# Elevator Simulation System Report

Nathan El-Khoury

Demo link: https://youtu.be/FrOwDL6uNB4

# Use Cases and Diagram

## Use Cases

**USE CASE 1:** Regular Elevator Usage

**Primary Actor**:
1. Elevator Passenger

**Stakeholders and Interests:**
1. Elevator Passengers: Interested in arriving at their destination safely.
2. Building Owner: Interested in elevator functionality and user safety.
3. Building Management: Interested in elevator functionality and user safety.

**Pre-Conditions:** Elevator and related components are already installed and certified.

**Success Guarantee:** Elevator arrives at floor requested by user.

**Main Success Scenario:**
1. Passenger presses either “up” or “down” button at current floor
2. Button illuminates and remains illuminated until the elevator arrives.
3. Elevator arrives, rings a bell, and opens its doors for a fixed time (10 seconds)
4. Passenger enters the elevator and selects the destination floor.
5. Elevator closes its doors and moves up or down to the designated floor.
6. Elevator arrives at the selected floor and the passenger exits.

**Extensions:**

4a. Passenger does not enter the elevator.
   a. The elevator bell rings again, and the elevator closes after the fixed opening time of 10 seconds.

4b. Passenger or related objects overload the elevator
   a. Elevator overload system detection the overload and responds (Use Case 6)

4c. Passenger pressed “Help” button on elevator panel
   a. Elevator attempts to connect to the passenger and responds accordingly (Use Case 3)

5a. Passenger pressed manual door operation “Open”
   a. Elevator door remains open overriding default timing (Use Case 2)

5b. Passenger pressed manual door operation “Close”
   a. Elevator door closes overriding default timing (Use Case 2)

5c. Elevator Control System detects a obstruction
   a. Elevator detects a obstruction and responds ( Use Case 4)

5d. A fire alarm is signaled and alerts the elevator
   a. Elevator responds accordingly and transports to safe floor if possible ( Use Case 5)

5e. A power outage is signaled and alerts the elevator
   a. Elevator switches to backup power and moves to a safe floor (Use Case 7)

**USE CASE 2:** Passenger Uses Manual Door Operation

**Primary Actor:**

1. Elevator Passenger

**Stakeholders and Interests:**

1. Elevator Passenger: Interested in control over the elevator doors to fit their needs.
2. Building Owner: Interested in elevator functionality and user safety.
3. Building Management: Interested in elevator functionality and user safety.

**Pre-Conditions**: Elevator is operational and at the passenger's floor.

**Success Guarantee**: The elevator doors operate according to elevator passenger's input.

**Main Success Scenario**:

1. Passenger presses "up" or "down" and the elevator arrives at the user's floor.
2. Elevator passenger enters the elevator.
3. Elevator passenger presses the "open door" or "close door" buttons.
4. The elevator responds to passenger input overriding default timing, either opening beyond its default period or closing prematurely.

**Extensions**:

4a. Passenger attempts to close the door but it's obstructed.

   a. Elevator door sensors identify obstruction and reopen automatically.

**USE CASE 3:** Passenger Uses Elevator Help Button

**Primary Actor**: Elevator Passenger

**Stakeholders and Interests**:

1. Elevator Passenger: Wants immediate assistance in case of emergency.
2. Building Safety Services: Needs to respond as quickly as possible to emergencies in the elevator.
3. Building Owner: Interested in elevator functionality and user safety.

**Pre-Conditions**: Elevator is operational and has a working help button.

**Main Success** Scenario:

1. Passenger enters the elevator.
2. An emergency occurs which causes passenger to press help button.
3. The passenger presses the help button.
4. The control system connects the passenger to building safety services through a voice connection.
5. Building safety responds to passenger's help call and aids or contacts emergency services if needed.

**Extensions**:
4a. No response from building safety or from user within 5 seconds
  a. Control System calls 911 to report the emergency.

**USE CASE 4:** Elevator Door Obstacle Detection

**Primary Actor:** Elevator Control System

**Stakeholders and Interests:**
1. Elevator Passenger: Interested in safety during elevator operation.
2. Building Owner: Interested in elevator functionality and user safety.
3. Building Management: Needs safe and functional operation of elevators.

**Pre-Conditions**: Elevator is operational with correctly installed door sensors.

**Success Guarantee**: Elevator detects obstacle correctly and reopens.

**Main Success Scenario**:
1. Elevator doors begin to close.
2. Elevator sensors detect an obstacle which obstructs the door from closing.
3. Elevator door stops closing and reopens.
4. Elevator door attempts to close again.

**Extensions**:

3a. Obstacle is never removed from the doorway.
  a. Elevator door will continue to reopen.
  b. If there are repeated obstructions over a short period of time, the elevator audibly and visually displays a warning.

**USE CASE 5:** Elevator Fire Alarm Signaled

**Primary Actor:** Elevator Control System

**Stakeholders and Interests**:
1. Elevator Passenger: Interested in safety during fire emergency.
2. Building Management: Interested in safety of building occupants and correct operation of elevators.
3. Building Owner: Interested in elevator functionality and user safety.

**Pre-Conditions:** Working fire detection systems within building and elevators are operational.

**Success Guarantee:**

1. Elevator regular operations are correctly override and passengers are safely evacuated to nearest safe floor.

**Main Success Scenario:**

1. The control system receives a “Fire” alarm from the building or elevator itself.
2. All elevators are commanded to move to a safe floor.
3. An audio and text message are presented to passengers informing them of the emergency and to disembark once a safe floor is reached.
4. Elevator arrives at the safe floor and elevator passengers disembark from the elevator onto the safe floor.

**Extensions:**

4a. Elevator unable to reach the safe floor.

a. Elevator remains stationary.

**USE CASE 6:** Elevator Overload Detection

**Primary Actor**: Elevator Control System

**Stakeholders and Interests:**

1. Elevator Passengers: Interested in avoiding unsafe elevator conditions.
2. Building Management: Interested in preventing elevator malfunctions due to an overloaded elevator.
3. Building Owner: Interested in elevator functionality and user safety.

**Pre-Conditions:** Elevator is operational and has a working overload detection system.

**Success Guarantee:** Overload is detected in the elevator; a warning is issued, and the elevator remains stationary until load is reduced.

**Main Success Guarantee:**

1. Situation arises where elevator is loaded past its specifications.
2. Sensors indicate that the elevator is overloaded.
3. The control systems receive an “Overload” alarm signal.
4. The elevator ceases operation and an audio, and a text message are presented to passengers asking to reduce elevator load.
5. Elevator remains stationary until the load is reduced.
6. Elevator load is reduced, and elevator resumes operation.

**Extensions:**

4a. Passengers do not reduce the load.

a. Control system continues to keep elevator stationary and repeats the warning.

**USE CASE 7:** Elevator Power Outage Detected

**Primary Actor:** Elevator Control System

**Stakeholders and Interests:**

1. Elevator Passengers: Interested in safety during a power outage.
2. Building Management: Interested in passenger safety and correct response during a power outage.
3. Building Owner: Interested in elevator functionality and user safety.

**Pre-Conditions**: Elevator is installed and has functional power outage detection system and backup power system.

**Success Guarantee**: Elevator passengers are transported to a safe floor and elevator ceases operation till power returns.

