SOEN 343:
SOFTWARE ARCHITECTURE AND DESIGN

iHome

Phase III

Tarek Elalfi 40197527
Khaled Saleh 40210125
Jay Patel 40203705
Omar Shehata 40164194 Team Leader
Samy Arab 40209595

Instructor: Dr. Rodrigo Morales Alvarado

Winter 2024

Table of Contents

Contents

Table of Figures

1. Problem definition

In this section, the problem statement, product position statement, and product overview is discussed to analyse the smart home simulator problems, solutions, stakeholders, assumptions, and dependencies

1.1 Problem Statement

In this section, we'll explore challenges in the smart homes simulator for the stakeholders, propose improvements and a successful solution.

The problem	The underlying issue is that individuals often face challenges related to convenience, efficiency, and security in regards to their household. Manual controls and traditional security systems are outdated which leads to inefficiencies and inconvenience.
Who it affects	This primarily affects homeowners as they are the main users that suffer from the stated problem. Manufacturers as well as developers are also affected as they are needed to build modern devices that support smart home systems.
The impacts	As a result of the stated problem, this impacts energy efficiency, convenience, security, safety and quality of life all in a negative manner.
Our solution	Our solution is to develop a smart home simulator application, iHome, that provides management features to modern smart home devices, up to date security features and a visual representation of the household that displays all smart devices. This solution will address the need for the modern usage of real-time monitoring without having to be manually within the household, improving scalability and flexibility. A user-friendly interface will allow users to easily navigate amongst the various modules available within our application. All in all, home owners can enjoy the benefits that a smart home environment offers such as convenience, safety and peace of mind, thus addressing all the problems previously stated.

1.2 Product Position Statement

In this section, we'll examine the product position statement for the smart home simulation platform. We'll discuss its target audience, unique value proposition, and differentiation from other solutions like Google Home and Amazon Alexa.

Who is this for?	The target customers are those affected by the problem at hand. In this case, our solution is catered towards homeowners.
The statement of need	Our application gives homeowners the opportunity to relieve themselves from the stress that comes with continuously monitoring all devices and security measures within the household. The smart home simulator addresses such issues of stress and ultimately will improve the quality of life of users.
What is *iHome*?	iHome is a smart home simulator application that allows users to navigate to 4 primary modules from the dashboard.
The statement of key benefit	Our application offers a seamless user-friendly experience where users can easily identify what they want by choosing the appropriate module to work with. Modules can manage the house layout, smart home devices, security measures as well as heating, all of which can be done without the need for the user to be within the household.
Our competition	Various tech companies such as Google and Amazon provide smart home technologies such as Google Home and Amazon Alexa.
How our product stands out from the competition	Although our competition provides smart home devices that can all interact well together, the issue is that they are often cumbersome to use, many of which rely on voice commands and require for users to be in close proximity to the household. Our application will address these issues, providing users with the flexibility to use a seamless user-interface to manage all of their smart home needs whenever and wherever they want.

1.3 Product Overview

In this section, an overview of the whole project in terms of the perspective, assumptions and dependencies for the smart home simulator.

1.3.1 Product Perspective

The Smart Home Simulator is an independent platform designed specifically for managing and optimizing simulated smart home automation systems. It provides a user-friendly interface and intuitive tools for experimentation, refinement, and innovation in smart home technology. Although the simulator is self-contained, it interacts with various modules, devices and technologies commonly used in smart home environments.

1.3.2 Assumptions and Dependencies

In this section, we'll outline the assumptions and dependencies crucial for developing the smart home simulation platform. We'll cover user knowledge, software requirements, and data access for realistic simulations.

Assumptions	Dependencies
Users have basic knowledge of using smart home technology, devices and concepts.	Availability of necessary software libraries and frameworks for the simulator smart home development.
The simulator runs on standard computers with access to necessary resources.	Adequate hardware resources (CPU, memory, storage) to ensure smooth operation and responsiveness of the simulator.
Simulated smart home modules and devices accurately reflect real-world behavior and interactions.	Ensure that the simulated application platform can accurately interact with the external files for the house layout necessary for realistic simulation scenarios.
Users have access to the necessary data and resources for configuring and running simulations.	The devices that are controlled by the modules send/receive signals that are needed by the smart home simulator

Problem Scope

The project scope for the course encompasses problem understanding, requirement analysis, UML modeling, software architecture, implementation of design patterns and practical applications of concepts following modern coding practices. The goal of the project is to comply with regiments that can be found in the modern world such as iterative development, strict guidelines and adhering to design principles.

2. Technology Used

In this section, we discuss what tools/languages are team will use for team collaboration, control version system, monitoring and verification, coding and development framework.

2.1 Team Collaboration

In our project ecosystem, various tools play distinct yet interconnected roles, harmonizing our development efforts seamlessly. Discord serves as our virtual meeting room, facilitating scrums and general discussions among team members. It doubles as a platform for file sharing and collaboration on design endeavors, fostering efficient communication and idea exchange.

2.2 Control Version System and Monitoring/Verification

Jira takes charge of our project's organizational backbone, meticulously tracking the status of technical artifacts and user stories. This meticulous oversight ensures swift progress aligned with scrum methodologies, keeping our development cycles agile and effective. Meanwhile, GitHub Wiki acts as our shared repository of technical documentation, ensuring all team members have unfettered access to vital resources.

2.3 Coding

For the backend infrastructure, we harness the robustness of Spring Boot, a powerful framework built on top of the Spring framework. This choice enables us to develop high-performing web applications and APIs efficiently, benefiting from its comprehensive suite of features, including sophisticated routing, security, and data access. On the frontend, React.js takes center stage, orchestrating our user interface with reusable components and contemporary design principles, all while upholding exceptional performance standards.

Git stands as our vigilant guardian of version control, facilitating seamless collaboration, code merging, and historical tracking across our distributed team. MongoDB, our database of choice, embraces the NoSQL paradigm, storing data in nimble JSON documents. This flexibility, coupled with its seamless integration with Next.js, ensures a smooth flow of data from server to client, in perfect harmony with the MVC architectural pattern.

2.4 Development Framework

To ensure a consistent and efficient development experience, we've rallied behind VS Code as our IDE of choice. Its robust feature set, bolstered by a vibrant ecosystem of extensions and community support, fosters a cohesive environment for coding excellence. And when it comes to design and modeling, Draw.io emerges as our trusted ally, empowering us to visualize and iterate upon system architectures with ease.

Finally, Figma takes center stage in our quest for stunning user interfaces, offering a canvas where creativity knows no bounds. With its suite of tools for interface design, we sculpt elegant, responsive UIs that elevate the user experience to new heights. Together, these tools form the bedrock of our development journey, enabling us to transform ideas into reality with precision and finesse.

3. Context diagram

A context diagram serves to illustrate the scope of a system alongside its interactions with external entities such as actors, physical objects or small-scale systems. Each external entity interacts with the system, while the system may also interact with entities. In the context of our project, the context diagram shown below helps to better understand the design of our smart home simulator system.

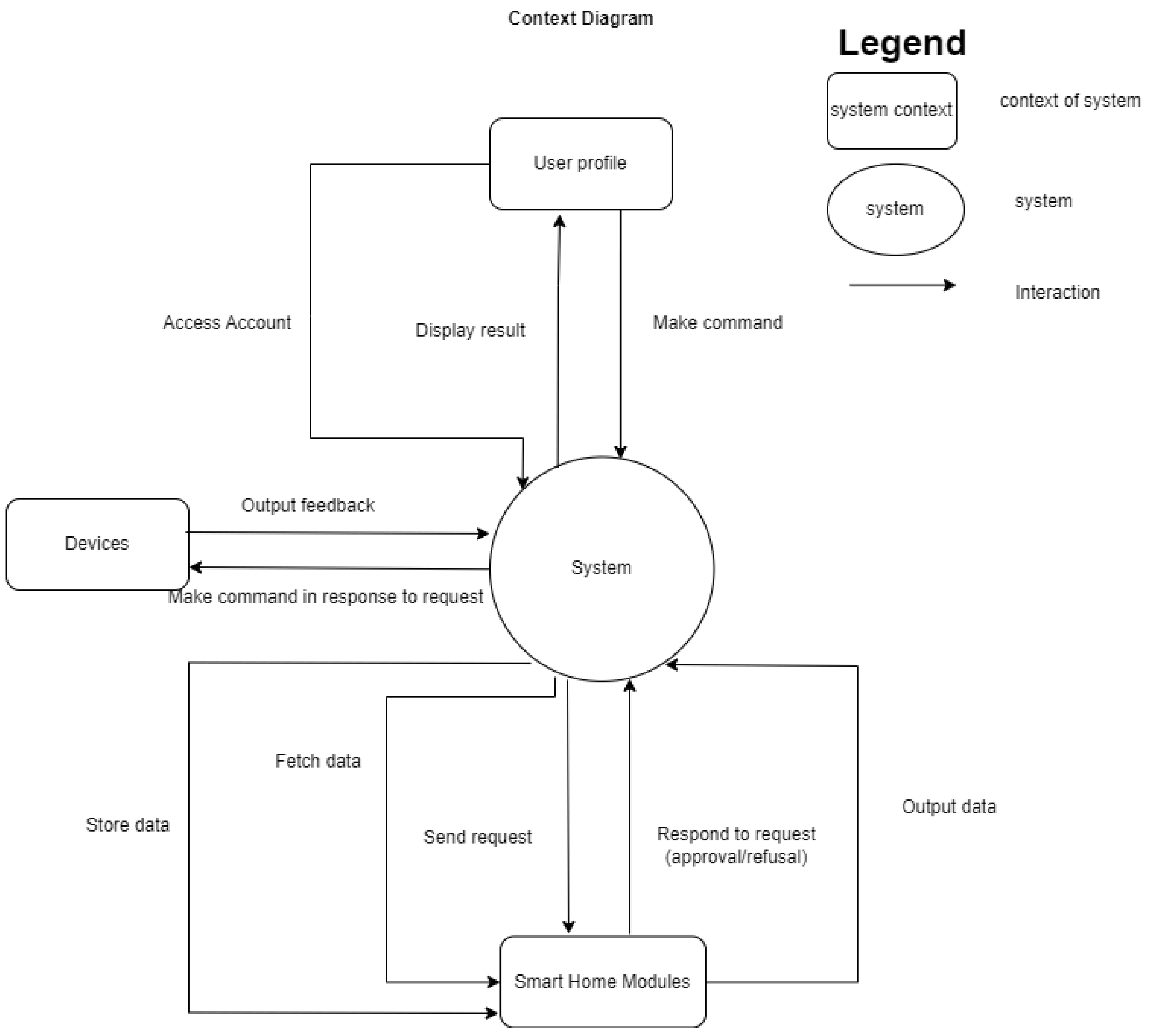


Figure 1: Context Diagram

In the context diagram of our smart home system, the system itself is positioned at the center and acts as the core intelligence and control hub, charged among other things with dispatching messages to the correct modules. The system is surrounded by three components: the user profile which contains all the users of every type (Family members, Guests, and strangers), the devices and the smart home modules. The User profile component encompasses the structure itself along with its inhabitants, indicating the environment in which

the smart home system operates. Furthermore, regarding the Devices segment, which include sensors, thermometers, cameras, lights, blinds, along with all the other possible appliances fall under this umbrella term that symbolizes the various endpoints within the home that interact with the system. Lastly, the Smart Home Modules component englobes the software responsible for managing and coordinating the interconnected device, facilitating communication, automation and user interaction. This context diagram illustrates the broader context in which the smart home system operates, highlighting its integration with both the physical environment and the technological infrastructure within the household.
A user will send a command to the system (through a user interface), the system will read and redirect a request to the relevant module, which will check the validity of the request and return an answer to the system. Based on this answer, the system will send a command to the desired device (if approved) and consequently let the user know of any changes (if any). In the context of fetching information like at what time a given door was opened or light closed, the process would entail a similar path, going from a user command, a system request to the appropriate module, and an output from that module.

4. Domain model (Nothing changed for Phase 3)

A domain model is a structured representation of the core elements and relationships within a problem domain. It defines entities and relationships, aiding in system design by abstracting away unnecessary details. The following domain model illustrates each component and the interaction between them.

