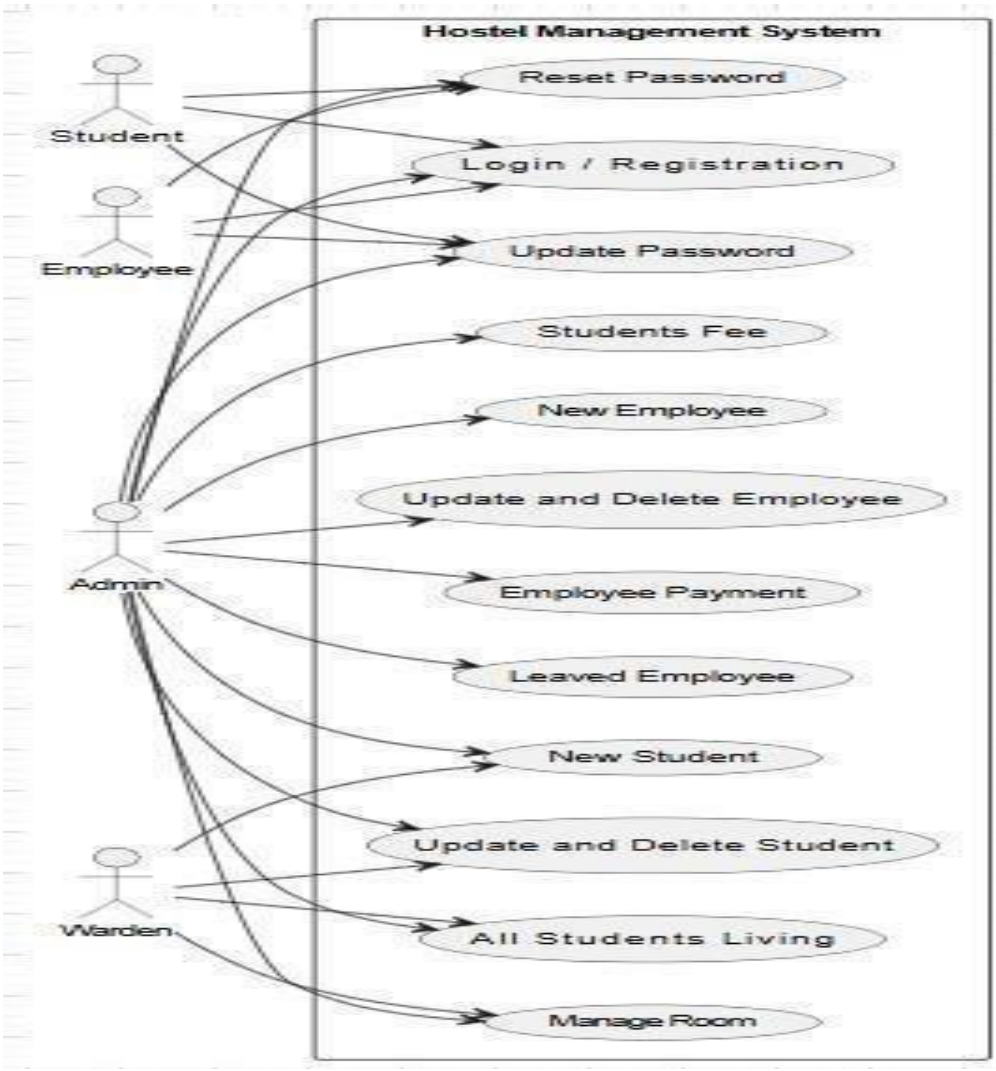
PROJECT TITLE: HOTEL MANAGEMENT SYSTEM

Contents

Nouman Khan Sardar Zain
Abdullah sajid
Daniyal Murtaza
Kazim Shaukat
Sayyam Tahir
Uzair Arif
Adil Bashir

NAME: Nouman Khan

REGISTRAION NO: SP23-BSE-012



Use Case Name:

Manage Room

Primary Actor:

Hostel Manager / Admin **Secondary Actors:**

Maintenance Staff, Room Allocation System, Hostel Warden **Stakeholders and Interests:**

- **Hostel Manager** / **Admin**: Needs to allocate, update, and maintain rooms for students or employees, ensuring proper occupancy records and room conditions.
- **Maintenance Staff**: Responsible for ensuring that rooms are properly maintained and cleaned.
- **Hostel Warden**: Manages room allocation for students/employees, ensures room conditions are suitable for habitation.

Preconditions: 1. The hostel has rooms available in the system for

allocation or maintenance.

2. The system has accurate records of all rooms, including room numbers, current occupants, and room status.
3. The user (Hostel Manager/Admin) is logged in and has the necessary permissions to manage rooms.
4. The system is functioning and can update room allocation, status, and maintenance records.

Postconditions:

1. Room status is updated correctly (e.g., vacant, occupied, under maintenance).
2. Room is assigned to the relevant student, employee, or guest if available.
3. Room maintenance is tracked, and relevant tasks are assigned to the maintenance staff.
4. Reports on room occupancy and status are updated for record-keeping and auditing purposes.
5. Notifications are sent to the concerned parties (e.g., student, warden, maintenance staff) when a room is allocated or requires attention.

Main Success Scenario (Basic Flow):

1. **Trigger:** The Hostel Manager/Admin needs to perform a room management action (e.g., allocate, update status, schedule maintenance).
2. **Hostel Manager Action:**
    - The Hostel Manager logs into the **Hostel Management System**.
    - The system displays an overview of **all rooms** in the hostel, including their status (e.g., available, occupied, under maintenance).
3. **Room Allocation (if applicable):** ○ The Hostel Manager navigates to the **Room Allocation** section and selects an available room.
    - The system displays a list of **available rooms** (with the room's size, type, and other relevant details). ○ The Hostel Manager selects the room and assigns it to a **new student, employee, or guest**. ○ The system updates the room's **status** to **occupied** and records the occupant's details.
4. **Room Status Update (if applicable):**

- If the room status needs updating (e.g., marking a room as **under maintenance** or **vacant**), the Hostel Manager selects the room and updates the **status**. ○ The system prompts the Hostel Manager to provide details about the status update (e.g., maintenance issues or reasons for vacancy).

5. **Room Maintenance (if applicable):** ○ If the room requires maintenance (e.g., cleaning, repairs), the Hostel Manager can select the **maintenance option**. ○ The system notifies the **Maintenance Staff** about the required tasks and provides them with room details. ○ Maintenance staff records completion of tasks, and the room status is updated to **ready for occupancy** or **vacant** once maintenance is done.
6. **Reports Update:**
   - The system automatically updates the room **occupancy report**, including details of current occupants, vacant rooms, and maintenance status.
   - The **Room Status Report** is generated for auditing and tracking purposes.
7. **Notification Sent:** ○ The system sends a **notification** to the concerned parties (e.g., student/employee about room allocation, maintenance staff about maintenance tasks, warden about room status changes).

Alternative Flows (Extensions):

1. **Room Allocation to a New Occupant:**
   - **Step 3A:** If the room is not available, the system prompts the Hostel Manager to either choose a different room or add the new occupant to a waiting list.
   - **Step 3B:** The Hostel Manager can view **pending allocations** and choose a room accordingly.
2. **Room Maintenance Required:** ○ **Step 5A:** If the room is in need of cleaning or repairs, the system notifies the **maintenance staff** to schedule the necessary tasks. ○ **Step 5B:** The maintenance staff updates the status once the tasks are completed, and the room is marked as **ready for occupancy**.
3. **Room Reallocation:**
   - **Step 3A:** If a student/employee requests to move to a different room, the system allows the Hostel Manager to **reassign the room** and automatically update the status of the old room to **vacant**.

Exception Flows:

1. **Room Not Found:**

- If the system cannot find the selected room (due to incorrect room number or system issues), the system alerts the Hostel Manager. ○ The Hostel Manager is prompted to verify the room number or try again with a different room.

2. **Room Overbooking:**
   - If a room has already been allocated to someone else (e.g., due to a system error), the system alerts the Hostel Manager and asks them to select a different room. 3. **Maintenance Issue Not Addressed:**
   - If the maintenance staff fails to complete their task in a timely manner (e.g., due to resource shortage), the system generates an **alert** for follow- up actions and escalates the issue to the Hostel Manager.

4. **User Permissions Error:** ○ If a user without appropriate permissions (e.g., non- admin staff) tries to manage room allocations, the system denies access and shows an **Access Denied** message.

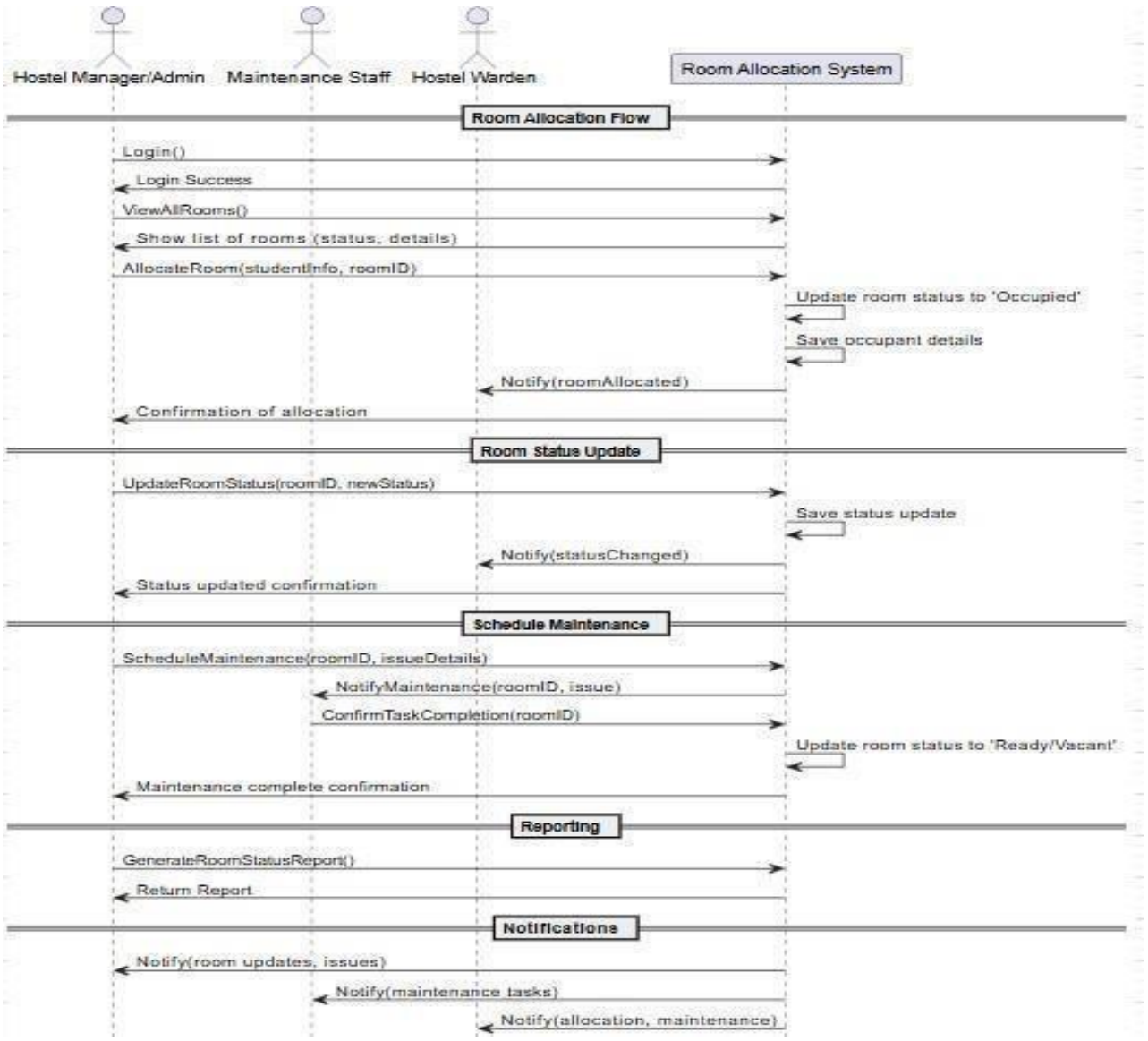
Trigger:

- The trigger for this use case is the **need to manage rooms** within the hostel, including allocating rooms, updating their status, and scheduling maintenance.

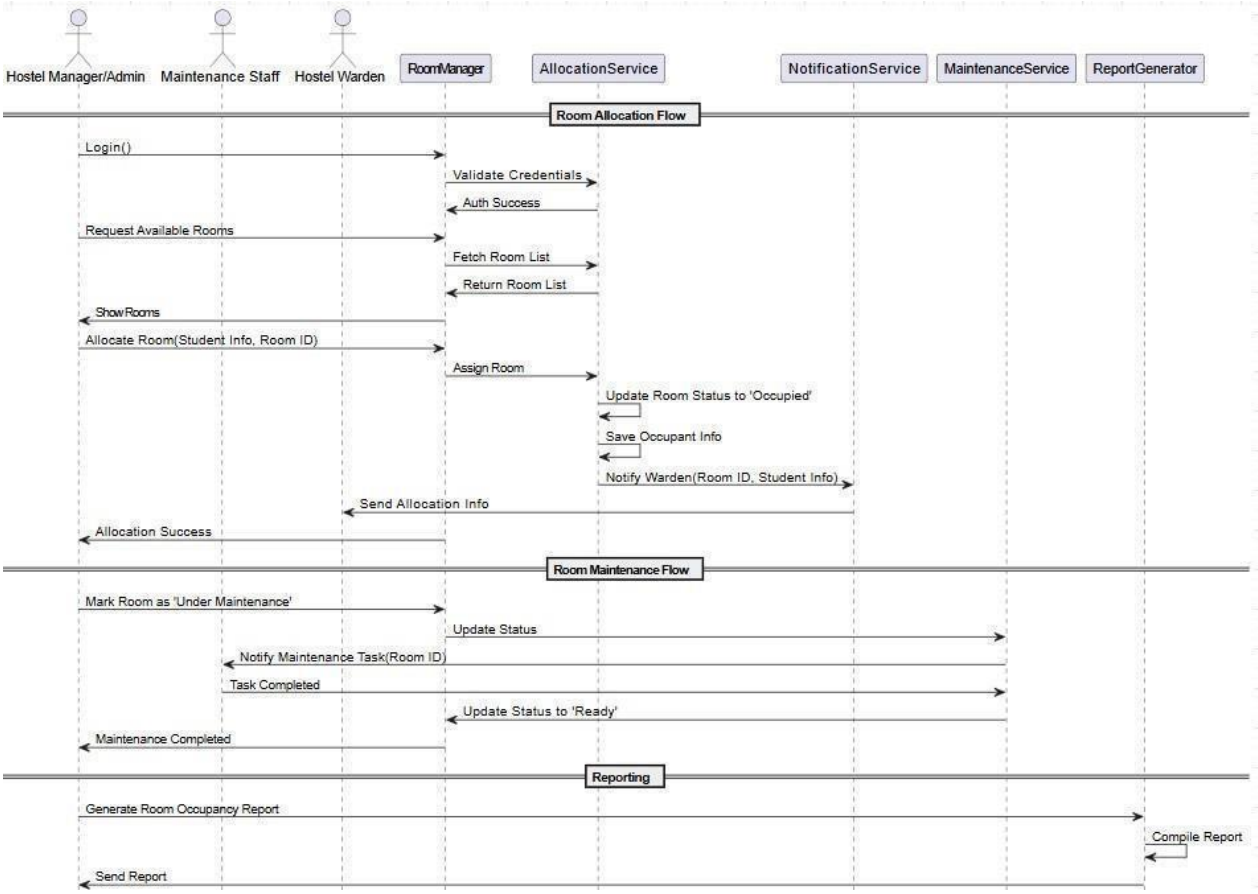
Special Requirements:

- **Data Security & Privacy:** Only authorized personnel (Hostel Manager, Admin) should have permission to modify room allocations and update room statuses.
- **Real-Time Updates:** The system must ensure that room status changes (vacancy, maintenance, allocation) are reflected in real time to avoid overbooking and confusion.
- **Maintenance Tracking:** The system should allow the **maintenance staff** to track progress on maintenance tasks and provide feedback on completed jobs.
- **Automated Notifications:** The system must send automated notifications to the concerned parties (student, employee, maintenance staff, and warden) whenever a room status is changed or allocated.
- **Reporting:** The system should support detailed reports on **room occupancy**, **maintenance schedules**, and **room availability**, which are important for auditing and operational analysis.
- **Scalability:** The system should be scalable to accommodate different room types, occupancy limits, and multi-building hostels.

