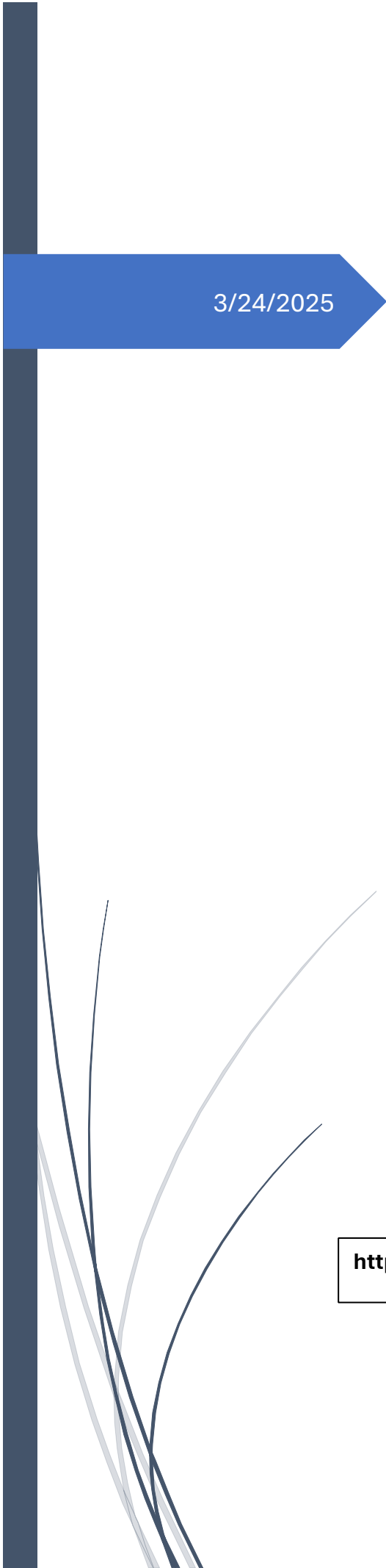




3/24/2025

SPC Assignment on

Smart Home Device Manager System for
Bright Minds Academy

Module Name: Scripting and Programming Concepts

Module Code: ACCA 5036

Student Name: RAPHAEL JERRY APPONSAH APPIAH

Student ID: 2337090

Lecturer's Name: Koushik Modak

<https://github.com/jerry258025/smart-home-device-manager>

Table of Contents

Introduction.....	1
Main Body	1
1. Analysis	1
2. Software Design and Development.....	3
Design:	3
Implementation Decisions	9
Source Code	11
3. Testing and Debugging Report	16
4. Evaluation	4
Conclusion	23
References	4
Appendices	25

Introduction

Bright Minds Academy is employing the use of Smart classroom technologies in order to improve the quality of teaching and learning through the use of smart devices such as Smart lighting , smart thermostats, and projectors, Currently, these devices are under manual management by the staff , making the task of locating and ensuring that these devices are functioning properly difficult . This method of management is error- prove, inefficient, and causes delays in detecting any devices that require maintenance. To rectify this situation, this require develop the Smart Home Device Management Program using Python programming language. The progress is easy to use because it uses a simple menu driven interface, allowing staff members to create devices, update their status, display all created devices and locate certain devices [1], [5]

Main Body

1. Analysis

- **Requirements Analysis**

The requirement analysis is an important step in the development of software as it determines what the users want from the proposed system and gives direction on how to design and implement the system [1], [2] .In case of Smart Home Device manager,

the requirement analysis divides requirements into functional and non-functional requirements.. Functional requirement are the functions that the system will carry out. They include enabling the staff to add smart devices and the location of those devices, updated the status of the devices to either online , offline or under maintenance, viewing all the devices that have been registered and searching a certain device using its name. The system will also be required to validate the user input to avoid duplication of devices and wrong input. Non-functional requirement include the attributes and features expected in the application such as usability, reliability performance, maintainability and scalability [3], [4] . *Functional Requirements*

