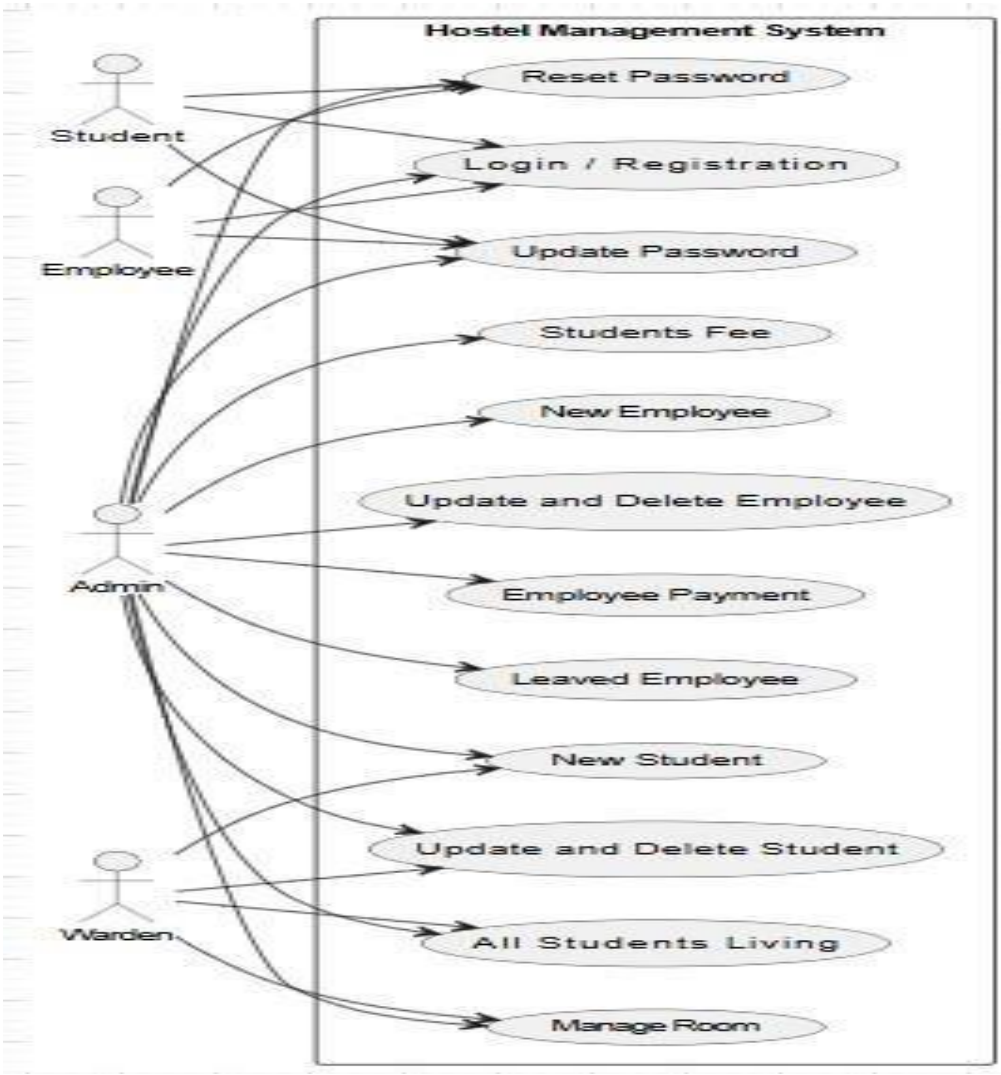
PROJECT TITLE: HOTEL MANAGEMENT SYSTEM

Contents

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

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.

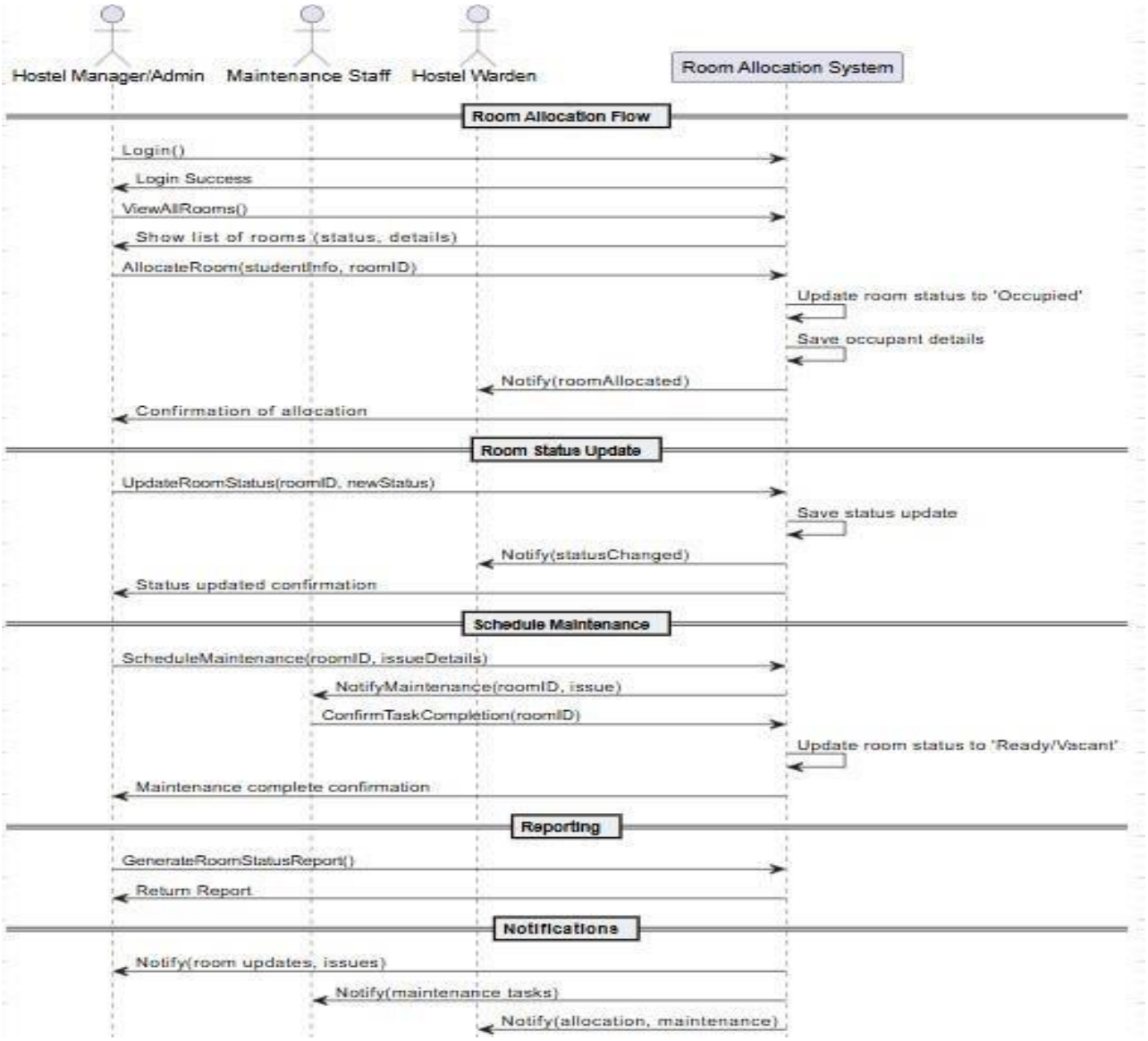
Trigger:

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

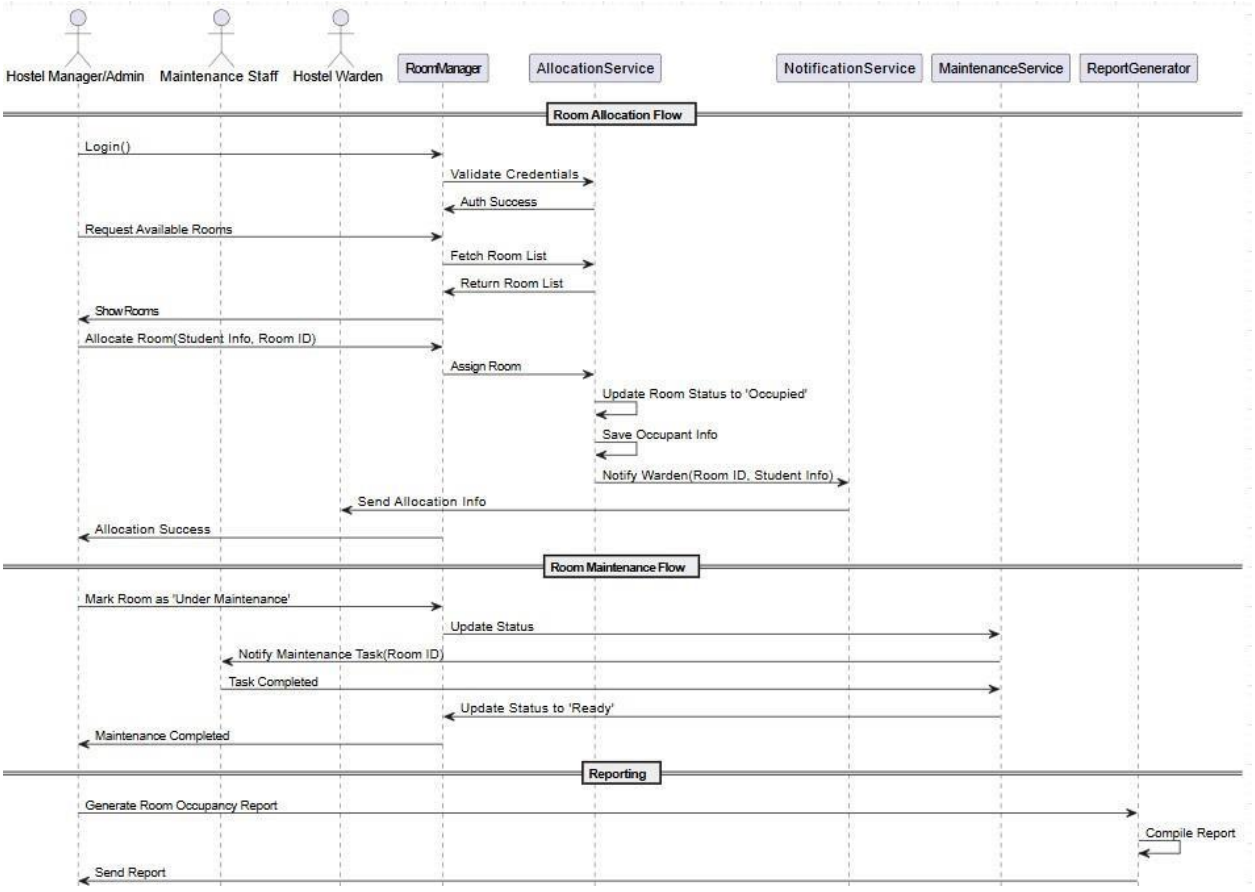
Special Requirements:

- **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.

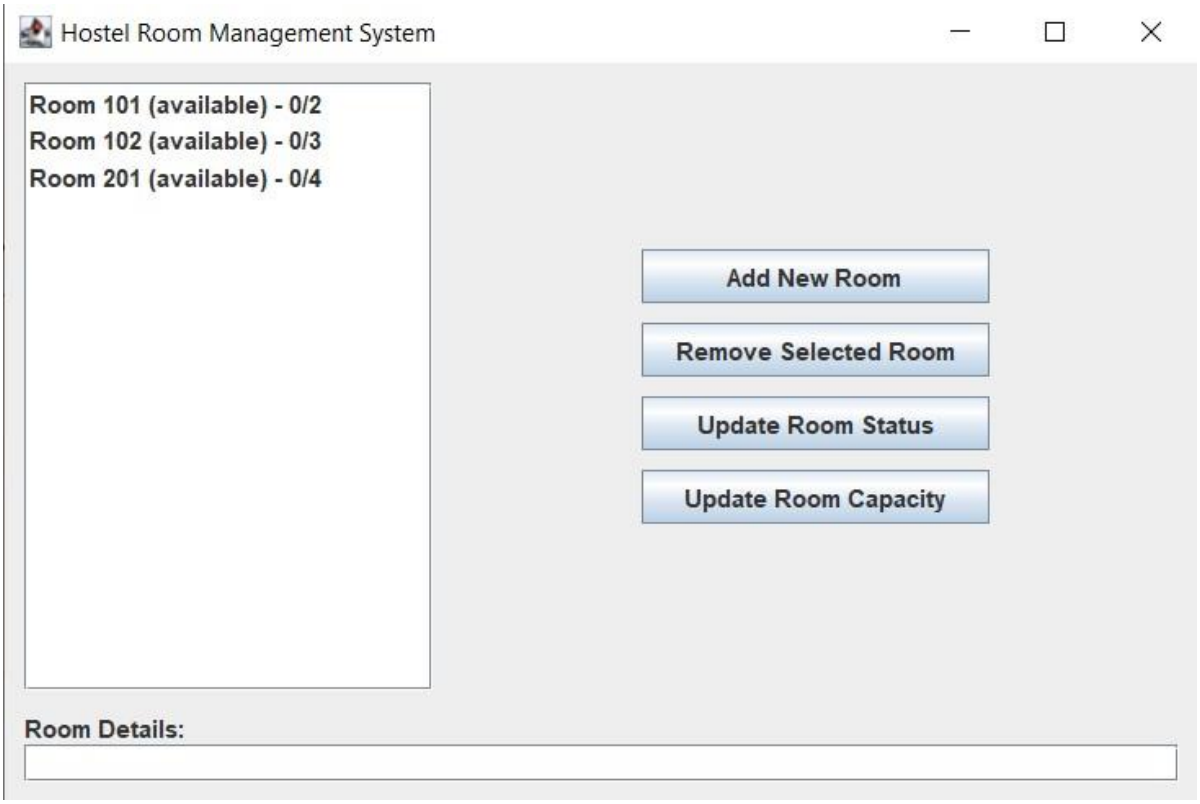
System Sequence Diagram:



Sequence Diagram:



UI Prototype:

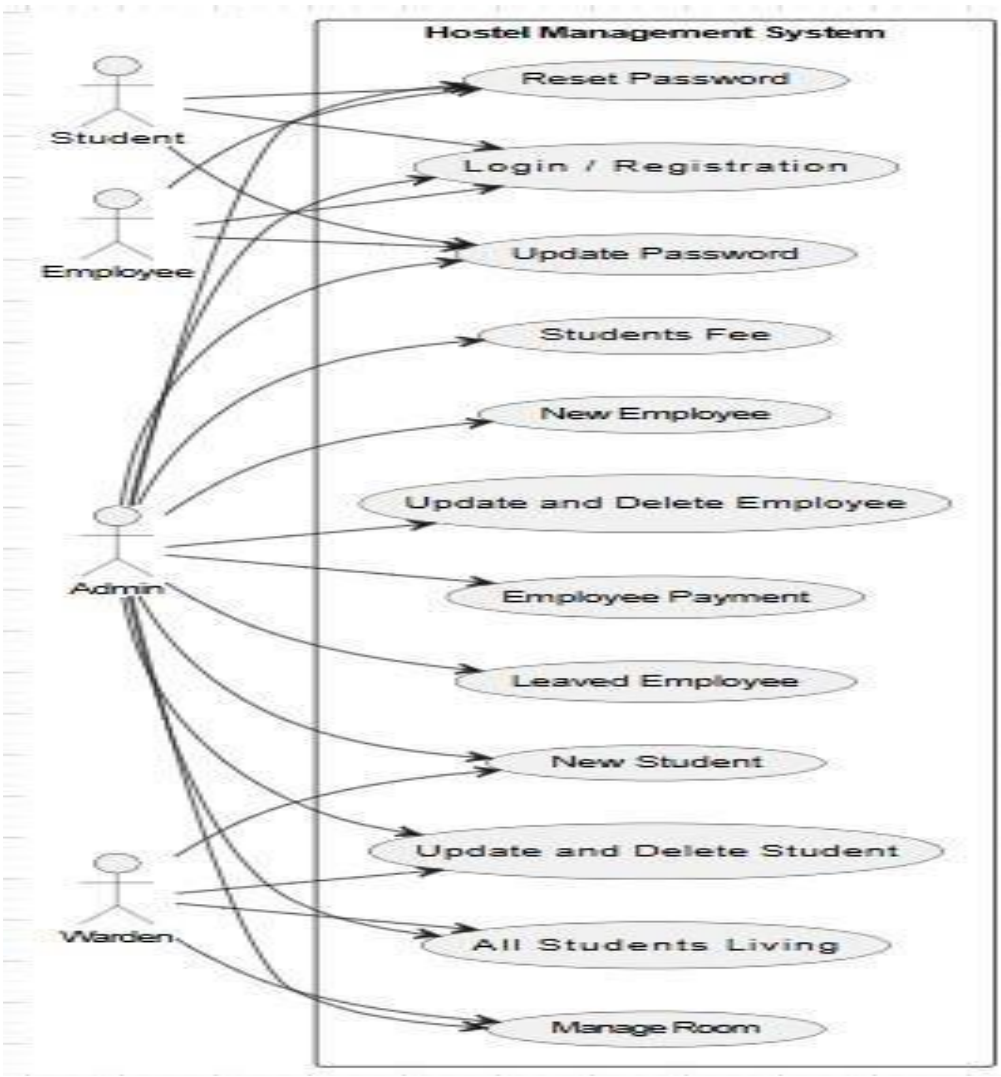


NAME: SARDAR ZAIN

REGISTRATION NO : SP23-BSE-013

TASK:

DOCUMENTATION AND CODING FOR ONLY **LOGIN** USECASE



Full Address Use Case (LOGIN)

Use Case: User Login

Use Case ID: UC-001

Primary Actor: User

Goal: Securely authenticate users and grant access to role-specific functionalities.

Scope: Authentication System

Description:

This use case describes the process by which a user (student, employee, or admin) logs into the system using valid credentials. Upon successful authentication, the user is redirected to a personalized dashboard with features and permissions tailored to their role.

Trigger:

The user clicks on the “Sign In” or “Login” button from the application or website.

Preconditions:

1. The user must be registered in the system.
2. The user must possess valid login credentials (email/ID and password).
3. The system and its authentication services must be operational.

Postconditions:

Success:

- The user is authenticated and redirected to their role-specific dashboard.
- The login timestamp and last login details are updated in the system logs.

