Use Case Name:

Employee Management

Primary Actor:

Hostel Manager / Admin

Secondary Actors:

HR Department, Employee

Stakeholders and Interests:

- **Employee:** Expects accurate storage and updates of personal and professional data.
- **HR Department:** Needs updated employee information for payroll, clearance, and administrative actions.
- **Hostel Manager/Admin:** Needs efficient tools to manage employee records (add, update, delete, view).
- **System:** Must ensure data integrity, prevent duplication, and allow easy record handling.

Preconditions:

1. The system is running, and the user has the necessary permissions.
2. The GUI (EmployeeManagementGUI) is accessible from the main menu.
3. The EmployeeManager class is initialized and holds employee data.
4. At least one employee may already be present (default entries).

Postconditions:

1. Employee records are added, updated, deleted, or retrieved successfully.

2. The employee list is always in sync with the current operations.
3. Duplicates are not allowed by employee name (handled using .equals()).
4. The GUI reflects any changes to the employee list immediately.

Main Success Scenarios (Basic Flows)

➤ Use Case 1: Add New Employee

Trigger: Hostel Manager clicks the "Add" button after entering employee details in the GUI.

Steps:

1. Admin opens the Employee Management GUI.
2. Enters name, email, phone, and experience in the form.
3. Clicks the **Add** button.
4. System checks if:
    - All fields are filled.
    - An employee with the same name does not already exist.
5. If valid, the new employee is added to the internal list.
6. The employee list in the GUI is updated to reflect the addition.

Postcondition: New employee is added and visible in the employee list.

➤ Use Case 2: Update Existing Employee

Trigger: Hostel Manager selects an employee from the list, edits their details, and clicks the "Update" button.

Steps:

1. Admin selects an employee from the JList.

2. The employee’s details populate the form fields.
3. Admin modifies the required fields.
4. Clicks **Update**.
5. System validates:
    - The selected employee exists.
    - The new name is not a duplicate (unless unchanged).
6. If valid, system updates the employee’s record in the list.

Postcondition: Employee details are updated in the list and in memory.

➤ Use Case 3: Delete Employee

Trigger: Hostel Manager selects an employee from the list and clicks the "Delete" button.

Steps:

1. Admin selects an employee from the JList.
2. Clicks the **Delete** button.
3. System removes the selected employee from the list.
4. GUI is updated to reflect deletion.

Postcondition: Employee is removed from system and no longer visible in the list.

➤ Use Case 4: View Employees

Trigger: Admin opens the Employee Management GUI.

Steps:

1. The GUI calls employeeManager.getAllEmployees() on load.
2. The returned list is displayed in the JList component.

Postcondition: The current state of employee records is visible in the GUI.

Alternative Flows (Extensions)

❖ Duplicate Employee Name on Add

- System prevents addition if another employee with the same name (case-insensitive) already exists.
- Displays an error message: "Employee already exists."

❖ Duplicate Name on Update

- If updated name matches another employee's name (not the selected one), update is denied.
- Displays error message: "Another employee with this name already exists."

❖ No Employee Selected for Update/Delete

- If no item is selected from the JList:
    - System shows error message: "Please select an employee to update/delete."

❖ Empty Fields on Add/Update

- System shows a message: "Please fill all fields" if any input field is empty.

Exception Flows

✗ System Error (e.g., Null Pointer)

- If internal errors occur during actions, a general error message is shown.
- GUI remains operational.

✗ Employee Record Not Found (during Update/Delete)

- If selected employee is removed externally or doesn't exist anymore, update/delete will fail silently or be ignored.

Trigger

- Any CRUD operation initiated by the user (Admin/Manager) via the Employee Management GUI.

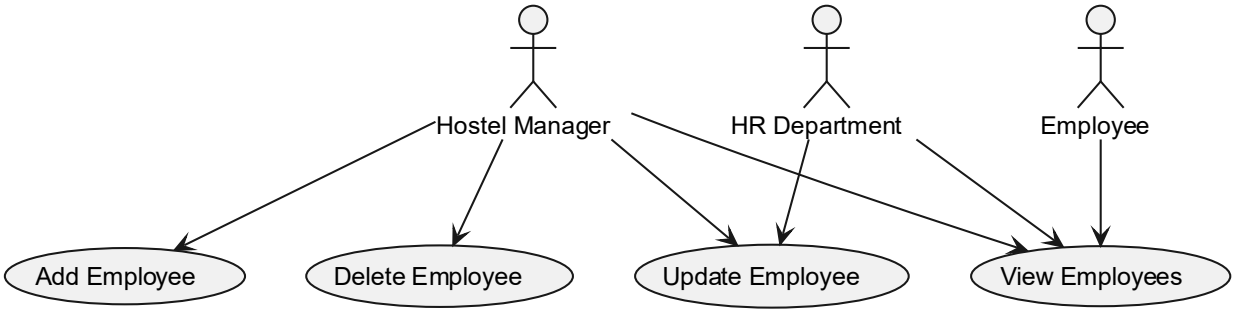
Special Requirements

- **Data Validation:** Name uniqueness is enforced via equals() and hashCode().
- **Live Updates:** GUI automatically reflects current employee list.
- **Error Handling:** User-friendly messages for invalid operations.
- **Security (Logical):** Only authorized users (Admin/HR) can access this module (assumed by design).
- **Usability:** Dark-themed UI with modern styling for accessibility and visibility.

