Vishrutha Gopa
101267228

COMP 3004 Assignment 3

Elevator System Simulator in Qt C++

Use Cases

USE CASE 1: PASSENGER INTERACTION WITH ELEVATOR SYSTEM
Primary Actor: Passenger
Scope: Elevator System
Level: User Goal
Stakeholders and Interests:
- Passenger: Wants to reach a specific destination floor.

Preconditions: Passenger wants to use the elevator.
Success Guarantees Elevator arrives at the requested floor, and passengers can board/exit the elevator. Elevator moves to the selected destination floor accurately.
Main Success Scenario:

1. Passenger presses the "up" or "down" button on the floor.
2. Elevator system receives the request and illuminates the corresponding button.
3. Elevator arrives, rings a bell, and opens its doors.
4. Passengers exit or board within a fixed time of 5 seconds.
5. Passengers inside the elevator select their destination floor using the button panel.
6. Elevator system registers the selected destination.
7. Elevator rings a bell, closes its doors, and proceeds to move toward the selected destination.
8. The system displays updates to show the current floor number.
9. Elevator arrives at the selected floor, and notifies the control system.

Extensions:
4a. Passenger wants to control door timing:
- 4a1. Passengers can use the "open door" button to hold the doors open beyond the default time
- 4b2. Premature closure of doors can be achieved by pressing the "close door" button.

*a. The passenger faces an emergency:
- *a1.The passenger presses the "Help" button inside the elevator (Use Case 2)

*b. The Elevator's door is block:
- *b1. It detects an obstacle while the door is closing. (Use Case 3)

*c. The control system receives a "Fire" alarm signal:
- *c1. Control system receives a "Fire" alarm signal. (Use Case 4)

*d. The elevator sensors indicate an overload while passengers are inside the elevator:
- *d1. Elevator does not move. (Use Case 5)

*e. The control system receives a "Power Out" alarm signal:
- *e1. Control system receives a "Power Out" alarm signal. (Use Case 6)

USE CASE 2: PASSENGER REQUEST EMERGENCY ASSISTANCE
Primary Actor: Passenger
Scope: Elevator System
Level: User Goal
Stakeholders and Interests:
- Passenger: Seeks assistance in an emergency.
- Building Safety Service: Responsible for responding to help requests.

- Emergency Services: Involved in cases where a 911 call needs to be placed.

Preconditions: Passenger is inside the elevator facing an emergency and presses the "Help" button.
Success Guarantees: Passenger is connected to building safety services or emergency services are contacted.
Main Success Scenario:

1. Elevator control system receives a "Help" alarm signal.
2. Passenger is connected to building safety services through a voice connection.
3. If no response within 5 seconds, or from the passenger, a 911 emergency call is placed.

Extensions:
*a. No response from building safety within 5 seconds or from the passenger:

- *a1. Initiates a 911 emergency call.
- *a2. The emergency services are informed of the situation and can take appropriate action.

USE CASE 3: DETECTION OF OBSTACLES IN ELEVATOR DOORS
Primary Actor: User/Simulation Control Interface
Scope: Elevator System
Level: System Function
Stakeholders and Interests:

- Passengers: Want safe door operations.

Preconditions: Door is closing, and the user activates the "Obstacle Detected" simulation button
Success Guarantees: Door stops closing, opens, and passengers are warned of the obstacle.
Main Success Scenario:

1. The control system receives a simulated signal.
2. The system stops the door from closing and opens it.
3. Passengers are warned of the obstacle.

USE CASE 4: INITIATION OF FIRE ALARM RESPONSE
Primary Actor: User/Simulation Control Interface
Scope: Elevator System
Level: System Function
Stakeholders and Interests:

- Passengers: Need to be informed and evacuated during a fire.

Preconditions: User triggers the "Fire Alarm" simulation button in the control interface.
Success Guarantees: Elevators move to safe floors, passengers are informed, and emergency procedures are followed.
Main Success Scenario:

1. Control system receives a signal.
2. All elevators are moved to a safe floor.
3. Audio and text messages inform passengers of the emergency and instruct them to disembark once the safe floor is reached.

