Smart Home Simulator

SOEN 343

Lab Section WJ-X

March 13th 2024

Clara Gagnon (40208598)(Team Leader) Role: Smart Home Core Implementation, System Architecture and Design Patterns
Ahmad Elmahallawy (40193418) Role: Smart Home Dashboard Implementation, System Architecture, Simulation Parameters and Design Patterns
Gulnoor Kaur (40114998) Role: System Architecture and Context of the Simulation
Liam Daigle (40207583) Role: House Layout, Backend Implementation, Use Case Diagram, State Machine Diagram and Smart Home Core Implementation
Vanessa DiPietrantonio (40189938) Role: Context of the Simulation Implementation, Use Case Tables, Use Case Diagram, Sequence Diagram, Design Patterns
Jessica Beauchemin (40188873) Role: Use Case Tables, Activity Diagram

Table of Content

1. System Architecture

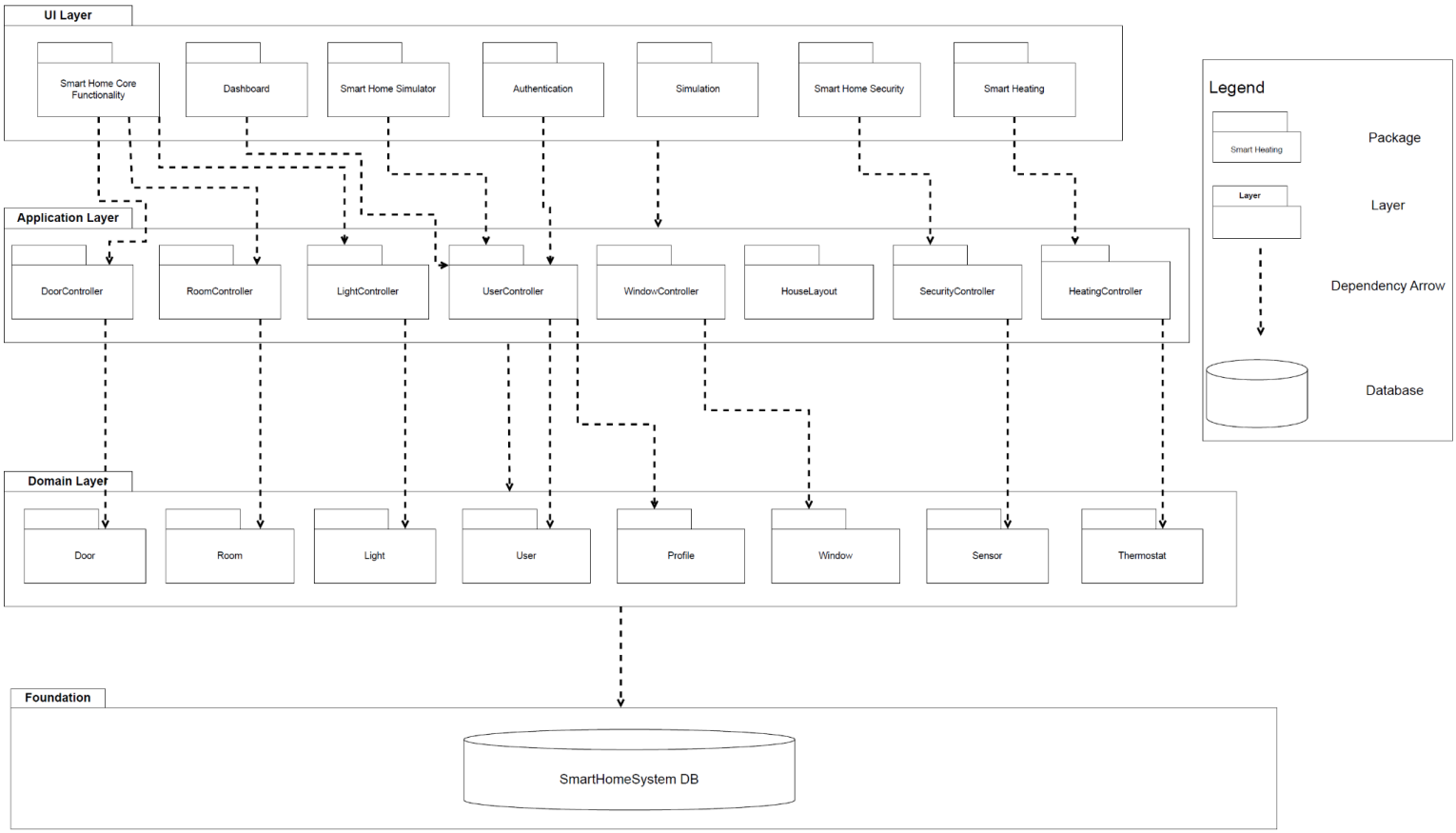


Figure 1: System architecture Diagram

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.

2. Use Case Model

2.1 Use Case Tables

2.1.1 House Layout

ID:	**UC-1.1**
Title:	Read and Load House-Layout File
Description:	The goal of the use case is for the simulator to read and load a house-layout file, ensuring accurate representation of rooms and their attributes.
Primary Actor:	User, System
Preconditions:	• User has the simulator application installed and running. • Current house-layout file exists and is accessible. • Current house-layout file is compatible with the house-layout input in the simulator system.
Postconditions:	• The house-layout file is successfully loaded into the simulator. • The layout is displayed on the dashboard.
Inputs:	• House-layout file

Outputs:	• Confirmation message indication successful loading of the house layout • Visual representation of the loaded house layout.
Main Success Scenario:	1. The user opens the simulator. 2. The user clicks the “Load House Layout” button to load a new house layout file. 3. The system prompts the user to select the desired house layout file from their directory on their device. 4. The user selects the appropriate file. 5. The system validates the file format and content. 6. The system reads and loads the layout data from the file and extracts relevant information. 7. A confirmation message is displayed, indicating successful loading of the house layout.

2.1.2 Simulation Parameters

ID:	**UC-2.1**
Title:	Configure Simulation Start Parameters (Date and Time, Weather Conditions)
Description:	The goal of the use case is to enable the user to define the environmental parameters necessary for starting the simulation, including the date and time, as well as the weather conditions inside and outside the property.
Primary Actor:	User
Preconditions:	• The SHS system is initialized and accessible. • User is authenticated in the system.
Postconditions:	• Simulation’s environmental parameters are set

	• The system is ready for the initiation of the simulation under the set conditions.
Inputs:	• Date and time for simulation start • Weather conditions inside and outside the property
Outputs:	• Confirmation message indicating successful setting of simulation parameters • Display of environmental conditions on the simulator's dashboard
Main Success Scenario:	1. The user logs into the sim. 2. The user selects the option to define simulation start parameters. 3. The system prompts the user to input the date and time for simulation start. 4. The user enters the desired date and time. 5. The system prompts the user to input weather conditions inside and outside the property. 6. The user enters the relevant weather information. 7. The system validates the input parameters. 8. The system confirms the setting of simulation parameters. 9. A confirmation message is displayed, indicating successful setting of simulation parameters. 10. The system is now ready to initiate the simulation using the defined parameters.

