UNIVERSITI POLY-TECH MALAYSIA

UPTM

Name: SYED AMER AIDID BIN SYED MOHD BAKRI		
Id Number: AM2207011659		
Lecturer Name: **AKMAL BIN AZMER**		**Section No.:** **02**
Course Name and Course Code: SWC2373 EMERGING TECHNOLOGY		**Submission Date:** WEEK 13/ 10th November 2023
Assignment Title: Assignment		**Extension & Late Submission:** Allowed / **Disallowed**
Assignment Type: **INDIVIDUAL**	**% of Assignment Mark:** **40%**	**Returning Date:**

Penalties:

1. 10% of the original mark will be deducted for every one-week period after the submission date
2. No work will be accepted after two weeks of the deadline
3. If you were unable to submit the coursework on time due to extenuating circumstances, you may be eligible for an extension
4. Extension will not exceed one week 3.

Declaration: I/We the undersigned confirm that I/we have read and agree to abide by these regulations on plagiarism and cheating. I/we confirm that this of work is my/our own. I/we consent to appropriate storage of our work for checking ton ensure that there is no plagiarism/academic cheating.

Signature(s): syed

Name: SYED AMER AIDID BIN SYED MOHD BAKRI

This section may be used for feedback or other information

TABLE OF CONTENT

INTRODUCTION

Cisco Webex is a collaboration platform that allows for video conferencing, online meetings, screen sharing, and other forms of communication. It's used for online meetings, webinars, and team collaboration.

Webex provides an API (Application Programming Interface) that enables developers to incorporate Webex functionality into third-party apps or create customized applications that use Webex features. The Webex API provides a set of RESTful APIs that developers can use to programmatically interact with Webex services.

To use the Webex API, you must first create a developer account on the Webex for Developers platform. This account allows you to view the API documentation. The API allows you to integrate Webex capabilities into your app, such as video conferencing and chat. Understand the OAuth 2.0 authentication procedure, which includes getting API keys and access tokens. Explore the API documentation to learn about the endpoints and functionalities that are accessible. Then, construct the app in your choice programming language, integrating the Webex API's capabilities. In the Webex API sandbox, test your application to ensure it works as planned. When you're ready, request production access to test your app in a live environment. Keep up to current on API updates and announcements for future development.

OBJECTIVES

The task that needs to be achieved in this assignment is to create a troubleshooting tool that checks the user’s details when a conferencing session is being held. To achieve that I need to:

1. Learn how to integrate the Webex API into a custom application.
2. Implement secure user authentication using Webex tokens and OAuth 2.0.
3. Create a troubleshooting tool application that verifies user information during a conferencing session.
4. Make a user-friendly interface with a menu system to allow for seamless interaction.
5. Carefully test the application, providing evidence of successful functionality.
6. In a written report, document the analysis and development process.
7. Document the analysis and development process in a written report.
8. Use git and upload all code to a GitHub repository.
9. Provide error handling and acknowledgment messages.
10. Show room management features such as listing, creating, and sending messages to rooms.
11. Improve my problem-solving abilities by addressing the assignment's specific criteria.

LITERATURES REVIEW

1. **Software and Programming Language:**
    - Python: Python is a versatile and widely-used programming language suitable for various applications. It is known for its readability and ease of use. In this assignment, the Webex Teams SDK (Software Development Kit) for Python is utilized. The SDK simplifies interactions with the Webex API by providing convenient methods and abstractions.
    - VSCode (Visual Studio Code): VSCode is a lightweight, extensible code editor widely used for Python development. Its features, including syntax highlighting, debugging support, and extensions.

2. **Dependencies:**
    - *webexteamssdk:* The Webex Teams SDK for Python is a primary dependency. It provides classes and methods to interact with the Webex Teams API, facilitating tasks such as sending messages, creating rooms, and retrieving user details.
    - *VSCode Extensions:* While not explicitly mentioned, VSCode extensions may enhance the development experience. Extensions for Python support, such as linting and code formatting, contribute to code quality.

