**UNIVERSITY OF ENGINEERING & TECHNOLOGYMARDAN**

**DEPARTMENT OF COMPUTER SCIENCE**

FINAL YEAR PROJECT PROPOSAL



**TITLE OF THE PROJECT: Hostel Management System**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Registration #** | **Email** |
| Mumtaz Ali | 21MDBCS124 | Engrmumtazali01@gmail.com |
| Muhammad Maaz | 21MDBCS151 | maazkhan29456@gmail.com |
| Muhammad Abubakar | 21MDBCS169 | m.abubaakar755@gmail.com |

**Proposed Supervisor:**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Contents

[Introduction 2](#_Toc185462483)

[Problem Statement 3](#_Toc185462484)

[Objectives 3](#_Toc185462485)

[System Models 13](#_Toc185462486)

[Use Case Diagram 15](#_Toc185462487)

# Introduction

The Hostel Management System is a comprehensive software solution designed to automate and streamline hostel operations, addressing the challenges faced by traditional manual management methods [[1]](#_References). This innovative system replaces outdated processes with a user-friendly interface that benefits hostel administrators, staff, and students alike [[2]](#_References). By efficiently handling tasks such as room bookings, student registrations, and fee payments, the system not only saves time but also significantly reduces errors, ultimately enhancing the overall experience for all stakeholders involved in hostel life [[1]](#_References) [[3]](#_References).

Key features of the Hostel Management System include:

1. Centralized Database: Maintains accurate and up-to-date information on students, rooms, and financial transactions [[2]](#_References).
2. Automated Room Allocation: Streamlines the process of assigning rooms based on availability and student preferences [[1]](#_References).
3. Online Fee Management: Facilitates easy payment and tracking of hostel fees, reducing administrative burden [3].
4. Complaint Management: Provides a platform for students to register and track their complaints or maintenance requests [2].
5. Visitor Management: Enhances security by keeping track of visitors entering and leaving the hostel premises [[1]](#_References).

# Problem Statement

Managing hostel facilities manually is inefficient, error-prone, and lacks real-time data handling. The specific problem addressed by this system is to replace manual data entry, room allocation, and fee management with a computerized system that is faster, more accurate, and capable of handling various administrative tasks seamlessly.

# Objectives

* Room and Hostel Management
* Automated Mess Bill Notification
* Biometric Attendance Integration
* Online Payment Integration
* Complaint Management System

**Non-Functional Requirements**

**1. Performance Requirements**

* **System Response Time**:
  + The system should respond to all queries (e.g., student login, attendance marking, fee status checks) within **2 seconds**.
  + Biometric attendance verification (fingerprint data) should take no more than **3 seconds**.
* **Data Processing**:
  + The system should handle up to **500 concurrent users** (students, admins, and visitors) without performance degradation.
* **Attendance Updates**:
  + Attendance and mess status should be updated **instantly** after fingerprint verification.

**2. Scalability Requirements**

* The system must be scalable to accommodate:
  + **1000+ students** and their records.
  + **Multiple hostels** with different room configurations.
  + **Daily attendance records** for mess management.
* Database should support **horizontal scaling** to add new hostels, messes, or rooms.

**3. Availability Requirements**

* The system should have **99.9% uptime** to ensure accessibility for admins, students, and staff.
* Scheduled maintenance should not exceed **4 hours per month**.

**4. Security Requirements**

* **Data Protection**:
  + All sensitive data (e.g., passwords, fingerprint data) must be encrypted using industry standards like **AES-256**.
* **Role-Based Access Control (RBAC)**:
  + Admins can perform CRUD operations on all entities.
  + Students can only access their information, view their attendance, and fee status.
* **Biometric Security**:
  + Fingerprint data should be securely stored in a **hashed format** to prevent misuse.
* **Audit Trail**:
  + Maintain logs for all admin activities, including:
    - Updates to fee statuses.
    - Room allocations.
* **Authentication**:
  + Students and admins must log in using **secure credentials**.
  + Admins require **two-factor authentication**.

**5. Usability Requirements**

* The user interface (UI) should be:
  + **Intuitive** and easy to navigate for non-technical users like students and hostel wardens.
  + Accessible across devices: desktops, tablets, and mobile phones.
* **Language Support**:
  + The system should support **multi-language options** (e.g., English and local languages).