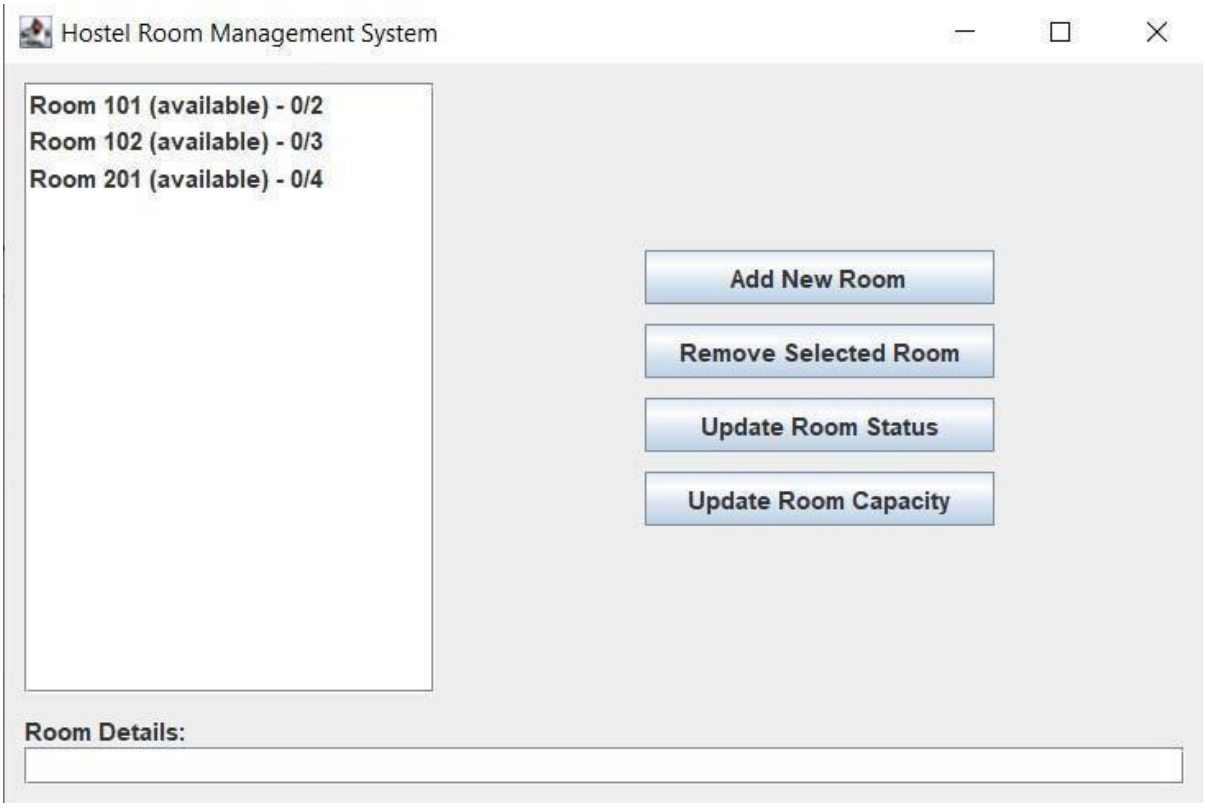
System Sequence Diagram:



Sequence Diagram:



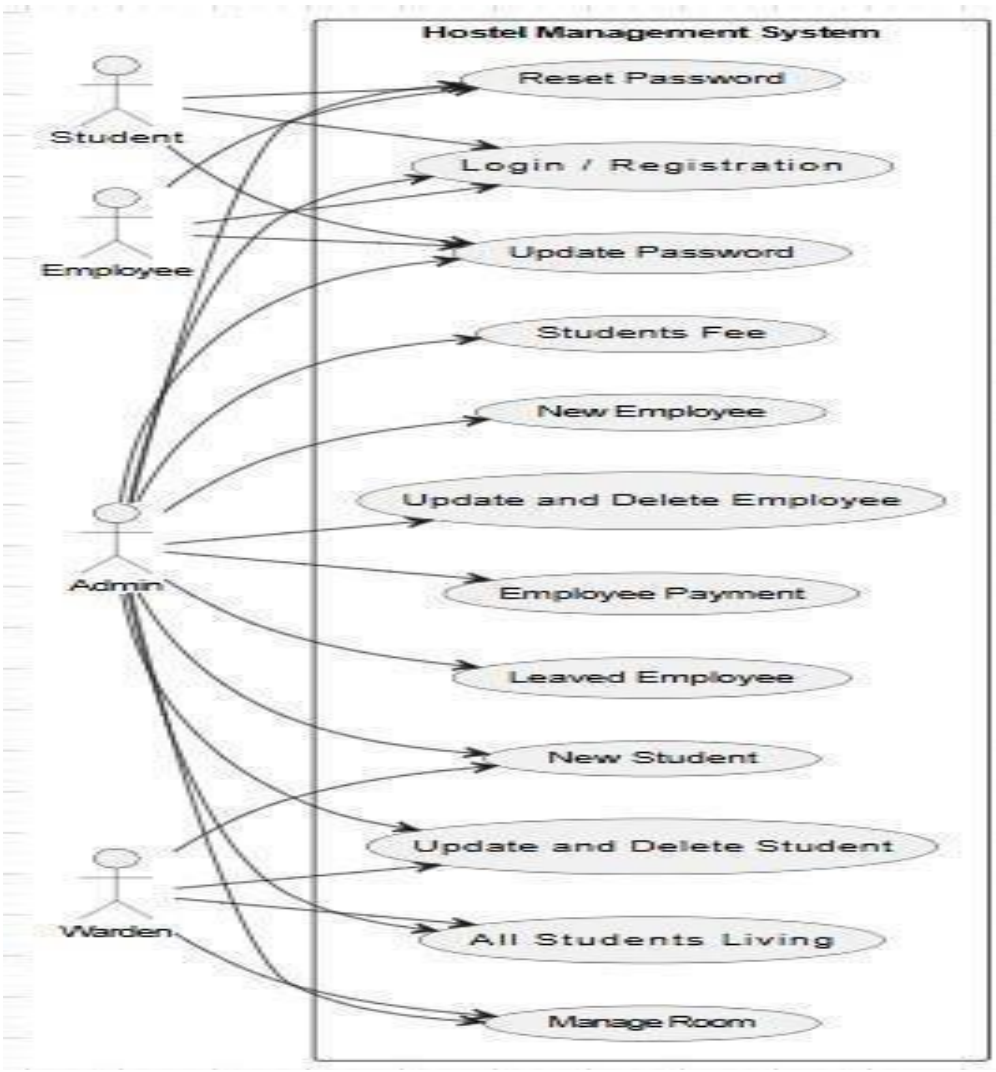
UI Prototype:



NAME: SARDAR ZAIN REGISTRATION NO : SP23-BSE-013

TASK:

DOCUMENTATION AND CODING FOR ONLY LOGIN USECASE



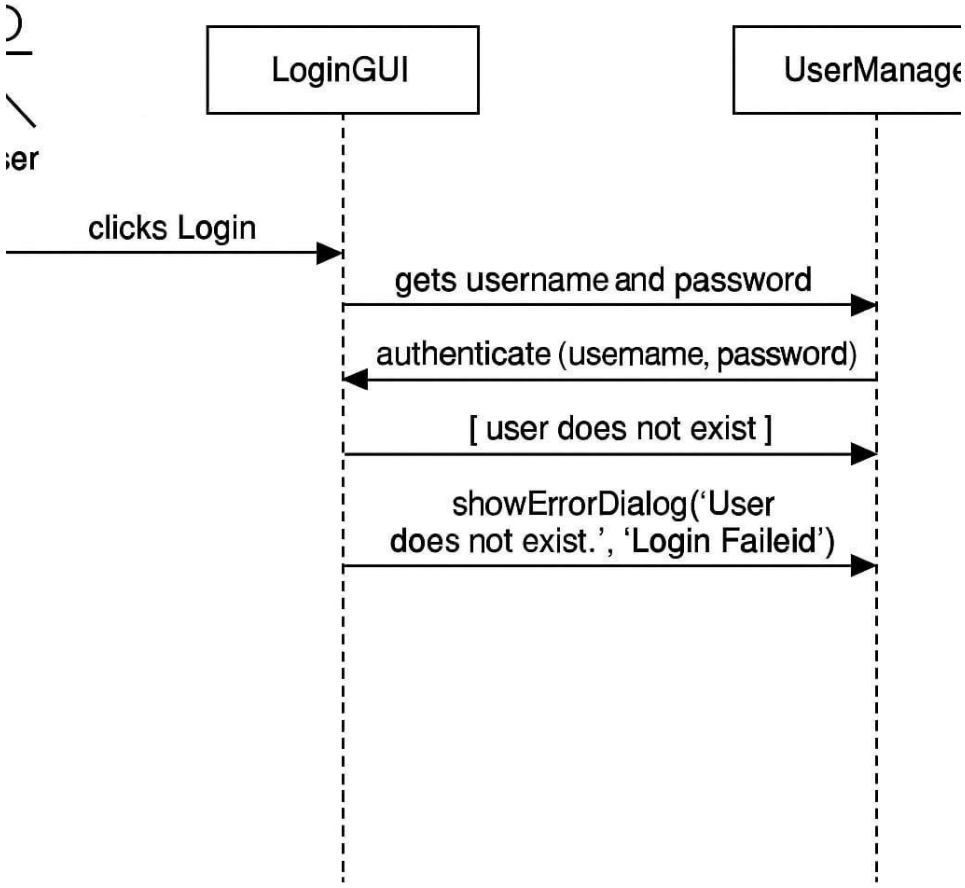
LOGIN USE CASE:

Fully Dressed Use Case – Login

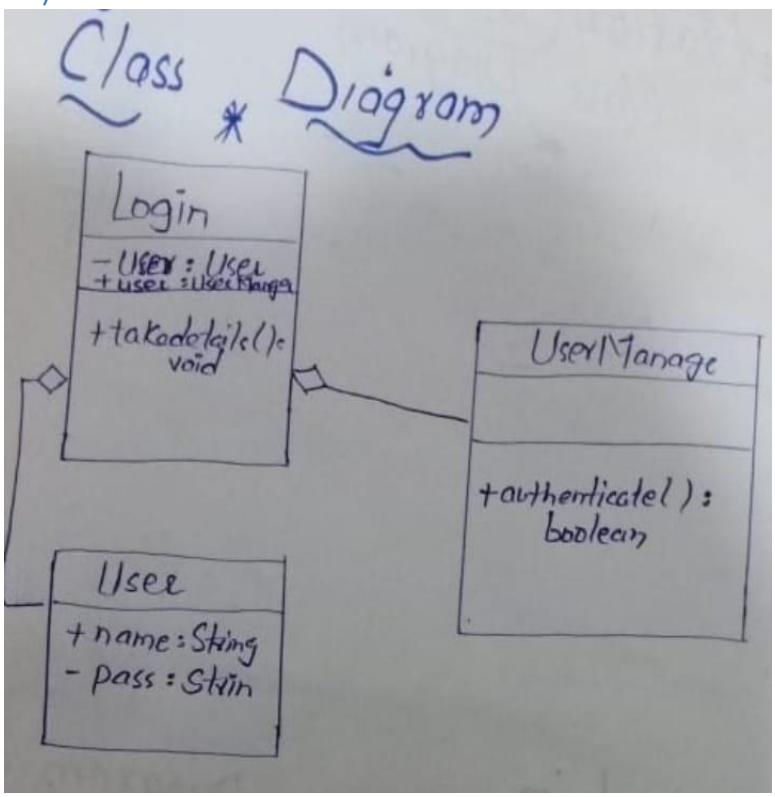
Use Case Name: User Login

Element	Description	Element	Description
Primary Actor	User		
Goal	Log into the Hostel Management System		
Preconditions	- User is registered - `users.txt` contains the user credentials		
Postconditions	- If credentials are valid, user is taken to Main Menu - If not, an error message is shown		
Main Success Scenario	1. User enters username and password 2. System checks if both fields are filled 3. `LoginGUI` calls `userManager.authenticate()` 4. If credentials match in `users.txt`, show Main Menu		
Extensions (Alternate Flows)	- 3a. If any field is empty → show "Username or password required" - 4a. If authentication fails → show "User does not exist"		
Exceptions	- File not found or read error in `UserManager` leads to login failure silently (could be improved)		
Priority	High		
Frequency of Use	Every user session		
Special Requirements	- Case-insensitive usernames - Case-sensitive passwords		

Login (SEQUENCE DIAGRAM)



Login (CLASS DIAGRAM)



REGISTRATION USE CASE:

Use Case Name: Register New

User

Scope:

Hostel Management System

Level:

User Goal

Primary Actor:

Unregistered User

Stakeholders and Interests:

- **User:** Wants to create a new account securely and easily.

Preconditions:

- The user is not already registered in the system (not present in `users.txt`).

Postconditions:

- A new user account is added to the file, allowing future login.

Main Success Scenario:

1. The user opens the registration window.
2. The user enters a username and password.
3. The user confirms the password.
4. The system checks that all fields are filled.
5. The system verifies that both passwords match.
6. The system calls the `UserManager` to store the user.
7. The user's credentials are saved in the file `users.txt`.
8. The system shows a success message.

Extensions (Alternative Flows):

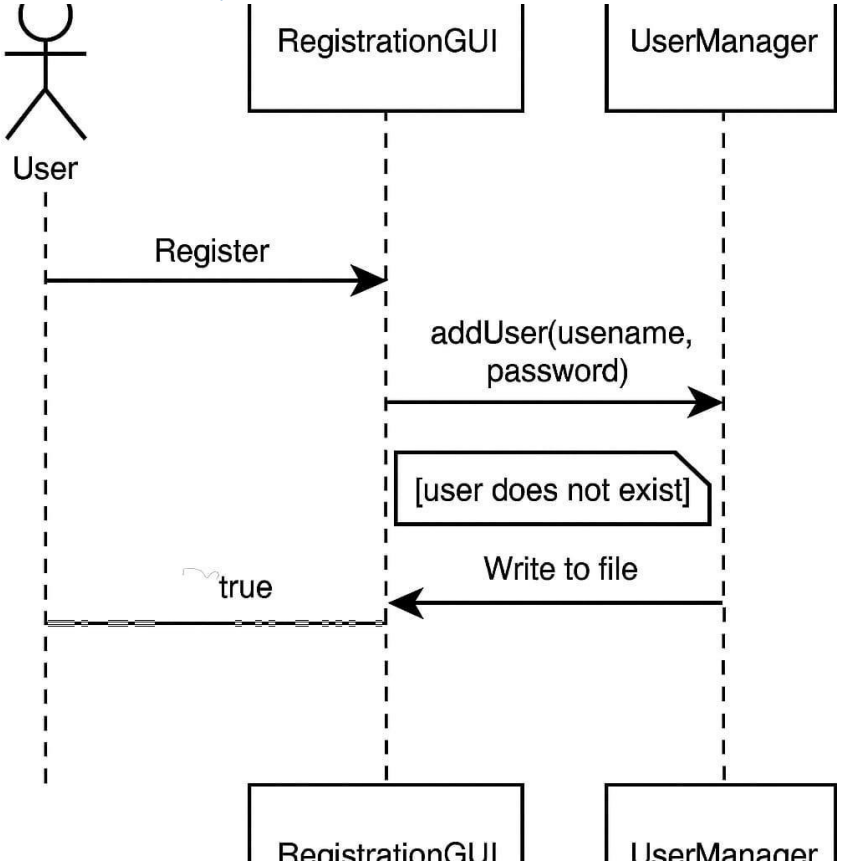
- If the username or password is empty, the system displays an error.
- If the passwords do not match, the system alerts the user.
- If an error occurs while saving the user (e.g., file issue), the system notifies the user.

