Smart Home Simulator

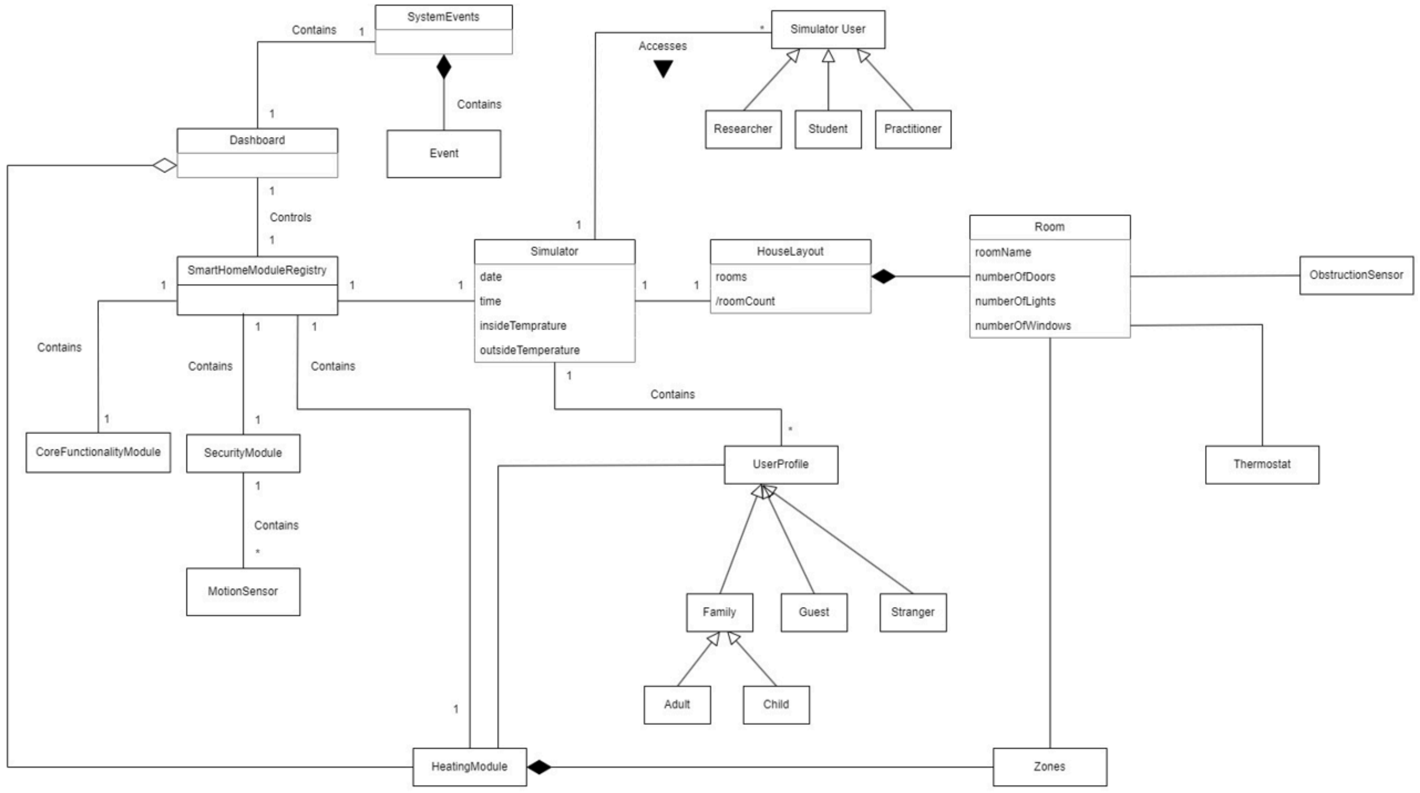
SOEN 343
Lab Section WJ-X

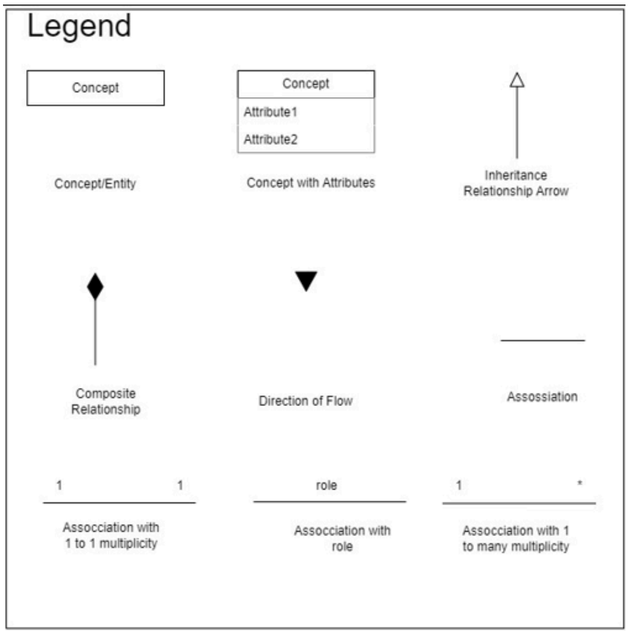
March 25th 2024

Clara Gagnon (40208598)(Team Leader) Role: Smart Home Heater Implementation, Design Patterns
Ahmad Elmahallawy (40193418) Role: System Architecture, Smart Home Heater Implementation, General Implementation
Gulnoor Kaur (40114998) Role: Domain Diagram
Liam Daigle (40207583) Role: Smart Home Dashboard Implementation
Vanessa DiPietrantonio (40189938) Role: Sequence Diagrams, Simulation Parameters Implementation, General Implementation
Jessica Beauchemin (40188873) Role: Use Case Model: Use Case Tables, Use Case Diagram, Sequence Diagrams, State Machine Diagrams, Activity Diagrams

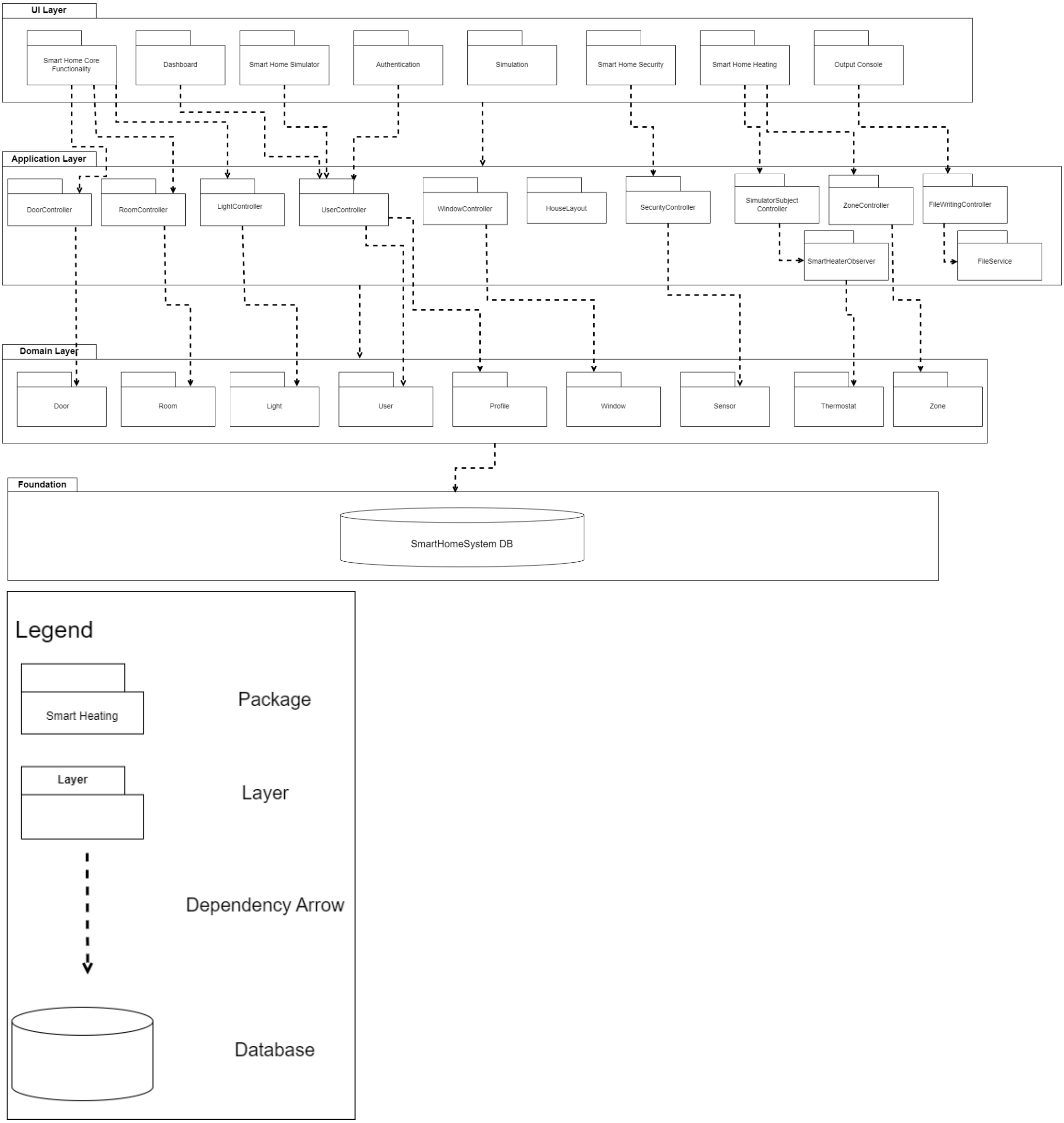
Table of Content

1.Domain Model Diagram





2. System Architecture



UI (View)
Description: The UI layer in our smart home simulation system, built with React frontend, encompasses the visual interface through which users interact with the simulated environment. It provides a user-friendly dashboard for controlling devices, monitoring the home's status, and adjusting simulation parameters. More specifically, the UI layer consists of frontend folders for the Smart Home Core, Dashboard, Smart Home Simulator, Smart Home Security and Smart Heating.