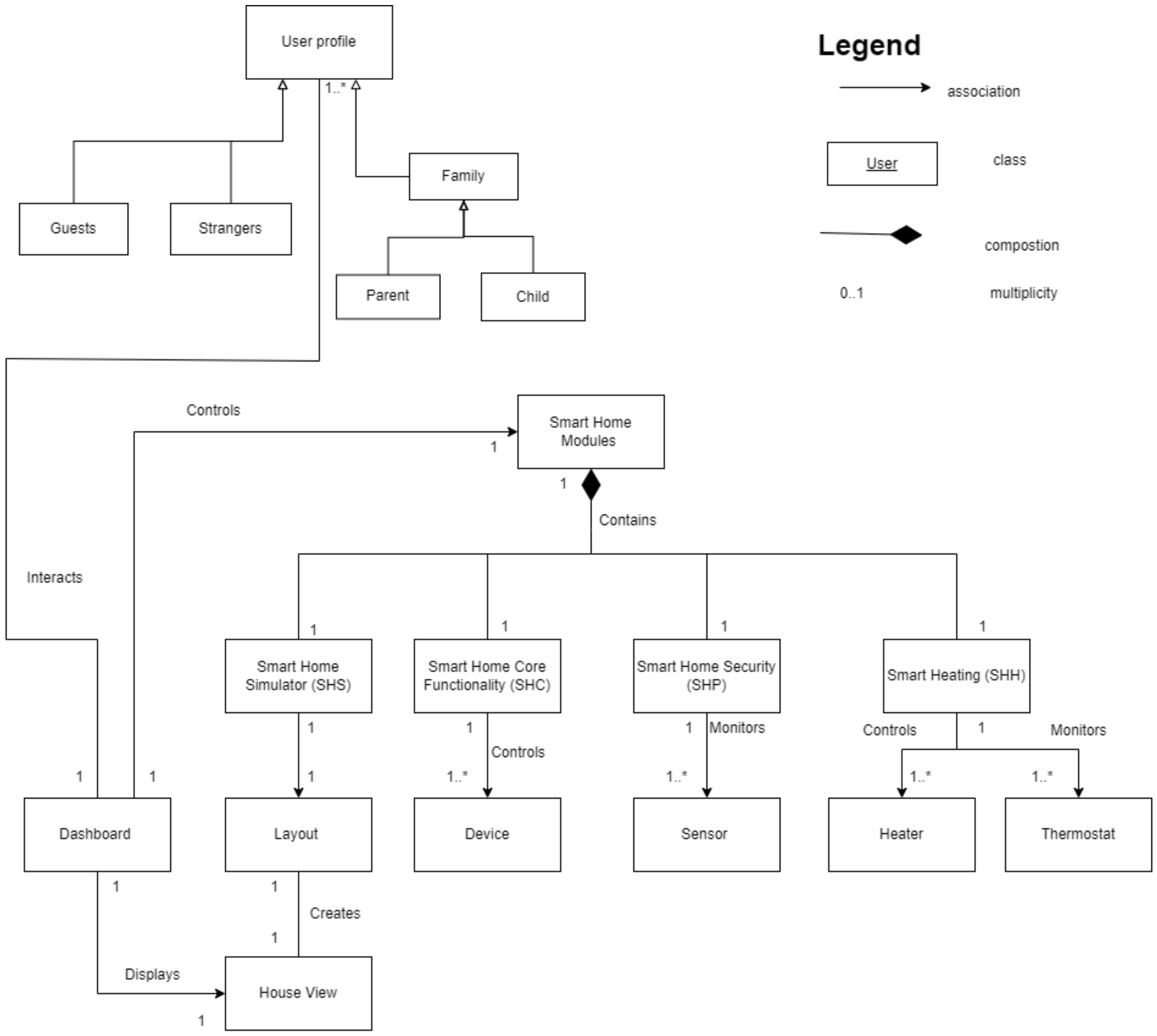


Figure 2: Domain Model

In the domain model of our smart home system, the user profile component encompasses the users inside the house, including Family members, guests, and strangers. The Dashboard component displays the main window of the application along with the modules and the House View. The House View component will be the 2D representation of the interior of the house. Furthermore, the Layout component will be a purely backend/logical component that reads the information about the interior organization of the house and translates this data in a way that the House View component can render. The Sensor, Heater and Thermostat components will abstract the behaviour of its real life counterparts. The Smart Home Modules component contains all the modules (Smart Home Simulator, Smart Home Core Functionality, Smart Home Security, Smart Heating) of this application and purely acts as a container. Finally, the Device component encompasses all the hardware that might be needed for the SHC component to function properly, for example, garage doors, front/back doors, light bulbs, etc.

PHASE II &PHASE III:

5.System Architecture (Nothing changed for Phase 3)

To better understand the architecture of our system, a package diagram was created to better illustrate the dependencies and relationship among components, which aids in understanding the structure of our system. We have defined 4 layers as shown in the below figure.

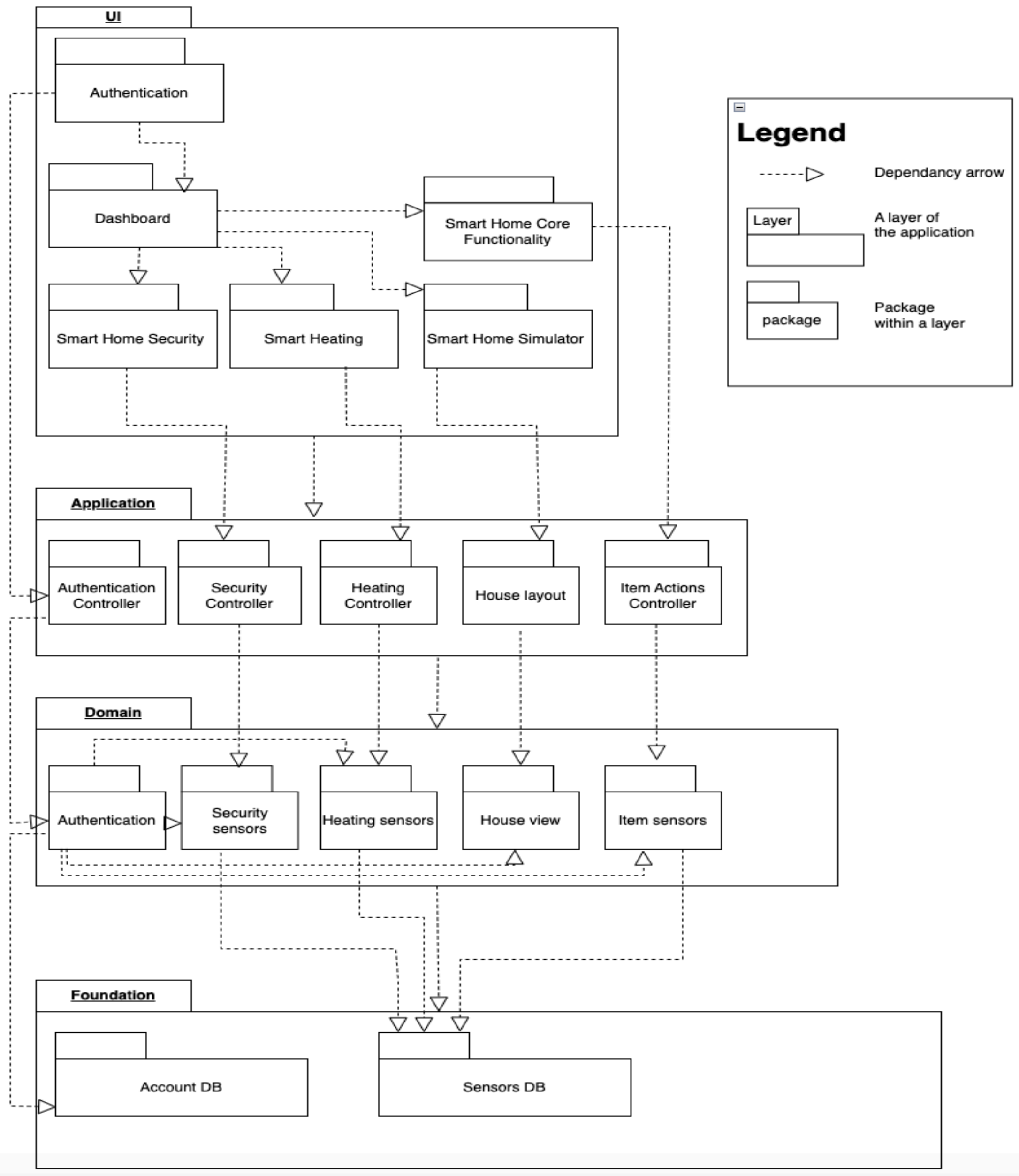


Figure 3. Package Diagram

Given the package diagram, the 4 layers are described as follows based on our application:

UI layer: For the system architecture, the UI Layer is the topmost layer and is intended to serve as the interface users will interact with. They interact with primary features such as accessing their account (authentication) and, in doing so, they are given access to the dashboard, providing modules such as Smart Home Core Functionality, Smart Home Security, Smart Heating and Smart Home Simulator.

Application Layer: The second layer, the Application Layer, handles system logic from the interface such as page transitions, session state and button submissions. Such operations are handled by their respective controllers such as Authentication, Security, Heating, House Layout and Item Actions.

Domain Layer: The third layer, the Domain Layer, contains the core business application logic of our system, handling all possible implementations defined by our product. Each controller submits their request to their respective domain packages such as Authentication, Security Sensors, Heating Sensors, House View and Item Sensors. Note that based on the requirements, only users with access will be able to utilize the sensors, hence why arrows are coming out of Authentication and pointing to Security Sensors, Heating Sensors and Item Sensors.

Foundation Layer: Finally, the Foundation Layer contains necessary resources that must be stored in some form of database. We have defined 2 types of databases; one for users (Accounts) and one for all types of sensors (Sensors). As such, all Security, Heating, and Item sensors will be found in the Sensors DB.

All in all, layering our system as such serves to understand better the core functionalities and how operations flow within our system. Note that the larger width of our foundation layers implies that this layer has a wider range of functionality. This is why as we go deeper into the layers, each layer's width becomes wider.

6. Use Case Model (Phase 3 use cases are added)

"Use Case Model," outlines key use cases for a smart home simulator, detailing scenarios from loading layouts to controlling smart devices. It includes diagrams to visualize user-system interactions, sequence flows, system states, and simulation contexts, offering insights into the simulator's functionalities and mechanics.

6.1 Use Cases

In the following section, we will document each required use case along with its details including preconditions, postconditions, inputs,outputs,etc. These use cases will be combined and their functionality will be summarized in the system use case diagram.

ID:	**UC1**
Title:	Read and load a house-layout file
Description:	User inputs a text file which will be read and interpreted into a 2D view of the house.
Primary Actor:	User
Preconditions:	● The user has the smart home simulator opened on their device. ● User needs to have an account ● User needs to be logged in ● User needs to be a Parent ● User has valid House layout text file
Postconditions:	● A 2D view is displayed on the dashboard.
Inputs:	House layout text file
Outputs:	2D view of house interior

Main Success Scenario:	● User opens home simulator ● User creates an account as a parent ● User signs in with that account ● User selects Smart Home Simulator (SHS) ● User inputs a valid text file ● System displays house interior design as a 2D diagram

Table 1: Use case to read and load the house layout

ID:	**UC2**
Title:	Add User Profiles
Description:	User creates a new account
Primary Actor:	User
Preconditions:	● The user has the smart home simulator opened on their device. ● User does not have an existing account
Postconditions:	● A user account is created
Inputs:	● User Information: First name, Last name, User Type, Phone Number, Email, Password
Outputs:	Prompt for user to log in with their provided credentials
Main Success Scenario:	● User opens home simulator ● User creates an account ● System prompts them to sign in

Table 2: Use case to add user profiles

ID:	**UC3**
Title:	Remove User Profiles
Description:	User removes an existing account
Primary Actor:	User
Preconditions:	• The user has the smart home simulator opened on their device. • User has an existing account
Postconditions:	• A user account is removed
Inputs:	• User Information: Email, Password
Outputs:	Prompt that the account has been successfully removed
Main Success Scenario:	• User opens home simulator • User signs in with their account • User goes to the settings of his user profile • User chooses the Delete Account option • User has to re-input their email and password • System prompts that the account has been deleted • User is signed out

Table 3: Use case to remove user profiles

ID:	**UC4**
Title:	Edit User Profiles
Description:	User edits an existing account
Primary Actor:	User

Preconditions:	● The user has the smart home simulator opened on their device. ● User has an existing account ● User is logged in
Postconditions:	● A user account is updated
Inputs:	● User Information: First name, Last name, User Type, Phone Number, Email, Password
Outputs:	Prompt that the account has been successfully updated
Main Success Scenario:	● User opens home simulator ● User signs in with their account ● User goes to the settings of his user profile ● User chooses the Edit Account option ● User enters the credentials he wishes to update ● System prompts that the account has been updated

Table 4: Use case to edit user profile

ID:	**UC5**
Title:	Set Date and time
Description:	User can set date and time shown in the simulator
Primary Actor:	User
Preconditions:	● The user has the smart home simulator opened on their device. ● User needs to be logged in ● User navigates to the dashboard ● System prompts the user to set a new date and time
Postconditions:	● The new date and time are displayed on the dashboard.