Special Requirements:

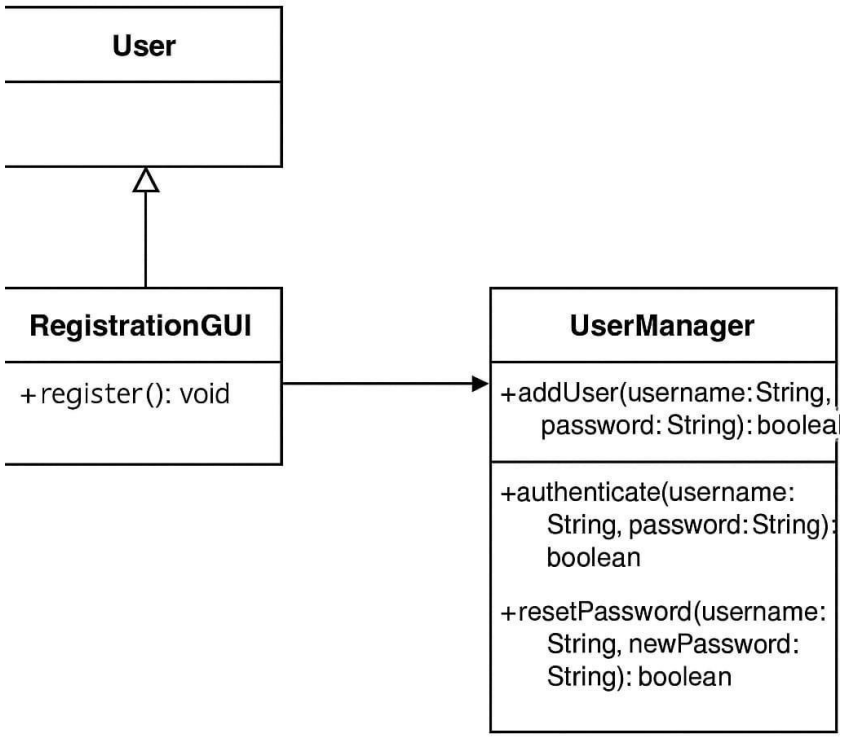
- Passwords are stored in plain text (for now) in the file `E:/users/users.txt`.
- Input validation must be performed before writing to the file.

- GUI design should not interfere with business logic.

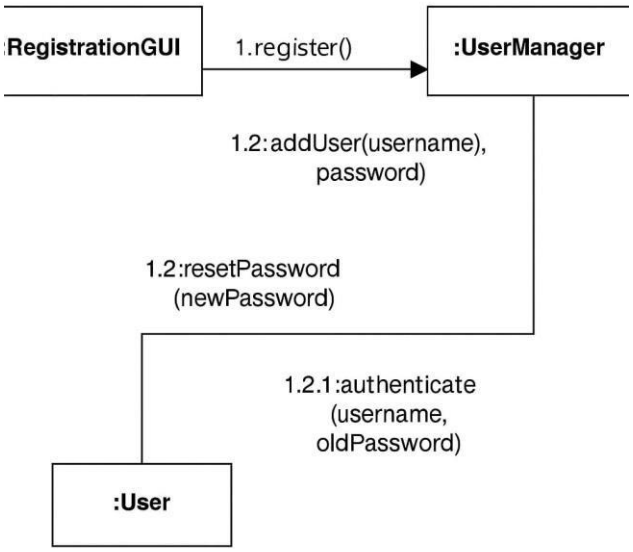
Registration (SEQUENCE DIAGRAM)



Registration (CLASS DIAGRAM)



Registration (Communication Diagram)



Abdullah Sajid Sp23-bse-019

Fully Dressed Use Case: Update Employee Details

Use Case Name: UC003 - Update Employee Details

Goal: To allow a Hostel Administrator to modify the existing details of an employee.

Scope: The Hostel Management System's Employee Management Module (EmployeeManagementGUI, EmployeeManager, Employee classes).

Level: User Goal

Primary Actor: Hostel Administrator

Stakeholders & Interests:

Hostel Administrator: Wants to easily correct or update employee information to maintain accurate records.

Employee: Wants their information to be up-to-date and accurate in the system.

Hostel Management System: Needs to ensure data integrity, especially preventing duplicate employee names and reflecting changes accurately in the displayed list.

Preconditions:

The EmployeeManagementGUI is open and visible.

The EmployeeManager instance is initialized and contains the list of employees.

At least one employee exists in the list to be selected for update.

Postconditions:

Successful Update: The selected employee's details (name, email, phone, experience) are updated in the EmployeeManager's internal list, the employeeList in the GUI is refreshed to reflect the changes, and a success message is displayed.

Failed Update (Validation/Conflict): The employee's details remain unchanged, and an appropriate error message is displayed to the Administrator.

Main Success Scenario:

Administrator opens Employee Management GUI.

The system (via EmployeeManagementGUI constructor) initializes the UI, fetches existing employees from EmployeeManager, and displays them in the employeeList.

Administrator selects an employee from the employeeList.

The JList's selection model registers the chosen Employee object.

Administrator clicks the "Update Selected" button.

System Action: EmployeeManagementGUI.UpdateEmployeeListener.actionPerformed() is invoked.

System Action: The system checks if employeeList.getSelectedValue() is null. (It is not, in this success scenario).

System Action: The system creates a JPanel with JTextFields for Name, Email, Phone, and Experience.

System Action: The current details of the selectedEmployee are used to pre-populate these JTextFields.

System Action: customizeOptionPaneUI() is called to apply custom styling to the JOptionPane.

System Action: A JOptionPane.showConfirmDialog() is displayed, presenting the pre-filled fields for editing.

Administrator modifies one or more fields in the dialog and clicks "OK".

System Action: resetOptionPaneUI() is called to revert UIManager properties.

System Action: The system retrieves the trimmed text from all JTextFields (newName, newEmail, newPhone, newExperience).

System Action: The system performs an input validation check for empty fields (newName.isEmpty() || newEmail.isEmpty() || ...). (In this success scenario, all fields are non-empty).

System Action: employeeManager.updateEmployee(selectedEmployee.getName(), newName, newEmail, newPhone, newExperience) is called.

System Action: Inside EmployeeManager.updateEmployee():

The manager finds the employee by originalName (selectedEmployee.getName()).

It checks if newName is different from originalName. (If it is, it performs a check to ensure newName does not conflict with another existing employee's name using employees.stream().anyMatch(...)). (In this success scenario, there is no conflict).

The setName(), setEmail(), setPhone(), setExperience() methods of the found Employee object are called to update its attributes.

EmployeeManager.updateEmployee() returns true.

System Action: Back in EmployeeManagementGUI.UpdateEmployeeListener, since the update was successful, refreshEmployeeList() is called.

employeeListModel.clear() removes all items from the JList.

employeeManager.getAllEmployees() retrieves the updated list of employees.

employeeListModel.addElement(emp) re-adds all employees, including the updated one, to the JList.

System Action: showInfoDialog("Employee details updated successfully!", "Success") is displayed to the Administrator.

Extensions (Alternative Flows):

2a. Administrator clicks "Update Selected" but no employee is selected.

System Action: The selectedEmployee check (employeeList.getSelectedValue()) returns null.

System Action: showErrorDialog("Please select an employee to update.", "Selection Error") is displayed.

End Use Case.

4a. Administrator clicks "Cancel" in the update dialog.

System Action: JOptionPane.showConfirmDialog() returns JOptionPane.CANCEL_OPTION (or CLOSED_OPTION).

System Action: resetOptionPaneUI() is called.

End Use Case. (No changes made, no message displayed).

4b. Administrator leaves one or more required fields empty in the update dialog and clicks "OK".

System Action: The input validation check (newName.isEmpty() || ...) evaluates to true.

System Action: showErrorDialog("All fields are required.", "Input Error") is displayed.

Loop: The system returns to Step 4, allowing the Administrator to correct the input.

4c. Administrator changes the employee's name to a name that already exists for another employee.

System Action: The system retrieves the trimmed text from all JTextFields.

System Action: employeeManager.updateEmployee() is called.

System Action: Inside EmployeeManager.updateEmployee(), the duplicate name check (employees.stream().anyMatch(...)) identifies a conflict.

System Action: EmployeeManager.updateEmployee() returns false.

System Action: Back in EmployeeManagementGUI.UpdateEmployeeListener, the else block is executed.

System Action: showErrorDialog("Failed to update employee. Check if the new name already exists for a different employee.", "Update Failed") is displayed.

Loop: The system returns to Step 4, allowing the Administrator to correct the input.

Technology & Data:

User Interface: Java Swing (JFrame, JPanel, JList, JScrollPane, JButton, JLabel, JTextField, JOptionPane). Custom painting (paintComponent, GradientPaint, RoundRectangle2D.Double) and UIManager manipulation are used for aesthetics.

Data Model: Employee class (POJO for employee attributes).

Business Logic: EmployeeManager class (manages the List<Employee>, handles adding, removing, and updating employees, including uniqueness checks).

Data Flow:

EmployeeManagementGUI receives user input from JTextFields.

EmployeeManagementGUI passes input to EmployeeManager.updateEmployee().

EmployeeManager modifies the Employee object(s) within its internal List<Employee>.

EmployeeManager provides the updated list to EmployeeManagementGUI via getAllEmployees().

EmployeeManagementGUI updates its DefaultListModel and JList to reflect changes.

JOptionPane is used for user input and feedback.

fully dressed use case of update student:

Use Case: Update Hostel Student Name

This use case describes how a Hostel Administrator or Hostel Staff Member modifies an existing student's name within the Hostel Student Management System.

1. Use Case Name

Update Hostel Student Name

2. Goal

To change the name of an existing student record to a new, accurate name within the hostel management system.

3. Actors

Primary Actor: Hostel Administrator / Hostel Staff Member

4. Preconditions

The Hostel Student Management GUI application is launched and visible.

The system has an active connection to the `StudentManager` to access and modify student data.

At least one student record exists in the system.

5. Postconditions

Success: The selected student's name is updated in the system, and the change is reflected in the displayed student list.

Failure:The student's name remains unchanged, and an appropriate error message is displayed to the Administrator.

6. Main Success Scenario (Basic Flow

1. The Hostel Administrator views the list of students in the Hostel Student Management GUI.
2. The Administrator identifies the student whose name needs to be updated (e.g., "Ali Raza").
3. The Administrator clicks on "Ali Raza" in the student list to select it.
4. The Administrator clicks the "Update Student button.
5. A dialog box titled "Update Student: Ali Raza" appears. The input field inside this dialog is pre-filled with the current

name, "Ali Raza".

6. The Administrator deletes the old name and enters the new, desired student name(e.g., "Ali Raza Khan").

7. The Administrator clicks the "OK" button in the dialog.

8. The system successfully updates the student's name from "Ali Raza" to "Ali Raza Khan" in its internal records.

9. The system automatically updates the displayed student list, showing "Ali Raza Khan" in place of the old name.

10. The system displays a success message: "Student 'Ali Raza' updated to 'Ali Raza Khan' successfully."

7. Extensions (Alternate Flows/Error Conditions)

7a. No Student Selected:

1. Steps 1-2 (Main Success Scenario).

2. The Administrator clicks the "Update Student" button without selecting any student from the list.

3. The system displays an error message: "Please select a student to update."

4. The action is aborted, and no update dialog appears.

7b. New Student Name is Empty:

1. Steps 1-5 (Main Success Scenario).

2. The Administrator clears the pre-filled name in the dialog or enters only spaces, then clicks "OK".

3. The system displays an error message: "New student name cannot be empty."

4. The dialog remains open, allowing the Administrator to enter a valid name or click "Cancel".

7c. New Student Name Already Exists (for another student):

1. Steps 1-5 (Main Success Scenario).

2. The Administrator enters a new name (e.g., "Fatima Khan") that is already assigned to a different existing student in the list.

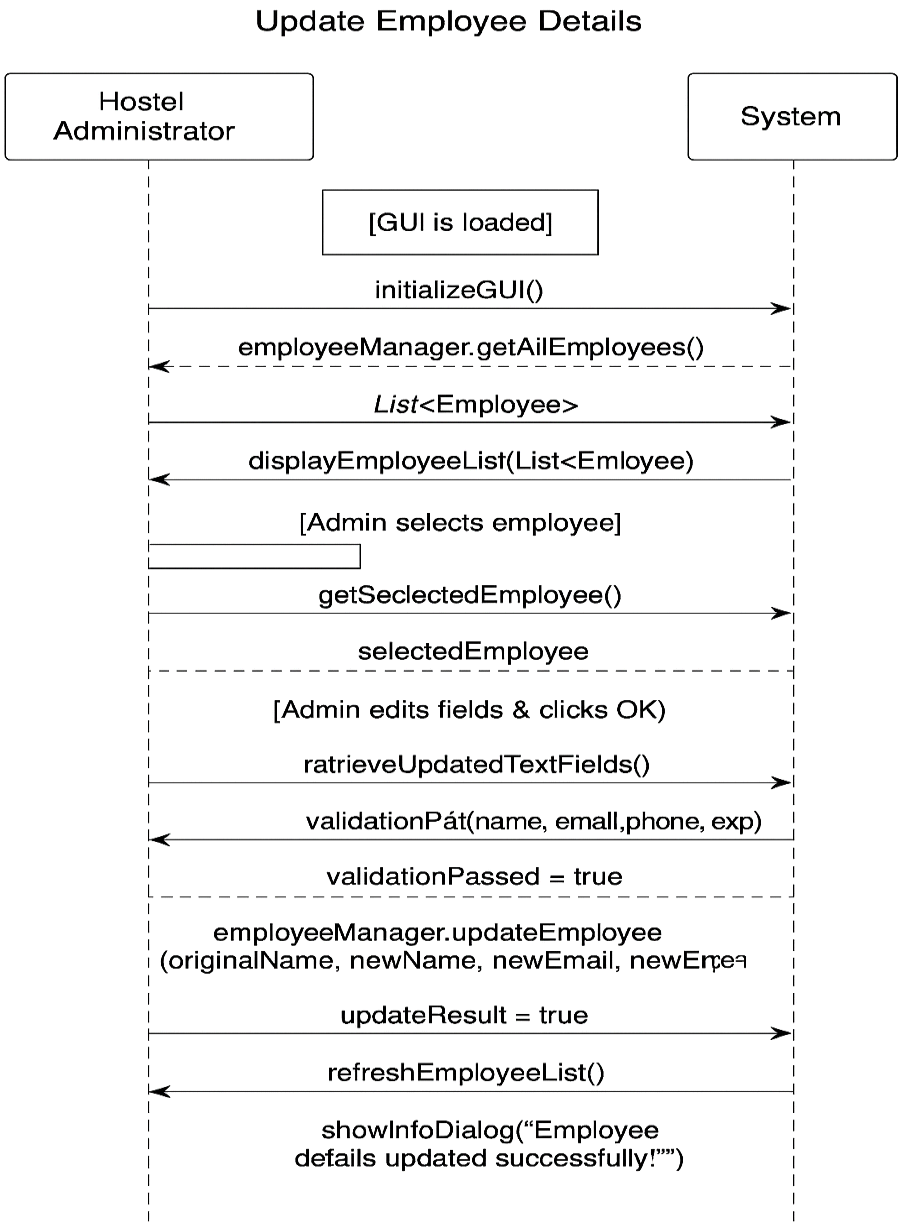
3. The system displays an error message: "Failed to update student '[old name]'. New name might be a duplicate or invalid."

