Throughout the course of this module, I embarked on a journey that expanded my understanding of these complex subjects. It was an intensive learning experience that challenged my knowledge and skills. In this reflective piece, I will detail my journey through the module, my learning experience, the challenges I encountered, and my role as a part of a development team for the team project. In general, my reflective piece follows the guidelines of Kolb's learning theory (Kolb, 1984).

The module commenced with an introduction to secure systems architecture (SSA), which laid the foundation for our exploration of the intricacies of operating systems and distributed systems. I found this introductory unit enlightening, as it provided insights into the historical evolution of these systems and their relevance in the present world. It was interesting to learn how the demands of users have driven changes in these systems over time. This historical perspective was critical in helping me appreciate the context in which these systems operate.

As we progressed into the heart of the module, I was tasked with understanding the principles and techniques of secure coding practices. This was one of the most challenging yet rewarding parts of the module. I realized the critical importance of coding securely, especially in an era where cybersecurity is paramount. Additionally, adapting platforms and systems using code refactoring for security enhancement was a valuable skill that I developed. This unit instilled in me the notion that security should be an integral part of the development process from the beginning.

Unit 3 introduced the concept of modelling and system design. It was intriguing to explore the differences between SysML and UML. I found it critical to understand the role of modelling in systems engineering and how it can aid in solving complex problems. Learning about NP-hard problems was intellectually stimulating, and it pushed me to think about the limitations of what can be achieved in computing. Threat modelling was another highlight of this unit, as it helped us anticipate and address security concerns effectively.

The subsequent unit delved into system modelling tools and discussed the relevance of scientific methods in cybersecurity. I appreciated the focus on the scientific approach to solving cybersecurity problems. The unit reinforced the idea that SSA is a dynamic field, and continuous learning is essential.

Unit 5 dealt with the current and future challenges of operating and distributed systems and provided me with a deep appreciation for the complexities and scale of operations in the modern world. It made me realize the significance of staying informed about the latest developments and research in the field to address these challenges effectively.

The most demanding and rewarding part of this module was our team project, which spanned Units 3 to 6, and brought together essentially all of the information we had been given so far, and required further research and skills development, which I personally found to be very challenging but rewarding. Working as a part of a development team allowed me to apply the knowledge and skills I gained throughout the module in a practical context. Our project involved designing and developing a simulated distributed home IoT system, and I was tasked with implementing the simulated IoT devices.

I quickly found that it was challenging to design the devices in complete isolation of the server component that would be communicating with the devices. I went through about seven iterations of code, each time reusing some aspects of my previous iteration and completely redoing others, and each time getting to a better and more elegant solution. Ultimately, I managed to develop a very substantial fully-fledged prototype with two major components:

1. A device service that simulates one of four IoT devices (Motion Sensor, SmartLock, SmartLight, and Thermostat). The service listens for incoming requests, such as readings activation/deactivation, on/off, change of threshold value, disconnection etc.
2. A HUB service starts up and manages connections to running device services, and acts as a command-line user interface which allows the user to do a variety of actions (as mentioned in point 1 above) either on a specific device or on all devices.

This formed the core of functionality in the system. Screenshots of my original contribution are provided below in the Appendix to this reflective piece.

Our testing process involved both manual and automated testing. The manual testing allowed us to evaluate the user interface and functionality from a user's perspective. Automated testing included the use of unit tests to confirm code correctness, in addition to using pylint to improve on the code quality, and using online tools to test the hypothesis. Such rigorous security and functional tests would have been impractical to execute manually.

I think the biggest challenge that faced me personally was carrying out group work with team members in various time zones and locations. Ultimately, however, these challenges are true to everyday life and they helped me grow immensely.

Finally, I wanted to deeply commend and thank our lecturer who patiently, passionately and diligently engaged with us to convey and share his profound knowledge with us. Looking forward to the next module.

References:

Kolb, D.A., 1984. Experiential learning: Experience as the source of learning and development. Prentice Hall.