Inputs:	Date and Time (Time Zone)
Outputs:	Date and Time on Dashboard
Main Success Scenario:	● User opens home simulator ● User signs in with that account ● System redirects user to the dashboard ● User sets a new date and time on the dashboard ● System displays the new date and time entered by the user

Table 5: Use case to set date and time

ID:	**UC6**
Title:	Log in using an existing user profile and set house location
Description:	User logs in with existing credentials for a corresponding account
Primary Actor:	User
Preconditions:	● The user has the smart home simulator opened on their device. ● System prompts user to sign in
Postconditions:	● User is signed in
Inputs:	● User Information: Email, Password
Outputs:	Redirected to dashboard
Main Success Scenario:	● User opens home simulator ● System prompts them to sign in ● User enters their credentials ● System redirects user to the dashboard

Table 6: Use case to log in existing user profile

ID:	**UC7**
Title:	System shall allow simulator's user to grant permissions for another user
Description:	User chooses to grant permission for a user with less permission to action a command that is outside the other user's jurisdiction
Primary Actor:	User
Preconditions:	● The lower permission user has the smart home simulator opened on their device. ● Lower permission user is signed in ● Lower permission user chooses a smart home module ● Lower permission user chooses to act outside their jurisdiction ● The higher permission user has the smart home simulator opened on their device. ● Higher permission User is signed in ● System prompts screen to allow higher permission user to give system access to perform action
Postconditions:	● Lower permission user is granted access ● System performs action ● System shows prompt to indicate action has been done
Inputs:	● Confirmation: Yes/No
Outputs:	Prompt that shows that action has been done
Main Success Scenario:	● The Guest user has the smart home simulator opened on their device. ● Guest User is signed in

	● Guest user chooses Smart Home Core Functionality ● Guest user chooses to close the door ● The Parent user has the smart home simulator opened on their device. ● Parent User is signed in ● System prompts screen to allow parent user to give system access to perform action ● Parent user gives access to the guest to close the door ● System closes door ● System shows prompt that indicates the door has been closed

Table 7: Use case to System shall allow simulator's user to grant permissions

ID:	**UC8**
Title:	System shall allow simulator's user to deny permissions for another user
Description:	User chooses to deny permission for a user with less permission to action a command that is outside the other user's jurisdiction
Primary Actor:	User
Preconditions:	● The lower permission user has the smart home simulator opened on their device. ● Lower permission user is signed in ● Lower permission user chooses a smart home module ● Lower permission user chooses to act outside their jurisdiction ● The higher permission user has the smart home simulator opened on their device. ● Higher permission User is signed in

	• System prompts screen to allow higher permission user to give system access to perform action
Postconditions:	• Lower permission user is denied access • System refuses action • System shows prompt to indicate action has been refused
Inputs:	• Confirmation: Yes/No
Outputs:	Prompt that shows that action has been refused
Main Success Scenario:	• The Guest user has the smart home simulator opened on their device. • Guest User is signed in • Guest user chooses Smart Home Core Functionality • Guest user chooses to close the door • The Parent user has the smart home simulator opened on their device. • Parent User is signed in • System prompts screen to allow parent user to give system access to perform action • Parent user denies to give access to guest to close the door • System refuses to close door • System shows prompt that indicates door has not been closed

Table 8: Use case to System shall allow simulator's user to deny permissions

ID:	**UC9**
Title:	System Saves User Permissions
Description:	System shall save in a file the profiles, with their corresponding permissions, to avoid reentering the information every time you open the simulator

Primary Actor:	System
Preconditions:	● The smart home simulator has at least one profile available
Postconditions:	● Profiles are saved into a database
Inputs:	● User profile(s)
Outputs:	Records in database that include profile(s) details
Main Success Scenario:	● User has the smart home simulator opened on their device. ● User creates an account ● System calls database to save created account details ● Database saves profile record

Table 9: Use case to System shall save user permission

ID:	**UC10**
Title:	Start simulation
Description:	The system enables users to initiate or halt a simulation which models device behaviour in the smart home without actual device control.
Primary Actor:	User
Preconditions:	● User is logged into the smart home dashboard. ● User has permission to manage simulations.
Postconditions:	● Simulation is started/stopped
Inputs:	● User Email ● User Password
Outputs:	Prompt that states that simulation has started

Main Success Scenario:	● User has the smart home simulator opened on their device. ● User creates an account ● System calls database to save created account details ● Database saves profile record ● User presses start simulation button ● System prompts that simulation is started

Table 10: Use case to start the simulation

ID:	**UC11**
Title:	Modify date and time
Description:	The system permits the user to adjust the date and time settings, affecting the scheduling and timing of smart home operations.
Primary Actor:	User
Preconditions:	● User is authenticated on the smart home dashboard. ● User has access rights to system settings.
Postconditions:	● The system's date and time are updated.
Inputs:	● User Email ● User Password ● User selection of date and time
Outputs:	Updated system date and time.
Main Success Scenario:	● User logs into the smart home dashboard. ● User navigates to the system settings. ● User enters new date and time. ● System confirms and applies the changes.

Table 11: Use case to modify date and time

ID:	**UC12**

Title:	Move the logged user to different room
Description:	The system allows a logged-in user to update their current room location in the smart home environment.
Primary Actor:	User
Preconditions:	● User is logged into the smart home system. ● User has the capability to update room location status.
Postconditions:	● The system reflects the user's new room location.
Inputs:	● User Email ● User Password ● User selection of new room
Outputs:	User's location updated within the system.
Main Success Scenario:	● User accesses the smart home system. ● User selects the option to change the current room. ● User chooses a new room from the list. ● System updates the user's location to the new room.

Table 12: Use case to move the logged user to different rooms

ID:	**UC13**
Title:	Place people in specific rooms, or outside home
Description:	The system allows users to assign simulated individuals to specific rooms or set them as being outside the home, affecting the simulation of the smart environment.
Primary Actor:	User
Preconditions:	● User is signed into the smart home management dashboard.

	• The simulation mode is accessible and modifiable by the user.
Postconditions:	• Simulated individuals are placed in designated rooms or marked as outside.
Inputs:	• User Email • User Password • Selection of individuals and their intended locations within or outside the home.
Outputs:	Updated simulation reflecting the new locations of individuals.
Main Success Scenario:	• User logs into the dashboard. • User navigates to the simulation management area. • User selects individuals to place in specific rooms or outside the home. • System updates the simulation to reflect the new placements.

Table 13: Use case to place people in specific rooms

ID:	**UC14**
Title:	Modify temperature outside home
Description:	The system provides functionality for the user to alter the simulated external temperature setting, impacting smart home environmental controls.
Primary Actor:	User
Preconditions:	• User is signed into the smart home dashboard. • User has permission to adjust environmental simulations.
Postconditions:	• The simulated external temperature setting is updated.
Inputs:	• User Email • User Password

	● Desired external temperature value selected by the user.
Outputs:	Updated external temperature in the simulation.
Main Success Scenario:	● User logs into the dashboard. ● User accesses the environmental control settings. ● User sets a new external temperature. ● System updates to simulate the new external temperature.

Table 14: Use case to modify outside temperature

ID:	**UC15**
Title:	Block windows movement by putting an arbitrary object
Description:	The system allows users to simulate the effect of placing an arbitrary object that blocks the movement of windows, affecting how windows can be opened or closed within the simulation.
Primary Actor:	User
Preconditions:	● User is logged into the smart home simulation dashboard. ● User has access to window control simulations.
Postconditions:	● Window movement is restricted as if blocked by an object in the simulation.
Inputs:	● User Email ● User Password ● The user's selection of a window to block and the choice of an arbitrary object.
Outputs:	A simulation update where the selected window's movement is blocked.

Main Success Scenario:	● User logs into the simulation dashboard. ● User navigates to the window controls section. ● User selects a window to block and chooses an object to simulate blocking it with. ● System updates to restrict the selected window's movement, simulating the effect of the object.

Table 15: Use case to block windows movement

ID:	**UC16**
Title:	Open doors
Description:	System should allow users to open doors
Primary Actor:	User
Preconditions:	● The user has the smart home simulator opened on their device. ● User is signed in. ● User must have permission to open the door
Postconditions:	● The door is open
Inputs:	● User email, user type,and door number
Outputs:	● Door is open
Main Success Scenario:	● User opens home simulator ● User signs in ● System redirects user to the dashboard ● User chooses the Smart home core functionality module ● User chooses door number ● User selects open door ● System prompts the door to be opened

Table 16: Use case to open doors

ID:	**UC17**
Title:	Close doors
Description:	System should allow users to close doors
Primary Actor:	User
Preconditions:	● The user has the smart home simulator opened on their device. ● User is signed in. ● User must have the permission to close the door
Postconditions:	● The door is closed
Inputs:	● User email, user type, and door number
Outputs:	● Door is closed
Main Success Scenario:	● User opens home simulator ● User signs in ● System redirects user to the dashboard ● User chooses the Smart home core functionality module ● User chooses door number ● User selects close door ● System prompts the door to be closed

Table 17: Use case to close doors

ID:	**UC18**
Title:	Open windows
Description:	System should allow users to open windows

Primary Actor:	User
Preconditions:	• The user has the smart home simulator opened on their device. • User is signed in. • User must have permission to open the window
Postconditions:	• The window is opened
Inputs:	• User email, user type, and window number
Outputs:	• Window is opened
Main Success Scenario:	• User opens home simulator • User signs in • System redirects user to the dashboard • User chooses the Smart home core functionality module • User chooses window number • User selects open window • System prompts the window to be opened

Table 18: Use case to open windows

ID:	**UC19**
Title:	Close windows
Description:	System should allow users to close windows
Primary Actor:	User
Preconditions:	• The user has the smart home simulator opened on their device. • User is signed in. • User must have permission to close the window
Postconditions:	• The door is closed
Inputs:	• User email, user type, and window number

Outputs:	● Window is closed
Main Success Scenario:	● User opens home simulator ● User signs in ● System redirects user to the dashboard ● User chooses the Smart home core functionality module ● User chooses window number ● User selects close window ● System prompts the window to be closed

Table 19: Use case to close windows

ID:	**UC20**
Title:	Turn on lights
Description:	System should allow users to turn on the lights
Primary Actor:	User
Preconditions:	● The user has the smart home simulator opened on their device. ● User is signed in. ● User must have permission to turn on the lights
Postconditions:	● The lights are turned on
Inputs:	● User email, user type, and light bulb number
Outputs:	● Lights are turned on
Main Success Scenario:	● User opens home simulator ● User signs in ● System redirects the user to the dashboard ● User chooses the Smart home core functionality module ● User chooses light bulb number ● User selects turn on lights

	• System prompts the lights to be turned on

Table 20: Use case to turn on lights

ID:	**UC21**
Title:	Turn off lights
Description:	System should allow users to turn off the lights
Primary Actor:	User
Preconditions:	• The user has the smart home simulator opened on their device. • User is signed in. • User must have the permission to turn off the lights
Postconditions:	• The lights are turned off
Inputs:	• User email, user type, and light bulb number
Outputs:	• Lights are turned off
Main Success Scenario:	• User opens home simulator • User signs in • System redirects the user to the dashboard • User chooses the Smart home core functionality module • User chooses light bulb number • User selects turn off lights • System prompts the lights to be turned off

Table 21: Use case to turn off lights

ID:	**UC22**
Title:	Set Auto mode for room entry

Description:	The system shall allow users to enable an Auto mode that automatically turns on or off the lights when it detects someone entering or leaving a room.
Primary Actor:	User
Preconditions:	● The user has the smart home simulator opened on their device. ● User is signed in. ● User must have the permission to configure Auto mode settings.
Postconditions:	● Auto mode for turning lights on or off upon room entry is set.
Inputs:	● User email, user type, room number, and Auto mode settings
Outputs:	● Auto mode is configured for the specified room.
Main Success Scenario:	● User opens home simulator. ● User signs in. ● System redirects the user to the dashboard. ● User chooses the Smart home core functionality module. ● User selects room to configure for Auto mode. ● User accesses Auto mode settings. ● User configures Auto mode to turn on/off lights upon entry. ● System saves Auto mode settings. ● System confirms that Auto mode is set.