4. The dialog remains open, allowing the Administrator to enter a unique name or click "Cancel".

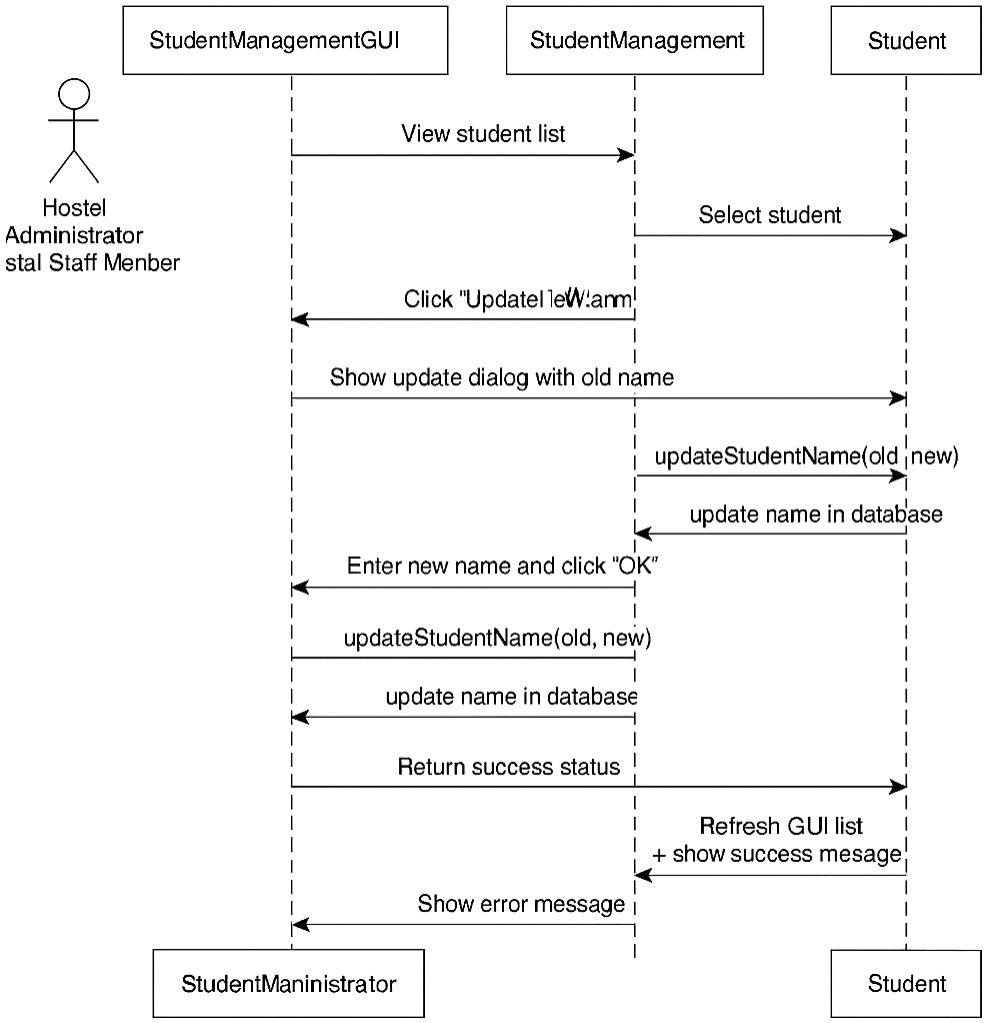
7d. Update Operation Cancelled by Administrator

1. Steps 1-5 (Main Success Scenario).
2. The Administrator clicks the "Cancel" button in the "Update Student" dialog.
3. The system takes no action.
4. The student's name in the list remains unchanged.

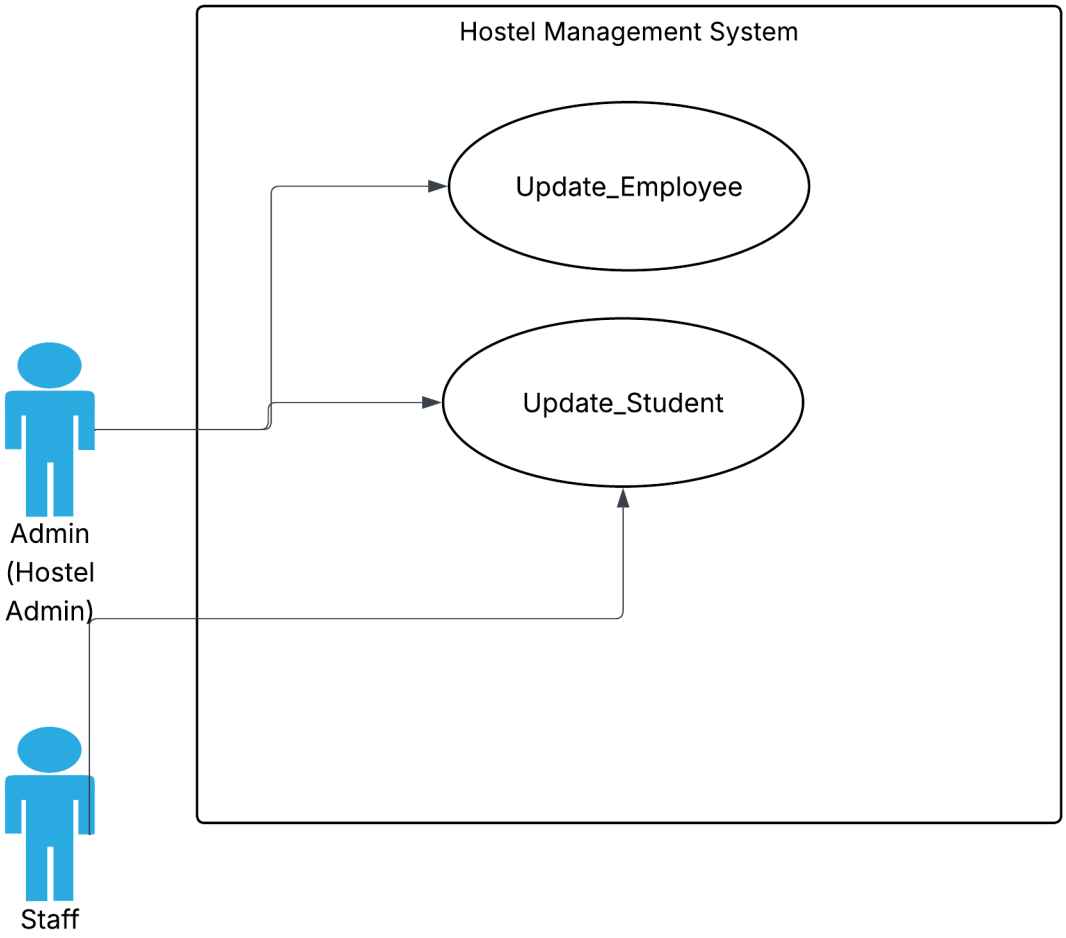
System seqeunce diagram for update employees:



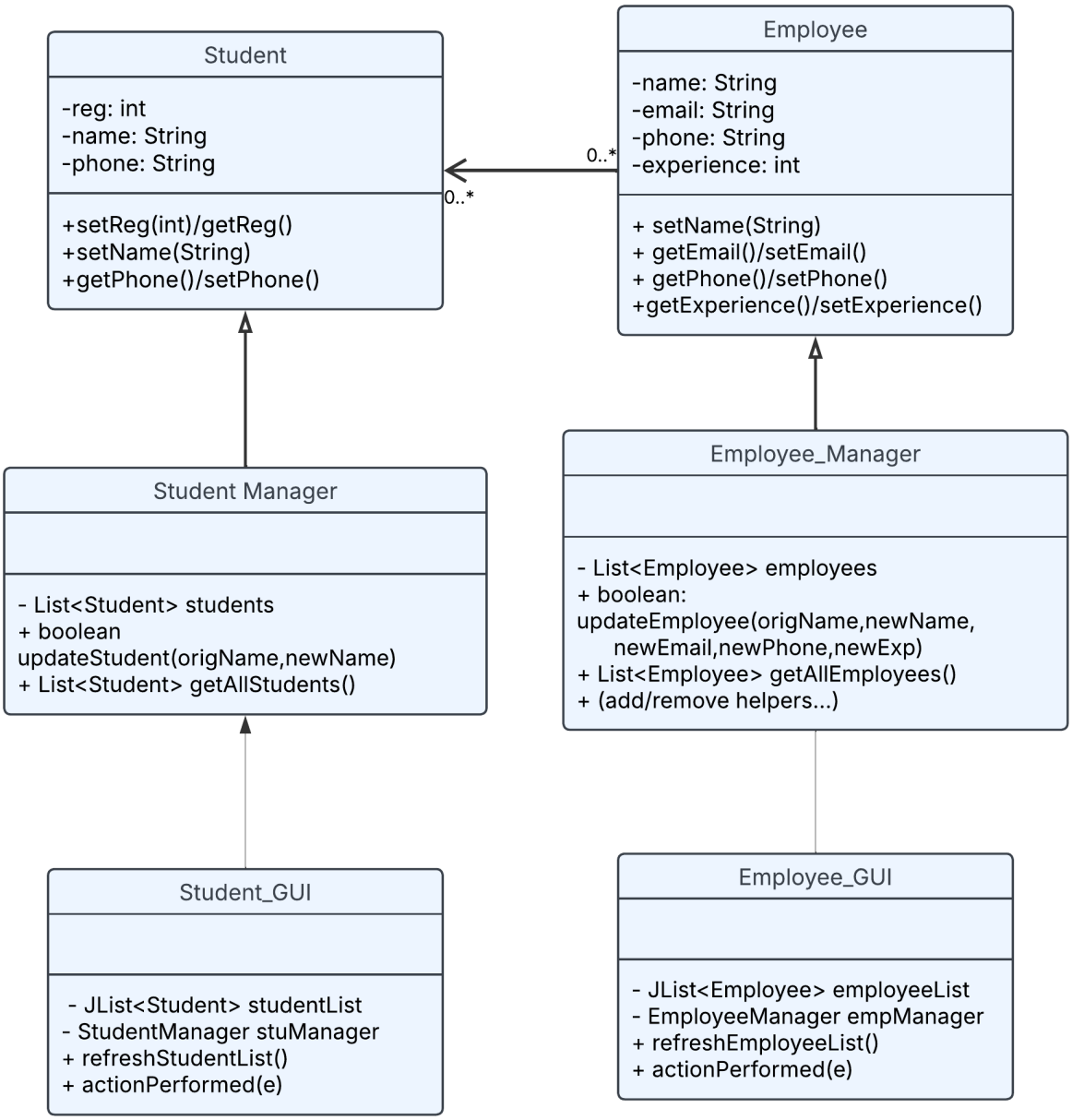
System sequence diagram of update student name:



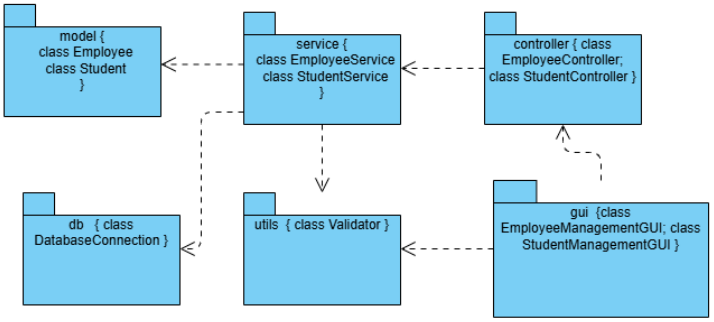
UseCase Diagram:



Class Diagram:



Package Diagram:



Daniyal Murtaza SP23-BSE-001

Use Case: Reset Password

Use Case Name	Reset Password
Primary Actor	Hostel Management System User (e.g., Student, Staff, Admin)
Stakeholders and Interests	- **Users**: Want a secure and easy way to regain access to their accounts if they forget their password. - **System Admin**: Wants to ensure password reset process is secure to prevent unauthorized access.
Preconditions	- User must have an existing account in the system. - User must have provided a valid email or phone number during registration.
Postconditions	- The user's password is updated and they can log in using the new password.
Trigger	User clicks on "Forgot Password" link on the login page.

Main Success Scenario (Basic Flow)

1. **User** clicks on "Forgot Password?" on the login screen.
2. **System** prompts user to enter their registered email or phone number.
3. **User** enters the email/phone number and submits the form.
4. **System** validates the input and checks if it is associated with a registered account. ○ **Success**: The email/phone number is registered in the system.
5. **System** generates a password reset token or OTP (One-Time Password) and sends it to the user's email or phone number.
6. **User** receives the token/OTP and enters it on the password reset screen.
7. **System** verifies the token/OTP.
    - **Success**: The token/OTP is valid and not expired.
8. **System** prompts the user to enter a new password and confirm it.
9. **User** enters and confirms the new password.
10. **System** validates the new password format (e.g., length, complexity).
    - **Success**: Password meets the complexity requirements.
11. **System** updates the user's password in the database.
12. **System** displays a success message and redirects the user to the login page.

Alternate Flows (Alternate Scenarios)

4a. Invalid email/phone number entered

- **Step 4a1**: System displays an error message: "No account found with this email/phone." • **Step 4a2**: User is prompted to try again or contact support.
    - **Alternative**: User may choose to go back to the login screen and try again or request additional help.

6a. Invalid or expired token/OTP

- **Step 6a1**: System displays an error message: "Invalid or expired token." • **Step 6a2**: User can request a new token/OTP.
    - **Alternative**: User may need to re-enter their email or phone number to receive a new token/OTP.

10a. Passwords do not match or do not meet complexity rules

- **Step 10a1**: System displays an error message: “Passwords do not match” or “Password must contain at least 8 characters, a number, and a symbol.”
- **Step 10a2**: User is prompted to re-enter the new password and confirm it. ○ **Alternative**: If the user forgets the complexity rules, the system can display the exact criteria for password strength.

Additional Success Case Scenarios and their Alternatives

Success Case Scenario 1: Password reset completed successfully and logged in immediately

- **Step 12**: After resetting the password, the system automatically logs the user in with their new credentials.
- **Alternative Case**:
  - If the auto-login fails (e.g., incorrect password entered or session issues), the system redirects the user to the login screen with an appropriate message:

    “Password reset successful, please log in with your new password.”

Success Case Scenario 2: User chooses to reset password via email

- **Step 5**: The user receives an email with a password reset link.
  - **Success**: User clicks the link and is redirected to the password reset form.
  - **Alternative**: If the email fails to arrive or gets delayed, the user can manually click "Resend Link" to receive a new reset link.
- **Step 6**: User enters the token received from the email and follows the steps outlined in the main success scenario.

Success Case Scenario 3: User resets password via phone OTP

- **Step 5**: The system sends an OTP via SMS to the registered phone number.
  - **Success**: The user receives the OTP and enters it correctly. ○ **Alternative**: If the user does not receive the SMS, they can choose to re-request the OTP or use an alternate method (e.g., email).

Special Requirements

- Reset link/token should expire within a specified timeframe (e.g., 15 minutes).

- Passwords must follow security standards (e.g., min. 8 characters, upper/lowercase, number, symbol).
- All sensitive data (tokens, passwords) should be transmitted securely using HTTPS and stored securely (e.g., hashed passwords).
- Option to limit the number of reset attempts to prevent brute force attacks.