**6. Reliability Requirements**

* The system should:
  + Ensure accurate attendance marking with **99.5% accuracy**.
  + Maintain consistency of data for students' fee statuses and complaints.
* **Backups**:
  + Automatic backups should be taken **daily** to prevent data loss.

**7. Maintainability Requirements**

* Code should follow **modular structure** for easy updates.
* The system should allow:
  + Addition of new hostels, mess menus, and rooms without impacting the existing functionality.
* **Logs and Debugging**:
  + All errors must be logged for efficient troubleshooting.

**8. Portability Requirements**

* The system should be deployable on:
  + Web servers (e.g., Apache, NGINX).
  + Cloud platforms like **AWS** or **Azure**.
* Compatible with modern browsers: Chrome, Firefox, Edge.

**9. Compliance Requirements**

* The system should comply with:
  + **GDPR** for data privacy (student and biometric data).
  + **ISO 27001** standards for information security.
* Ensure **legal compliance** for storing and using fingerprint data.

**Functional Requirements**

**1. User Management**

* **Admin Functionalities**:
  + Admin can **create, update, delete, and view** student records.
  + Admin can manage room allocations (assign or unassign rooms to students).
  + Admin can **approve or reject visitor requests**.
  + Admin can manage complaints lodged by students.
  + Admin can **update fee statuses** (e.g., paid/unpaid).
  + Admin can **generate reports** on attendance, payments, and complaints.
* **Student Functionalities**:
  + Students can **log in** to view their profile details.
  + Students can **check their room information**.
  + Students can **view fee status** for hostel and mess.
  + Students can **mark complaints** against management.
  + Students can check **attendance records** (mess and biometric attendance).
* **Authentication**:
  + System must support secure login for both admins and students.
  + Password must be **hashed and stored securely**.

**2. Room Management**

* Admin can:
  + Add new rooms and specify room type (e.g., single, two-seater, three-seater, six-seater).
  + Assign students to rooms based on availability.
  + Update room status (e.g., "Occupied" or "Available").
* Students can view their **room details** (Room No., Type, Floor, etc.).

**3. Hostel Management**

* Admin can:
  + Add or update hostel details (name, address, room capacity).
  + View the **number of rooms** and room types available.
  + Track the total **number of students** in the hostel.

**4. Mess Management**

* Admin can:
  + Add or update **mess menus** for different days.
  + Track students’ **mess attendance** based on biometric verification.
  + Generate mess bills based on attendance records.
* Students can:
  + View their **mess attendance** records.
  + View the **mess menu** for the day or week.

**5. Attendance Management**

* Students mark their mess attendance using **biometric fingerprint verification**.
* System stores:
  + Attendance Date, Time, and Status (e.g., Present/Absent).
* Admin can view:
  + Daily/Monthly attendance reports for all students.
* System generates mess bills based on **number of meals attended**.

**6. Visitor Management**

* Admin can:
  + Manage visitor requests submitted by students.
  + Approve or reject visitor entry based on submitted requests.
* Students can:
  + Submit **visitor details** (Name, Contact Info, Time In/Out).
  + Request approval from the admin for visitor entry.

**7. Complaint Management**

* Students can:
  + Lodge complaints against hostel management or services.
  + Specify the **complaint type** and provide a description.
* Admin can:
  + View, update, or resolve complaints.
  + Track the status of complaints (e.g., Open, In Progress, Resolved).

**8. Payment Management**

* Admin can:
  + Add payment details for each student (e.g., mess fees, hostel fees).
  + Update fee status as **Paid** or **Unpaid**.
* Students can:
  + View their payment history, including:
    - Fee type (Mess/Hostel)
    - Amount Paid
    - Due dates and payment status.

**9. Reports and Analytics**

* **Admin Reports**:
  + Generate monthly or weekly reports for:
    - Attendance (Hostel & Mess)
    - Payments (Paid/Unpaid Students)
    - Room allocations
    - Complaints status
* **Student Reports**:
  + Generate personal attendance records and payment details.

**10. Notification System**

* System will notify:
  + Students about **fee payment deadlines**.
  + Admins about pending complaints and visitor requests.
  + Students about the **approval or rejection** of visitors.