Table 22: Use case to set auto mode for room entry

ID:	**UC23**
Title:	Update House View in Smart Home Dashboard
Description:	The system shall update the graphical representation of the house view on the smart home dashboard to reflect the current state of the simulation when the simulation mode is activated.
Primary Actor:	User
Preconditions:	● The user has the smart home dashboard opened on their device. ● User is signed in. ● The dashboard is connected to the home simulation system.
Postconditions:	● The house view on the dashboard reflects the current state of the simulation.
Inputs:	● User email, user type, simulation mode status

Outputs:	• Updated graphical representation of the house view in simulation mode.
Main Success Scenario:	• User opens smart home dashboard. • User signs in. • System redirects the user to the dashboard homepage. • User activates the simulation mode using the on-toggle button. • System detects the simulation mode is on. • System updates the house view to graphically represent the current state of the simulation. • User views the updated house view that reflects the simulated statuses of devices and sensors.

Table 23: Use case to update house view

Smart heating (SHH) Module Use case(Phase 3)

ID:	**UC24**
Title:	Configure Smart Home Heating/Cooling Zones
Description:	This use case allows the user to divide the smart home into different heating and cooling zones via the Smart Home Hub (SHH) interface. It enables customized temperature settings for different areas in the home, allowing for energy-saving strategies like setting a lower temperature in the garage compared to living spaces.
Primary Actor:	User
Preconditions:	• User is signed in. • The user has the smart home heating tab opened on their device. • House Layout is read and rendered
Postconditions:	• The home is successfully divided into zones with specified temperature settings. • All rooms are assigned to at least one zone.
Inputs:	• User email • User Password • List of room(s) inside each zone
Outputs:	• List of Zones with each room inside that zone
Main Success Scenario:	• User signs in. • User opens SHH tab • User selects zoning setting • User adds rooms 1 and 2 to zone 1

	● User adds rooms 3 and 4 to zone 2 ● User saves configuration ● System saves zones

Table 24: Use case to configure Smart Home Heating/Cooling zones

ID:	**UC25**
Title:	Set Variable Temperature Schedules per Zone
Description:	Users can set up temperatures for each zone for up to 3 periods in the day.
Primary Actor:	User
Preconditions:	● User is signed in. ● The user has the smart home heating tab opened on their device. ● House Layout is read and rendered ● Zones have been assigned
Postconditions:	● Temperatures are assigned to each zone depending on time period.
Inputs:	● User email ● User Password ● Temperature for each zone ● Time period for each temperature
Outputs:	● List of assigned temperatures per zone per time period
Main Success Scenario:	● User signs in. ● User opens SHH tab ● User selects zone 1. ● User assigns it 8 degrees for the morning and 12 degrees for the afternoon. ● User saves configuration. ● System saves zones’ temperatures and periods.

Table 25: Use case to set variable temperature schedules per zone

ID:	**UC26**
Title:	Display Current Room Temperature
Description:	Capability of SHH to display the temperature of any room within a house. It ensures that users can monitor the temperature in real time to maintain comfort and manage their home's heating and cooling efficiency.
Primary Actor:	User

Preconditions:	● User is signed in. ● The user has the smart home heating tab opened on their device. ● House Layout is read and rendered
Postconditions:	● Temperature displayed for the chosen room
Inputs:	● User email ● User Password ● User selection of which room to display temperature for
Outputs:	● Temperature for desired room
Main Success Scenario:	● User signs in. ● User opens SHH tab ● User selects display temperature button for room 1 ● System displays temperature for room 1

Table 26:Use case to display current room Temperature

ID:	**UC27**
Title:	Inside/Outside Temperature Monitoring
Description:	Capability of SHH to monitor temperature of indoors and outdoors to be able to assist additional functionality.
Primary Actor:	User
Preconditions:	● User is signed in. ● The user has the smart home heating tab opened on their device. ● House Layout is read and rendered
Postconditions:	● SHH has record of indoor and outdoor temperature
Inputs:	● User email ● User Password ● Temperature data for indoors ● Temperature data for outdoor
Outputs:	● Record of indoor temperature ● Record of outdoor temperature
Main Success Scenario:	● User signs in. ● System receives indoor temperature ● System receives outdoor temperature ● System keeps records of both temperatures.

Table 27:Use case to inside/outside temperature Monitoring

ID:	**UC28**
Title:	Automatic Air conditioning shutdown
Description:	This use case enables the smart home system to automatically shut down the air conditioning unit when the external temperature is cooler than the internal temperature during summer months.
Primary Actor:	Smart Home System
Preconditions:	● The smart home system is operational, with access to both indoor and outdoor temperature sensors. ● It is summertime, based on the system's internal date set. ● The air conditioning unit is currently operational.
Postconditions:	● The air conditioning unit is turned off when external temperatures are cooler than inside.
Inputs:	● Current indoor temperature (from indoor sensors). ● Current outdoor temperature (from outdoor sensors). ● Status of the air conditioning unit (on/off).
Outputs:	● Command to shut down the air conditioning unit if the outdoor temperature is cooler than the indoor temperature.
Main Success Scenario:	● The system continuously monitors indoor and outdoor temperatures during summer. ● The system detects that the outdoor temperature is cooler than the indoor temperature. ● The system checks if the air conditioning unit is currently on. ● If the air conditioning unit is on, the system sends a command to shut down the unit. ● The system logs the action taken for reference and potential user notification.

Table 28:Use case to automatic air conditioning shutdown

ID:	**UC29**
Title:	Automatic Window opening
Description:	This use case details the smart home system's capability to automatically open windows when it detects that the external temperature is cooler than the internal temperature during the summer.
Primary Actor:	Smart Home System
Preconditions:	● The smart home system is operational, with access to both indoor and outdoor temperature sensors. ● It is summertime, based on the system's internal date set. ● Windows are currently closed.
Postconditions:	● Windows are opened automatically to allow cooler external air to reduce the indoor temperature.
Inputs:	● Current indoor temperature (from indoor sensors). ● Current outdoor temperature (from outdoor sensors). ● Status of windows (open/closed).
Outputs:	● Command to open windows if the outdoor temperature is cooler than the indoor temperature.
Main Success Scenario:	● The system continuously monitors indoor and outdoor temperatures during summer. ● The system detects that the outdoor temperature is cooler than the indoor temperature. ● The system checks the status of windows (specifically, if they are closed). ● If windows are closed, the system sends a command to open them. ● The system logs the action taken for reference and may notify the user of the automatic adjustment.

Table 29:Use case to automatic window opening

ID:	**UC30**
Title:	Obstruction Detection for Automated Window Operation
Description:	This use case outlines the Smart Home Heating (SHH) system's capability to detect obstructions that prevent the safe opening or closing of automated windows. The focus is on ensuring that

	windows do not attempt to open or close if an object is blocking their path.
Primary Actor:	Smart Home System
Preconditions:	• The smart home system is operational, with windows equipped with automated opening and closing mechanisms. • Windows are integrated with sensors capable of detecting obstructions in their path. • A command to open or close a window has been issued, either automatically by the system or manually by the user.
Postconditions:	• The operation of opening or closing a window is stopped if an obstruction is detected.
Inputs:	• Command to open or close a window. • Sensor data regarding the presence of an obstruction in the window's path.
Outputs:	• Window is not closed if an obstruction is detected.
Main Success Scenario:	• A command is issued to open or close a window. • Prior to executing the command, the system checks for any obstructions using the window's integrated sensors. • If an obstruction is detected, the system aborts the window operation.

Table 30:Use case to Obstruction Detection for Automated Window Operatio

•

ID:	**UC31**
Title:	Notification on Failed Window Operation
Description:	This use case describes the process by which the Smart Home Heating (SHH) system notifies users when it is unable to execute a command related to window operation, such as opening or closing a window, due to an obstruction or any other issue that prevents the action from being completed.
Primary Actor:	Smart Home System
Preconditions:	• The smart home system is operational and is integrated with window control mechanisms.

	● The user has attempted to initiate a window operation (open/close) either manually or through automated system commands.
Postconditions:	● The user is informed via a notification that the system cannot complete the window operation.
Inputs:	● Command from the user or system to open/close a window. ● Feedback from the window mechanism or sensors indicating a failure to execute the command.
Outputs:	● A notification sent to the user detailing the reason why the window operation could not be executed.
Main Success Scenario:	● The system receives a command to open or close a window. ● The system attempts to execute the command. ● The system encounters an issue (e.g., an obstruction, or mechanical failure) that prevents the operation from being completed. ● The system generates a notification detailing the issue and sends it to the user. ● The user receives the notification, informing them of the failed operation and the reason.

Table 31:Use case to send notification on failed window operation

ID:	**UC32**
Title:	Temperature Adjustment for Energy Savings
Description:	This use case ensures that the Smart Home Heating (SHH) system detects when the house is unoccupied (i.e., all residents have left) and automatically lowers the temperature setting to 17 degrees Celsius during the winter season. This feature aims to optimize energy savings by reducing heating in an empty house while maintaining a baseline temperature.
Primary Actor:	Smart Home System
Preconditions:	● The SHH system is operational and has been configured to manage heating settings. ● The system has access to sensors or devices capable of detecting when residents have left the house (e.g., door sensors, geolocation from smartphones).

	● It is winter, as determined by the system's internal calendar or the geographic location's climate patterns.
Postconditions:	● The heating setting is automatically lowered to 17 degrees Celsius when the house is detected to be unoccupied and is restored to the regular setting when occupancy is detected again.
Inputs:	● Signals from sensors or devices indicating the presence or absence of residents. ● Current indoor temperature setting. ● External temperature or season confirmation to ensure it is winter.
Outputs:	● Adjustment of the heating system's temperature setting to 17 degrees Celsius upon detecting absence. ● Notification to users about the temperature adjustment for transparency and manual override if needed.
Main Success Scenario:	● The system continuously monitors the inputs from sensors or devices to determine the occupancy status of the house. ● Upon detecting that the house has become unoccupied, the system checks if it is winter. ● If it is winter, the system automatically lowers the heating setting to 17 degrees Celsius. ● The system sends a notification to the users, informing them of the automatic temperature adjustment. ● Upon detecting that residents have returned, the system restores the temperature to the regular setting.

Table 32:Use case for temperature adjustment for energy savings

6.2 Use case diagram for the whole system **(Nothing changed for Phase 3)**

In the system, there are various roles including Parent, Child, Guest, and Stranger. Each of these roles interacts with different components of the smart home system.

The "Smart Home Simulator (SHS)" is a central component, which includes the "Home Layout". It likely allows users to simulate various smart home scenarios. This simulator can be influenced by "Simulation parameters" and the "Context of the simulation", suggesting that the system can adjust simulations based on specific criteria or conditions.

There are also modules for "Smart Heating (SHH)" and "Smart Home Security (SHP)", indicating specialized functions within the smart home system for controlling heating and managing security, respectively. These modules can probably be adjusted or monitored through the simulation parameters and context.

The "Smart home core functionality (SHC) module" is another essential part of the system, which likely includes the basic functions required for operating the devices in the house.