Appendix – Screenshots of my Contribution to the Project:

```
PS [redacted]SSA\Project\smart-home> python .\deviceservice.py
IoT Device Simulator.
The server username and password are hard-coded in this simulation to avoid making you have to type them over and over.
There is also a pre-defined list of simulated devices that you select from for convenience

Here's the pre-defined list of devices to link into the home:
0: SmartLight| {"identifier": "Light1", "status": "active", "threshold": 50, "switch": "on", "brightness": 90}
1: SmartLight| {"identifier": "Light2", "status": "active", "threshold": 30, "switch": "on", "brightness": 90}
2: SmartLight| {"identifier": "Light3", "status": "active", "threshold": 80, "switch": "on", "brightness": 90}
3: MotionSensor| {"identifier": "Motion1", "status": "active", "threshold": 5, "switch": "on", "motion": 0}
4: MotionSensor| {"identifier": "Motion2", "status": "active", "threshold": 8, "switch": "on", "motion": 0}
5: MotionSensor| {"identifier": "Motion3", "status": "active", "threshold": 3, "switch": "on", "motion": 0}
6: SmartLock| {"identifier": "Lock1", "status": "active", "threshold": 0, "switch": "on"}
7: SmartLock| {"identifier": "Lock2", "status": "active", "threshold": 0, "switch": "on"}
8: SmartLock| {"identifier": "Lock3", "status": "active", "threshold": 0, "switch": "on"}
9: Thermostat| {"identifier": "Therm1", "status": "active", "threshold": 23, "switch": "on", "temp": 15.0}
10: Thermostat| {"identifier": "Therm2", "status": "active", "threshold": 30, "switch": "on", "temp": 15.0}
11: Thermostat| {"identifier": "Therm3", "status": "active", "threshold": 15, "switch": "on", "temp": 15.0}
Which device should this program instance simulate?:
```

Figure 1: Initial list of pre-defined devices the user can choose to simulate. Note that these devices can be added to very easily.

```
PS [redacted]\SSA\Project\smart-home> python
 .\hub.py
Simulated IoT Device Controller HUB
This device listens for connection requests by smart IoT devices
It 'registers' these devices once they request to connect the first time
It also allows the user to get access to the device functions



 Also, usually the protocol would require exchange of public keys for secure
 communications.In this demo, we are assuming this has been done and the pub
lic keys have been generated and exchanged.
For demo purposes, the hub and all devices public keys have been generated u
sing the './Inits/initializeAll.py' script.
They will be loaded from file by hub and devices and used for communications



HUB Interface Menu:
0: List all registered devices
1: List connected devices
2: Get device readings
3: Set device threshold
4: Activate device
5: Deactivate device
6: Switch on device
7: Switch off device
8: Disconnect device from HUB
9: Quit

Enter your choice:
```

Figure 2: Initial HUB server screen.