ID:	**UC-2.2**
Title:	Add/Remove/Edit User Profiles
Description:	The goal of this use case is to enable the user to add, remove, or edit user profiles within the smart home simulator system, which are used for granting or denying accessibility to actions performed by other system

	modules. The profiles represent the home's virtual people, such as family (adult and children), guests, and strangers.
Primary Actor:	User
Preconditions:	• The SHS system is initialized and accessible. • The user is authenticated in the system.
Postconditions:	• The user profile is added, removed or edited as per the administrator's actions.
Inputs:	• User profile details, for adding and editing • User profile identifier, for removing
Outputs:	• Confirmation message indicating successful addition, removal, or editing of user profiles. • Updated registry for user profiles in the system
Main Success Scenario:	1. The SHS system administrator accesses the system interface 2. The administrator selects the tab to manage user profiles on the simulator dashboard. 3. The system presents options to add, remove, or edit user profiles. 4. For adding a user profile: a. The administrator inputs the user's unique details and saves the profile. b. The system validates the input and adds the user profile. 5. For removing a user profile: a. The administrator selects the user profile to be removed and confirms the deletion. b. The system confirms the removal of the selected user profile. 6. For editing a user profile: a. The administrator selects an existing user profile in the system to be edited.

	b. The administrator modifies the profile details and saves changes. c. The system validates the changes and updates the user profile. 7. The system displays a confirmation message, indicating the completion of the requested action. 8. User profiles are updated in the system according to the administrator's actions.

ID:	**UC-2.3**
Title:	Log in using Existing User Profile and Set House Location
Description:	The goal of this use case is to enable the user to log into the simulator using an existing user profile and set the location of the virtual smart home within the simulator.
Primary Actor:	User
Preconditions:	● The simulator is initialized and accessible. ● The user profile is defined within the system.
Postconditions:	● The user is successfully logged into the simulator using the selected user profile ● The house location is set within the simulator.
Inputs:	● User credentials (username and password) ● House location information (e.g. kitchen, living room, etc.)
Outputs:	● Confirmation message indicating successful login and house location setting. ● Display of the user's house location on simulator dashboard

Main Success Scenario:	1. The Simulator's User accesses the simulator interface. 2. The user selects the option to log in using an existing user profile. 3. The system prompts the user to input their credentials (username and password). 4. The user provides the required authentication information. 5. The system verifies the provided information against the existing user profiles. 6. The system logs the user into the simulator using the selected user profile. 7. The user is prompted to set the house location within the simulator. 8. The user enters the house location information (e.g., kitchen, living room, etc.). 9. The system saves the house location setting. 10. Confirmation message is displayed, indicating the completion of the login process and house location setting. 11. The user gains access to the simulator and interacts with smart home modules based on the associated permissions of their profile.

ID:	**UC-2.4**
Title:	Grant/Deny Permissions for Executing Commands on Smart Home Modules
Description:	The goal of this use case is to enable the user to log into the simulator using an existing user profile and set the location of the virtual smart home within the simulator.
Primary Actor:	System, User
Preconditions:	● The user is logged into the simulator with appropriate permissions. ● Smart home modules are registered within the system

Postconditions:	• Permissions for executing commands on registered smart home modules are successfully granted or denied by the user.
Inputs:	• Command to be executed • Target smart home module • Permission status (grant or deny).
Outputs:	• Confirmation message indicating permission granted or denied to the user.
Main Success Scenario:	1. The Simulator's User accesses the simulator interface. 2. The user navigates to the registered smart home module for which they want to use. 3. The user selects the command to be executed within the smart home module. 4. The system checks if permission is granted based on the user's profile. 5. The system displays a message granting or denying permissions for executing the command on the selected module based on their user profile.

ID:	**UC-2.5**
Title:	Save User Profiles and Permissions in File
Description:	The goal of this use case is to save user profiles and permissions in a file format within the simulator so that users can avoid the need for reentering this essential information for controlling access and interactions within the simulator and smart home modules each time they reopen the simulator.
Primary Actor:	System
Preconditions:	• User profiles with corresponding permissions are defined within the simulator.

Postconditions:	• Updated file containing user profiles and their associated permissions is saved.
Inputs:	None
Outputs:	None
Main Success Scenario:	1. The Simulator's User accesses the simulator interface. 2. The user selects the option to save the user profile (during adding or editing user profile). 3. The system retrieves the information of permissions that correspond with the user profile. 4. The system saves the user profile and corresponding permissions in a file format. 5. The system confirms the successful saving of the user profile. 6. The user can now reopen the simulator without the need for reentering user profile information and permissions.

2.1.3 Context of the Simulation

ID:	**UC-3.1**
Title:	Change Time Speed of Simulator
Description:	The system allows the simulator’s user to change the time speed.
Primary Actor:	System, User
Preconditions:	• A simulation is currently active. • The time speed element exists and is open for modification.

Postconditions:	• The time speed of the simulation context is successfully adjusted according to the user's selection. • The simulation runs at the newly set time speed until the user makes new adjustments. • The system displays the current time speed.
Inputs:	• User selection of simulator time speed.
Outputs:	• Adjusted time speed. • Simulation running at new time speed.
Main Success Scenario:	1. The user accesses the simulation settings. 2. The system provides and displays the option to adjust the time speed of the simulator. 3. The user selects a time speed. 4. The system adjusts the time speed of the simulation according to the user's selection. 5. The simulation continues to run at the set times speed until the user makes new adjustments.

ID:	**UC-3.2**
Title:	Simulation ON/OFF Toggle
Description:	The system allows the simulator's user to turn ON and OFF the simulation.
Primary Actor:	System, User
Preconditions:	The smart home simulator is running. If the simulation is active; the ON button is deactivated.

	If the simulation is inactive; the OFF button is deactivated.
Postconditions:	The simulation is started or stopped depending on the user's selection. The system displays the current simulation state. If the simulation's ON button was selected; deactivate the ON button. If the simulation's OFF button was selected; deactivate the OFF button.
Inputs:	User command to start or stop the simulation.
Outputs:	Simulation state is set to start or stop.
Main Success Scenario:	1. The user accesses the simulation settings. 2. The system provides and displays the option to start or stop the simulation. 3. The user selects start or stop simulation using the on-toggle button. 4. The system initiates the simulation if the user selects start and the system stops the simulation if the user selects stop.

ID:	**UC-3.3**
Title:	Change of Date and Time
Description:	The system allows the simulator's user to modify the date and time of the simulation.
Primary Actor:	System, User
Preconditions:	The date element exists and is open for modification.

	The time element exists and is open for modification.
Postconditions:	The simulator's date and/or time is updated depending on the user's selection. The system confirms the modification.
Inputs:	User command to modify the date and/or time.
Outputs:	Simulation's date and/or time attribute is modified and system confirms successful modification.
Main Success Scenario:	1. The user accesses the simulation settings. 2. The system provides and displays the option to change the date and time of the simulation. 3. The user adjusts the date and/or time. 4. The system adjusts the date and time of the simulation according to the user's selection.

ID:	**UC-3.4**
Title:	Move Logged User to Different Room
Description:	The system allows a logged-in user to be moved to a different room within the smart home simulator house layout.
Primary Actor:	System, User
Preconditions:	The user exists. The user is logged into the system. The destination location exists within the simulation environment.

Postconditions:	The logged-in user is successfully moved to a different room. The position of the logged-in user is updated with the new location. The system confirms the modification.
Inputs:	User's selection of destination room.
Outputs:	Logged-in user's location is successfully updated and system confirms successful modification.
Main Success Scenario:	1. The user accesses the simulation settings. 2. The system provides and displays the option to change the position of a logged-in user within the home layout. 3. The user selects the new location of the logged-in user. 4. The system updates and confirms the location of the logged-in user according to the user's selection.