**System Models**

**1. User Interaction Model**

* **Description**:  
  This model focuses on how users (Admins and Students) interact with the system.
* **Components**:
  + Admin interacts via a **web interface** to manage rooms, complaints, payments, and attendance.
  + Students interact via the **web portal** to view their data, submit complaints, and check attendance.
  + Biometric devices integrate with the system for mess attendance verification.

**2. Process Model**

* **Description**:  
  This model represents the flow of data and actions:
  + Student logs into the system → Views their attendance, payments, and room details → Submits complaints.
  + Admin logs into the system → Updates room allocations, resolves complaints, and monitors student attendance.
* **Key Steps**:
  + **Login**: Authentication based on role (Admin/Student).
  + **Attendance**: Biometric input → Database update.
  + **Complaint Handling**: Student complaint → Admin resolves → Status updates.
  + **Payment Updates**: Admin updates payment status → Students view fee history.

**3. Data Model**

* **Description**:  
  This model focuses on how data is stored and linked in the database.
* **Entities**:
  + Hostel
  + Room
  + Student
  + Admin
  + Visitors
  + Complaint
  + Mess
  + Attendance
  + Payment
* **Relationships**:
  + Room → Student (One-to-Many)
  + Mess → Attendance (One-to-Many)
  + Student → Payment (One-to-Many)
  + Admin → Complaint (One-to-Many)

**Hardware Requirements**

**Server-Side Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| | **Component** |  | | --- | --- | | **Specification** |
| |  |  | | --- | --- | | **Processor** |  | | Intel Xeon or AMD Ryzen 8-Core |
| |  |  | | --- | --- | | **RAM** |  | | 16 GB or higher |
| |  |  | | --- | --- | | **Network** |  | | Gigabit Ethernet Connection |
| **Operating System** | Windows 10/11, macOS, Linux |
| |  |  | | --- | --- | | **Storage** |  | | |  |  | | --- | --- | |  | 128 GB HDD/SSD |  |  | | --- | |  | |

**Client-Side Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| | **Component** |  | | --- | --- | | **Specification** |
| |  |  | | --- | --- | | **Processor** |  | | Intel Xeon or AMD Ryzen 8-Core |
| |  |  | | --- | --- | | **RAM** |  | | 16 GB or higher |
| **Operating System** | Windows 10/11, macOS, Linux |
| |  |  | | --- | --- | | **Storage** |  | | |  |  | | --- | --- | |  | 128 GB HDD/SSD |  |  | | --- | |  | |
| |  | | --- | | **Browser** | | Google Chrome, Firefox, Microsoft Edge |

**Biometric Device Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| | **Component** |  | | --- | --- | | **Specification** |
| |  |  | | --- | --- | | **Device** |  | | Fingerprint Scanner (DigitalPersona, SecuGen, or similar) |
| |  |  | | --- | --- | | **Connectivity** |  | | USB 2.0/3.0 or Wireless |
| |  |  | | --- | --- | | **Storage** |  | | |  |  | | --- | --- | |  | 128 GB HDD/SSD |  |  | | --- | |  | |

**Software Requirements**

**Server-Side Software**

|  |  |
| --- | --- |
| Software | Specification |
| Backend Framework | Django (Python-based web framework) |
| Programming Language | Python 3.8 or higher |
| Database | SQLite (Lightweight, embedded database) |
| Web Server | Gunicorn or uWSGI |
| Libraries/Dependencies | Django ORM for database interactions |
| Operating System | Windows 10/11, Ubuntu 20.04 LTS, or macOS |
| Storage | File system for storing images or data files. |

**Client-Side Software**

|  |  |
| --- | --- |
| Software | Specification |
| Frontend | HTML5, CSS3, JavaScript, Bootstrap |
| UI Framework | Bootstrap 5 (for responsive UI design) |
| Browser | Google Chrome, Mozilla Firefox, Microsoft Edge |
| Additional Libraries | jQuery, React.js (optional) |

**Development Tools**

|  |  |
| --- | --- |
| Tool/Software | Specification |
| IDE/Editor | VS Code, PyCharm, or Sublime Text |
| Version Control | Git (GitHub for repository management) |
| Testing Tools | Django Test Framework (built-in) |
|  | PyTest for unit testing |
| Virtual Environment | venv or virtualenv (for dependency isolation) |
| Package Management | pip (Python's package installer) |
| Deployment | Heroku (Free for deployment testing) |
|  | PythonAnywhere (Django-friendly hosting) |