3. **Architecture between Webex API and the Application:**
    - The architecture is likely based on the principles of object-oriented programming (OOP). The WebexTeamsAPI class from the Webex Teams SDK represents the client interacting with the Webex API. The application interacts with this class to perform actions such as listing rooms or sending messages.
    - The Webex Teams API itself follows a RESTful architecture, where different endpoints correspond to various functionalities. The SDK abstracts the complexities of HTTP requests and authentication, allowing developers to focus on the application logic

4. **<u>Connection between Application and Webex API:</u>**
    - The application connects to the Webex API by creating an instance of the WebexTeamsAPI class with a specific access token (access_token). This access token is crucial for authenticating and authorizing the application to access the Webex Teams API on behalf of the user.
    - Methods provided by the WebexTeamsAPI class, such as rooms.list(), are used to make HTTP requests to the Webex API endpoints. These requests fetch information like room details, which can then be processed and displayed by the application.
    - The application's logic is structured around utilizing these SDK methods to perform actions, presenting a clear and readable codebase.

DEVELOPMENT OF APPLICATIONS

1. This code imports modules from the python library. First we import WebexTeamsApi and ApiError modules from webexteamssdk library that is essential for us to create webex applications. Then we import os so we can run cli command like clear the terminal

```
from webexteamssdk import WebexTeamsAPI, ApiError
import os
```

2. This function checks the connection. It takes an api object as a parameter. We check connections by running api.people.me(). If the output is the information of the user, it means the connection is successful. If it is something else, then it is an error. The type of error is ApiError, so we can catch the error and give proper output to user to let them know that the access token they enter may be incorrect

```
def test_Connection(api):
    try:
        api.people.me()
        return True
    except ApiError:
        return False
```

3. This function list 5 rooms that users have. Also take an api object as a parameter. We declare rooms and assign api.room.list() to it. So the rooms contain a list or iterable of objects. So we iterate through the rooms using for loops with [:5] to loop it 5 times only. And we display the rooms details using print format. The variable i is for the numbering, so user can just enter the number to choose which room they want.

```
def listRooms(api):
    rooms = api.rooms.list()
    print("\nRooms List:")
    i = 1
    for room in rooms[:5]:
        print(f"\n{i}. Room Name: {room.title}\n   Room Id:
{room.id}\n   Date Created: {room.created}\n   Date of Last
Activity: {room.lastActivity}")
        i = i + 1
```

4. os.system("clear") clear the terminal so it easy for user to see the output of this applications. Secondly, we prompt user to enter their access token. Then we create an instance of WebexTeamsAPI name api and give the access token given by user as argument. This api object will be use in the 2 functions we create earlier.

```
# clear terminal
os.system("clear")

# prompt access token from user
access_token = input("enter access token: ")

# create WebexTeamsApi instance
api = WebexTeamsAPI(access_token=access_token)
```

5. After getting the access token, we prompt user with menu with 5 options to choose from. We also put rest of the code in while loop. So it loops until user say stop.

```
while True:
    choice = input("\nOption 0: Test Connection\nOption 1: Display my
info\nOption 2: List Rooms\nOption 3: Create room\nOption 4: Send
Message\n\nEnter your option: ")
```

6. First option is to check the connections. We use the functions we created earlier to test the connection. If the functions return true, we display “Connections success” to the user. However if the connection is failed, we tell user the connection failed and enter valid token first to use other options.

```
if choice == '0':
        if test_Connection(api):
            print("\nConnection success")

        else:
            print("\nConnection failed\nPlease enter valid access token 
if you want to use other option")

    elif choice == '1':

        if test_Connection(api):
            me = api.people.me()
            email_list = ', '.join(me.emails)

            print("\nDisplay Name: " + me.displayName +
            "\nNickname     : " + me.nickName +
            "\nEmails       : " + email_list)

        else:
            print("Please enter valid access token first")
```