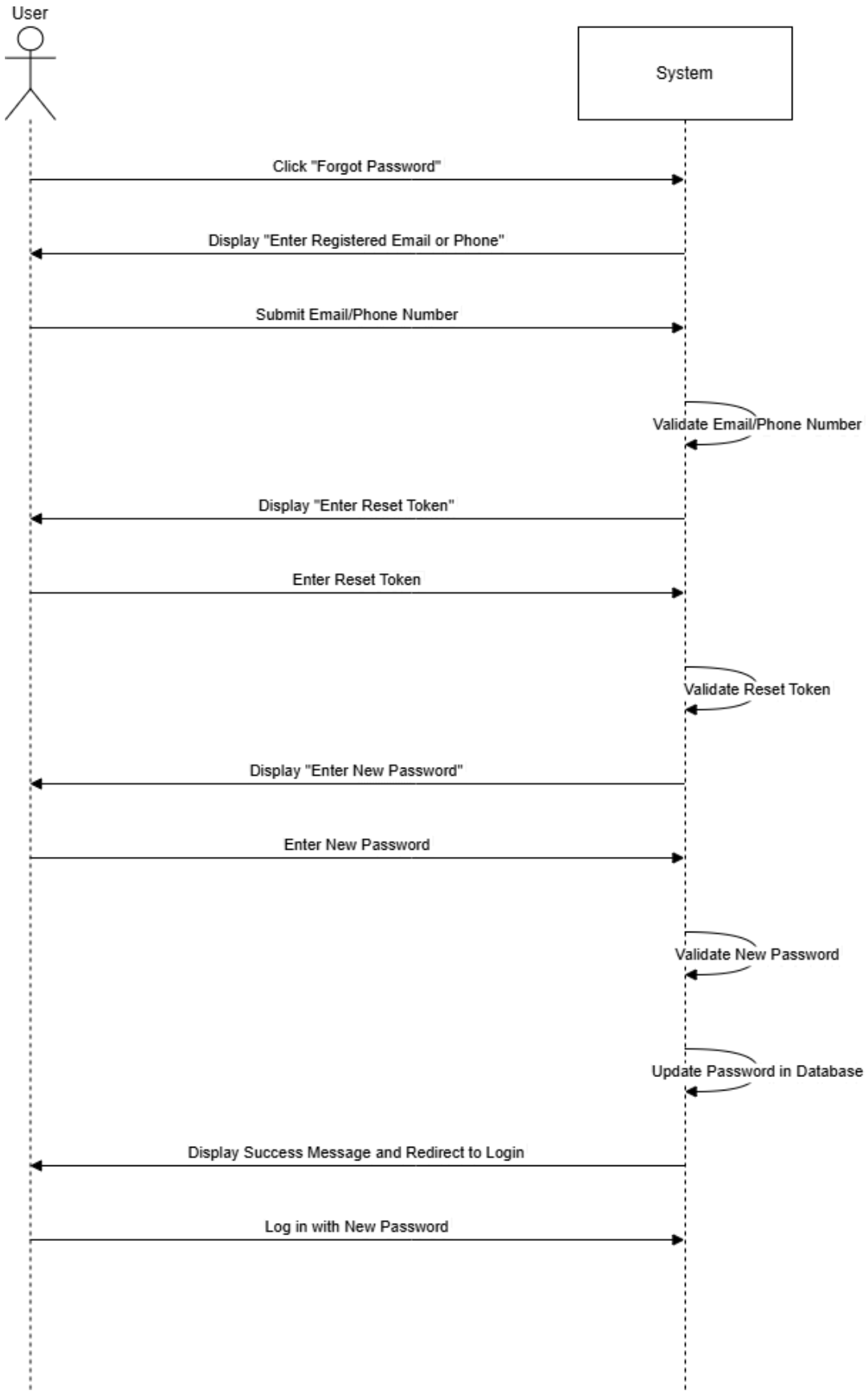
Frequency of Use

- Occasional: Typically when a user forgets their password or wants to update it for security.

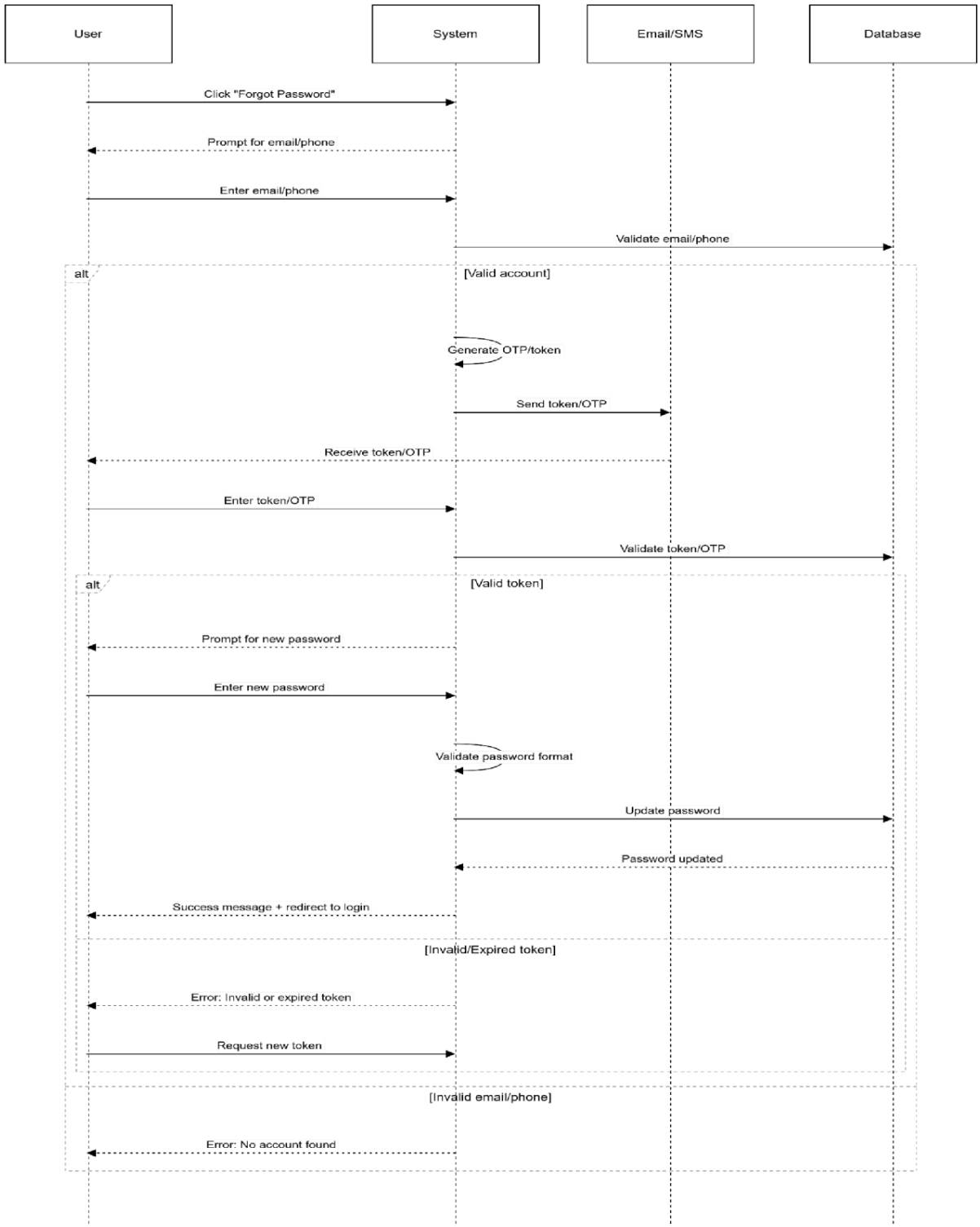
Open Issues

- Should the system allow password reset using both email and phone?
- Should the user be notified via email/phone after a successful reset for security awareness?

SSD For Reset Use case:


User
System
Click "Forgot Password"
Display "Enter Registered Email or Phone"
Submit Email/Phone Number
Validate Email/Phone Number
Display "Enter Reset Token"
Enter Reset Token
Validate Reset Token
Display "Enter New Password"
Enter New Password
Validate New Password
Update Password in Database
Display Success Message and Redirect to Login
Log in with New Password

SDD:



NAME **KAZIM SHAUKAT**

REG NO **SP23-BSE-024**

A .STUDENT FEE MANAGEMENT

Use Case ID: UC_Admin_001

Use Case Name: Manage Student Fees

Primary Actor: Administrator

Goal: To efficiently manage all aspects of student fees within the hostel room management system, including setting up fee structures, recording payments, tracking dues, applying penalties, and generating relevant reports.

Preconditions:

1. The Administrator is logged into the Hostel Room Management System with appropriate permissions.
2. Student records and their assigned rooms are already present in the system.
3. Fee structures for the current academic period (or relevant duration) have been defined in the system (refer to UC_Admin_002: Set Up Fee Structures).
4. Payment gateways (if applicable for online payments) are configured (refer to UC_Admin_003: Configure Payment Gateways).

Postconditions:

- Student fee records are accurately updated based on payments, penalties, waivers, and adjustments.
- Financial reports reflecting fee status and collection are available.
- Students receive appropriate notifications regarding their fee status.
- The system maintains an audit trail of all fee-related administrative actions.

Main Flow:

1. **Administrator Accesses Fee Management Module:** The Administrator navigates to the "Fee Management" section of the system.
2. **Administrator Selects Action:** The system presents the Administrator with various options for managing student fees, including:

   ○ View Student Fee Status ○

   Record Payment (Offline) ○

   Apply Late Fee ○ Manage

   Waivers/Discounts ○ Adjust

   Fee Balance ○ Generate Fee

   Reports ○ Send Fee

   Notifications

3. **Scenario A: View Student Fee Status:**
    - 3a.1. The Administrator searches for a specific student by ID, name, room number, or other relevant criteria.
    - 3a.2. The system displays the student's fee details, including:
        - ✦ Total fee due for the current period.
        - ✦ Amount paid to date. ✦ Outstanding balance.
        - ✦ Payment history with dates and methods.
        - ✦ Applied waivers or discounts.
        - ✦ Any applicable late fees.
        - ✦ Due date for the next payment (if installments are enabled).
    - 3a.3. The Administrator may view more detailed information about specific payments or adjustments.
    - 3a.4. The Administrator can optionally export the student's fee statement.
4. **Scenario B: Record Payment (Offline):**
    - 3b.1. The Administrator selects the "Record Payment" option. ○ 3b.2. The Administrator searches for the student who made the offline payment.
    - 3b.3. The Administrator enters the payment details:
        - ✦ Amount paid.
        - ✦ Payment method (e.g., Cash, Cheque, Bank Transfer).
        - ✦ Date of payment.
        - ✦ Reference number (if applicable, e.g., cheque number). ✦ Optional notes.
    - 3b.4. The Administrator confirms the payment details. ○ 3b.5. The system updates the student's fee status and payment history.
    - 3b.6. The system generates a record of the offline transaction in the audit log.
5. **Scenario C: Apply Late Fee:**
    - 3c.1. The Administrator selects the "Apply Late Fee" option.
    - 3c.2. The Administrator can either:
        - ✦ Select individual students who have overdue fees.
        - ✦ Filter students based on overdue status and a specific date range.
    - 3c.3. The system automatically calculates the late fee amount based on the defined late fee policy (refer to UC_Admin_004: Define Fee Payment Policies). ○ 3c.4. The Administrator reviews the list of students and the calculated late fees. ○

      3c.5. The Administrator confirms the application of late fees. ○ 3c.6. The

system updates the outstanding balance for the affected students and records the late fee application in their fee details and the audit log. ○ 3c.7. The system may automatically send late fee notifications to the affected students (if configured).

6. **Scenario D: Manage Waivers/Discounts:**
    - 3d.1. The Administrator selects the "Manage Waivers/Discounts" option.
    - 3d.2. The Administrator searches for the student to whom a waiver or discount needs to be applied. ○ 3d.3. The Administrator selects the type of waiver/discount and enters the relevant details (e.g., percentage, fixed amount, duration, reason).
    - 3d.4. The Administrator confirms the application of the waiver/discount.
    - 3d.5. The system updates the student's outstanding balance and records the waiver/discount in their fee details and the audit log.
7. **Scenario E: Adjust Fee Balance:**
    - 3e.1. The Administrator selects the "Adjust Fee Balance" option. ○ 3e.2. The Administrator searches for the student whose fee balance needs adjustment. ○ 3e.3. The Administrator enters the adjustment amount (positive or negative) and a mandatory reason for the adjustment.
    - 3e.4. The Administrator confirms the adjustment. ○ 3e.5. The system updates the student's outstanding balance and records the adjustment with the reason in their fee details and the audit log.
8. **Scenario F: Generate Fee Reports:**
    - 3f.1. The Administrator selects the "Generate Fee Reports" option.
    - 3f.2. The system presents various report options (e.g., Outstanding Fees, Payment Summary, Fee Collection by Date, Overdue Fees). ○ 3f.3. The Administrator selects the desired report type and specifies any necessary filters (e.g., date range, block, room type, fee status).
    - 3f.4. The system generates the report and displays it to the Administrator (e.g., in a table, chart). ○ 3f.5. The Administrator can optionally export the report in various formats (e.g., CSV, PDF).
9. **Scenario G: Send Fee Notifications:**
    - 3g.1. The Administrator selects the "Send Fee Notifications" option.
    - 3g.2. The Administrator can choose to send notifications to:
        - ✦ Individual students (by searching).
        - ✦ Groups of students (e.g., all students with overdue fees, students in a specific block).
    - 3g.3. The Administrator selects the type of notification (e.g., Payment Reminder, Overdue Fee Notice, Fee Policy Update). ○ 3g.4. The Administrator composes the notification message (the system may provide templates). ○ 3g.5. The

Administrator selects the delivery method (e.g., Email, SMS, In-app notification).

- 3g.6. The Administrator sends the notifications.
- 3g.7. The system records the sent notifications in a communication log.

10. **Administrator Logs Out:** The Administrator logs out of the system.

Alternative Flows:

- **A1: No Student Found (in any search scenario):**
  - The system displays an error message indicating that no matching student was found.
  - The Administrator can refine their search criteria or create a new student record (if necessary and permitted).
- **B1: Invalid Payment Details:**
  - If the Administrator enters invalid payment details (e.g., non-numeric amount), the system displays an error message prompting them to correct the input.
- **C1: No Late Fee Policy Defined:**
  - If no late fee policy is defined in the system, the system displays a message indicating that late fees cannot be applied. The Administrator needs to define the policy first.
- **D1: Invalid Waiver/Discount Details:**
  - If the Administrator enters invalid waiver/discount details (e.g., percentage outside the valid range), the system displays an error message.
- **E1: Invalid Adjustment Amount:**
  - If the Administrator enters a non-numeric adjustment amount, the system displays an error message. • **F1: No Data for Report:** ○ If there is no data matching the selected report criteria, the system displays a message indicating that no records were found.
- **G1: Invalid Notification Details:**
  - If the Administrator enters an invalid email address or phone number for a student, the system may display a warning.