ID:	**UC-3.5**
Title:	Place People in Specific Rooms or Outside Home
Description:	The system allows the user to place people in specific rooms or outside home.
Primary Actor:	System, User
Preconditions:	The person element exists. The destination location exists within the simulation environment.

Postconditions:	The selected people are successfully placed in the specified locations within the simulation environment. The new position of the people is updated. The system confirms the modification.
Inputs:	User selects the person or people to move and specifies their new location.
Outputs:	Updates and confirms the placement of the individuals in the specified location.
Main Success Scenario:	1. The user accesses the simulation settings. 2. The system provides and displays the option to change the position of a person within the home layout. 3. The user selects the new location of the individuals. 4. The system updates and confirms the location of the logged-in user according to the user's selection.

ID:	**UC-3.6**
Title:	Modify Outside Temperature
Description:	The system allows for the modification of the temperature outside of the home.
Primary Actor:	System, User
Preconditions:	The outside temperature element exists and is open for modification. The outside temperature is within the -90˚ to 60˚ Celsius range.

Postconditions:	The outside temperature is successfully modified as per the simulator user's selection. The system confirms the modification.
Inputs:	User command to modify the outside home temperature.
Outputs:	Outside temperature is updated and user receives notification from system of successful modification.
Main Success Scenario:	1. The user accesses the simulation settings. 2. The system provides and displays the option to change the outside temperature. 3. The user adjusts the outside temperature. 4. The system adjusts the outside temperature according to the user's selection.

ID:	**UC-3.7**
Title:	Block Window Movement by Putting an Arbitrary Object
Description:	The system allows the user to block the movement of windows by placing an arbitrary object in their path.
Primary Actor:	System, User
Preconditions:	The window exists and is not already blocked.
Postconditions:	Successful placement of arbitrary object. Window is updated. The system confirms the modification.

Inputs:	User command to block windows movement.
Outputs:	Window element is updated as being blocked and the system sends confirmation of successful blocking of windows movement.
Main Success Scenario:	1. The user identifies the need to block the movement of windows. 2. The user accesses the simulation settings. 3. The user selects the window that requires blocking. 4. The user selects the option to block window movement. 5. The system confirms to the user that the windows' movement has been successfully blocked.

2.1.4 Smart Home Core Functionality (SHC) Functionality

ID:	**UC-4.1**
Title:	Open/Close Doors
Description:	The goal of this use case is to enable users or smart home modules to open or close doors within the simulated environment.
Primary Actor:	User, Smart Home Module
Preconditions:	• The user is logged into the smart home system • The smart home module is active with access to door control functionalities.
Postconditions:	• The specified door is successfully opened or closed according to the user's command.
Inputs:	• Command (open or close) • Door identifier (e.g., door name or ID)

Outputs:	• Updated door command and status is displayed on the Output Console as confirmation of successful execution of the command.
Main Success Scenario:	1. User accesses the smart home module. 2. User selects the option to control doors. 3. User specifies the desired action (open or close). 4. User selects the door to be operated . 5. System verifies the user's permission and the command for the door to be operated. 6. System executes the command to open or close the specified door. 7. The system updates the door status. 8. The system displays the command and updated status of the door on the Output Console as confirmation of successful execution of the command.

ID:	**UC-4.2**
Title:	Open/Close Windows
Description:	The goal of this use case is to facilitate users or smart home modules in opening or closing windows within the simulated environment.
Primary Actor:	User, Smart Home Module
Preconditions:	• The user is logged into the smart home system • The smart home module is active with access to window control functionalities.
Postconditions:	• The specified window is successfully opened or closed according to the user's command.

Inputs:	• Command (turn on or off). • Window identifier (e.g., window name or ID).
Outputs:	• Updated window command and status is displayed on the Output Console as confirmation of successful execution of the command.
Main Success Scenario:	1. User accesses the smart home module. 2. User selects the option to control windows. 3. User specifies the desired action (open or close). 4. User selects the window to be operated . 5. System verifies the user's permission and the command for the window to be operated. 6. System executes the command to open or close the specified window. 7. The system updates the window status. 8. The system displays the command and updated status of the window on the Output Console as confirmation of successful execution of the command.

ID:	**UC-4.3**
Title:	Turn On/Off Lights
Description:	The goal of this use case is to allow users or smart home modules to turn on or off lights within the simulated environment.
Primary Actor:	User, Smart Home Module
Preconditions:	• The user is logged into the smart home system • The smart home module is active with access to light control functionalities.

Postconditions:	• The specified light fixture is successfully turned on or off according to the user's command.
Inputs:	• Command (turn on or off) • Light fixture identifier (e.g., light name or ID)
Outputs:	• Updated light command and status is displayed on the Output Console as confirmation of successful execution of the command.
Main Success Scenario:	1. User accesses the smart home module. 2. User selects the option to control lights. 3. User specifies the desired action (turned on or off). 4. User selects the light fixture to be operated. 5. System verifies the user's permission and the command for the light fixture to be operated. 6. System executes the command to turn on or off the specified light. 7. The system updates the light fixture status. 8. The system displays the command and updated status of the light fixture on the Output Console as confirmation of successful execution of the command.

ID:	**UC-4.4**
Title:	Turn On/Off Lights Automatically When Someone Enters A Room
Description:	The goal of this use case is to allow users to set an Auto mode for lights, which uses sensors to automatically turn lights on or off when a user enters or exits a room within the simulated environment.
Primary Actor:	User
Preconditions:	• The user is logged into the smart home system

	• The smart home module is active with access to window control functionalities.
Postconditions:	• Updated light status is displayed on the Output Console as confirmation of successful execution of the command.
Inputs:	• Command (activate or deactivate Auto mode)
Outputs:	• Confirmation message indicating the successful execution of the command.
Main Success Scenario:	1. User accesses the smart home system interface. 2. User selects the option to set Auto mode for lights. 3. User specifies the desired action (activate or deactivate). 4. System verifies the command 5. System sensors control activation/deactivation of lights in Auto mode. 6. System executes the command to activate or deactivate Auto mode for lights. 7. System logs and updates the Auto mode status accordingly. 8. System displays a confirmation message indicating the successful execution of the command.

2.1.5 Smart Home Dashboard

ID:	**UC-5.1**
Title:	Update House View for Current Simulation State
Description:	The system automatically updates the house view to graphically represent the current state of the simulation when the simulation is activated.

Primary Actor:	System, User
Preconditions:	● The house layout file has been read and displayed successfully. ● The user has activated the simulation using the on-toggle button.
Postconditions:	● The house view is continuously updated to accurately represent the current state of the simulation.
Inputs:	● Simulation activation state.
Outputs:	● Updated house view reflecting the current state of the simulation.
Main Success Scenario:	1. The user activates the simulation by toggling the on-toggle button. 2. The system starts the simulation process. 3. The system continuously monitors the state of various devices and sensors within the simulated house. 4. Whenever there is a change in the state of any device or sensor, the system updates the house view to reflect the current state. 5. The system graphically represents the current state of the simulation on the user interface.