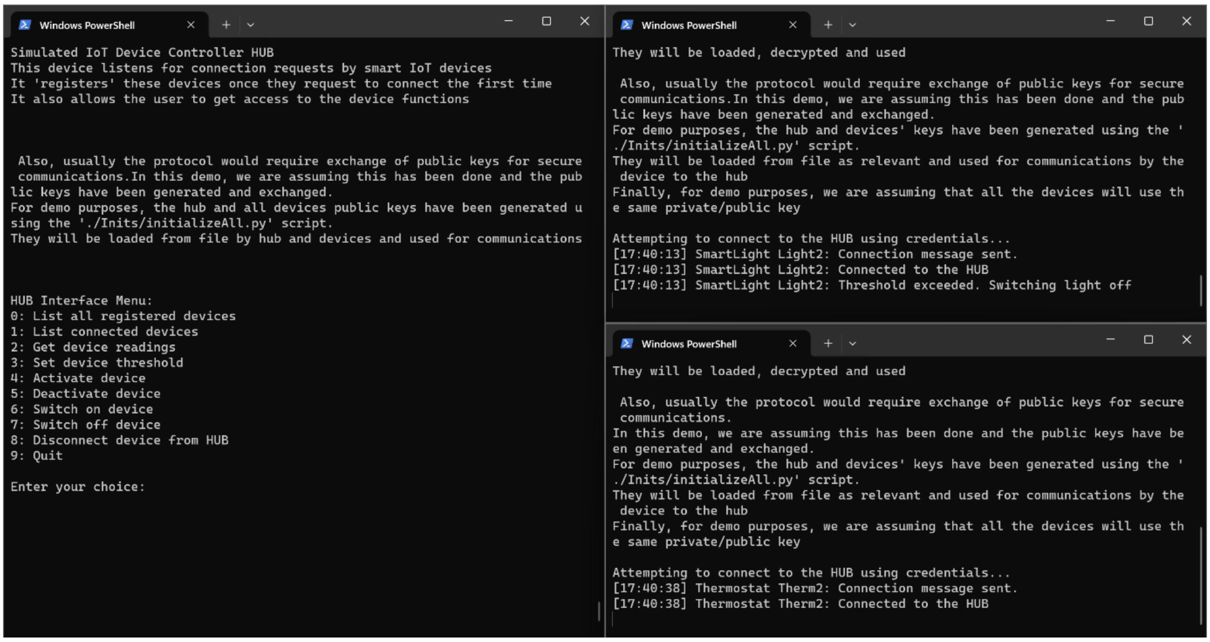

Figure 3:(Left) HUB server waiting for user commands; (Top-Right) Simulated Smart Light device waiting for requests from the HUB – also note that the simulated automatic sensor has detected the brightness exceeding the threshold thereby switching the light off; (Bottom-Right) Simulated Thermostat device waiting for requests from the HUB.