**Main Success Guarantee:**

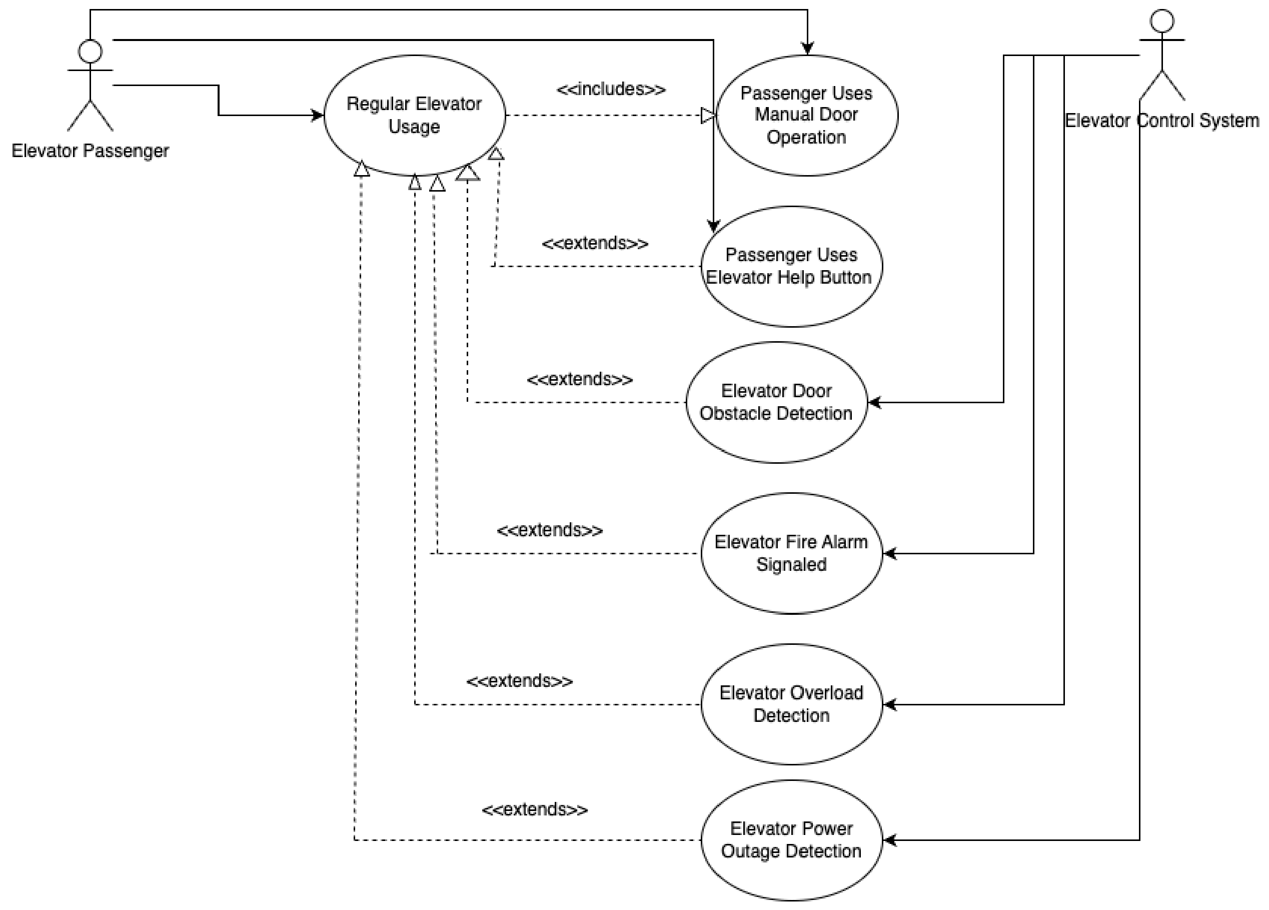
1. The control systems detect a power outage and receive a “Power Out” alarm signal.
2. Elevator switches to battery backup power
3. Passengers are informed of the power outage through audio and text messages.
4. The elevator moves to a safe floor.
5. Passengers are asked to disembark the elevator through audio and text messages.
6. Passengers disembark the elevator onto the safe floor.

**Extensions:**

2a. Elevator backup power fails.

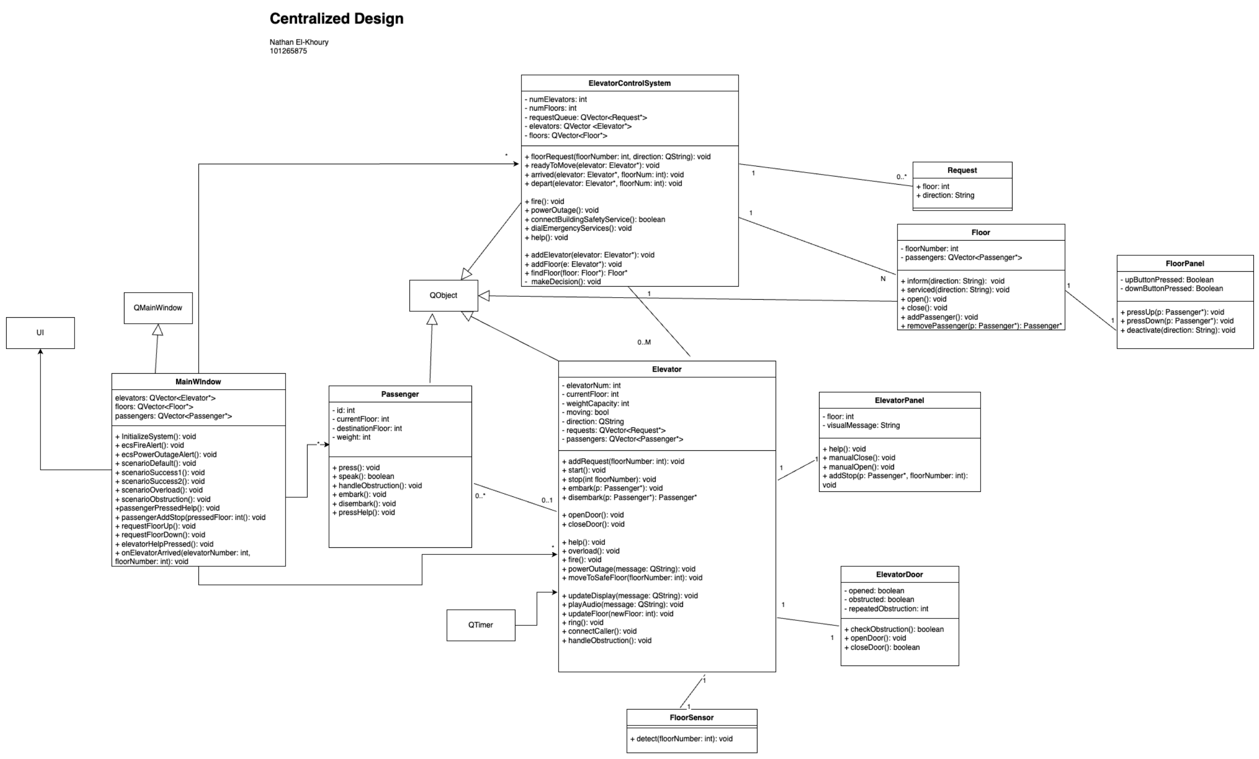
   a. Elevator remains stationary.

## Use Case Diagram:


Elevator Passenger
Regular Elevator Usage
<<includes>>
Passenger Uses Manual Door Operation
Elevator Control System
<<extends>>
Passenger Uses Elevator Help Button
<<extends>>
Elevator Door Obstacle Detection
<<extends>>
Elevator Fire Alarm Signaled
<<extends>>
Elevator Overload Detection
<<extends>>
Elevator Power Outage Detection

# UML Class Diagrams

## Centralized Design



Centralized Design
Nathan El-Khoury
101265875
ElevatorControlSystem
- numElevators: int
- numFloors: int
- requestQueue: QVector<Request*>
- elevators: QVector <Elevator*>
- floors: QVector<Floor*>
+ floorRequest(floorNumber: int, direction: QString): void
+ readyToMove(elevator: Elevator*): void
+ arrived(elevator: Elevator*, floorNum: int): void
+ depart(elevator: Elevator*, floorNum: int): void
+ fire(): void
+ powerOutage(): void
+ connectBuildingSafetyService(): boolean
+ dialEmergencyServices(): void
+ help(): void
+ addElevator(elevator: Elevator*): void
+ addFloor(e: Elevator*): void
+ findFloor(floor: Floor*): Floor*
- makeDecision(): void
Request
+ floor: int
+ direction: String
Floor
- floorNumber: int
- passengers: QVector<Passenger*>
+ inform(direction: String): void
+ serviced(direction: String): void
+ open(): void
+ close(): void
+ addPassenger(): void
+ removePassenger(p: Passenger*): Passenger*
FloorPanel
- upButtonPressed: Boolean
- downButtonPressed: Boolean
+ pressUp(p: Passenger*): void
+ pressDown(p: Passenger*): void
+ deactivate(direction: String): void
UI
QMainWindow
QObject
MainWIndow
elevators: QVector<Elevator*>
floors: QVector<Floor*>
passengers: QVector<Passenger*>
+ InitializeSystem(): void
+ ecsFireAlert(): void
+ ecsPowerOutageAlert(): void
+ scenarioDefault(): void
+ scenarioSuccess1(): void
+ scenarioSuccess2(): void
+ scenarioOverload(): void
+ scenarioObstruction(): void
+passengerPressedHelp(): void
+ passengerAddStop(pressedFloor: int(): void
+ requestFloorUp(): void
+ requestFloorDown(): void
+ elevatorHelpPressed(): void
+ onElevatorArrived(elevatorNumber: int, floorNumber: int): void
Passenger
- id: int
- currentFloor: int
- destinationFloor: int
- weight: int
+ press(): void
+ speak(): boolean
+ handleObstruction(): void
+ embark(): void
+ disembark(): void
+ pressHelp(): void
Elevator
- elevatorNum: int
- currentFloor: int
- weightCapacity: int
- moving: bool
- direction: QString
- requests: QVector<Request*>
- passengers: QVector<Passenger*>
+ addRequest(floorNumber: int): void
+ start(): void
+ stop(int floorNumber): void
+ embark(p: Passenger*): void
+ disembark(p: Passenger*): Passenger*
+ openDoor(): void
+ closeDoor(): void
+ help(): void
+ overload(): void
+ fire(): void
+ powerOutage(message: QString): void
+ moveToSafeFloor(floorNumber: int): void
+ updateDisplay(message: QString): void
+ playAudio(message: QString): void
+ updateFloor(newFloor: int): void
+ ring(): void
+ connectCaller(): void
+ handleObstruction(): void
ElevatorPanel
- floor: int
- visualMessage: String
+ help(): void
+ manualClose(): void
+ manualOpen(): void
+ addStop(p: Passenger*, floorNumber: int): void
ElevatorDoor
- opened: boolean
- obstructed: boolean
- repeatedObstruction: int
+ checkObstruction(): boolean
+ openDoor(): void
+ closeDoor(): boolean
QTimer
FloorSensor
+ detect(floorNumber: int): void
*
1
0..*
1
N
1
1
1
0..M
*
0..*
0..1
*
1
1
1
1
1
1

### Core Decisions

Core design pattern within the centralized design is a Mediator pattern. The ECS acts as the main control unit and communicates with all the elevators as a mediator to make decisions regarding request allocations. The ECS stores lists of Request objects (where the Request object effectively encapsulates request states within a singular class) and elevators.