7. The second option is to display user information. We first test the connection. If it is good, we run some code to get user information and display it to the user. If the connection fails, display some error to the user.

```
elif choice == '1':

        if test_Connection(api):
            me = api.people.me()
            email_list = ', '.join(me.emails)

            print("\nDisplay Name: " + me.displayName +
            "\nNickname     : " + me.nickName +
            "\nEmails       : " + email_list)

        else:
            print("Please enter valid access token first")
```

8. Third option is to list user’s 5 recent rooms and it details.

```
elif choice == '2':

        if test_Connection(api) == 1:
           listRooms(api)

        else:
            print("\nPlease enter valid access token first")
```

9. Fourth option is to create a new room. First we prompt the user to enter the name for this new room. If the new room name already exists, we tell the user it already exists. If it does not exists, then we create the new room with the name given by the user

```
elif choice == '3':
        if test_Connection(api):
            roomName = input("\nPlease enter new room name: ")

            # store list of rooms in rooms
            rooms = api.rooms.list()

            # iterate rooms to check if the room name already exists
            for room in rooms:
                if roomName == room.title:
                    print("exist")
                    break

            # if not exists, then create the room
            else:
                try:
                    api.rooms.create(roomName)
                    print("room create")
                except ApiError:
                    print("Error creating room!")
                except TypeError:
                    print("invalid value")
        else:
            print("\nPlease enter valid access token first")
```

10. So the last option is to send a message to a room. First we display the list of rooms to the user with its details, then prompt the user to enter which room they want to send a message to. After that, prompt the user what message they want to send. If the message is successfully sent to the room, we tell the user it successfully sent. However if it fails, we tell the user that there is an error sending the message.

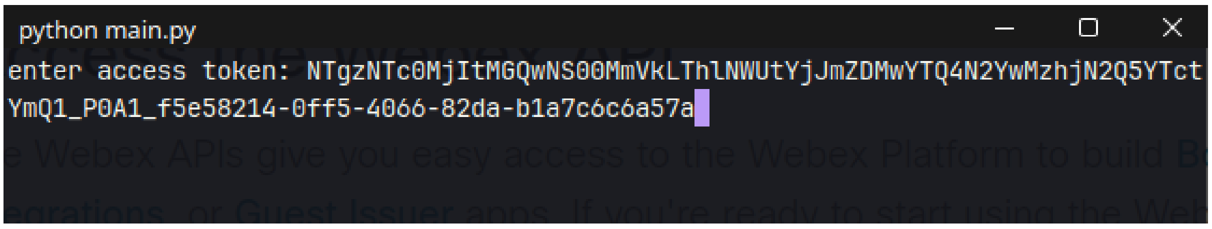
```
    elif choice == '4':
         if test_Connection(api) == 1:
             listRooms(api)

             # create array to store roomsId
             roomsId = []

             for room in api.rooms.list()[:5]:

                 # append the room's id in the array
                 roomsId.append(room.id)

             # room to send message to
             roomToSend = int(input("\nPlease enter which number room to
send message to: "))

             # message to send
             message = input("Enter message you want to send: ")

             if api.messages.create(roomsId[roomToSend-1],
text=message):
                 print("\nGG! Message sent!!!!!")

             else:
                 print("Error sending message")

         else:
             print("\nPlease enter valid access token first")
```

11. If the user enter invalid options, we simply tell the user the input is invalid. After running the option that user choose, we will prompt the user if they want to go back to the main menu. If the answer is ‘y’ we will clear the terminal and loop options again until the user say ‘n’.

```
else:
        print("invalid input")

    menu = input("\nGo to the main menu? (y/n): ")
    if menu == 'y':
        os.system("clear")
    elif menu == 'n':
        print("Exiting the program.")
        break
    else:
        print("invalid input")
        break
```

