# Complete Report on Elevator System Implementation and Testing

## 1. Introduction

The Elevator System is a simulation that assigns elevators to floor requests based on defined rules. The system consists of multiple elevators, each responsible for servicing specific floors, and a lobby that manages requests and delegates them to the appropriate elevator. The assignment of elevators is based on the floor requested and the direction of movement.

## 2. Objective

The objective of this report is to demonstrate the process of implementing, testing, and validating the elevator system. This includes:

* The design and implementation of the elevator assignment logic.
* The creation and execution of unit tests to verify the correctness of the implementation.
* The application of graph-based coverage techniques to ensure thorough testing of the system.

## 3. Code Implementation

The system is divided into two main components:

* **Elevator Class**: Represents an elevator and contains its properties such as its current floor, the range of floors it can service, and the direction it is moving.
* **Lobby Class**: Manages a collection of elevators and handles floor requests from users, delegating them to the appropriate elevator based on floor ranges and direction.

## 4. Test Case Design and Coverage

### 4.1 Control Flow Graph (CFG) for Each Test Case

Below is a detailed **Control Flow Graph (CFG)** that represents the flow of execution within the elevator assignment system.

### Test Case 1: Basic Elevator Assignment (Requesting Floor 3)

#### CFG for Test Case 1:

[Start]

|

v

[Generate Request] --> [Floor Check] --> [Elevator Range Check] --> [Assign Elevator]

| | |

v v v

[Assign Elevator]--> [End] <--[Mark Elevator Assigned]

#### Coverage:

* **Node Coverage**: All nodes (Generate Request, Floor Check, Elevator Range Check, Assign Elevator, Mark Elevator Assigned, End) are covered.
* **Edge Coverage**: All edges are covered between key steps of request processing.
* **Edge Pair Coverage**: All possible consecutive transitions (Generate Request → Elevator Range Check → Assign Elevator).
* **Prime Path Coverage**: Full path coverage from request generation to elevator assignment and task completion.

### Test Case 2: No Suitable Elevator (Requesting Floor 5)

#### CFG for Test Case 2:

[Start]

|

v

[Generate Request] --> [Floor Check] --> [Elevator Range Check] --> [No Suitable Elevator]

| | |

v v v

[Assign Elevator]--> [End] <--[Error Message Printed]

#### Coverage:

* **Node Coverage**: All nodes are covered, including error handling when no suitable elevator is found.
* **Edge Coverage**: All edges are covered, including the error handling path.
* **Edge Pair Coverage**: Covers the transitions from request generation to elevator range check and then to error message handling.
* **Prime Path Coverage**: Full path coverage from request generation to no suitable elevator error.

### Test Case 3: Forced Elevator Assignment (Requesting Floor 5)

#### CFG for Test Case 3:

[Start]

|

v

[Generate Request] --> [Floor Check] --> [Elevator Range Check] --> [Force Assignment]

| | |

v v v

[Assign Elevator]--> [End] <--[Forced Assignment Action]

#### Coverage:

* **Node Coverage**: All nodes are covered, ensuring forced assignment is tested.
* **Edge Coverage**

: All edges are covered, including forced assignment logic.

* **Edge Pair Coverage**: Transitioning between the floor check, elevator range check, and forced assignment paths.
* **Prime Path Coverage**: Full path from request generation to forced assignment and completion.

### Test Case 4: Multiple Requests and Assignments (Requesting Floors 3, 5, 7)

#### CFG for Test Case 4:

[Start]

|

v

[Generate Request] --> [Floor Check] --> [Elevator Range Check] --> [Assign Elevator] --> [Mark Elevator Assigned]

| | | |

v v v v

[Assign Elevator]--> [End] <--[Mark Elevator Assigned]

#### Coverage:

* **Node Coverage**: All nodes, including handling multiple requests, are covered.
* **Edge Coverage**: All edges between steps in request handling are covered.
* **Edge Pair Coverage**: Multiple edge pair combinations are tested.
* **Prime Path Coverage**: Full path coverage for sequential request processing and elevator assignment.

## 5. Graph-Based Coverage

### Conclusion on Graph-Based Coverage:

1. **Node Coverage**: Achieved in all test cases. Every function and decision point in the elevator assignment process is tested.
2. **Edge Coverage**: Achieved across all test cases. All possible transitions between decision points and actions have been tested.
3. **Edge Pair Coverage**: Achieved. Consecutive pairs of transitions between key logic steps are tested.
4. **Prime Path Coverage**: Achieved. All meaningful execution paths from request generation to assignment completion are covered.

All types of graph-based coverage—Node, Edge, Edge Pair, and Prime Path—are fully tested and validated in each test case, ensuring comprehensive testing of the elevator assignment system.