Application (Controller)
Description: The application layer, implemented with Spring Boot backend, serves as the backbone of the smart home simulation system. It manages user requests, coordinates interactions between the UI and domain layers, and executes business logic to simulate home behaviors and control devices. More specifically, the application layer comprises all the controllers for door, room, light, user, window, security and heating.

Domain (Model)
Description: The Domain layer serves as the backbone of our smart home system. It consists of the essential business logic that drives our operations. This includes managing data storage and retrieval for user preferences, device configurations, and simulation parameters in an efficient manner. Additionally, it enforces critical business rules and constraints, such as device usage limits and user permissions. Moreover, the Domain layer hosts the simulation engine, which governs the behavior of our simulated smart home environment over time and in response to user interactions.

Foundation
Description: The foundation layer represents the underlying infrastructure and core components upon which the rest of the system is built. This layer provides fundamental functionalities and services that support higher-level modules and applications. The main purpose of the foundation layer is to abstract away complexities, provide essential services, and ensure the stability, scalability, and maintainability of the entire system. It includes components responsible for managing data persistence, such as database management systems (DBMS).

3. Use Case Model

2.1 Use Case Tables - Smart Home Heating

ID:	**UC – 6.1**
Title:	Cooling and Heating Zones
Description:	Smart Home shall be divided up into zones for heating and cooling. The zones are set arbitrarily by the user through the SHH tab.
Primary Actor:	System, User
Preconditions:	The user is a parent and has access to zone settings. The system provides functionality for creating zones. All rooms exist and are open for modification.
Postconditions:	All rooms are assigned a zone.
Inputs:	User selects a zone for every room.
Outputs:	All rooms are assigned a zone.
Main Success Scenario:	1. Upon accessing the SHH tab, the user selects the "Cooling and Heating Zones" option. 2. The user creates a zone and selects the rooms to be included within the zone. 3. The user selects rooms to be included in each zone. 4. The system verifies the validity of the user's selections, ensuring that each room is assigned to a zone. 5. The system saves the zone configurations. 6. Confirmation is provided to the user, indicating the zones have been successfully created and configured.

ID:	**UC – 6.2**
Title:	Temperature Settings
Description:	SHH shall accept desired temperature settings for each zone, for 1 to 3 periods in the day.
Primary Actor:	System, User
Preconditions:	The user is a parent and has access to zone and temperature settings. The zone elements exist and are open for modification. All rooms exist and are open for modification. All zones have specific temperature settings. Each room has a temperature element. All temperature element exists and is open for modification. Each zone can access the temperature settings of every room. The up to three periods can be set (each with their own time frame with temperature setting). No periods can overlap in time.
Postconditions:	The temperature settings are set and modified for every zone.
Inputs:	The user selects the temperature for every zone and for which time period.
Outputs:	Temperature is modified in accordance with the zone and time period.

Main Success Scenario:	1. Upon accessing the SHH tab, the user selects the "Cooling and Heating Zones" option. 2. For each zone, the user sets the desired temperature settings by inputting the desired temperatures for cooling and heating modes. 3. The user also specifies the periods during which these temperature settings should be active. 4. The system verifies the validity of the user's selections, ensuring that each room is assigned to a zone and that temperature settings are within acceptable ranges. 5. The system saves the zone configurations and temperature settings. 6. Confirmation is provided to the user, indicating that the cooling and heating zones have been successfully configured.

ID:	**UC – 6.3**
Title:	Room’s Current Temperature Request
Description:	SHH shall display the current temperature of a room when requested by a user.
Primary Actor:	System, User
Preconditions:	The user selects the display room temperature option. The room exists.
Postconditions:	The temperature is displayed on the dashboard.
Inputs:	User requests to display room temperature.
Outputs:	Temperature of selected room.
Main Success Scenario:	1. The user selects the display temperature of a room on SHH tab. 2. The system displays the temperature of the selected room.

ID:	**UC – 6.4**
Title:	Indoor and Outdoor Temperature Monitoring
Description:	SHH shall monitor indoor and outdoor temperature.
Primary Actor:	System
Preconditions:	The temperature monitoring system is available and operational.
Postconditions:	Continuous monitoring of indoor and outdoor temperature.
Inputs:	Data from indoor temperature. Data from outdoor temperature.
Outputs:	Recorded indoor and outdoor temperature data.
Main Success Scenario:	1. The system continuously monitors indoor and outdoor temperatures. 2. The system records the room temperature and the outdoor temperature. 3. The collected data is processed and stored by the system.

ID:	**UC – 6.5**
Title:	Air Conditioning Auto Shut-Off
Description:	The system shall shut down the air conditioning if the temperature outside is cooler than the temperature inside during summertime.
Primary Actor:	System
Preconditions:	The system is monitoring the indoor and outdoor temperatures. Outside temperature is greater than 20˚C. At least one room is warmer than the outdoor temperature. Air conditioning is running.
Postconditions:	Air conditioning is turned off.
Inputs:	Rooms and outdoor temperature data collected by the system. Information indicating the current season.
Outputs:	Air conditioning is turned off.
Main Success Scenario:	1. The system continuously monitors the outdoor and indoor temperatures. 2. The system detects that at least one room has a temperature higher than the outdoor temperature. 3. The system identifies that it is Summer. 4. The system initiates the air conditioning shut-off procedure. 5. Confirmation is provided to the system indicating the successful shut off of the air conditioning.

ID:	**UC – 6.6**
Title:	Auto Open Windows
Description:	The system shall open windows if the temperature outside is cooler than the temperature inside during summertime and there is someone at home.
Primary Actor:	System
Preconditions:	The system is monitoring the indoor and outdoor temperatures. Outside temperature is greater than 20˚C. Air conditioning is turned off. At least one user is at home. At least one room is warmer than the outdoor temperature.
Postconditions:	All windows are opened.
Inputs:	Rooms and outdoor temperature data collected by the system. Information indicating the current season. At least one user is detected inside the home by the system.
Outputs:	Air conditioning is turned off. All windows are open.

Main Success Scenario:	1. The system continuously monitors the outdoor and indoor temperatures. 2. The system detects that at least one room has a temperature higher than the outdoor temperature. 3. The system identifies that it is Summer. 4. The system identifies that at least one user is at home. 5. The system checks if the air conditioning is running and if so, it initiates the air conditioning shut-off procedure. The system sends confirmation of the successful shut off of air conditioning. 6. The system opens all windows. 7. Confirmation is provided to the system indicating the successful opening of all windows.

ID:	**UC – 6.7**
Title:	Stop Windows from Closing in Case of Obstruction
Description:	SHH shall not open or close any windows if something is obstructing the desired path of the window.
Primary Actor:	System
Preconditions:	No window is currently blocked by any obstruction.
Postconditions:	Windows are opened or closed without obstruction.
Inputs:	Command to open or close window(s).
Outputs:	Alert indicating window is blocked or confirmation of successful opening/closing of window.
Main Success Scenario:	1. The system receives a command to open or close a window. 2. System checks if anything is obstructing the window. 3. If no obstruction is detected: a. the system proceeds to open or close the window and sends confirmation. 4. If there is an obstruction detected: a. the system alerts the user that there is an object obstructing the window.

ID:	**UC – 6.8**
Title:	Window Obstruction Notification
Description:	SHH shall send notifications to users if the windows cannot execute an order.(i.e., obstruction)
Primary Actor:	System
Preconditions:	The system detects an obstruction preventing window closure or opening.
Postconditions:	Notification is sent to users indicating the obstruction preventing the open/close window command.
Inputs:	Detection of window obstruction.
Outputs:	Notifying users of the obstruction preventing the open/close window command.
Main Success Scenario:	1. The system detects the obstruction of the window preventing its opening/closure. 2. The system sends a notification to users indicating the obstruction and the inability to close/open window.

ID:	**UC – 6.9**
Title:	Winter Energy Saving Mode
Description:	SHH must detect when you leave the house and lower the temperature settings during Winter to 17 degrees Celsius, leading to major energy savings.
Primary Actor:	System
Preconditions:	It is Winter; the outside temperature is less than 15˚C. All the rooms are empty (there are no users inside the home). Each room has a temperature element. All temperature element exists and is open for modification.
Postconditions:	The temperature of every room is set at 17 degrees Celsius. The system saves previous temperature settings.
Inputs:	System updates that every room is empty of users.
Outputs:	Temperature of every room is set at 17 degrees Celsius.
Main Success Scenario:	1. The system continuously monitors the occupancy status of each room. 2. System detects that all rooms are empty of users. 3. System accesses temperature settings in each room and saves the current temperature settings if no zones are set. 4. System sets the temperature to 17 degrees Celsius in every room. 5. Confirmation is sent to the system of successful adjustment of temperature settings.

ID:	**UC – 6.10**
Title:	Return Home Temperature Restoration
Description:	SHH must detect when users return home and restore the temperature settings to their previous state (before they left).
Primary Actor:	System
Preconditions:	System detects a user (user returns home). Winter energy-saving mode is active (UC-6.9). All temperature element exists and is open for modification. If the zones are not set, the system has saved the previous temperature settings.
Postconditions:	The temperature of every room is set according to the zone it is in.
Inputs:	System updates that at least one user has been detected inside a room.
Outputs:	Temperature settings of every room are restored.
Main Success Scenario:	1. The system continuously monitors the occupancy status of each room. 2. System detects that users have returned home. 3. System retrieves each zone's temperature settings or the previous temperature settings for each room. 4. System sets the temperature of each room to their respective previous values. 5. Confirmation is sent to the system of successful adjustment of temperature settings.

