COMSATS University Islamabad, Abbottabad Campus

Department of Computer Science

CSC392 Object-Oriented Software Engineering

**Project Proposal**

**HOSTEL MANAGEMENT SYSTEM**

|  |
| --- |
| **Group Members** |
| FA21-BSE-015  Hanzla Nouman |
| FA21-BSE-019  Laiba Binta Tahir |
| FA21-BSE-027  Muhammad Irfan |
| FA21-BSE-080  Arfah Ali |

…

Table of Contents

[CHAPTER 1 PROJECT PROPOSAL 7](#_Toc139729743)

[Introduction 7](#_Toc139729744)

[Vision and Business Case 7](#_Toc139729745)

[Use-Case Model 8](#_Toc139729746)

[Supplementary Specification 10](#_Toc139729747)

[Glossary 10](#_Toc139729748)

[Risk List & Risk Management Plan 11](#_Toc139729749)

[CHAPTER 2: USE CASES 12](#_Toc139729750)

[Use Case Diagram 12](#_Toc139729751)

[Use Cases Distribution 13](#_Toc139729752)

[Brief Level Use Cases 14](#_Toc139729753)

[Laiba binta tahir (FA21-BSE-019 – 4A) 14](#_Toc139729754)

[Use Case: Register. 14](#_Toc139729755)

[Use Case: login. 15](#_Toc139729756)

[Use Case: Add student . 15](#_Toc139729757)

[Use Case: view student. 15](#_Toc139729758)

[Use Case: delete student. 15](#_Toc139729759)

[Use Case: logout. 15](#_Toc139729760)

[Arfah Ali (FA21-BSE-080 – 4A) 16](#_Toc139729761)

[Use Case: View Incidents. 16](#_Toc139729762)

[Use Case: Add Incidents. 16](#_Toc139729763)

[Use Case: Feedback From the Student. 16](#_Toc139729764)

[Use Case: add mess menu. 16](#_Toc139729765)

[Use Case: view mess menu. 16](#_Toc139729766)

[Hanzla Nouman (FA21-BSE-015-4A) 16](#_Toc139729767)

[Use Case: Allocate Room 17](#_Toc139729768)

[Use Case: View Rooms 17](#_Toc139729769)

[Use Case: Terminate Allocation 17](#_Toc139729770)

[Use Case: Add New Room 17](#_Toc139729771)

[Use Case: Delete Room 17](#_Toc139729772)

[Use Case: Assign Facilities 18](#_Toc139729773)

[Use Case: Remove Facility 18](#_Toc139729774)

[Use Case: Generate Fee 18](#_Toc139729775)

[Muhammad Irfan (FA21-BSE-027-4A) 18](#_Toc139729776)

[Use Case: Add Warden 18](#_Toc139729777)

[Use Case: Delete Warden 19](#_Toc139729778)

[Use Case: View Wardens 19](#_Toc139729779)

[Use Case: Add Security 19](#_Toc139729780)

[Use Case: View Security 19](#_Toc139729781)

[Use Case: Delete Security 19](#_Toc139729782)

[Use Case: Add Cook 19](#_Toc139729783)

[Use Case: View Cooks 20](#_Toc139729784)

[Use Case: Delete Cooks 20](#_Toc139729785)

[Fully Dressed Use Cases 21](#_Toc139729786)

[Laiba binta tahir (FA21-BSE-019 – 4A) 21](#_Toc139729787)

[Use Case: Register. 21](#_Toc139729788)

[Use Case: Login. 22](#_Toc139729789)

[Use Case: Add student. 24](#_Toc139729790)

[Use Case: view Student. 25](#_Toc139729791)

[Use Case: Delete Student. 27](#_Toc139729792)

[Use Case: logout. 29](#_Toc139729793)

[Arfah Ali (FA21-BSE-080 – 4A) 30](#_Toc139729794)

[Use Case UC1: View Incidents. 30](#_Toc139729795)

[Use Case UC2: Add Incidents. 33](#_Toc139729796)

[Use Case UC3: Feedback From Student. 34](#_Toc139729797)

[Use Case4: add mess menu. 35](#_Toc139729798)

[Use Case5: view mess menu. 36](#_Toc139729799)

[Hanzla Nouman (FA21-BSE-015) 37](#_Toc139729800)

[Use Case UC1: Allocate Room 37](#_Toc139729801)

[Use Case UC2: View Rooms 39](#_Toc139729802)

[Use Case UC3: Terminate Allocation 40](#_Toc139729803)

[Use Case UC4: Add New Rooms 41](#_Toc139729804)

[Use Case UC5: Delete Room 42](#_Toc139729805)

[Use Case UC6: Assign Facility 43](#_Toc139729806)

[Use Case UC7: Remove Facility 44](#_Toc139729807)

[Use Case UC8: Generate Fee 46](#_Toc139729808)

[Muhammad Irfan (FA21-BSE-027-4A) 47](#_Toc139729809)

[Use Case UC1: Add Warden 47](#_Toc139729810)

[Use Case UC2: Delete Warden 49](#_Toc139729811)

[Use Case UC3: View Wardens 50](#_Toc139729812)

[Use Case UC4: Add security guard. 52](#_Toc139729813)

[Use Case UC5: View Security guard 53](#_Toc139729814)

[Use Case UC6: Delete Security guard 54](#_Toc139729815)

[Use Case UC7: Add Cook 55](#_Toc139729816)

[Use Case UC8: View Cook 56](#_Toc139729817)

[Use Case UC9: Delete Cook 57](#_Toc139729818)

[PROTOTYPES 59](#_Toc139729819)

[CHAPTER 3: DOMAIN MODEL 63](#_Toc139729820)

[Domain Model 63](#_Toc139729821)

[CHAPTER 4: SYSTEM SEQUENCE DIAGRAM (SSD) 64](#_Toc139729822)

[**Laiba binta tahir (FA21-BSE-019-4A)** 64](#_Toc139729823)

[Scenario: Register 64](#_Toc139729824)

[Scenario: login 65](#_Toc139729825)

[Scenario: add student 66](#_Toc139729826)

[Scenario: view student 67](#_Toc139729827)

[Scenario: delete student 68](#_Toc139729828)

[Scenario: logout 69](#_Toc139729829)

[**Arfah Ali (FA21-BSE-080-4A)** 70](#_Toc139729830)

[Scenario: View Incidents 70](#_Toc139729831)

[Scenario: Add incidents 71](#_Toc139729832)

[Scenario: Feedback from students 72](#_Toc139729833)

[Scenario: add menu 73](#_Toc139729834)

[Scenario: view menu 74](#_Toc139729835)

[**Hanzla Nouman (FA21-BSE-015-4A)** 75](#_Toc139729836)

[Scenario: Allocate Room 75](#_Toc139729837)

[Scenario: View Rooms 76](#_Toc139729838)

[Scenario: Terminate Allocation 77](#_Toc139729839)

[Scenario: Add New Room 78](#_Toc139729840)

[Scenario: Delete Room 79](#_Toc139729841)

[Scenario: Assign Facility 80](#_Toc139729842)

[Scenario: Remove Facility 81](#_Toc139729843)

[Scenario: Generate Fee 82](#_Toc139729844)

[**Muhammad Irfan (FA21-BSE-027-4A)** 83](#_Toc139729845)

[Scenario: Add Warden:- 83](#_Toc139729846)

[Scenario: View Warden:- 84](#_Toc139729847)

[Scenario: Add Security Guard:- 85](#_Toc139729848)

[Scenario: View Security Guard:- 86](#_Toc139729849)

[Scenario: Delete Security Guard:- 87](#_Toc139729850)

[Scenario: Add Cook:- 88](#_Toc139729851)

[Scenario: View Cook:- 89](#_Toc139729852)

[Scenario: Delete Cook:- 90](#_Toc139729853)

[CHAPTER 5: OPERATION CONTRACT 91](#_Toc139729854)

[Laiba Binta Tahir(FA21-BSE-019-4A) 91](#_Toc139729855)

[Arfah Ali (FA21-BSE-080-4A) 94](#_Toc139729856)

[Hanzla Nouman (FA21-BSE-015-4A) 100](#_Toc139729857)

[Muhammad Irfan (FA21-BSE-027-4A) 102](#_Toc139729858)

[CHAPTER 6: SOFTWARE ARCHITECTURE 109](#_Toc139729859)

[Package diagram 109](#_Toc139729860)

[CHAPTER 7: INTERACTION DIAGRAM 110](#_Toc139729861)

[Communication diagram 110](#_Toc139729862)

[Laiba binta tahir (FA21-BSE-019-4A) 110](#_Toc139729863)

[Arfah Ali (FA21-BSE-080) 111](#_Toc139729864)

[Muhammad Irfan (FA21-BSE-027) 112](#_Toc139729865)

[Sequence diagram 113](#_Toc139729866)

[Laiba binta tahir (FA21-BSE-019-4A) 113](#_Toc139729867)

[Arfah Ali (FA21-BSE-080) 114](#_Toc139729868)

[Hanzla Nouman(FA21-BSE-015) 115](#_Toc139729869)

[Muahmmad Irfan(FA21-BSE-027) 116](#_Toc139729870)

[CHAPTER 8: CLASS DIAGRAM 117](#_Toc139729871)

# CHAPTER 1 PROJECT PROPOSAL

## Introduction

1.1 PROJECT OVERVIEW

The hostel management system is web-based software to provide university students accommodation to the university hostel more efficiently. This project also keeps details of the hostellers and applied students. It is headed by multiple administrators . This document is intended to minimize human works and make hostel allocation an easier job for students and hostel authorities by providing online application for hostel, automatically select the students from the waiting list and mess calculation, complaint registration, etc. Students will get approval notification in their mails. Hostellers can view hostel fee and mess menu by login into the online system.

1.2 PROBLEM STATEMENT

There are a lot of drawbacks in keeping and maintaining a hostel. Especially with a manual system. Since most hostels are being run by only one hostel manager, the number of students in a room are sometimes not known by the warden.

This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented.

We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

1.3 OBJECTIVES

• To make it easier for data collection, storage and referencing reliable.

• To maintain the students as hostellers and waiting list students separately.

• To process allotment list.

## Vision and Business Case

***Vision:***

Our hostel management system aims to provide a streamlined and efficient solution for managing hostel operations, improving the student experience, and increasing overall productivity. By leveraging modern technologies, we seek to transform hostel management into an automated and user-friendly process that benefits both students and administrators alike.

***Business Case:***

Hostel management is a complex and time-consuming process that requires significant resources and effort to manage effectively. Traditional methods of managing hostels often involve manual processes such as paper-based record keeping, which can be error-prone, time-consuming, and difficult to manage.

Our hostel management system seeks to address these challenges by providing a comprehensive solution that automates many of the tasks involved in hostel management. This includes features such as room allocation, meal plans, student profiles, visitor management, and maintenance tracking. By automating these tasks, we aim to save time and increase the efficiency of hostel operations.

Additionally, our system includes features that enhance the student experience, such as the ability to view their room details, meal plans, and other relevant information through a user-friendly interface. This not only improves the overall student experience but also reduces the workload of administrators by reducing the number of inquiries and requests they receive from students.

Overall, our hostel management system aims to provide a cost-effective and efficient solution for managing hostel operations while also improving the student experience. By leveraging modern technologies and automation, we believe that our system will help hostels to run more smoothly, improve productivity, and ultimately increase profitability.