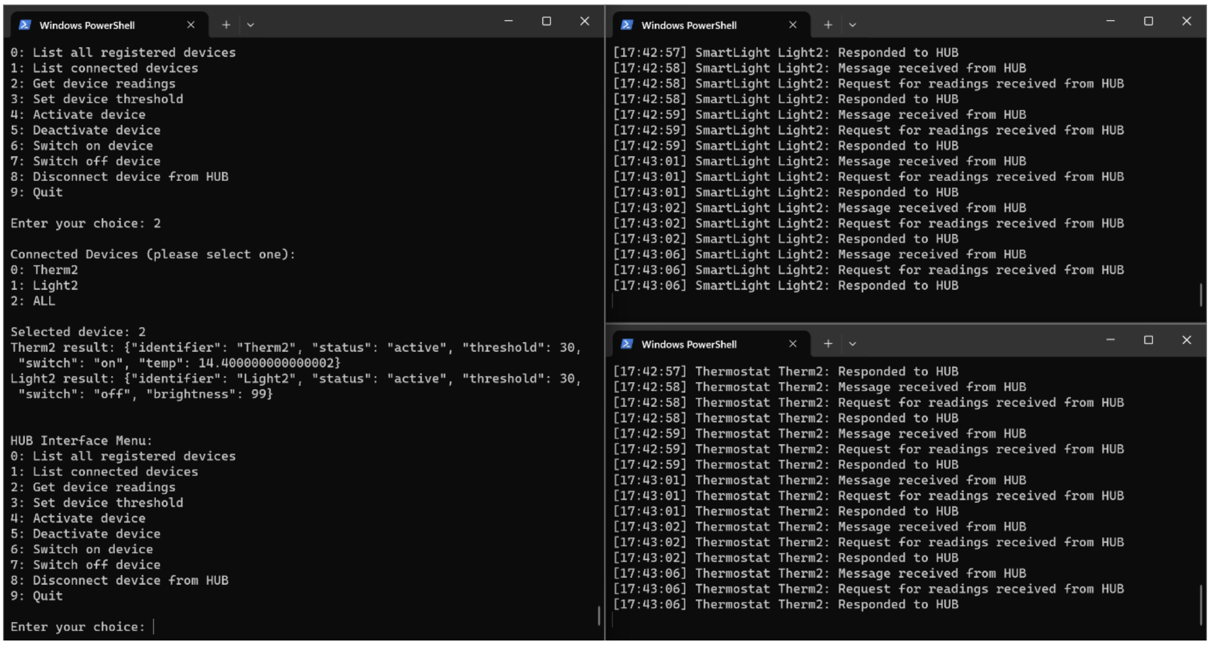

Figure 4:(Left) HUB server initiated a "Get device readings” and specified “ALL” devices; (Right) Devices received the request and responded with readings.

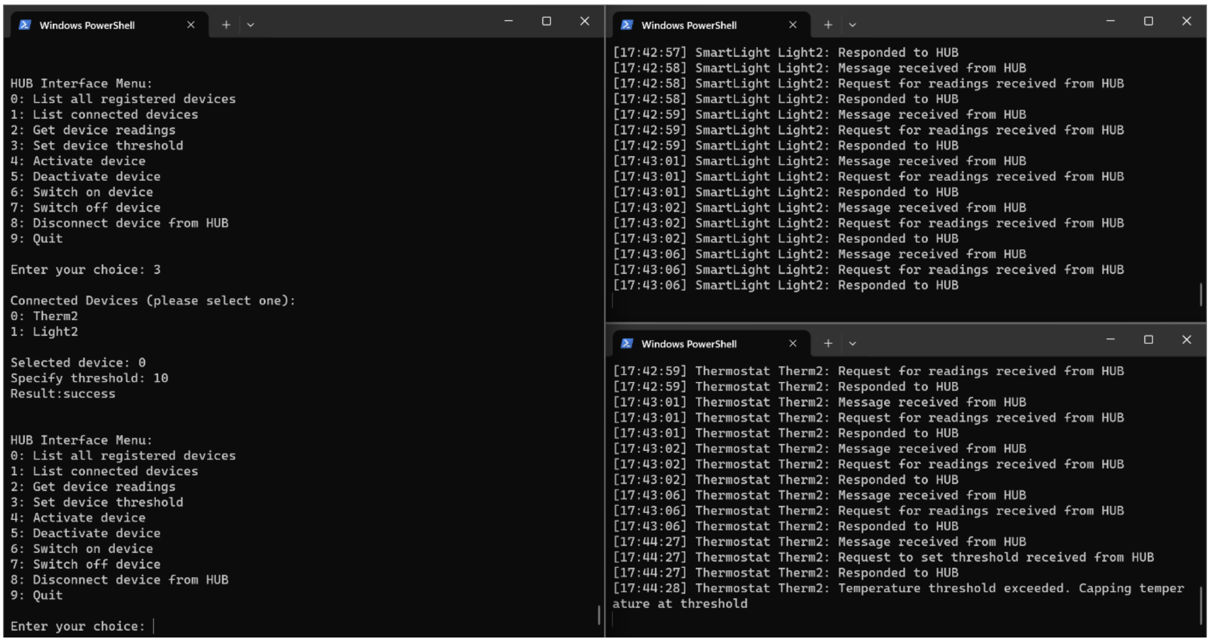

Figure 5: (Left) HUB server initiated a "Set device threshold "on the Thermostat; (Bottom-Right) Thermostat received the request and adjusted the threshold - note that the new threshold was less than the current temperature so the simulated automatic sensor proceeded to cap the temperature at the threshold.