2.2 Use-case diagram for the whole system

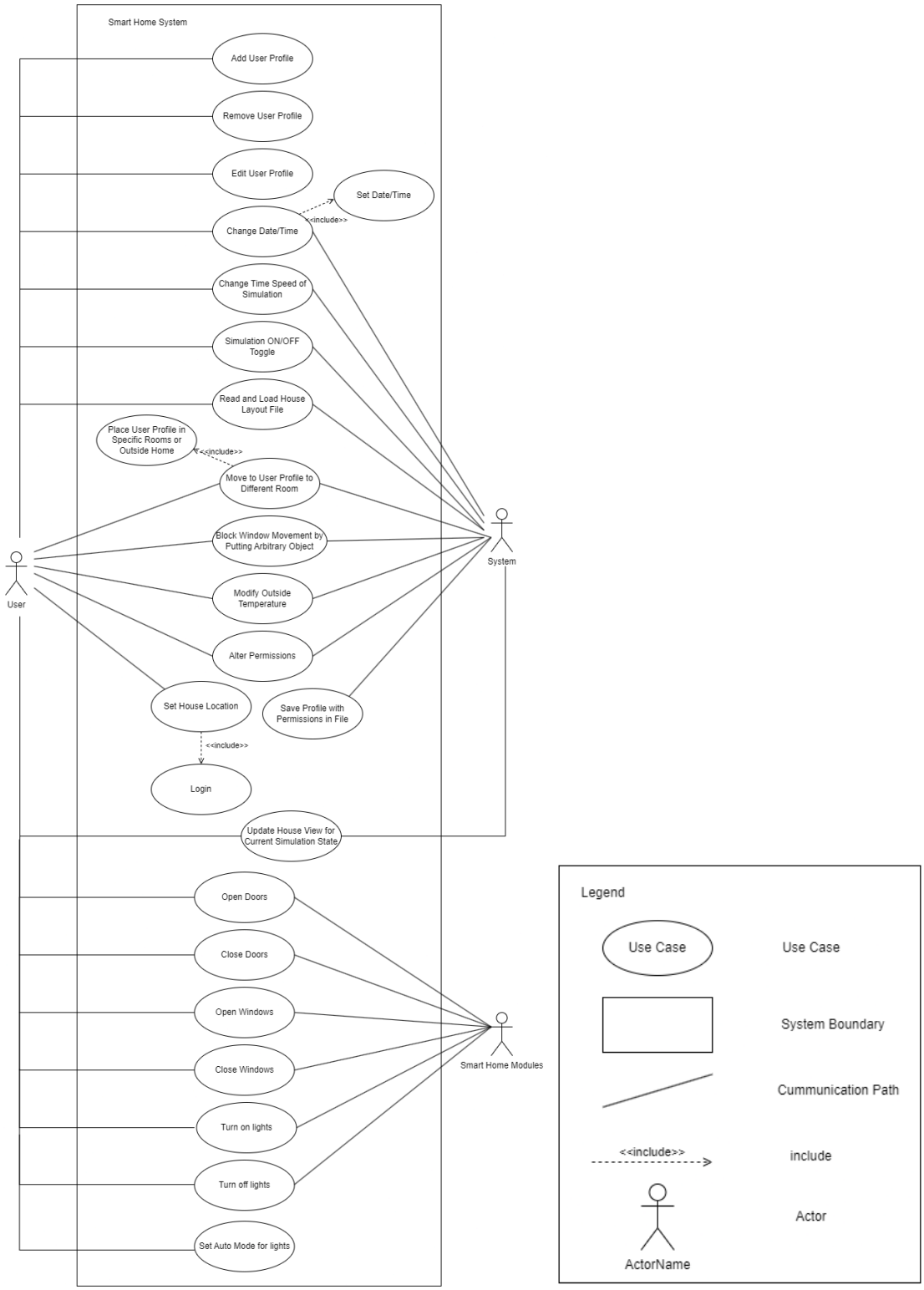


Figure 2: Use Case Diagram

2.3 Sequence diagram for the Smart home core functionality (SHC) module