## Use-Case Model

Functional Requirements for Hostel Management System

* Allocaton /Booking
* record reservations
* record the student first name
* record the student’s last name
* record the number of students
* record the room number
* display the default room fee display whether or not the room is guaranteed.
* generate a unique confirmation number for each.
* record the expected check-in date and time
* The system shall record the expected checkout date and time
* The system shall record customer feedback

2.  Food

* The system shall track all meals purchased in the hotel .
* The system shall record payment and payment type for meals
* The system shall bill the current room if payment is not made at time of service
* The system shall accept reservations for services.

3. Management

* display the hotel occupancy for a specified period of time (days; including past, present, and future dates).
* display projected occupancy for a period of time (days).
* display room revenue for a specified period of time (days).
* display food revenue for a specified period of time (days).
* display an exception report, showing where default room and food prices have been overridden
* allow for the addition of information, regarding rooms, rates, menu items, prices, and user profiles
* allow for the deletion of information, regarding rooms, rates, menu items, prices, and user profiles
* allow for the modification of information, regarding rooms, rates, menu items, prices, and user profiles
* allow managers to assign user passwords

## Supplementary Specification

There are a lot of software requirements specifications included in the non-functional requirements of the Hostel Management System, which contains various processes, namely Security, Performance, Maintainability, and Reliability.

**Security:**

● user Identification: The system needs the user to recognize herself or himself using the phone.

● Logon ID: Any users who make use of the system need to hold a Logon ID and password.

● Modifications: Any modifications like insert, delete, update, etc. for the database can be synchronized quickly and executed only by the ward administrator.

● Front Desk Staff Rights: The staff at the front desk can view any data in the Hostel Management system.

● Administrator rights: The administrator can view as well as alter any information in the Hostel Management System.

**Performance:**

● Response Time: The system provides acknowledgment in just one second once the 'user's information is checked.

● Capacity: The system needs to support at least 1000 people at once.

● User-Interface: The user interface acknowledges within five seconds.

● Conformity: The system needs to ensure that the guidelines of the Microsoft accessibilities are followed.

**Maintainability:**

● Back-Up: The system offers efficiency for data backup.

● Errors: The system will track every mistake as well as keep a log of it.

**Reliability:**

● Availability: The system is available all the time.

## Glossary

|  |  |
| --- | --- |
| *HMS* | Hostel management system |
| *DFD* | Data flow diagram |
| *UC* | Use case diagram |
|  |  |

## 

## Risk List & Risk Management Plan

**Risk List:**

***Technical Risks*** *-* These risks include system failures, software bugs, and other technical issues that may disrupt the system's operations or cause data loss.

***Security Risks*** *-* These risks include data breaches, unauthorized access to the system, and other security threats that may compromise the system's integrity and the safety of students' information.

***Human Risks*** *-* These risks include human error, misconduct, and other issues that may arise from the actions of administrators or other users of the system.

***Operational Risks*** *-* These risks include issues related to the management of the hostel, such as room allocation, billing and payments, maintenance tracking, and other operations that may affect the system's performance.

**Risk Management Plan:**

***Technical Risks*** *-* To mitigate technical risks, the system will be regularly tested for bugs and other technical issues. A backup system will also be implemented to ensure that data is not lost in case of a system failure.

***Security Risks*** *-* To mitigate security risks, the system will be protected by advanced security protocols such as encryption, firewalls, and regular vulnerability testing. Access to the system will be restricted to authorized users only, and data will be regularly backed up and secured.

***Human Risks*** *-* To mitigate human risks, the system will be designed to limit the potential for user error, such as providing clear and intuitive user interfaces and requiring authorization for critical functions. Administrators will also be required to undergo training and follow strict protocols to ensure the safety and integrity of student data.

***Operational Risks*** *-* To mitigate operational risks, the system will be designed to automate and streamline hostel operations, such as room allocation and maintenance tracking. The system will also provide detailed analytics and reports to help administrators identify and address operational issues before they become major problems.

Overall, by implementing a comprehensive risk management plan, the hostel management system can minimize the potential for risk and ensure the smooth and efficient operation of the system.

# CHAPTER 2: USE CASES

## Use Case Diagram



## Use Cases Distribution

|  |  |  |
| --- | --- | --- |
| S#. | Group Member | Assigned Use Cases |
| 1 | FA21-BSE-015  Hanzla Nouman | Allocate rooms:   * Add room * View room * Generate fee * Terminate allocation * Assign facilities to students * Booked room |
| 2 | FA21-BSE-019  Laiba Binta Tahir | Register  Login  Logout  Menue  Manage students   * Add student * View student * Delete student * modify student |
| 3 | FA21-BSE-027  Muhammad Irfan | Manage warden.   * Add warden. * View warden. * Delete warden. * modify warden.   Manage staff   * Add security * View security * Delete security * Add cook * View cook * Delete cook |
| 4 | FA21-BSE-080  Arfah Ali | Delete Incident  Add incidents  View incidents.  Feedback from students  Add Menu  View Menu |

## Brief Level Use Cases

### Laiba binta tahir (FA21-BSE-019 – 4A)

Register

Login

* Add menu.
* View menu.

Manage students.

* Add student.
* View student
* Delete student.

Logout

#### Use Case: Register.

The user selects the register option in the hostel management system. The user provides their personal details and student ID for registration. The system validates the entered information and creates a new account for the student. Upon successful registration, the system displays a confirmation message and redirects the user to the login page.

#### Use Case: login.

The user selects the login option in the hostel management system. The user enters their username and password. The system verifies the user's credentials. If the user's credentials are valid, the system logs the user into the system and displays the appropriate dashboard based on their role (student or administrator). If the user's credentials are invalid, the system displays an error message and prompts the user to re-enter their credentials.

#### Use Case: Add student .

The user selects the "Add Student" option from the main menu. The system prompts the user to enter the student's personal information, including name, contact details, and identification information. The user enters the student's information into the system. The system validates the information and confirms that it is complete and accurate. The user selects the student's preferred room type and bed preference. The system checks the availability of the preferred room and allocates a room to the student if available. The user enters the student's fee details and confirms the payment. The system confirms the payment and updates the student's record accordingly. The system displays a confirmation message to the user.

#### Use Case: view student.

The user selects the "View Student" option from the main menu. The system prompts the user to enter the student's ID or name. The user enters the student's ID or name and submits the query. The system retrieves the student's details from the database and displays them in a clear and easy-to-understand format. The user can view the details of the selected student(s) and perform further actions, such as updating their records, contacting them, or generating reports.

#### Use Case: delete student.

The admin logs into the hostel management system and selects the "Delete *student*" option from the main menu. The system retrieves the list of all registered *students* and displays their details, such as name, contact information, on the screen. The admin selects the *student* to delete from the list and confirms the action. The system removes the selected *student* from the system, revokes their access to the system, and updates the records accordingly. The system then displays a confirmation message to the admin.

#### Use Case: logout.

The user selects the "Logout" option from the main menu. The system prompts the user to confirm that they want to log out. The user confirms the action by selecting "Yes." The system terminates the user's session and logs them out of the system. The system displays a confirmation message to the user.

### Arfah Ali (FA21-BSE-080 – 4A)

* View facilities
* Apply for hostel
* Add incidents
* Submit fee(hostel/mess)
* View incidents and reports.
* Feedback from students

#### Use Case: View Incidents.

The user navigates to the "Incidents" section of the Hostel Management System. The system displays a list of all incidents reported in the hostel, sorted by date and time. The user can filter the incidents by date range, location, or severity. The user can view the details of each incident, including the date and time of the incident, location, description, severity level, and status. The user can update the status of the incident if necessary.

#### Use Case: Add Incidents.

The Student navigates to the “Incidents” section of the system. The system displays a form for the Student to fill in the details of the incident. The Student fills in the details of the incident including the ID, day, date, time, location, description. The Student submits the incident report through the system. The system stores the incident report in the database.

#### Use Case: Feedback From the Student.

The Student navigates to the “Feedback” section of the system. The system displays a form for the Student to fill in their feedback. The Student fills in the feedback form with their comments, suggestions or complaints. The Student submits the feedback through the system. The system stores the feedback in the database for review and improvement by the appropriate staff member. The Student receives a confirmation of the successful submission.

#### Use Case: add mess menu.

The user selects the “Menu" option from the main menu. The system displays a table of Mess Menu and Buttons. The user selects Add Menu. The form prompts the user to add new items to the menu for each meal (breakfast, lunch, and dinner) for the desired. The user enters the data and click on Submit. System validates the fields and Adds the food stock to the database or list.

#### Use Case: view mess menu.

The user selects the "View Mess Menu" option from the main menu. The system displays the mess menu for the current date and time. The user can view the menu and make decisions about their meals.

### Hanzla Nouman (FA21-BSE-015-4A)

* Allocate Room
  + Allocate the Room to the Student
  + Terminate Allocation
* Add Room
* Delete Room
* View Rooms (Room no, Building, Location etc.)
* Assign Facilities
* Remove Facility

#### Use Case: Allocate Room

The warden logs into the hostel management system and selects the "Allocate Room" option from the main menu. The system displays the list of available rooms, and the warden selects an available room based on the student's preferences and needs. The system allocates the selected room to the student, updates the availability status of the room in the system, and generates a fee receipt for the allocated room. The warden prints the fee receipt and hands it over to the student.

#### Use Case: View Rooms

The warden logs into the hostel management system and selects the "View All Rooms" option from the Rooms Menu. The system displays a list of all the rooms, including allocated and unavailable rooms, along with their room number, room type, and availability status. The warden can scroll through the list and view the details of each room.

#### Use Case: Terminate Allocation

The warden receives a request from a student to terminate their room allocation due to personal reasons. The warden logs into the hostel management system and selects the "Terminate Room Allocation" option from the Rooms Menu. The system displays a list of all allocated rooms and the details of the students occupying them. The warden selects the room allocated to the student and confirms the action. The system terminates the room allocation for the student, updates the availability status of the room in the system. The system then sends a notification to the student, informing them of the termination of their room allocation.

#### Use Case: Add New Room

The warden logs into the hostel management system and selects the "Add Room" option from the Rooms Menu. The system prompts the warden to enter the room details such as room number, room type, and availability status. The warden enters the details and submits the form. The system adds the new room to the list of available rooms and displays a confirmation message to the warden.

#### Use Case: Delete Room

The warden logs into the hostel management system and selects the "Delete Room" option from the Rooms Menu. The system displays a list of available rooms and prompts the warden to select a room to delete. The warden selects a room to delete and confirms the action. The system removes the selected room from the list of available rooms and displays a confirmation message to the warden.

#### Use Case: Assign Facilities

The warden logs into the hostel management system and selects the "Assign Facilities" option from the Facilities Menu. The system prompts the warden to enter the student's details, such as their name and registration number. The warden enters the details and selects the facilities to assign to the student, such as a study table, chair, or cupboard. The system assigns the selected facilities to the student and displays a confirmation message to the warden.

#### Use Case: Remove Facility

The warden logs into the hostel management system and selects the "Remove Facility" option from the Facilities Menu. The system prompts the warden to enter the student's details, such as their name and registration number. The warden enters the details and selects the facility to remove for the student, such as a study table, chair, or cupboard. The system removes the selected facility from the student's record and updates the availability status of the facility in the system. The system then displays a confirmation message to the warden. The use case ends.

#### Use Case: Generate Fee

The warden logs into the hostel management system and selects the "Generate Fee Challan" option from the main menu. The system retrieves the list of all registered students and displays their details, such as name, roll number, and hostel fees, on the screen. The warden selects the student for whom the challan needs to be generated and confirms the action. The system generates the fee challan for the selected student, which includes the amount to be paid, the due date, and other details. The system then updates the student's tab with the fee challan details. The system displays a confirmation message to the warden.