Failure:

- The system displays appropriate error messages without granting access.

Main Flow (Normal Flow):

1. **User Accesses Login Interface:**
    - The user navigates to the login page via a URL or application.
2. **User Inputs Credentials:**
    - The user provides a valid username (email or ID) and password.
3. **System Validates Credentials:**
    - The system verifies if the username exists in the database.
    - If the username exists, the system compares the provided password against the stored hash.
4. **Successful Authentication:**
    - The user is logged in.

- The system redirects the user to their respective dashboard (Student, Admin, or Staff).
- The login event is logged with a timestamp.

5. **Access to Functionalities:**
   - The user is granted access to features and data as per their assigned role.

Alternative Flows (Invalid Inputs):

- **A1: Invalid Username:**
  - Condition: The entered username does not exist.
  - System Response: “User does not exist.”
- **A2: Invalid Password:**
  - Condition: The username exists but the password is incorrect.
  - System Response: “Your password is incorrect.”

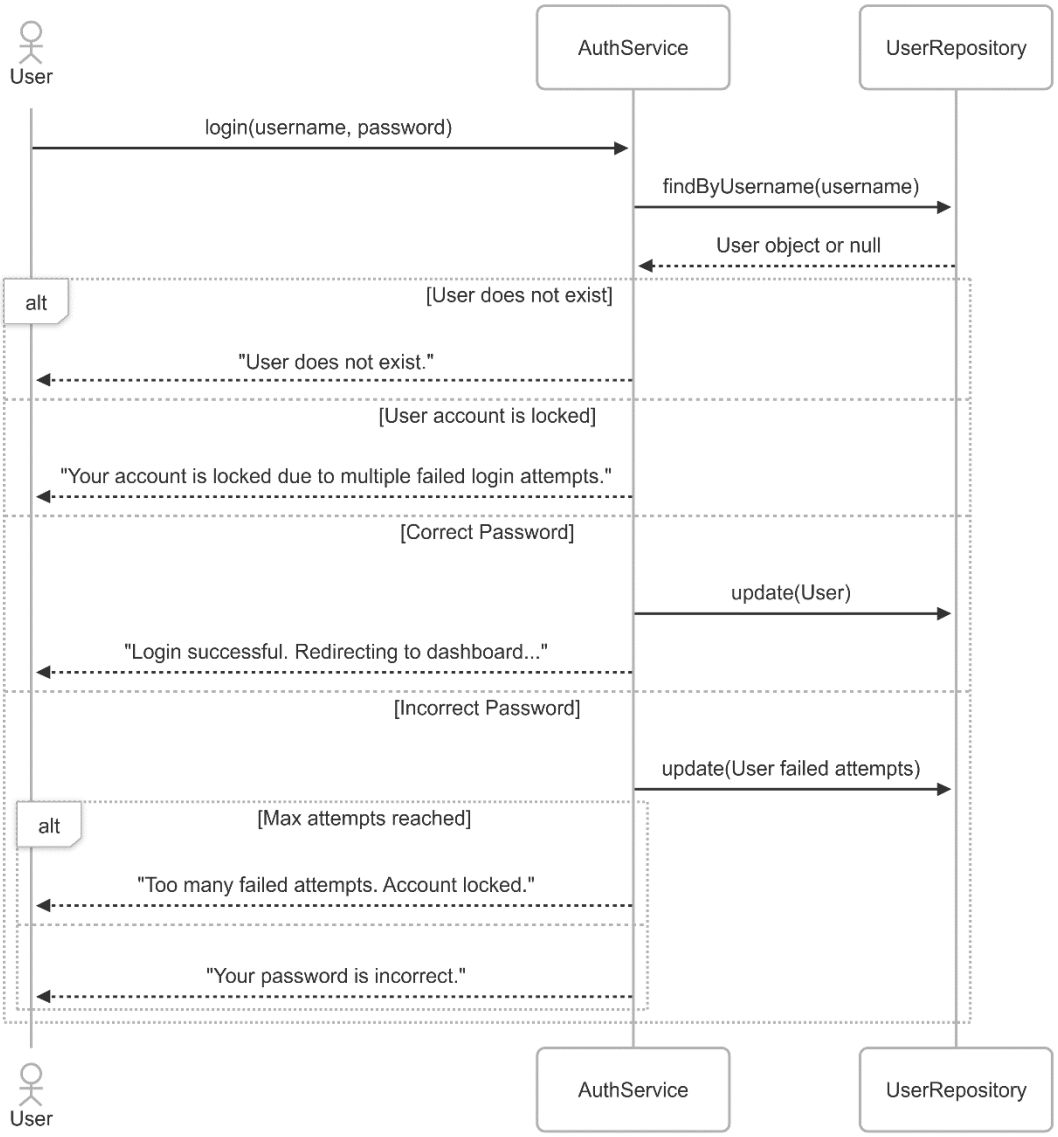
Exceptional Flows:

- **E1: Connection Timeout:**
  - Condition: The network connection times out during login.
  - System Response: The user is notified: “Connection timeout. Please check your internet connection and try again.”

Extension Points:

- **Account Lockout Mechanism (Security Extension):**
  - Trigger: Multiple consecutive failed login attempts (e.g., 5 attempts).
  - Action: The system temporarily locks the account and notifies the user via email.
  - Unlocking may require admin intervention or user action via email verification.
-

- **System sequence Diagram:**



Abdullah Sajid
Sp23-bse-019

Fully dressed use case for add, update and delete employee

Use Case Name:

New Employee Registration

Primary Actor:

HR Manager

Secondary Actors:

IT Administrator, Department Manager

Stakeholders and Interests:

- **New Employee:** Wants their profile set up and access granted to relevant systems.
- **HR Manager:** Needs to ensure employee details are recorded accurately and onboarding is smooth.
- **IT Administrator:** Needs to provide necessary access credentials and work tools.
- **Department Manager:** Wants the employee registered correctly for role assignment.

Preconditions:

- The new employee has accepted the job offer.
- All mandatory documents have been submitted.
- The HR Manager is logged into the system with appropriate permissions.
- The system is functioning properly.

Postconditions:

- Employee data is saved in the system.
- Employee is assigned a unique ID.
- Login credentials are generated.
- Notifications are sent to IT and the Department Manager.
- The onboarding checklist is initiated.

Main Success Scenario (Basic Flow):

- **Trigger:** The HR Manager initiates the process after receiving a signed offer letter.

- **HR Input:**
- Logs into the system and navigates to the Employee Management section.
- Selects “Add New Employee.”
- Enters personal details, job title, department, and documents.
- **System Action:**
- Validates mandatory fields.
- Assigns a unique employee ID.
- Stores data in the HRMS database.
- **IT Notification:**
- System sends notification to IT for system setup.
- IT assigns email, access credentials, and work tools.
- **Manager Notification:**
- System sends notification to Department Manager about new hire.
- **Onboarding Setup:**
- System generates onboarding checklist (training, documentation, etc.).
- **Confirmation:**

HR receives confirmation and sends welcome email to the employee.

Alternative Flows (Extensions):

- **Missing Document:**
- Step 2A: If documents are missing, system halts process and prompts to upload.
- **Duplicate Employee:**

- Step 3A: If employee with same ID exists, system shows a duplicate warning.
- **IT Setup Delays:**
- Step 4A: If IT doesn't acknowledge setup, reminder is auto-triggered.

Exception Flows:

- **System Error:**
- System fails during registration, HR is alerted and logs the issue.
- **Permission Denied:**
- Unauthorized users attempting registration are denied access.

Trigger:

- HR initiates upon final hiring confirmation.

Special Requirements:

- Secure handling of personal data.
- Compliance with labor laws.
- Role-based access controls.

Use Case Name:

Delete Employee Record

Primary Actor:

HR Manager

Secondary Actors:

Admin

Stakeholders and Interests:

- **HR Manager:** Wants to remove employee data cleanly.
- **Admin:** Ensures proper auditing and no data loss.

Preconditions:

- Employee has officially left the organization.
- All exit formalities and clearances are complete.
- HR has appropriate access rights.

Postconditions:

- Employee status marked as "Inactive" or "Deleted."
- Records are archived.
- IT access is revoked.
- Notifications are sent to Admin and relevant departments.

Main Success Scenario (Basic Flow):

- **Trigger:** HR receives final clearance form.
- **HR Action:**
- Logs in and navigates to Employee Management.
- Searches for employee by ID.
- Selects “Delete Employee” option.
- **System Prompts:**

    Confirms last working day and exit reason.
- Validates all dues are cleared.

- **Access Revocation:**
- Notifies IT to deactivate accounts and tools.
- **Archiving:**
- Moves data to archive for audit purposes.
- **Notification:**
- Sends update to Admin and Manager.

Alternative Flows:

- **Pending Dues:**
- Step 3A: Process paused until dues cleared.
- **Exit Not Approved:**
- Step 2A: System alerts if no exit approval attached.

Exception Flows:

- **Record Not Found:**
- HR is prompted to recheck ID.
- **System Crash:**
- Logs error and halts deletion.

Trigger:

- Exit clearance form submission.

Special Requirements:

- GDPR compliance for data deletion.
- Proper access logs of deletion action.