Exceptions:

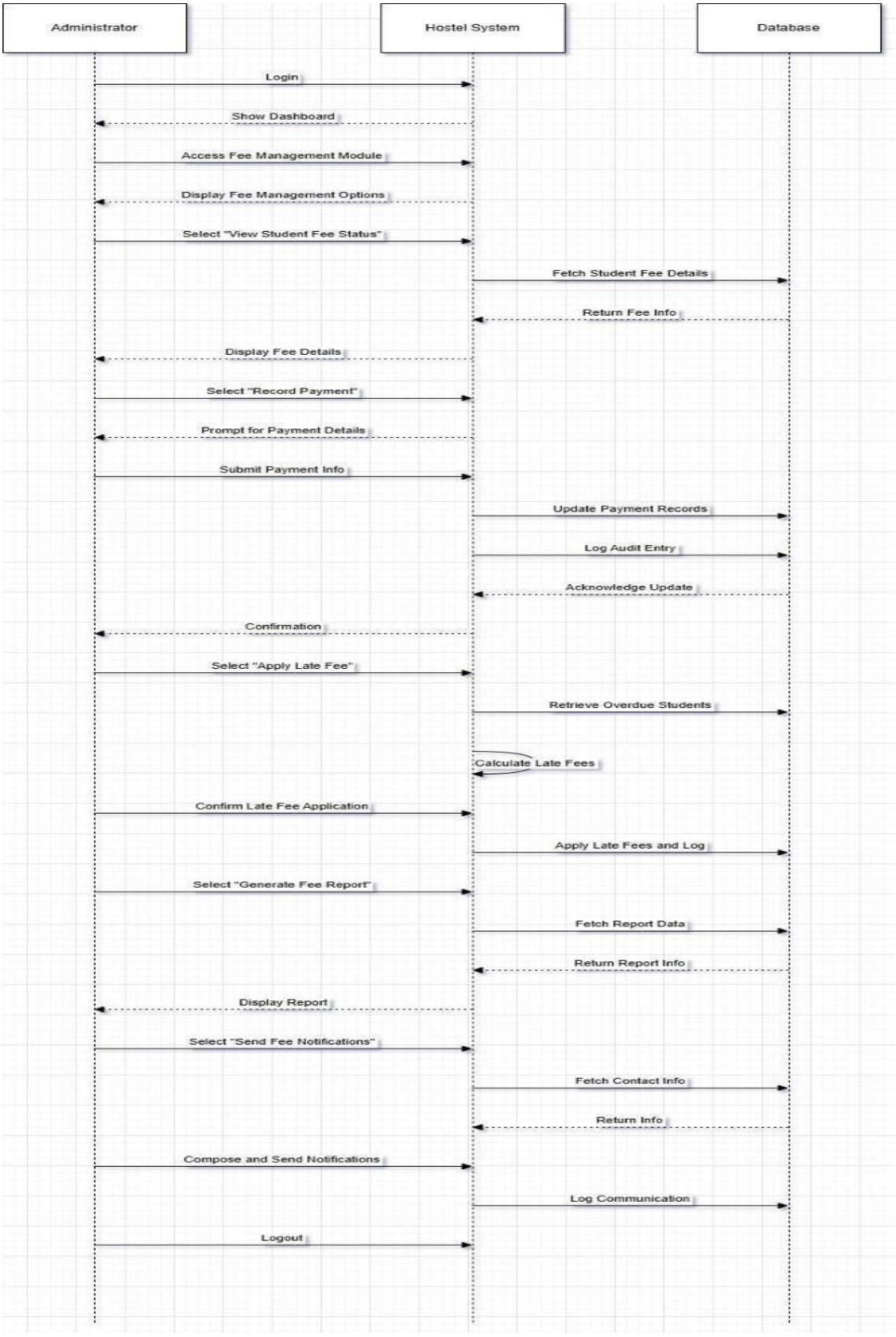
- **System Errors:** If the system encounters any technical errors during the process (e.g., database connection issues), it will display an appropriate error message to the Administrator.
- **Insufficient Permissions:** If the Administrator does not have the necessary permissions to perform a specific action, the system will display an authorization error.

Extension Points:

- Integration with accounting software for automated financial reconciliation.
- Implementation of automated installment reminders for students.
- Features for generating invoices or fee receipts for students.
- Workflow for handling disputed fee payments.

- Integration with student information systems (SIS) for automatic student data synchronization.

This fully dressed use case provides a detailed description of how an administrator manages student fees within the hostel room management system, covering various aspects of the fee management process.



B. Fully Dressed Use Case: Manage Employee Payments (Administrator)

Use Case ID: UC_Admin_002

Use Case Name: Manage Employee Payments

Primary Actor: Administrator

Goal: To efficiently manage all aspects of employee payments within the hostel room management system, including setting up payment structures, recording payments, processing salaries, managing deductions, and generating payroll reports.

Preconditions:

1. The Administrator is logged into the Hostel Room Management System with appropriate payroll management permissions.
2. Employee records with relevant details (name, employee ID, designation, salary structure, bank details) are already present in the system.
3. Payment structures (salary components, pay scales) have been defined in the system (refer to UC_Admin_003: Set Up Employee Payment Structures).
4. Bank integration (if applicable for direct transfers) is configured (refer to UC_Admin_004: Configure Bank Integration).
5. Attendance and leave data for the payment period are finalized in the system (if integrated for payroll calculation).

Postconditions:

- Employee payment records are accurately generated and updated.
- Salaries are processed and disbursed to employees (either recorded as paid offline or initiated for online transfer).
- Payroll reports for the specified period are available.
- Employees may receive payment slips or notifications.
- The system maintains an audit trail of all employee payment-related administrative actions.

Main Flow:

1. **Administrator Accesses Employee Payment Module:** The Administrator navigates to the "Employee Payment" or "Payroll" section of the system.
2. **Administrator Selects Action:** The system presents the Administrator with various options for managing employee payments, including:

○ Process Payroll for Period ○ Record Offline Payment ○ Manage Deductions/Allowances ○ View Payment History (Employee-wise) ○ Generate Payroll Reports ○ Generate Payment Slips ○ Configure Payment Schedules

3. **Scenario A: Process Payroll for Period:**
    - 3a.1. The Administrator selects the "Process Payroll" option. ○ 3a.2. The Administrator specifies the payment period (e.g., month, week). ○ 3a.3. The system retrieves the salary structure for all active employees.
    - 3a.4. **(If Attendance/Leave Integration Exists):** The system automatically fetches attendance and leave data for the specified period and calculates payable days/hours for each employee. ○ 3a.5. The system calculates the gross salary, deductions (e.g., taxes, provident fund), and net salary for each employee based on their salary structure, attendance (if applicable), and any pre-defined deductions/allowances.
    - 3a.6. The Administrator reviews the calculated payroll for all employees.
    - 3a.7. The Administrator can make manual adjustments to individual employee payments if necessary (refer to Scenario C: Manage Deductions/Allowances or a separate "Adjust Payment" scenario). ○ 3a.8. The Administrator approves the payroll for the specified period. ○ 3a.9. The system marks the payroll as processed and generates payment records for each employee.
4. **Scenario B: Record Offline Payment:**
    - 3b.1. The Administrator selects the "Record Offline Payment" option. ○ 3b.2. The Administrator searches for the employee who was paid offline.
    - 3b.3. The Administrator enters the payment details:
        - ✦ Payment period.
        - ✦ Net amount paid.
        - ✦ Payment method (e.g., Cash, Cheque, Bank Transfer - if recorded manually). ✦ Date of payment.
        - ✦ Reference number (if applicable, e.g., cheque number). ✦ Optional notes.
    - 3b.4. The Administrator confirms the payment details. ○ 3b.5. The system updates the employee's payment history and marks the payroll for that period as "Paid (Offline)".
    - 3b.6. The system generates a record of the offline transaction in the audit log.
5. **Scenario C: Manage Deductions/Allowances:**
    - 3c.1. The Administrator selects the "Manage Deductions/Allowances" option. ○ 3c.2. The Administrator can either:

- Apply a standard deduction/allowance to multiple employees (e.g., a bonus for all staff).
- Manage specific deductions/allowances for an individual employee (e.g., loan repayment, special allowance).

- 3c.3. The Administrator selects the type of deduction/allowance, the amount (fixed or percentage), and the effective period. ○ 3c.4. For individual employees, the Administrator searches for the employee and then adds, edits, or removes specific deductions/allowances. ○ 3c.5. The Administrator confirms the changes. ○ 3c.6. The system updates the employee's payment structure and records the changes in the audit log. These changes will be reflected in the next payroll processing.

6. **Scenario D: View Payment History (Employee-wise):**
   - 3d.1. The Administrator selects the "View Payment History" option.
   - 3d.2. The Administrator searches for a specific employee by ID or name.
   - 3d.3. The system displays the employee's payment history, including:
     - Payment period.
     - Gross salary.
     - Total deductions.
     - Net salary.
     - Payment date.
     - Payment method.
     - Status (Paid, Pending).
     - Link to view/download the payment slip (if generated).
7. **Scenario E: Generate Payroll Reports:**
   - 3e.1. The Administrator selects the "Generate Payroll Reports" option. ○ 3e.2. The system presents various report options (e.g., Monthly Payroll Summary, Salary Register, Deduction Summary, Bank Transfer List). ○ 3e.3. The Administrator selects the desired report type and specifies the period and any other relevant filters (e.g., department, employee category). ○ 3e.4. The system generates the report and displays it to the Administrator (e.g., in a table). ○ 3e.5. The Administrator can optionally export the report in various formats (e.g., CSV, PDF).
8. **Scenario F: Generate Payment Slips:**
   - 3f.1. The Administrator selects the "Generate Payment Slips" option. ○ 3f.2. The Administrator selects the payment period for which slips need to be generated. ○ 3f.3. The Administrator can choose to generate slips for all employees or a specific group/individual. ○ 3f.4. The system generates payment slips for the selected employees, detailing their earnings, deductions, and net pay for the specified period. ○ 3f.5. The Administrator can preview, print, or send the payment slips to employees (e.g., via email through the system).

9. **Scenario G: Configure Payment Schedules:**
    - 3g.1. The Administrator selects the "Configure Payment Schedules" option.
    - 3g.2. The Administrator can define the organization's payment frequency (e.g., monthly, bi-weekly).
    - 3g.3. The Administrator can set specific pay dates for each period. ○ 3g.4. The system uses these schedules for automated payroll processing reminders and reporting.
10. **Administrator Logs Out:** The Administrator logs out of the system.

Alternative Flows:

- **A1: No Active Employees:**
    - If there are no active employees in the system, the payroll processing cannot proceed, and the system displays a message.
- **A2: Missing Salary Structure for Employee:**
    - If an employee record is missing a defined salary structure, the system will flag this employee and may prevent payroll processing until the structure is assigned.
- **B1: Invalid Payment Details:**
    - If the Administrator enters invalid payment details (e.g., non-numeric amount), the system displays an error message.
- **C1: Invalid Deduction/Allowance Details:**
    - If the Administrator enters invalid deduction/allowance details (e.g., percentage outside the valid range), the system displays an error message.
- **E1: No Data for Report:**
    - If there is no payroll data matching the selected report criteria, the system displays a message indicating that no records were found.
- **F1: Error Generating Payment Slips:**
    - If there is an error during payment slip generation, the system displays an error message.

Exceptions:

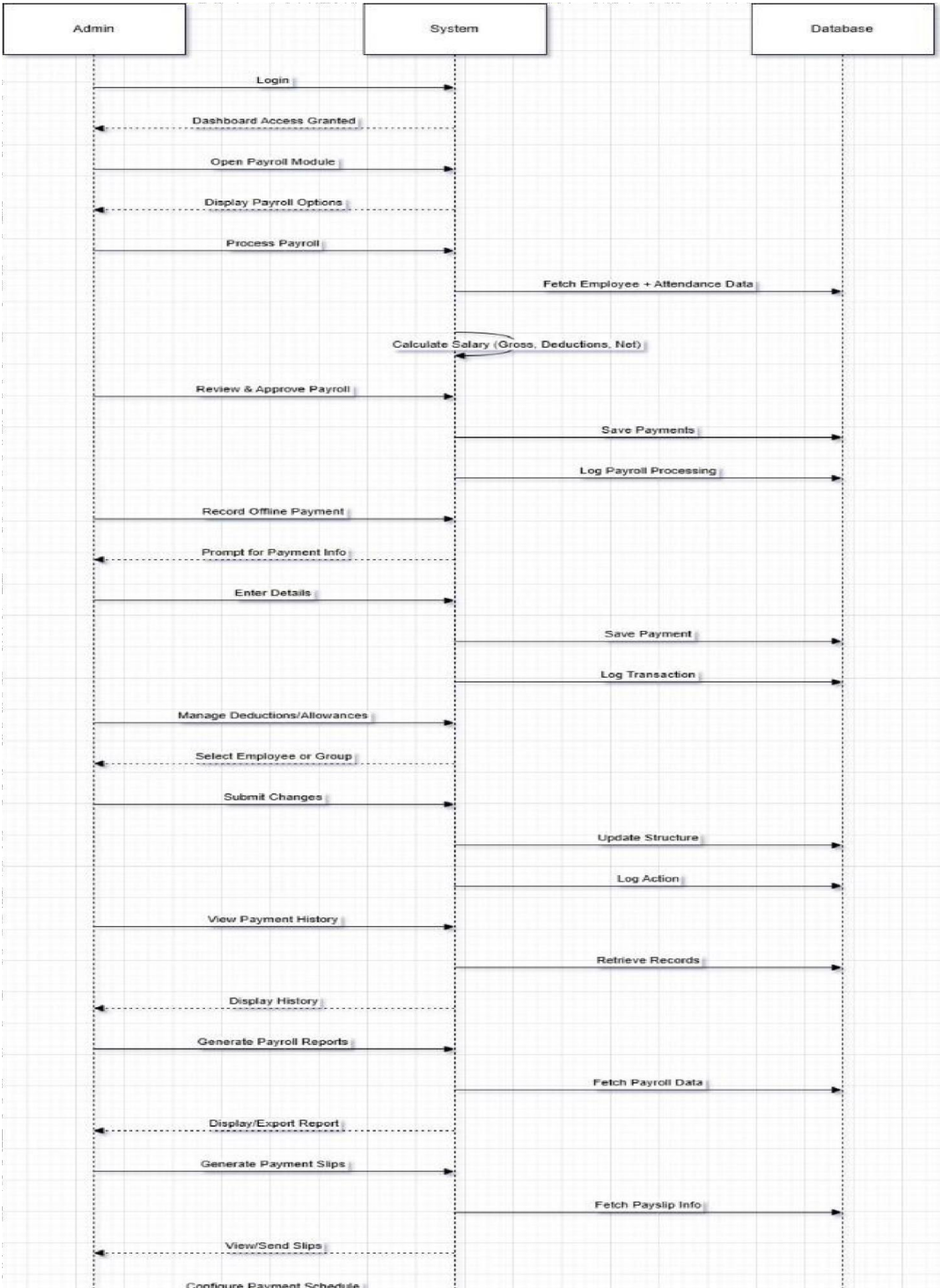
- **System Errors:** If the system encounters any technical errors (e.g., database issues, bank integration failures), it will display an appropriate error message.
- **Insufficient Permissions:** If the Administrator does not have the necessary permissions to perform a specific action, the system will display an authorization error.

Extension Points:

- Integration with accounting software for automated journal entries.
- Automated tax calculations and compliance reporting.
- Employee self-service portal for viewing payment history and downloading slips.

- Advanced features for managing arrears, loans, and other complex payment scenarios.
- Integration with time and attendance systems for automated payroll calculation based on work hours.
- Support for multiple payment currencies.

This fully dressed use case provides a comprehensive description of how an administrator manages employee payments within the hostel room management system, covering the essential processes involved in payroll administration.


Admin
System
Database
Login
Dashboard Access Granted
Open Payroll Module
Display Payroll Options
Process Payroll
Fetch Employee + Attendance Data
Calculate Salary (Gross, Deductions, Net)
Review & Approve Payroll
Save Payments
Log Payroll Processing
Record Offline Payment
Prompt for Payment Info
Enter Details
Save Payment
Log Transaction
Manage Deductions/Allowances
Select Employee or Group
Submit Changes
Update Structure
Log Action
View Payment History
Retrieve Records
Display History
Generate Payroll Reports
Fetch Payroll Data
Display/Export Report
Generate Payment Slips
Fetch Payslip Info
View/Send Slips
Configure Payment Schedule

Use Case Name:

Leaved Employee Management

Primary Actor:

Hostel Manager / Admin

Secondary Actors:

HR (Human Resources), Employee

Stakeholders and Interests:

- **Employee**: Wants their departure to be processed properly, with appropriate records and clearances.
- **Hostel Manager** / **Admin**: Responsible for processing the departure, updating records, and managing any necessary follow-up actions.
- **HR (Human Resources)**: Manages employee status and exit formalities, including final payments and settlement of dues.