### Muhammad Irfan (FA21-BSE-027-4A)

* Manage Wardens
  + Add warden.
  + View wardens
  + Delete warden.
* Manage staff
  + Add security.
  + View security.
  + Delete security
  + Add cook
  + View cook
  + Delete cook

#### Use Case: Add Warden

The admin logs into the hostel management system and selects the "Add Warden" option from the main menu. The system prompts the admin to enter the details of the new warden, such as their name, contact information, and job title. The admin enters the details and confirms the addition of the new warden. The system creates a new user account for the warden with login credentials and grants appropriate access to the system. The admin receives a confirmation message from the system.

#### Use Case: Delete Warden

Delete Warden: The admin logs into the hostel management system and selects the "Delete Warden" option from the main menu. The system retrieves the list of all registered wardens and displays their details, such as name, contact information, and job title, on the screen. The admin selects the warden to delete from the list and confirms the action. The system removes the selected warden from the system, revokes their access to the system, and updates the records accordingly. The system then displays a confirmation message to the admin.

#### Use Case: View Wardens

The admin logs into the hostel management system and selects the "View Wardens" option from the main menu. The system retrieves the list of all registered wardens and displays their details, such as name, contact information, and job title, on the screen. The admin can sort the list based on specific criteria, such as job title or date of registration, and can also filter the list based on certain parameters, such as active or inactive status. The admin can view the details of any specific warden by clicking on their name in the *list.*

#### Use Case: Add Security

Add Security Personnel: The admin logs into the hostel management system and selects the "Add Security Personnel" option from the main menu. The system prompts the admin to enter the details of the new security personnel, such as their name, contact information, and job title. The admin enters the details and confirms the addition of the new security personnel. The system creates a new user account for the security personnel with login credentials and grants appropriate access to the system. The admin receives a confirmation message from the system.

#### Use Case: View Security

The admin logs into the hostel management system and selects the "View Security Personnel" option from the main menu. The system retrieves the list of all registered security personnel and displays their details, such as name, contact information, and job title, on the screen. The admin can sort the list based on specific criteria, such as job title or date of registration, and can also filter the list based on certain parameters, such as active or inactive status. The admin can view the details of any specific security personnel by clicking on their name in the list.

#### Use Case: Delete Security

The admin logs into the hostel management system and selects the "Delete Security Personnel" option from the main menu. The system retrieves the list of all registered security personnel and displays their details, such as name, contact information, and job title, on the screen. The admin selects the security personnel to delete from the list and confirms the action. The system removes the selected security personnel from the system, revokes their access to the system, and updates the records accordingly. The system then displays a confirmation message to the admin.

#### Use Case: Add Cook

The admin logs into the hostel management system and selects the "Add Cook" option from the main menu. The system prompts the admin to enter the details of the new cook, such as their name, contact information, and job title. The admin enters the details and confirms the addition of the new cook. The system creates a new user account for the cook with login credentials and grants appropriate access to the system. The admin receives a confirmation message from the system.

#### Use Case: View Cooks

The admin logs into the hostel management system and selects the "View Cooks" option from the main menu. The system retrieves the list of all registered cooks and displays their details, such as name, contact information, and job title, on the screen. The admin can sort the list based on specific criteria, such as job title or date of registration, and can also filter the list based on certain parameters, such as active or inactive status. The admin can view the details of any specific cook by clicking on their name in the list.

#### Use Case: Delete Cooks

The admin logs into the hostel management system and selects the "Delete Cook" option from the main menu. The system retrieves the list of all registered cooks and displays their details, such as name, contact information, and job title, on the screen. The admin selects the cook to delete from the list and confirms the action. The system removes the selected cook from the system, revokes their access to the system, and updates the records accordingly. The system then displays a confirmation message to the admin.

## Fully Dressed Use Cases

### Laiba binta tahir (FA21-BSE-019 – 4A)

#### 

#### Use Case: Register.

**Use Case Name:** Register

**Scope:** Hostel Management System

**Primary Actor:** Student

**Goal in Context:** The primary goal of this use case is to allow students to register for an account in the hostel management system to access its features and functionalities.

**Preconditions:**

The student has a device with internet access.

The student has the necessary information required for registration, such as their personal details and student ID.

**Trigger:**

The student selects the "Register" option on the login page of the hostel management system.

**Main Success Scenario:**

|  |  |  |
| --- | --- | --- |
| Step | User Interaction | System Response |
| 1 | Student selects the "Register" option. | The system displays the registration form. |
| 2 | Student enters their personal details and student ID into the registration form. | The system validates the entered information to ensure it meets the required criteria. |
| 3 | Student submits the registration form. | If the entered information is valid, the system creates a new account for the student and saves the details in database. |
| 4 |  | The system displays a success message indicating that the registration is successful. |
|  |  | The system redirects the student to the login page. |

**Extensions:**

If the entered information is not valid (e.g., invalid student ID or incomplete personal details), the system will display an error message and prompt the student to correct the information.

**Alternate Flow:**

If the student already has an account with the hostel management system, they should select the "Login" option instead of the "Register" option.

**Postconditions:**

The student's registration is successfully completed, and they can now log in to the hostel management system using their newly created account.

The system logs the registration activity for future reference and auditing purposes.

**Exceptions:**

If the system encounters technical issues during the registration process, the student may not be able to register. In this case, the student can contact technical support or try again later.

#### Use Case: Login.

**Use Case Name:** Login

**Scope:** Hostel Management System

**Primary Actor:** Administrator or Student

**Goal in Context**: The primary goal of this use case is to allow administrators or students to log into the hostel management system to access their respective features and functionalities.

**Preconditions:**

The user has a valid account with the hostel management system.

The user has a device with internet access.

The user has their login credentials, including their username and password.

**Trigger:**

The user attempts to log into the hostel management system by entering their login credentials.

**Main Success Scenario:**

|  |  |
| --- | --- |
| **User Interaction** | **System Response** |
| 1. User enters their login credentials (username and password). | 1. System validates the user's credentials by checking them against the credentials stored in the database. |
| 2. User clicks the login button. | 2. If the credentials are valid, the system logs the user into the hostel management system and displays the main menu. |
|  | 3. If the credentials are not valid, the system displays an error message indicating that the login attempt has failed. |

**Extensions:**

If the user enters an incorrect username or password, the system will display an error message and prompt the user to re-enter their credentials.

If the user forgets their password, they can request a password reset link or contact the administrator for assistance.

**Alternate Flow:**

If the user does not have a valid account with the hostel management system, they can create an account by clicking the "register" button on the login page.

**Postconditions:**

The user is successfully logged into the hostel management system and has access to their respective features and functionalities.

The system logs the user's login activity for security and audit purposes.

**Exceptions:**

If the system experiences technical issues, the user may not be able to log in. In this case, the user can contact technical support or try again later.

#### Use Case: Add student.

**Use Case Name:** Add Student

**Scope:** Hostel Management System

**Level:** User Goal (Primary)

**Stakeholders and Interests:**

Hostel Staff: Wants to manage student records and information efficiently, including registration, room allocation, and fee payment.

Students: Want to register for accommodation in the hostel and pay the necessary fees.

**Preconditions:**

The user is authorized to access the hostel management system.

The student has expressed interest in registering for accommodation in the hostel.

**Postconditions:**

The student's information is added to the hostel management system.

The student can be allocated a room based on availability and preferences.

The student can pay the necessary fees for accommodation and other services.

**Basic Flow of Events:**

|  |  |
| --- | --- |
| **User Interaction** | **System Response** |
| 1. User selects the option to add a new student. | 1. System prompts the user to enter the student's information, such as name, contact information, room number, and any other relevant details. |
| 2. User enters the student's information. | 2. System validates the information provided by the user to ensure that it is complete and accurate. |
| 3. User confirms the addition of the new student. | 3. If the information provided by the user is valid, the system adds the new student's record to the database and displays a success message to the user. |
|  | 4. The success message includes the student's ID, which is assigned by the system, as well as their name and other relevant details. |
| 4. User reviews the new student's record. | 5. System waits for the user to finish reviewing the new student's record before returning to the main menu. |
|  | 6. If the user selects to add another student, the system repeats the above process. Otherwise, it returns to the main menu. |

**Alternative Flows:**

If the student has already paid the fees, the user can skip the fee payment step and proceed to allocate a room.

If there are no rooms available for the student's preferred room type and bed preference, the system will display alternative options for the user to choose from.

**Exceptional Flows:**

If there is an error in the system while adding the student's information, the system displays an error message to the user and prompts them to try again.

If there is a problem with the payment, the system will display an error message and prompt the user to enter the correct details.

#### Use Case: view Student.

**Use Case Name**: View Student

**Scope**: Hostel Management System

**Level**: User Goal (Primary)

**Stakeholders and Interests:**

Hostel Staff: Want to view the details of a particular student, including their personal information, academic performance, and disciplinary record.

Parents/Guardians: Want to view the details of their child's stay in the hostel, including their room allocation, attendance, and progress report.

Students: Want to view their own details and stay updated on their academic and personal progress.

**Preconditions:**

The user is authorized to access the hostel management system.

The details of the student(s) are available in the system database.

**Postconditions:**

The user can view the details of the selected student(s) in a clear and easy-to-understand format.

The student(s) can access their own details and stay updated on their progress and stay in the hostel.

**Main Success Scenario:**

|  |  |
| --- | --- |
| **User Interaction** | **System Response** |
| 1. User selects the option to view a student. | 1. System displays a list of all the students registered in the hostel. |
| 2. User selects the student to be viewed. | 2. System retrieves the student's record from the database and displays the details to the user. |
|  | 3. System displays the student's ID, name, contact information, room number, and any other relevant details. |
| 3. User reviews the student's information. | 4. System waits for the user to finish reviewing the student's information before returning to the main menu. |
|  | 5. If the user selects to view another student, the system repeats the above process. Otherwise, it returns to the main menu. |

**Alternative Flows:**

If the user enters an invalid or non-existent ID or name, the system displays an error message and prompts the user to try again.

If the user is not sure about the student's ID or name, the system can provide suggestions based on the search query.

**Exceptional Flows:**

If there is an error in the system while retrieving or displaying the student's details, the system displays an error message to the user and prompts them to try again.

If the user does not have permission to view the student's details, the system displays an error message and denies access.

#### Use Case: Delete Student.

**Use Case Name:** Delete student.

**Primary Actor:** Administrator

**Goal in Context**: The primary goal of this use case is to allow the administrator to delete a student from the hostel management system and associated records.

**Preconditions**

The administrator is logged in to the hostel management system.

The administrator has the appropriate privileges to delete a student record.

The student record to be deleted exists in the system.

**Trigger:**

The administrator selects the option to delete a student record from the system.

**Main Success Scenario:**

|  |  |
| --- | --- |
| **User Interaction** | **System Response** |
| 1. User selects option to delete a student. | 1. System checks whether student ID exists. |
| 2. User enters ID of student to be deleted. | 2. If student ID exists, system displays confirmation message and asks for confirmation to delete. |
| 3. User confirms deletion. | 3. If confirmed, system removes student record from database. |
|  | 4. System displays success message to user indicating student has been successfully deleted. |
|  | 5. If student ID does not exist, system displays error message indicating that student could not be found and prompts user to try again. |

**Extensions:**

If the student record to be deleted has pending fees or dues, the system will prompt the administrator to settle the outstanding balance before proceeding with the deletion.