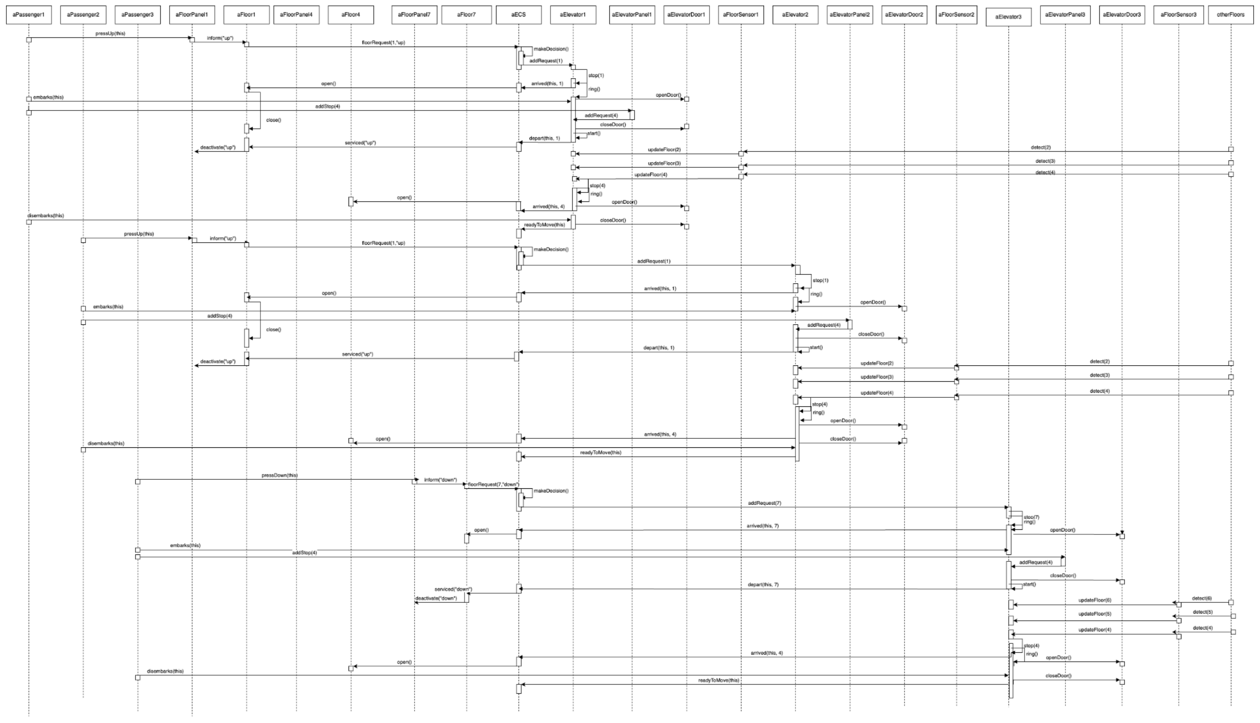
Utilizing a mediator pattern provides a central authority for the system. The ECS manages all communication between elevators and handles request allocation. Centralized decision-making through this mediator guarantees a cohesive point of command as a result enhancing system coherence. This structure links relevant components such as floors and elevators through the elevator control system.

### Class Choices

Functions such as buttons have been made methods within classes such as ElevatorPanel and FloorPanel. By including the buttons within relevant classes it encapsulates the button functionality within panel classes. This helps bundle data with the methods that operate on said data. Additionally, from a simplicity perspective this decision simplifies the design by reducing the number of classes making the system easier to maintain and extend. Similarly, displays and speakers are methodically integrated within the Elevator class, adhering to the principle of functional consolidation for simplicity and efficiency.

## Sequence Diagrams

### Success Scenario 1



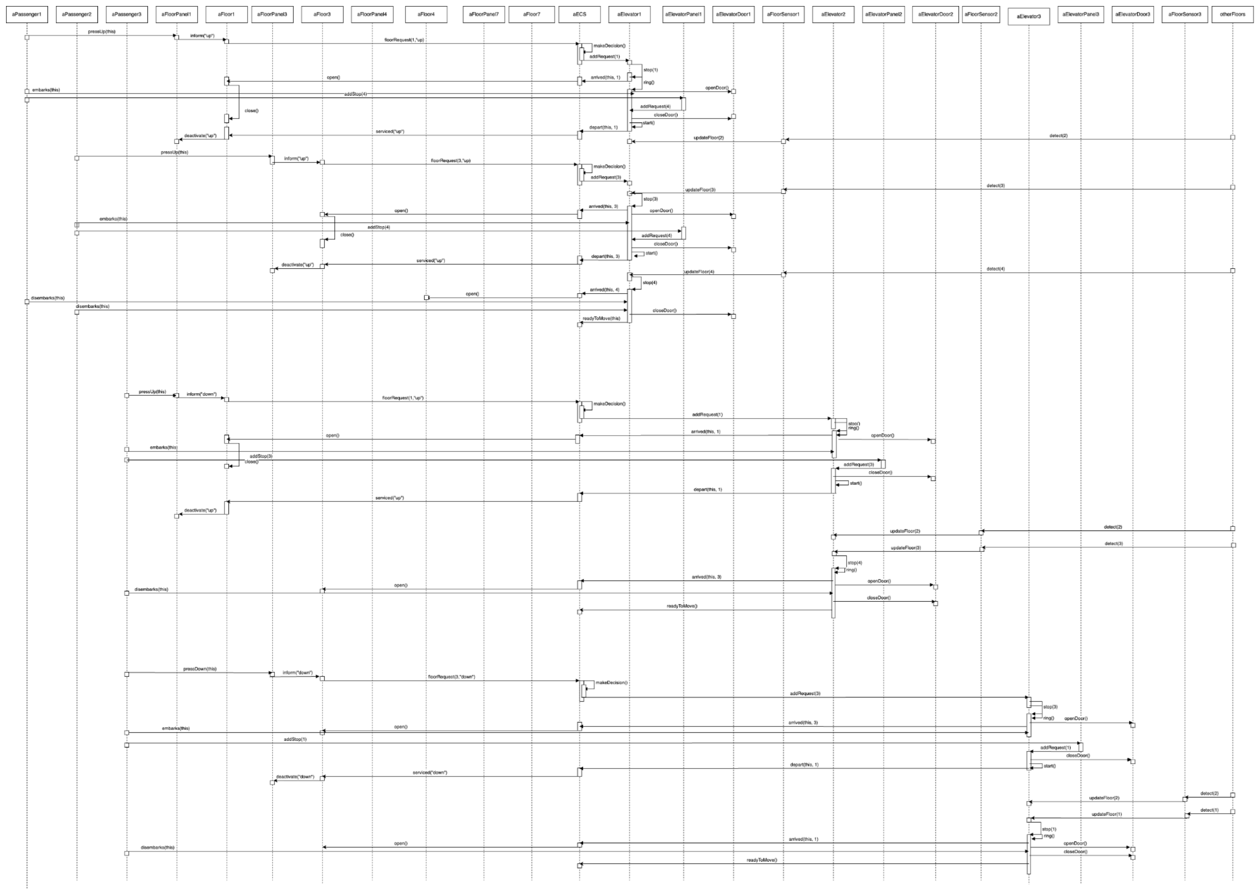
Preconditions for this success scenario include Elevator 1 and Elevator 2 are ready to move and on floor 1. Elevator 3 is ready to move and on floor 7. Passenger 1 and Passenger 2 start on floor 1 and Passenger 3 is starting on floor 7. When a passenger requests an elevator they do initialize the request by using a FloorPanel. The floor panel then notifies the Floor and the Floor notifies the ECS. The ECS contains the logic regarding decision making and will decide how to allocate the requests upon receiving one. The first request received by the ECS in this diagram is added to aElevator1. aElevator1 receives the request and because he is already at the given floor stops, rings its bell and notifies the ECS that it has arrived. The ECS takes this information and opens the according floor door via accessing aFloor1. As the ECS follows the Mediator pattern the ECS handles main communication between the large components such as Elevators and Floors. Following the opening of the doors a passenger embarks and adds a stop through the ElevatorPanel, the ElevatorPanel then sends this request to the Elevator and adds it to its requests. After a few seconds the elevator closes its doors and prepares to leave. It calls start() and alerts the ECS that it is departing and where from. The ECS uses this information to close the corresponding floor door and deactivate the previously activated panel buttons. aElevator1 is now started and begins moving to its destination floor, once it arrives it follows a similar loop as before. The elevator makes a call to stop and rings its bell indicating that the elevator has arrived and will open its doors shortly. aElevator notifies the ECS that it has arrived and the floor that it is arriving on. The ECS then communicates this information to the correct Floor. Entering a similar loop as before, opening the floor door, the passenger now disembarks (for simulation purposes leaves the elevator) and shortly after the elevator closes its doors. At this point, aElevator1 has now completed all its assigned requests and indicates to the ECS that it is readyToMove().

