**PROJECT TITLE: HOTEL MANAGEMENT SYSTEM**

Contents

Nouman Khan

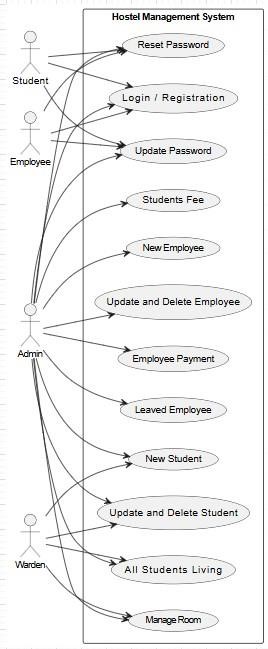
Sardar Zain

Abdullah sajid

**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.

1. The system has accurate records of all rooms, including room numbers, current occupants, and room status.
2. The user (Hostel Manager/Admin) is logged in and has the necessary permissions to manage rooms.
3. 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):** o 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**. o 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):** o If the room requires maintenance (e.g., cleaning, repairs), the Hostel Manager can select the **maintenance option**. o The system notifies the **Maintenance Staff** about the required tasks and provides them with room details. o 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:** o **Step 5A:** If the room is in need of cleaning or repairs, the system notifies the **maintenance staff** to schedule the necessary tasks. o **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. o 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:** o 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.

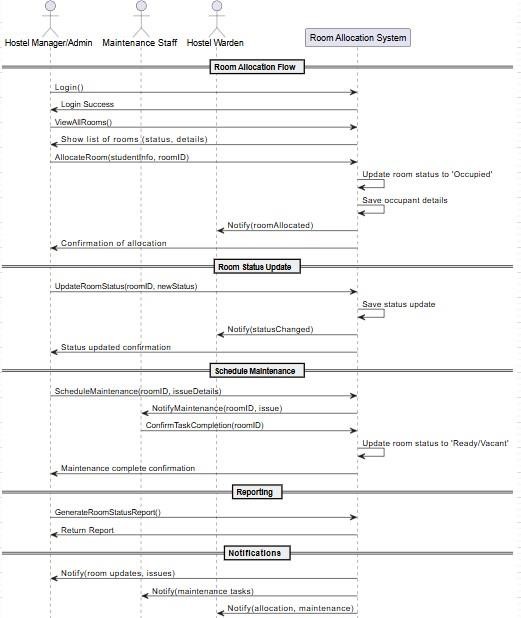
**Trigger:**

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

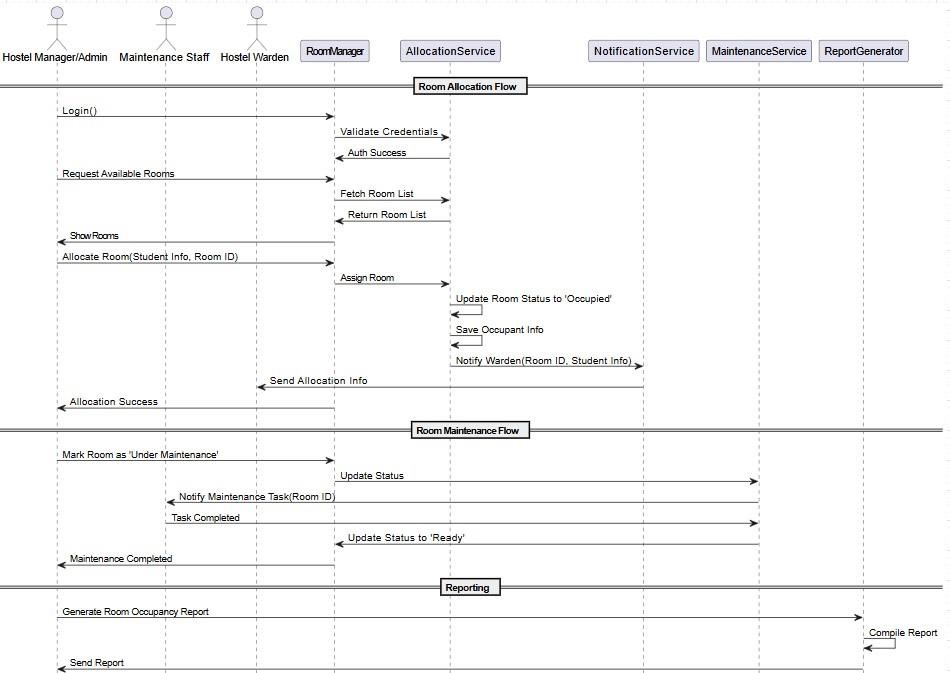
**Special Requirements:**

* **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.