**Database Management**

|  |  |
| --- | --- |
| Software Component | Specification |
| Database | SQLite3 (Built-in Django DB engine) |
| Management Tool | Django Admin Panel (for database CRUD operations) |
| Data Backup | SQLite .db file stored securely in file system. |
| Database File | Stored locally as db.sqlite3. |

**Biometric Integration Software**

|  |  |
| --- | --- |
| Component | Specification |
| Integration SDK | Fingerprint SDK (e.g., SecuGen, DigitalPersona) |
| Integration Tools | Python-based API or libraries like pyUSB for device communication |
| Storage Format | Fingerprint data stored securely as hashed BLOB. |

**Network Requirements**

|  |  |
| --- | --- |
| Component | Specification |
| Internet Speed | Minimum 10 Mbps for smooth operation |
| Network Latency | Should not exceed 100 ms |
| Bandwidth | 1 GBPS for server-side connectivity |

# 

# System Models

## Use Case Diagram

Use Cases and Interactions

For Student

1. Log In:
   * The student logs into the Hostel Management System to gain access to the features.
2. Scan Fingerprint:
   * The student scans their fingerprint for verification to mark attendance.
   * The system verifies the fingerprint against stored biometric data.
3. Mark Attendance:
   * Once the fingerprint is verified, the student can mark their attendance.
4. Update Mess Attendance:
   * The system updates the student's attendance record in the mess attendance log.
5. View Attendance:
   * Students can view their attendance records to check their attendance status.

For Admin

1. Manage Students:
   * Admins can add, update, or delete student records, including room assignments and attendance logs.
2. Generate Reports:
   * Admins can generate reports related to attendance, mess bills, or student activity.
3. View Reports:
   * Admins can view the generated reports to monitor hostel and mess activities.

A diagram of a company

Description automatically generated

## Sequence Diagram

Actors and Components

1. Student: Initiates the attendance process by logging in and interacting with the biometric device.
2. Biometric Device: Scans the student's fingerprint and sends the data for verification.
3. System: Validates credentials, verifies fingerprints, updates attendance records, and fetches attendance details.
4. Attendance Database: Stores attendance records and returns the requested data.

Steps Involved

1. Log In

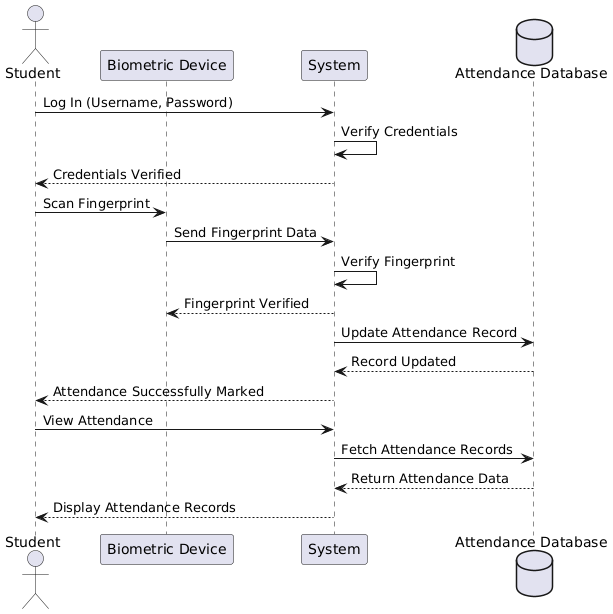
* Student → System: The student logs in using their username and password.
* System → Attendance Database: Verifies the credentials.
* System → Student: Sends a response indicating that credentials are verified.