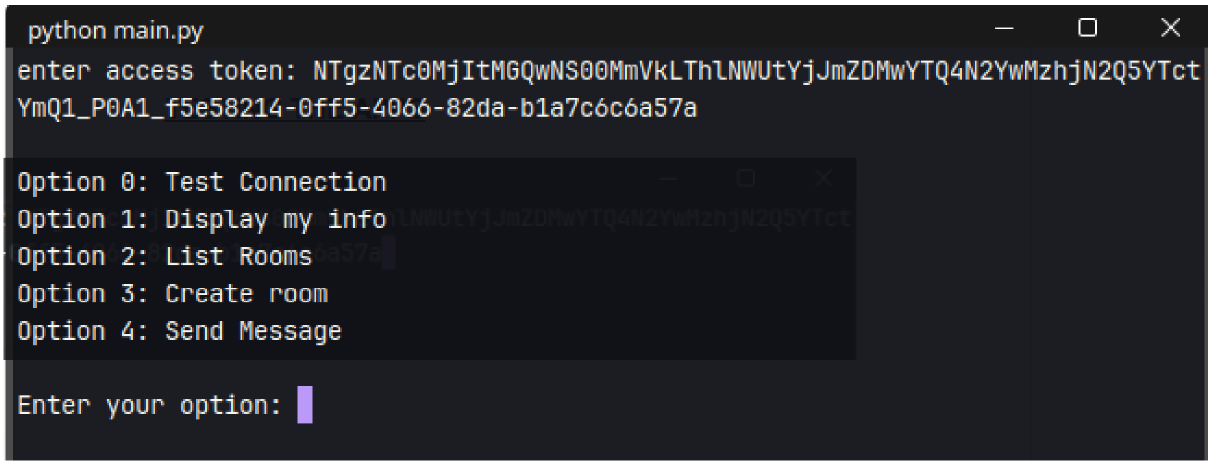
TESTING THE APPS

1. Enter access token



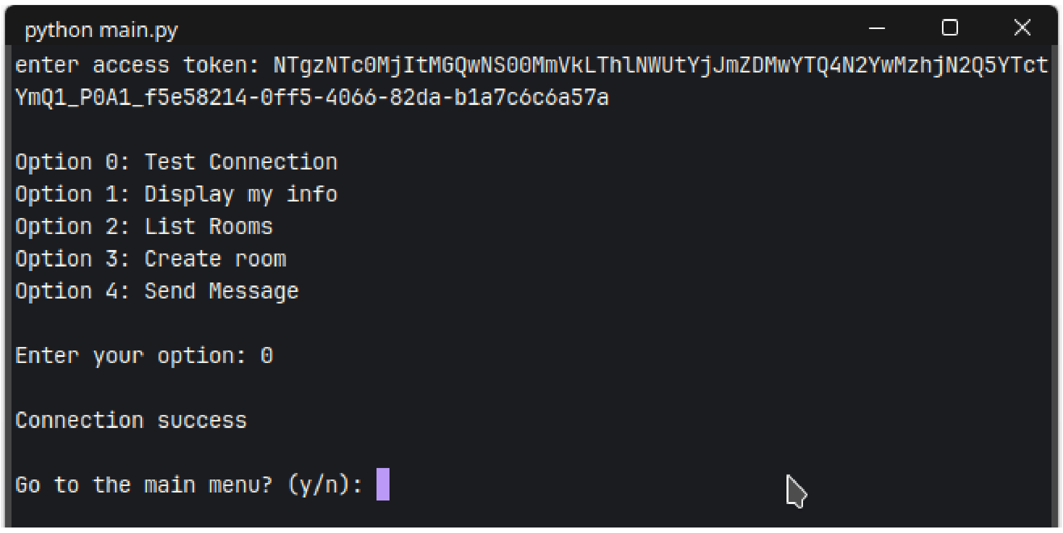
```
python main.py
enter access token: NTgzNTc0MjItMGQwNS00MmVkLThlNWUtYjJmZDMwYTQ4N2YwMzhjN2Q5YTct
YmQ1_P0A1_f5e58214-0ff5-4066-82da-b1a7c6c6a57a
```