The previous paragraph indicated the general loop for success scenario of Elevators picking up and transporting passengers. This continues for aPassenger2 who starts similarly on floor 1 but now aElevator2 is the closest and available elevator so the ECS allocates the request to aElevator2. aElevator2 then completes the request in a similar manner. aPassenger3 starts on floor 7 and makes a request to head down. The ECS decides to hand this request to aElevator3 and aElevator3 completes the same loop completing the request.

Some design decisions that are relevant within the success scenarios are adding requests made via the elevator panel directly to the elevator. These requests are not routed to the ECS but will be available to the ECS when it's making a future decision. This is because in real life use cases if a passenger enters a request within a particular elevator the passenger will never be transported to a different elevator to service that request. Therefore, within the simulation this logic remains and the passengers request remains in its current elevator.

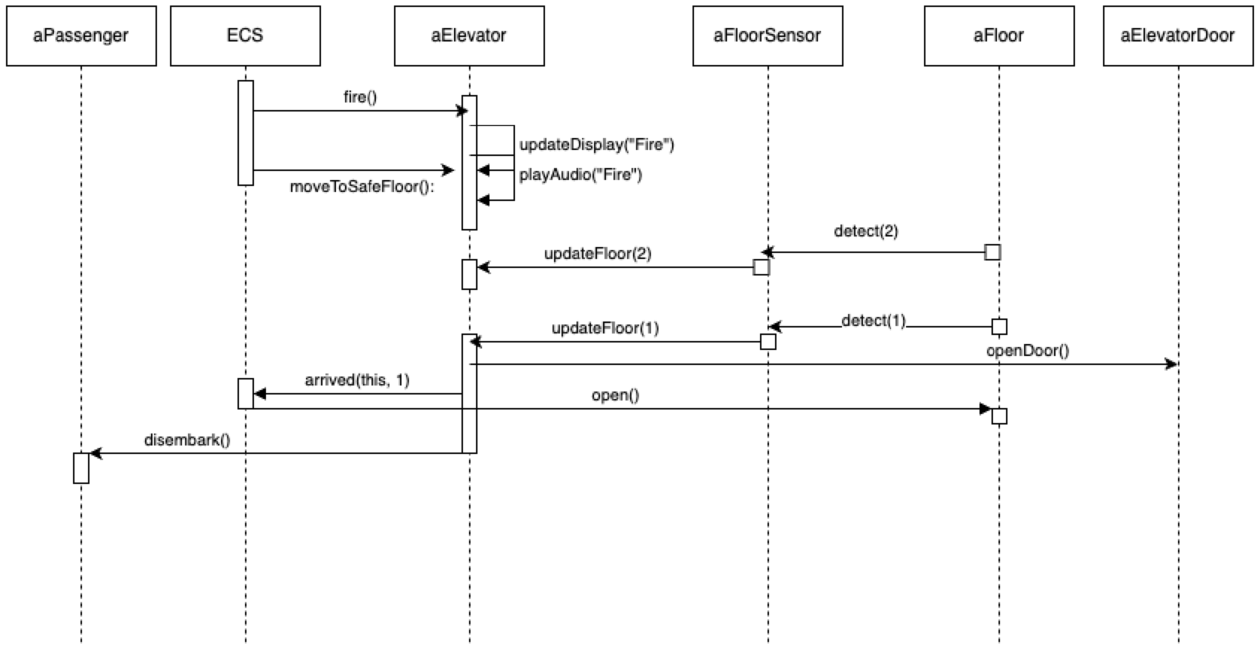
Note within the Sequence diagram some details are omitted for relevancy sake such as door obstruction, internal calls missing from closeDoor() or overload checks when passengers embark on the elevator. Additionally, an object named otherFloors was introduced to enable a clear way of traveling between floors for the elevators.

## Success Scenario 2



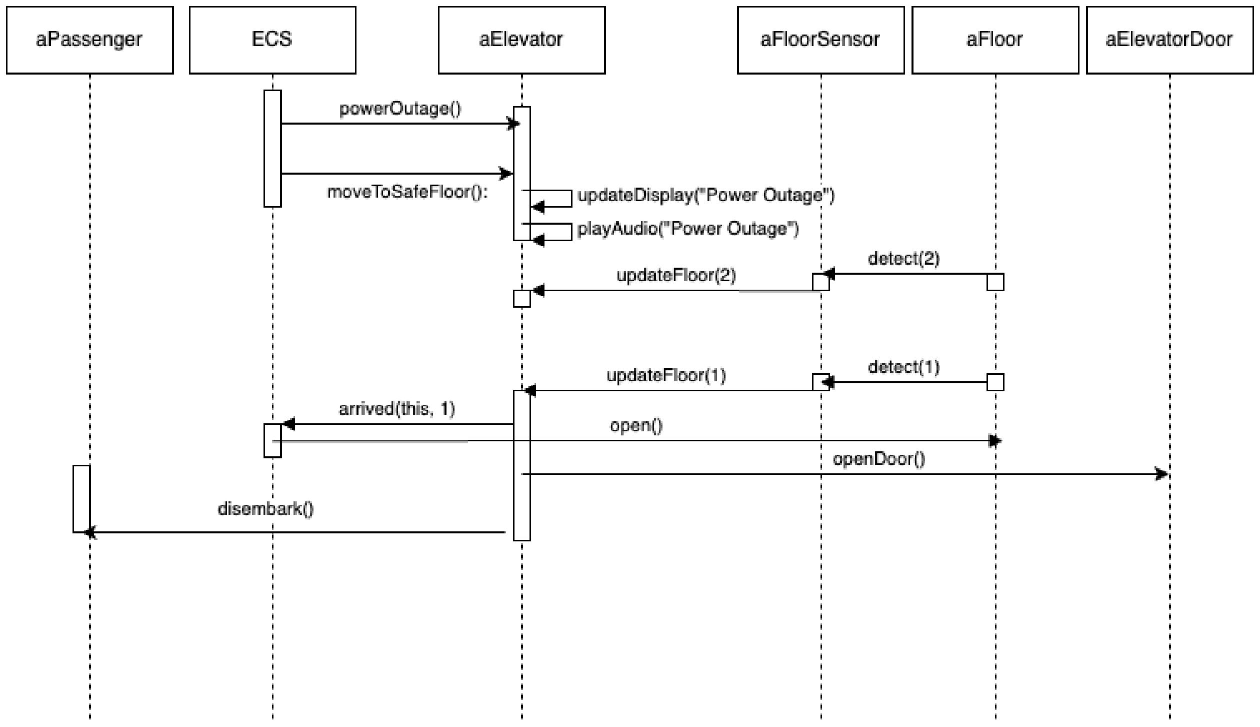
Preconditions for this success scenario include Elevator 1 and Elevator 2 are on floor 1 and Elevator 3 is on floor 3. Passenger 1 begins on floor 1, Passenger 2 begins on floor 3 and Passenger 3 begins on floor 1. The loop for transporting passengers remains mostly the same as success scenario 1, as a result some details will be omitted. aPassenger1 begins this diagram with a request to go up from floor 1, the ECS allocates this request to aElevator1 and aElevator1 begins servicing the request. While in motion aPassenger2 who is on floor 3 requests to go up. The ECS receives this request and allocates this request to aElevator1 as it is closest and heading in the same direction, making aElevator1 the logical choice for the ECS. aElevator1 then completes the requests from aPassenger1 and aPassenger2, these two passengers disembark on floor 4. aPassenger3 then requests to up from floor 1, the ECS receives this request and allocates it to aElevator2, after the elevator arrives aPassenger 3 enters and adds a request to go to floor 3 and aElevator2 completes the request. After arriving at floor 3, aPassenger3 then requests to head down from floor 3. The ECS receives this request and decides between aElevator2 and aElevator3 as they are both on floor 3 and ready to move. Since both decisions are viable the ECS chooses aElevator3 as it has not recently been used. The reason behind this selection provides some insight on how the ECS decides, it will use a modified queue based system for the elevators that also takes in account ready to move status, distance, and the direction the elevator is traveling in. Back to the process, aElevator3 arrives at aFloor3 and aPassenger3 enters. aPassenger3 presses his destination on the ElevatorPanel which is communicated to the elevator. aElevator3 completes arrives at the destination floor and aPassenger3 exits the elevator. The elevator then communicates to the ECS that it has completed all its requests and communicates a readyToMove() command to the ECS.