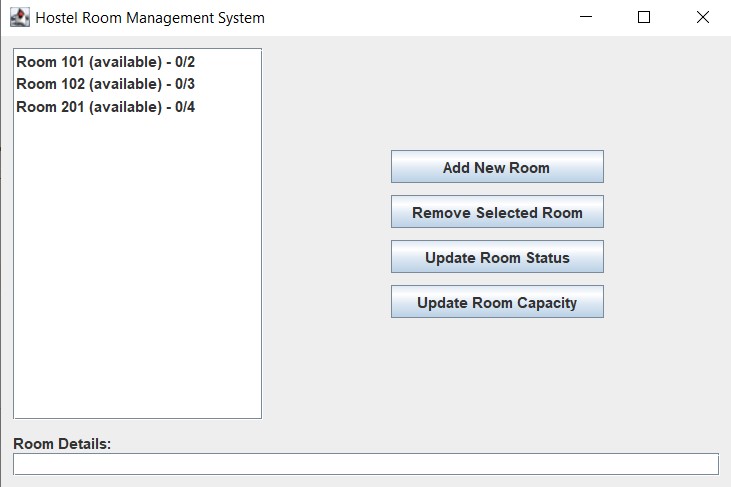
**System Sequence Diagram:**



**Sequence Diagram:**



**UI Prototype:**

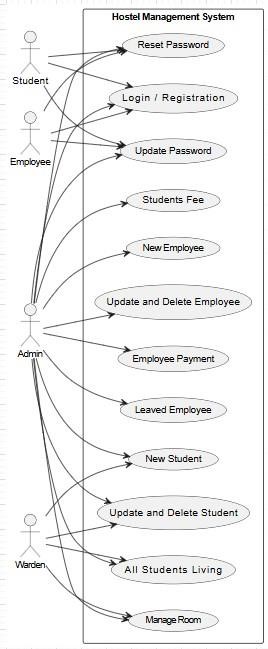


NAME: SARDAR ZAIN

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):**

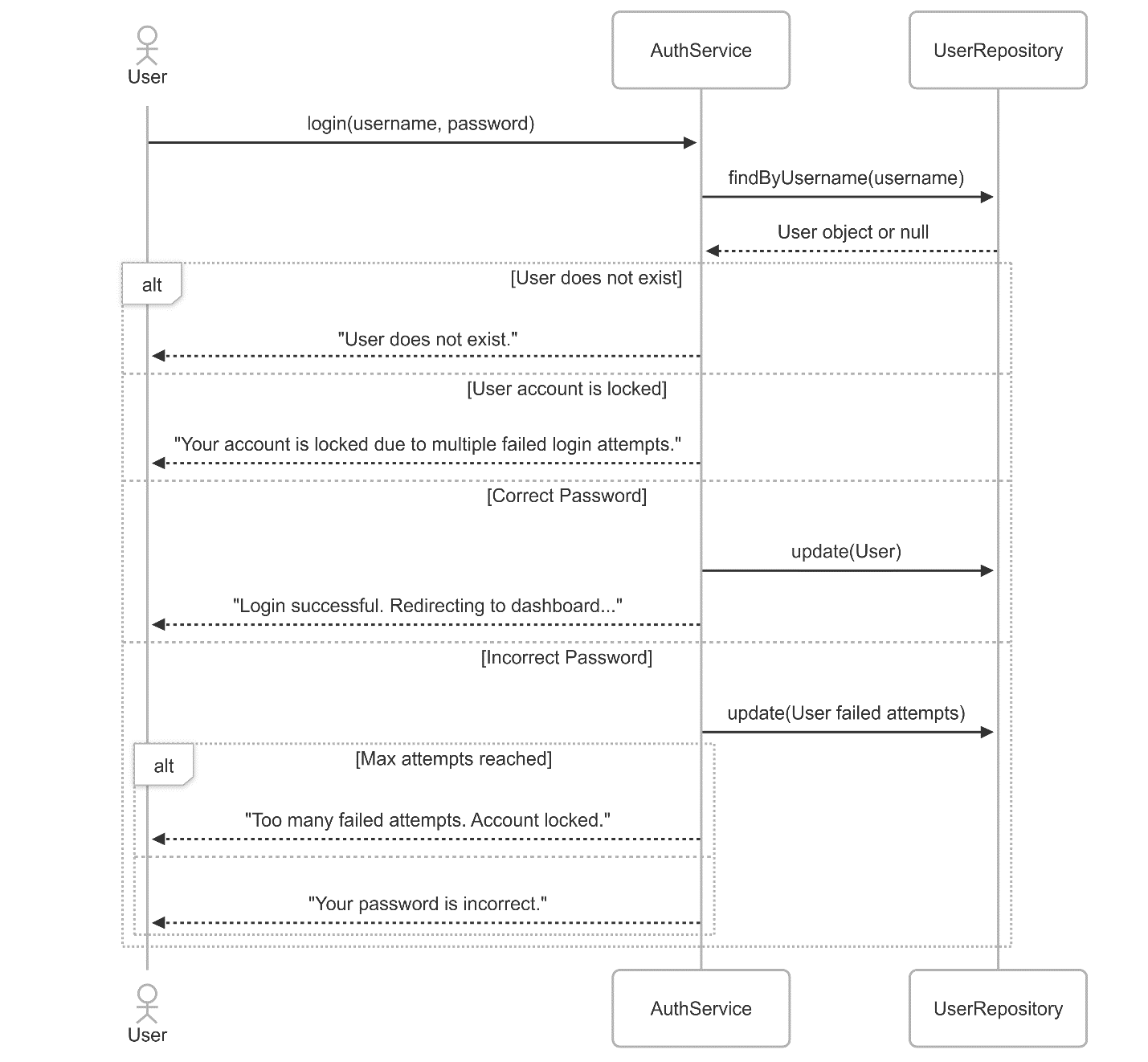
* **A1: Invalid Username:**
  + Condition: The entered username does not exist.
  + System Response: “User does not exist.”
* **A2: Invalid Password:**
  + Condition: The username exists but the password is incorrect.
  + System Response: “Your password is incorrect.”