If the student record to be deleted is linked to other records, such as room allocations or meal plans, the system will prompt the administrator to reassign these records to another student or delete them along with the student record.

**Alternate Flow:**

If the administrator accidentally selects the wrong student record for deletion, they can cancel the deletion and select the correct record.

**Postconditions:**

The student record and associated records are successfully deleted from the hostel management system.

The system logs the administrator's action for audit purposes.

**Exceptions:**

If the administrator does not have the appropriate privileges to delete a student record, the system will display an error message and prevent the deletion from proceeding.

If the student record to be deleted does not exist in the system, the system will display an error message and prevent the deletion from proceeding.

#### Use Case: logout.

**Use Case Name:** logout.

**Scope:** Hostel Management System

**Level:** User Goal (Primary)

**Stakeholders and Interests:**

Hostel Staff: Wants to ensure the security and confidentiality of the data within the system and prevent unauthorized access to the system.

System Administrator: Wants to ensure that the system operates smoothly and efficiently.

**Preconditions:**

The user is currently logged in to the hostel management system.

The user has completed their tasks within the system and wants to log out.

The user is an authorized user of the hostel management system.

**Postconditions:**

The user is logged out of the hostel management system.

The user's session is terminated, and they can no longer access the system until they log in again.

The system is secure, and the data within the system is protected from unauthorized access.

**Main Success Scenario**

|  |  |
| --- | --- |
| 1. User selects "Logout" option from the main menu. | System displays a confirmation message to the user, asking them to confirm that they want to log out. |
| 2. User confirms that they want to log out. | System revokes the user's access to the system and logs them out of the system, returning to the login screen. |

**Alternative Flows:**

If the user selects "No" when prompted to confirm their logout, the system returns to the previous screen without logging the user out.

If the user is inactive for a certain period, the system may automatically log the user out to ensure the security of the system.

**Exceptional Flows:**

If there is an error in the system while the user is logging out, the system displays an error message to the user and prompts them to try again. If the issue persists, the user can contact the system administrator for assistance.

### Arfah Ali (FA21-BSE-080 – 4A)

#### Use Case UC1: View Incidents.

**Use Case Name:** View Incidents

**Scope:** Hostel Management System

**Primary Actor:** Hostel Staff

**Stakeholders and Interests:**

Hostel Staff: Wants to view a list of incidents reported in the hostel.

Hostel Manager: Wants to ensure that all incidents are recorded and managed properly.

Hostel Residents: Expect that incidents are handled effectively and efficiently.

**Preconditions:**

The user is logged into the Hostel Management System.

The user has the necessary permissions to view incidents.

Incidents have been reported and recorded in the system.

**Postconditions:**

The user has a list of all incidents reported in the hostel.

**Main Success Scenario:**

| **User Interaction** | **System Response** |
| --- | --- |
| 1. Warden selects "View Incidents" from the dashboard | 2. System displays a list of all incidents reported in the hostel. |
| 3. Warden selects a specific incident to view details. | 4. System displays the details of the selected incident, including the time, location, and description. |
| 5. Warden can search for specific incidents by date, location, or keyword. | 6. System filters the incident list based on the user's search criteria. |
| 7. If the incident is resolved, the system updates the status to "closed" and provides a summary of the resolution | 8. System sends a notification to the user with the resolution details |

**Extensions (or Alternative Flows):**

*A1. No incidents have been reported in the hostel.*

The system displays a message indicating that there are no incidents to view.

*A2. The user does not have permission to view incidents.*

The system displays an error message indicating that the user does not have the necessary permissions.

*A3. The user encounters an error while viewing incidents.*

The system displays an error message and prompts the user to try again.

*A4. The user wants to add a new incident.*

The user can navigate to the "Add Incident" section of the Hostel Management System to report a new incident.

*A5. The user wants to export incident data.*

The user can export the incident data to a CSV or Excel file for further analysis**.**

#### Use Case UC2: Add Incidents.

**Actor:** Student

**Preconditions:**

Students must be registered in the hostel management system.

The incident must have occurred in the hostel premises.

**Postconditions:**

The incident report is added to the hostel management system.

Main Success Scenario:

| **Student Interaction** | **System Response** |
| --- | --- |
| 3. Student selects the “Incident" option. | 4. The system displays the already reported incidents table. |
| 5. Student selects the “Add Incident" option. | 6. The system displays the form to fill the required fields. |
| 7. Student fills out the incident reporting form with the details of the incident. |  |
| 8. Student submits the incident report. | 9. System Validates the field. |
|  | 10. The system adds the incident report to the hostel management system and assigns it to the relevant staff member for resolution. |
|  | 11. Student receives a confirmation display that says “Incident Registered”. |

**Alternative Scenarios:**

A1. If Student or staff member is not registered in the hostel management system:

The system displays an error message and prompts the user to register.

A2. If the incident reporting form is not displayed:

The system displays an error message and prompts the user to contact the hostel management staff.

A3 If the incident report cannot be added to the hostel management system due to technical issues:

The system displays an error message and prompts the user to try again later.

#### Use Case UC3: Feedback From Student.

**Actor:** Studen

**Preconditions:**

Student must be registered in the hostel management system.

Student must have used the hostel facilities.

**Postconditions:**

The feedback is added to the hostel management system.

The feedback is reviewed by the relevant staff member for improvement.

**Main Success Scenario:**

| **Student Interaction** | **System Response** |
| --- | --- |
| 1. Student opens the hostel management system and logs in. | 2. The system displays the main menu of the hostel management system. |
| 3. Student selects the "Feedback" option. | 4. The system displays the feedback form. |
| 5. Student fills out the feedback form with their comments and suggestions. |  |
| 6. Student submits the feedback form. | 7. The system adds the feedback to the hostel management system and assigns it to the relevant staff member for improvement. |
| 8. Student receives a confirmation of the feedback submission. |  |

**Alternative Scenarios:**

A1. If Student is not registered in the hostel management system:

The system displays an error message and prompts the user to register.

A2. If the feedback form is not displayed:

The system displays an error message and prompts the user to contact the hostel management staff.

A3. If the feedback cannot be added to the hostel management system due to technical issues:

The system displays an error message and prompts the user to try again later.

#### Use Case4: add mess menu.

**Use Case Name:** Add Mess Menu

**Scope**: Hostel Management System

**Level**: User Goal (Primary)

**Stakeholders and Interests:**

Hostel Staff: Want to add new food items to the mess menu for the week/month.

Students: Want to have a variety of food options in the mess.

**Preconditions:**

The user is authorized to access the hostel management system.

The mess has been stocked with the necessary food items.

**Postconditions:**

The new food items are added to the mess menu for the selected Day.

The updated menu is displayed to the Table in a clear and easy-to-understand format.

**Main Success Scenario**

|  |  |
| --- | --- |
| **User Interaction** | **System Response** |
| 1. User selects the "Menu" option from the main menu | 2. System displays a table of Mess Menu and Buttons. |
| 3. System displays a table of Mess Menu and Buttons. | 4. Form prompts the user to add new items to the menu for each meal (breakfast, lunch, and dinner). |
| 5. User enters the data for each meal and clicks "Submit". | 6. System validates the fields and adds the food stock to the database or list. |
|  | 7. System displays a success message indicating that the menu items have been successfully added. |

**Alternative Flows:**

If a food item is not available, the user can add an alternative option.

If the user is not sure about the food items or quantities to add, the system can provide suggestions based on previous menus and student feedback.

**Exceptional Flows:**

If there is an error in the system while updating the menu, the system displays an error message to the user and prompts them to try again.

If there is a problem with the food stock or preparation, the system will notify the mess staff and suggest corrective action.

#### Use Case5: view mess menu.

**Use Case Name:** View Mess Menu

**Scope:** Hostel Management System

**Level:** User Goal (Primary)

**Stakeholders and Interests:**

Hostel Staff: Wants to create and update the mess menu, communicate changes to the students, and manage meal planning and inventory.

Students: Want to view the mess menu to plan their meals and make dietary decisions.

**Preconditions:**

The user is authorized to access the hostel management system.

The mess menu has been created and updated by the hostel staff.

The mess menu is accessible through the hostel management system.

**Postconditions:**

The user can view the mess menu for the specified date and time.

The user can make informed decisions about their meals based on the mess menu.

The hostel staff can manage the meal planning and inventory based on the mess menu.

**Main Success Scenario**

|  |  |
| --- | --- |
| 1. User selects "View Menu" option from the main menu of the hostel management system. | System retrieves the list of all available menus and displays them on the screen. |
| 2. User selects the menu to be viewed from the list. | System displays the details of the selected menu, including its name, description, and the items included in the menu. |
| 3. User reviews the details of the menu. | System allows the user to navigate back to the list of available menus or exit the menu viewing screen. |

**Alternative Flows:**

If the user wants to view the mess menu for a different date or time, they can select the desired date and time from the menu options.

If the user wants to see more details about a specific dish, they can click on the dish name to view the ingredients and preparation method.

### Hanzla Nouman (FA21-BSE-015)

#### Use Case UC1: Allocate Room

**Use Case Name:** Allocate Room to Student

**Scope:** Hostel Management System

**Level:** User Goal

**Primary Actor:** Warden

**Stakeholder and Interest:**

* Warden: To allocate the room to the student and generate a fee receipt for the room.
* Student: To get an allocated room and fee receipt for the allocated room.

**Precondition:**

* The warden is logged into the hostel management system.
* The student has completed the necessary application and registration processes.

Main Success Scenario:

|  |  |
| --- | --- |
| **User Interaction** | **System Response** |
| 1. The warden selects the "Allocate Room" option from the main menu. | 2. The system retrieves the list of all unallocated rooms and displays them on the screen. |
| 3. The warden selects a room from the list to allocate to the student. | 4. The system checks if the selected room is available and not already allocated to another student. |
|  | 5. The system prompts the warden to enter the student's name and ID number |
| 6. The warden enters the student's name and ID number into the system. |  |
|  | 7. The system checks if the entered student information matches the registered student information in the system. |
|  | 8. The system allocates the selected room to the student and updates the room status to "Allocated." |
|  |  |
|  | 9. The system generates a confirmation message with the allocated room number and displays it to the warden. |
| 10. The warden informs the student of their allocated room. |  |
|  |  |

**Extensions (Alternative Flows):**

* If there are no available rooms, the system displays a message to the warden indicating that there are no rooms available and prompts the warden to add more rooms to the system.
* If the student changes their mind about the room allocation, the warden can cancel the allocation and select a different room for the student.
* If there are any issues with the fee receipt, the warden can update the details and regenerate the fee receipt.

**Special Requirements:**

* The system should have a list of available rooms.
* The system should have a fee calculation system.

**Technology and Data Variation List:**

* Hostel Management System
* Available Rooms List
* Fee Calculation System

**Open Issues:**

* None.

#### Use Case UC2: View Rooms

**Use Case Name:** View Rooms

**Scope:** Hostel Management System

**Level:** User Goal

**Primary Actor:** Warden

**Stakeholders and Interests:**

* **Warden:** Wants to view all available rooms in the hostel.
* **Students:** Want to know the available rooms in the hostel.

**Preconditions:**

* The Warden must be logged into the Hostel Management System.