2.2 Use-Case Diagram for the Whole System

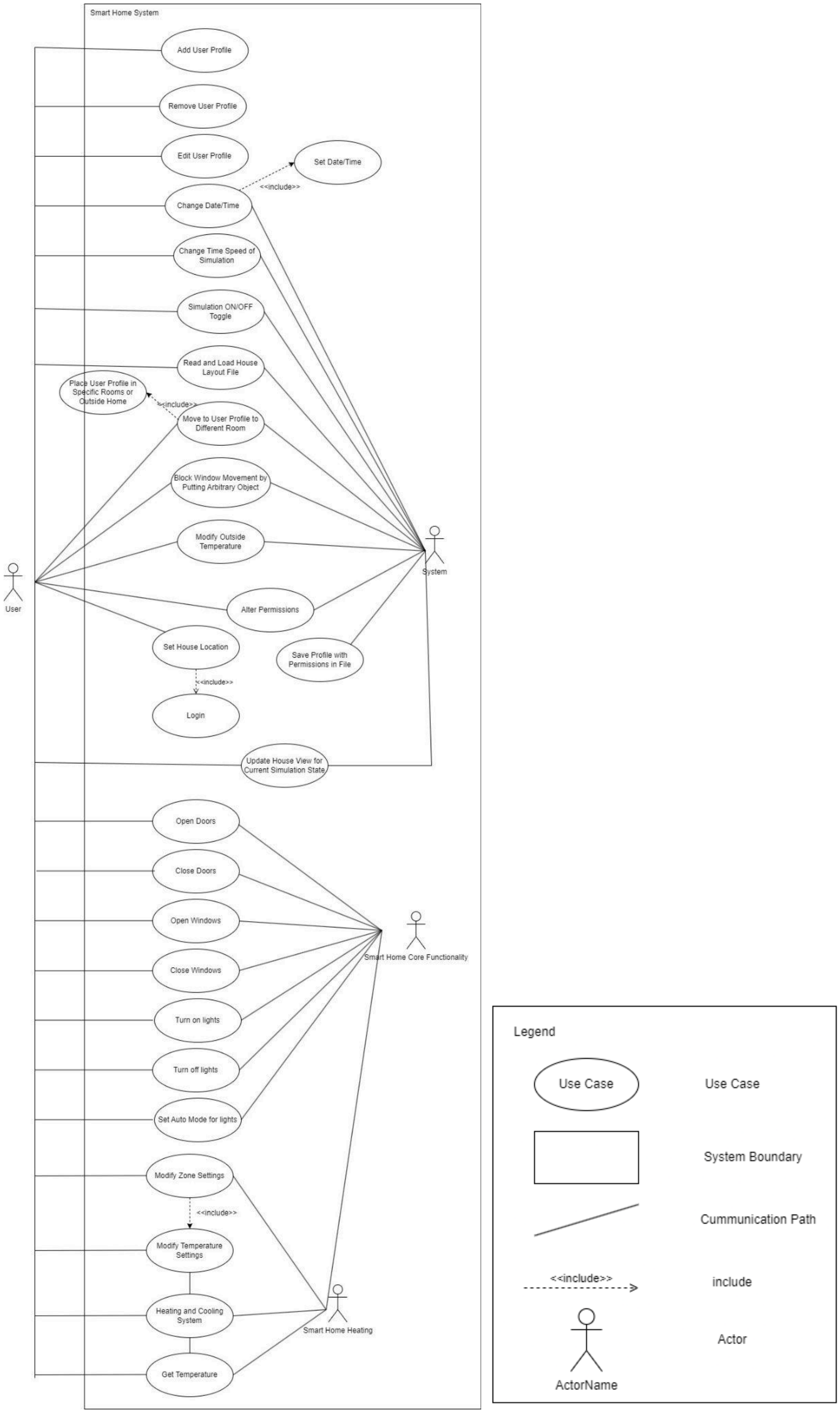


Figure 2: Use Case Diagram

2.3 Sequence Diagram for the Smart Home Core Functionality (SHC) Module

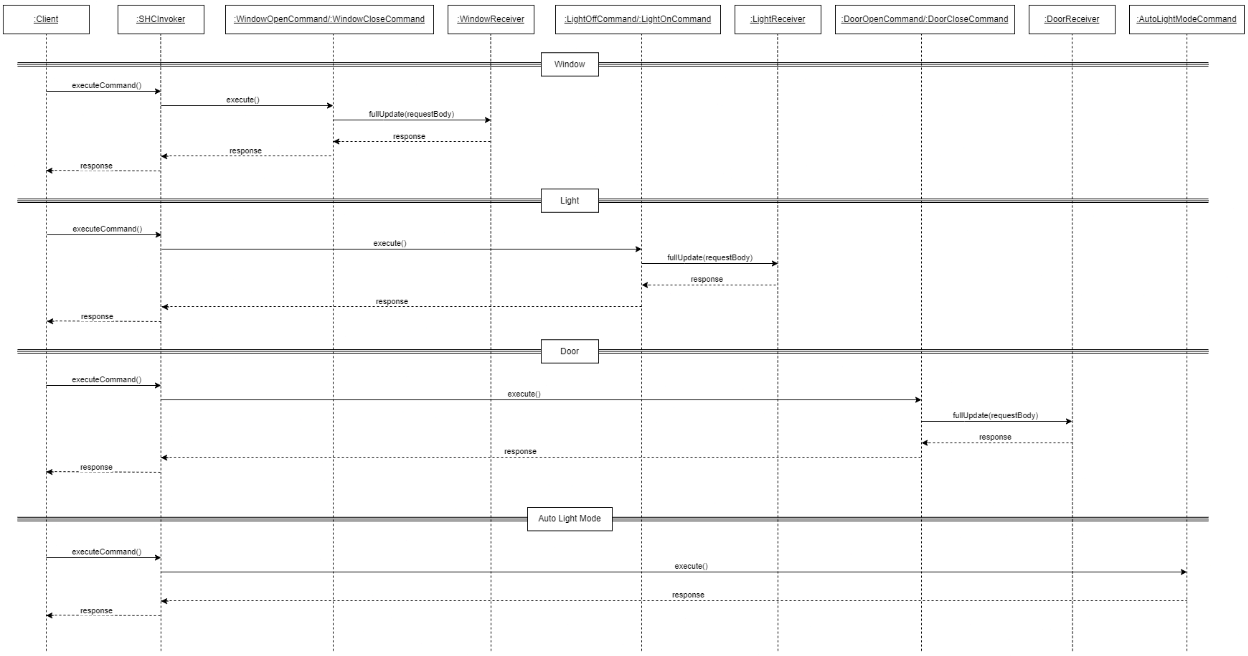


Figure 3: Sequence Diagram

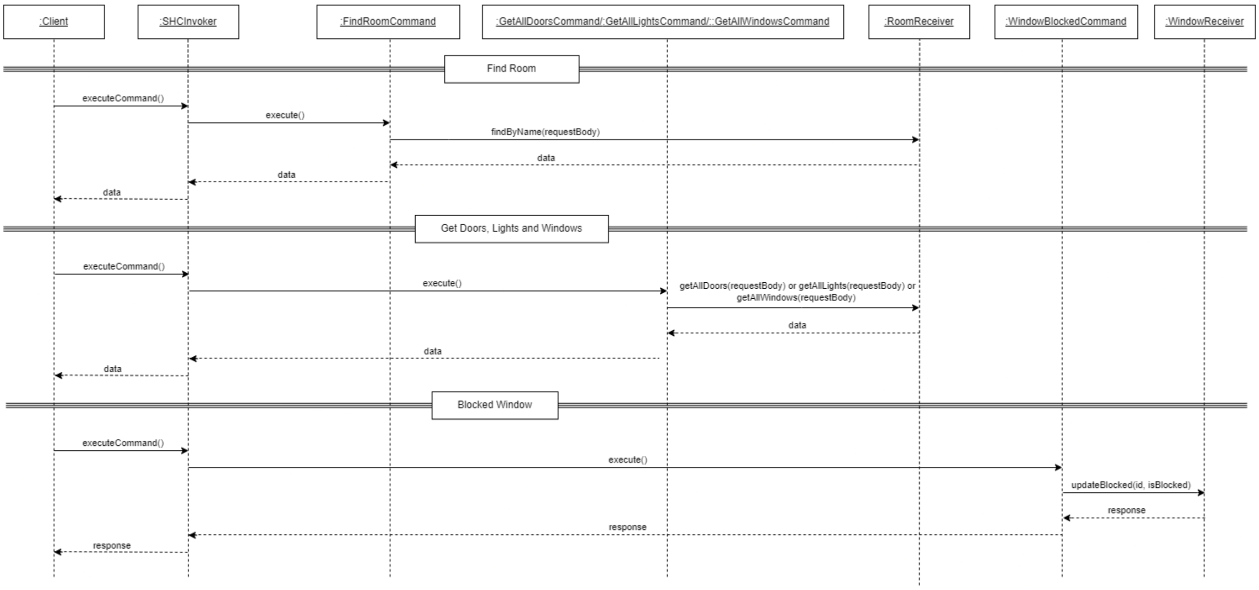


Figure 4: Note that this is an extension of Figure 3

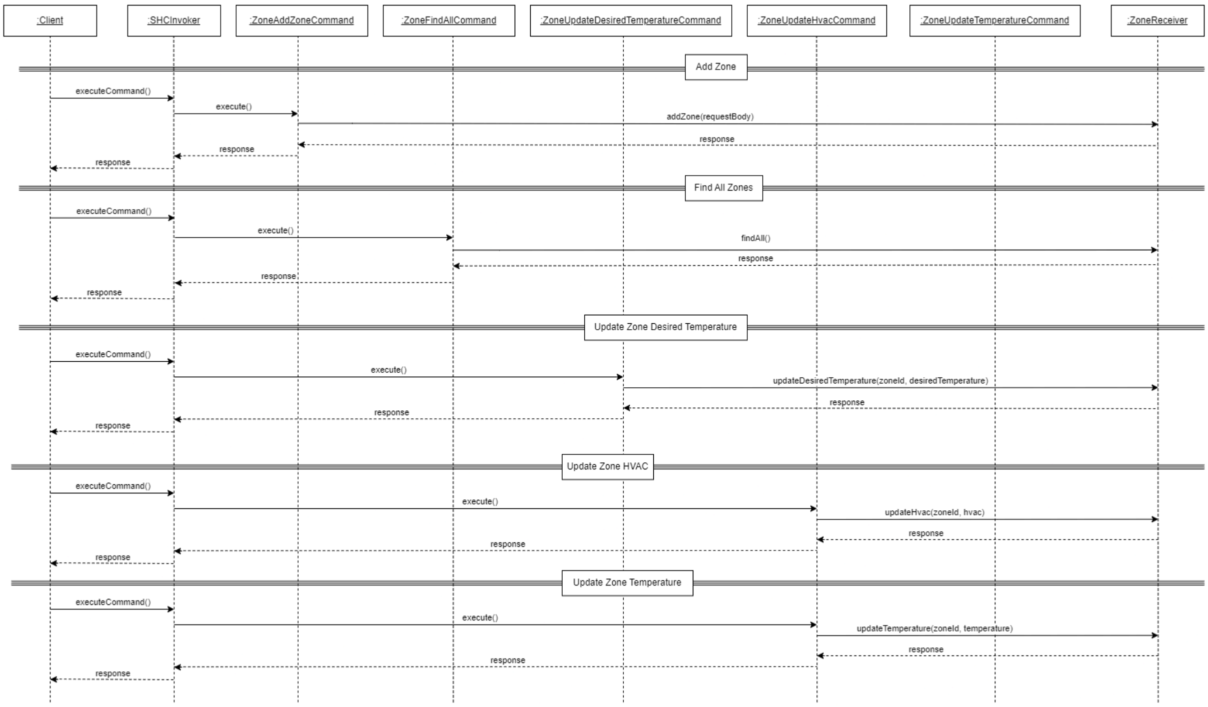
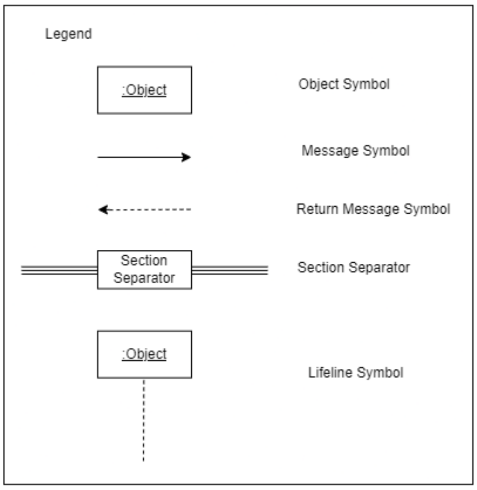


Figure 5: Note that this includes SHH in Sequence Diagram (extension of Figure 3 and 4)