**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.

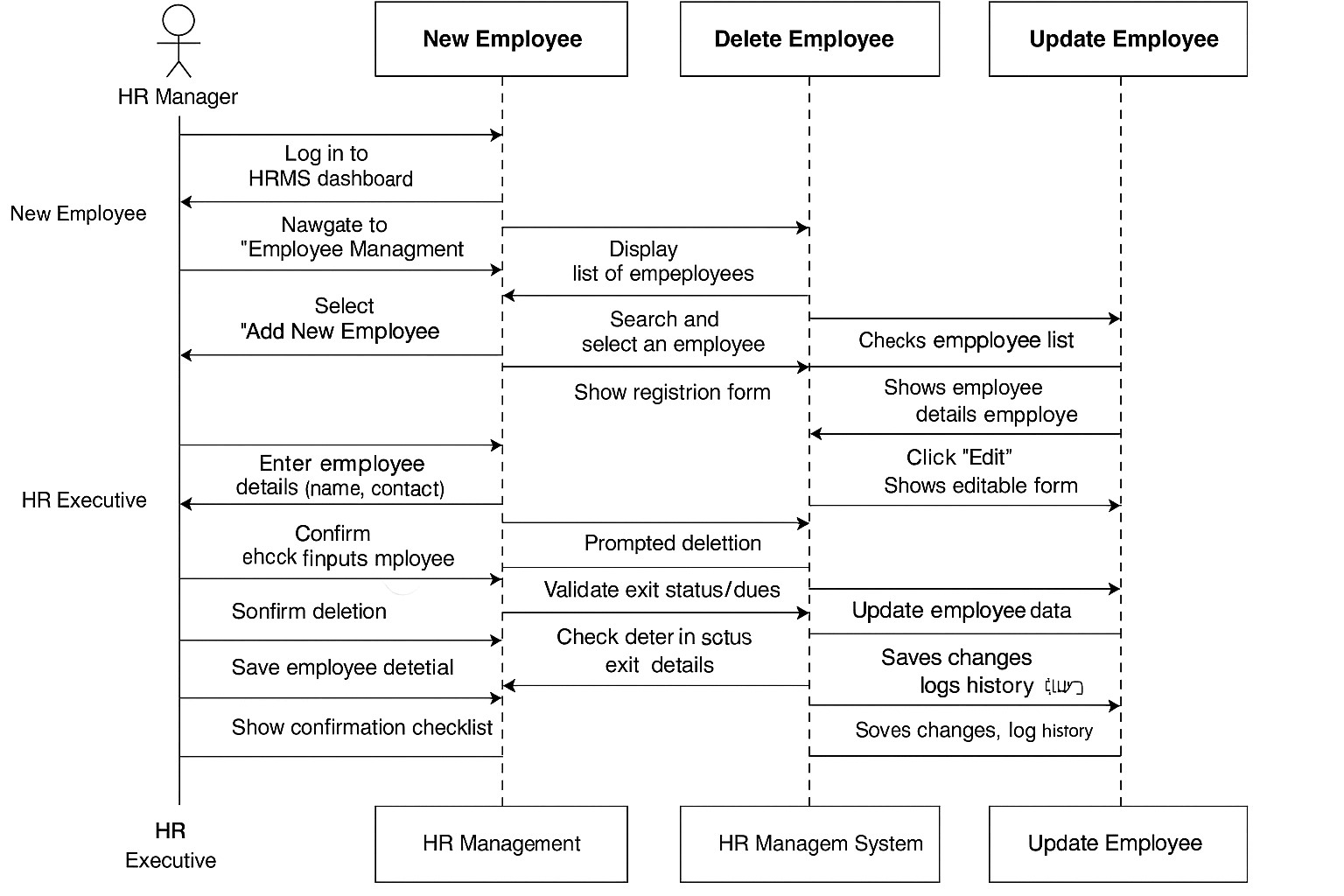
**Trigger:**

* Employee request or organizational change.