- *Add new Devices:* Adding new devices is the key functional requirement . Staff will enter a unique device name , such as Smart Light A1, along with the classroom location. Also the system will let staff update device status using the following values.
- *Update device status: Online and offline and Uder Maintenance.* Thus , the system will help users keep track of the status of each device.
- *View device list:* There will be a view device list operation that will show all registered device along with their locations and statuses.
- *Simple Search:* Moreover, there will be a search operation that will allow entering the device name and obtaining its information.. Input validation will guarantee that no black values, duplicated device names or illegal status values are entered
- *User-Friendly Interface:* Those requirements were selected since they solve the current problems but do not make the application complex [3] ,[4].
- *Scalability:* Flexibility to expand the system later.

Non-Functional Requirements

The system should also satisfy several quality requirements. Usability will be guaranteed by providing clear menu and messages since the staff might have different technical skills.

- *Reliability:* Reliability will be achieved by validation which will decrease input errors .
- *Performance:* The performance of this application will be best.

- **Security:** The maintainability will be achieved by dividing the source code into functions..
- **Accessibility:** Besides, the design will have basic scalability to add new functions in further releases.

Abstraction : Abstraction was used in identifying only the information necessary to solve the current problem [3]. Although the real smart -device platform could handle IP addresses , sensors, power consumption, manufacturer and remote control, those issues are beyond the scope of the project. Thus, each device will be identified by only three attributes- device name, location and status,;

Problem Decomposition :

The problem decomposition was used to split the whole system into several smaller parts. The application will be split into five main operations: adding a devices , searching for a device and exit. Each operation will be implemented as a python function. The decomposition allows designing, understanding , testing and debugging the program since the components can be developed separately [1], [2] . In general , the analysis shows that the proposed solution concentrates on the business requirements without making the application too complex

2. Software Design and Development

2.1. Design

Before the implementation of the Smart Home Device Manager, the program was first designed to guarantee the creation of the system with a well- structured and rational design. Design of the project shows the way users use the system by presenting a menu consisting of five options that include adding a device, updating its status, showing a list of devices, searching for the device, and exit from the program. Flowchart was used to present the process of operations and decision -making in the system. Pseudocode was made as well as to present the main logic of the program in a simplified manner before programming it in python language

2.1 .1 Flowchart

A flowchart for the instruction of the Smart Home Device Manager was created in order to illustrate how the logical flow or decisions in this application should be carried out. The flowchart starts at the launch of the program and shows the main menu where the user is supposed to make one of five decisions. Depending on the choice made by the user, either a new device will be added , the status of the device will be updated, a list of devices will be displayed, a list of devices will be displayed , a particular devices will be searched, or the application will be closed.

The flowchart also illustrates the way in which the program sustains constant communication with the user. Once the operation , either addition or search for a device, is done , the program goes back to the main menu where another operation can be carried out. The process keeps going on until the exit choice is selected by the user. The use of decision symbols helps to clarify the different ways through which the program can run . It also depicts the way in which the program deals with wrong inputs without stopping the process. The creation of the flowchart before coding helped to plan program