2.4 State-machine diagram for the context of the simulation

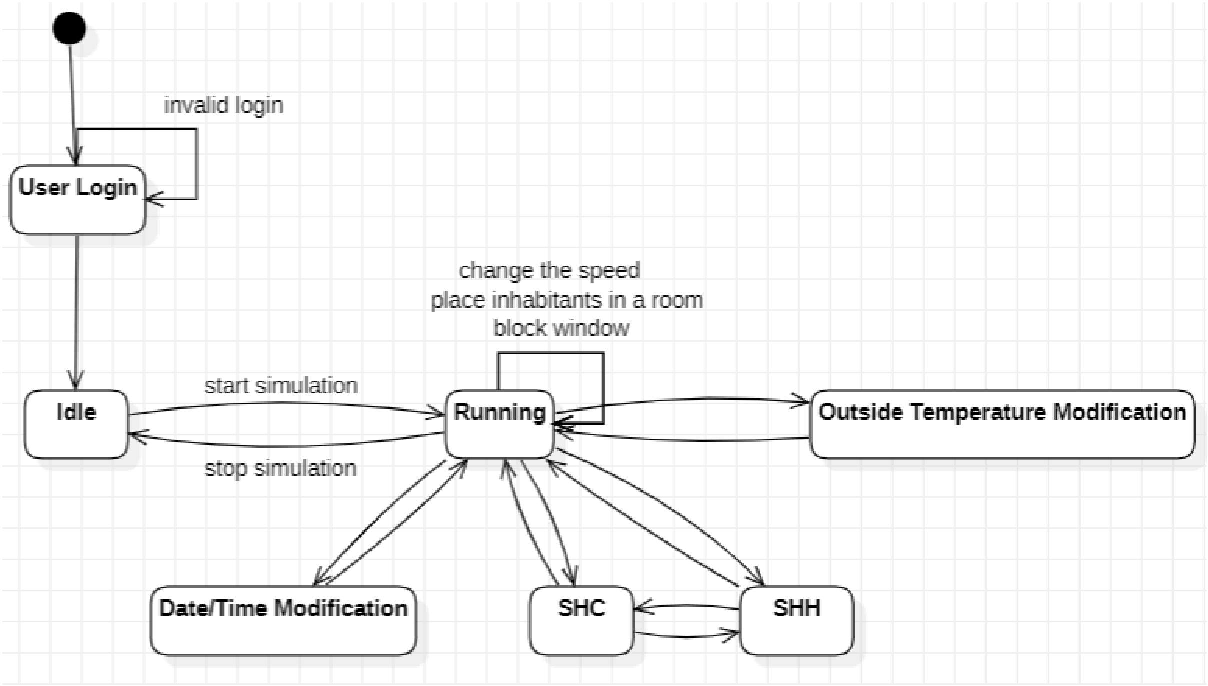
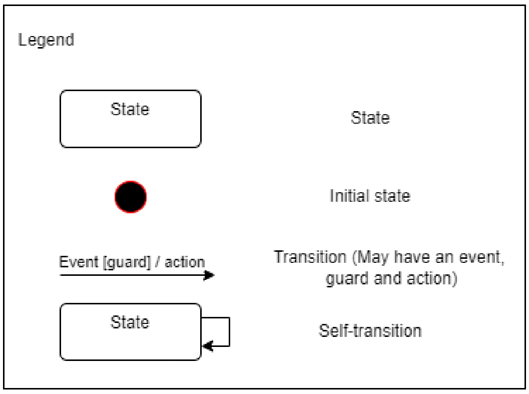


Figure 6: State Machine Diagram - Context of Simulation



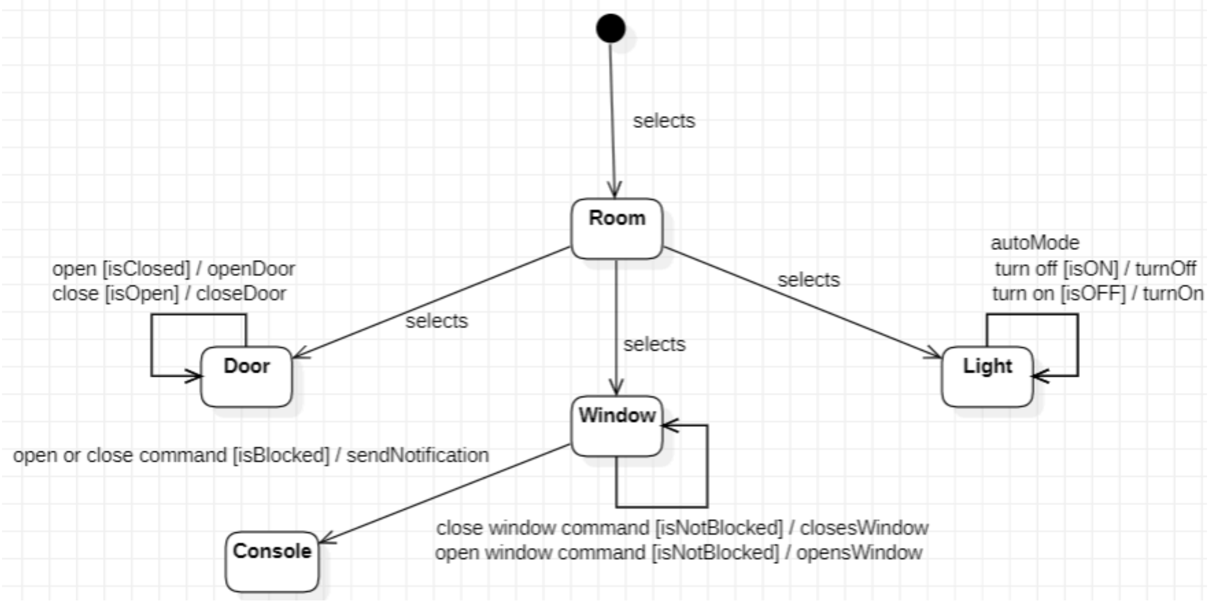

selects
Room
open [isClosed] / openDoor
close [isOpen] / closeDoor
selects
selects
selects
autoMode
turn off [isON] / turnOff
turn on [isOFF] / turnOn
Door
Window
Light
open or close command [isBlocked] / sendNotification
Console
close window command [isNotBlocked] / closesWindow
open window command [isNotBlocked] / opensWindow