Lastly, there is a "Smart home dashboard", which is probably a user interface where the different roles (Parent, Child, Guest, and Stranger) can interact with and control the smart home functionalities.

In summary, the use case diagram outlines how different users interact with a smart home system that is capable of simulating and controlling various aspects such as heating, security, and other core functionalities through a centralized dashboard.

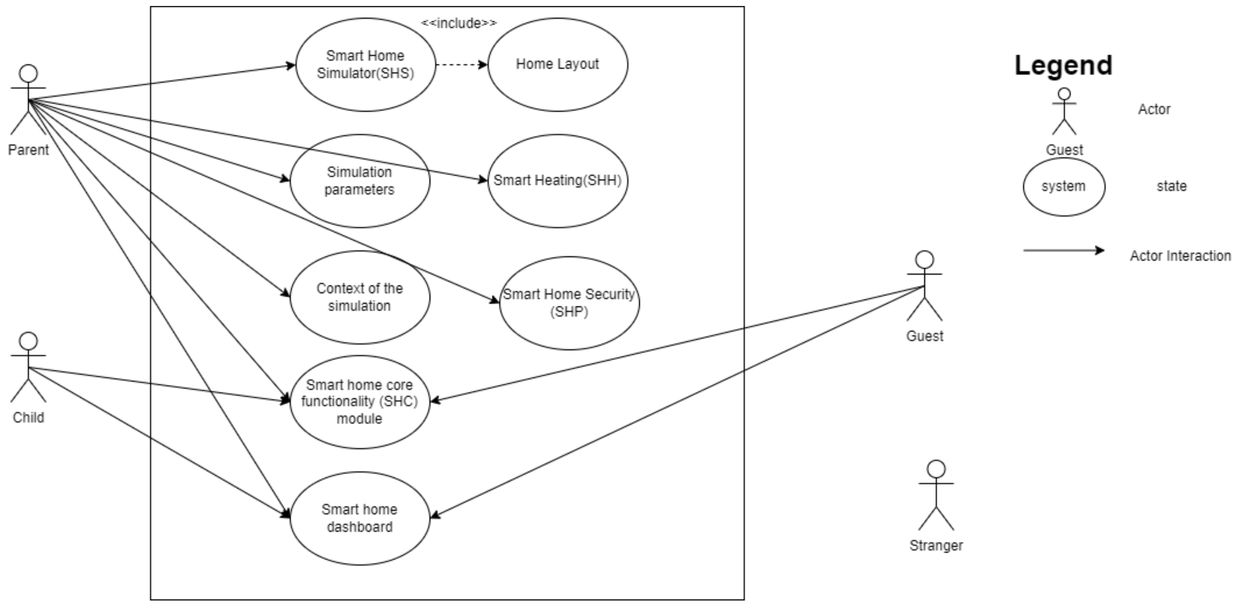


Figure 4. System Use case Diagram

6.3 Sequence Diagram **(Nothing changed for Phase 3)**

For this following section, we have picked one of the granular use cases, in particular the use case titled “System shall allow users to close doors”, to elaborate it further revealing its inner workings through constructing a sequence diagram for it.

As documented in the sequence diagram, the request is sent by the User object and received by the Dashboard object initially. The Dashboard object pipelines the request down to the module responsible for this functionality, in this case the Smart Home Core Functionality. The SHC module finds the door concerned with this request and gets its status. If the door is closed it returns back to the Dashboard object that the door is closed, else if it's open it will close the door and return a confirmation.

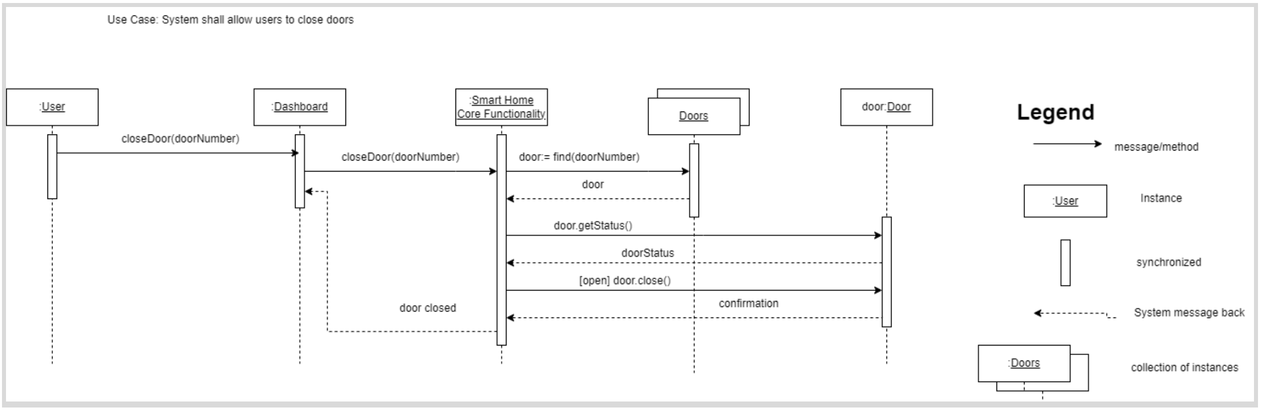


Figure 5. System sequence Diagram

6.4 State-machine diagram for the system (SHH Diagram is added)

The following State Machine Diagram provides a concise representation of the dynamic behaviour of the Smart Home simulator system. It outlines the various states and transitions within the system, offering users a clear understanding of how the simulator operates in response to different inputs and scenarios.

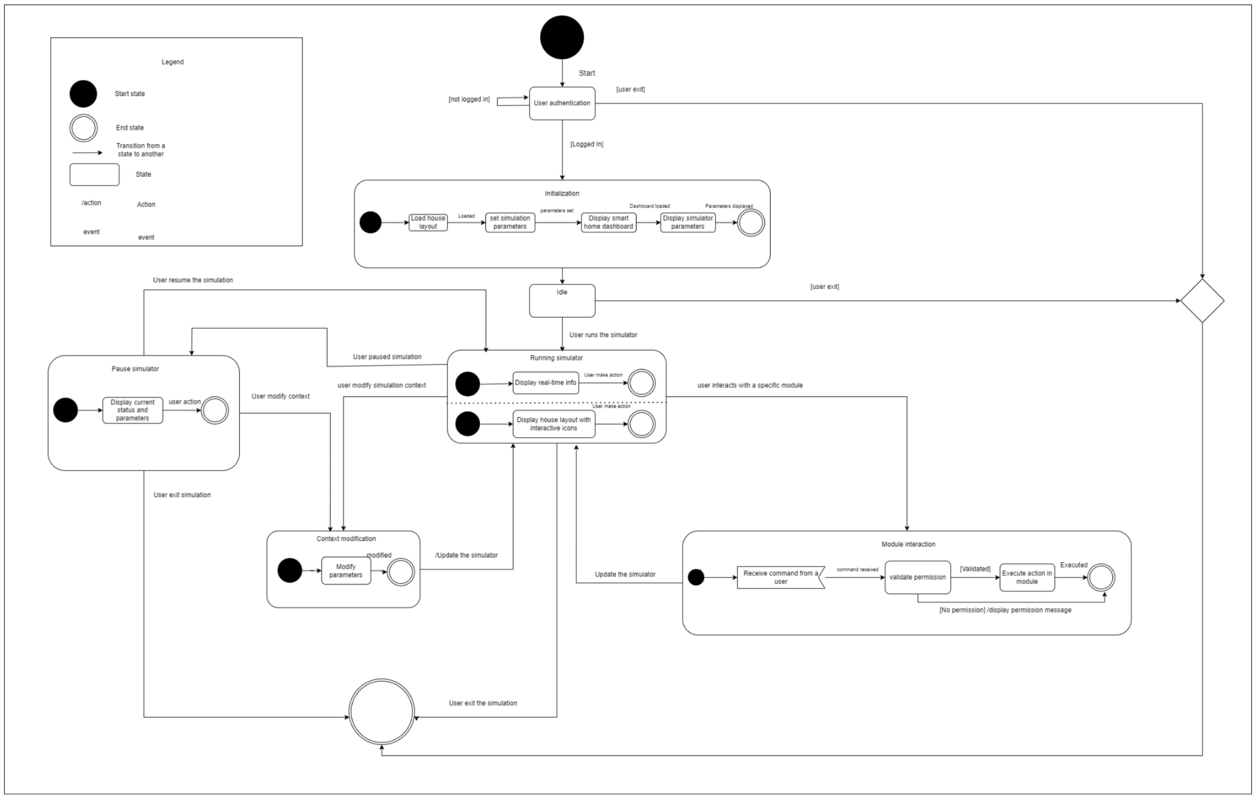


Figure 6. State machine Diagram for the whole system

1) **Authentication state:**
- This is the initial state when the simulator is launched.
- Has the user log-in/sign-ups

2) **Initialization State:**
- Load the house layout file.
- Set simulation parameters such as date, time, weather, and user profiles.
- Display the smart home dashboard.
- Display the parameter.

3) **Idle State:**
- In this state, the simulation is ready to be started.

4) **Simulation Running State:**
- Display simulation parameters and current status.

- Allow the user to modify the simulation context if needed by transitioning to the context state.
- Display real-time information such as temperature, date, time, and logged-in user.
- Display the house layout with interactive icons representing the actions and statuses of various devices.
- Listen for user interactions and commands from smart home modules.

5) Context Modification State:
- This state is entered when the user decides to modify the simulation context during runtime.
- Allow the user to modify parameters such as date, time, user location, weather, etc.
- Update the simulation accordingly.

6) Module Interaction State:
- Entered when a user interacts with a specific smart home module.
- Receive commands from the user or other modules.
- Validate permissions for the requested actions.
- Execute actions such as opening/closing windows, turning on/off lights, setting temperatures, etc.

7) Simulation Paused State:
- Entered when the user pauses the simulation.
- Display current simulation status and parameters.
- Allow the user to resume, modify context, or exit the simulation.

8) Final state when the user exits the simulator.
- Close all connections and release resources.
- Terminate the application.

Transitions between states are triggered by user actions, system events, or internal conditions. For example, from the Idle State, the system transitions to the Simulation Running State when the user starts the simulation. Similarly, from the Simulation Running State, the system can transition to the Context Modification State when the user decides to modify simulation parameters. These transitions ensure the system behaves dynamically based on user input and simulation events.

6.4.1 State-machine diagram for the SHH Module

The following State Machine Diagram provides a concise representation of the dynamic behaviour of the Smart home heating Module. It outlines the various states and transitions within the system, offering users a clear understanding of how the SHH operates in response to different inputs and scenarios.

