UML DESIGN

```
+--------------------------------------+          +---------------------+
|                        Building                       |           |              Lift                |
+--------------------------------------+          +---------------------+
| - lifts: vector<Lift>                            |           | - currentLevel: int |
+--------------------------------------+           | - maxLevel: int          |
| + Building(numLifts: int, maxLevel: int) |     |                                  |
| + requestLift(liftIndex: int, level: int): |     | + Lift(maxLevel: int)|
|      void                                                   |     | | + moveToLevel(level: int): void |
| + operateLifts(): void                              |    | | + operate(): void |
+--------------------------------------+    +---------------------+
```

DESIGN SPECIFICATION

Title: Elevator System

Overview
The Elevator Simulation System is a program designed to simulate the operation of elevators within a building. Users can input the number of elevators and the maximum level of the building, request elevator movements, and observe the simulated operation of the elevators.

Features

Elevator Management:
Create a specified number of elevators within the building.
Allow elevators to move to requested levels.
Simulate the operation of elevators.

User Interaction:
Users can input the number of elevators and the maximum level of the building.
Users can request elevator movements by specifying the elevator index and destination level.
Users can choose to operate the elevators.

DESIGN REQUIREMENT

System Requirements

Hardware Requirements

Personal computer or device capable of running C++ programs.
Input device (keyboard) for user interaction.
Output device (monitor) for displaying program output.

Software Requirements
C++ compiler for support.

Functional Requirements:

Elevator Class
Initialization:

The system must allow the creation of elevator objects with a specified maximum level.
Elevators must be initialized at level 1.
Movement:

Elevators must be capable of moving to specified levels within the valid range (1 to maxLevel).
Invalid level requests must be handled, and an error message should be displayed.
Operation Simulation:

Elevators must have a method to simulate their operation, e.g., moving to requested levels.

Building Class
Initialization:

The system must allow the creation of a building object with a specified number of elevators and a maximum level.

Request Handling:

The system must allow users to request elevator movements by specifying the elevator index and destination level.
Invalid elevator indices must be handled, and an error message should be displayed.

Operation Simulation:

The system must have a method to simulate the operation of all elevators within the building.

User Interface (Main Function)
Input:

Users must be able to input the number of elevators and the maximum level of the building.
Users must be able to input elevator index and destination level for elevator movements.

Output:

The system must output messages indicating the movement and operation of elevators.
Error messages must be displayed for invalid inputs.

Control Flow:

The system must allow users to terminate the elevator movement input loop by entering -1.

Non-functional Requirements
1. Performance
The system should have low response times for elevator movements and operations.

2. Usability
The user interface should be intuitive and provide clear instructions to users.

3. Reliability
The system should handle invalid inputs gracefully and avoid crashes.

4. Portability
The program should be compatible with common C++ compilers on various platforms.

5. Maintainability
Code should be well-documented and adhere to coding standards for ease of maintenance.

Assumptions and Constraints

The program assumes a basic console-based user interface for simplicity.
The maximum level and number of elevators must be positive integers.
The system does not consider advanced elevator scheduling algorithms for simplicity.
Risks and Mitigations

Risk Involved: Invalid user input may cause unexpected behavior.

Associated Mitigation: Implement input validation and provide informative error messages.

Risk Posed: Performance issues with a large number of elevators or levels.

Associated Mitigation: Optimize the code and algorithms for better performance.