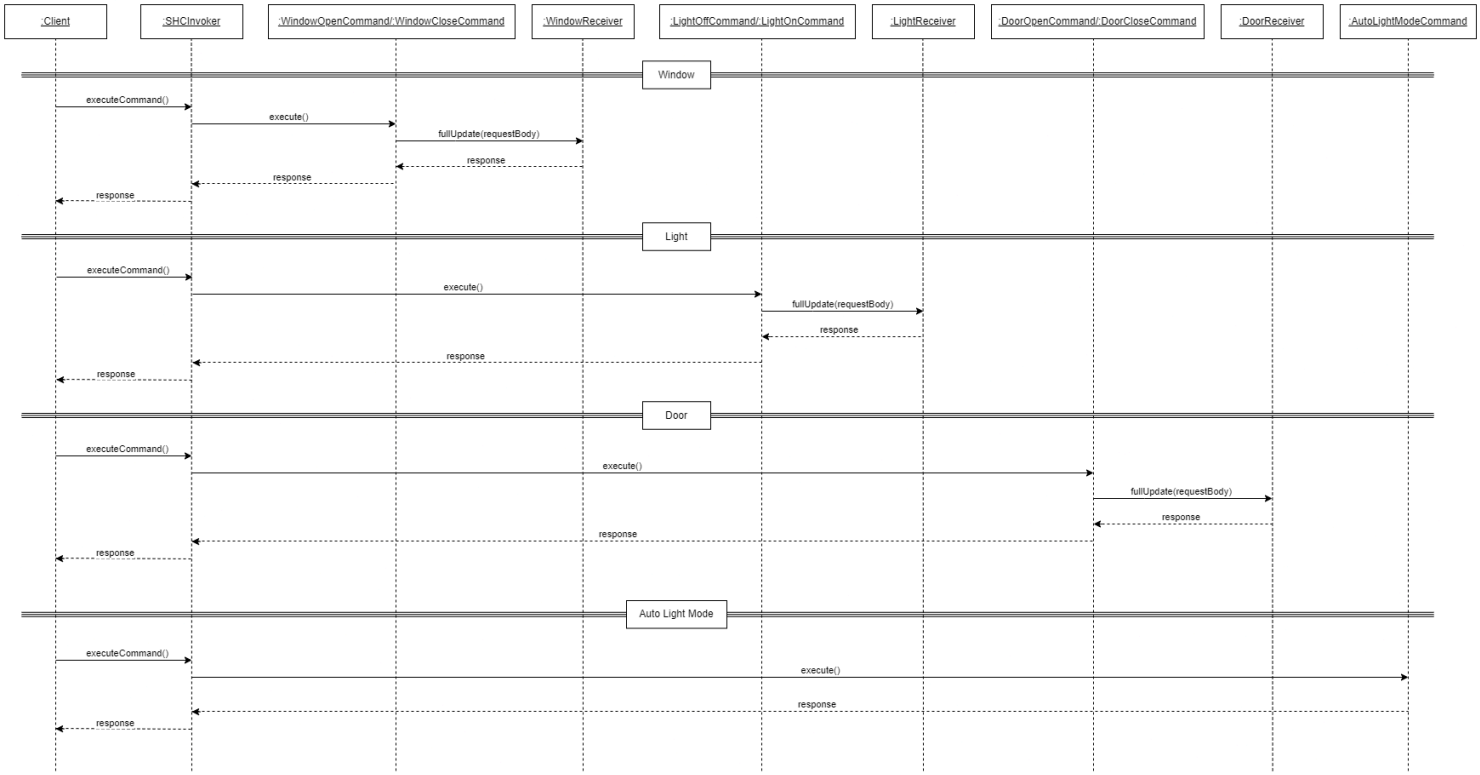


Figure 3: Sequence Diagram

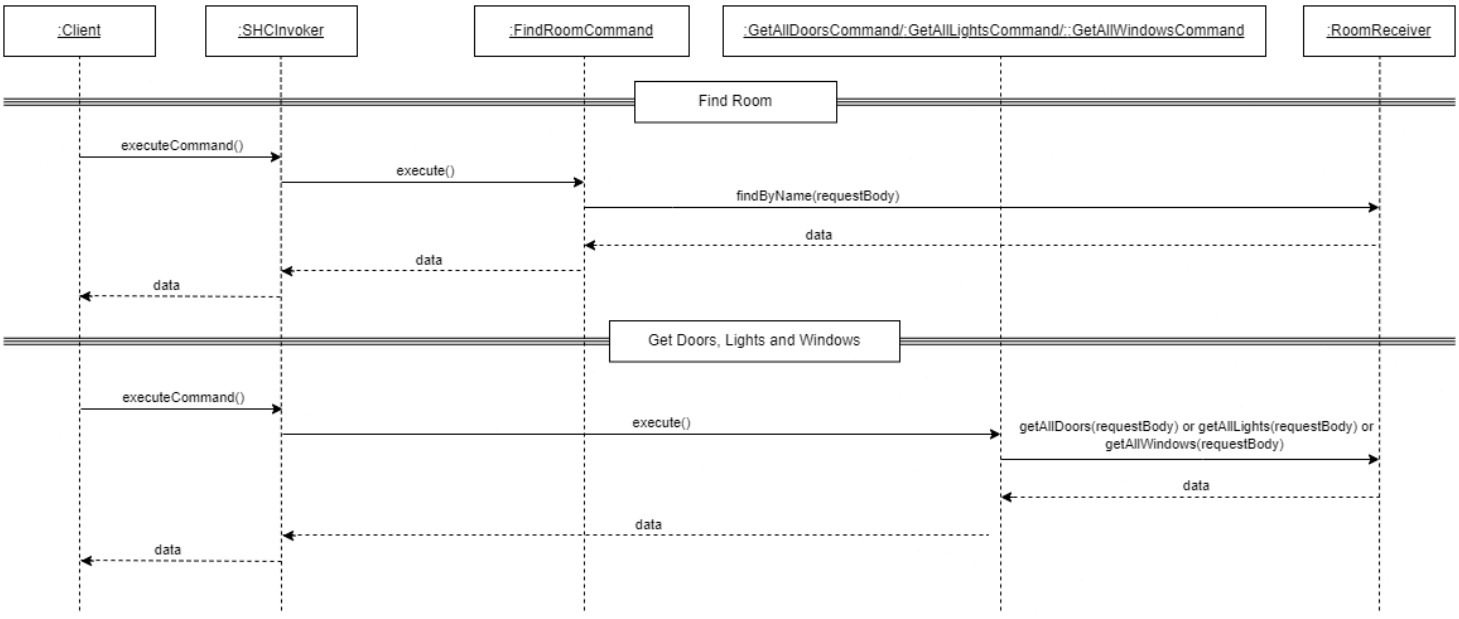
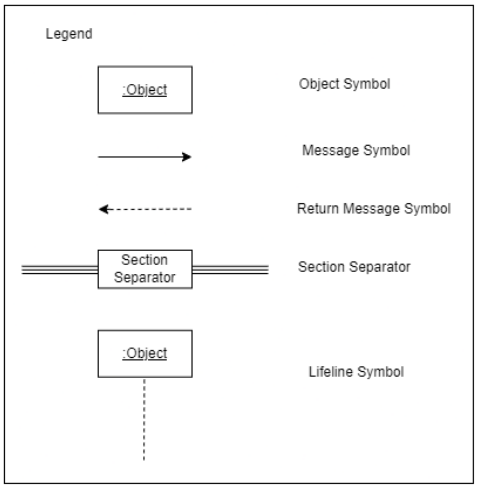