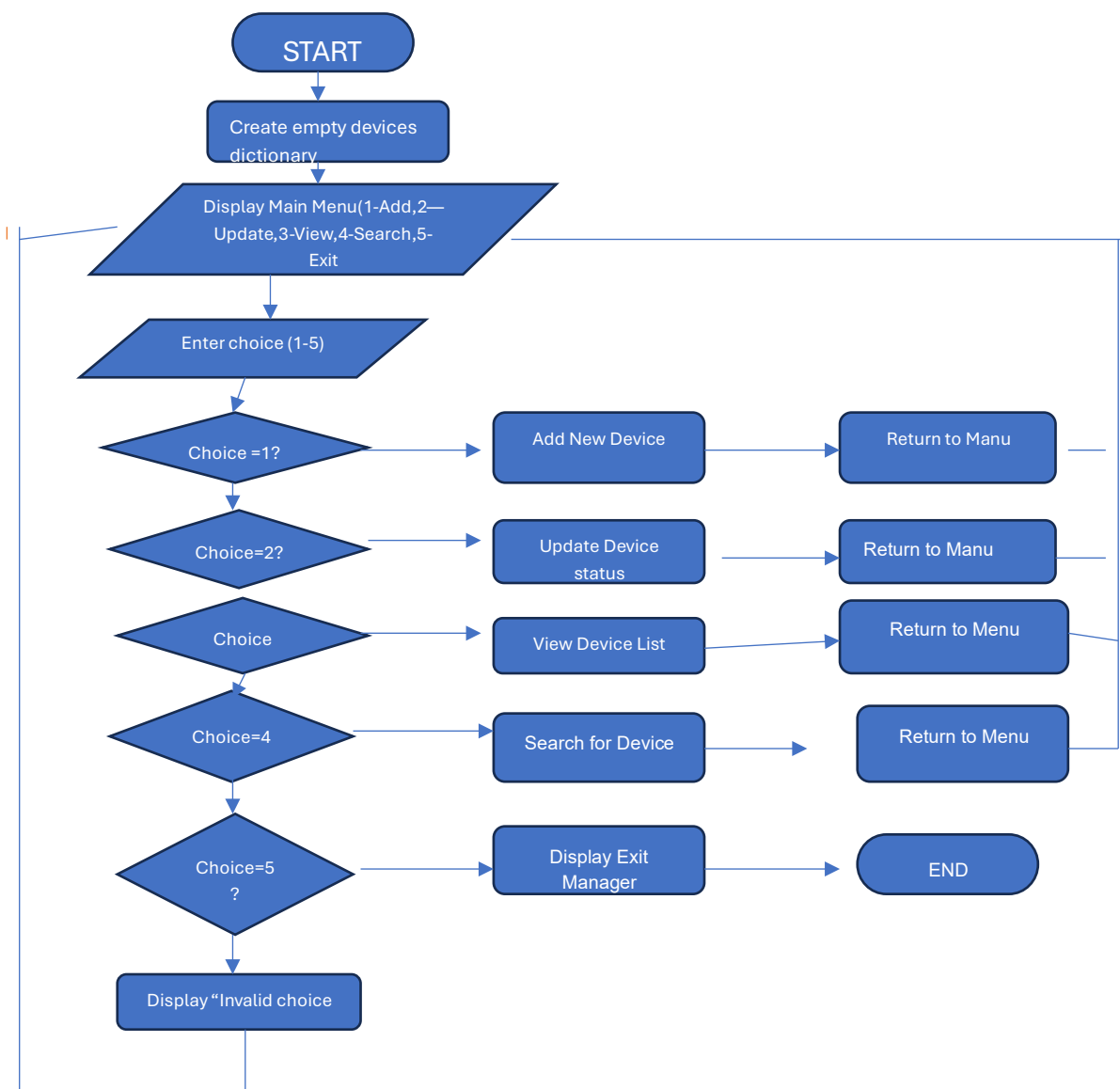


Fig 1. Flowchart of the Smart Home Device Manager

2.1.2 Pseudo -code

Create Empty Dictionary Called devices

REPEAT

DISPLAY "Smart Home Device Manager"

DISPLAY "" Add a New Device"

DISPLAY "2 Update Device Status"

DISPLAY "3 View Device List"

DISPLAY "4.. Search for Device"

DISPLAY "5. EXIT"

INPUT Choice

IF choice = "1" THEN

CALL add _ device

ELSE IF choice = "2" THEN

CALL update _ status

ELSE IF choice = "3" THEN

CALL search _ device

ELSE IF choice = "4" THEN

CALL view _ device

ELSE IF choice = "5" THEN

DISPLAY "Thank you for using the Smart Home Device Manager"

ELSE DISPLAY "Invalid choice. Please enter a number from 1 to 5 "