USE CASE 5: ACTIVATION OF OVERLOAD ALARM
Primary Actor: User/Simulation Control Interface
Scope: Elevator System
Level: System Function

Vishrutha Gopa
101267228

Stakeholders and Interests:
- Passengers: Want safe and efficient elevator operations.

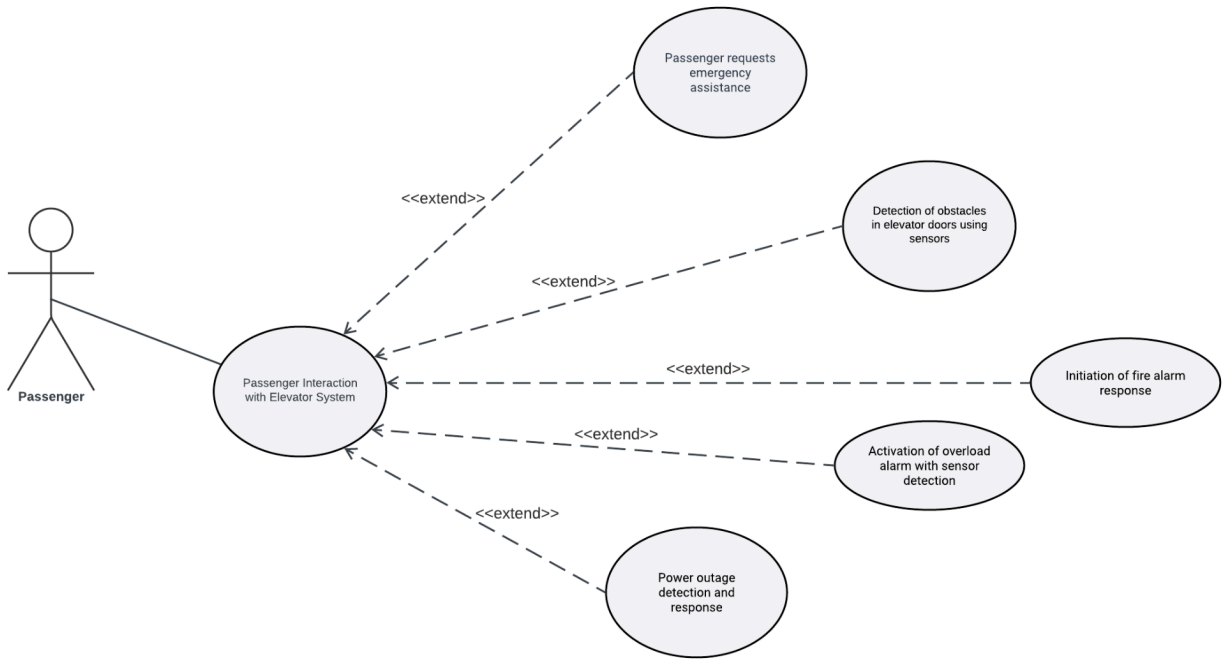
Preconditions: User activates the "Overload" simulation button to indicate that the passenger or cargo load exceeds the carrying capacity.
Success Guarantees: Elevator does not move until the load is reduced, and passengers are informed.
Main Success Scenario:

1. Sensors are simulated to indicate an overload condition.
2. Passengers receive audio and text messages instructing them to reduce the load as part of the simulation.
3. The elevator does not move for 10 seconds, simulating the response to an actual overload and waiting for passengers to disembark.

USE CASE 6: POWER OUTAGE DETECTION AND RESPONSE
Primary Actor: Power Outage System
Scope: Elevator System
Level: System Function
Stakeholders and Interests:
- Passengers: Need information and guidance during a power outage.

Preconditions: Power outage detected in the building.
Success Guarantees: Passengers are informed, moved to a safe floor, and asked to disembark during a power outage.
Main Success Scenario:

1. Control system receives a signal.
2. Audio and text messages inform passengers of the power outage.
3. Each elevator is moved to a safe floor.
4. Passengers are asked to disembark via audio and text messages.



Vishrutha Gopa
101267228

Design Documentation

UML Class Diagram

Note: Please refer to the folder *UML Class Diagrams* in the submission for the diagrams. Signals were not included in the Class Diagram as the professor mentioned it was not required.