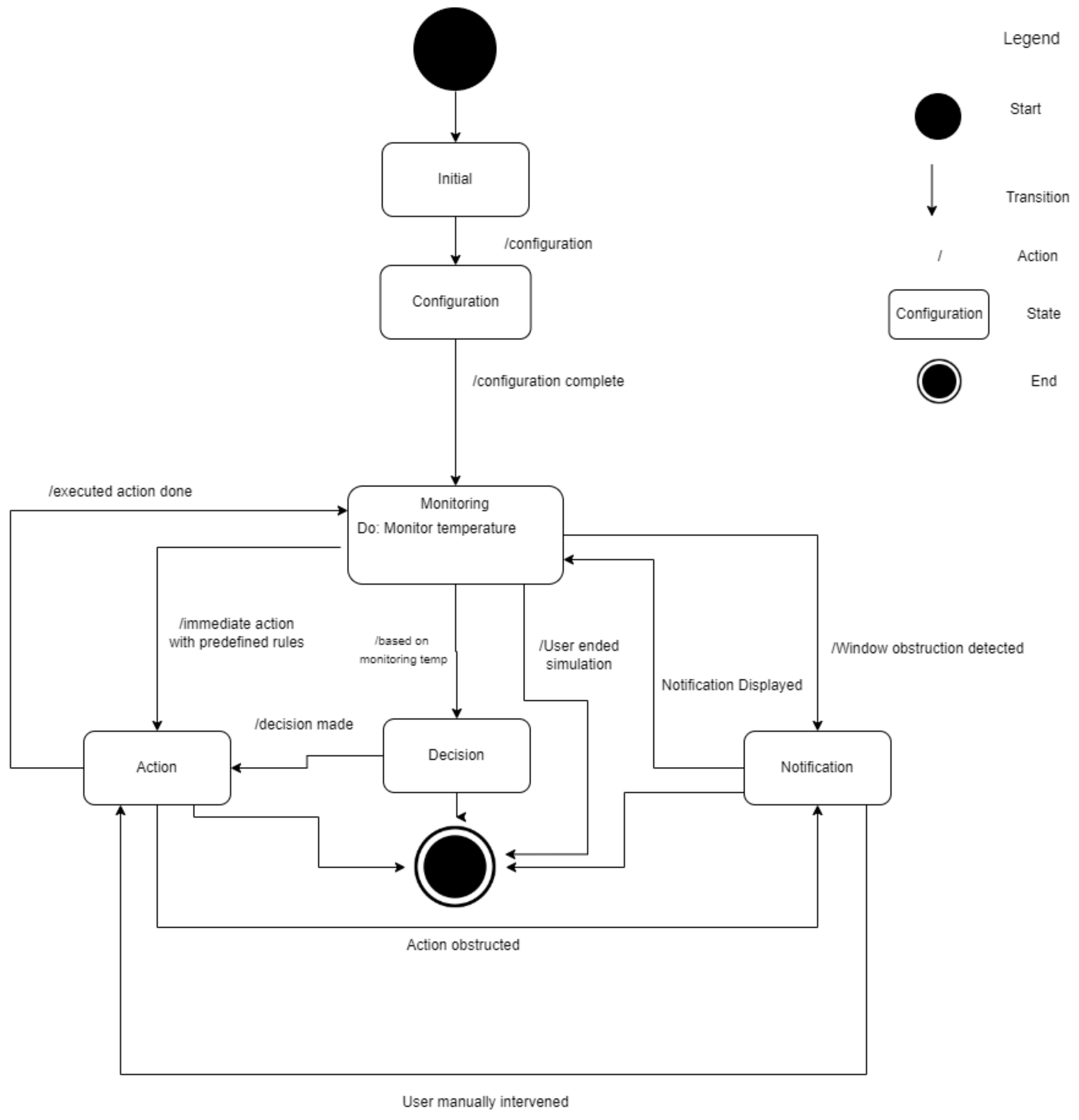


Figure 7. State machine Diagram for the Smart Home Heating

1) **Initial state:**
    - The SHH module is either in a powered-off state or awaiting input to configure zones and desired temperatures..

2) **Configuration State:**

- In this state, zones are defined, and desired temperature settings for each zone for up to 3 periods in the day are set.

3) **Monitoring State:**
    - The SHH module actively monitors indoor and outdoor temperatures to make decisions on heating/cooling actions.

4) **Decision State:**
    - Based on the current temperatures and the configured settings, decisions are made on whether to heat or cool the zones, open/close windows, or send notifications about obstructions or temperature anomalies.

5) **Action State:**
    - Actions are taken to adjust the indoor environment. This includes turning on/off the heating or cooling, opening or closing windows (if not obstructed), and overriding zone temperatures if specified by the user.

6) **Module Interaction State:**
    - Entered when a user interacts with a specific smart home module.
    - Receive commands from the user or other modules.
    - Validate permissions for the requested actions.
    - Execute actions such as opening/closing windows, turning on/off lights, setting temperatures, etc.

7) **Notification State:**
    - If the system encounters an obstruction when trying to open/close a window or if the outdoor temperature conditions warrant a notification (e.g., potential for pipes to freeze), notifications are sent to the users..

8) **Final state when the user exits the simulator.**
    - The system returns to the Monitoring State to continuously evaluate the environment and adjust as needed or remains in a passive state until reactivated or reconfigured.

State transitions are initiated by user interactions, system occurrences, or predefined conditions. For instance, moving from the Idle State to the Simulation Running State occurs when the user initiates the simulation. Likewise, the shift from the Simulation Running State to the Context Modification State happens upon the user's adjustment of simulation settings. These state changes allow the system to adapt and respond to user actions and simulation developments dynamically.

6.5 Activity Diagram for Context of the Simulation

The following diagram shows the flow for the Context of the Simulation, highlighting the main components and describing the dynamic process throughout the whole simulation lifecycle.

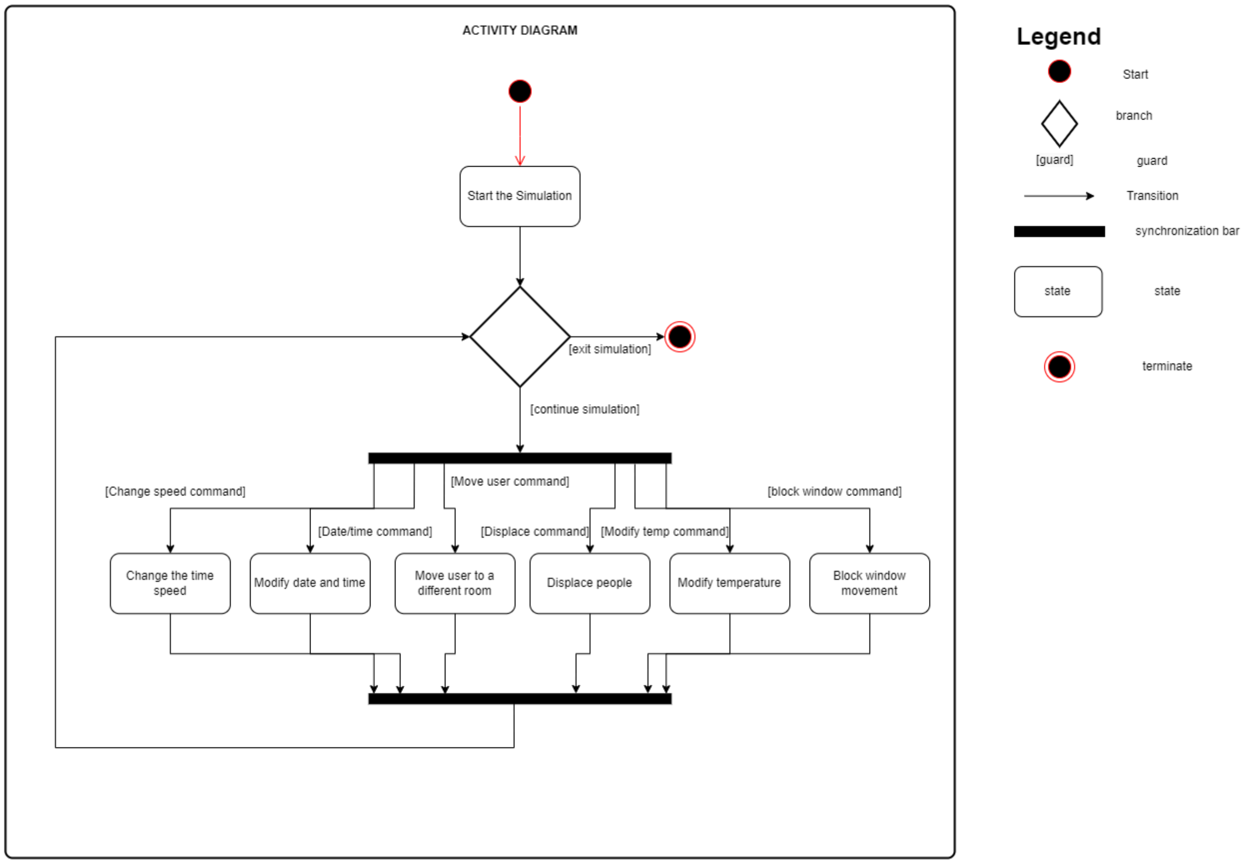


Figure 8. Activity Diagram for context of simulator

1) **Start of the Simulation:**
   This denotes the beginning of the simulation and opens up all the commands to the user, including the option to not run any commands and exit the simulation.The fork shown describes that any of the following commands can be run in no specific order.

2) **Change speed command:**
   This command allows the user to change the speed at which time passes by.

3) **Modify date and time:**
   This command allows the user to modify the date and the time of the day to simulate different situations.

4) **Move user to a different room:**
   This command allows the user to move himself in a different room.

5) **Displace people:**
   This command allows the user to move people in different rooms or outside the home.

6) **Modify temperature:**
   This command allows the user to modify the temperature in the simulation.

7) **Block window movement:**
   This command allows the user to stop window movement by placing a random object on a window.

The natural continuation of this activity shows us that after each command, the user can choose to exit the simulation or run another command.

6.5.1 Activity Diagram for SHH Module

The following activity diagram shows the flow for the SHH module, highlighting the main components and the flow of activities. Each component is described below the figure.

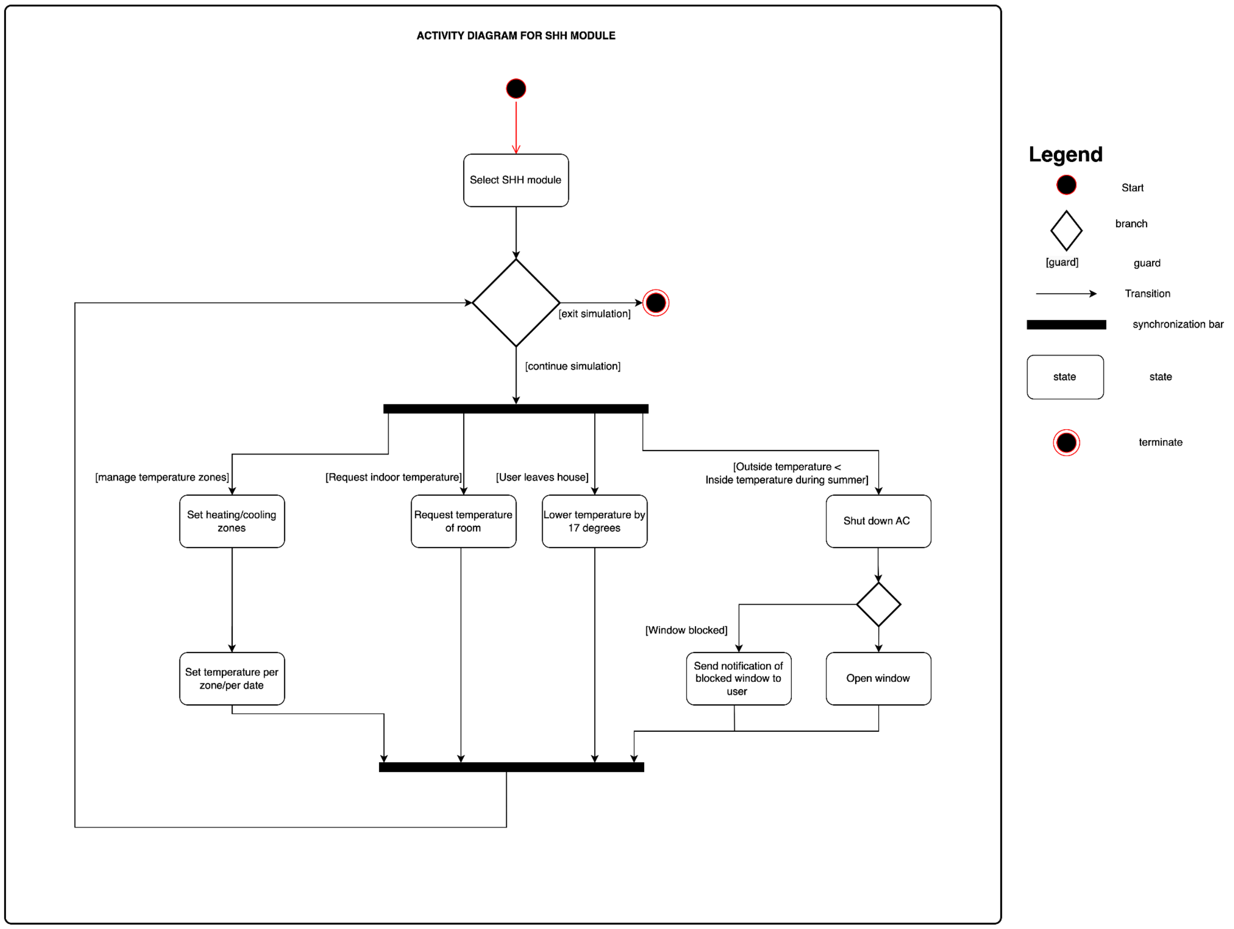

ACTIVITY DIAGRAM FOR SHH MODULE
Select SHH module
[exit simulation]
[continue simulation]
[manage temperature zones]
[Request indoor temperature]
[User leaves house]
[Outside temperature < Inside temperature during summer]
Set heating/cooling zones
Request temperature of room
Lower temperature by 17 degrees
Shut down AC
[Window blocked]
Set temperature per zone/per date
Send notification of blocked window to user
Open window
Legend
Start
branch
[guard]
guard
Transition
synchronization bar
state
state
terminate

Figure 9. Activity Diagram for SHH

1) Selecting SHH module

This denotes the beginning of managing all the features found within the SHH module, providing all necessary commands to the user. Given that there are multiple available commands, the fork shown describes the selection of commands to choose from, regardless of the order.