2. Choose options



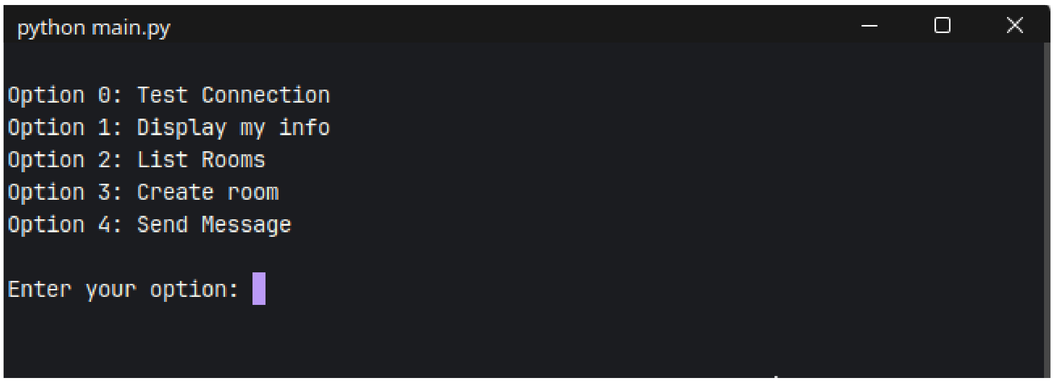
```
python main.py
enter access token: NTgzNTc0MjItMGQwNS00MmVkLThlNWUtYjJmZDMwYTQ4N2YwMzhjN2Q5YTct
YmQ1_P0A1_f5e58214-0ff5-4066-82da-b1a7c6c6a57a

Option 0: Test Connection
Option 1: Display my info
Option 2: List Rooms
Option 3: Create room
Option 4: Send Message

Enter your option:
```

3. Option 0. Then choose to continue



```
python main.py
enter access token: NTgzNTc0MjItMGQwNS00MmVkLThlNWUtYjJmZDMwYTQ4N2YwMzhjN2Q5YTct
YmQ1_P0A1_f5e58214-0ff5-4066-82da-b1a7c6c6a57a

Option 0: Test Connection
Option 1: Display my info
Option 2: List Rooms
Option 3: Create room
Option 4: Send Message

Enter your option: 0

Connection success

Go to the main menu? (y/n):
```

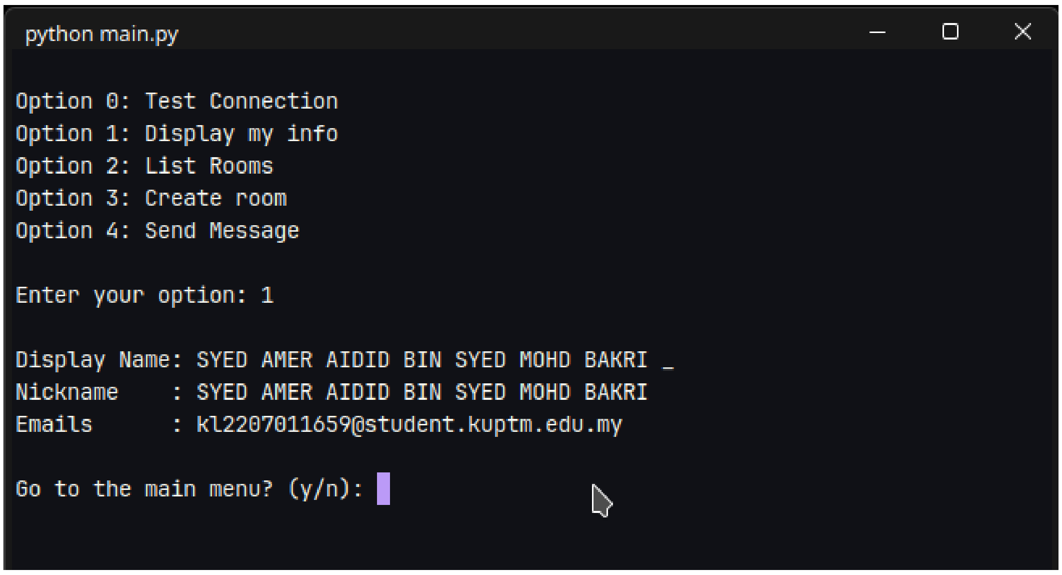
4. After choose to continue, the terminal will be clear and the application will prompt the options again



python main.py

```
Option 0: Test Connection
Option 1: Display my info
Option 2: List Rooms
Option 3: Create room
Option 4: Send Message

Enter your option:
```