Preconditions:

1. The employee has been registered in the hostel management system and has all relevant details (e.g., role, room allocation, employment status) recorded.
2. The employee has provided formal notice or communicated their departure intention to HR and the hostel management team.
3. The employee's departure has been approved by the relevant parties (HR, Hostel Manager, etc.).
4. The system is functioning, and the user (Hostel Manager/Admin) is logged in with proper permissions.

Postconditions:

1. The employee's status is updated to **"Leaved"** in the system.

2. The employee's **room and facilities** (if applicable) are marked as vacated.
3. The employee's **attendance and payroll records** are updated.
4. Any **pending dues**, payments, or clearances are flagged or settled.
5. A **confirmation notification** is sent to the employee, HR, and the hostel management team.
6. The employee's data is **archived** for future reference and compliance purposes.
7. A report is generated for **leaved employees** for record-keeping and audits.

1. **Trigger:**
   The HR or hostel management team opens the Employee Management GUI and decides to manage employee records (e.g., add, update, remove employees).

2. **Viewing Employee List:**
   - The system displays a list of current employees with their names and experience.
   - The HR/Hostel Manager selects an employee from the list to update or remove, or opts to add a new employee.

3. **Adding an Employee:**
   - The HR/Hostel Manager clicks **Add Employee**.
   - The system displays a dialog prompting for the new employee's **Name**, **Email**, **Phone**, and **Experience**.
   - The HR/Hostel Manager fills in all fields and submits. ○ The system validates the input (all fields required) and checks for duplicate employee names.
   - If validation passes, the employee is added to the system and the list updates.
   - The system confirms success to the user.

4. **Updating Employee Details:**
   - The HR/Hostel Manager selects an employee and clicks **Update Selected**. ○ The system shows a dialog pre-filled with the employee's current details. ○ The HR/Hostel Manager edits the details and submits. ○ The system validates the input and ensures the new name is unique if changed. ○ If validation passes, the employee record updates and the list refreshes.
   - The system confirms success.

5. **Removing an Employee:**
   - The HR/Hostel Manager selects an employee and clicks **Remove Selected**. ○ The system asks for confirmation. ○ Upon confirmation, the employee is removed from the system and the list refreshes.
   - The system confirms success.

6. **Closing the GUI:**
   - The HR/Hostel Manager clicks the close button. ○ The system closes the Employee Management window safely.

Alternative Flows (Extensions):

2. **Add Employee - Duplicate Name:**

- If the HR/Hostel Manager tries to add an employee with a name that already exists, ○ The system shows an error message: "Employee with this name already exists. Please use a unique name."
- The employee is not added until the name is unique.

3. **Update Employee - Duplicate Name Conflict:**
- If the HR/Hostel Manager changes the name of an employee to a name that already exists for a different employee,
- The system shows an error message: "Failed to update employee. Check if the new name already exists for a different employee." ○ The update is rejected until a unique name is provided.

4. **Remove Employee - No Selection:**
- If the HR/Hostel Manager clicks **Remove Selected** without selecting an employee, ○ The system shows an error message: "Please select an employee to remove."

5. **Update Employee - No Selection:**
- If the HR/Hostel Manager clicks **Update Selected** without selecting an employee, ○ The system shows an error message: "Please select an employee to update."

6. **Add/Update Employee - Incomplete Input:**
- If any of the input fields (Name, Email, Phone, Experience) are left empty during Add or Update operations, ○ The system shows an error message: "All fields are required."

7. **Removal Confirmation Cancelled:**
- If the HR/Hostel Manager cancels the removal confirmation dialog, ○ The employee remains in the system and the list is unchanged.

Exception Flows:

1. System Error During Update: ○ If there's an error while updating the employee's status, the process is halted, and the system displays an **error message**.
   - The system logs the error, and the HR/Hostel Manager can attempt the update again after troubleshooting.
2. Employee Record Not Found: ○ If the employee's record cannot be located (e.g., due to system issues), the system prompts the HR/Hostel Manager to verify the employee's details and attempt the update again.
3. Incomplete Exit Formalities:

- If exit formalities (such as return of room keys or company equipment) are incomplete, the system alerts the HR/Hostel Manager, and the status update cannot be completed until these tasks are resolved.

4. User Permissions Error: ○ If someone without the appropriate permissions (e.g., unauthorized staff) tries to mark an employee as "Leaved," the system denies access and shows an "**Access Denied**" message.

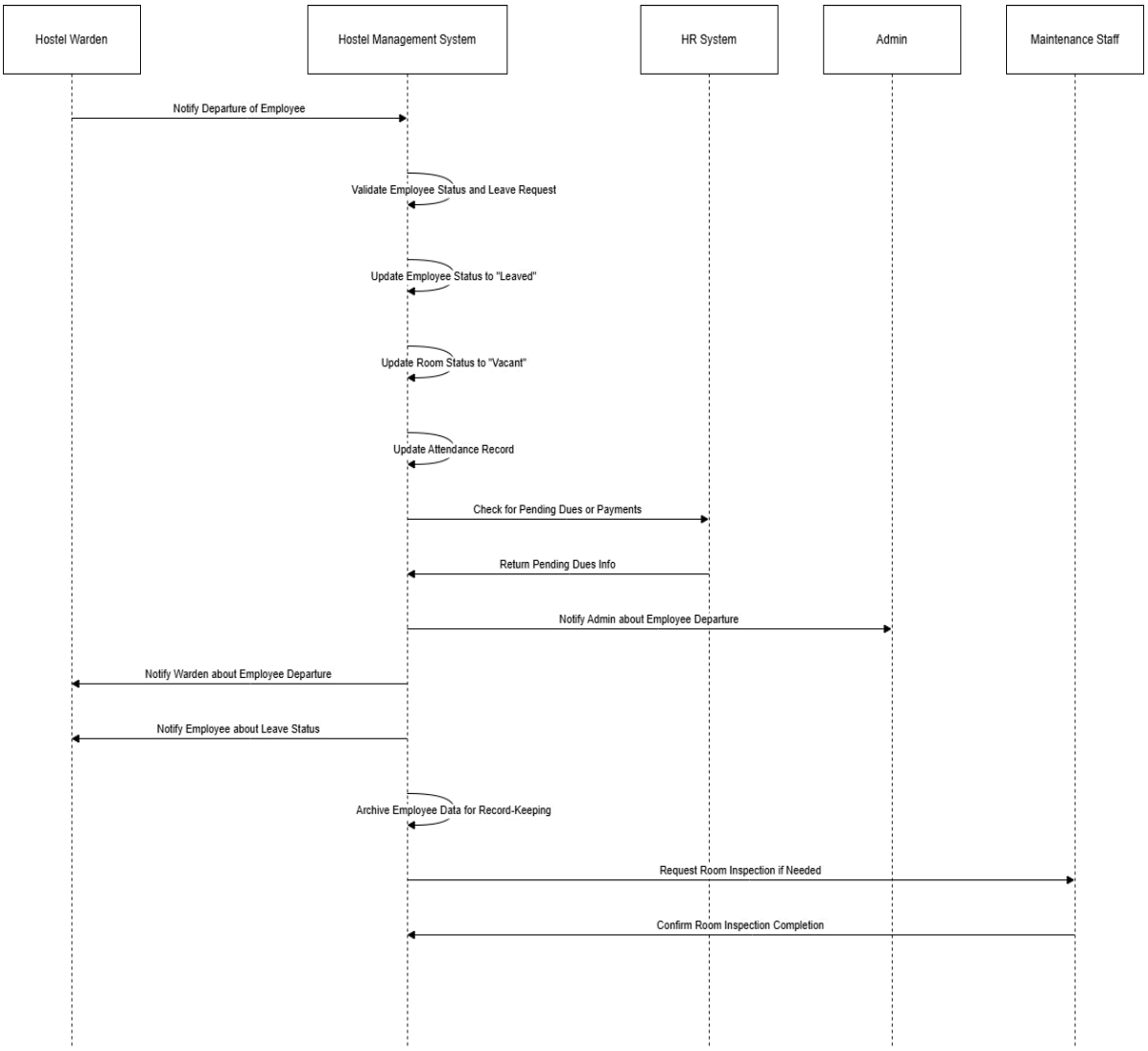
Trigger:

- The **trigger** for this use case occurs when an employee notifies the HR or Hostel Manager of their departure (e.g., resignation, contract completion, etc.), and formal approval or acknowledgment is received to proceed with updating their status.

Special Requirements:

- **Data Security & Privacy:** Employee data must be handled with confidentiality, ensuring that only authorized personnel can access or modify the records.
- **Exit Formalities:** The system should support exit formalities like returning company property, keys, and other hostel-related equipment.
- **Notifications:** Automatic notifications should be sent to the employee, HR, and hostel management team upon the successful processing of the employee's departure.
- **Reports:** A report on leaved employees should be generated for auditing and record- keeping, including details on dues, room vacancies, and payroll settlements.
- **System Availability:** The system must be available for HR and Hostel Managers to complete employee exit formalities at any time.

SSD for leaved Employee:



NAME : Uzair Arif

REGISTRATION NO : SP23-BSE-168

Use Case: New Student Admission in Hostel

Primary Actor: Hostel Warden

Secondary Actors: Admin, Student

Preconditions

- Student has been institutionally admitted.
- Required documents have been submitted.
- System is operational with valid warden access. Rooms are available, or a waiting list is active.

Post conditions

- Student is registered in the hostel system.
- Room assigned and marked as occupied.
- Attendance record initialized.
- Welcome notification sent.

Main Success Scenario

1. Admin logs in to the system and navigates to the "Student Management" section.
2. Selects "New Admission" and enters student details along with submitted documents.
3. The system verifies institutional admission.
4. Room is auto-assigned or manually selected.
5. Hostel ID card is generated.
6. System sends confirmation to all parties involved.

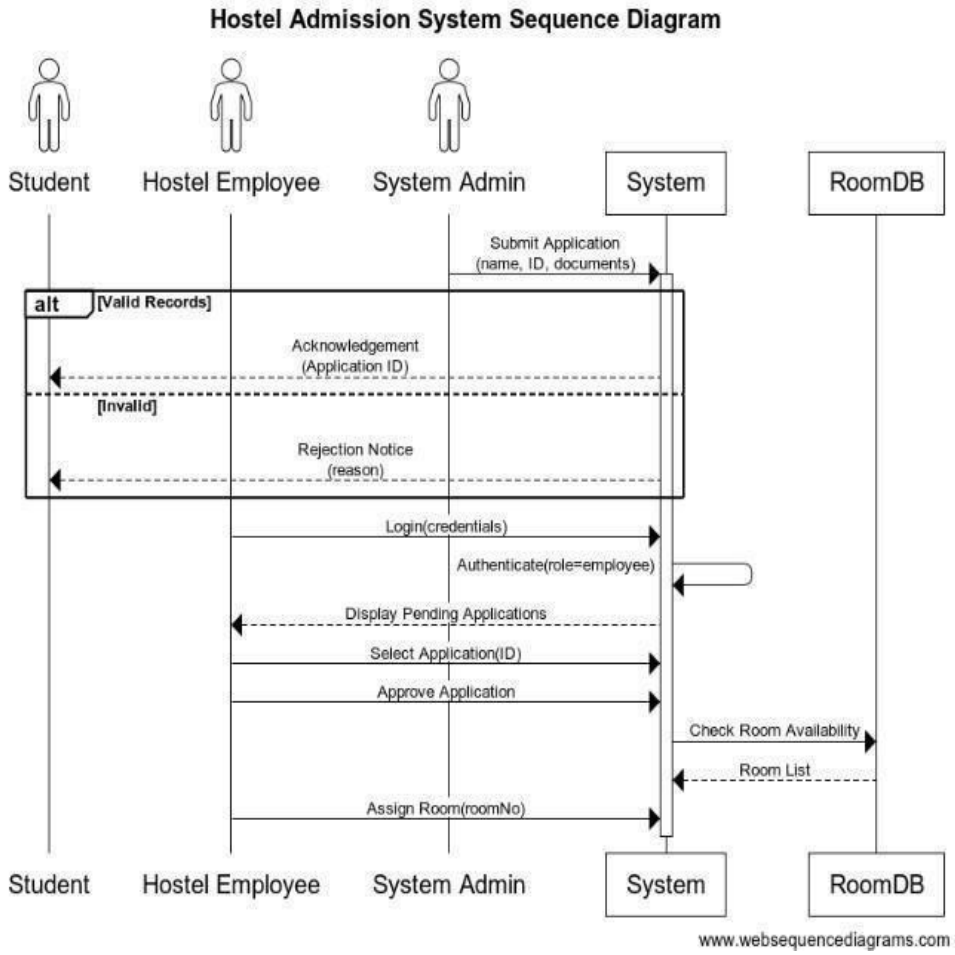
Alternative Flows

- **A1**: No rooms available → Student is placed on a waitlist.
- **A2**: Missing documents → System flags for completion.
- **A3**: Duplicate entry detected → System alerts warden.

Exception Flows

- **E1**: System crash during entry → Data recovery protocol is activated.
- **E2**: Unauthorized access attempt → Incident is logged and admin is alerted.

System sequence Diagram



Name **Adil Bashir Reg**

No **SP23 BSE 020**

Fully Dressed uses cases:

1. Student Living

Use Case ID: UC-SL-001

Use Case Name: Student Living

Primary Actor: Student

Trigger: System sends a notification or when a student moves in.

Preconditions:

- The student is enrolled at a university or college.
- The student has access to the student housing system (online portal or office).
- Available student accommodations exist for the student's location.

Postconditions:

- The student is successfully placed in a suitable accommodation.
- The housing office has updated records of student housing assignments. Payment and contracts are successfully handled through the system.

Basic Flow (Main Success Scenario): 1.

Student logs into the Housing Portal:

- The student enters their login credentials (username and password) into the housing portal.
  - The system verifies the student's identity based on their university ID.
  - The system presents available housing options based on the student's academic year, preferences, and budget.

2.

Student searches for accommodations: ○ The student filters available housing based on criteria like:

- Price range
- Room type (single, shared, suite, etc.)
- Location (on-campus or off-campus)
- Amenities (Wi-Fi, study spaces, gym access, etc.) ○ The system returns a list of accommodations matching the student's filters.

3.

Student selects a housing option:

- The student selects a suitable accommodation from the list.
- The system provides detailed information about the accommodation, including:
  - Rent price and payment schedule
  - Lease terms and duration
  - Photos or virtual tour ▪ Reviews from other students

4.

Student applies for the accommodation:

- The student submits an application for the chosen accommodation. ○ The system requires the student to enter personal information (e.g., emergency contact, ID verification) and possibly a deposit. ○ The student can opt to pay an application fee through the portal if required.

5.

Housing Office reviews and approves the application:

- The housing office receives the student's application.
- The system notifies the housing office of new applications. ○ The housing office reviews the application and confirms that the student meets eligibility criteria. ○ The housing office approves the student's application for housing.

6.

Student signs the housing contract:

- The system generates an electronic lease agreement with terms and conditions. ○ The student reviews and digitally signs the agreement through the portal. ○ The system updates the housing office's records, confirming the student's housing assignment.

7.

Payment process:

- The system generates a payment schedule for the student, outlining the rent and due dates.
- The student makes an initial payment through the housing portal (this may include a deposit, first month's rent, etc.). ○ The system confirms the payment and updates the housing records.

8.

Student moves into accommodation:

- The system sends an email or app notification to the student with the move-in date, room number, and instructions for key collection.
- The student moves into the accommodation on the designated date. ○ The system logs the student's move-in date and updates housing occupancy records.

Alternative Flows:

1. **Application Denied:**

- If the housing office denies the student's application (due to eligibility issues, missing documents, or other reasons), the system notifies the student.
- The student may be offered alternative accommodations or given the option to reapply or appeal.

2. Payment Failure:

- If the student's payment fails (due to issues like insufficient funds or incorrect payment information), the system notifies the student immediately.
- The student is prompted to retry the payment or select a different payment method.