2) Set heating/ cooling zones

This command allows to set zones all across the household, where the zones may encompass multiple rooms. Following the selection of zones, according temperatures must then be set for each of the defined zones.

3) Set temperature per zone/per date
This command is followed up after setting the zones; users set the temperatures based on the zone and the date.

4) Request temperature of room
This command displays the temperature of all rooms.

5) Lower temperature by 17 degrees
This command lowers the temperature of the house when the user leaves the house, which leads to major energy savings.

6) Shut down AC
This command shuts the AC off when the outside temperature is lower than the inside temperature during summer. Following this command, the SHH module will either open the windows or fail, sending a notification to the user.

7) Open window
This command is followed up after shutting down the AC; windows can be opened without any disruption.

8) Send Notification to user
This command is followed up after shutting down the AC; windows cannot be opened due to some obstruction. This leads to the SHH modules sending a notification to the user indicating that the window is blocked.

7 Design Patterns (Added observer for SHH)

This section discusses the application of observer, command, and singleton design patterns in the smart home simulator. The observer pattern tracks environmental changes, the command pattern manages actions like opening and closing doors, and the singleton pattern ensures a single dashboard instance. UML class diagrams illustrate the structure and interactions of these patterns, enhancing the simulator's functionality and design clarity.

7.1 Design Patterns Used

For the smart home simulator, we decided we will use observer, command, and singleton patterns. The observer design pattern used for the smart home simulator(SHS) module is observable and is responsible for providing an API where smart home modules can be subscribed to track environmental conditions like temperature inside and outside, date and time and they get notified if any change happens. The command design pattern used in Smart Home Core Functionality (SHC) has a list of commands (open/close window(s), open/close the main door/ backyard door, garage, etc.). A singleton design pattern is used for the dashboard since we only need one instance for it and all the modules use this single instance.

7.2 Design pattern UML class diagram implementation

Observer design pattern:

- Subject Interface: Contains methods for adding and removing observers and a method to notify them.
- Smart Home Simulator: Implements the Subject interface, holds observer modules, and includes attributes for home layout, date, indoor and outdoor temperature, devices, and sensors.
- Observer Interface: Defines an "update" method for observers to implement.
- Dashboard: Acts as an observer, which can display objects and update the smart home simulator state.
- Smart Home Modules: Represent concrete observers that implement the "update" method.

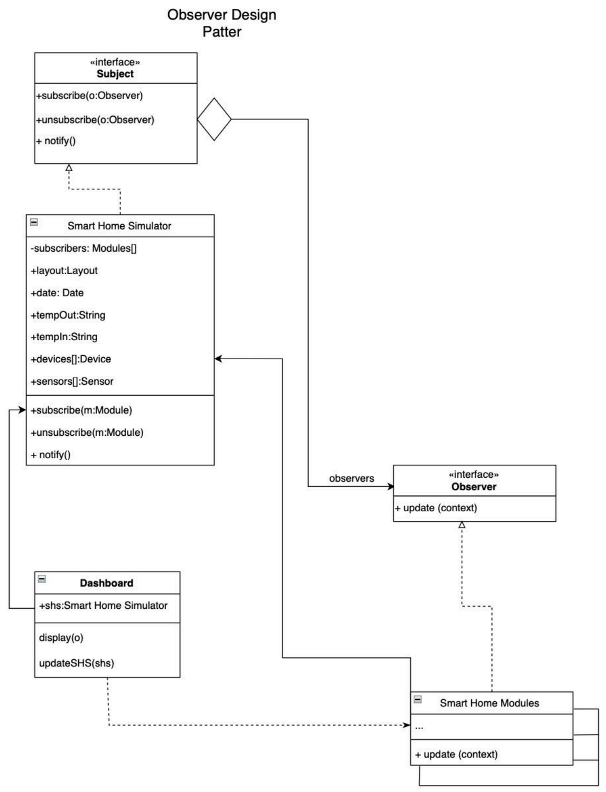


Figure 10. Observer design pattern

Command design pattern:

- Command Interface: An interface "Command" defines the "execute" method.
- Concrete Command Classes: There are several concrete classes implementing the "Command" interface, such as "DoorCloseCommand", "LightOnCommand", "WindowOpenCommand", etc., each with an "execute()" method.
- Receiver Classes: Corresponding "Receiver" classes like "DoorReceiver", "LightReceiver", "WindowReceiver", etc., have specific operations that are invoked by the command's "execute()" method.
- Invoker Class: A class "SHCInvoker" has a method to execute commands.
- Client: Represents the entity that creates a command and sets its receiver.
- Interaction: The sequence diagram shows the flow of execution from the client setting up commands, passing them to the invoker, and the invoker executing these commands which then call operations on the receivers.

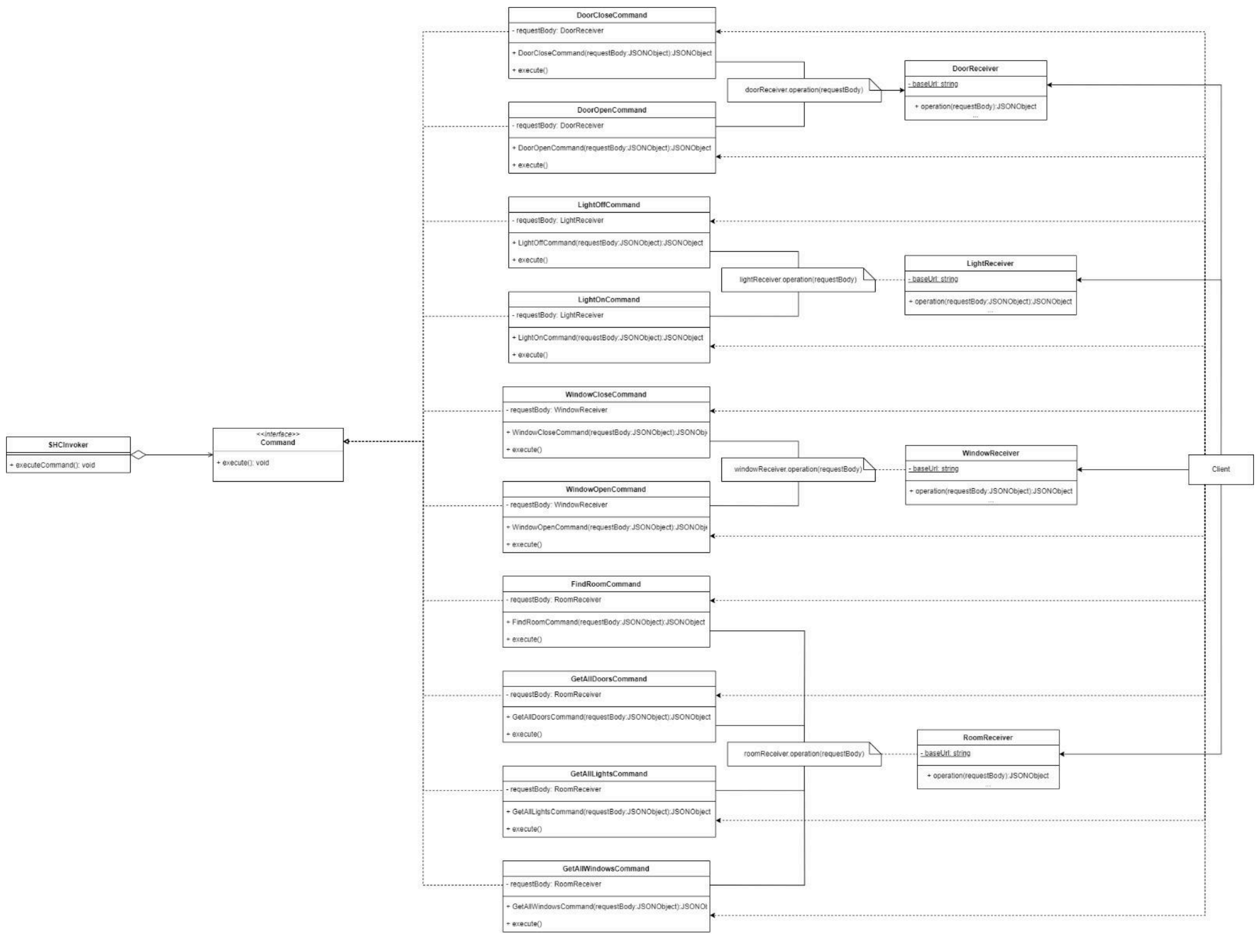

DoorCloseCommand
- requestBody: DoorReceiver
+ DoorCloseCommand(requestBody:JSONObject):JSONObject
+ execute()
DoorOpenCommand
- requestBody: DoorReceiver
+ DoorOpenCommand(requestBody:JSONObject):JSONObject
+ execute()
doorReceiver.operation(requestBody)
DoorReceiver
- baseUrl: string
+ operation(requestBody):JSONObject
LightOffCommand
- requestBody: LightReceiver
+ LightOffCommand(requestBody:JSONObject):JSONObject
+ execute()
LightOnCommand
- requestBody: LightReceiver
+ LightOnCommand(requestBody:JSONObject):JSONObject
+ execute()
lightReceiver.operation(requestBody)
LightReceiver
- baseUrl: string
+ operation(requestBody:JSONObject):JSONObject
WindowCloseCommand
- requestBody: WindowReceiver
+ WindowCloseCommand(requestBody:JSONObject):JSONObj
+ execute()
WindowOpenCommand
- requestBody: WindowReceiver
+ WindowOpenCommand(requestBody:JSONObject):JSONObj
+ execute()
windowReceiver.operation(requestBody)
WindowReceiver
- baseUrl: string
+ operation(requestBody:JSONObject):JSONObject
SHCInvoker
+ executeCommand(): void
<<interface>>
Command
+ execute(): void
Client
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):JSONOt
+ execute()

Figure 11. Command design pattern

SIngleton design pattern:

- A "User" : can interact with the "Dashboard" class.
- "Dashboard": has a private instance and constructor, ensuring only one instance is created.

- The "getDashboard()" method: provides access to the Singleton instance of "Dashboard".

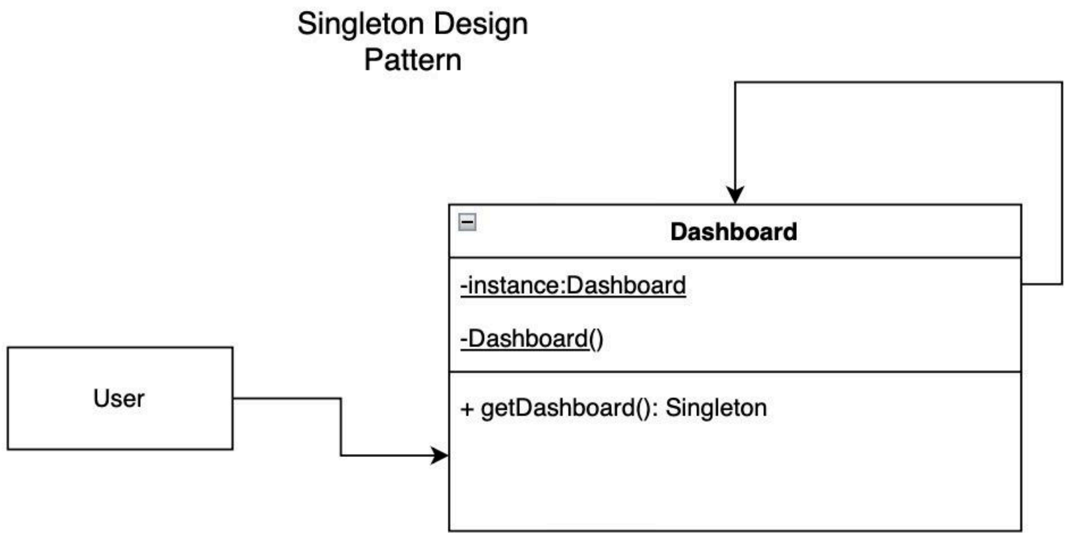


Figure 12. Singleton design pattern

2nd Observer Pattern for SHH:

- The decision to implement the Smart Home Heater (SHH) as an observer and the Smart Home Simulator (SHS) as the subject is aligned with the Observer pattern's principles, which facilitate a responsive and maintainable system.
- In this setup, SHH automatically receives updates from SHS when changes occur, allowing it to adjust the heating accordingly without tight coupling between components.
- This design supports easy scalability and modularity in the smart home ecosystem.