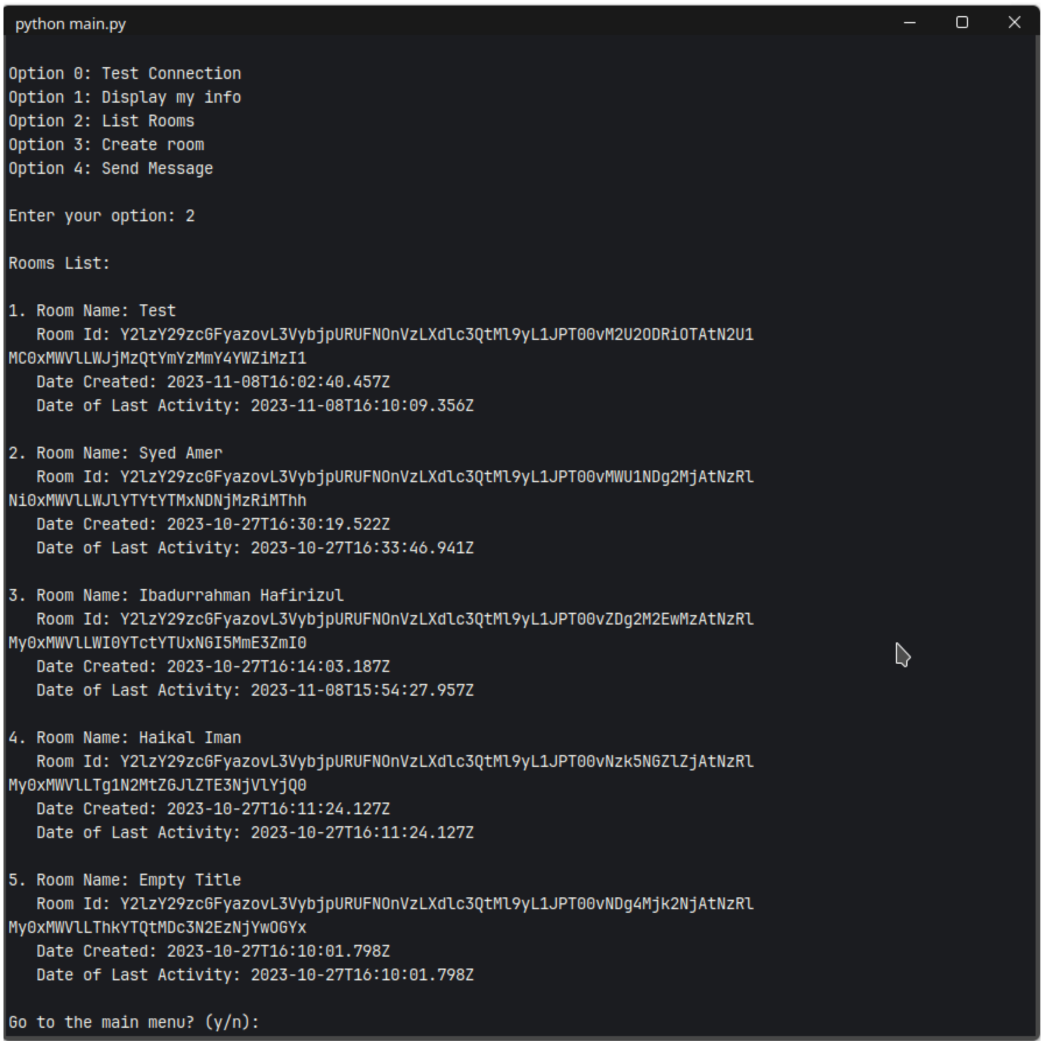
5. Then choose Option 1.