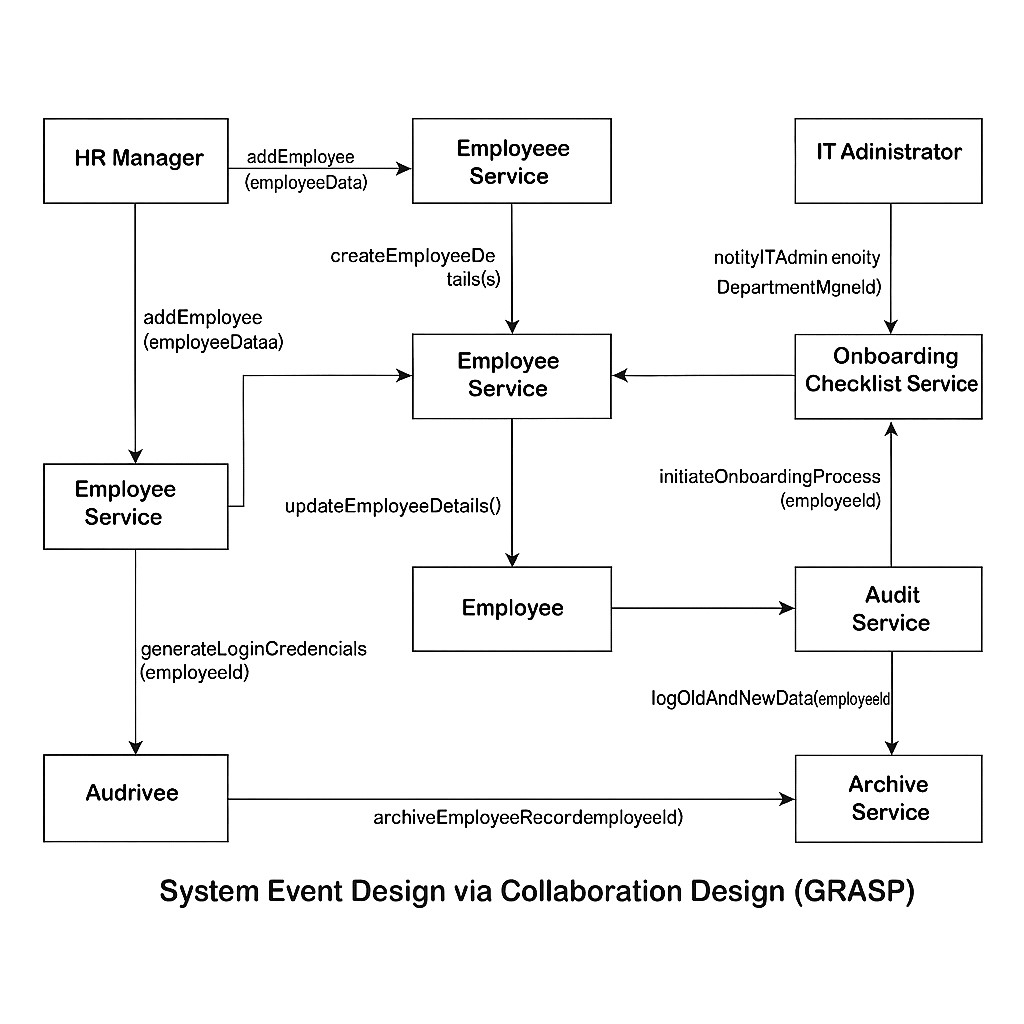
**Special Requirements:**

* Editable logs for compliance.
* Restricted updates (e.g., salary changes require approval).
* Backup before every major change.