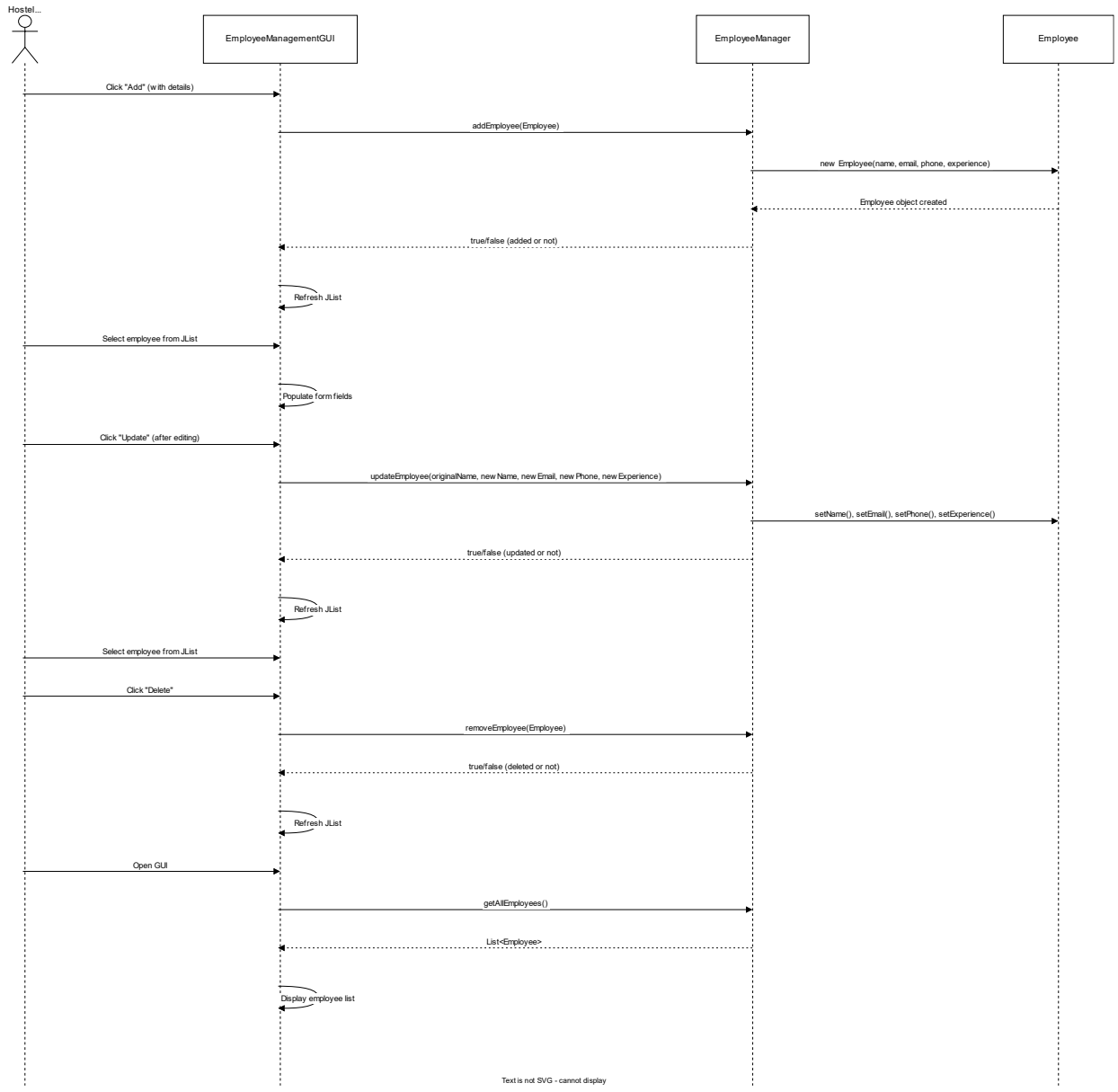
Design Principles and Patterns Used

Concept / Principle	Description
Single Responsibility Principle (SRP)	**Each class has a clearly defined responsibility:** • Employee handles only employee data (Model). • EmployeeManager manages logic (Controller). • EmployeeManagementGUI handles UI (View).
Low Coupling	EmployeeManagementGUI interacts with EmployeeManager but does not manage data directly. This separation improves modularity and testability.
High Cohesion	**Each class focuses only on related functionality:** • Employee for employee data. • EmployeeManager for employee logic. • GUI for user interaction.
Model-View-Controller (MVC) Pattern	**A lightweight implementation of MVC:** • Employee as the Model. • EmployeeManager as the Controller. • EmployeeManagementGUI as the View.
Encapsulation	Fields in Employee are private and accessed only through getters and setters, ensuring data protection and integrity.
Observer Concept (Manual Refresh)	After each add, update, or remove action, the GUI manually refreshes the employee list. This mimics a basic observer pattern manually.

Use Case Diagram: -



SSD: -


Hostel...
EmployeeManagementGUI
EmployeeManager
Employee
Click "Add" (with details)
addEmployee(Employee)
new Employee(name, email, phone, experience)
Employee object created
true/false (added or not)
Refresh JList
Select employee from JList
Populate form fields
Click "Update" (after editing)
updateEmployee(originalName, new Name, new Email, new Phone, new Experience)
setName(), setEmail(), setPhone(), setExperience()
true/false (updated or not)
Refresh JList
Select employee from JList
Click "Delete"
removeEmployee(Employee)
true/false (deleted or not)
Refresh JList
Open GUI
getAllEmployees()
List<Employee>
Display employee list
Text is not SVG - cannot display