Use Case Name:

Update Employee Details

Primary Actor:

HR Executive

Secondary Actors:

Department Manager

Stakeholders and Interests:

- **HR Executive:** Needs to ensure up-to-date employee data.
- **Department Manager:** Wants current data for work management.
- **Employee:** Expects accurate personal and professional records.

Preconditions:

- Employee already exists in the system.
- HR is logged in with permissions.
- Reason for update is valid (e.g., promotion, address change).

Postconditions:

- Updated details are stored.
- Timestamped audit record is created.
- Relevant departments are notified.

Main Success Scenario (Basic Flow):

- **Trigger:** Employee submits request or HR receives update mandate.
- **HR Action:**

- Logs in and locates employee record.

  Clicks on “Edit” and makes necessary changes.
- **System Validation:**
- Checks for valid inputs (e.g., phone number format).
- **Audit Logging:**
- Saves a snapshot of old vs. new data.
- **Save Changes:**
- Confirms and commits changes.
- **Notifications:**
- Sends update to relevant parties (e.g., payroll, IT).

Alternative Flows:

- **Unauthorized Update:**
- Step 2A: If user doesn’t have edit rights, system blocks changes.
- **Invalid Inputs:**
- Step 3A: Prompts HR to correct data format.

Exception Flows:

- **Update Conflict:**
- Concurrent update attempt results in merge conflict.
- **Missing Justification:**
- System prompts for update reason before saving.

Trigger:

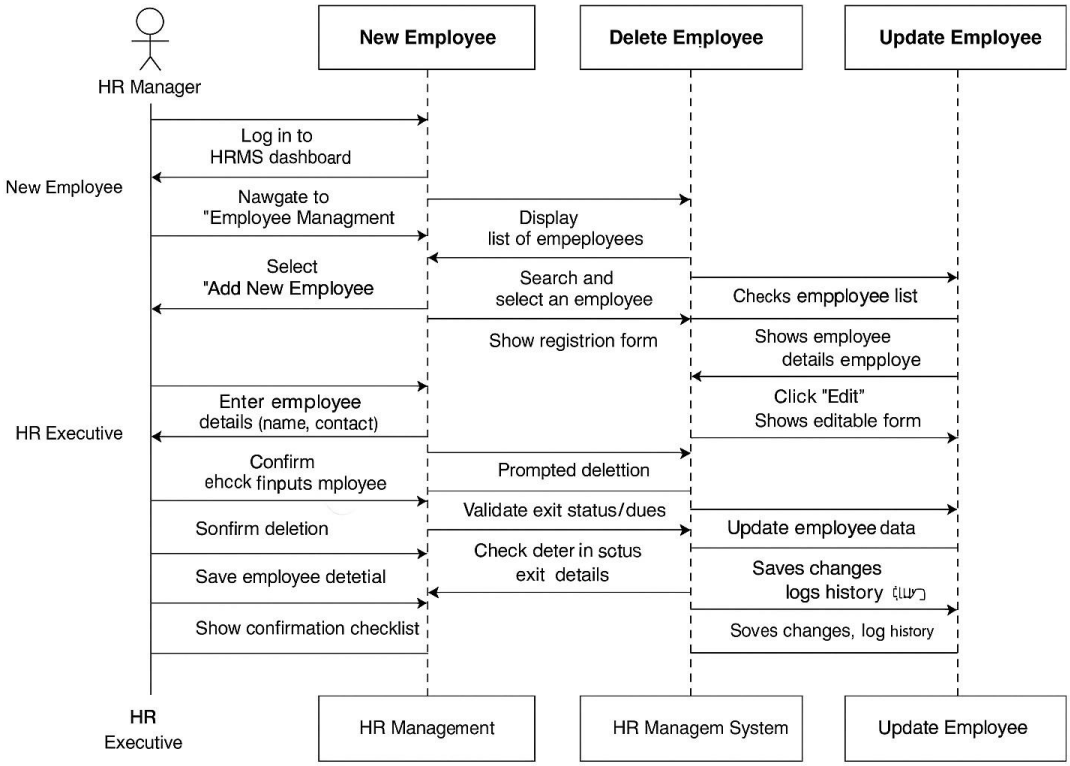
- Employee request or organizational change.

Special Requirements:

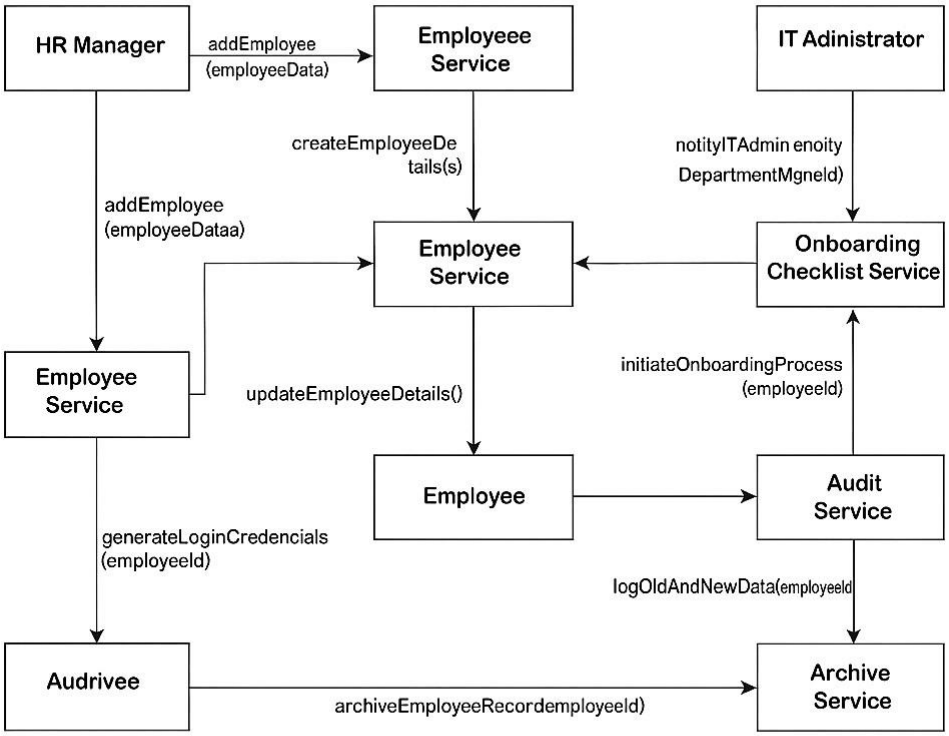
- Editable logs for compliance.
- Restricted updates (e.g., salary changes require approval).

- Backup before every major change.

System Sequence Diagram:

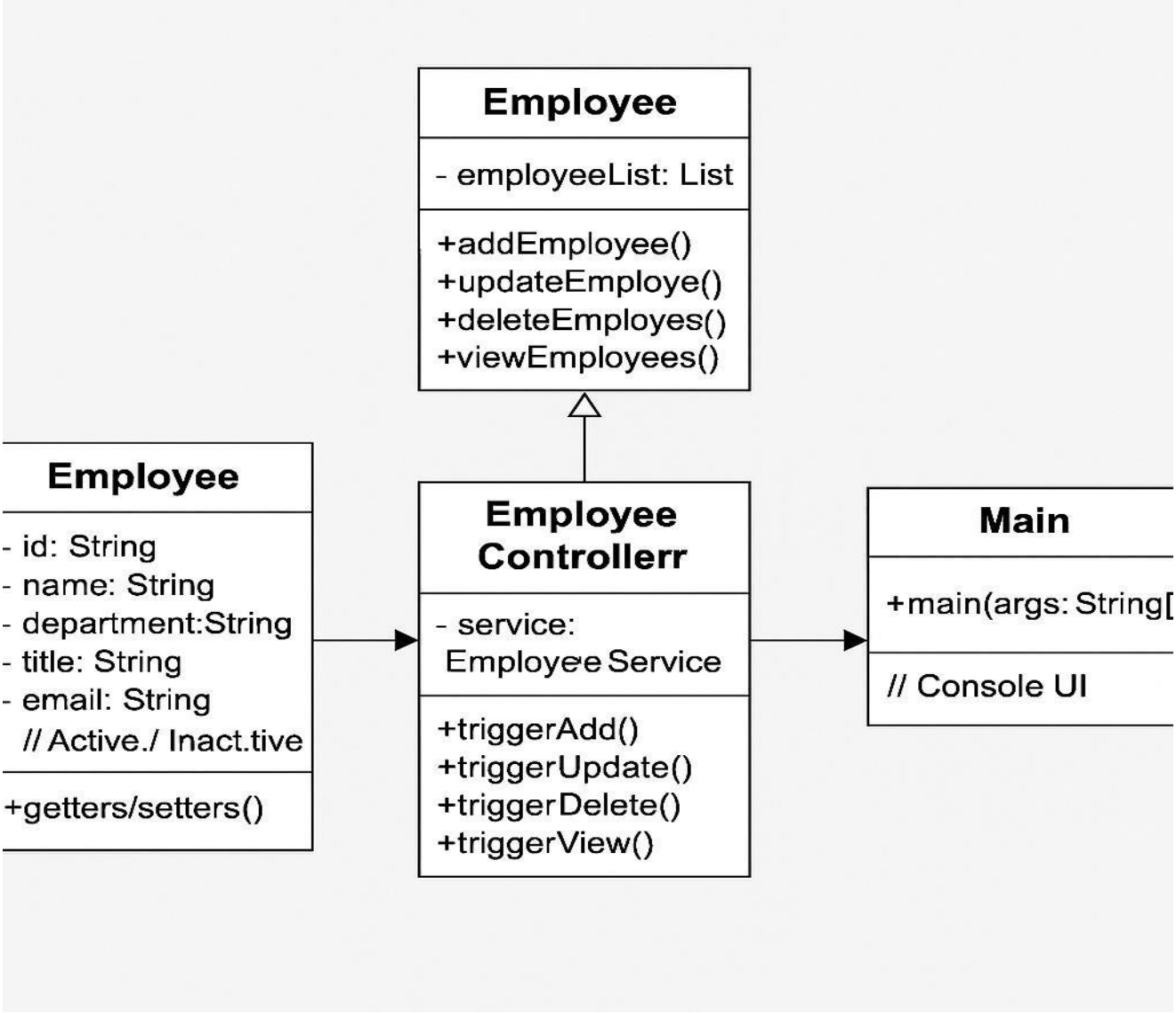


System Event Design via collaboration diagram:

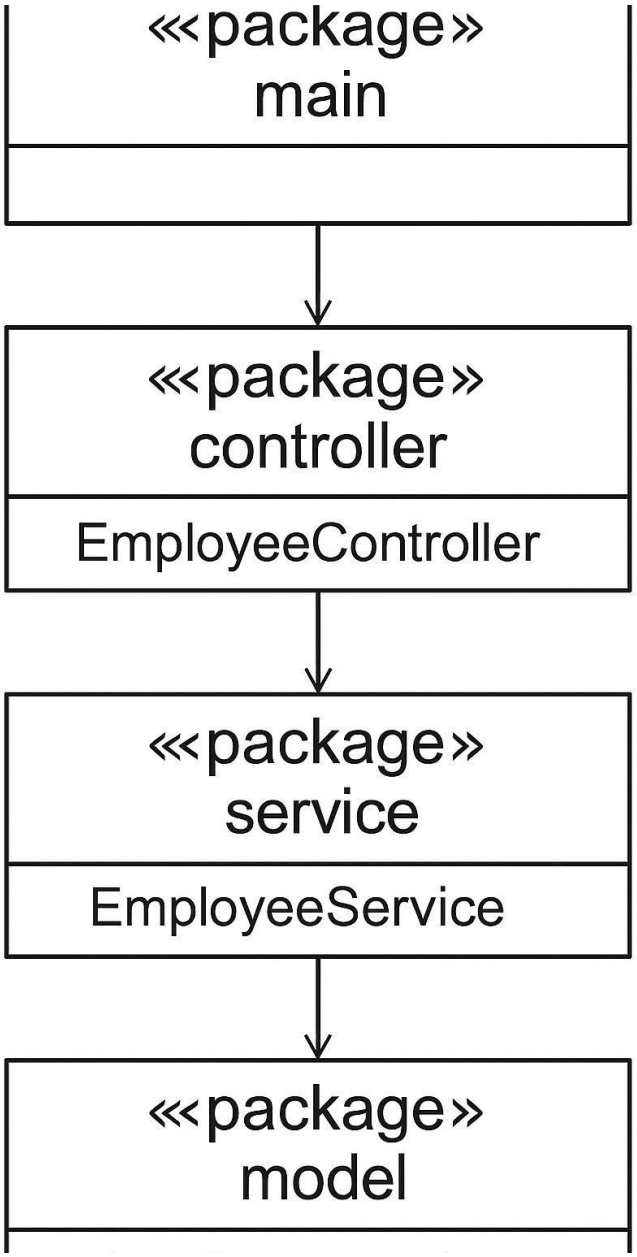


System Event Design via Collaboration Design (GRASP)

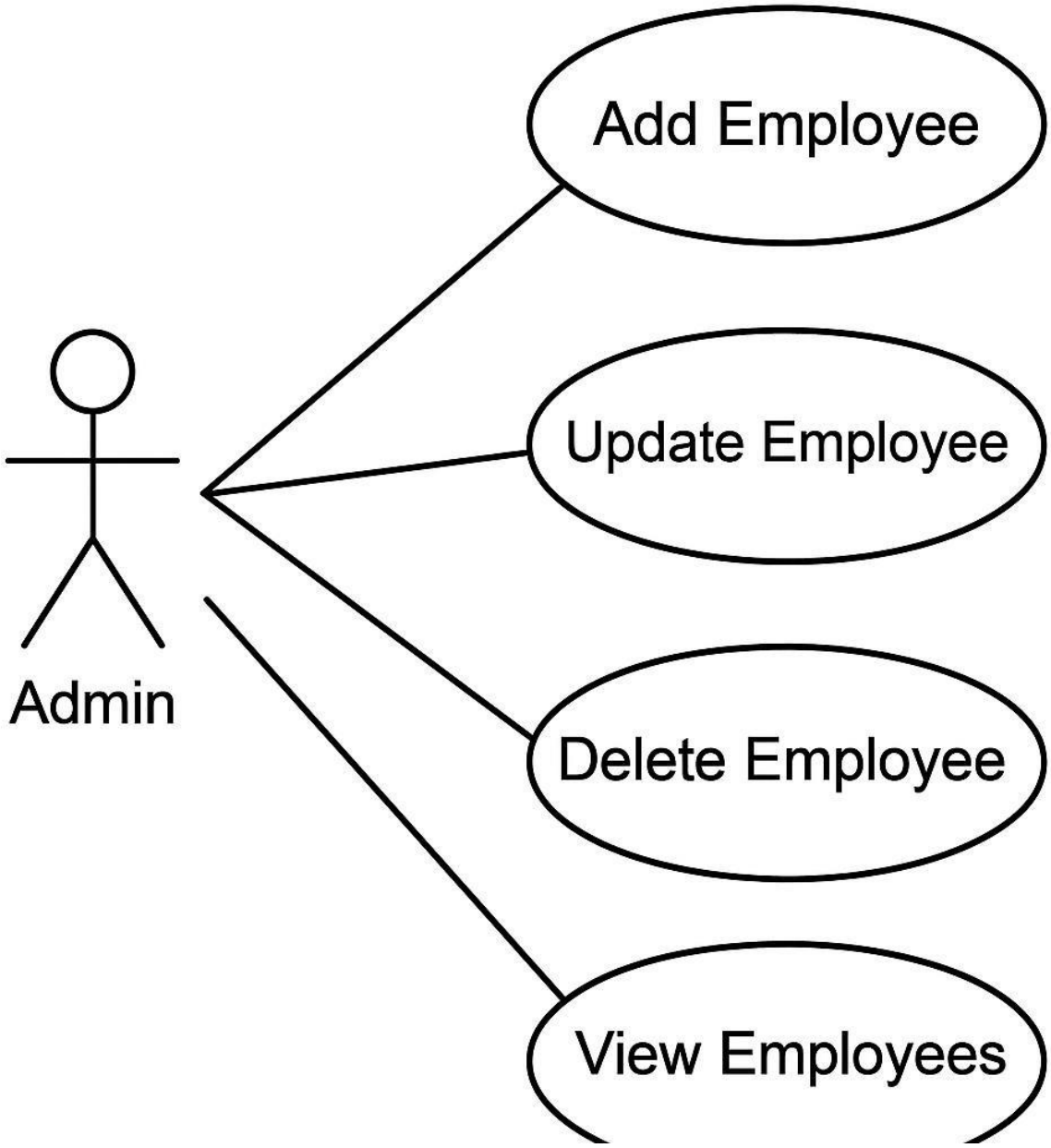
Class diagram:



Package diagram:



Use case diagram:



Prototype(example coding) – Employee Management:

```
// === MODEL ===
    class Employee
    {
```

```
    private String id;
    private String name;     private String
department;          private  String  title;
private String email;       private String
status; // Active / Inactive

    // Constructor, Getters, Setters
}

// === SERVICE ===
      class
      EmployeeService {
    // Store employees in a simple list or map
    // Create new employee
    // Update existing employee
    // Delete (or mark as inactive)
    // View all employees
}

// === CONTROLLER ===
class EmployeeController {     private EmployeeService
      service = new EmployeeService();
```

```
    // Methods to trigger add, update, delete, and view operations
}

// === VIEW (for console interaction, basic UI)
    === public class Main {     public static void
    main(String[] args) {
      // Initialize controller
      // Menu-driven interaction for:
      // 1. Add Employee
      // 2. Update Employee
      // 3. Delete Employee
      // 4. View Employees
      // 0. Exit
   }
}
```

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.