## Fire



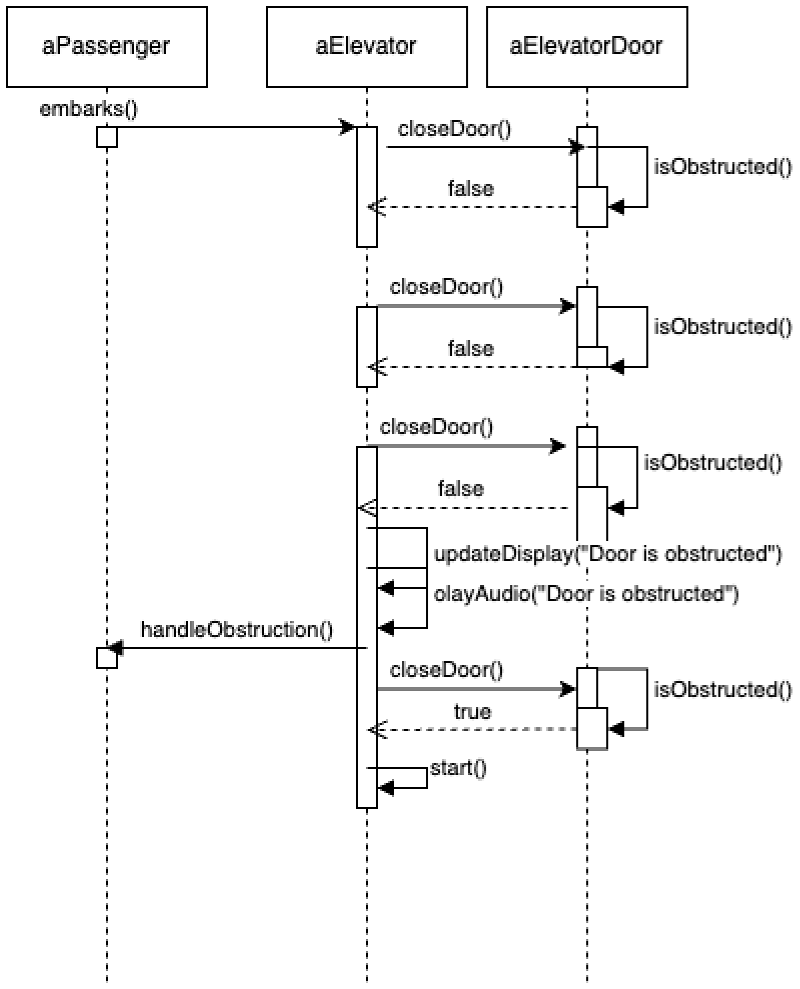
This diagram shows how an elevator handles a fire emergency. Preconditions for this sequence diagram include the ECS receiving a fire report. The ECS then triggers a fire() call to the elevators (in this diagram to aElevator). The elevator handles updating its display and playing audio messages regarding the fire and the ECS communicates to the elevator which floor to go to. The system for handling emergencies is simple and straightforward, after receiving a request the elevator promptly heads to the safe floor. Once it arrives, the elevator notifies the ECS and any passengers inside disembark().

## Power Outage



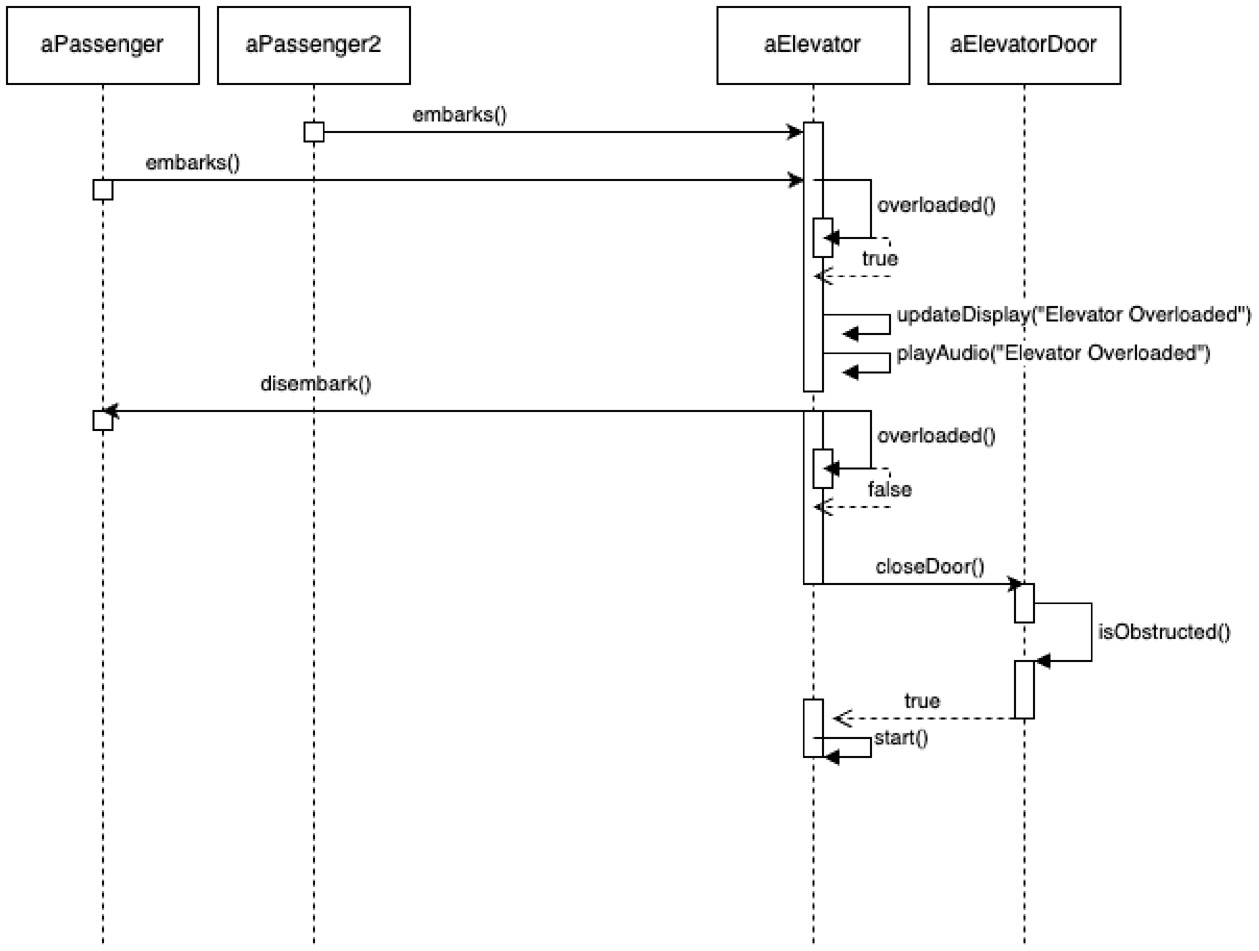
This diagram shows how an elevator handles a power outage. Preconditions for this sequence diagram include the ECS receiving a power outage report. The ECS triggers a powerOutage() call to the elevators. Following the initial call, this sequence diagram operates very similarly to a fire() call. After receiving the powerOutage() call the elevators switch to backup power then are told to move to the safe floor by the ECS. The elevator plays audio messages and updates its display. Once the elevator reaches the safe floor, it communicates to the ECS that it has arrived, opens its doors and the passengers exit.