END IF

UNTILL choice = "5"

END

Add Device pseudocode

FUNCTION add _ device

INPUT device _ name

INPUT location

REMOVE extra spaces from device_ name

REMOVE extra spaces from location

IF device _ name is empty OR location is empty THEN

DISPLAY "Device name and location cannot be empty"

RETURN

END IF

STORE device _ name , location and status "offline" in devices

DISPLAY "Device added successfully "

END FUNCTION

Update Status pseudocode

FUNCTION update _ status

INPUT device _ name

CONVERT device _ name to lowercase

IF device _ name does not exist in devices THEN

DISPLAY "device not found"

RETURN

END IF

DISPLAY "1. Online

DISPLAY "2. Offline"

DISPLAY "3. Uder Maintenance"

INPUT status _ choice

IF status _ choice = "1" THEN

SET new _ status =" Online"

```

ELSE IF status _ choice = "2" THEN
    SET new _ status = 'Offline "
ELSE IF status _ choice = "£" THEN
    SET new _ status = "Under Maintenance"
ELSE
    DISPLAY "Invalid status choice"
    RETURN
END IF
UPDATE device status with new _ status
DISPLAT "Device status updated successfully"
END FUNCTION

```

View Devices Pseudocode

```

FUNCTION view _ devices
    IF devices is empty THEN
        DISPLAY "No devices have been added"
        RETURN
    END IF
    FOR each device in devices
        DISPLAY device name
        DISPLAY device location
        DISPLAY device status
    END FOR
END FUNCTION

```

Search Device pseudocode;

```

FUNCTION search _ device found"
    INPUT device _ name
    CONVERT device _ name to lowercase
    If device _ name exists in devices THEN

```

```

        DISPLAY "Device found"
        DISPLAY device name
        DISPLAY device location
        DISPLAY device status
    ELSE
        DISPLAY "device not found"
    END IF
END FUNCTION

```

Implementation Decisions

✦ **Data Structures:** In order to store devices data , the program uses python dictionary as the main data structures. Dictionaries were used since they allows organizing information using key-value pairs and make searching , increasing, and updating of the particular record easy. Information on devices can include their names, locations , and status. Other variable that are used in the program are temporary storage of information entered by users. Such variables are for example , device_ name, which stores the device name ; location, which stores location of the device: status which stores its operational status: and choice, which store stores users choice from menu. These variables simply the program design and facilitate the transferer of information between various actions Use of dictionaries along with properly named variables provides well- organized program structure, which is appropriate for the limited of data in the Smart Home Device Manager.

- ✦ *Programming Constructs:* Several programming concepts in python are employed by the program to control its flow of execution . Functions are used to divide the program into logical, manageable parts such as adding devices, updating status of the device, viewing the devices and searching for the devices. Conditionals are used to make decisions about what program will do based on inputs from the users . For example, it users conditionals to decide which menu option will be executed and to check whether the device being searched exists or not. A loop is used to run the program while it runs the main menu until the user selects option 5 which means exiting

- ✦ *Python Modules and Libraries:* Implementation of the smart home device manager software was done using basic python features and the software does not require any external libraries for python. The software makes use of basic constructs, including input(), print(), dictionaries , loops, conditionals, and functions. This is because the software is basically a simple command line application which means that no complicated features available in external libraries are required for implementation . It ensures that the software remains simple and light- weight

Source Code

2.3. Device Dictionary and Add Device Function

```
# Store the new device

devices[device_key] = {

    "name": device_name,

    "location": location,

    "status": "Offline"

}

print("Device added successfully.")
```

Fig 9b – Source Code for the Device Dictionary Function.

Fig 9 a

```
def add_device():

    # Ask the user for the device name and location

    device_name = input("Enter device name: ").strip()

    location = input("Enter room/location: ").strip()
```

Fig 9 a- Beginning of the Add Device Function

Device information is saved using the dictionary. The `add_device()` function takes device name and its location as an input, check whether the inputs are correct or not, look for duplicate and add a new device having offline status by default.