Sequence Diagrams for 2 Success Scenarios with Explanation

Note: Please refer to the folder *Sequence Diagrams* in the submission for the diagrams. Only the explanation is included in this document.

In this simulation, all elevators start at Floor 1. Elevators are assigned based on their proximity to the requesting passenger and their current direction of movement.

Scenario 1: Three Different Elevators for Three Passengers on Different Floors

Passenger 1 Requests an Elevator:

- Passenger 1 requests an elevator from floor 2 and wants to travel up.
- The Elevator Control System (ECS) allocates Elevator 1 for this request.
- After entering Elevator 1, Passenger 1 selects floor 7 as the destination.
- ECS coordinates Elevator 1's upward movement until it reaches floor 7.

Passenger 2 Requests an Elevator:

- Passenger 2 requests an elevator from floor 3 and wants to travel up.
- The Elevator Control System (ECS) allocates Elevator 2 for this request.
- After entering Elevator 2, Passenger 2 selects floor 6 as the destination.
- ECS coordinates Elevator 2's upward movement until it reaches floor 6.

Passenger 3 Requests an Elevator:

- Passenger 3 requests an elevator from floor 2 and wants to travel down.
- The Elevator Control System (ECS) allocates Elevator 3 for this request.
- After entering Elevator 3, Passenger 3 selects floor 1 as the destination.
- ECS coordinates Elevator 3's downward movement until it reaches floor 1.

Scenario 2: Passengers share an Elevator.

Passengers 1 and 2 Share an Elevator:

- Passenger 1 is located on floor 1 and wants to travel upwards.
- The Elevator Control System (ECS) efficiently allocates Elevator 1 for Passenger 1’s request.
- After boarding Elevator 1, Passenger 1 selects floor 7 as the destination.
- Right after, Passenger 2 who is located on floor 4 and wants to travel upwards.
- The Elevator Control System (ECS) efficiently allocates Elevator 1 for this passenger's request

also, given their starting point and direction.
- Elevator 1 moves and stops to pick up Passenger 2 on floor 4.
- After boarding Elevator 1, Passenger 2 selects floor 6.
- ECS coordinates Elevator 1's journey, first stopping at floor 6 to accommodate Passenger 2's exit before continuing upwards to floor 7 for Passenger 1.

Passenger 3 Requests an Elevator:
- Passenger 3 is on floor 3 and wishes to go down to floor 2.
- ECS allocates Elevator 2 for Passenger 3's downward journey.
- After entering Elevator 2, Passenger 3 selects floor 2 as the destination.
- ECS directs Elevator 2 to descend to floor 2.

Additional Passenger 4 Requesting an Elevator (this ensures that all 3 elevators are used):
- Passenger 4 is on floor 1 and wants to travel upward to floor 4.
- ECS allocates Elevator 3 for Passenger 4's journey.
- Upon entering Elevator 3, Passenger 4 selects floor 4 as the destination.
- ECS guides Elevator 3 to ascend, stopping at floor 4 for Passenger 4.

UML Sequence Diagrams for 5 Safety Scenarios with Explanation

Note: Please refer to the folder *Sequence Diagrams* in the submission for the diagrams. Only the explanation is included in this document.

Safety Scenario 1: Help Alarm

When a passenger triggers the "Help" button in the elevator, the system activates the safety_state and coordinates the elevator to the closest safe floor. If the system successfully connects to the building safety service, a voice connection is established. However, if there is no response from the building safety service within 5 seconds, the system initiates a 911 emergency call. Using a random number generator, there's a 60% chance of connecting to the building safety service and a 40% chance of placing a 911 emergency call.

Safety Scenario 2: Door Obstacles

When the "Block Door" button is pressed, the ElevatorControlSystem coordinates with the Elevator to handle the situation. If an obstruction is detected while the doors are trying to close, the ElevatorControlSystem instructs the Elevator to open the doors. Additionally, warning messages about the obstruction are displayed and logged in the Console Log. Finally, the safety state is reset once the situation is resolved.

Safety Scenario 3: Fire Alarm

When the "Fire Alarm" button is pressed, the ElevatorControlSystem coordinates with the Elevator to handle the situation. Upon detection of the fire alarm, passengers are told to disembark once the

elevator reaches a safe floor. The ElevatorControlSystem coordinates with the Elevator to move to the nearest safe floor. Subsequently, the doors are opened to let passengers exit. Warning messages and instructions are displayed and logged in the Console Log.