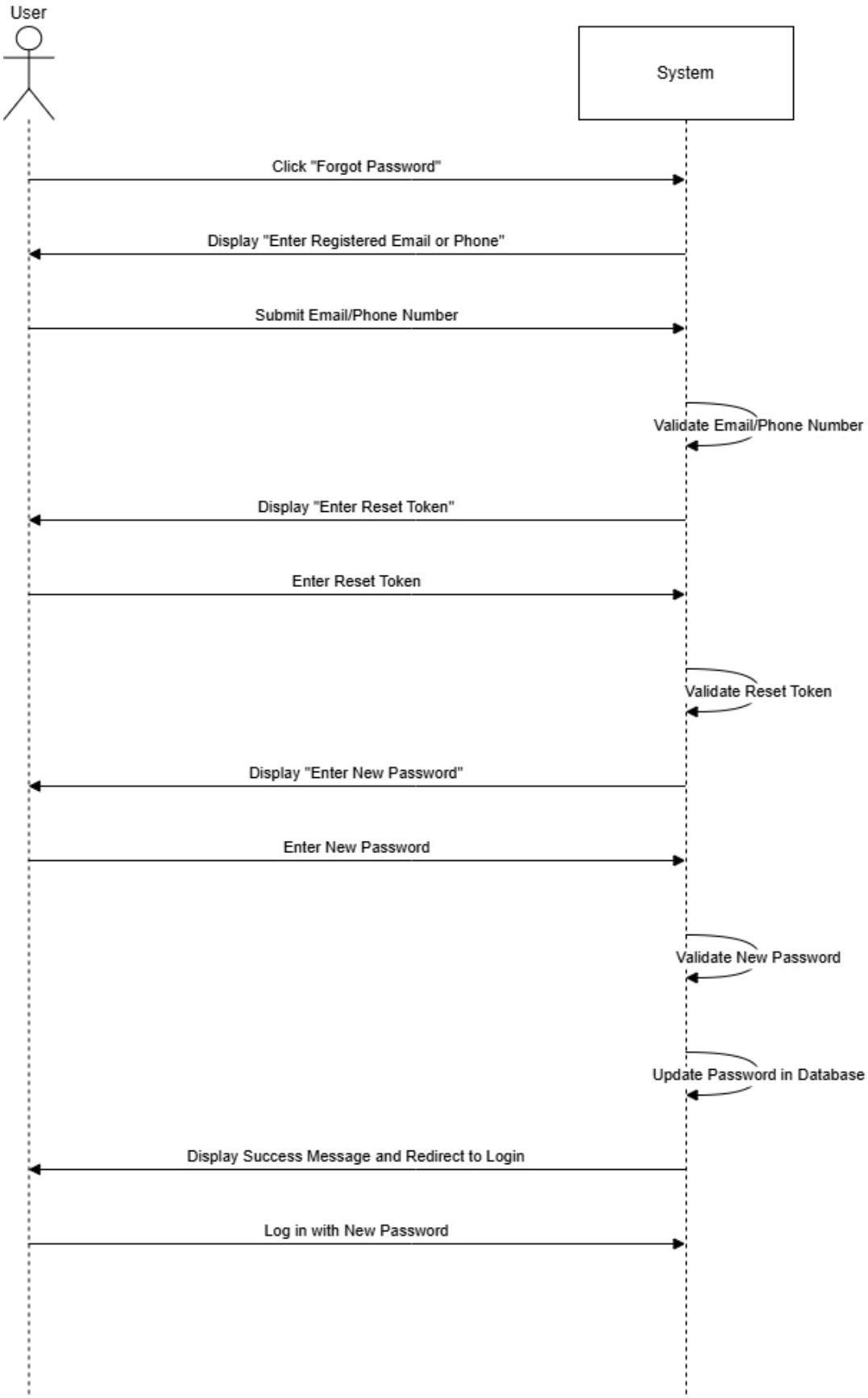
Frequency of Use

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

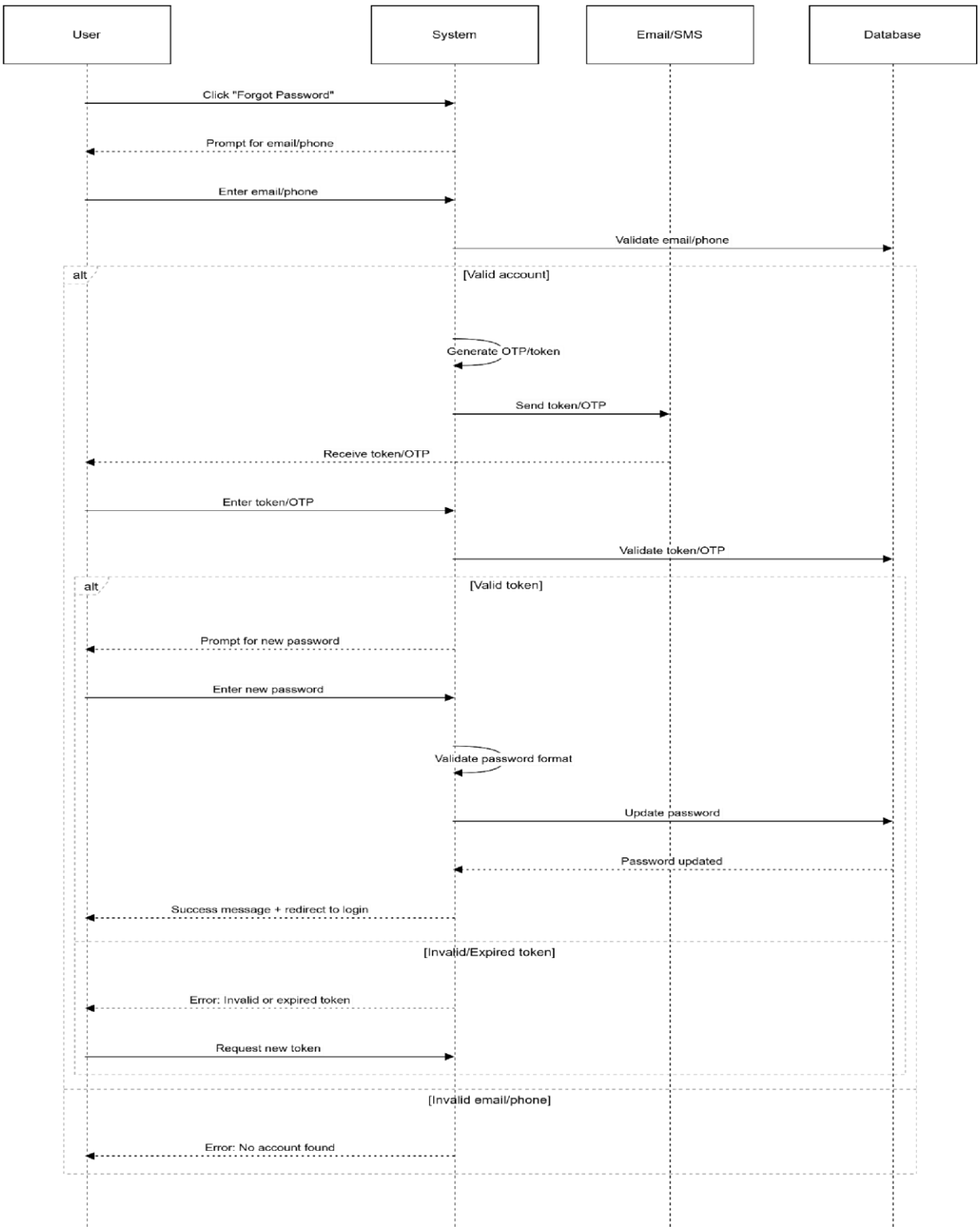
Open Issues

- 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:


User
System
Email/SMS
Database
Click "Forgot Password"
Prompt for email/phone
Enter email/phone
Validate email/phone
alt
[Valid account]
Generate OTP/token
Send token/OTP
Receive token/OTP
Enter token/OTP
Validate token/OTP
alt
[Valid token]
Prompt for new password
Enter new password
Validate password format
Update password
Password updated
Success message + redirect to login
[Invalid/Expired token]
Error: Invalid or expired token
Request new token
[Invalid email/phone]
Error: No account found

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.