Figure 7: State Machine Diagram - SCH

2.5 Activity Diagram for the Context of the Simulation

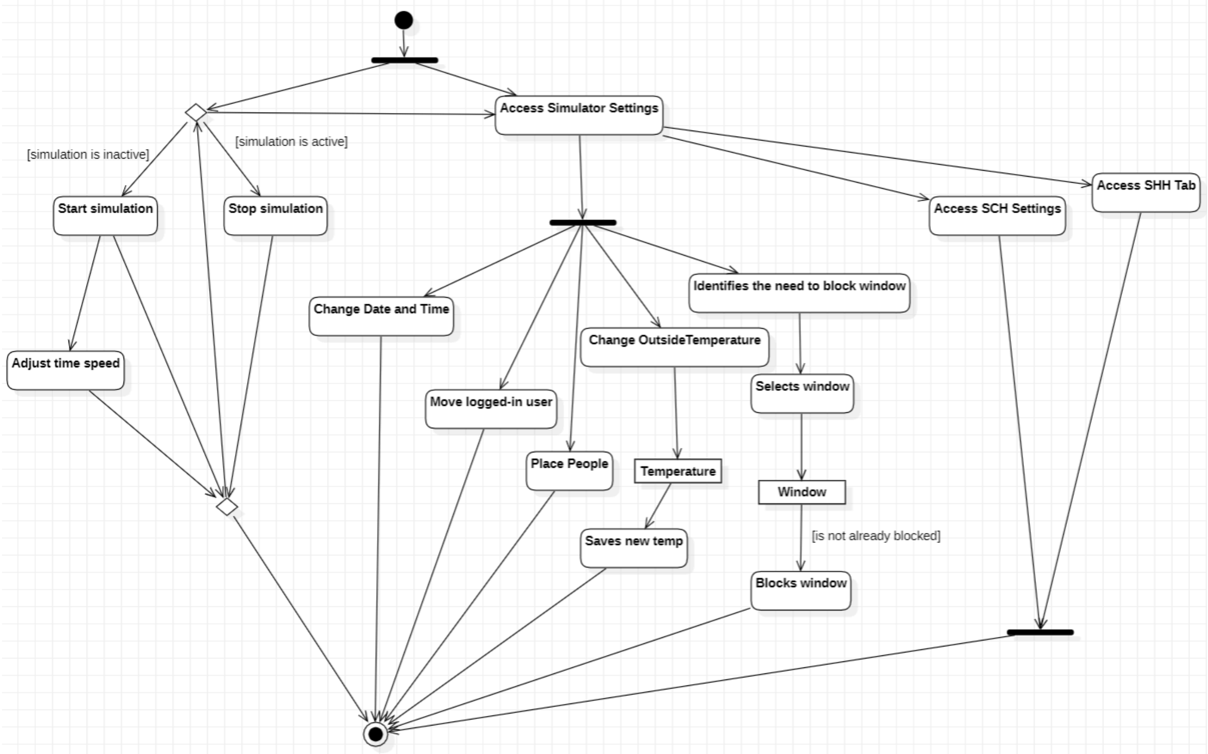


Figure 8: Activity Diagram for Context of the Simulation

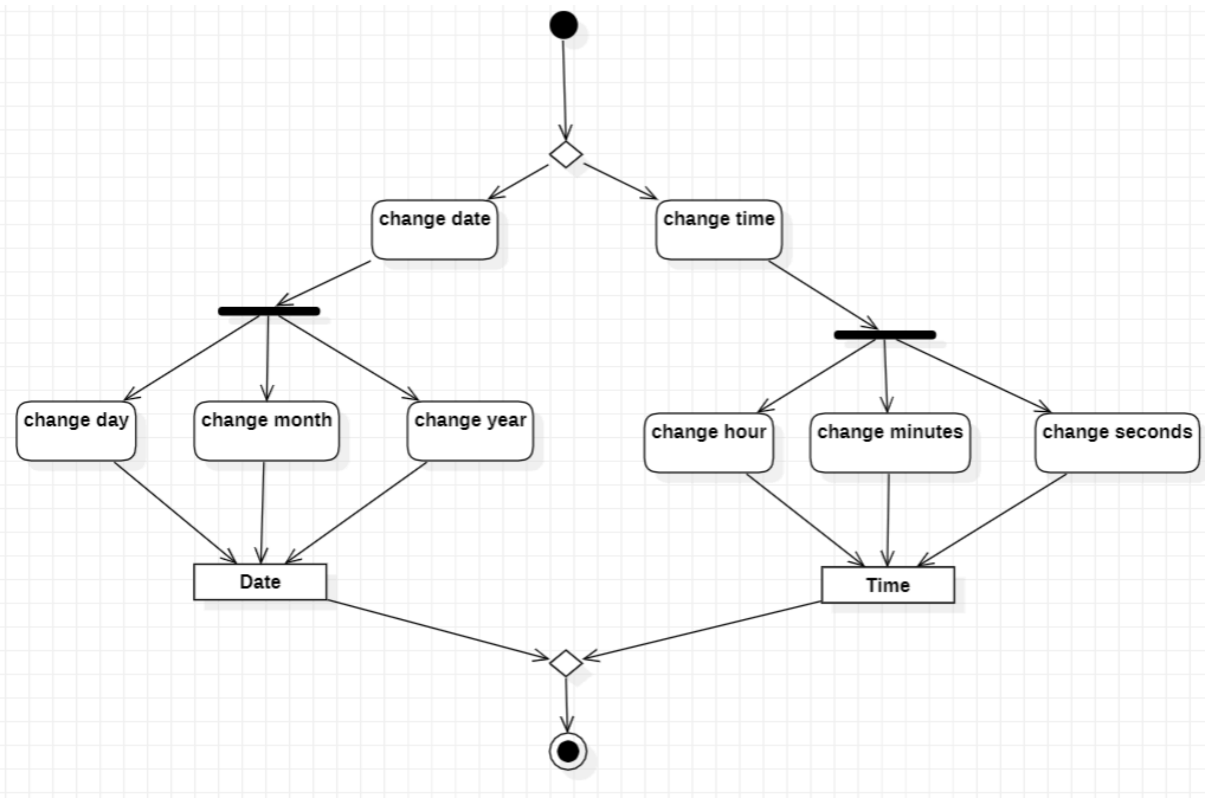
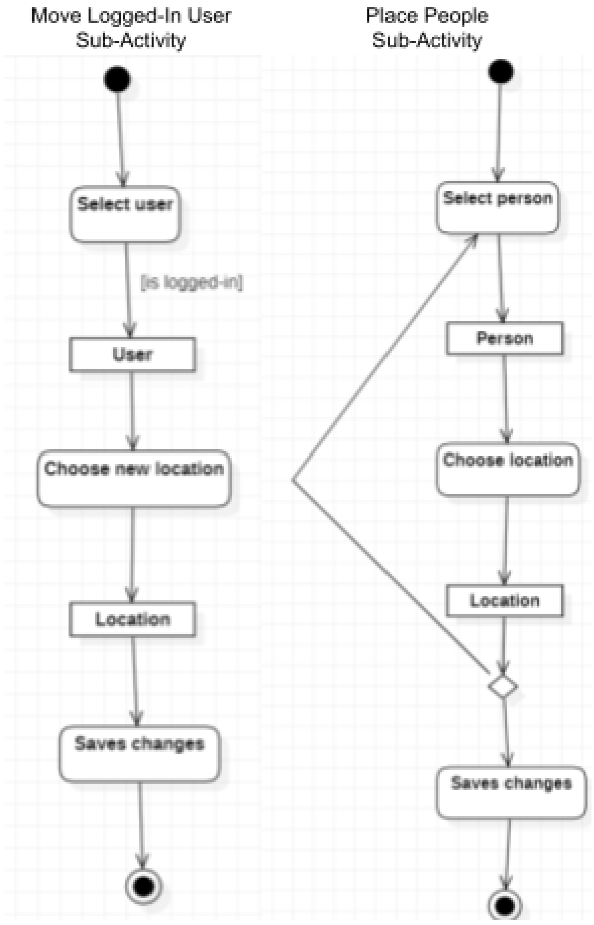