Safety Scenario 4: Overload

When the overload button is pressed, the system responds by setting the safety state to true. The Console Log logs an emergency message about the overload detection and announces a warning to reduce the load. Additionally, it displays a warning message indicating the overload. A single-shot timer is initiated for a 10-second delay to handle the overload situation. After the delay, normal operation is resumed, and the Console Log logs a message indicating the normal operation has resumed. Finally, the safety state is set to false.

Safety Scenario 5: Power Outage

In the "Safety Scenario 5: Power Outage" sequence diagram, when the power outage button is pressed, the system responds by setting the safety state to true. The Console Log logs a warning message requesting passengers to disembark once the safe floor is reached. Then, the Elevator Control System coordinates the elevator to the closest safe floor. The elevator moves to the safe floor and opens its doors. The Console Log announces the request for passengers to disembark and displays a message instructing passengers to disembark once the elevator stops.

State Machine Diagrams

State Diagram for Elevator

Note: Please refer to the folder *State Diagrams* for the diagram.

Explanation:

1. Idle:
    a. Waiting for new requests.
    b. When a new request is detected and the request queue is not empty, the elevator transitions to the Moving state to fulfill the request.
2. Moving:
    a. In the Moving state, the elevator is moving toward the requested floor.
    b. It remains in this state until it reaches the requested floor or receives a safety signal.
3. Stopping:
    a. Upon reaching the requested floor or receiving a safety signal, the elevator transitions to the Stopping state.
    b. Here, it stops and prepares to open its doors.
4. Door Opening:
    a. After stopping, the elevator opens its doors to allow passengers to enter or exit.
    b. While the doors are open, the elevator checks for any safety signals.
    c. If a safety signal is detected, the doors remain open; otherwise, they transition to the

Door Closing state.

5. Door Closing:
    a. After ensuring no safety signals are present and the doors have been open for an appropriate duration, the elevator transitions to the Door Closing state.
    b. Here, the doors begin to close to prepare for the next request.
6. Check for Additional Requests:
    a. After the doors close, the elevator checks if there are any additional requests in the queue.
    b. If requests are found, the elevator transitions back to the Moving state to fulfill them.
    c. Otherwise, it remains in the Idle state until a new request is received.

State Diagram for Elevator Control System

Note: Please refer to the folder *State Diagrams* for the diagram.

Explanation:

1. Idle (Monitoring for Requests):
    a. The ECS is waiting for requests from users.
2. Allocate Requests:
    a. Transitioned to when the ECS receives a new request from a user or elevator.
    b. In this state, the ECS analyzes incoming requests and allocates the most suitable elevator to fulfill them.
3. Coordinate Elevators:
    a. Transitioned after allocating an elevator to a request.
    b. The ECS manages the elevator's queue of pending requests and coordinates its movement.
4. Handling Emergencies:
    a. Handle Help Signal
        i. After handling the help signal, the ECS transitions back to the previous state
    b. Handle Overload Signal
        i. After handling the overload situation, the ECS returns to the previous state.
    c. Handle Block Door Signal
        i. After handling the block door signal, the ECS transitions back to the previous state.
    d. Handle Fire Alarm Signal
        i. After handling the fire alarm, the ECS transitions to a final state.
    e. Handle Power Outage Signal
        i. After handling the power outage, the ECS transitions to a final state.