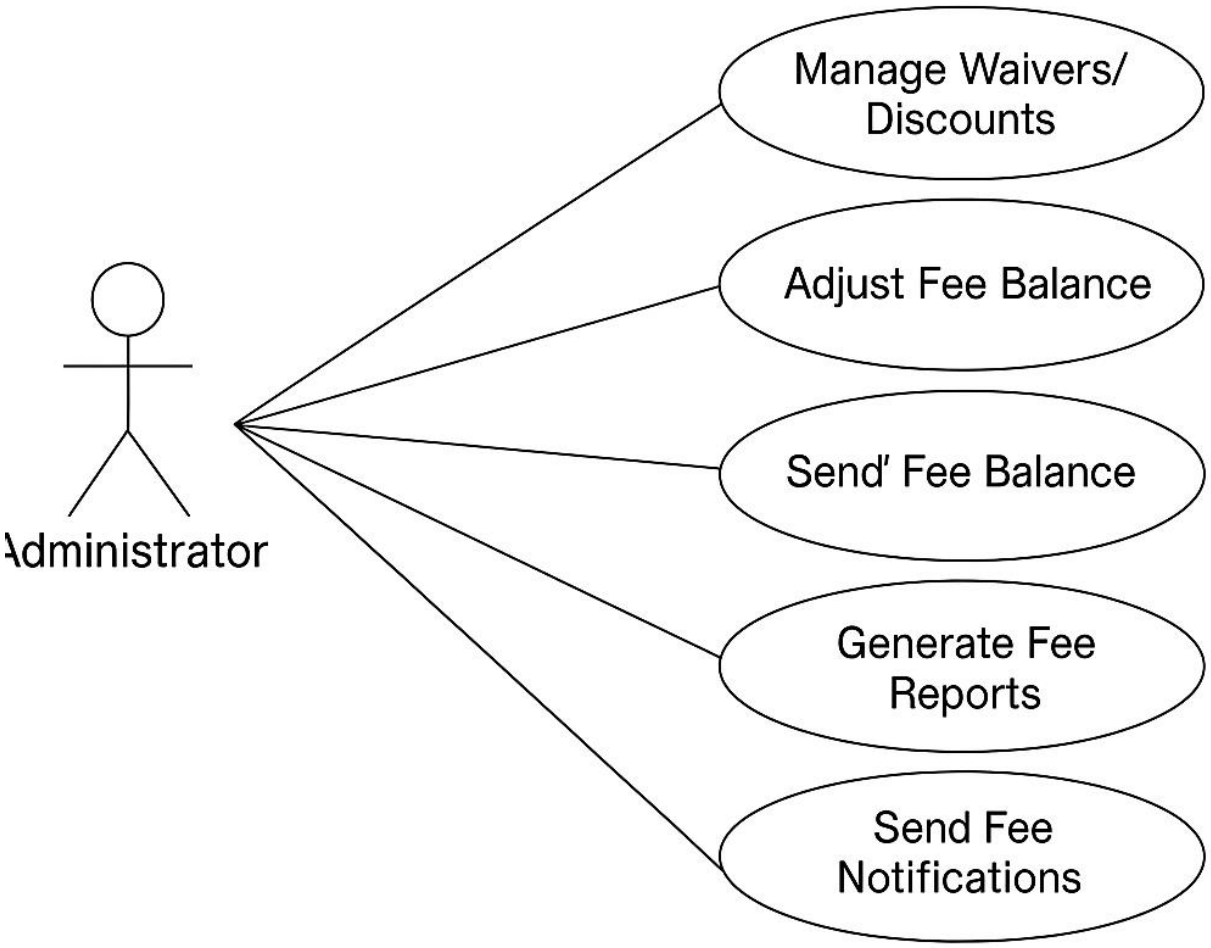
Extension Points:

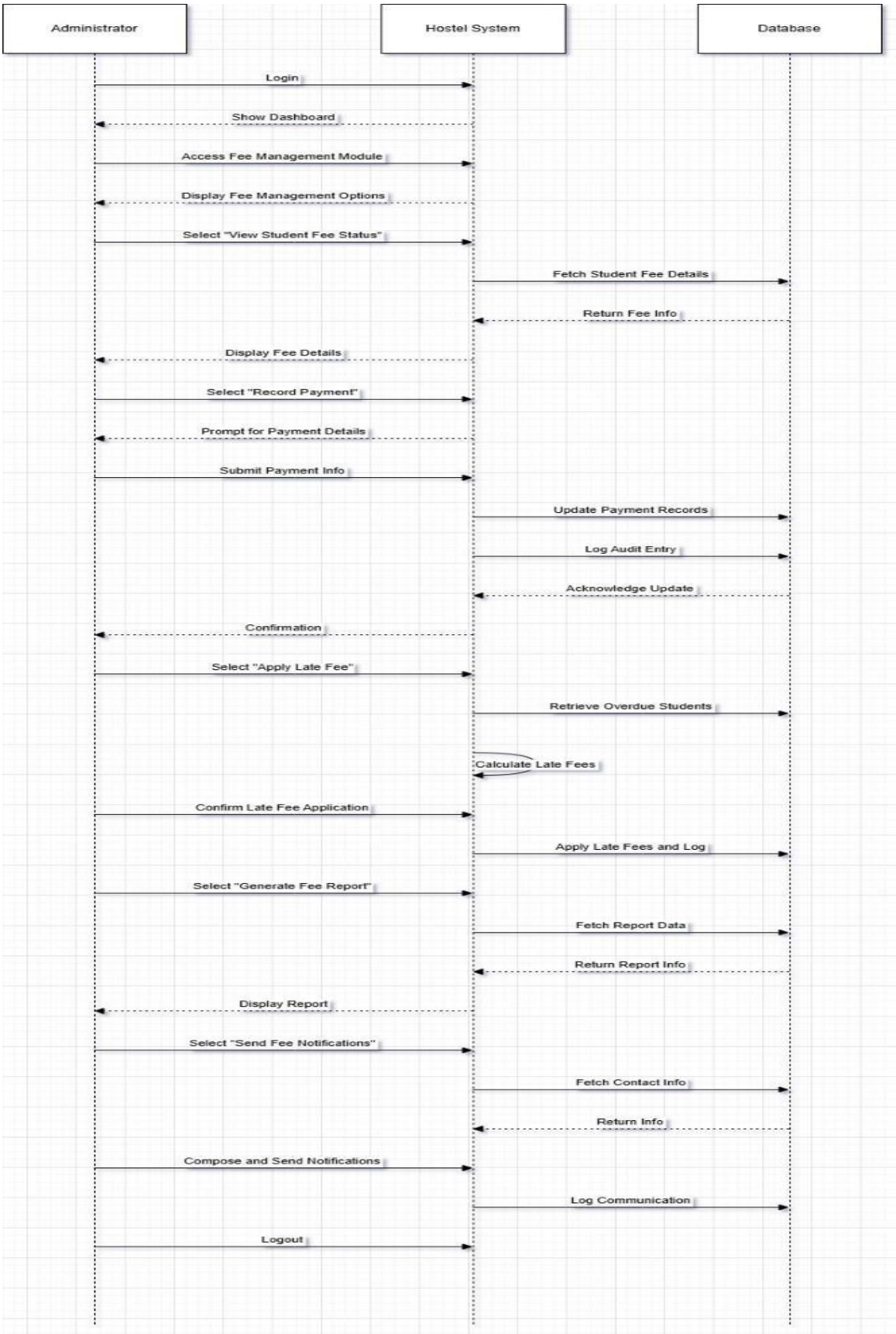
- Integration with accounting software for automated financial reconciliation.
- Implementation of automated installment reminders for students.
- Features for generating invoices or fee receipts for students.
- Workflow for handling disputed fee payments.
- Integration with student information systems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

IC_Admin_0_001

STUDENT FEE MANAGEMENT




Administrator
Hostel System
Database
Login
Show Dashboard
Access Fee Management Module
Display Fee Management Options
Select "View Student Fee Status"
Fetch Student Fee Details
Return Fee Info
Display Fee Details
Select "Record Payment"
Prompt for Payment Details
Submit Payment Info
Update Payment Records
Log Audit Entry
Acknowledge Update
Confirmation
Select "Apply Late Fee"
Retrieve Overdue Students
Calculate Late Fees
Confirm Late Fee Application
Apply Late Fees and Log
Select "Generate Fee Report"
Fetch Report Data
Return Report Info
Display Report
Select "Send Fee Notifications"
Fetch Contact Info
Return Info
Compose and Send Notifications
Log Communication
Logout

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:

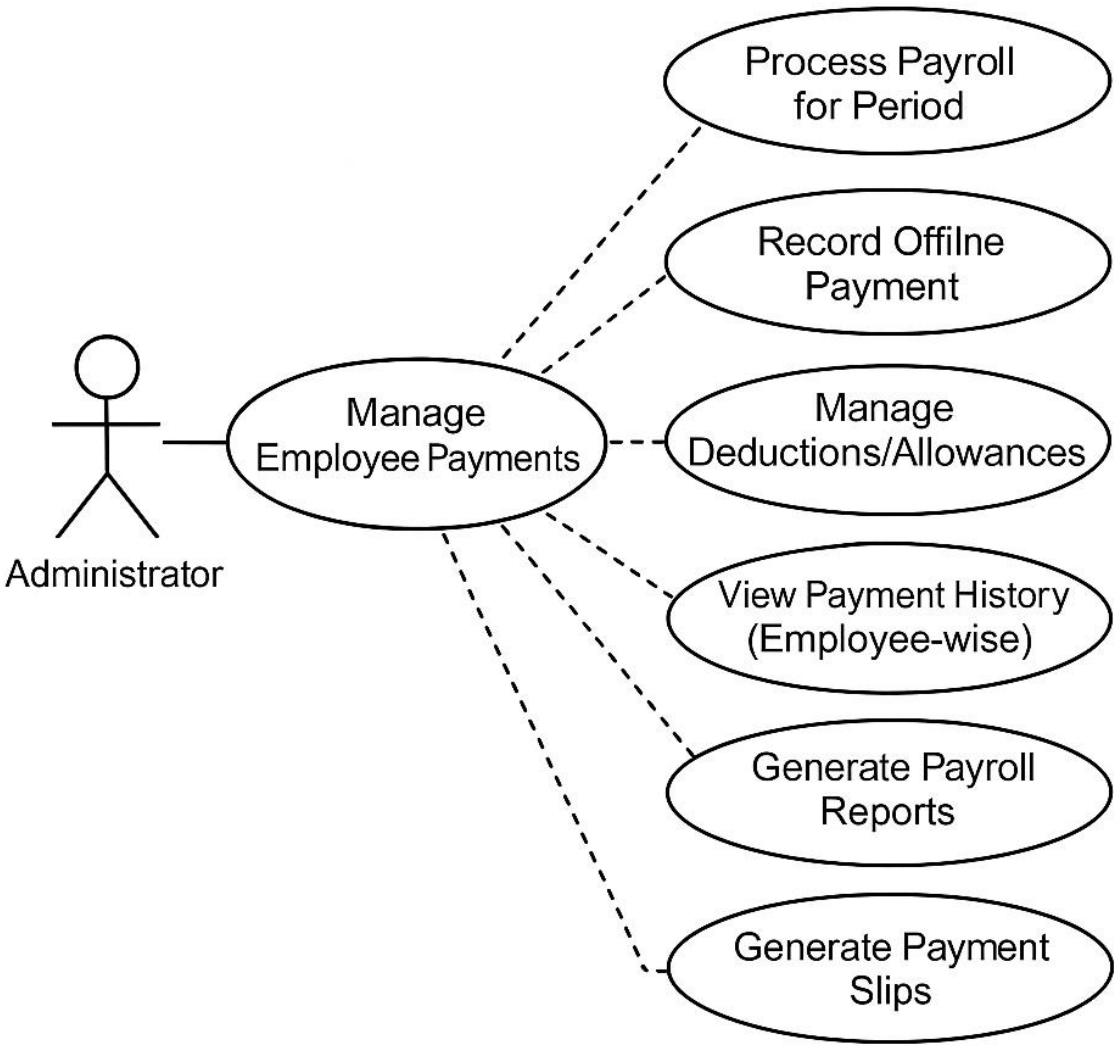
- **System Errors:** If the system encounters any technical errors (e.g., database issues, bank integration failures), it will display an appropriate error message.
- **Insufficient Permissions:** If the Administrator does not have the necessary permissions to perform a specific action, the system will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