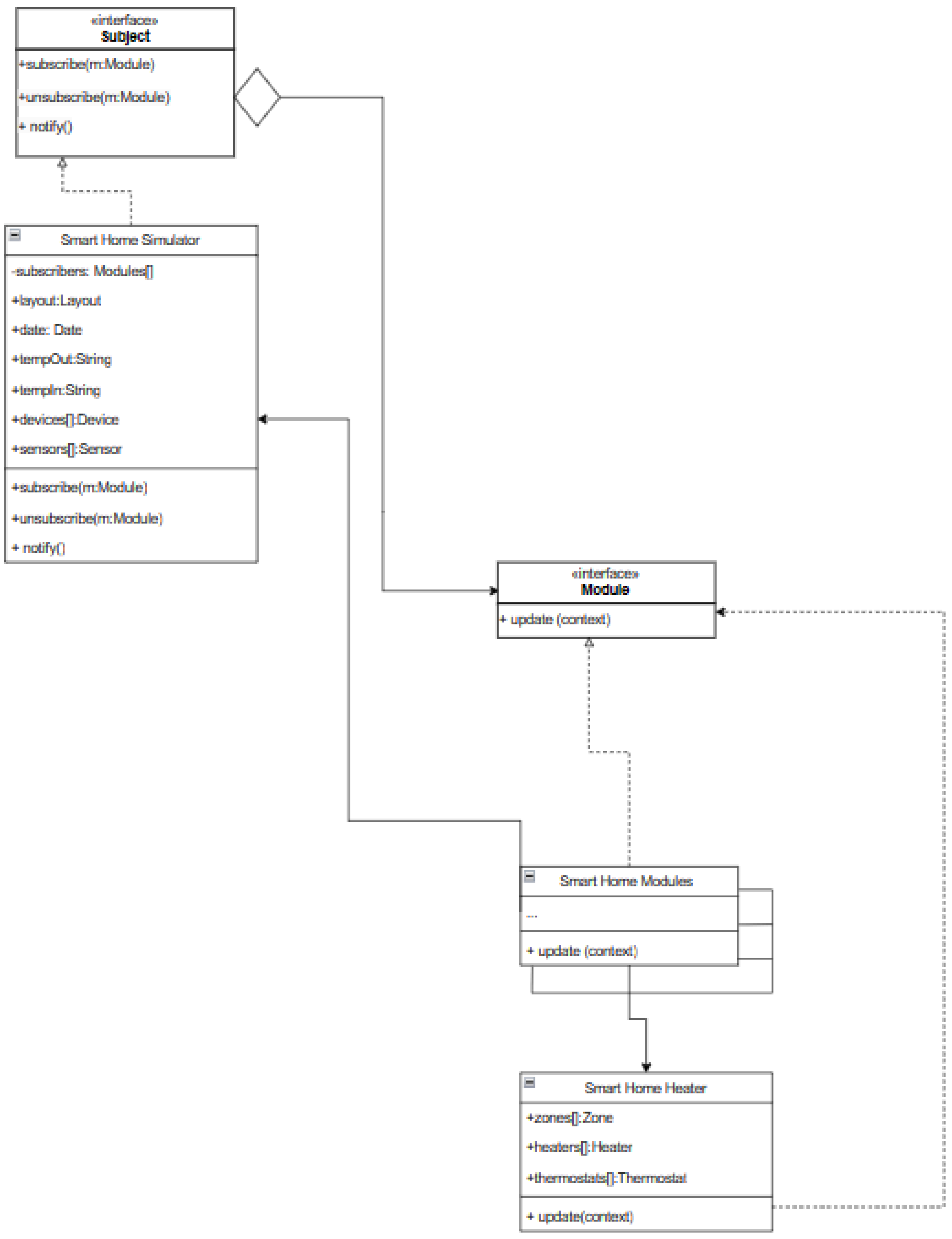
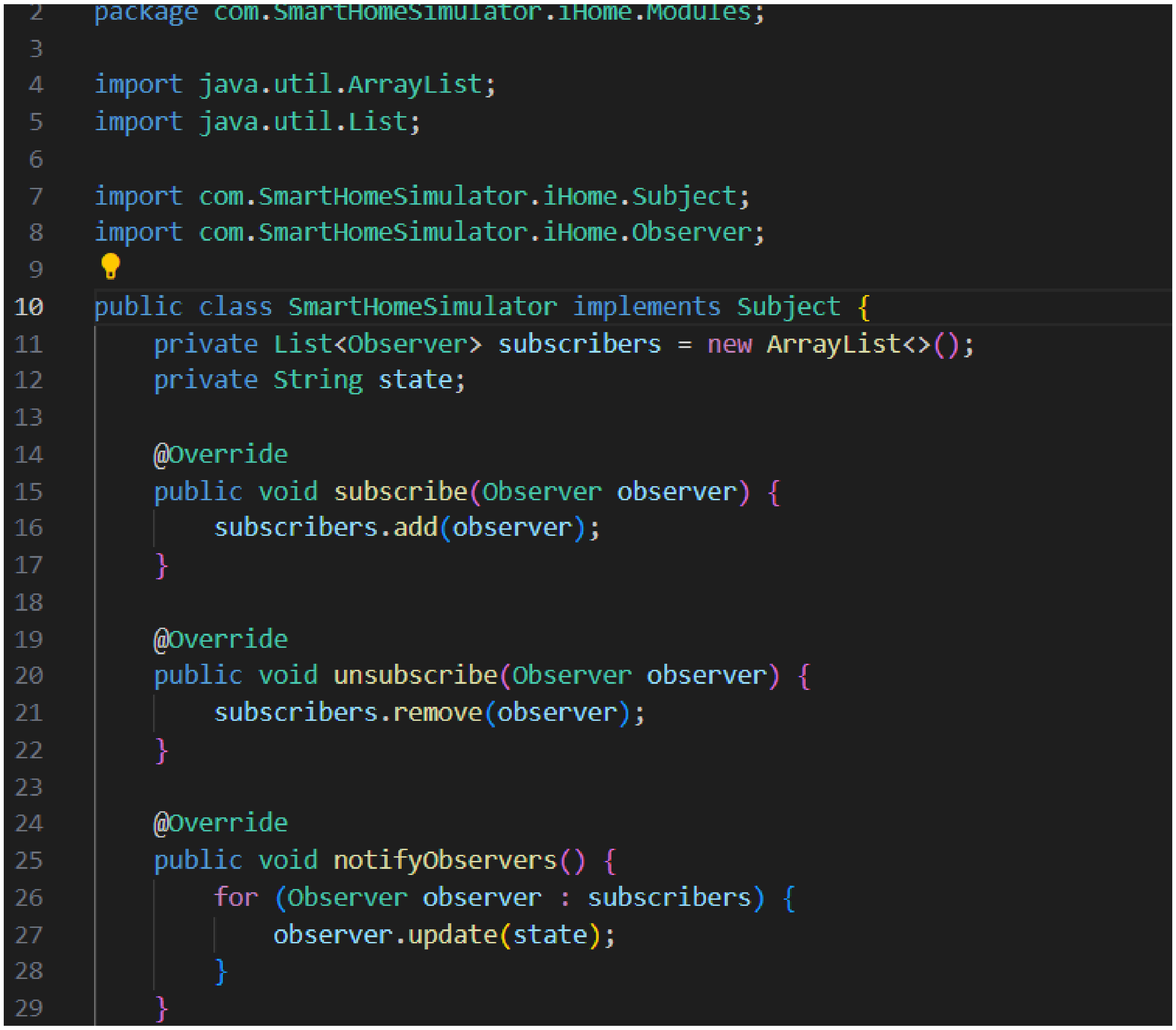


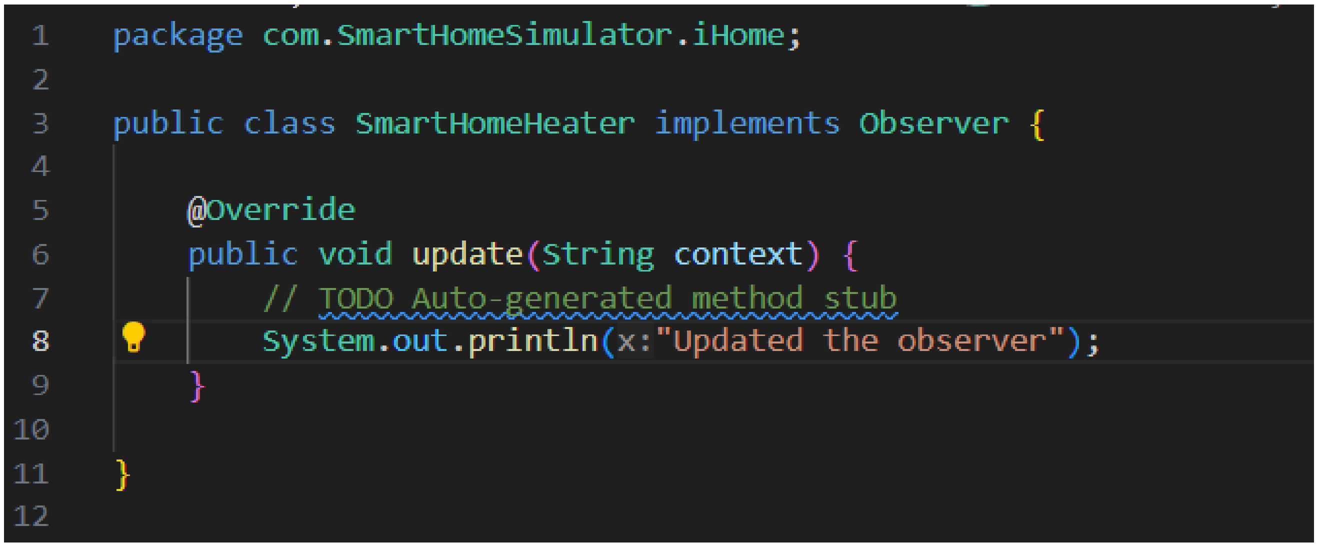
Figure 13. Observer design pattern for SHH

Snippets of code for the applied observer pattern:



```java
package com.SmartHomeSimulator.iHome.Modules;

import java.util.ArrayList;
import java.util.List;

import com.SmartHomeSimulator.iHome.Subject;
import com.SmartHomeSimulator.iHome.Observer;

public class SmartHomeSimulator implements Subject {
    private List<Observer> subscribers = new ArrayList<>();
    private String state;

    @Override
    public void subscribe(Observer observer) {
        subscribers.add(observer);
    }

    @Override
    public void unsubscribe(Observer observer) {
        subscribers.remove(observer);
    }

    @Override
    public void notifyObservers() {
        for (Observer observer : subscribers) {
            observer.update(state);
        }
    }
```

Figure 14. Snippet code SHS for observer pattern



```java
package com.SmartHomeSimulator.iHome;

public class SmartHomeHeater implements Observer {

    @Override
    public void update(String context) {
        // TODO Auto-generated method stub
        System.out.println(x:"Updated the observer");
    }

}
```

Figure 15. Snippet code SHH for observer pattern