**Main Success Scenario:**

| **Actor Action** | **System Response** |
| --- | --- |
| 1. The Warden selects the "View Rooms" option from the main menu. | 2. The system retrieves the list of all available rooms in the hostel. |
|  | 3. The system displays the list of rooms along with their current occupancy status, room type, and room number. |
| 4. The Warden can view the list of all available rooms in the hostel. |  |

**Extensions:**

3a. If no rooms are available in the hostel, the system displays a message stating that all rooms are currently occupied.

| **Actor Action** | **System Response** |
| --- | --- |
| 1. The Warden selects the "View Rooms" option from the main menu. | 2. The system checks the availability of rooms in the hostel. |
|  | 3. The system displays a message stating that all rooms are currently occupied. |
| 4. The Warden cannot view the list of available rooms in the hostel. |  |

**Special Requirements:**

* The system must have access to the database of rooms and their occupancy status.

**Technology and Data Variation List:**

* The Hostel Management System should be accessible via a computer or mobile device.
* The system should display the list of available rooms in a clear and organized manner.

**Open Issues:**

* None.

#### Use Case UC3: Terminate Allocation

Use Case: Terminate Allocation

Scope: Hostel Management System

Level: User Goal

Primary Actor: Warden

Stakeholders:

* Students: Who requests for room termination.
* Administration: Who manages the hostel and need to keep track of room allocation status.

Precondition: Warden is logged into the hostel management system.

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| Warden selects "Terminate Room Allocation" option from Rooms menu. | System displays a list of all allocated rooms and the details of the students occupying them. |
| Warden selects the room allocated to the student. | System confirms the action and terminates the room allocation for the student. |
|  | System updates the availability status of the room in the system. |
|  | System sends a notification to the student, informing them of the termination of their room allocation. |

Extensions:

* If the selected room is not allocated to any student, the system displays an error message and prompts the warden to select another room.
* If the warden cancels the termination request, the system returns to the main menu.

Special Requirements:

* The system must have up-to-date records of all allocated rooms and their occupancy status.
* The system must be able to send notifications to students regarding termination of their room allocation.

Technology and Data Variation List:

* Hostel Management System: The system must be able to handle data related to room allocation, student records, and notifications.
* Notification System: The system must have a notification system in place to send messages to students.

Open Issues:

* The system must have a mechanism to handle cases where the student disagrees with the termination of their room allocation.
* The system must be able to handle cases where the student has outstanding fees or dues before terminating their room allocation.

#### Use Case UC4: Add New Rooms

**Scope**: Hostel Management System

**Level**: User Goal

**Primary** **Actor**: Warden

**Stakeholders:**

* Students: To have access to available rooms and select a suitable one.
* Administration: To maintain records of available rooms and their status.

**Precondition:** The warden must be logged into the hostel management system.

**Main Success Scenario:**

| **User Interaction** | **System Response** |
| --- | --- |
| The warden selects the "Add Room" option in the Rooms Tab | The system prompts the warden to enter the room details such as room number, room type, availability status, room fee, room location |
| The warden enters the details and submits the form | The system adds the new room to the list of available rooms and displays a confirmation message to the warden |

**Extensions (Alternative Flows):**

* If the room already exists in the system, the system will display an error message and prompt the warden to enter a unique room number.
* If the room details are invalid or incomplete, the system will display an error message and prompt the warden to enter the correct details.

**Special Requirements:** The system must have the ability to store and update records of available rooms.

**Technology and Data Variation List:**

* The system must be compatible with web browsers and operating systems.
* The data entered by the warden must be validated to ensure accuracy and completeness.

**Open Issues:** None

#### Use Case UC5: Delete Room

Scope: Hostel management system

Level: User goal

Primary actor: Warden

Stakeholders:

* Students: Who will be affected by the availability of rooms
* Warden: Who manages the rooms
* Hostel administration: Who oversee the hostel management

Precondition: The warden must be logged in to the hostel management system

Main Success Scenario:

| **User Action** | **System Response** |
| --- | --- |
| Warden selects "Delete Room" from the Rooms Menu | System displays a list of available rooms |
| Warden selects a room to delete | System prompts the warden to confirm the action |
| Warden confirms the action | System removes the selected room from the list of available rooms |
|  | System displays a confirmation message to the warden |

Extensions (Alternative flows):

* If the warden cancels the action, the system returns to the list of available rooms without making any changes.
* If the selected room is already allocated to a student, the system displays an error message, and the room cannot be deleted.

Special Requirements: None

Technology and Data Variation List:

* The hostel management system must be accessible through a web browser or mobile application.
* The list of available rooms must be stored in a database and updated in real-time.

Open issues: None

#### Use Case UC6: Assign Facility

Scope: Hostel management system

Level: User-goal level

Primary Actor: Warden

Stakeholders:

* Students: who will be assigned facilities.
* Maintenance Staff: who will maintain and manage the assigned facilities. Preconditions:
* The warden must be logged into the hostel management system.
* The facilities to be assigned must be available.

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| The warden selects the "Assign Facilities" option from the Facilities Menu. | The system prompts the warden to enter the student's details, such as their name and registration number. |
| The warden enters the student's details and selects the facilities to assign to the student. | The system assigns the selected facilities to the student. |
|  | The system displays a confirmation message to the warden. |

Extensions (Alternative Flows):

* If the selected facilities are not available, the system displays an error message to the warden and the use case terminates.

Special Requirements:

* The system should keep track of the availability status of each facility.
* The system should have the ability to generate reports on the assigned facilities.

Technology and Data Variation List:

* The system should be compatible with different browsers and operating systems.
* The system should support different types of facilities and their respective details.

Open Issues:

* How to handle the assignment of facilities to multiple students at once.
* How to handle the maintenance and repair of the assigned facilities.

#### Use Case UC7: Remove Facility

Scope: Hostel Management System

Level: User goal

Primary Actor: Warden

Stakeholders:

* Students: Their facilities records are updated after a facility is removed.
* Hostel management staff: The system updates the availability status of the facility in the system after it is removed.

Precondition: The warden must be logged into the hostel management system and have the necessary permissions to remove facilities.

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| The warden logs into the hostel management system and selects the "Remove Facility" option from the Facilities Menu. | The system displays a list of facilities assigned to students. |
| The warden enters the student's details, such as their name and registration number. | The system verifies the student's details and displays the list of facilities assigned to the student. |
| The warden selects the facility to remove for the student, such as a study table, chair, or cupboard. | The system removes the selected facility from the student's record and updates the availability status of the facility in the system. |
|  | The system displays a confirmation message to the warden. |

Extensions:

3a. The warden selects an invalid facility to remove.

* The system displays an error message and prompts the warden to select a valid facility.

3b. The warden enters an invalid student detail.

* The system displays an error message and prompts the warden to enter a valid detail.

Special Requirements:

* The system must keep a record of all the facilities assigned to each student.
* The system must update the availability status of the facility in the system after it is removed.
* The system must be able to handle multiple requests to remove facilities simultaneously.

Technology and Data Variation List:

* Hostel Management System: The use case should work on any platform running the hostel management system.
* Data: The system should handle variations in student data such as name and registration number.

Open Issues:

* How should the system handle a case where a facility that has been removed needs to be added back?
* What happens if the facility that the warden wants to remove is currently in use by the student?
* How should the system handle case where multiple wardens want to remove the same facility at the same time?

#### Use Case UC8: Generate Fee

Scope: Hostel Management System

Level: User Goal

Primary Actor: Warden

Stakeholders:

* Admin: sets up the system and provides access to wardens
* Students: pay the hostel fee

Preconditions:

* The warden has logged into the hostel management system
* The list of registered students and their hostel fees are available in the system

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| Warden selects "Generate Fee Challan" option | System displays list of registered students with their details, such as name and roll number |
| Warden selects the student for the challan | System generates the fee challan for the selected student, which includes the due amount, due date, and other details |
| Warden confirms the action | System updates the student's tab with the fee challan details |
|  | System displays a confirmation message to the warden |

Extensions (Alternative Flows): 2a. If the system cannot retrieve the list of registered students:

| **User Interaction** | **System Response** |
| --- | --- |
| Warden selects "Generate Fee Challan" option | System displays an error message that no students are registered |

2b. If the warden cannot find the desired student:

| **User Interaction** | **System Response** |
| --- | --- |
| Warden selects the student for the challan | System displays an error message that the student could not be found |

3a. If the warden cancels the action:

| **User Interaction** | **System Response** |
| --- | --- |
| Warden cancels the action | System displays a message that the action has been cancelled |

Special Requirements:

* The system should calculate the hostel fees for each student based on predefined rates and room types
* The system should allow for partial payment of fees and should update the student's tab accordingly

Technology and Data Variation List:

* The system should be accessible via a web browser
* The system should be able to handle a large number of registered students and their hostel fees
* The system should be able to store and retrieve data on a secure server

Open Issues:

* How should the system handle late fee payments and penalties?
* Should the system allow for automatic generation of fee challans based on due dates?

### Muhammad Irfan (FA21-BSE-027-4A)

#### Use Case UC1: Add Warden

**Use Case Name: Add Warden**

**Scope:** Hostel Management System

**Level:** User Goal

**Primary Actor:** Admin

**Stakeholders and Interests:**

* Admin: Wants to add a new warden to the system.
* Warden: Wants to be added to the system to manage the hostel effectively.

**Preconditions:**

* The admin is logged into the system.
* The admin has the necessary privileges to add a new warden.

**Main Success Scenario:**

| **User Interaction** | **System Response** |
| --- | --- |
| 1. The admin selects the "Add New Warden" option from the main menu. | 2. The system displays a form for adding a new warden on the screen. |
| 3. The admin fills in the details of the new warden, including name, contact information, and address. | 4. The system validates the input and creates a new warden profile with the entered details. |
| 5. The admin assigns a username and password to the new warden for accessing the system. | 6. The system verifies the uniqueness of the username and password and associates them with the new warden profile. |
| 7. The admin saves the new warden profile to the system. | 8. The system updates the records and displays a success message to the admin. |

**Extensions:**

* 3a. If the admin enters invalid or incomplete information, the system displays an error message and prompts the admin to correct the input.
* 5a. If the admin chooses a username or password that is already taken, the system displays an error message and prompts the admin to select a different one.

**Special Requirements:**

* The system must ensure that each warden has a unique username and password.
* The system must provide appropriate validation and error messages for input fields.

**Technology and Data Variation List:**

* The system can store wardens' data in a database or other storage system.
* The system can use various forms of validation to ensure data quality and consistency.

**Open Issues:**

* How should the system handle case where a warden's information needs to be updated or deleted?
* Should the system provide any additional functionality or tools for managing wardens?

#### Use Case UC2: Delete Warden

**Use Case Name:** Delete Warden

**Scope:** Hostel Management System

**Level:** User goal

**Primary Actor:** Admin

**Stakeholders and Interests:**

* Admin: Wants to delete a warden from the system.
* Warden: May lose access to the system if deleted.

**Preconditions:**

* The admin must be logged into the hostel management system.
* There must be at least one warden registered in the system.

**Main Success Scenario:**

| **User Interaction** | **System Response** |
| --- | --- |
| 1. The admin selects the "Delete Warden" option from the main menu. | 2. The system displays a list of all registered wardens. |
| 3. The admin selects the warden to delete from the list. | 4. The system displays a confirmation message asking the admin to confirm the deletion. |
| 5. The admin confirms the deletion. | 6. The system removes the selected warden from the system, revokes their access to the system, and updates the records accordingly. |
| 7. The system displays a success message to the admin. | 8. The use case ends. |

**Extensions:**

* 3a. If there are no wardens registered in the system, the system displays a message to the admin stating that there are no wardens to delete.
* 5a. If the admin cancels the deletion, the system returns to step 1 of the main success scenario.