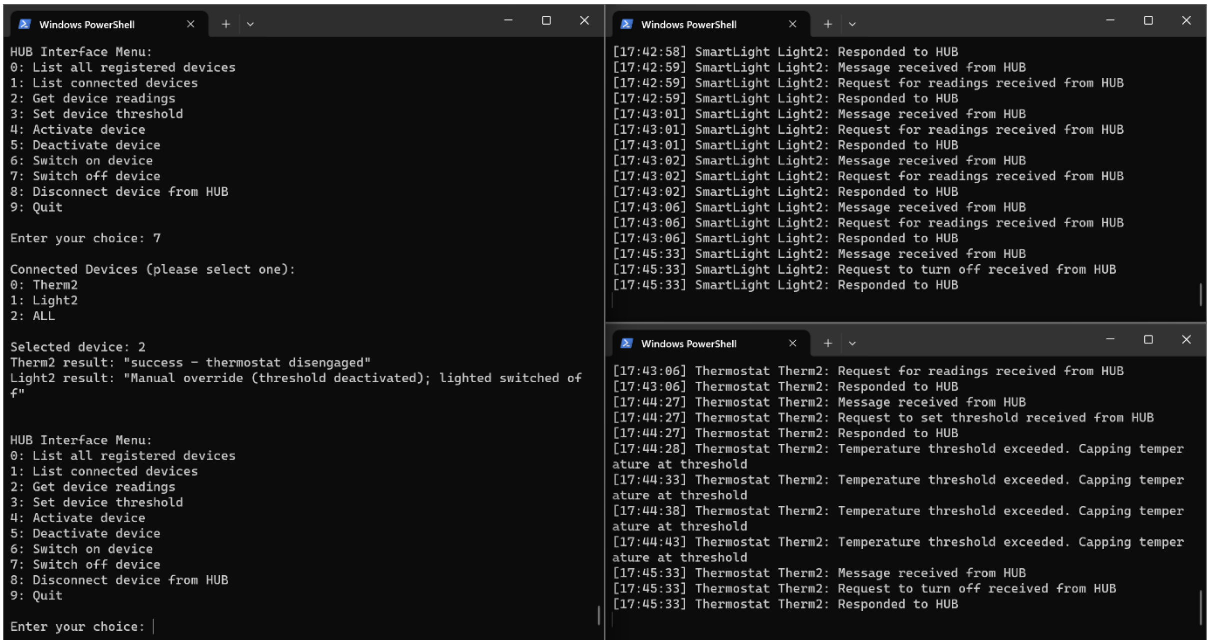

Figure 6: (Left) HUB server initiated "Switch off device" action on "ALL" devices; (Right) Devices receive the request, initiate the switch_off action and respond to the server with success.

