships.

**4.2.2 Key Data Elements**

1. **User Table:**
   * UserID (Primary Key)
   * Username
   * Password
   * Name
   * Contact Number
   * Role
2. **Room Table:**
   * RoomID (Primary Key)
   * RoomNumber
   * RoomType
   * OccupancyStatus
   * MaintenanceID (Foreign Key)
3. **Incident Table:**
   * IncidentID (Primary Key)
   * Date
   * Time
   * Description
   * UserID (Foreign Key)
4. **Feedback Table:**
   * FeedbackID (Primary Key)
   * Category
   * Description
   * UserID (Foreign Key)
5. **Maintenance Table:**
   * MaintenanceID (Primary Key)
   * Date
   * Time
   * Description
   * RoomID (Foreign Key)

**4.2.3 Relationships**

1. User-to-Incident Relationship:
   * One-to-Many relationship between User and Incident.
2. Room-to-Maintenance Relationship:
   * One-to-One relationship between Room and Maintenance.
3. User-to-Feedback Relationship:
   * One-to-Many relationship between User and Feedback.

## **4.3 Reports**

**4.3.1 Overview** The reporting system within the HMS provides administrators with valuable insights into hostel operations. Various reports aid in decision-making and performance assessment.

**4.3.2 Key Reports**

1. **Occupancy Report:**
   * Provides data on room occupancy status, helping administrators manage room allocations effectively.
2. **Incident Summary Report:**
   * Summarizes incidents reported within a specific period, including their resolution status.
3. **Fee Submission Report:**
   * Details fee submission status, outstanding payments, and late fees.
4. **Feedback Analysis Report:**
   * Analyzes feedback received from residents, highlighting areas for improvement.
5. **Maintenance History Report:**
   * Lists historical maintenance activities for rooms and common areas.

## **4.4 Data Acquisition, Integrity, Retention, and Disposal**

**4.4.1 Data Acquisition**

1. User Registration:
   * User data is acquired through the registration process, including necessary details for identification and role assignment.
2. Incident Recording:
   * Data related to incidents is acquired when users report issues or noteworthy occurrences.
3. Room Allocation:
   * Data related to room allocation is acquired during the room allocation process, capturing resident details and preferences.

**4.4.2 Data Integrity**

1. Referential Integrity:
   * Enforce referential integrity constraints to maintain consistency between related tables (e.g., User and Incident).
2. Validation Rules:
   * Implement validation rules to ensure data accuracy, such as valid date formats and allowable room types.

**4.4.3 Data Retention**

1. Archiving Historical Data:
   * Maintain a historical archive for critical data, such as user activities and incident records.
2. Retention Policies:
   * Define data retention policies to determine the duration for which specific data is retained.

**4.4.4 Data Disposal**

1. Secure Deletion:
   * Implement secure data deletion processes to permanently remove sensitive information from the system.
2. Compliance with Regulations:
   * Adhere to regulatory requirements for data disposal, ensuring proper erasure of personal and sensitive data.

# **5. External Interface Requirements**

# 5.1 User Interfaces

**5.1.1 Overview** The user interfaces of the Hostel Management System (HMS) are designed to provide a seamless and intuitive experience for different categories of users.

**5.1.2 User Classes and Interfaces**

1. **Administrator Interface:**
   * Comprehensive dashboard displaying key metrics and access to all modules.
   * Interactive tools for user management, report generation, and system configuration.
2. **Staff Interface:**
   * Module-specific interfaces for efficient management of incidents, room allocations, feedback, and other operational tasks.
   * Accessible tools for staff to view and update relevant information.
3. **Resident Interface:**
   * Personalized dashboard showing room details, fee status, and feedback options.
   * Intuitive interfaces for fee submission, incident reporting, and event participation.
4. **System Guest Interface:**
   * Limited access interface providing basic information about the hostel without requiring login.
   * Read-only access to general hostel information.
5. **Support/IT Personnel Interface:**
   * Specialized interface for technical support staff with access to system maintenance and troubleshooting tools.
   * Comprehensive logs and diagnostic features for issue resolution.

## 5.2 Software Interfaces

**5.2.1 Database Management System (DBMS)**

* The HMS interacts with a chosen DBMS (e.g., MySQL, PostgreSQL) to store and retrieve data.
* Database queries for efficient data processing and retrieval.

**5.2.2 Web Browsers**

* The system is compatible with modern web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

**5.2.3 Third-Party Integrations**

* Interfaces with external systems, such as financial systems for fee processing.

#### 5.3 Hardware Interfaces

**5.3.1 Server Hardware**

* The HMS requires a robust server hosted by the university to manage centralized data and access control efficiently.

**5.3.2 Client Devices**

* End-users access the HMS through various client devices, including desktop computers, laptops, tablets, and smartphones.