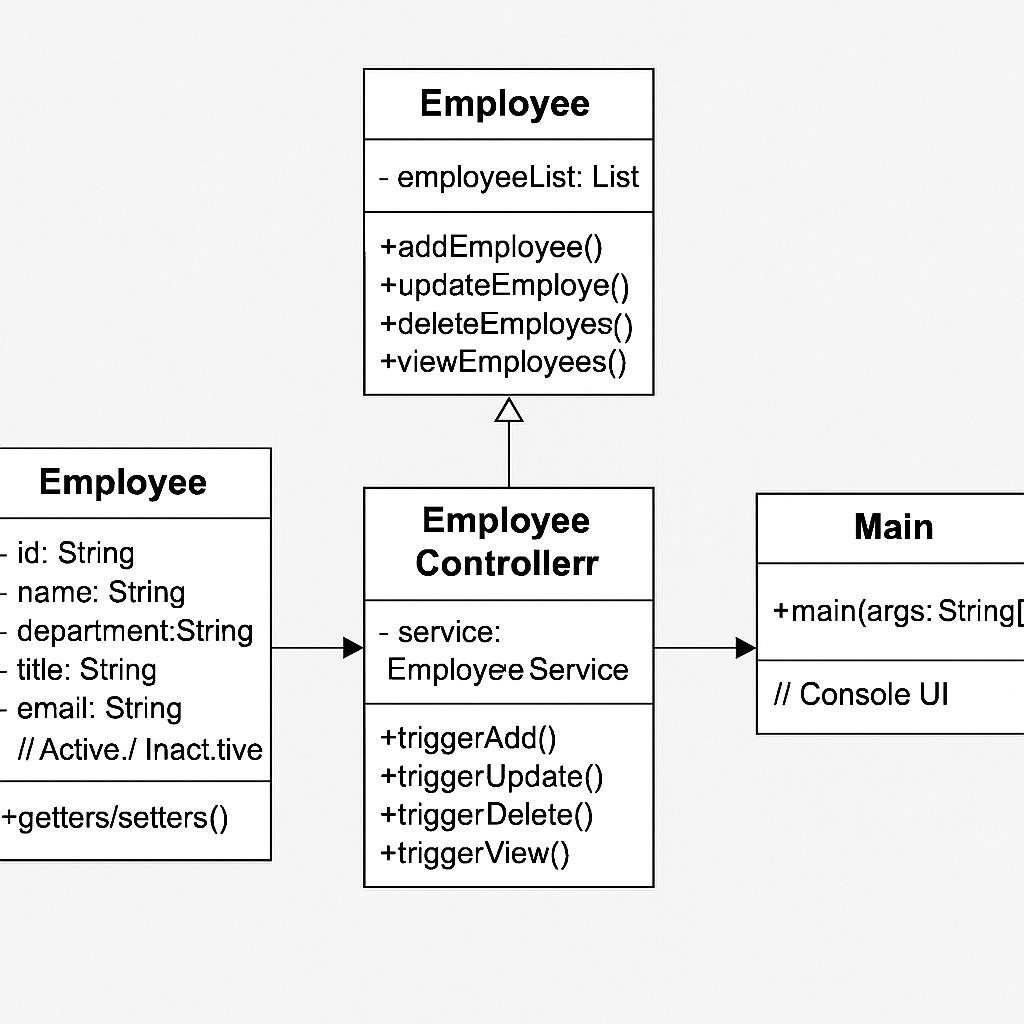
**System Sequence Diagram:**