python main.py

```
Option 0: Test Connection
Option 1: Display my info
Option 2: List Rooms
Option 3: Create room
Option 4: Send Message

Enter your option: 1

Display Name: SYED AMER AIDID BIN SYED MOHD BAKRI _
Nickname     : SYED AMER AIDID BIN SYED MOHD BAKRI
Emails       : kl2207011659@student.kuptm.edu.my

Go to the main menu? (y/n):
```

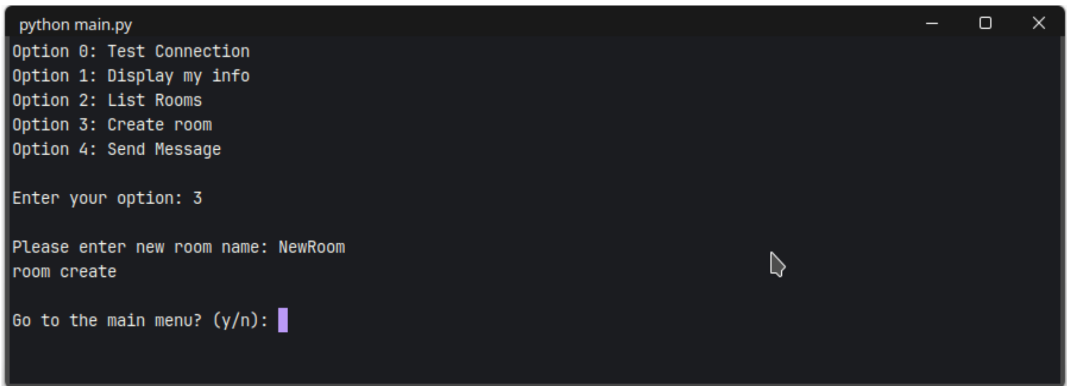
6. Option 2



python main.py

```
Option 0: Test Connection
Option 1: Display my info
Option 2: List Rooms
Option 3: Create room
Option 4: Send Message

Enter your option: 2

Rooms List:

1. Room Name: Test
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vM2U2ODRiOTAtN2U1
MC0xMWVlLWJjMzQtYmYzMmY4YWZiMzI1
   Date Created: 2023-11-08T16:02:40.457Z
   Date of Last Activity: 2023-11-08T16:10:09.356Z

2. Room Name: Syed Amer
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vMWU1NDg2MjAtNzRl
Ni0xMWVlLWJlYTYtYTMxNDNjMzRiMThh
   Date Created: 2023-10-27T16:30:19.522Z
   Date of Last Activity: 2023-10-27T16:33:46.941Z

3. Room Name: Ibadurrahman Hafirizul
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vZDg2M2EwMzAtNzRl
My0xMWVlLWI0YTctYTUxNGI5MmE3ZmI0
   Date Created: 2023-10-27T16:14:03.187Z
   Date of Last Activity: 2023-11-08T15:54:27.957Z

4. Room Name: Haikal Iman
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNzk5NGZlZjAtNzRl
My0xMWVlLTg1N2MtZGJlZTE3NjVlYjQ0
   Date Created: 2023-10-27T16:11:24.127Z
   Date of Last Activity: 2023-10-27T16:11:24.127Z

5. Room Name: Empty Title
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNDg4Mjk2NjAtNzRl
My0xMWVlLThkYTQtMDc3N2EzNjYwOGYx
   Date Created: 2023-10-27T16:10:01.798Z
   Date of Last Activity: 2023-10-27T16:10:01.798Z

Go to the main menu? (y/n):
```