Fig 10a. – Beginning and Device Validation of the Update status Function

The update Device Status feature is illustrated in fig.10, where users can update the status of a device to Online, Offline or Under Maintenance. This feature validates the choice and updates the record with the relevant device information.

Fig 11 a- Beginning of the View Device List Function

```
def update_status():
    # Ask for the device name
    device_name = input("Enter device name: ").strip().lower()
    # Check whether the device exists
    if device_name not in devices:
        print("Device not found.")
        return

def view_devices():
    # Check whether any devices have been added
    if not devices:
        print("No devices have been added.")
        return
    new_status = "Online"
    elif status_choice == "2":
        new_status = "Offline"
    elif status_choice == "3":
        new_status = "Under Maintenance"
```

11b.

Fig. 11 b -Displaying Device Information

```
for device in devices.values():
    print("Device:", device["name"])
    print("Location:", device["location"])
    print("Status:", device["status"])
```

Fig 11 – Displaying Device Information .

The View Device operation is demonstrated in fig.11. This operation will

```
def search_device():  
    # Ask for a device name  
    device_name = input("Enter device name to search: ").strip().lower()  
    # Search for the device  
    if device_name in devices:  
        device = devices[device_name]  
        print("\nDevice found:")
```

verify whether any device exist and will show device information that is stored. It will employ a loop to display the name, location, and status of each device.

Fig 12a-Beginning of the Search Device Function

```
        print("\nDevice found:")  
        print("Device:", device["name"])  
        print("Location:", device["location"])  
        print("Status:", device["status"])  
    else:  
        print("Device not found.")
```

12b- Search Results and Device Validation

The search feature is illustrate in fig. 12, enabling the user to locate any device and show its name, location , and status information .

Fig 13. – Main Menu Function and Menu Options.

The code snippet of the main _ menu() function is provided in fig .13 below.
This function offers time menu through which the user can choose the

```
while True:
```

```
    print("\n=== Smart Home Device Manager ===")  
    print("1. Add New Device")  
    print("2. Update Device Status")  
    print("3. View Device List")  
    print("4. Search for a Device")  
    print("5. Exit")
```

action they need to take. The while True loop ensures that the menu runs indefinitely until the user decides to stop using it. The menu offers five choices namely: Add new device, update status, and exit. These choices link the user to the respective functions within the program ,. This makes the application easy to use and multiple action can be carried out using a single run of a program

3. Testing and Debugging Report

- a) The testing phase of programme development is an important aspect of SDLC , since it allows to make sure that a software application works properly and meets the needs of its target audience. Testing is used to find bugs, improve the quality of the software and ensure its stable and reliable functioning under various circumstances. Good testing helps detect bugs at an early stage of the software development, thus avoiding the costs of bug fixation once the software has been released. Furthermore, it increases the confidences level of users by providing the stability. , security and accuracy of results. Besides, testing ensures that the software meets the functional and non-functional requirement including usability, reliability and maintainability [7], [9] .

- b) Types of Programming Errors and Debugging :

There are different errors that can arise in the process of development programs in python. Syntax error arises if the code violates the grammatical rules of python . For instance, missing , a colon in an if statement causes the program to fail. The program usually gives the location of the syntax error so that the developer can analysis and correct the erroneous statement [3] . Runtime error arises during the execution of a program. For instance , accessing information with a wrong dictionary key could cause a runtime error. This is possible to avoid by checking for the existence of a device before accessing its information.

Local error arises when the program runs successfully but gives an incorrect result. For instance, giving a newly added device a wrong default status , is a logical errors are more difficult to detect since they do not trigger any error message on the screen. They are solved by thorough testing , analysis of program logic and comparison of actual results with expected results [2], [7] .