Figure 9: Activity Diagram for the Sub-Activity Change Date and Time


Move Logged-In User Sub-Activity

Place People Sub-Activity

Figure 10: Move and Place People Sub-Activity

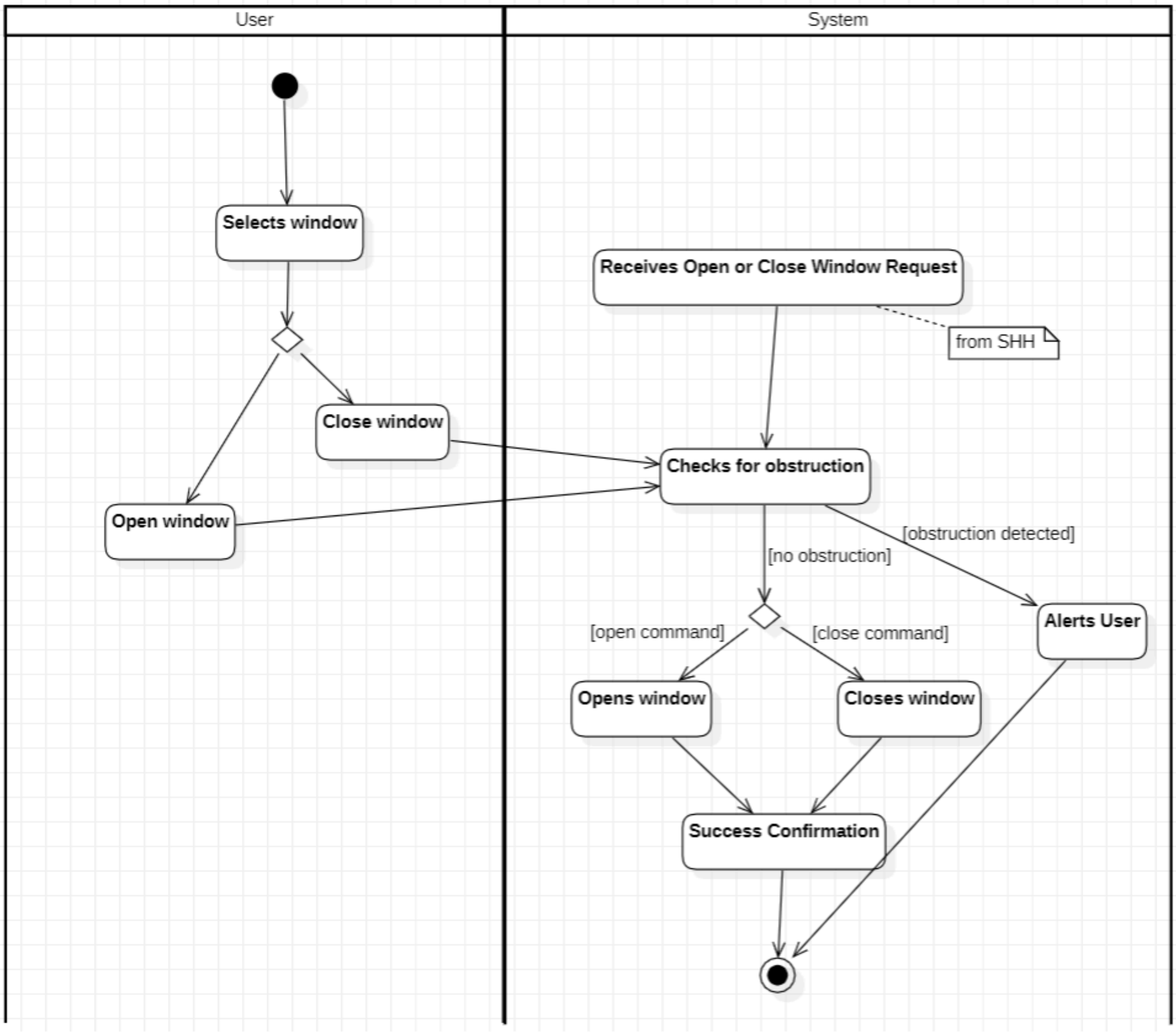
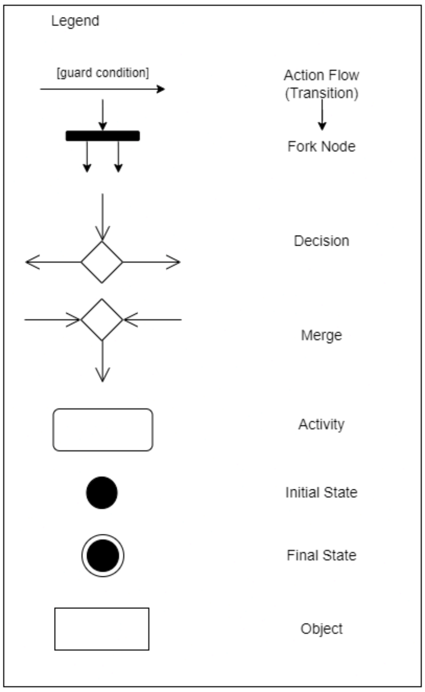


Figure 11: Window Obstruction SHC Module Sub-Activity



2.6 State Machine Diagrams of the SHH Module

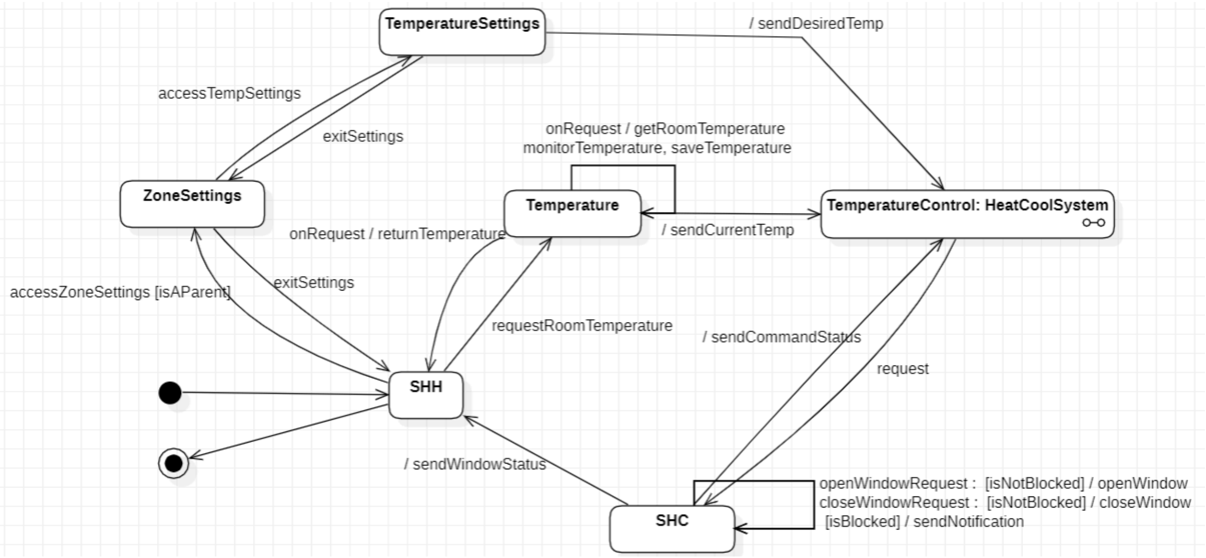

TemperatureSettings
/ sendDesiredTemp
accessTempSettings
exitSettings
onRequest / getRoomTemperature
monitorTemperature, saveTemperature
ZoneSettings
Temperature
TemperatureControl: HeatCoolSystem
onRequest / returnTemperature
/ sendCurrentTemp
accessZoneSettings [isAParent]
exitSettings
requestRoomTemperature
/ sendCommandStatus
request
SHH
/ sendWindowStatus
openWindowRequest : [isNotBlocked] / openWindow
closeWindowRequest : [isNotBlocked] / closeWindow
[isBlocked] / sendNotification
SHC

Figure 12: Smart Home Heating Module State Machine

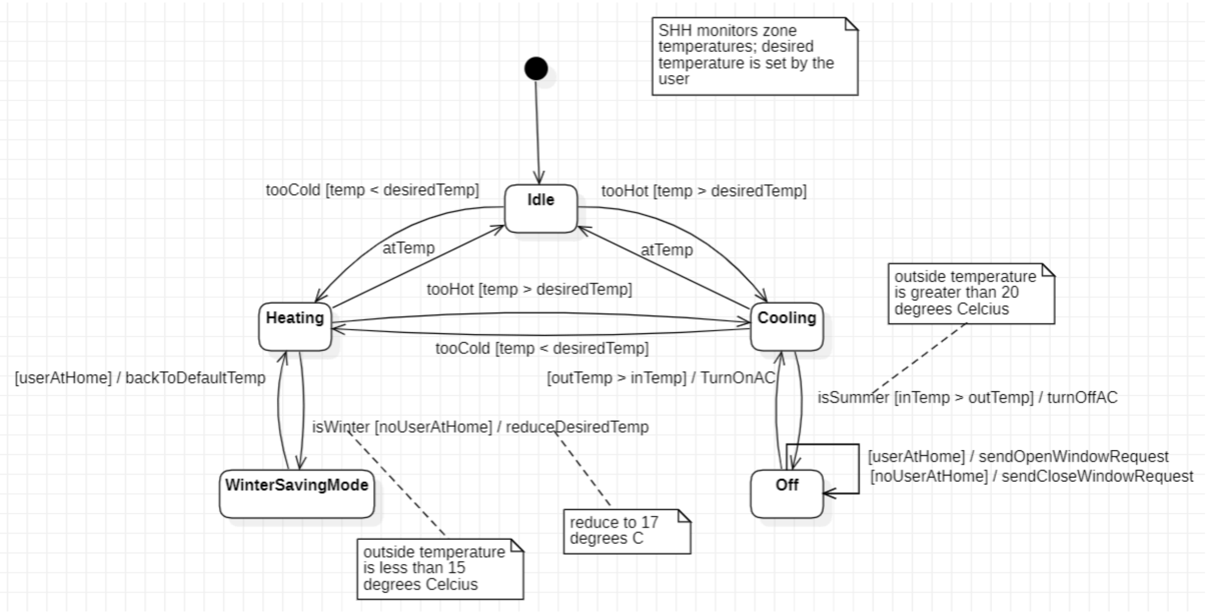
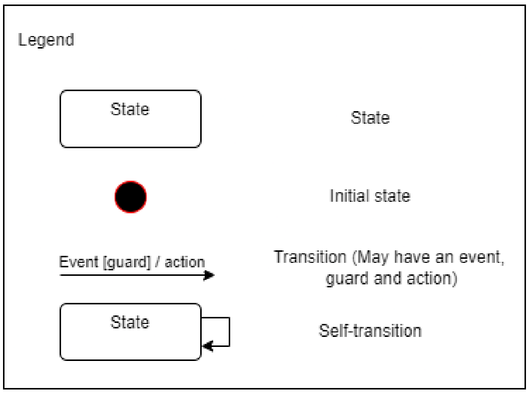

SHH monitors zone temperatures; desired temperature is set by the user
tooCold [temp < desiredTemp]
Idle
tooHot [temp > desiredTemp]
atTemp
atTemp
tooHot [temp > desiredTemp]
outside temperature is greater than 20 degrees Celcius
Heating
Cooling
tooCold [temp < desiredTemp]
[userAtHome] / backToDefaultTemp
[outTemp > inTemp] / TurnOnAC
isSummer [inTemp > outTemp] / turnOffAC
isWinter [noUserAtHome] / reduceDesiredTemp
[userAtHome] / sendOpenWindowRequest
[noUserAtHome] / sendCloseWindowRequest
WinterSavingMode
Off
reduce to 17 degrees C
outside temperature is less than 15 degrees Celcius