## Obstruction



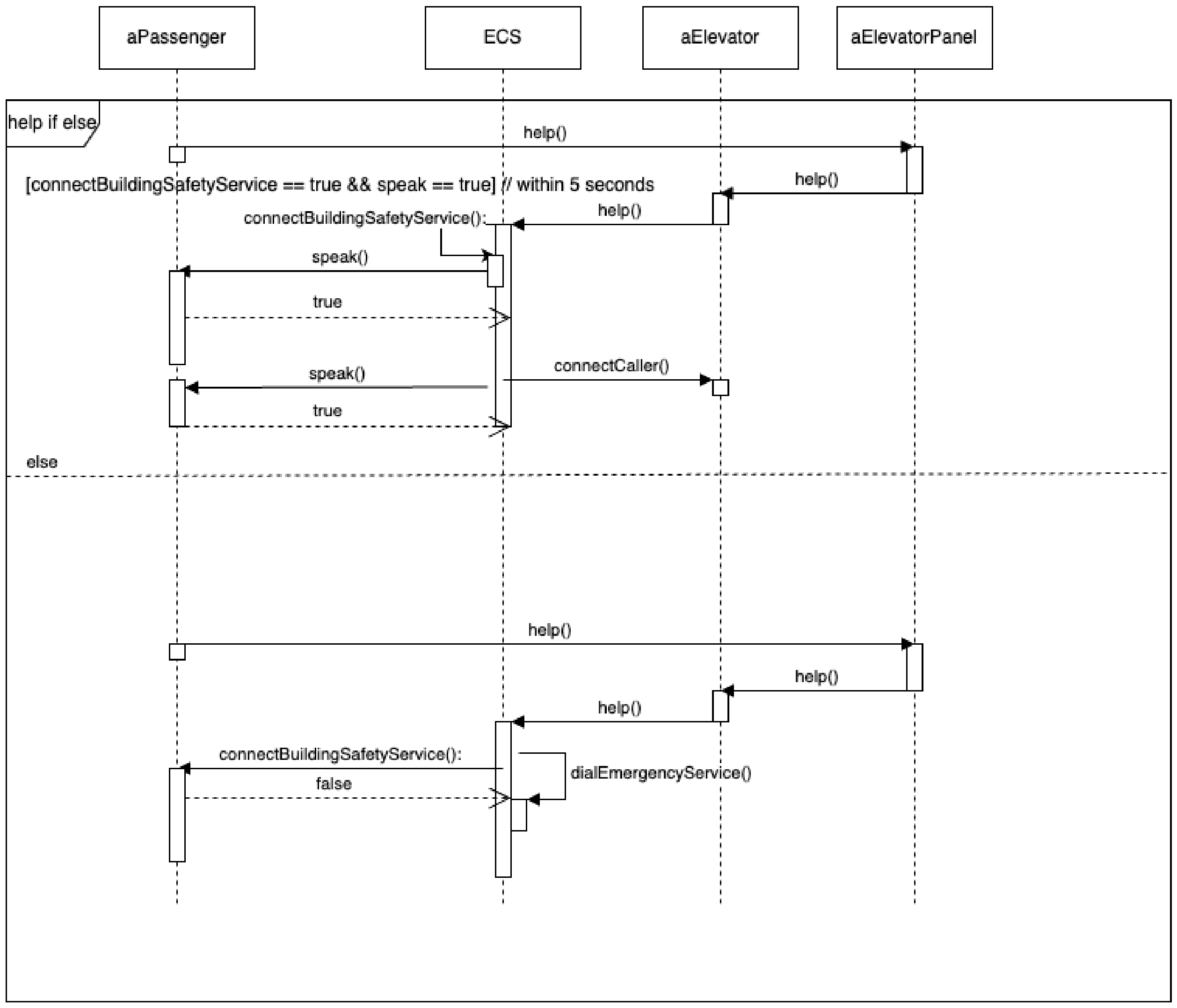
This sequence diagram shows how an elevator handles a door obstruction. Preconditions for this sequence diagram are that the elevator has already arrived at the passengers floor and opened its doors. Following a aPassenger entering the elevator as the door closes an internal call occurs which checks if the door is obstructed. Since closeDoor() returns false this means the door has not closed. The elevator attempts to close its door again, after several closeDoor() returning false the elevator audibly and visually notifies passengers of an obstruction. The elevator then calls upon the passenger to handleObstruction(). Now that the obstruction is handled, the next closeDoor() call returns true indicating the door has closed.

## Overloaded



This sequence diagram indicates how an elevator handles being overloaded. Preconditions for this sequence diagram are that the elevator has already arrived at the passengers floor and opened its doors. The elevator updates its weightCapacity based on the passengers entering. In the diagram the second passenger entering increases the weight capacity beyonds its limits triggering an internal overload() call within the elevator. The elevator then audibly and visually notifies passengers and instructs aPassenger to disembark() . Following the passenger leaving the elevator its weight capacity is now in limits and can close its door and indicate to the ECS its ready to move. The elevator will need a weight detection system, either as a sensor attached to the elevator floor or as reading the passenger data directly. This is not its own class as that would  over complicate the system for a simple objective. This can be completed as internal logic within overload.

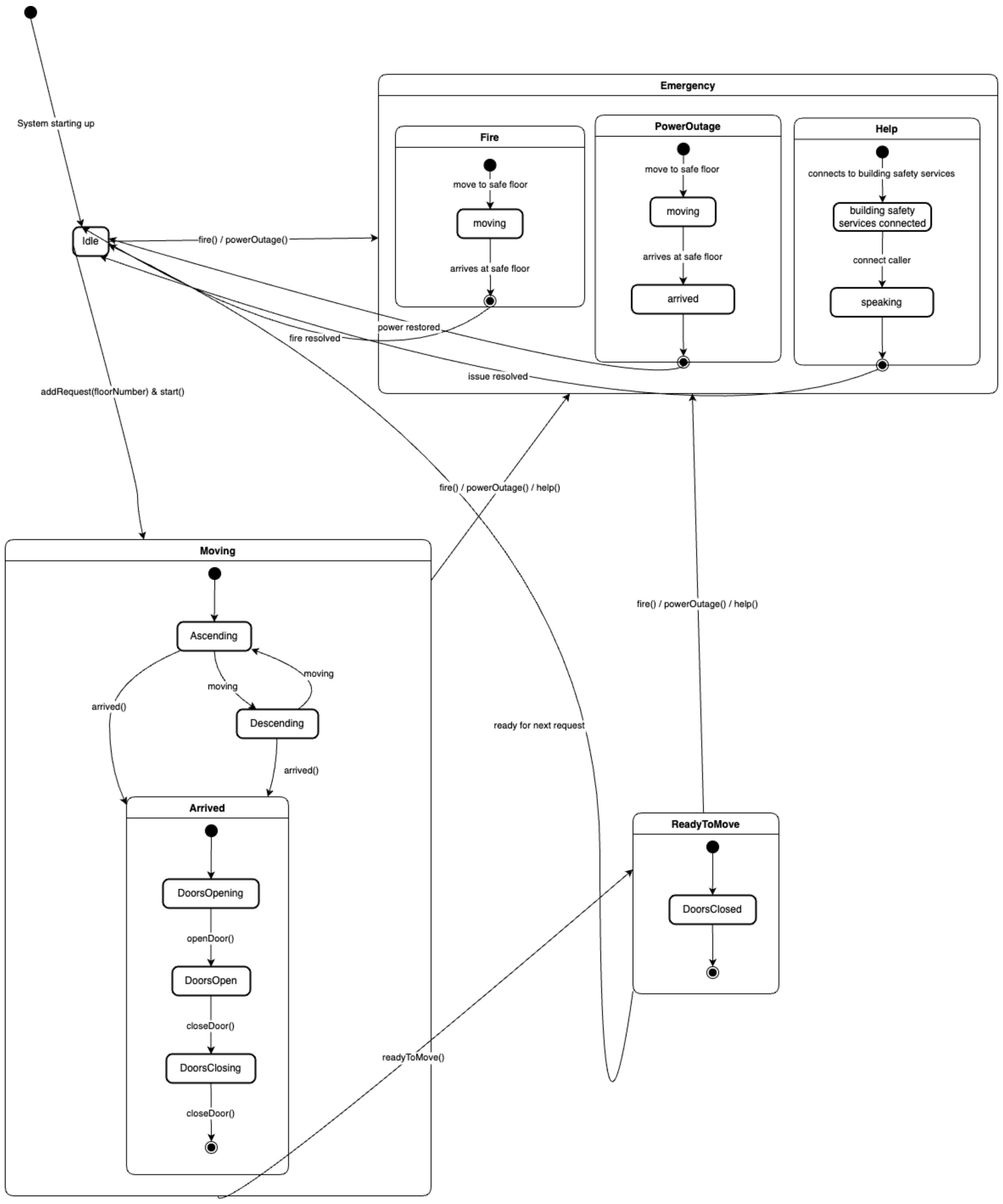
## Help



This sequence diagram indicates how the system handles a help request. The preconditions for this diagram is that the passenger is already within the elevator. The sequence starts with the passenger pressing the help button on the elevator panel. Following the button press the elevator notifies the ECS and the ECS attempts to connect the passenger to the building safety services. If the connection is successful and after the initial connection the passenger is responsive the ECS connects the caller to the elevator and the passenger then speaks to receive help. If either the building safety services or the passenger are unresponsive, the ECS then calls dialEmergencyService() which dials for emergency services.