Debugging was therefore employed in the process of developing this program to detect errors, correct the code and test the correct program. c) .

```
=== Smart Home Device Manager ===
1. Add New Device
2. Update Device Status
3. View Device List
4. Search for a Device
5. Exit
Enter your choice (1-5): 1
Enter device name: Smart Light A1
Enter room/location: Room 1
Device added successfully.
```

Fig. : 1. Successful Addition of Smart Light A1

The test was performed to ensure the Add New Device features functions as expected with proper data entry . User chose option 1 from the main menu screen and entered “Smart Light A1 “ as the name of the device and “ Room 1 “as its location . Program accepted the data and showed the success message “ Device added successfully. . “ This output was exactly what we were expecting in this test scenario. It proves that our program has capability to accept, validate and store correct data in the dictionary to . Thus , the Add New Device features successfully passed this normal test

Test Two..

```
=== Smart Home Device Manager ===
1. Add New Device
2. Update Device Status
3. View Device List
4. Search for a Device
5. Exit
Enter your choice (1-5): 1
Enter device name: Smart Light A1
Enter room/location: Room 1
This device already exists.
```

Fig 2 : Duplicate Device Validation Test

This test was conducted in order to confirm the ability of Smart Home Device Manager to detect and block duplicate devices from being added to the system . The user chose option number 1 from the main menu , and then input the following data into the system ., device name “ Smart Light A1 “ and location “ Room 1 “ . As the same device has been added previously in the previous test , the program successfully detected the duplicate record and showed the following message,. “This device already exists “ . It confirms the proper functioning of the validation mechanism used in the program. The actual outcome corresponded to the expected one , hence, the test was passed successfully

Test three

```
Enter your choice (1-5): 2
Enter device name: Smart Light A1

Select new status:
1. Online
2. Offline
3. Under Maintenance
Enter choice: 1
Device status updated successfully.
```

Fig 3. Successful updated of Device Status.

The testing process for the update Device is illustrated in fig. 3 . The user picked the Smart Light A1 and was offered three choices of status ; Online , Offline , and Under Maintenance. Option 1 (Online) was choice to change the device status . The program interpreted the choice and showed message “ Device status updated successfully”. It proves that the program is able to recognize an existing device, get the correct choice of status, and update the information accordingly . The real output corresponds to the expected one, which confirms the correct operation of the device status updating function.

Test Four

```
--- Device List ---  
Device: Smart Light A1  
Location: Room 1  
Status: Online  
-----
```

Fig . 4. The view Device List

Figure 4 shows a successful test of the view Device List function in the Smart Home Device Manager . The user chose option 3 on the main menu to see all devices that were saved in the system at the moment . The program successfully fetched and showed the details of smart Light A1 , specifying that its location is Room 1 and its status is online. These details match those that were added when the device was first added into the system and chosen as a status in the last update test. Thus, the test proves that the program can save, fetch and show information about devices without modifying the saved data. The output received matches the one that was expected. Which means that view Device List function works as it should .

Test five

```
Device found:  
Device: Smart Light A1  
Location: Room 1  
Status: Online
```

Fig 5. Successful Search for an Existing Smart Ghome Device.

Figure 5 shows that the search for devices functionality has been successfully tested. The search menu item was chosen by the tester, who inputted " Smart Light A1 " as the name of the device. The record has been found successfully and the corresponding information about the device , including its location , which is Room1, and its status, which is Online , has been displaced. It proves that the search functionality is capable of retrieving information on devices stored in the system. The actual output matches the expected one.

Test six

```
3. View Device List  
4. Search for a Device  
5. Exit  
Enter your choice (1-5): 4  
Enter device name to search: Smart Camera B1  
Device not found.
```