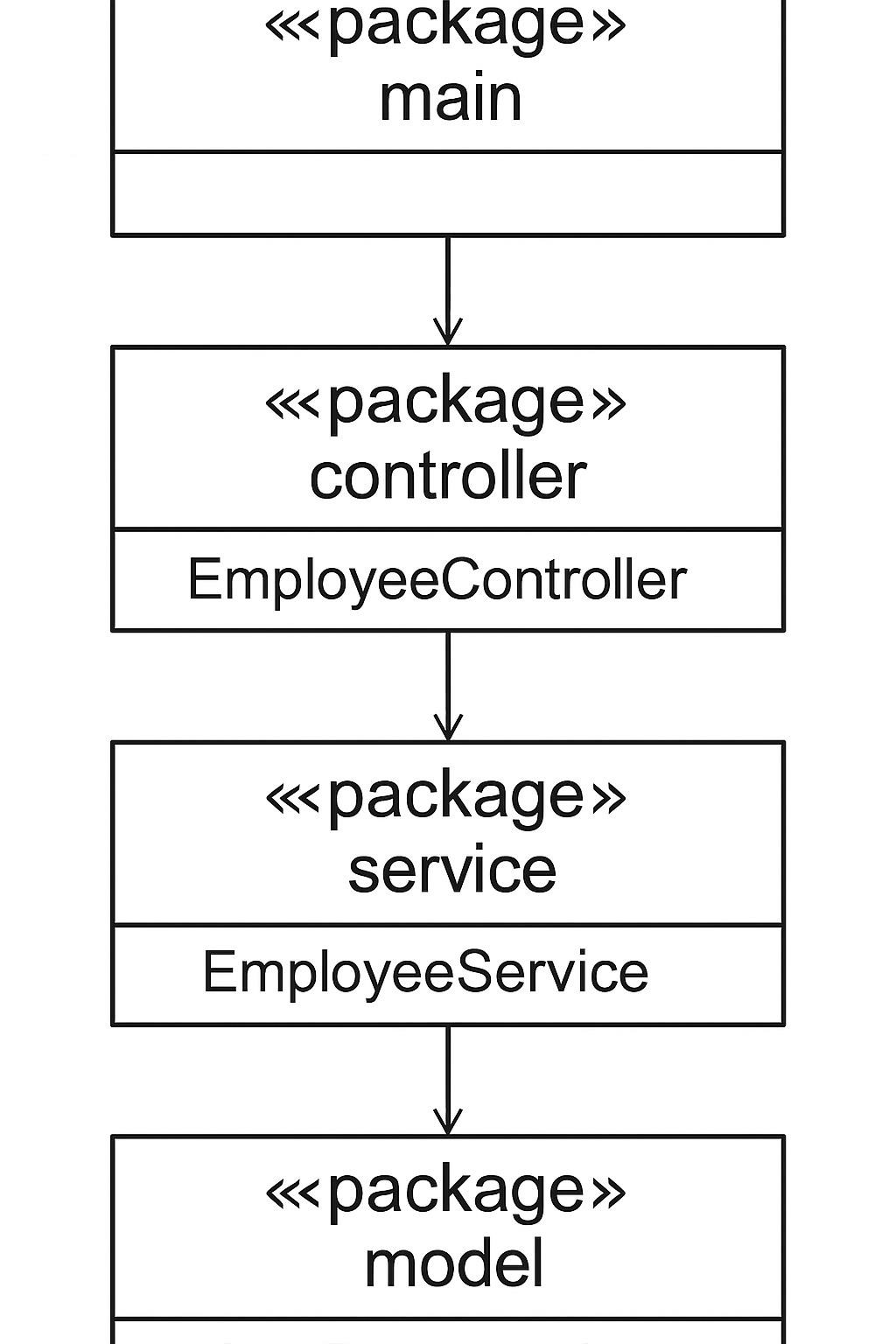
**System Event Design via collaboration diagram:**