3. Student Cancels Application: If the student decides to cancel the application after submission, the system allows the student to withdraw the application.

- The housing office is notified of the cancellation, and the accommodation becomes available to other students.

Extensions:

1. Student requests a Room Change: If the student faces issues with the accommodation (e.g., noise, maintenance), they can request a room change through the housing portal.

- The housing office receives the request and processes it according to availability and student needs.

2. Student Requests Maintenance: The student can submit a maintenance request through the portal for issues like plumbing or heating problems.

- The system notifies the housing office or facilities team, who will schedule repairs.

Non-Functional Requirements:

- **Usability**: The system must have a user-friendly interface, especially since many students are unfamiliar with complex systems.
- **Performance**: The system should be able to handle high traffic, especially during peak housing application seasons (beginning and end of the academic year).
- **Security**: Sensitive student data such as personal identification, payment details, and housing preferences must be encrypted and stored securely.

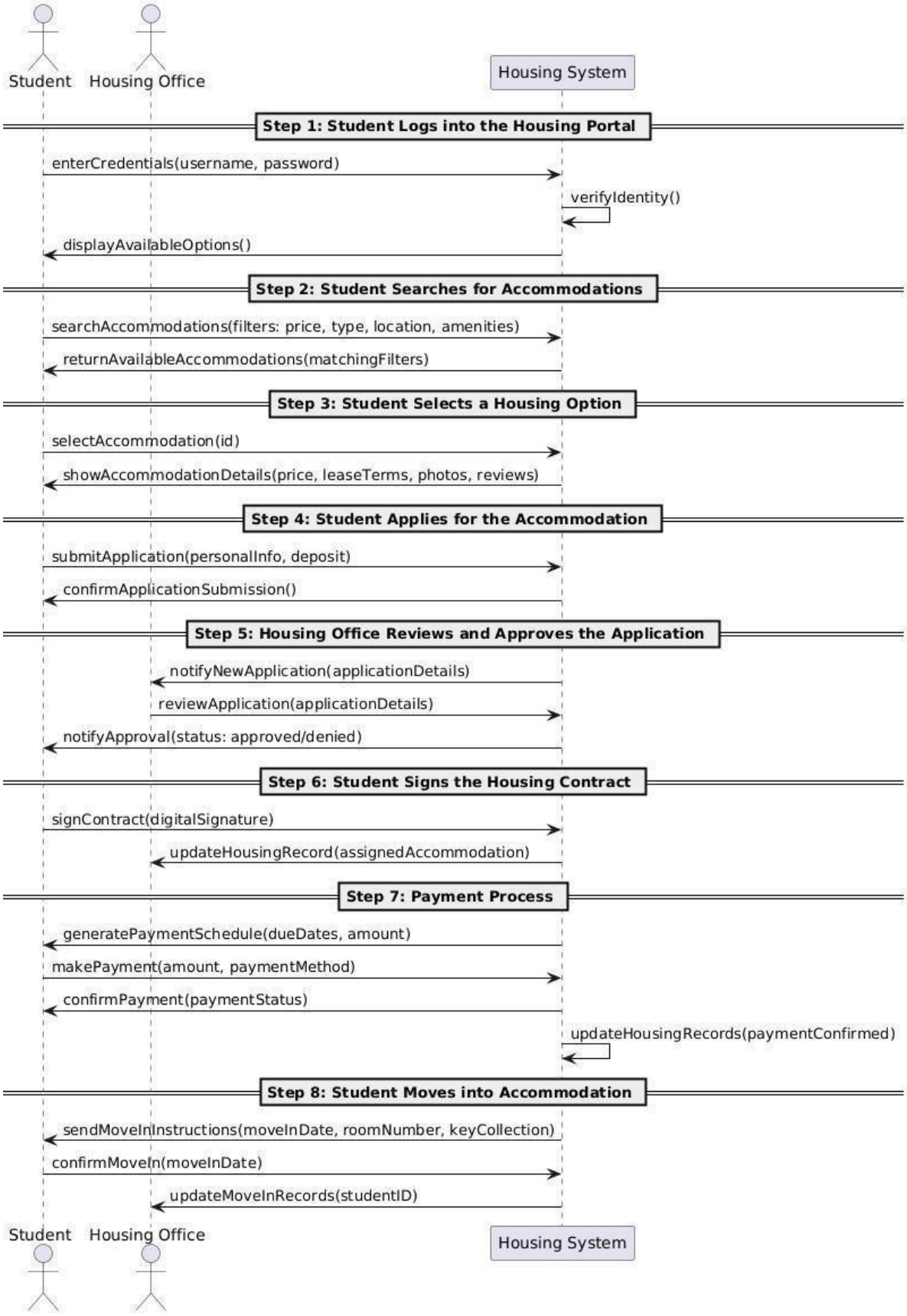
Frequency of Use:

- **High Frequency**: During housing application periods at the beginning of each semester or academic year.
- **Occasional Use**: Maintenance requests, contract renewals, or payment scheduling.

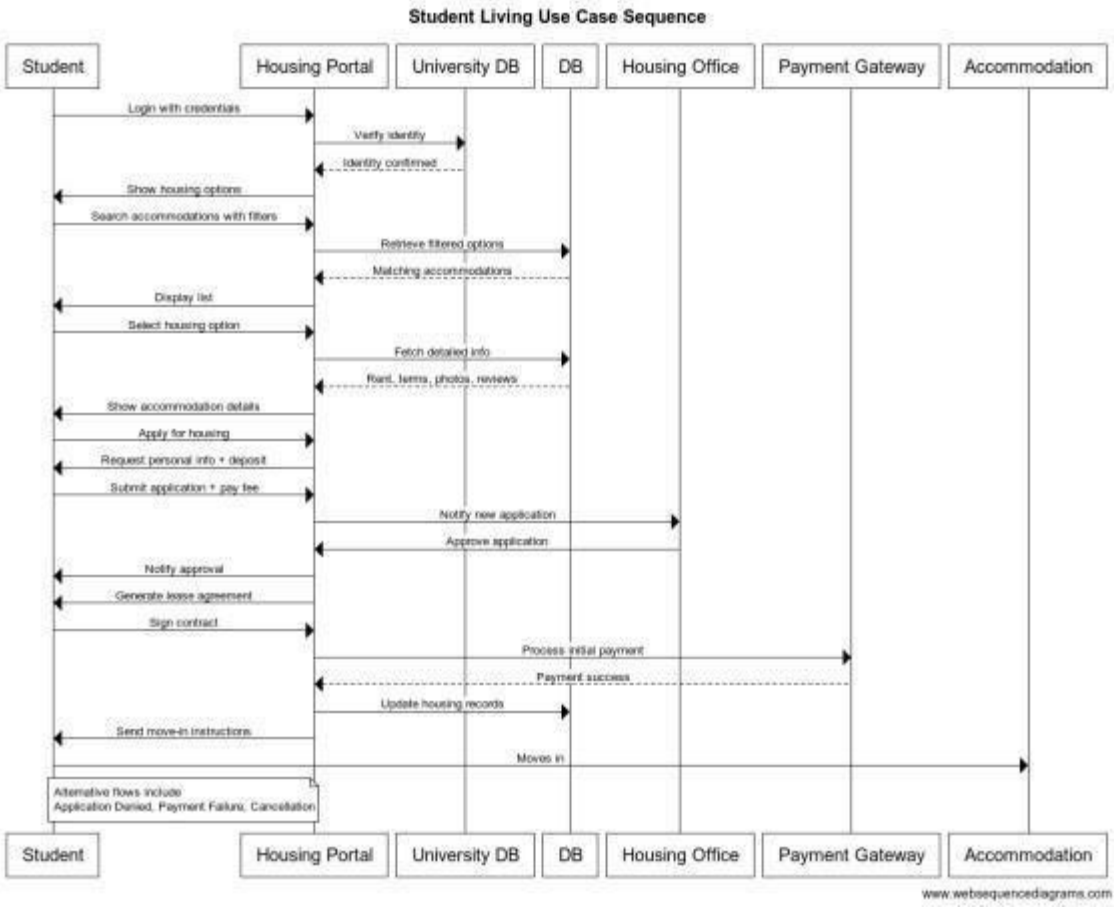
Assumptions:

- Students have internet access and basic computer skills.
- The housing system integrates with the university's student records system to verify enrollment status.
- The system supports various payment methods (credit/debit card, bank transfer, etc.).

SSD:


Student
Housing Office
Housing System
Step 1: Student Logs into the Housing Portal
enterCredentials(username, password)
verifyIdentity()
displayAvailableOptions()
Step 2: Student Searches for Accommodations
searchAccommodations(filters: price, type, location, amenities)
returnAvailableAccommodations(matchingFilters)
Step 3: Student Selects a Housing Option
selectAccommodation(id)
showAccommodationDetails(price, leaseTerms, photos, reviews)
Step 4: Student Applies for the Accommodation
submitApplication(personalInfo, deposit)
confirmApplicationSubmission()
Step 5: Housing Office Reviews and Approves the Application
notifyNewApplication(applicationDetails)
reviewApplication(applicationDetails)
notifyApproval(status: approved/denied)
Step 6: Student Signs the Housing Contract
signContract(digitalSignature)
updateHousingRecord(assignedAccommodation)
Step 7: Payment Process
generatePaymentSchedule(dueDates, amount)
makePayment(amount, paymentMethod)
confirmPayment(paymentStatus)
updateHousingRecords(paymentConfirmed)
Step 8: Student Moves into Accommodation
sendMoveInInstructions(moveInDate, roomNumber, keyCollection)
confirmMoveIn(moveInDate)
updateMoveInRecords(studentID)
Student
Housing Office
Housing System

SDD:



Fully Dressed uses cases:

2. Student Leaving

Use Case ID:	UC-SL-002
Use Case Name:	Student Leaving
Primary Actor:	Student
Trigger:	The student receives an **email notification**

from the housing system, reminding them that their housing contract will end in one week

Preconditions:

- The student has received permission or is obligated to leave the accommodation (end of semester, graduation, etc.).
- The student is aware of the move-out date and related deadlines.
- The housing portal is accessible to the student for processing their move-out details.

Postconditions:

- The student has successfully vacated the accommodation.
- The housing office has completed the move-out procedures and updated the housing records.
- Any outstanding payments or damages have been resolved or accounted for. The accommodation is available for new tenants.

Basic Flow (Main Success Scenario):

1. **Student Logs into Housing Portal**: ○ The student logs into the housing portal using their student credentials. ○ The system verifies the student's identity and shows the current housing assignment.
2. **Student Initiates Move-Out Process**:
   - ○ The student clicks on the "Move-Out" button or tab within the portal.
   - ○ The system prompts the student with move-out instructions and deadlines (e.g., return keys, clean the room, etc.). ○

3. **System Notifies Student of Requirements**:
   ○ The system provides the student with a checklist for vacating the accommodation, including:
   - Cleaning instructions (e.g., vacuuming, trash removal).
   - A move-out inspection date.
   - Key return process.
   - A reminder to settle any unpaid rent or fees.
   - An option to submit a maintenance request if there are any issues with the room (e.g., broken furniture, plumbing issues).

4. **Student Schedules Move-Out Inspection**:
   ○ The student schedules a move-out inspection with the housing office or property management through the portal, selecting a time that fits their schedule. ○ The system confirms the inspection appointment and sends a reminder to the student.
5. **Student Prepares for Move-Out**:
   ○ The student packs their belongings and cleans the room, following the provided guidelines. ○

The student ensures that all furniture and appliances are in good condition or submits a maintenance request if necessary.

6. **Move-Out Inspection Occurs**: ○ The housing office or property manager conducts a room inspection. ○ The system generates an inspection checklist that is reviewed during the inspection. ○ If the room is in good condition, the inspection is marked as "Passed." ○ If damages are found, the system generates a report and an estimate for repairs. ○ The system records the findings and sends a notification to the student with the inspection results.

7. **Student Settles Financial Obligations**:
   - ○ The system checks if the student has any outstanding payments (e.g., rent, utilities, damage fees). ○ If there are any unpaid fees, the system presents the student with the outstanding balance.
   - ○ The student makes the final payment for any outstanding charges through the portal. ○ The system confirms the payment and updates the student's account.

8. **Key Return Process**:
   - ○ The system provides the student with instructions on how to return the accommodation keys (e.g., drop-off location, office hours).
   - ○ The student returns the keys to the designated location.
   - ○ The system logs the return of the keys and marks the move-out as complete.

9. **Student Confirms Move-Out**:
   ○ Once the inspection is passed and any fees are settled, the student confirms their departure through the portal. ○ The system sends a final confirmation message to the student, informing them that their housing contract has ended and their move-out is complete.

10. **Accommodation Availability Updated**:
    - The system updates the housing records, marking the accommodation as available for new tenants.
    - The student's record is updated to reflect their move-out status.

Alternative Flows:

1. Damages Detected During Inspection:

- If the housing office or property manager detects damages (e.g., broken furniture, stained carpets), the system generates a damage report and an estimate for repairs.

- The student is notified about the damages, and a charge is applied to their final bill.
- If the student disputes the charges, they can submit a request for further review via the portal.

2. **Payment Failure:** If the student is unable to pay any outstanding fees (e.g., rent or damage charges), the system sends a reminder email.
   - The student may be given additional time to pay or offered a payment plan if applicable.
   - The housing office may hold the student's security deposit or take other actions to ensure payment.

3. **Student Cancels Move-Out:**
   - If the student changes their mind and decides to stay longer in the accommodation (with approval), they can cancel the move-out process through the portal.
   - The system updates the student's housing status and notifies the housing office of the cancellation.

Non-Functional Requirements:

- **Usability**: The portal should have an intuitive user interface for students, ensuring they can easily complete all necessary tasks (move-out, payment, inspection).
- **Performance**: The system must handle peak move-out periods without delays or crashes.
- **Security**: The system must ensure that all student data, especially financial transactions and room inspection details, is kept secure and private.

Frequency of Use:

- **High Frequency**: Typically occurs at the end of each academic term (semester or year), especially during graduation, transfer, or other terminations of housing contracts.
- **Occasional Use**: Student requests for room changes, extensions, or early move-out due to unforeseen circumstances.

Assumptions:

- The student has completed all prior requirements (e.g., rent payment, accommodation contract) before initiating the move-out process.
- The housing office provides adequate information on how to properly vacate the accommodation.
- The student is not leaving mid-term without prior approval.

SSD:


Student
Housing Office
Housing System
Step 1: Student Logs into Housing Portal
enterCredentials(username, password)
verifyIdentity()
displayHousingAssignment()
Step 2: Student Initiates Move-Out Process
clickMoveOutButton()
displayMoveOutInstructionsAndDeadlines()
Step 3: System Notifies Student of Requirements
displayMoveOutChecklist(clearingInstructions, inspectionDate, keyReturnProcess, unpaidFeesReminder, maintenanceRequestOption)
Step 4: Student Schedules Move-Out Inspection
scheduleMoveOutInspection(time)
confirmInspectionAppointment(time)
sendInspectionReminder(time)
Step 5: Student Prepares for Move-Out
prepareRoom(cleaning, packBelongings, maintenanceRequest)
updateRoomStatus(readyForInspection)
Step 6: Move-Out Inspection Occurs
notifyInspection(time)
conductRoomInspection()
generateInspectionChecklist()
notifyInspectionResults(status: passed/damages)
Step 7: Student Settles Financial Obligations
checkOutstandingFees()
makePayment(amount)
confirmPayment(paymentStatus)
updateStudentAccount()
Step 8: Key Return Process
provideKeyReturnInstructions(location, officeHours)
returnKeys(location)
logKeyReturn()
Step 9: Student Confirms Move-Out
confirmMoveOut()
sendMoveOutConfirmation()
Step 10: Accommodation Availability Updated
updateHousingRecords(availableForNewTenants)
updateStudentRecord(moveOutStatus)
Student
Housing Office
Housing System

SDD:

Student Leaving Sequence Diagram