**Special Requirements:**

* The system must keep a record of all deleted wardens.
* The system must ensure that the deleted warden's access to the system is revoked immediately.

**Technology and Data Variation List:**

* The system must be compatible with different web browsers and operating systems.
* The system must be able to handle many registered wardens and deleted records.

**Open Issues:**

* How will the system ensure that the deleted warden's personal information is securely deleted from the system?
* How will the system handle any errors that occur during the deletion process?

#### Use Case UC3: View Wardens

Scope: Hostel Management System

Level: User-goal level

Primary Actor: Admin

Stakeholders:

* Wardens: Their details will be displayed in the list and can be viewed individually.
* Students: They may be impacted indirectly by the performance of the wardens.
* Hostel Management System: The system will provide the list of wardens and their details to the admin.

Precondition: The admin must be logged into the Hostel Management System.

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| Admin selects "View Wardens" option from the main menu | System retrieves the list of all registered wardens |
| Admin views the list of wardens displayed on the screen | System displays details of all registered wardens, such as name, contact information, and job title |
| Admin sorts the list based on specific criteria or filters the list based on certain parameters | System reorders or filters the list as per the admin's selection |
| Admin clicks on a specific warden's name from the list to view their details | System displays the details of the selected warden |

Extensions (Alternative flows):

1. If there are no registered wardens, the system displays a message indicating the same.
2. If the admin selects to filter the list based on certain parameters that do not match any registered wardens, the system displays a message indicating that there are no results matching the criteria.
3. If the admin selects a specific warden's name that has been removed or deactivated, the system displays a message indicating that the warden details are not available.

Special Requirements:

* The system should have access to a database containing details of all registered wardens.
* The system should provide the option to sort or filter the list of wardens based on specific criteria or parameters.
* The system should display the warden details in an organized and easy-to-read format.

Technology and Data Variation List:

* The system should be able to retrieve and display wardens' details from the database, irrespective of the size of the data.
* The system should be able to handle various search parameters and criteria, such as job title, date of registration, active or inactive status, etc.
* The system should be able to display the details of wardens in different formats, such as tables or lists, depending on the admin's preference.

Open Issues:

* How frequently should the list of registered wardens be updated in the system?
* What level of access should be granted to the admin for viewing warden details?
* How will the system ensure data privacy and security for the warden and admin details?

#### Use Case UC4: Add security guard.

Scope: Hostel Management System

Level: User goal level

Primary Actor: Admin

Stakeholders:

* Admin: To add new security personnel to the system
* Security Personnel: To be added to the system and granted appropriate access

Precondition: Admin is logged into the hostel management system and has appropriate permissions to add security personnel.

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| Admin selects "Add Security Personnel" option from the main menu | System prompts admin to enter details of new security personnel |
| Admin enters name, contact information, and job title of new security personnel | System creates new user account with login credentials for security personnel |
| Admin confirms addition of new security personnel | System grants appropriate access to the system for the security personnel |
|  | System displays a confirmation message to the admin |

Extensions:

* If the admin enters incomplete or invalid details for the new security personnel:
  + System prompts the admin to correct the details.
* If there is an existing user account with the same contact information:
  + System prompts the admin to either merge or overwrite the existing user account or cancel the addition of the new security personnel.
* If there is an error in creating a new user account or granting appropriate access:
  + System displays an error message to the admin and rolls back the changes made.

Special Requirements:

* The system must generate unique login credentials for each new security personnel.
* The system must verify the contact information of the new security personnel to ensure that there are no duplicate accounts with the same contact information.

Technology and Data Variation List:

* Hostel management system software
* Database containing details of the security personnel and their access permissions.
* User interface for admin to enter details of the new security personnel.

Open Issues:

* How to handle situations where the new security personnel already have an existing account in the system?
* What are the specific access permissions that should be granted to the new security personnel?
* How to ensure the security of the login credentials generated for the new security personnel?

#### Use Case UC5: View Security guard

Scope: Hostel management system

Level: User goal level

Primary Actor: Admin

Stakeholders: Admin, Security Personnel

Precondition: Admin is logged into the hostel management system.

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| Admin selects "View Security Personnel" option from the main menu. | System retrieves the list of all registered security personnel. |
| Admin views the details of security personnel such as name, contact information, and job title, on the screen. | System displays the details of security personnel. |
| Admin sorts the list based on specific criteria, such as job title or date of registration. | System sorts the list based on the selected criteria. |
| Admin filters the list based on certain parameters, such as active or inactive status. | System filters the list based on the selected parameters. |
| Admin clicks on the name of any specific security personnel in the list. | System displays the details of the selected security personnel. |

Extensions:

| **User Interaction** | **System Response** |
| --- | --- |
| Admin selects "View Security Personnel" option from the main menu. | System displays an error message if no security personnel are registered. |
| Admin filters the list based on certain parameters, such as active or inactive status. | System displays an error message if no security personnel meet the selected criteria. |
| Admin clicks on the name of any specific security personnel in the list. | System displays an error message if the selected security personnel details are not available. |

Special Requirements: None Technology and Data Variation List:

* The system should display the details of security personnel in a user-friendly manner, suitable for viewing on a computer screen or mobile device.
* The system should be able to retrieve and display large amounts of data quickly and accurately.
* The system should be compatible with various web browsers and operating systems.

Open Issues:

* Security and privacy concerns regarding the display of personal information of security personnel.
* Possibility of unauthorized access to the system and security personnel data.

#### Use Case UC6: Delete Security guard

Scope: Hostel Management System

Level: User goal

Primary Actor: Admin

Stakeholders: Security Personnel

Precondition: The admin is logged into the hostel management system.

Main Success Scenario:

| **User Actions** | **System Responses** |
| --- | --- |
| 1. The admin selects the "Delete Security Personnel" option from the main menu. | 2. The system retrieves the list of all registered security personnel and displays their details, such as name, contact information, and job title, on the screen. |
| 3. The admin selects the security personnel to delete from the list. | 4. The system confirms the action and removes the selected security personnel from the system. |
| 5. The system revokes the security personnel's access to the system. | 6. The system updates the records accordingly. |
| 7. The system displays a confirmation message to the admin. |  |

Extensions:

4a. The admin cancels the delete action.

* The system cancels the delete action and returns to the main menu.

Special Requirements:

* The system should have appropriate access control mechanisms to prevent unauthorized deletion of security personnel.

Technology and Data Variation List:

* The system must support the deletion of security personnel records from the database.
* The system must support the revocation of security personnel's access to the system.

Open Issues:

* None.

#### Use Case UC7: Add Cook

Scope: Hostel management system

Level: User goal

Primary Actor: Admin

Stakeholders:

* Cooks who will use the system to access their work schedule and other relevant information.
* Hostel residents who rely on the cooks to provide food services. Precondition: Admin is logged into the hostel management system.

Main Success Scenario:

| **User Action** | **System Response** |
| --- | --- |
| Admin selects "Add Cook" option from main menu | System displays form to enter new cook details |
| Admin enters the details of the new cook | System confirms the details with the admin |
| Admin confirms the addition of the new cook | System creates a new user account with login credentials |
| System grants appropriate access to the new cook | System sends confirmation message to the admin |

Extensions:

* If the new cook details are incomplete or incorrect:
  + System displays an error message and prompts the admin to correct the details.
* If the admin cancels the addition of the new cook:
  + System cancels the addition process and displays a message to the admin.

Special Requirements:

* The system should validate the entered details of the new cook to ensure data integrity and accuracy.
* The system should provide appropriate access to the new cook based on their job title and responsibilities.

Technology and Data Variation List:

* The system should support different input methods such as keyboard and voice input.
* The system should support different languages for input and output.
* The system should store the new cook's details securely in the database.

Open Issues:

* Should the system prompt the admin to enter the new cook's work schedule as well during the addition process?
* How will the system manage multiple cooks with the same name?

#### Use Case UC8: View Cook

Scope: Hostel Management System

Level: User goal

Primary Actor: Admin

Stakeholders:

* Admin: The user who wants to view the details of the registered cooks.
* Cooks: The individuals whose details are being viewed.
* Hostel Management System: The system that manages the registration and details of the cooks.

Precondition: The admin has logged into the hostel management system.

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| The admin selects the "View Cooks" option from the main menu. | The system retrieves the list of all registered cooks and displays their details, such as name, contact information, and job title, on the screen. |
| The admin can sort the list based on specific criteria, such as job title or date of registration, and can also filter the list based on certain parameters, such as active or inactive status. | The system updates the list based on the admin's selection criteria. |
| The admin can view the details of any specific cook by clicking on their name in the list. | The system displays the details of the selected cook on the screen. |

Extensions (Alternative Flows):

* If there are no registered cooks, the system displays a message indicating that there are no cooks to display.

Special Requirements: None.

Technology and Data Variation List:

* The system should display the list of cooks in a user-friendly and easily understandable format.
* The system should have the ability to sort and filter the list based on specific criteria and parameters.

Open Issues:

* None.

#### Use Case UC9: Delete Cook

Scope: Hostel Management System

Level: User Goal

Primary Actor: Admin

Stakeholders: Cooks, Hostel Management Staff

Precondition: The admin is logged into the Hostel Management System.

Main Success Scenario:

| **User Interaction** | **System Response** |
| --- | --- |
| 1. Admin selects "Delete Cook" option from the main menu. | 2. System retrieves the list of all registered cooks and displays their details, such as name, contact information, and job title, on the screen. |
| 3. Admin selects the cook to delete from the list and confirms the action. | 4. System removes the selected cook from the system, revokes their access to the system, and updates the records accordingly. |
| 5. System displays a confirmation message to the admin. | 6. Admin acknowledges the confirmation message. |

Extensions (Alternative Flows): 3a. Admin cancels the deletion of a cook:

| **User Interaction** | **System Response** |
| --- | --- |
| 1. Admin selects "Delete Cook" option from the main menu. | 2. System retrieves the list of all registered cooks and displays their details, such as name, contact information, and job title, on the screen. |
| 3a. Admin cancels the deletion of a cook. | 3b. System returns the admin to the main menu. |

Special Requirements:

* Only an admin is authorized to delete cooks from the system.
* The system must update all relevant records upon deletion of a cook.

Technology and Data Variation List:

* Hostel Management System
* Database of registered cooks and their details
* Confirmation message displayed to the admin.