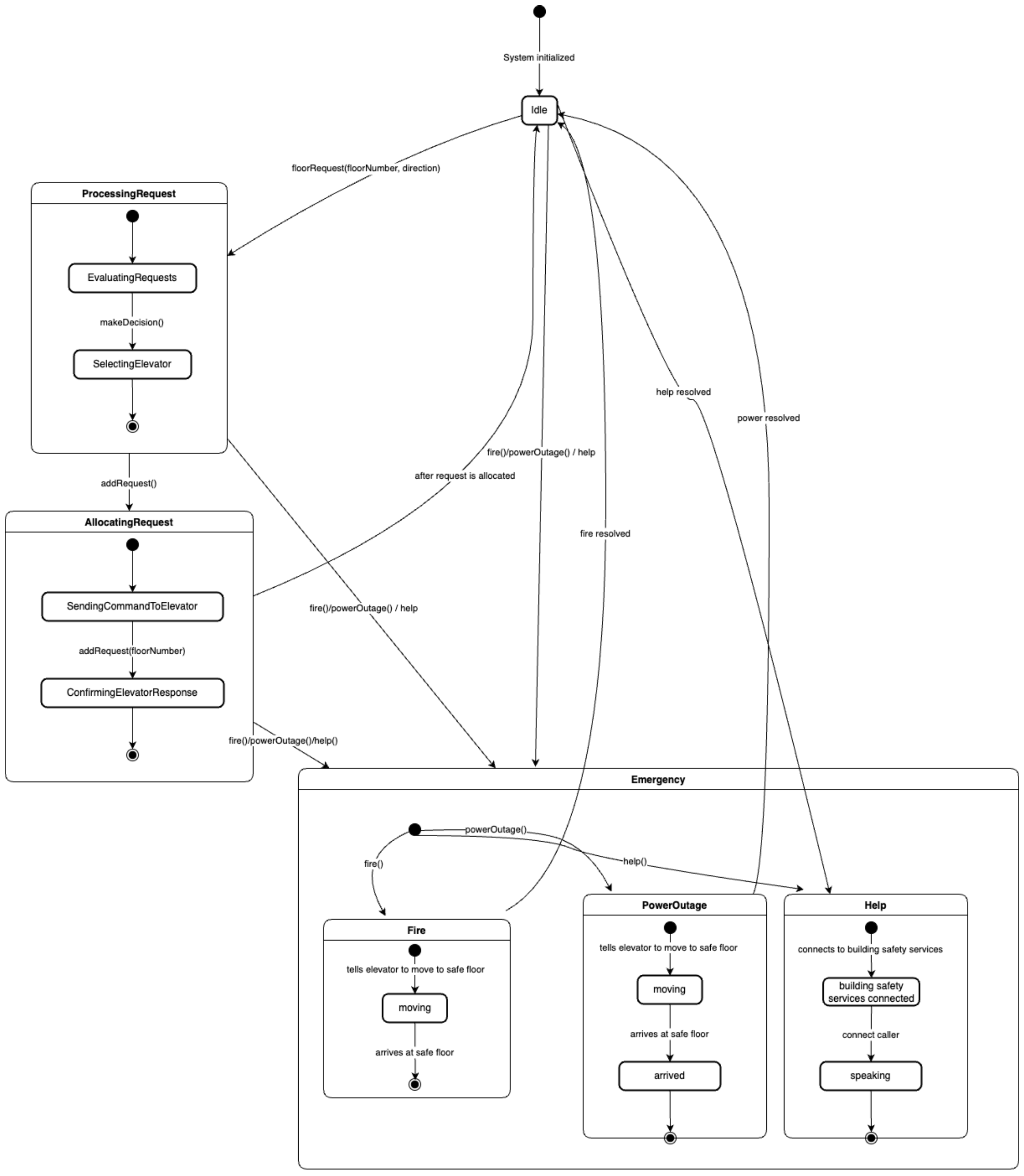
# State Diagrams

## Centralized



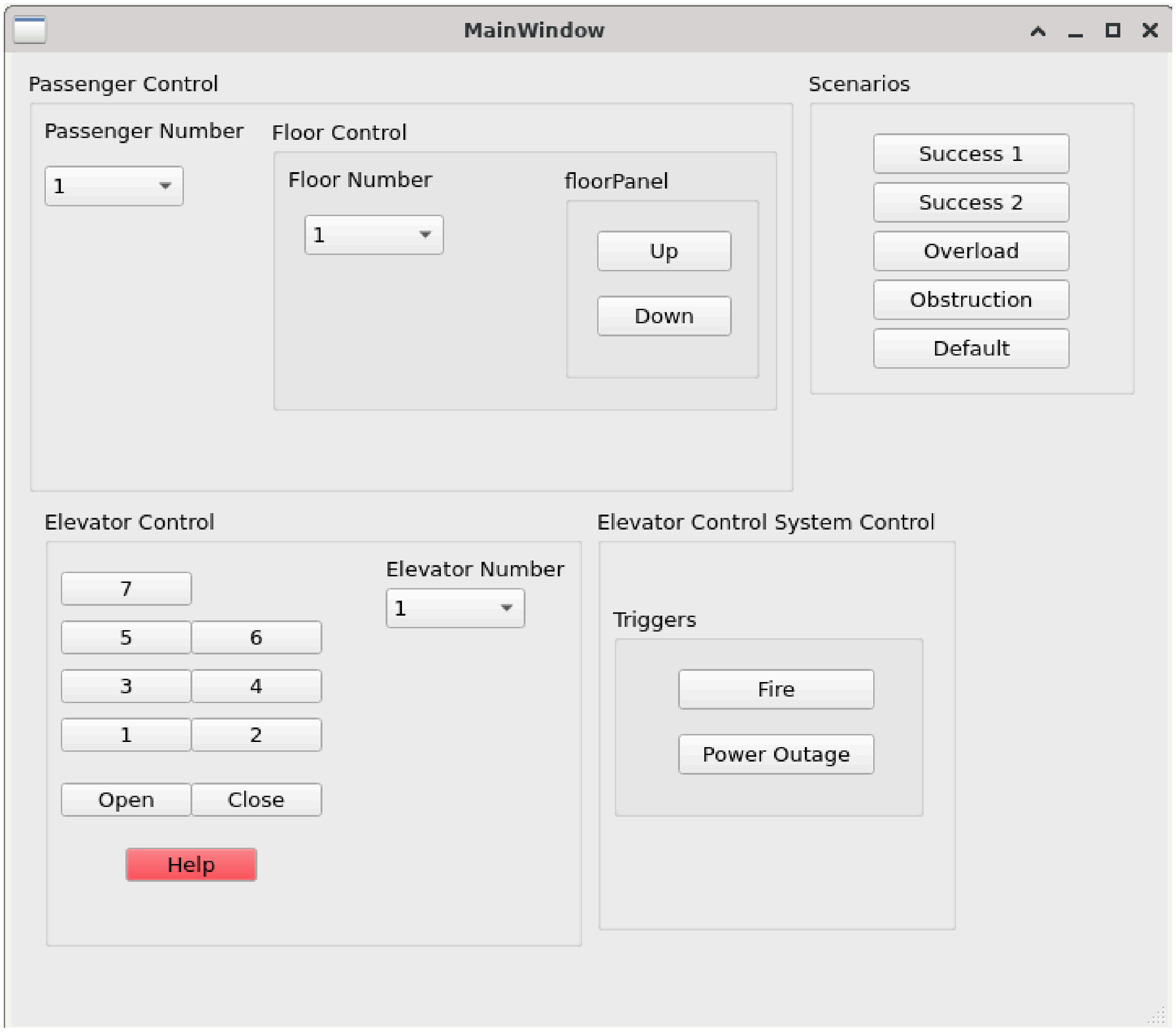
The centralized state diagram shows the four primary states idle, moving, ready to move and emergency. The elevator state remains like this until a request is added and it calls start. After which it enters a moving state where the elevator is either moving up or down towards its destination. As a sub state of moving the elevator enters an arriving state which involves opening and closing of the doors to allow a passenger to exit. The elevator exits the moving state when a arrived() is called and the doors have closed, following this the elevator enters a ready to move state, this state is similar to idle but indicates the elevator just finished all its requests. Some additional states include the emergency states, within the emergency state are the three sub states: fire, power outage, and help. The emergency state can be reached from idle, moving or ready to move.

## Controller



The controller state diagram shows the state diagram for the ECS within the centralized design. This state diagram shows the four primary states of the ECS idle, processing request, allocating request, and emergency. The controller starts in an idle state after being initialized, this is until it receives a floorRequest() call from a floor. The controller then enters a processing request state where it evaluates and selects the elevator to handle the request. After selection, the elevator enters an AllocatingRequest state where it adds the request to the elevator and updates the elevator's requests. The controller can enter an Emergency state from any of its states, this emergency state operates similarly as it did in the centralized and decentralized design however here the controller is telling the elevators of the emergency (triggering their emergency state) and where to go.