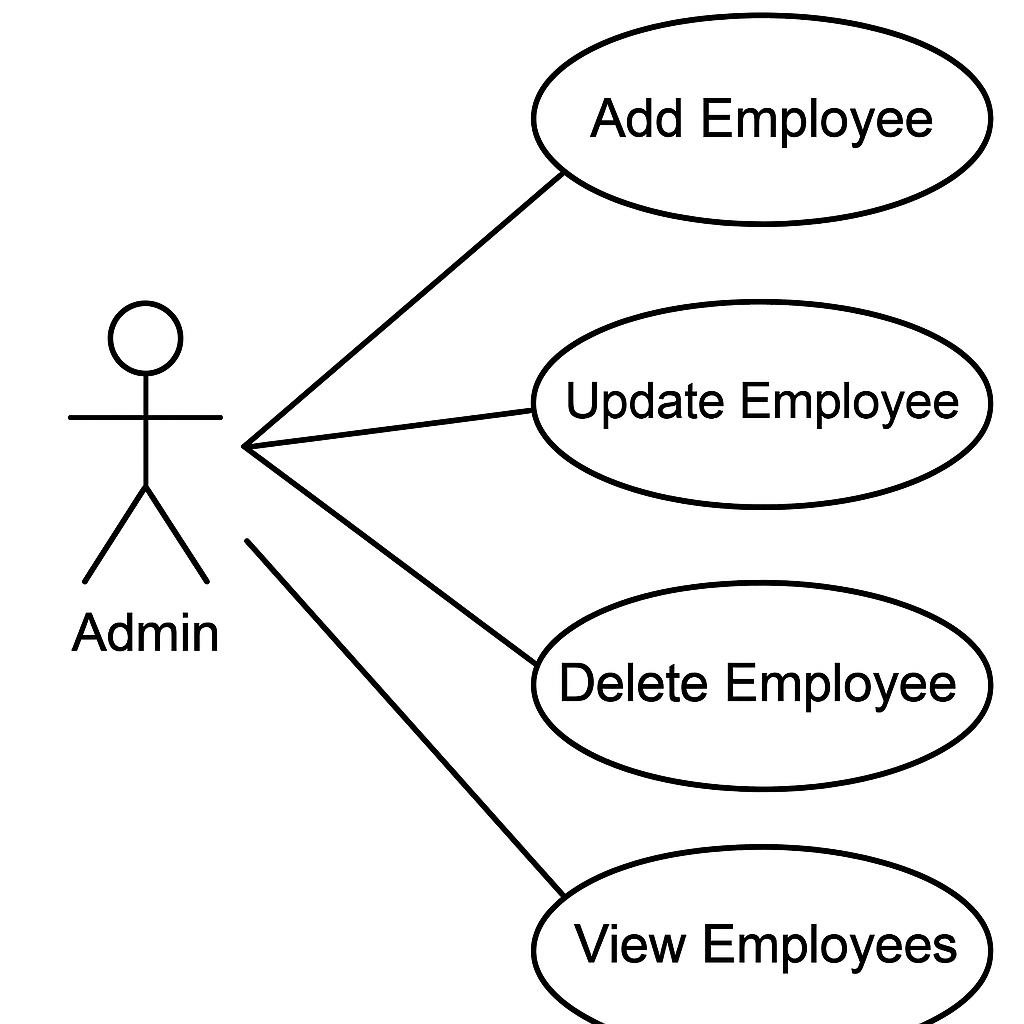
**Class diagram:**



**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

}

}