Open Issues:

* None

## PROTOTYPES















# CHAPTER 3: DOMAIN MODEL

## Domain Model



# CHAPTER 4: SYSTEM SEQUENCE DIAGRAM (SSD)

### **Laiba binta tahir (FA21-BSE-019-4A)**

#### Scenario: Register



#### Scenario: login



#### Scenario: add student

Table

Description automatically generated

#### Scenario: view student

Diagram, table

Description automatically generated with medium confidence

#### Scenario: delete student

Table

Description automatically generated with medium confidence

#### Scenario: logout

Diagram

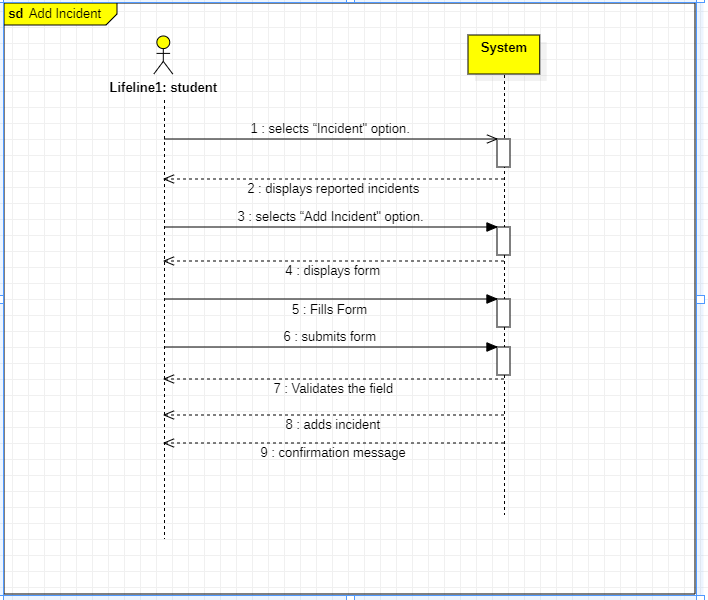
Description automatically generated with medium confidence

### **Arfah Ali (FA21-BSE-080-4A)**

#### Scenario: View Incidents



#### Scenario: Add incidents



#### Scenario: Feedback from students



#### Scenario: add menu

Table

Description automatically generated

#### Scenario: view menu



### **Hanzla Nouman (FA21-BSE-015-4A)**

#### Scenario: Allocate Room



#### Scenario: View Rooms



#### Scenario: Terminate Allocation



#### Scenario: Add New Room



#### Scenario: Delete Room



#### Scenario: Assign Facility



#### Scenario: Remove Facility



#### Scenario: Generate Fee



### **Muhammad Irfan (FA21-BSE-027-4A)**

#### Scenario: Add Warden:-



#### Scenario: View Warden:-

A screenshot of a computer

Description automatically generated with low confidence

#### Scenario: Add Security Guard:-

A screenshot of a computer screen

Description automatically generated with low confidence

#### Scenario: View Security Guard:-

A screenshot of a computer screen

Description automatically generated with low confidence

#### Scenario: Delete Security Guard:-

A screenshot of a computer

Description automatically generated with medium confidence

#### Scenario: Add Cook:-

A screenshot of a computer screen

Description automatically generated with low confidence

#### Scenario: View Cook:-

A screenshot of a computer

Description automatically generated with medium confidence

#### A screenshot of a computer Description automatically generated with medium confidenceScenario: Delete Cook:-

# CHAPTER 5: OPERATION CONTRACT

## Laiba Binta Tahir(FA21-BSE-019-4A)

|  |  |
| --- | --- |
| **Operation** | **Register (Personal Details, Student ID)** |
| Cross-Reference | This contract operation is related to the use case of registering for an account in the hostel management system. |
| Pre-Conditions | 1. The user must have accessed the registration form of the hostel management system. |
|  | 2. The user must have the necessary information required for registration. |
|  | - Personal details (such as name, email, contact information) |
|  | - Student ID |
| Post-Conditions | 1. The system will validate the entered information to ensure it meets the required criteria. |
|  | 2. If the entered information is valid, the system will create a new account for the student and save the details in the database. |
|  | 3. The system will display a success message indicating that the registration is successful. |
|  | 4. The system will redirect the user to the login page. |

|  |  |
| --- | --- |
| **Operation** | **Login** (Username, Password) |
| Cross-Reference | This contract operation is related to the use case of logging into the hostel management system. |
| Pre-Conditions | 1. The user must have accessed the login page of the hostel management system.  2. The user must have a registered account on the hostel management system.  3. The user must have entered valid credentials, including their username and password. |
| Post-Conditions | 1. The system will authenticate the user's credentials and verify that they are valid and registered on the system.  2. The system will grant the user access to their account dashboard on the hostel management system.  3. The user will be able to view and manage their hostel accommodation applications and other relevant information.  4. If the user's credentials are not valid, the system will deny access to the account dashboard and display an error message prompting the user to enter valid credentials. |

|  |  |
| --- | --- |
|  |  |
| **Operation** | **AddStudent** (StudentName, StudentID, Gender, RoomNumber, HostelName) |
| Cross-Reference | This contract operation is related to the use case of adding a new student to the hostel management system. |
| Pre-Conditions | 1. The user must have administrative privileges to access the "Add Student" functionality in the hostel management system.  2. The user must have entered valid student details, including the student's name, ID, gender, room number, and the name of the hostel where the student will reside. |
| Post-Conditions | 1. The system will validate the student's details and ensure that the student does not already exist in the system.  2. The system will add the new student's details to the hostel management system and assign them a unique student ID.  3. The student's details will be saved to the system's database for future reference. <br> 4. The system will generate a confirmation message indicating that the student has been successfully added to the hostel management system. |

|  |  |
| --- | --- |
| **Operation** | **ViewStudent** |
| Parameters | StudentID |
| Cross-Reference | This contract operation is related to the use case of viewing a student's details in the hostel management system. |
| Pre-Conditions | 1. The user must have administrative or student privileges to access the "View Student" functionality in the hostel management system.  2. The user must have entered a valid student ID for the student whose details they wish to view. |
| Post-Conditions | 1. The system will validate the student ID entered by the user and ensure that it exists in the system's database.  2. The system will retrieve the student's details from the database and display them on the screen for the user to view.  3. If the user has administrative privileges, they will be able to edit or delete the student's details from the system.  4. If the user has student privileges, they will only be able to view their own details and will not have access to edit or delete them. |

|  |  |
| --- | --- |
|  |  |
| **Operation** | **DeleteStudent** (StudentID) |
| Cross-Reference | This contract operation is related to the use case of deleting a student from the hostel management system. |
| Pre-Conditions | 1. The user must have administrative privileges to access the "Delete Student" functionality in the hostel management system.  2. The user must have entered a valid student ID for the student they wish to delete from the system. |
| Post-Conditions | 1. The system will validate the student ID entered by the user and ensure that it exists in the system's database.  2. The system will delete the student's details from the hostel management system's database, including their accommodation details and any other relevant information.  3. The system will generate a confirmation message indicating that the student has been successfully deleted from the hostel management system. |
| **Operation** | **Logout**(none) |
| Cross-Reference | This contract operation is related to the use case of logging out of the hostel management system. |
| Pre-Conditions | 1. The user must be currently logged in to the hostel management system. |
| Post-Conditions | 1. The system will clear the user's session data and invalidate their login credentials, effectively logging them out of the system. <br> 2. The system will redirect the user to the login page and display a confirmation message indicating that they have been successfully logged out. |

## Arfah Ali (FA21-BSE-080-4A)

|  |  |
| --- | --- |
| Operation | AddIncident (IncidentID, Day, Date, Time, Location, Description) |
| Cross-Reference | This contract operation is related to the use case of adding incidents in the hostel management system. |
| Pre-Conditions | 1. The student must be registered in the hostel management system. 2. The incident must have occurred in the hostel premises. |
| Post-Conditions | 1. The incident report is added to the hostel management system. 2. The student receives a confirmation display stating "Incident Registered." |

|  |  |
| --- | --- |
| Operation | NotifyStudent |
| Cross-Reference | Feedback from Student |
| Pre-Conditions | A student must have submitted feedback through the feedback system.  The feedback must have been received and processed by the hostel management system.  The hostel administrators must have responded to the feedback and provided a resolution or follow-up action. |
| Post-Conditions | The student who submitted the feedback will receive a notification regarding the response and any relevant follow-up actions.  The notification will include details about the feedback, the response, and any relevant follow-up actions.  The student will be able to view the notification in their account on the hostel management system. |

|  |  |
| --- | --- |
| Operation | DownloadPaymentReceipt |
| Cross-Reference | Submit Fee |
| Pre-Conditions | The student must have submitted their fee through the payment system.  The payment must have been received and processed by the hostel management system.  The payment receipt must be generated and available for download. |
| Post-Conditions | The student will be able to download the payment receipt for their fee.  The payment receipt will include details such as the amount paid, the payment date and time, the payment method used, and any relevant transaction IDs or confirmation numbers.  The payment receipt will be saved in the student's account on the hostel management system for future reference. |

|  |  |
| --- | --- |
| Operation | EnterPaymentDetails |
| Cross-Reference | Submit Fee |
| Pre-Conditions | The student must have selected the payment method they wish to use for submitting the fee.  The payment gateway or service being used must be available and functioning correctly.  The student must have access to the necessary information to complete the payment, such as their bank account or credit card details. |
| Post-Conditions | The student's payment details will be stored securely in the hostel management system.  The payment details will include information such as the amount to be paid, the payment method used, and any relevant transaction IDs or confirmation numbers.  The student will be able to proceed with submitting the payment. |

|  |  |
| --- | --- |
| Operation | SearchFacilities |
| Cross-Reference | SelectFacilty |
| Pre-Conditions | The student must have already logged into the hostel management system.  The student must have selected the "Select Facilities" option from the system menu.  The student must have access to the list of available facilities. |
| Post-Conditions | The system will display a list of facilities that match the search criteria entered by the student.  The student will be able to view the details of each facility, such as its price, availability, and location.  The student will be able to select one or more facilities from the list. |

|  |  |
| --- | --- |
| Operation | SearchForIncidents |
| Cross-Reference | View Incident |
| Pre-Conditions | The warden must be logged in to the hostel management system.  The warden must have the necessary permissions to view incidents.  The incidents must exist in the system. |
| Post-Conditions | The system should display a list of incidents that match the search criteria.  The search results should include the incident's basic details (e.g. ID, summary, status, priority).  The warden should be able to select an incident from the search results to view its full details.  If no matching incidents are found, the system should display an appropriate message. |

|  |  |
| --- | --- |
| Operation | AttachFiles |
| Cross-Reference | Add Incident |
| Pre-Conditions | The student must be logged in to the hostel management system.  The student must have the necessary permissions to add an incident and attach files.  The incident must not be closed.  The file(s) to be attached must exist and be accessible by the student. |
| Post-Conditions | The file(s) should be successfully attached to the incident and stored in the system.  The date and time of the attachment should be recorded.  The student who attached the file(s) should be identified and recorded.  The incident status should remain the same. |

|  |  |
| --- | --- |
| Operation | AddIncidentDescription |
| Cross-Reference | Add Incident |
| Pre-Conditions | The student must be logged in to the hostel management system.  He has appropriate permissions to add incident descriptions.  An incident must already exist that the student wants to add a description for. |
| Post-Conditions | The date and time of the update should be recorded.  The student who added the facility should be identified and recorded.  The incident status should remain the same. |

|  |  |
| --- | --- |
| Operation | UpdateStatus |
| Cross-Reference | Add Incident |
| Pre-Conditions | The incident must exist in the system.  The incident's status must be allowed to be updated (e.g. not closed). |
| Post-Conditions | The incident's status should be updated with the new value set by the warden.  The date and time of the status update should be recorded.  The incident's details should be saved in the system with the updated status.  If the incident status change triggers a notification to be sent, it should be sent according to the system's notification rules. |

|  |  |
| --- | --- |
| **Operation** | **AddMenu** (Day, Breakfast, Lunch, Dinner) |
| Cross-Reference | This contract operation is related to the use case of adding a new menu to the hostel management system. |
| Pre-Conditions | 1. -The user is authorized to access the hostel management system.  2. The user must have entered valid menu details, including the Day, Breakfast, Lunch, Dinner. |
| Post-Conditions | 1. The system will validate the menu details and ensure that the menu form does not has Null field.  2. The system will add the new menu's details to the hostel management system.  3. The menu details will be saved to the system's database for future reference.  4. The system will generate a confirmation message indicating that the menu has been successfully added to the hostel management system. |

|  |  |
| --- | --- |
|  |  |
| **Operation** | **ViewMenu** (MenuID) |
| Cross-Reference | This contract operation is related to the use case of viewing a menu in the hostel management system. |
| Pre-Conditions | 1. The user must have student or administrative privileges to access the "View Menu" functionality in the hostel management system.  2. The user must have entered a valid menu ID for the menu they wish to view. |
| Post-Conditions | 1. The system will validate the menu ID entered by the user and ensure that it exists in the system's database.  2. The system will retrieve the menu details from the database and display them on the screen for the user to view.  3. If the user has administrative privileges, they will be able to edit or delete the menu details from the system.  4. If the user has student privileges, they will only be able to view the menu details and will not have access to edit or delete them. |