2. Fingerprint Scanning and Verification

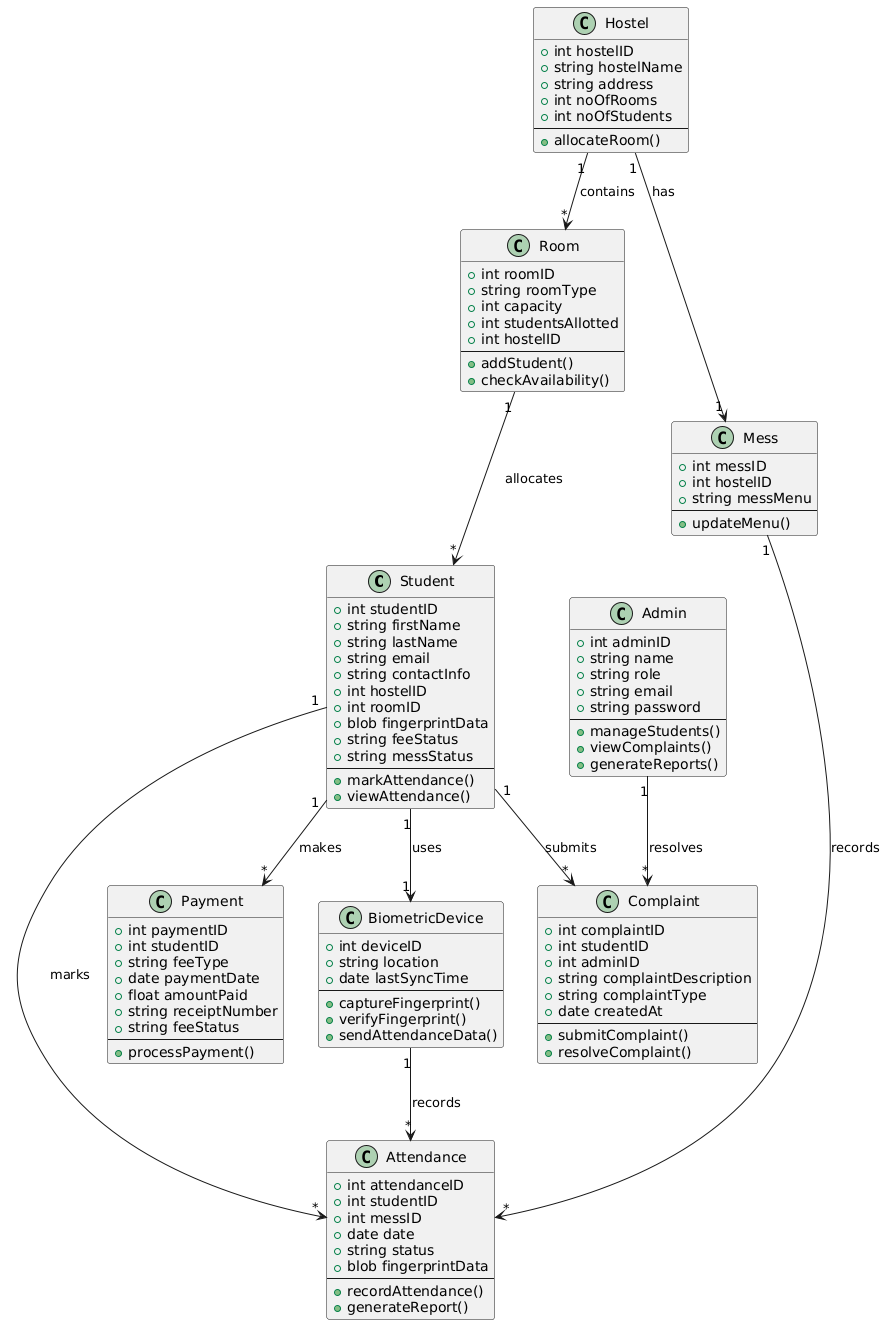
* Student → Biometric Device: The student scans their fingerprint.
* Biometric Device → System: Sends the fingerprint data to the system.
* System → Attendance Database: Verifies the fingerprint against stored data.
* System → Biometric Device: Sends confirmation that the fingerprint is verified.
* Biometric Device → System: Marks attendance.
* System → Attendance Database: Updates the attendance record.
* Attendance Database → System: Confirms that the record is updated.
* System → Student: Sends a message that attendance was successfully marked.

3. View Attendance

* Student → System: The student requests to view attendance records.
* System → Attendance Database: Fetches the attendance records.
* Attendance Database → System: Returns the attendance data.
* System → Student: Displays the attendance records to the student.



**Class Diagram:**



**Flow Chart:**

A diagram of a software system

Description automatically generated