## 5.4 Communications Interfaces

**5.4.1 Internet Communication**

* Communication between the HMS and end-users occurs over the internet, ensuring remote access for residents and staff.

**5.4.2 Email Notifications**

* The system sends email notifications for important updates, such as incident resolutions, fee deadlines, and event announcements.

**5.4.3 API Integration**

* Potential integration with university systems through APIs to ensure seamless data flow and accuracy.

**5.4.4 Security Protocols**

* Secure communication protocols (e.g., HTTPS) to safeguard data during transmission between the server and client devices.

# **6. Quality attributes**

## **6.1 Availability:**

Aim for 99.9% availability to ensure uninterrupted access for users, with scheduled downtimes communicated in advance.

## **6.2 Correctness:**

Ensure that the system consistently produces correct results for tasks such as fee calculations, room allocations, and data retrieval.

## **6.3 Flexibility:**

Provide customizable settings or configurations to meet varying requirements of different hostels within the university.

## **6.4 Portability:**

Ensure compatibility with major web browsers and support for different operating systems (Windows, macOS, Linux).

## **6.5 Reliability:**

Minimize system failures, errors, and downtime to enhance user trust in the reliability of the HMS.

## **6.6 Reusability:**

Design modular and well-documented code, promoting the reuse of components in future developments.

## **6.7 Robustness:**

Implement error-handling mechanisms and conduct thorough testing to ensure the system gracefully handles unforeseen scenarios.

## **6.8 Testability:**

Design the system with a modular structure to facilitate unit testing, integration testing, and overall system testing.

Note: The criteria associated with each attribute should be used as measurable benchmarks during the development and testing phases to ensure the overall quality and effectiveness of the Hostel Management System.

# **7. Other requirements**

## **Appendix A: Glossary**

## **Appendix B: Analysis models**

### **Activity diagram**

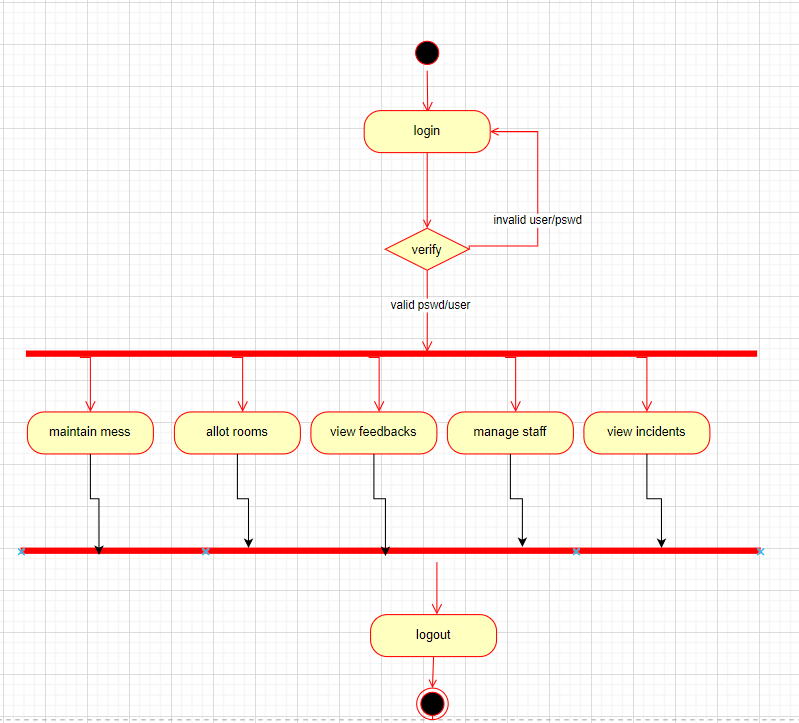


Figure 2 Activity diagram - admin activities

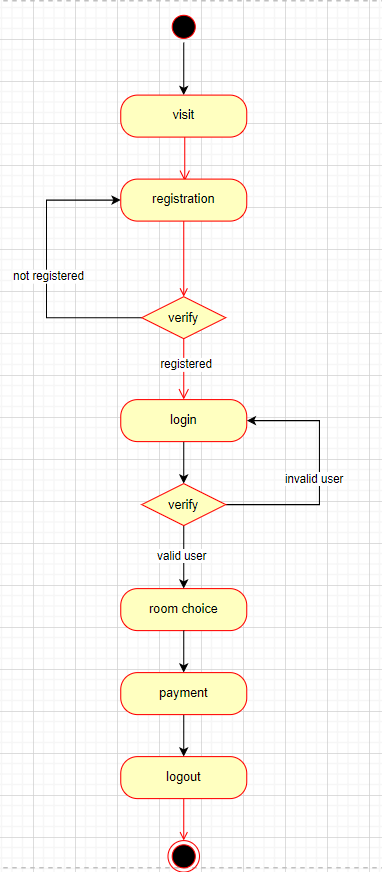


Figure Activity diagram - student activity