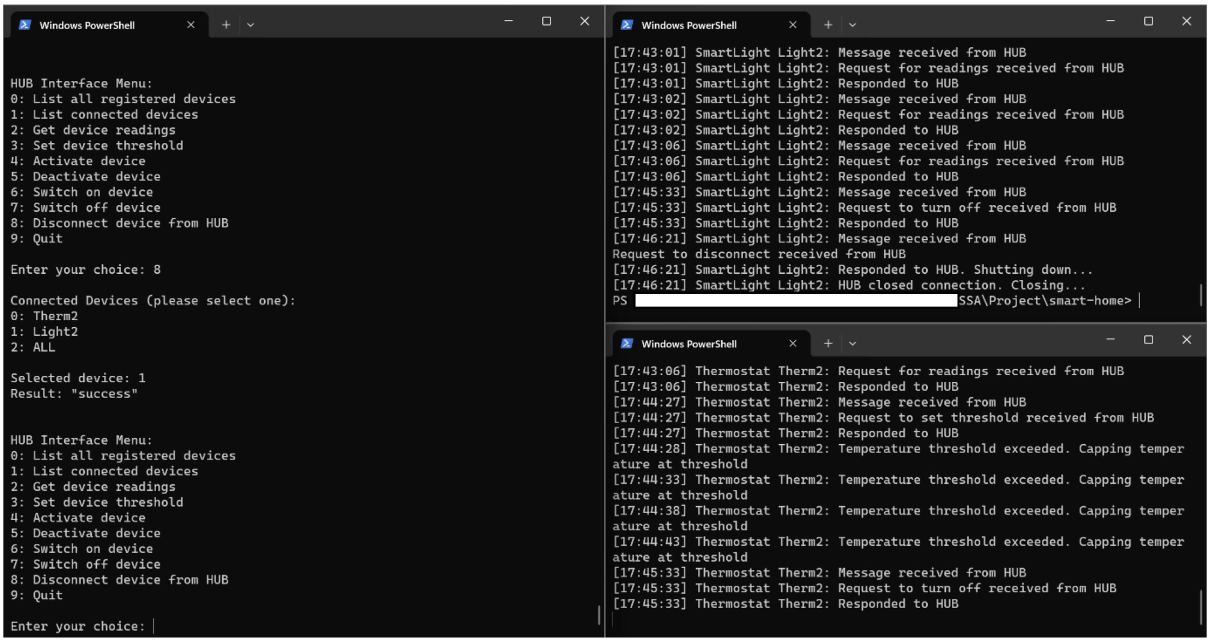

Figure 7: (Left) HUB server initiates a “Disconnect device from HUB” command and specifies only the Smart Light (Light2); (Top-Right) The Smart Light receives the request, shuts down and responds with success.

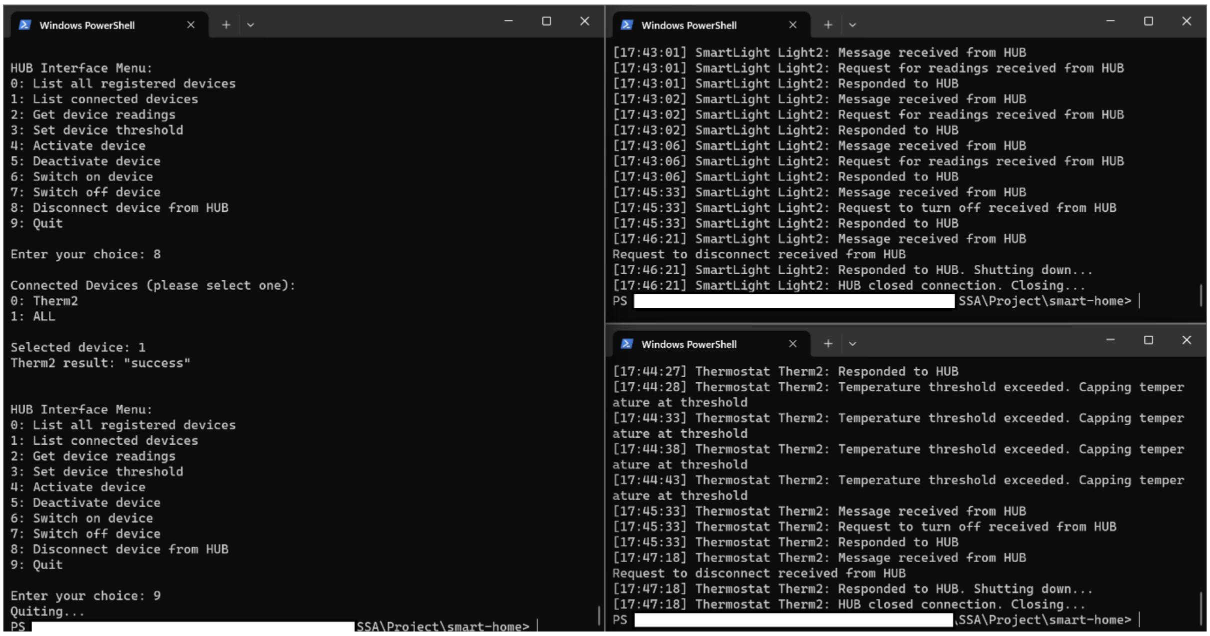

Figure 8:(Left) HUB server initiates a “Disconnect device from HUB” command and specifies “ALL” devices, and then initiates the “Quit action”; (Bottom-Right) The Thermostat (which is the only connected device) receives the disconnection request, shuts down and responds with success.