Fig .6 Search for a Non – Existing Device

Test 6 was related to verifying the functioning of the search for a Device function with the user of a device that does not exist in the Smart Home Device Manager. Figure 6 provides an example of the execution of this test was the verification of the proper working of the program in case of failure of the search. The user chose the option 4- search for a Device and entered “Smart Camera B1” as the device name. This device has never been added to the system before, hence, it is not presented in the device dictionary,. The entered name of the device was verified by searching the existing records. Since Smart Camera B1 could not found, the message “Device not found” was correctly displayed. Thus, the user was informed about the fact that the requested device is not resisted in the system. This test proved the right implementation of validation procedures in the search procedure. The program tries to access data in the case when a device exists but searches for the entered name in the dictionary and does not access the non-existent data. In this way , possible tun-time errors are prevented. This increases reliability and stability of the program and makes it more user – friendly as the user receives an explanation of why the information on the device was not displayed. After the test, the program was successful working and took the user back to the main menu to perform another operation without shutting down the application . The expected result was the rejection of the unknown device name and the corresponding message being displayed. The actual result coincided with the expectations. Thus, Test 6 has been successful.

Test seven.

```
Smart Home Device Manager
Enter your choice (1-5): 7
Invalid choice. Please enter a number from 1 to 5.
```

Fig 7. Testing Invalid Manu Input and Error Handling

Figure 7 illustrate the capability of input validation and error handling by the Smart Home Device Manager. The aim of this test was to understand how the program reacts when the user enters an option from the main menu that does not belong to the list of the valid choices. As it was noted above, the main menu contains five options marked with the numbers from 1 to 5 . During the test , the user entered the option equal to 7. This test confirmed that the program could identify the entered value as the invalid choice and showed the message “invalid choice. Please enter a number from 1 to 5. “This message allows the user to understand what has happened and what kind of data he should enter. However, entering the invalid input value did not stop the application or cause any error in python.

After showing the error message, the program returned to the main menu and offered the user to enter his choice once again . This reaction shows that the program users logical conditions for distinction of valid and invalid menu options. It is very useful feature that helps

to improve the usability of the program allowing the user to correct his mistakes. The expected output of the test was that the program will not accept option 7 and show the error message without stopping its execution. The actual output corresponds to the expected one. Thus, Test 7 was successful

Test 8

```
=== Smart Home Device Manager ===
1. Add New Device
2. Update Device Status
3. View Device List
4. Search for a Device
5. Exit
Enter your choice (1-5): 5
Thank you for using the Smart Home Device Manager.

>>> |
```

Fig. 8.- Testing the Exit Function

The result of successful testing of the Exit function in the Smart Home Device Manager are provided in figure 8. The test was performed to make sure that the program stops and exits the application correctly. If the user wants to do so. On the main menu screen, there were five available options, which include addition, modification of device status, viewing of device list, search of a certain device, and exiting the program. In order to perform Test 8, option 5 was chosen.

Upon entering 5, the application recognized the chose mode and showed a feed back to the user that . Thank you for using the Smart Home Device Manager “. This gives the user feedback that his choice to exit has been accepted. After that feedback, the python shell goes back to the >>> prompt. It indicates that the main loop of the application has stopped It also . indicates that the break statement used in the application works successfully. Once option 5 has been entered, the break statement breaks the main loop that repeats the main menu. It stops the repetition of the menu once the user has chosen to exit the application. There was no error message, unexpected output, or application crash in the process of the test. Thus, the actual result was the same as the expected result. It shows the exit function in this application is reliable, easy to use and user- friendly. In summary, Test 8 was successful

Evaluation

- **Challenges Faced:** Throughout the development of the Smart Home Device Management Program, several problems came up. First, it was a matter of choosing a suitable way to save information about the devices. A python dictionary was chosen to help effectively store and find the information about the names, location, and status of a particular device. Secondly, problems with input validation were faced, meaning that without validation, the user could enter either nothing, duplicates, or some invalid status choice. These problems could be solved with the help of conditional statements, strip () and lower () methods, as well as checking whether the device already exist. Other problems identified while testing the program include case-sensitive searches and wrong menu choices, and these problems could be solved through changing the logic of the program as well as creating proper messages for the user in the cases when the input was incorrect,[[1].