Figure 4: Note that this is an extension of Figure 3



2.4 State-machine diagram for the Context of the simulation

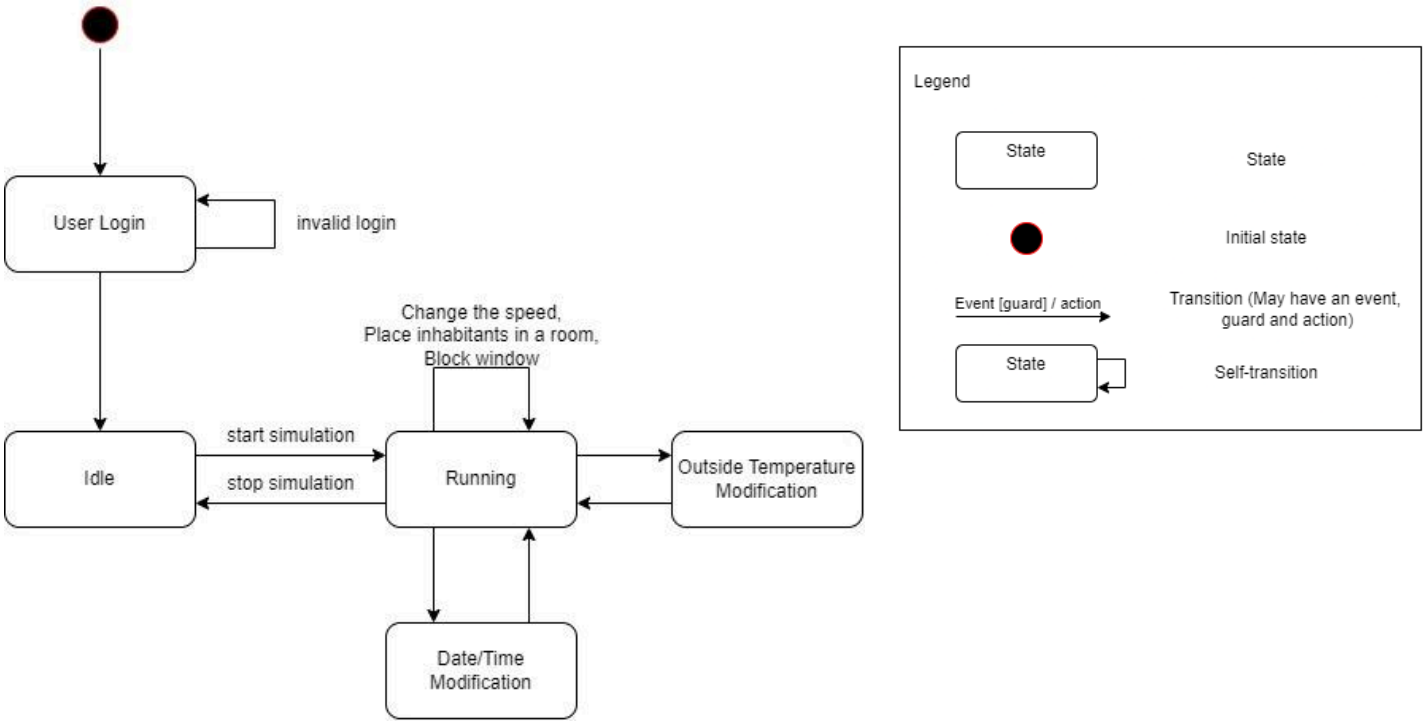


Figure 5: State Machine Diagram

2.5 Activity diagram for the Context of the simulation

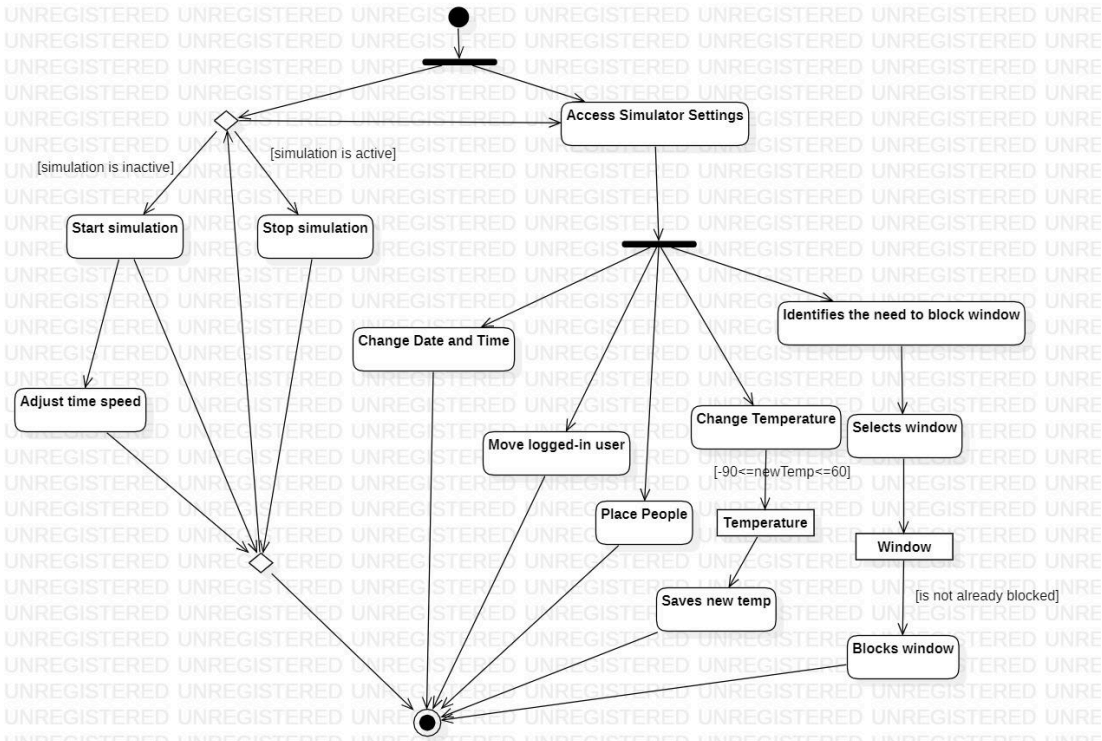


Figure 6: Activity Diagram for Context of the Simulation

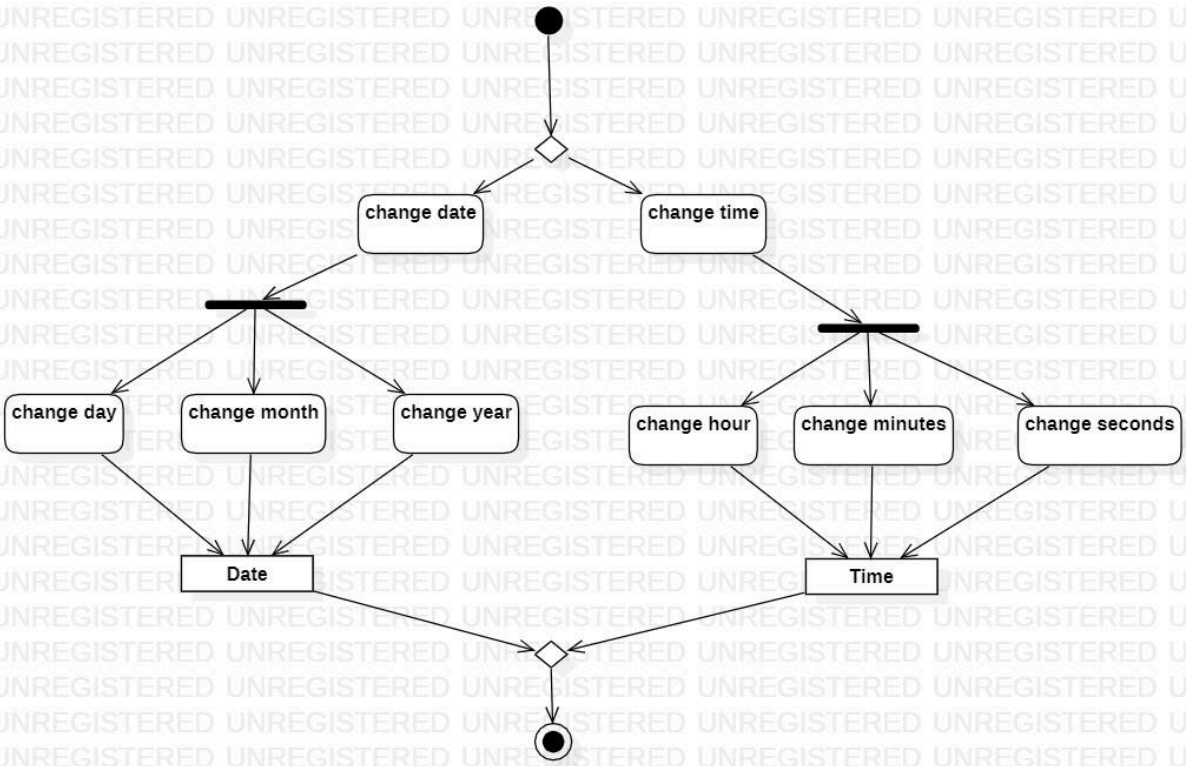


Figure 7: Activity Diagram for the Sub-Activity Change Date and Time

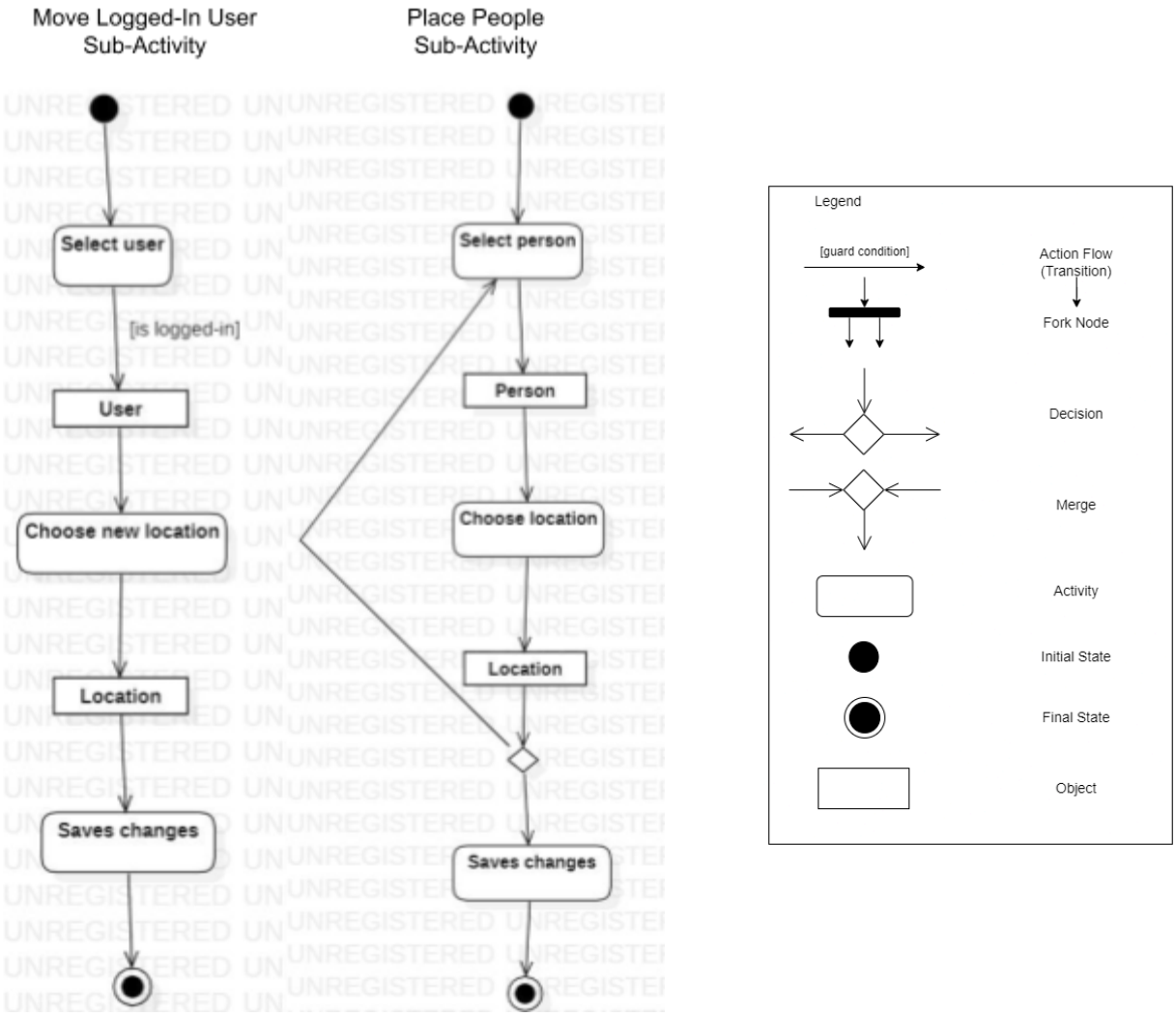


Figure 8: Move and Place People Sub-Activity

3. Design Patterns

Observer Pattern:

Implementing the observer pattern in the smart home simulator project would be a strategic decision due to its ability to handle the interactions and dependencies between various modules and components of the system. In a smart home environment, numerous events occur concurrently, such as changes in