Traceability Matrix

ID	Requirement	Related	Fulfilled by	Implemented by	Tested by	Description

Vishrutha Gopa
101267228

		Use Case				
1	Passengers can press "up" or "down" buttons on floors to request elevator service.	Use Case 1	MainWindow, ElevatorControlSystem	MainWindow: upButtonPressed(), downButtonPressed(), handleFloorPanelRequest(direction: int), startingFloorSpinBox();	To test, select your starting floor with the spin box, then press either the "up" or "down" button.	The GUI gets floor requests and communicates with the ElevatorControlSystem to handle the request.
2	Respond to floor requests efficiently.	Use Case 1	MainWindow, ElevatorControlSystem, Elevator	ElevatorControlSystem: allocateElevator(), addRequest()	Test by creating multiple floor requests from the GUI and observe the allocation efficiency through the Console Log.	The system uses an allocation algorithm to choose the most efficient elevator to respond to a request.
3	Elevators arrive at the requested floor.	Use Case 1	MainWindow, ElevatorControlSystem, Elevator	ElevatorControlSystem: moveElevator(elevatorId: int) Elevator: moveToFloor(floor: Int)	Monitor the Console Log for messages that confirm elevators are moving to and reaching the correct floors.	ElevatorControlSystem directs Elevators to move to specific floors.
4	Passengers inside elevators can select destination floors using button panels.	Use Case 1	MainWindow, ElevatorControlSystem	MainWindow: floorButtonPressed() ElevatorControlSystem: addRequest()	Use the GUI to select a floor from within an elevator and check the Console Log for the request.	Floor selections by passengers are processed by the ElevatorControlSystem. The ElevatorControlSystem adds the request to the Elevator's queue
5	Elevators have displays showing the	Use Case 1	MainWindow, ElevatorControlSystem	MainWindow: updateSystemDisplay(), logMessage(message:	Observe the GUI and Console Log	The Console Log and System display are

	current floor number and warning messages.			string) ElevatorControlSystem: statusUpdate(message: string)	for real-time updates on the current floor and any messages.	updated in real-time to show the current floor and relay important messages.
6	Elevators display and announce warning messages.	Use Case 1	MainWindow, ElevatorControlSystem	MainWindow: updateSystemDisplay(), logMessage(message: string) ElevatorControlSystem: statusUpdate(message: string)	Check the Console Log for any warning or emergency messages during operation.	Console Log displays and announces warnings and emergency messages.
7	Elevators notify them when they arrive on the floor.	Use Case 1	MainWindow, ElevatorControlSystem, Elevator	MainWindow: updateSystemDisplay(), logMessage(message: string) ElevatorControlSystem: statusUpdate(message: string), openElevatorDoors(elevatorId: int), closeElevatorDoors(elevatorId: int) Elevator: openDoors(), closeDoors()	Watch the elevator's floor movement, door opening, and bell sounds through the Console Log when an elevator reaches a floor.	Messages when elevators reach a floor are shown in the Console Log.
8	Passengers can control elevator door timing using the "open door" and "close door" buttons.	Use Case 1	MainWindow, ElevatorControlSystem, Elevator	MainWindow: openButtonPressed(), closeButtonPressed() ElevatorControlSystem: openElevatorDoors(elevatorId: int), closeElevatorDoors(elevatorId: int) Elevator: openDoors(), closeDoors()	Interact with the "open door" and "close door" buttons during an elevator stop and verify door action in the Console Log.	GUI allows door control, with actions processed by the ElevatorControlSystem.
9	Elevator	Use	MainWindow,	MainWindow:	Press the	There's a 60%

	system responds to "Help" button presses, connecting passengers to building safety service.	Case 2	ElevatorControlSystem	handleSafetyButtonPress(buttonName: QString) ElevatorControlSystem: safety_state, pressedSignal, openElevatorDoors(elevatorId: int), closeElevatorDoors(elevatorId: int) Elevator: moveToFloor(floor: Int), openDoors()	"Help" button in the GUI and observe the response action logged in the Console.	chance of the elevator connecting to the building safety service through a voice connection, while there's a 30% chance of placing a 911 emergency call.
10	Elevator system stops and opens doors if an obstacle blocks the door.	Use Case 3	MainWindow, ElevatorControlSystem, Elevator	MainWindow: handleSafetyButtonPress(buttonName: QString) ElevatorControlSystem: safety_state, pressedSignal, openElevatorDoors(elevatorId: int), closeElevatorDoors(elevatorId: int) Elevator: moveToFloor(floor: Int), openDoors()	Simulating an obstacle in the door path by pressing the 'Block Door' button in the Admin Panel of the GUI.	ElevatorControlSystem closes and opens the Elevator doors.
11	Elevator system alerts passengers in case of fire alarms and moves elevators to safe floors.	Use Case 4	MainWindow, ElevatorControlSystem, Elevator	MainWindow: handleSafetyButtonPress(buttonName: QString) ElevatorControlSystem: safety_state, pressedSignal, openElevatorDoors(elevatorId: int), closeElevatorDoors(elevatorId: int)	Trigger a fire alarm from the Admin Panel and observe elevators moving to designated safe floors.	Elevators react to fire alarms by moving to safe floors.