## GUI



The GUI above shows the layout.The GUI above shows the layout. The layout is split between passenger control, elevator control, and elevator control system control. With exception to the "Scenarios" section in the top right, this section helps enable pre-conditions for certain scenarios. The passenger control section handles most passenger related tasks and allows the user to select a passenger of interest. In the passenger section the user also can access floor controls, users can select which floor they would like the passenger to interact with via drop downs and the up/down buttons.The passenger number selected is maintained in the Elevator Control section. Within elevator control if the passenger selected is in the elevator selected the passenger can interact with the elevator through the destination buttons, open/close buttons and the help buttons. In the elevator control system control section the user can trigger scenarios such as a fire or a power outage. In the section named “Scenarios” the user can run some scenarios such as Overload and Obstruction as well as set some preconditions for testing. Success 1 and Success 2 enable pre-conditions required for those scenarios and Default resets elevator positions back to default (floor 1).

# Traceability matrix

## Centralized Matrix

| ID | Requirements | Use Case | Implemented By | Tested By | Description |
|---|---|---|---|---|---|
| 1.1 | A building has M elevators and N floors. | N/A | ECS, Elevator, Floor | N/A | The ECS will create the elevators and floors according to inputs |
| 1.2 | When a button is pressed it illuminates, and remains illuminated, until an elevator arrives to transport the customers who, at this floor, have requested an elevator going in a certain direction. | Regular Elevator Use | MainWindow, ECS, Elevator, FloorPanel, Floor, Passenger | GUI, ComboBoxes, PushButtons, FloorPanel pressUp() and pressDown() | The Passenger presses a button on the FloorPanel. The FloorPanel saves the direction within its state and notifies the Floor which notifies the ECS. The ECS is now notified and makes a decision for which elevator to add the request to. An elevator receives this request and when it arrives it notifies the ECS and the ECS notifies the |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Floor. The Floor also recognizes to switch the illumination states when the elevator arrives. |
| 1.3 | When the elevator arrives, it rings a bell, opens its doors (the elevator and floor doors) for a fixed time (10 seconds) allowing people to exit or board, rings the bell again, closes its doors and proceeds to another floor. | Regular Elevator Use | MainWindow, Elevator, ElevatorDoor, Passenger | GUI buttons, arrived() method in ECS | The Elevator calls ring() and open() to notify the Passenger that it has arrived. Passengers then call embark() to get onto the Elevator. The Elevator will time how long since open() was called to abide by the 10 second open rule. After, it will attempt to close the ElevatorDoor. |
| 1.4 | Once on-board passengers select one or more destination floors using a panel of buttons; there is one button for every floor. | Regular Elevator Use | MainWindow, ElevatorPanel, Elevator | GUI elevator destination buttons, addRequest() in Elevator | The ElevatorPanel contains the destination options from 1..N where N is the number of floors. Passengers call method with the floor they want to go to on the ElevatorPanel, the panel then relays this information to the Elevator. |
| 1.5 | The elevator has a display which shows passengers the current floor of the elevator. | Regular Elevator Use | MainWindow,Elevator, FloorSensor | GUI buttons, Elevator move() function updateDisplay() in Elevator | The Elevator updates its display when it is notified by the FloorSensor of a new floor. |
| 1.6 | There is also a pair of buttons on the elevator control panel marked “open door” and “close door”. These buttons can be used by a passenger to | Passenger Uses Manual Door Operation | MainWindow, Elevator, ElevatorPanel, ElevatorDoor | manualOpen() and manualClose() in ElevatorPanel | The ElevatorPanel contains methods called manualOpen() and manualClose(). These buttons will override default behavior, when |

| | | | | | |
|---|---|---|---|---|---|
| | override the default timing of the doors. The door will remain open beyond its default period if the “open door” button is held depressed; the doors can be closed prematurely by pressing the “door close” button. | | | | pressed the ElevatorPanel informs the Elevator, which will open or close the ElevatorDoor. |
| 1.7 | Inside the elevator there is also a help button linked to building safety service. | Passenger Uses Elevator Help Button | MainWindow, Passenger, Elevator, ElevatorPanel, ECS | Help button, help() method in ElevatorPanel | The Passenger can invoke help() via the ElevatorPanel. The ElevatorPanel then informs the Elevator which informs the ECS. The ECS will then connect the passenger to building safety services. |
| 2.1 | Each elevator has a sensor that notifies it when it arrives at a floor. | Regular Elevator Use | MainWindow, Elevator, Floor, FloorSensor | Elevator move() method and detect() method in FloorSensor | The floors notify the FloorSensor when it arrives at a Floor. The FloorSensor relays this information to the Elevator to notify the Elevator when it reaches the floor. |
| 2.2 | The elevator control system should ensure that the group of elevators services all (floor and on-board) requests expeditiously. | Regular Elevator Use | ECS | ElevatorControlSystem() makeDecision() and FloorPanel pressUp() or pressDown() | When the ECS receives a request it runs through its algorithm to pick the optimal choice for the given request. |
| 3.0 | Each elevator has a display and an audio system. The display shows the current floor number and warning messages that are | Regular Elevator Use | Elevator | updateDisplay() and playAudio() in Elevator | The Elevator updates its display whenever it is informed about a new floor or when informed about something requiring |

| | | | | | |
|---|---|---|---|---|---|
| | synced with audio warnings. | | | | a change in the visual messages. The Elevator also will play audio for warnings. |
| 4.0 | Help: The control system receives a “Help” alarm signal from an elevator indicating that the “Help” button has been pressed. In that case, the passenger is connected to building safety service through a voice connection. If there is no response from building safety within 5 seconds or if there is no response from a passenger a 911 emergency call is placed. | Passenger Uses Elevator Help Button | MainWindow, Passenger, Elevator, ElevatorPanel, ECS | Help button help() in ElevatorPanel | The Passenger can invoke help() via the ElevatorPanel. The ElevatorPanel then informs the Elevator which informs the ECS. The ECS will then attempt to connect to building safety services. If building safety services return false indicating the connection failed 911 is called. If the connection is good and the passenger doesn’t respond within 5 seconds 911 is called. If both return true the Passenger is connected. |
| 5.0 | Door obstacles: If the light sensor is interrupted when the door is closing, the control system stops the door from closing and opens it. If this occurs repeatedly over a short period of time, a warning is sounded over the audio system and a text message is displayed | Elevator Door Obstacle Detection | MainWindow, Elevator, ElevatorDoor, Passenger | Obstruction button via setObstruction() and checkObstruction in Elevator/ElevatorDoor | The ElevatorDoor checks for obstructions on closing and notifies the elevator via return value (if the door fails to close it returns false). If this occurs repeatedly , the Elevator notifies the Passenger through audio and visual messages, the Passenger then can use handleObstruction() to remove the obstruction. |

| | | | | | |
|---|---|---|---|---|---|
| 6.0 | Fire: The control system receives a "Fire" alarm signal from the building and commands all elevators to move to a safe floor. Similarly, a "Fire" alarm signal from the elevator itself will cause that elevator to go to a safe floor. In both cases an audio and text message are presented to passengers informing them of an emergency and asking them to disembark once the safe floor is reached | Elevator Fire Alarm Signaled | MainWindow, ECS, Elevator | Fire button fire() in ECS | The ECS receives a fire() method call and informs all Elevators overriding all their current requests and instructs them to move to a safe floor. The Elevator then moves to the safe floor. |
| 7.0 | Overload: The control system receives an "Overload" alarm signal from an elevator if the sensors indicate that the passenger or cargo load exceeds the carrying capacity. In that case, the elevator does not move and an audio and a text messages are presented to passengers asking for the load to be reduced before attempting to move again. | Elevator Overload Detection | MainWindow, Elevator, Passenger | Overload method overloaded() in Elevator | Prior to closing its doors the Elevator checks if it's overloaded. If overloaded() returns true the Elevator updates its display and plays an audio message indicating the overload. The Elevator calls disembark() on its passengers until no longer overloaded |
| 8.0 | Power out: The control system receives a "Power Out" alarm signal. In | Elevator Power | MainWindow, ECS, Elevator | Power outage button powerOutage() in ECS | The ECS receives a powerOutage() method call and informs all Elevators |

| | | | | | |
|---|---|---|---|---|---|
| | that case, an audio and a text messages are presented to passengers informing them of the power outage. Each elevator is then moved to a safe floor and passengers are asked to disembark via audio and text messages. The battery backup power is sufficient to do all of this. | Outage Detected | | | overriding all their current requests and instructs them to move to a safe floor. The Elevator then moves to the safe floor. |