temperature and motion detection. By employing the observer pattern, each smart home module can subscribe to relevant events or changes of interest within the system. This ensures that when an event occurs, all subscribed modules are notified automatically, allowing them to react accordingly. For instance, the smart heating module (SHH) can observe changes in both indoor and outdoor temperatures, adjusting heating or cooling settings based on these observations. Similarly, the smart home security module (SHP) can observe motion detector

triggers to activate security measures. Utilizing the observer pattern promotes modularity and reduces coupling between modules, enabling easier maintenance, scalability, and flexibility within the smart home system.

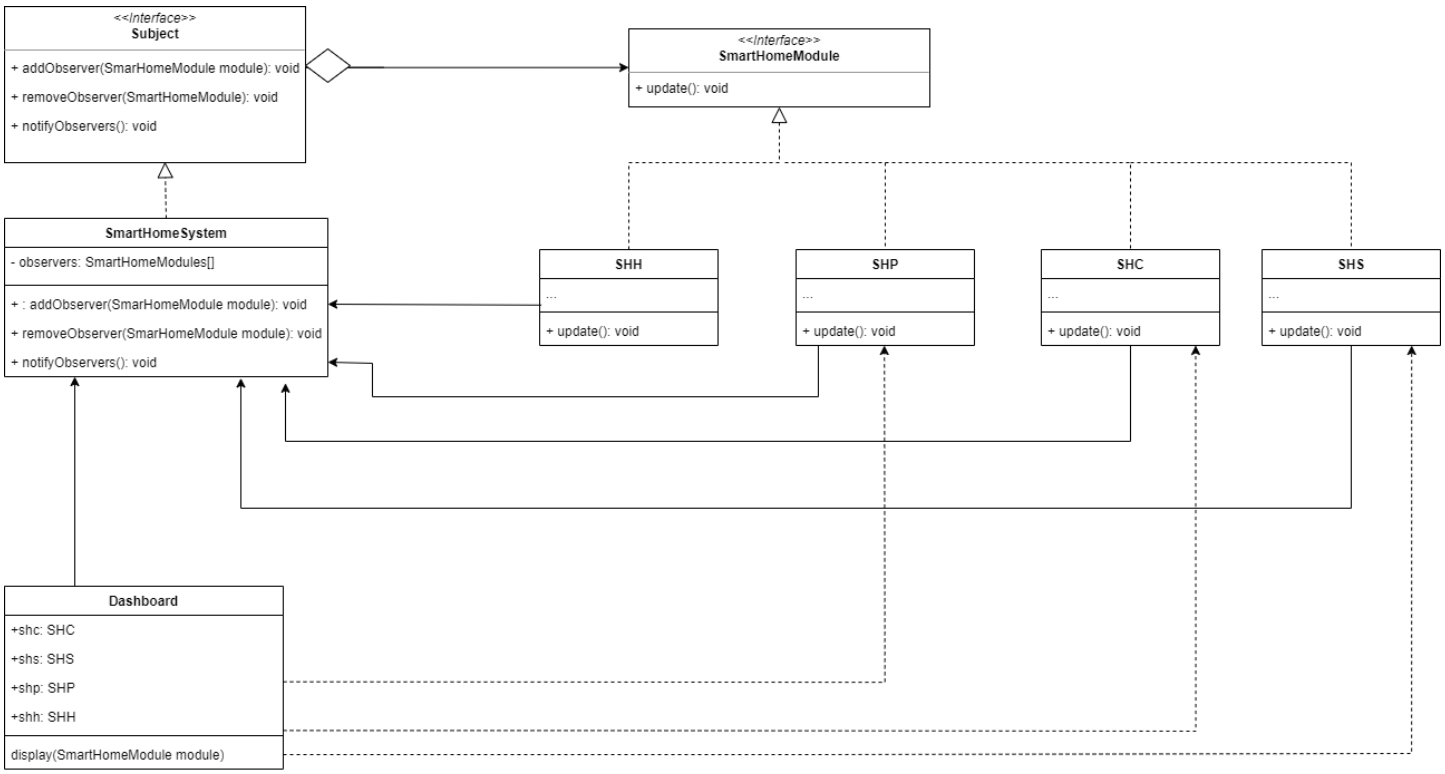


Figure 9: Observer Pattern Class Diagram

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.