				Elevator: moveToFloor(floor: Int), openDoors()		
12	Elevator system handles overload alarms.	Use Case 5	MainWindow, ElevatorControlSystem, Elevator	MainWindow: handleSafetyButtonPress(buttonName: QString) ElevatorControlSystem: safety_state, pressedSignal, openElevatorDoors(elevatorId: int), closeElevatorDoors(elevatorId: int) Elevator: moveToFloor(floor: Int), openDoors()	Simulate an overload in the elevator by pressing the 'Overload' button in the Admin Panel of the GUI and verify the system's response in the Console Log.	A QTimer object is utilized to introduce a delay in the system response. After the overload alarm is triggered, the program pauses for a 10-second delay before resuming normal operation.
13	Elevator system responds to power outage alarms, moving elevators to safe floors.	Use Case 6	MainWindow, ElevatorControlSystem	MainWindow: handleSafetyButtonPress(buttonName: QString) ElevatorControlSystem: safety_state, pressedSignal, openElevatorDoors(elevatorId: int), closeElevatorDoors(elevatorId: int) Elevator: moveToFloor(floor: Int), openDoors()	Trigger a power outage from the Admin Panel and observe elevators moving to designated safe floors.	Elevators react to power outages by moving to safe floors.
14	Variability in floors and elevators	NA	ElevatorControlSystem	ElevatorControlSystem: num_floors, num_elevators	Adjust the number of floors and elevators in the ElevatorControlSystem class	The backend is dynamic and can handle a variable number of floors and elevators, but the GUI currently supports a fixed number (i.e. 7 floors and 3

						elevators)

Discussion of Design Decisions

I chose to implement a centralized design for the elevator system using the mediator design pattern. The Elevator Control System (ECS) serves as the mediator, managing elevator operations and coordinating communication between elevators.

When a passenger presses a button on the elevator's button panel or on the floor's button panel, the request is sent to the ECS. The ECS determines which elevator is best suited to respond to the request. It considers factors such as the elevator's current location, and direction of travel. Once the ECS allocates the elevator, it assigns the floor request to that elevator, by adding it to the elevator's request queue.

With the floor request allocated, the ECS then directs the elevator's movement. It instructs the elevator on which floors to stop at and when to open or close its doors.