Figure 13: SHH Module State Machine Temperature Control Sub-Machine



2.7 Activity Diagram for Smart Home Heating Module

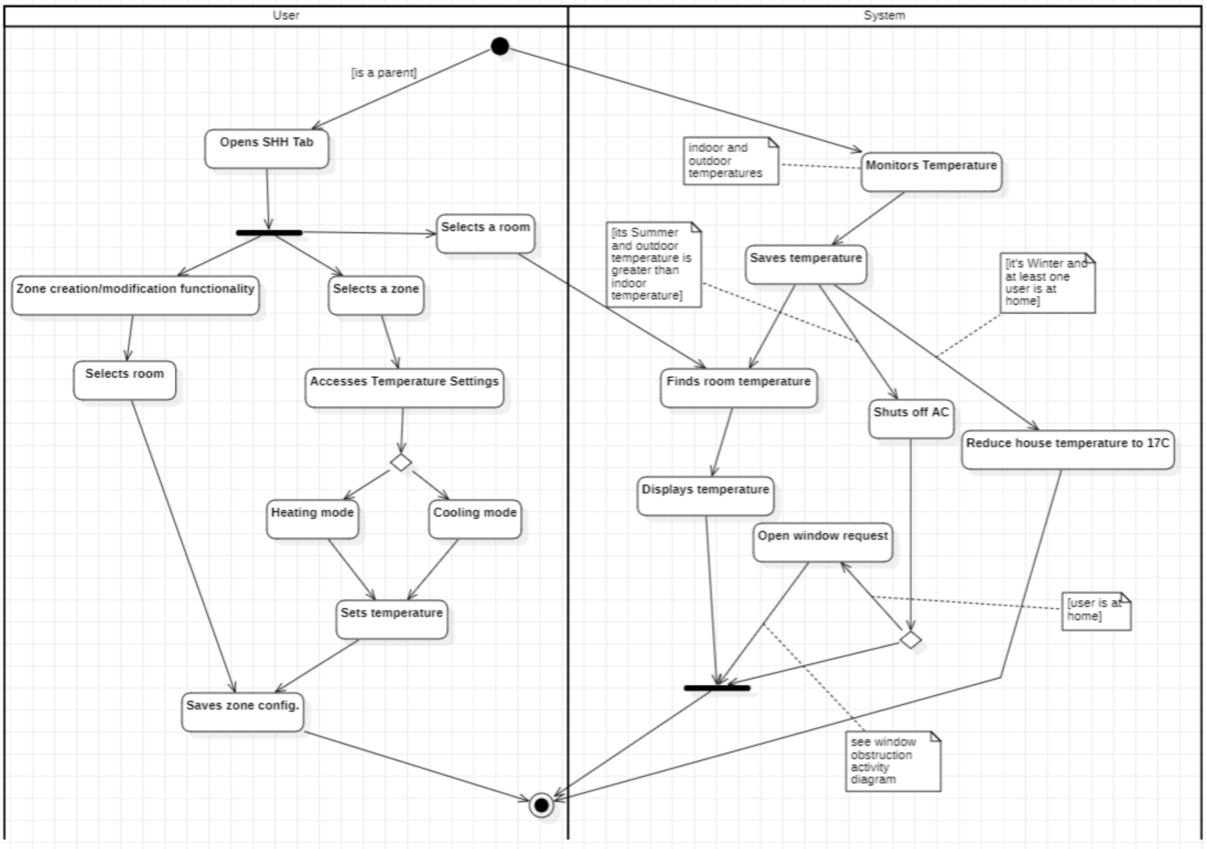
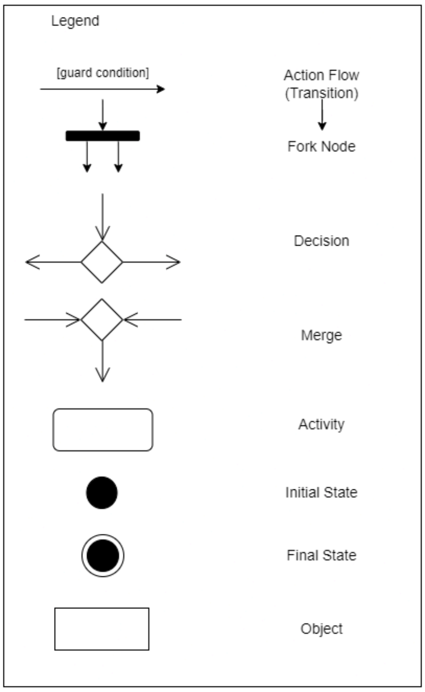

User
System
[is a parent]
Opens SHH Tab
Selects a room
Zone creation/modification functionality
Selects a zone
Selects room
Accesses Temperature Settings
Heating mode
Cooling mode
Sets temperature
Saves zone config.
indoor and outdoor temperatures
Monitors Temperature
[its Summer and outdoor temperature is greater than indoor temperature]
Saves temperature
[it's Winter and at least one user is at home]
Finds room temperature
Shuts off AC
Reduce house temperature to 17C
Displays temperature
Open window request
[user is at home]
see window obstruction activity diagram

Figure 14: Activity Diagram - SHH



4. Design Patterns

In Phase 3 we made some adjusted to our design patterns. Following the instructions, we modified our observer pattern to be more targeted to the SHH system. Secondly, since we were going to use a different pattern for the SHH and it did not make sense to use both for the SHH, we chose to update our third pattern. We are now implementing the Singleton pattern as our third pattern. All three of the patterns below have been implemented in our code.

4.1 Command Pattern

Adopting the command pattern in the smart home simulator project facilitates the encapsulation of requests as objects, allowing for parameterization of actions and decoupling of the requester from the executor. This pattern proves advantageous in scenarios where users or other modules issue commands to control various devices and functionalities within the smart home environment. Each command encapsulates a specific action along with its parameters, enabling easy queuing, logging, and undo/redo functionalities. For instance, commands such as "open window," "turn on lights," "open door" or “turn on auto mode” can be encapsulated as command objects, which are then executed by the Smart Home Core Functionality (SHC) module or other relevant modules.

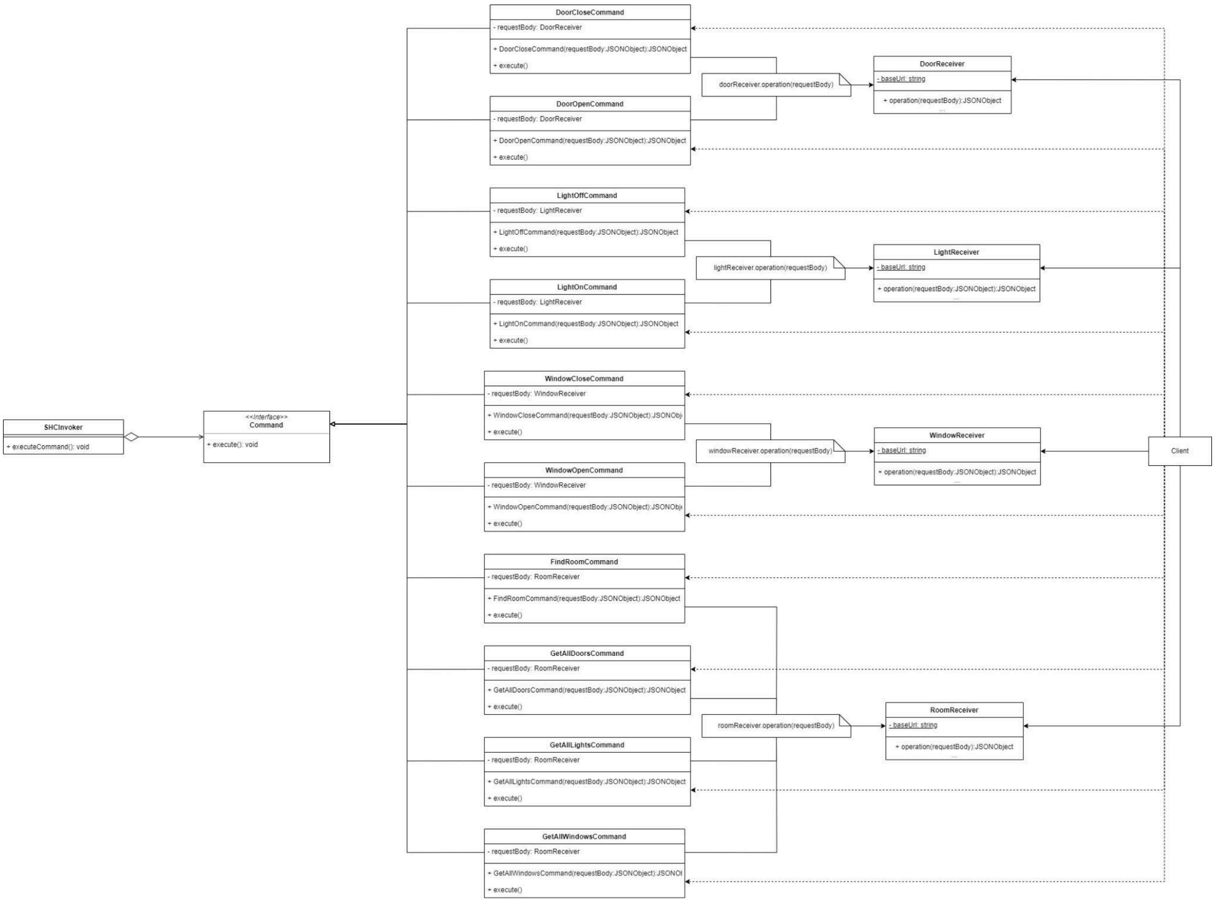

DoorCloseCommand
- requestBody: DoorReceiver
+ DoorCloseCommand(requestBody:JSONObject):JSONObject
+ execute()
doorReceiver.operation(requestBody)
DoorReceiver
- baseUrl: string
+ operation(requestBody):JSONObject
DoorOpenCommand
- requestBody: DoorReceiver
+ DoorOpenCommand(requestBody:JSONObject):JSONObject
+ execute()
LightOffCommand
- requestBody: LightReceiver
+ LightOffCommand(requestBody:JSONObject):JSONObject
+ execute()
lightReceiver.operation(requestBody)
LightReceiver
- baseUrl: string
+ operation(requestBody:JSONObject):JSONObject
LightOnCommand
- requestBody: LightReceiver
+ LightOnCommand(requestBody:JSONObject):JSONObject
+ execute()
WindowCloseCommand
- requestBody: WindowReceiver
+ WindowCloseCommand(requestBody:JSONObject):JSONObj
+ execute()
SHCInvoker
+ executeCommand(): void
<<Interface>>
Command
+ execute(): void
windowReceiver.operation(requestBody)
WindowReceiver
- baseUrl: string
+ operation(requestBody:JSONObject):JSONObject
Client
WindowOpenCommand
- requestBody: WindowReceiver
+ WindowOpenCommand(requestBody:JSONObject):JSONObj
+ execute()
FindRoomCommand
- requestBody: RoomReceiver
+ FindRoomCommand(requestBody:JSONObject):JSONObject
+ execute()
GetAllDoorsCommand
- requestBody: RoomReceiver
+ GetAllDoorsCommand(requestBody:JSONObject):JSONObject
+ execute()
roomReceiver.operation(requestBody)
RoomReceiver
- baseUrl: string
+ operation(requestBody):JSONObject
GetAllLightsCommand
- requestBody: RoomReceiver
+ GetAllLightsCommand(requestBody:JSONObject):JSONObject
+ execute()
GetAllWindowsCommand
- requestBody: RoomReceiver
+ GetAllWindowsCommand(requestBody:JSONObject):JSONOI
+ execute()