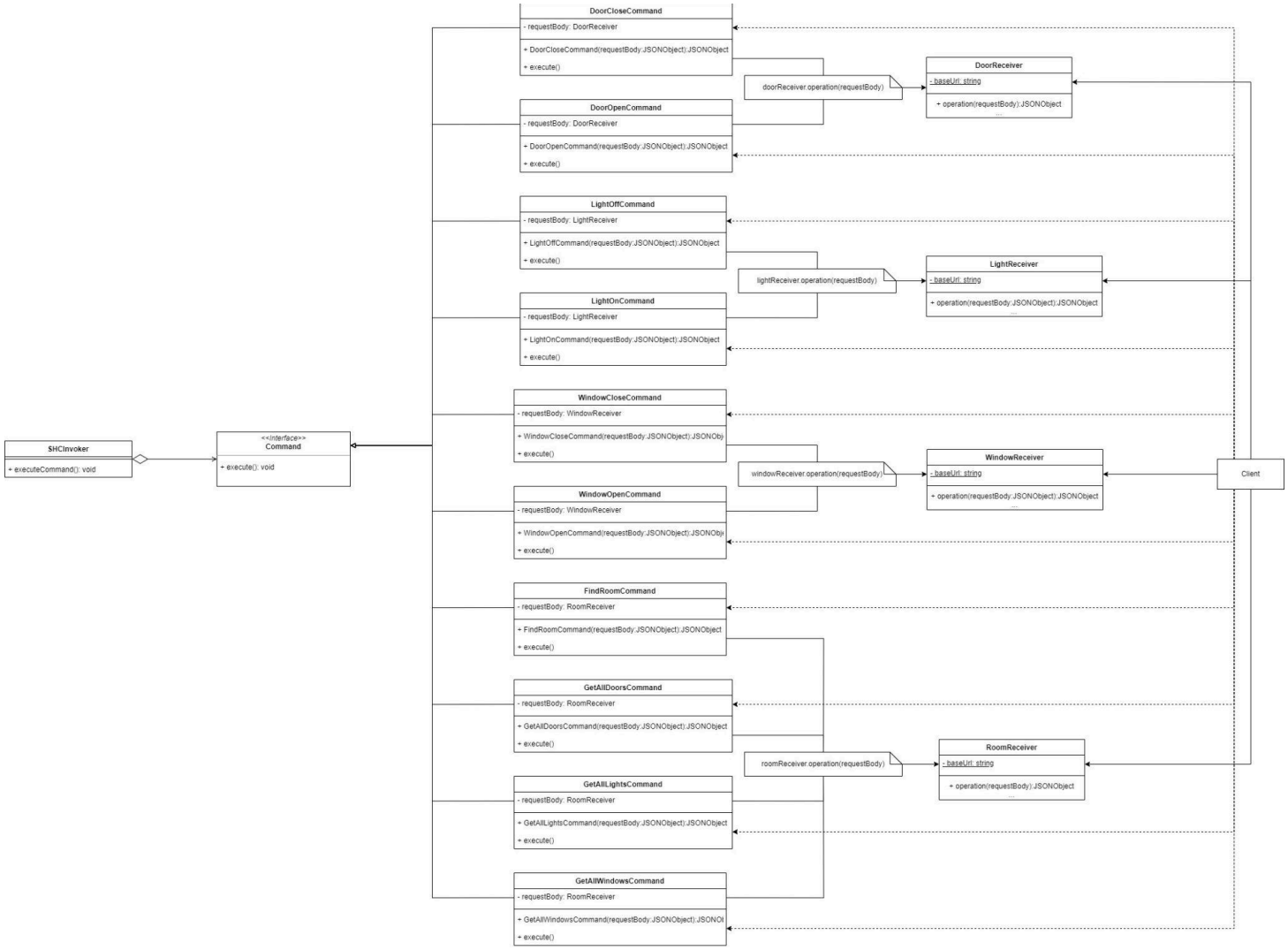


Figure 10: Command Pattern Class Diagram

State Pattern:

Implementing the State pattern in the smart home simulator project would be highly advantageous due to its ability to encapsulate and manage the diverse states and behaviors exhibited by various devices and systems within the smart home environment. In a dynamic setting like a smart home, devices such as heating systems, security modules, and lighting controls may transition between different operational states based on factors such as user preferences, environmental conditions, and system events. The State pattern allows each state of a device or system to be encapsulated into separate state classes, each responsible for defining specific behaviors and transitions associated with that state. For example, the smart heating module (SHH) could have states representing heating, cooling, standby, or off modes, with each state handling temperature adjustments and energy-saving strategies accordingly.

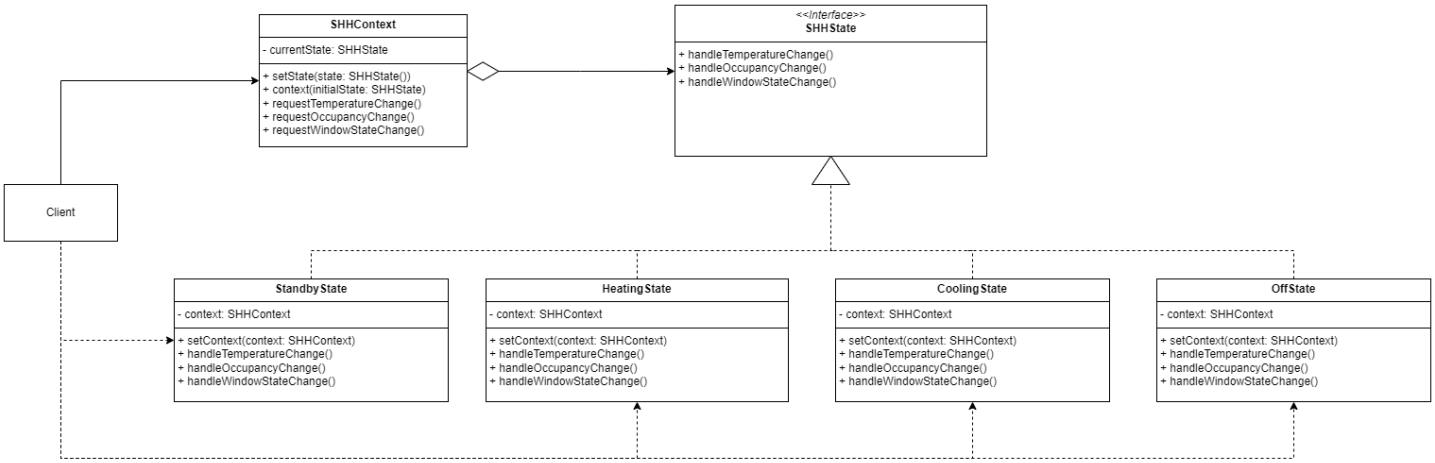
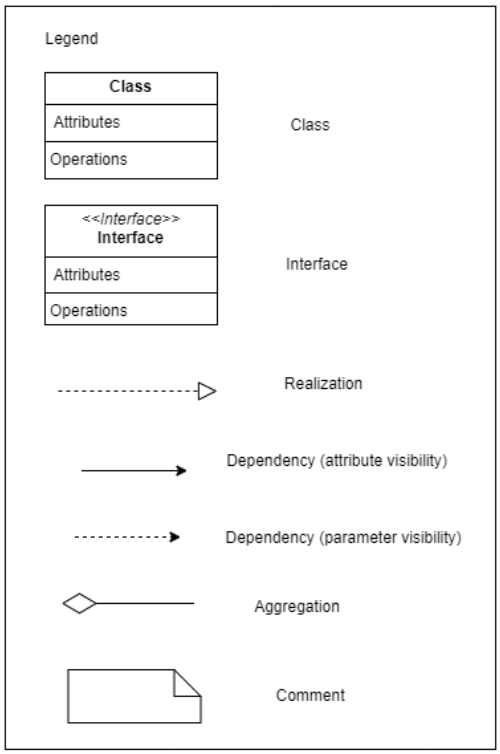


Figure 11: Command Pattern Class Diagram

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