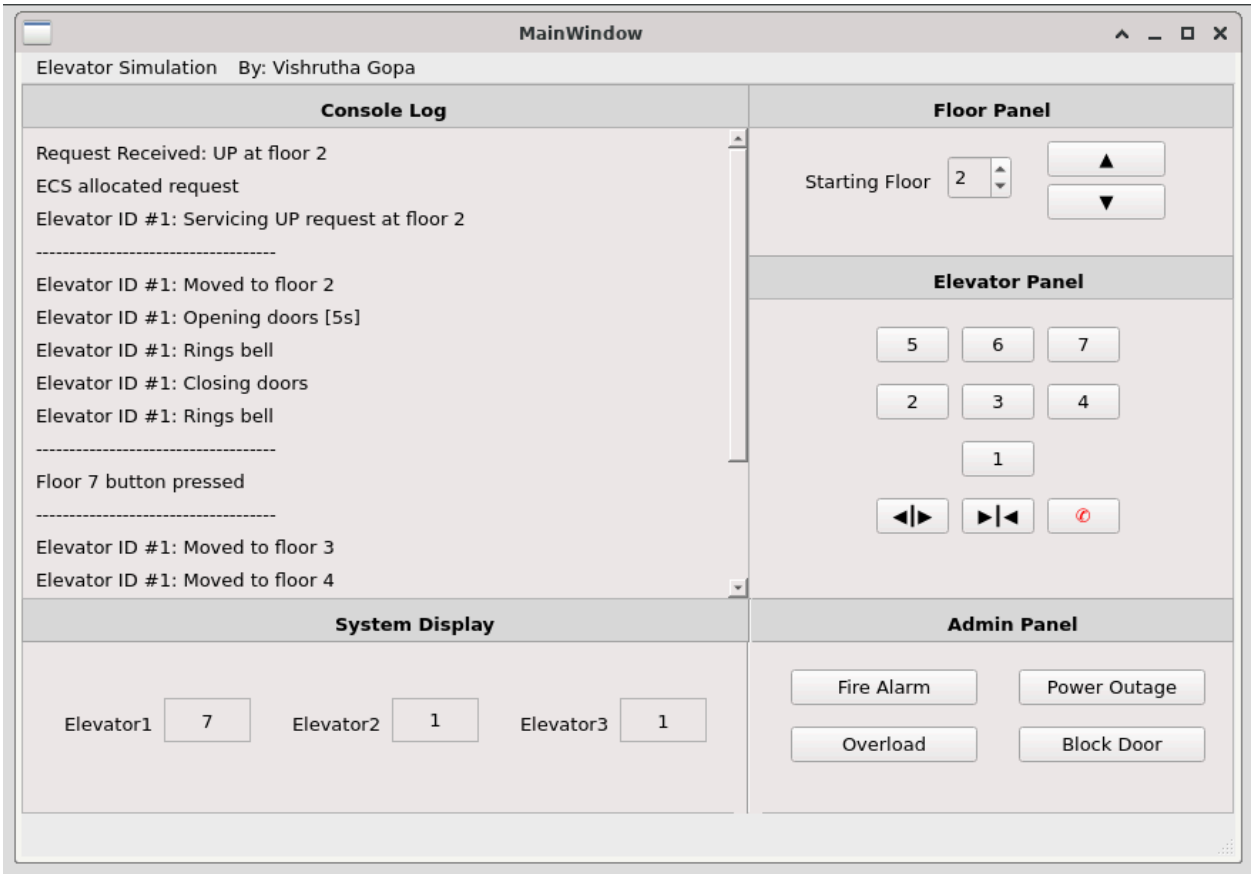
Furthermore, the ECS is responsible for handling safety-related events and alarms. Whenever a passenger or user triggers a safety-related signal by pressing a button on the Admin or Elevator Panel, such as for fire alarms, overload signals, or power outages, the ECS receives these signals from the MainWindow. Once received, the ECS takes appropriate action, such as moving elevators to safe floors or initiating emergency protocols.

Additional Notes to TAs

1. For the Fire Alarm scenario, I implemented a probabilistic approach to handle the connection with the Building Safety Service. When the fire alarm is triggered, there's a 60% chance of establishing a connection with the Building Safety Service through a voice connection. However, if there's no response within 5 seconds, there's a 40% probability of resorting to a 911 emergency call. This approach ensures a balance between attempting to connect with the Building Safety Service and resorting to an emergency call.
2. Please note that the number of floors and elevators in the ElevatorControlSystem header file can be adjusted as needed. While the backend is designed to handle a dynamic number of floors and elevators, the current GUI implementation supports a fixed number, specifically 7 floors and 3 elevators. To modify the number of floors or elevators, update the GUI components accordingly. This includes adjusting push buttons, spin boxes, and any other UI elements to accommodate the changes.
3. In Scenario 2, adding Passenger 4 was necessary to make sure Elevator 3 was used. Initially, Passengers 1 and 2 on floor 1 were going up, and Elevator 1 served them. When Passenger 3 on floor 3 requested to go down, Elevator 2 was allocated for the downward journey. To ensure all elevators were in use, Passenger 4 was introduced on floor 1, going up to floor 4, and Elevator 3 was allocated for this additional request. If a passenger (1,2,3) is reused on a specific floor, the

Elevator Control System (ECS) would prioritize allocating the elevator that is closest to that floor, rather than specifically allocating Elevator 3.

GUI (Qt Creator)



Youtube Video Link

- COMP 3004: Assignment 3 Video Demonstration | 2 Success Scenarios: https://youtu.be/k4EXgFPaubc
- COMP 3004: Assignment 3 Video Demonstration | 5 Safety Scenarios: https://youtu.be/9NwhojmTGM0