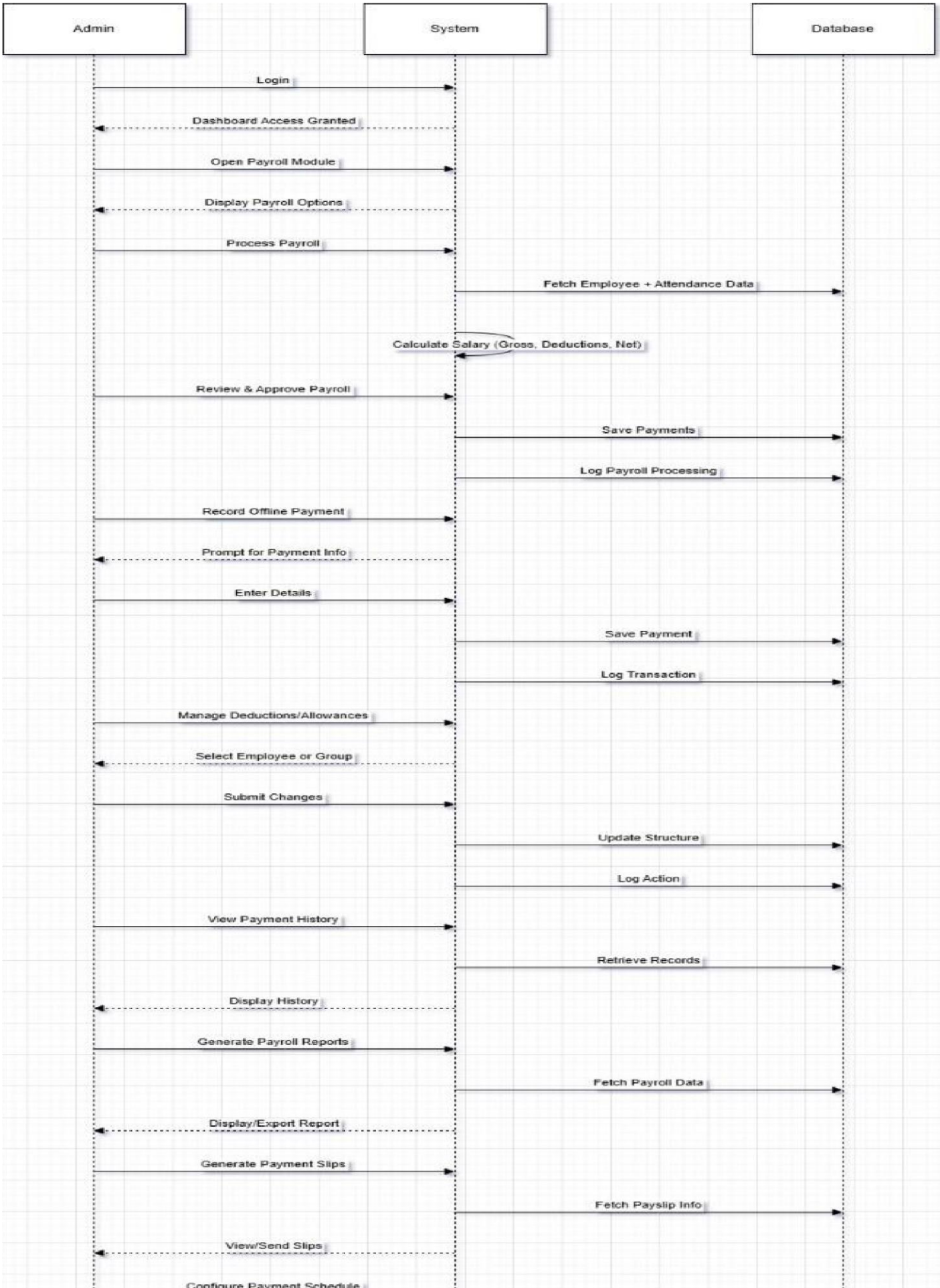
Extension Points:

- 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.

Manage Employee Payments




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):

1. **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.
2. **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.

3. **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."
4. **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."
5. **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."
6. **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.

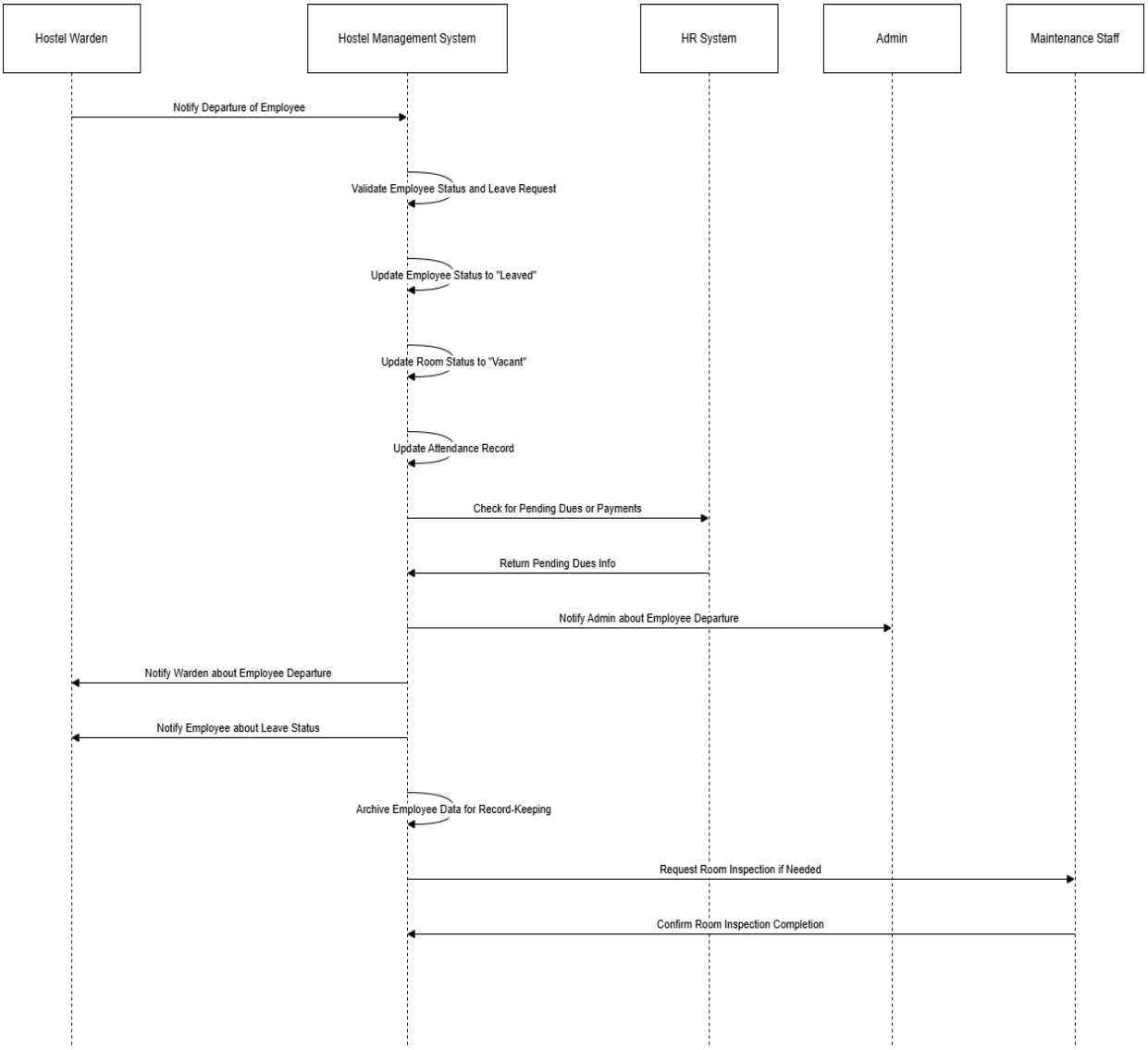
Trigger:

- 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