- **Future Improvements:**

Although the Smart Home Device Manager meets all the current project requirements, there are still many to improve the program. Among them is the change from temporary dictionaries to databases like SQLite or MySQL so that the information about the devices would be saved even when the program is not run anymore. Developing the graphical user interface with TK inter could increase its usability and make it easier to operate for those who are not familia with programming. Such possibilities as editing, deleting, tracking timestamps of status changes, categorization of devices (for example, projectors, thermostats, smart lights,) as well as user authentication with usernames and passwords could be added. Generating reports on the status of the devices in PDF or Excel format could also be considered. Finally, in advanced version, the integration of the program with the internet of Things technologies could make possible monitoring and updating the status of the devices connected in the classroom in real time. Moreover , further security measures can be incorporated in the form of user identification and passwords to ensure that there is no unauthorized access to information about devices. Moreover, there can be some automated device monitoring and logging of dates and times as

well as improved reporting facilities. These will make the application more secure and practical [1], [2]

- **Self-Reflections:**

The completion of this project helped me learn more about python programming as well as the process of software development. It provided experience in problem analysis, identifying requirements, decomposing problem into smaller once through abstraction. Moreover, practical skills in working with dictionaries, functions, loops, conditional statements, input validation were also gained. Creating the pseudocode and flowcharts gave the ability to understand the necessity of planning before writing the source code of the program. Testing and debugging turned out to be helpful to see how small problems may affect the program and how testing helps in finding and solving such problem.

The program not only improved my competence in programming , but also enhance my confidence while writing programs on my own. It was useful for me to know how significant it is to give meaningful names to variables, how to arrange codes into functions, and how to validate the input from the users

Conclusion

Smart Home Device Manager meets all requirements of Bright Minds Academy through providing an easy – to- used efficient way of managing smart classroom devices. It allows adding device, updating device status, viewing devices already registered, and finding certain devices using a menu- based interface. In the implementation of this program, the principles of using python dictionaries, user- defined functions, and input validation have been applied to make sure that the software is reliable and maintainable. Throughout the whole development process, some basic software engineering concept like problem decomposition, abstraction, testing, and debugging have been used to build a working application.

In general, this project shows the possibility of creating useful software applications with the help of python [1], [5]

Reference

Bibliography

- [1] I. Sommerville, Software Engineering, Harlow: Pearson, 2016.
- [2] R. S. Pressman and B. R. Maxim, Software Engineering :A PR actioner Approach, New York: McGraw- Hill Education, 2019, p. 9.

- [3] T. Gaddis, , Harlow: Pearson, 2021, p. 5.
- [4] A. B. Downey, How to Think Like a Computer Scientist,, 2, Ed., Sebastopol, CA: O Reilly, 2015.
- [5] Python Software Foundation , The Python Language Reference, Version 3.13, Wilmington, DE: Python Software Foundation, 2025.
- [6] B. W. Kernaghan and R. Pike, The Practice of Programing, Boston, MA: Addison Wesley, 1999.
- [7] S. McConnell, Code Complete, Redmond, WA: Microsoft Press, 2024.
- [8] M. Fowler, Refactoring :Improving the Design of Existing Code, Boston. MA: Addison-Wesley, 2019.
- [9] H. Washizaki, Guide to the Software Engineering Body of Knowledge(SWEBOK Guide), version 4.0, Washington, DC: IEEE Computer Society, 2024.
- [10] National Institute of Standards and Technology, “Framework for Improving Critical Infrastructure Cybersecurity. Version 2.0,” National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2024.
- [11] B. Eckel, “Thinking in Python: Design Patterns and Problem Solving Techniques,” Mind View, Inc, 2002.
- [12] E. Matthes, Python Crash Course: A Hands - on Project-Based Introduction to Programming, San Francisco: No Starch Press, 2023.
- [13] D. Beazley and B. Jones, Python Cookbook, Sebastopol: O` Reilly Media, 2013.
- [14] K. A. Lambert, Fundamentals of Python: First Program, Boston , MA: Cengage Learning, 2024.