## Hanzla Nouman (FA21-BSE-015-4A)

Operation Contract for UC1: Allocate Room

| **Attribute** | **Description** |
| --- | --- |
| Operation | AllocateRoom(studentID, roomNumber) |
| Cross Reference | Use Case UC1: Allocate Room |
| Preconditions | The warden is logged into the hostel management system.  The student has completed the necessary application and registration processes.  The selected room is available and not already allocated to another student. |
| Postconditions | The room is allocated to the student.  The room status is updated to "Allocated."  A confirmation message with the allocated room number is generated and displayed to the warden. |

Operation Contract for UC2: View Rooms

| **Attribute** | **Description** |
| --- | --- |
| Operation | ViewRooms() |
| Cross Reference | Use Case UC2: View Rooms |
| Preconditions | The warden is logged into the hostel management system. |
| Postconditions | A list of all available rooms in the hostel along with their current occupancy status, room type, and room number is displayed. |

Operation Contract for UC3: Terminate Allocation

| **Attribute** | **Description** |
| --- | --- |
| Operation | TerminateAllocation(studentID, roomNumber) |
| Cross Reference | Use Case UC3: Terminate Allocation |
| Preconditions | The warden is logged into the hostel management system.  The selected room is allocated to the student. |
| Postconditions | The room allocation for the student is terminated • The availability status of the room is updated in the system.  A notification is sent to the student, informing them of the termination of their room allocation. |

Operation Contract for UC4: Add New Rooms

| **Attribute** | **Description** |
| --- | --- |
| Operation | AddRoom(roomNumber, roomType, availabilityStatus) |
| Cross Reference | Use Case UC4: Add New Rooms |
| Preconditions | The warden is logged into the hostel management system.  The room number is unique and does not already exist in the system. |
| Postconditions | The new room is added to the list of available rooms.  A confirmation message is displayed to the warden. |

Operation Contract for UC5: Delete Room

| **Attribute** | **Description** |
| --- | --- |
| Operation | DeleteRoom(roomNumber) |
| Cross Reference | Use Case UC5: Delete Room |
| Preconditions | The warden is logged into the hostel management system.  The selected room is not allocated to a student. |
| Postconditions | The room is removed from the list of available rooms.  A confirmation message is displayed to the warden. |

Operation Contract for UC6: Assign Facility

| **Section** | **Description** |
| --- | --- |
| Operation | Assign Facility |
| Cross References | Use Case UC6: Assign Facility |
| Preconditions | The warden must be logged into the hostel management system.  The facilities to be assigned must be available. |
| Postconditions | The selected facilities are assigned to the student.  The system displays a confirmation message to the warden. |

Operation Contract for UC7: Remove Facility

| **Section** | **Description** |
| --- | --- |
| Operation | Remove Facility |
| Cross References | Use Case UC7: Remove Facility |
| Preconditions | The warden must be logged into the hostel management system and have the necessary permissions to remove facilities. |
| Postconditions | The selected facility is removed from the student's record.  The system updates the availability status of the facility.  The system displays a confirmation message to the warden. |

Operation Contract for UC8: Generate Fee

| **Section** | **Description** |
| --- | --- |
| Operation | Generate Fee Challan |
| Cross References | Use Case UC8: Generate Fee Challan |
| Preconditions | The warden has logged into the hostel management system. The list of registered students and their hostel fees are available in the system. |
| Postconditions | The system generates the fee challan for the selected student.  The student's tab is updated with the fee challan details.  The system displays a confirmation message to the warden. |

## Muhammad Irfan (FA21-BSE-027-4A)

|  |  |
| --- | --- |
| **Operation** | **addWarden** (Warden Name, Warden ID, Gender) |
| Cross-Reference | This contract operation is related to the use case of adding a new Warden to the hostel management system. |
| Pre-Conditions | 1. The user must have administrative privileges to access the "Add Warden" functionality in the hostel management system.  2. The user must have entered valid warden details, including the warden's name, ID, gender. |
| Post-Conditions | 1. The system will validate the warden's details and ensure that the warden does not already exist in the system.  2. The system will add the new warden's details to the hostel management system and assign them a unique student ID.  3. The warden's details will be saved to the system's database for future reference. 4. The system will generate a confirmation message indicating that the warden has been successfully added to the hostel management system. |

|  |  |
| --- | --- |
| **Operation** | **viewWarden** |
| Parameters | Warden ID |
| Cross-Reference | This contract operation is related to the use case of viewing a warden's details in the hostel management system. |
| Pre-Conditions | 1. The user must have administrative or warden privileges to access the "View Student" functionality in the hostel management system.  2. The user must have entered a valid warden ID for the warden whose details they wish to view. |
| Post-Conditions | 1. The system will validate the warden ID entered by the user and ensure that it exists in the system's database.  2. The system will retrieve the warden's details from the database and display them on the screen for the user to view.  3. If the user has administrative privileges, they will be able to edit or delete the warden's details from the system.  4. If the user has warden privileges, they will only be able to view their own details and will not have access to edit or delete them. |

|  |  |
| --- | --- |
|  |  |
| **Operation** | **deleteWarden** (Warden ID) |
| Cross-Reference | This contract operation is related to the use case of deleting a warden from the hostel management system. |
| Pre-Conditions | 1. The user must have administrative privileges to access the "Delete warden" functionality in the hostel management system.  2. The user must have entered a valid warden ID for the warden they wish to delete from the system. |
| Post-Conditions | 1. The system will validate the warden ID entered by the user and ensure that it exists in the system's database.  2. The system will delete the warden 's details from the hostel management system's database, including their accommodation details and any other relevant information.  3. The system will generate a confirmation message indicating that the warden has been successfully deleted from the hostel management system. |

|  |  |
| --- | --- |
| **Operation** | **addSecurityGuard** (Security guard Name, Security guard ID, Gender) |
| Cross-Reference | This contract operation is related to the use case of adding a new Security guard to the hostel management system. |
| Pre-Conditions | 1. The user must have administrative privileges to access the "Security guard" functionality in the hostel management system.  2. The user must have entered valid Security guard details, including the Security guard's name, ID, gender. |
| Post-Conditions | 1. The system will validate the Security guard's details and ensure that the Security guard does not already exist in the system.  2. The system will add the new Security guard's details to the hostel management system and assign them a unique Security guard.  3. The Security guard's details will be saved to the system's database for future reference.  4. The system will generate a confirmation message indicating that the Security guard has been successfully added to the hostel management system. |

|  |  |
| --- | --- |
| **Operation** | **View Security guard** |
| Parameters | Security guard ID |
| Cross-Reference | This contract operation is related to the use case of viewing a Security guard's details in the hostel management system. |
| Pre-Conditions | 1. The user must have administrative or warden privileges to access the " view Security guard" functionality in the hostel management system.  2. The user must have entered a valid Security guard ID for the Security guard whose details they wish to view. |
| Post-Conditions | 1. The system will validate the Security guard ID entered by the user and ensure that it exists in the system's database.  2. The system will retrieve the Security guard's details from the database and display them on the screen for the user to view.  3. If the user has administrative privileges, they will be able to edit or delete the Security guard's details from the system.  4. If the user has Security guard privileges, they will only be able to view their own details and will not have access to edit or delete them. |

|  |  |
| --- | --- |
|  |  |
| **Operation** | **Delete Security guard** (Security guard ID) |
| Cross-Reference | This contract operation is related to the use case of deleting a Security guard from the hostel management system. |
| Pre-Conditions | 1. The user must have administrative privileges to access the "Delete Security guard" functionality in the hostel management system.  2. The user must have entered a valid Security guard ID for the Security guard they wish to delete from the system. |
| Post-Conditions | 1. The system will validate the Security guard ID entered by the user and ensure that it exists in the system's database.  2. The system will delete the Security guard 's details from the hostel management system's database, including their accommodation details and any other relevant information.  3. The system will generate a confirmation message indicating that the Security guard has been successfully deleted from the hostel management system. |

|  |  |
| --- | --- |
| **Operation** | **Add Cook** (Cook Name, Cook ID, Gender) |
| Cross-Reference | This contract operation is related to the use case of adding a new Cook to the hostel management system. |
| Pre-Conditions | 1. The user must have administrative privileges to access the "Add Cook " functionality in the hostel management system.  2. The user must have entered valid Cook details, including the Cook 's name, ID, gender. |
| Post-Conditions | 1. The system will validate the Cook 's details and ensure that the Cook does not already exist in the system.  2. The system will add the new Cook 's details to the hostel management system and assign them a unique cook ID.  3. The Cook 's details will be saved to the system's database for future reference.  4. The system will generate a confirmation message indicating that the Cook has been successfully added to the hostel management system. |

|  |  |
| --- | --- |
| **Operation** | **View** **Cook** |
| Parameters | Cook ID |
| Cross-Reference | This contract operation is related to the use case of viewing a Cook 's details in the hostel management system. |
| Pre-Conditions | 1. The user must have administrative or warden privileges to access the "View Cook" functionality in the hostel management system.  2. The user must have entered a valid Cook ID for the cook whose details they wish to view. |
| Post-Conditions | 1. The system will validate the Cook ID entered by the user and ensure that it exists in the system's database.  2. The system will retrieve the Cook 's details from the database and display them on the screen for the user to view.  3. If the user has administrative privileges, they will be able to edit or delete the Cook's details from the system.  4. If the user has Cook privileges, they will only be able to view their own details and will not have access to edit or delete them. |

|  |  |
| --- | --- |
|  |  |
| **Operation** | **Delete** **Cook** (Cook ID) |
| Cross-Reference | This contract operation is related to the use case of deleting a Cook from the hostel management system. |
| Pre-Conditions | 1. The user must have administrative privileges to access the "Delete Cook " functionality in the hostel management system.  2. The user must have entered a valid Cook ID for the Cook they wish to delete from the system. |
| Post-Conditions | 1. The system will validate the Cook ID entered by the user and ensure that it exists in the system's database.  2. The system will delete Cook’s details from the hostel management system's database, including their accommodation details and any other relevant information.  3. The system will generate a confirmation message indicating that the Cook has been successfully deleted from the hostel management system. |

# CHAPTER 6: SOFTWARE ARCHITECTURE

## Package diagram



# CHAPTER 7: INTERACTION DIAGRAM

## Communication diagram

### Laiba binta tahir (FA21-BSE-019-4A)

*Add student :*

### Arfah Ali (FA21-BSE-080)

Add Incident :



### Muhammad Irfan (FA21-BSE-027)

addWarden:



## Sequence diagram

### Laiba binta tahir (FA21-BSE-019-4A)

Add student :



### Arfah Ali (FA21-BSE-080)

Add Incident :



### Hanzla Nouman(FA21-BSE-015)

Add Room :



### Muahmmad Irfan(FA21-BSE-027)

Add Warden:



# CHAPTER 8: CLASS DIAGRAM