Figure 15: Command Pattern

4.2 Observer Pattern

Implementing the observer pattern for the Smart Heating (SHH) module within the smart home simulator is a good design decision for several reasons. Firstly, the observer pattern facilitates a decoupled and modular architecture, enhancing the maintainability and scalability of the system. By employing this pattern, the SHH module can seamlessly integrate with the simulator without being tightly coupled to its implementation details. Secondly, the SHH needs to constantly monitor various environmental factors like indoor and outdoor temperatures, occupancy status, and system events, the observer pattern allows it to efficiently subscribe to relevant updates from the simulator. Whenever there's a change in temperature or occupancy status, the SHH is promptly notified, enabling it to make real-time adjustments to heating and cooling systems accordingly. Lastly, the observer pattern promotes flexibility since we can easily accommodate multiple observers if needed for other modules.

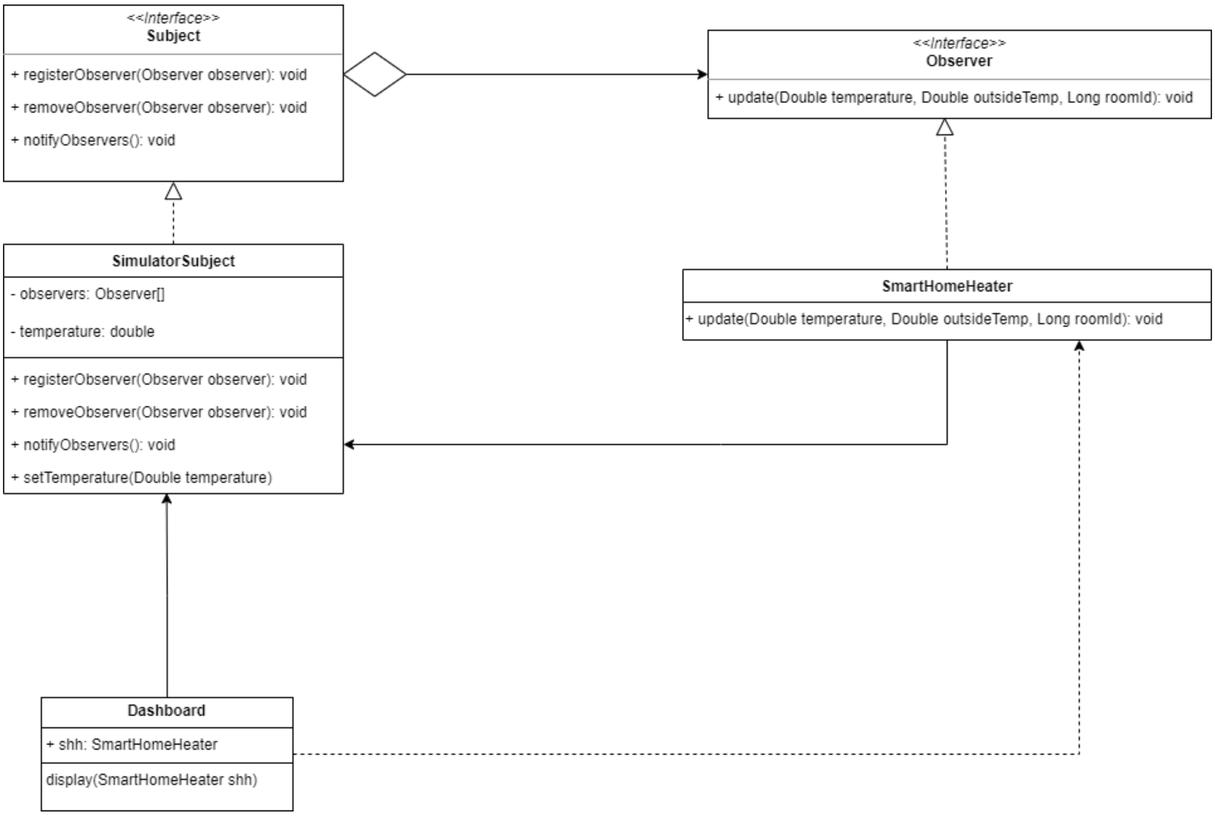

<<Interface>>
Subject
+ registerObserver(Observer observer): void
+ removeObserver(Observer observer): void
+ notifyObservers(): void
<<Interface>>
Observer
+ update(Double temperature, Double outsideTemp, Long roomId): void
SimulatorSubject
- observers: Observer[]
- temperature: double
+ registerObserver(Observer observer): void
+ removeObserver(Observer observer): void
+ notifyObservers(): void
+ setTemperature(Double temperature)
SmartHomeHeater
+ update(Double temperature, Double outsideTemp, Long roomId): void
Dashboard
+ shh: SmartHomeHeater
display(SmartHomeHeater shh)

Figure 16: Observer Pattern

4.3 Singleton Pattern

We decided to use the Singleton design pattern for the outdoor temperature since it encapsulates the overall temperature outside. It does not make logical sense to have more than one temperature for outside of the home, so using the Singleton pattern ensures that there are no errors when getting instances of OutdoorTemperature.

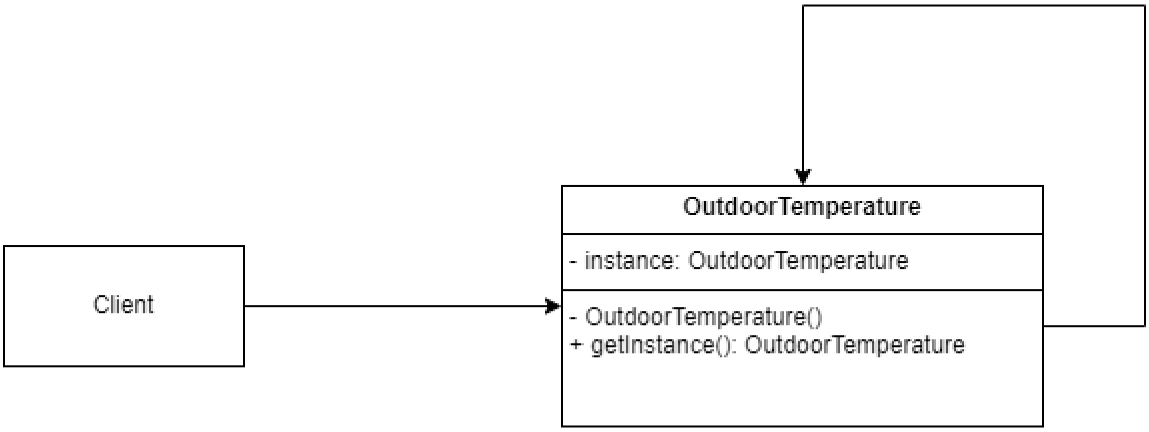
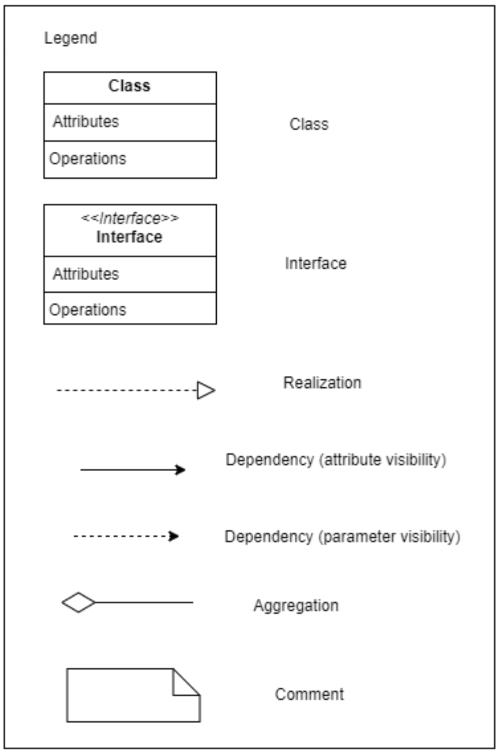

Client
OutdoorTemperature
- instance: OutdoorTemperature
- OutdoorTemperature()
+ getInstance(): OutdoorTemperature

Figure 17: Singleton Pattern

Below is the legend for the Design Patterns.



References

[1] https://www.conceptdraw.com/How-To-Guide/picture/Design-elements-UML-class-diagrams.png
[2] https://cs.uwlax.edu/~mzheng/CS743Fall19/UseCaseDiagrams.html
[3] https://conceptdraw.com/a3156c3/preview