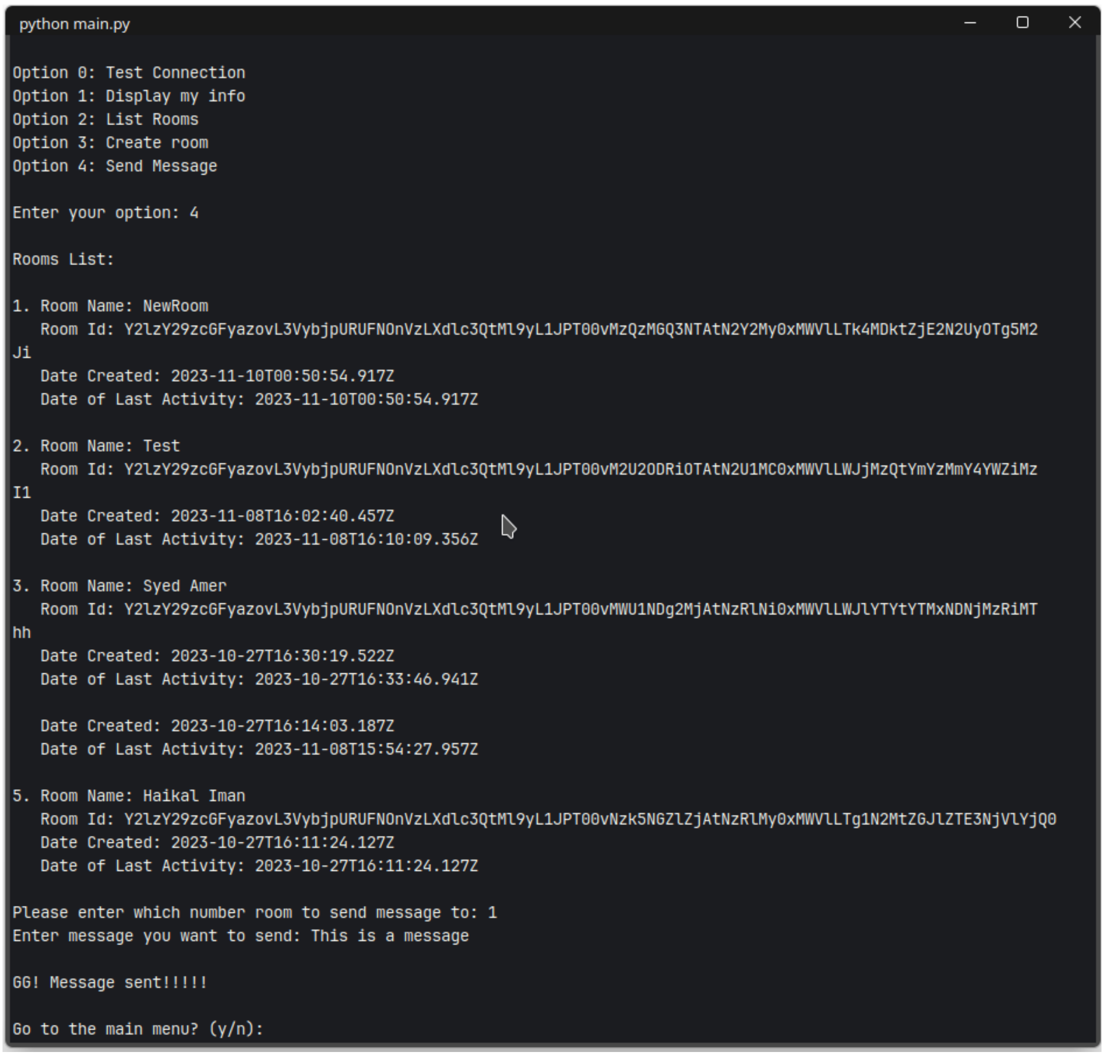
7. Option 3



python main.py

```
Option 0: Test Connection
Option 1: Display my info
Option 2: List Rooms
Option 3: Create room
Option 4: Send Message

Enter your option: 3

Please enter new room name: NewRoom
room create

Go to the main menu? (y/n):
```

8. <u>Option 4</u>



```
Option 0: Test Connection
Option 1: Display my info
Option 2: List Rooms
Option 3: Create room
Option 4: Send Message

Enter your option: 4

Rooms List:

1. Room Name: NewRoom
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vMzQzMGQ3NTAtN2Y2My0xMWVlLTk4MDktZjE2N2UyOTg5M2
Ji
   Date Created: 2023-11-10T00:50:54.917Z
   Date of Last Activity: 2023-11-10T00:50:54.917Z

2. Room Name: Test
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vM2U2ODRiOTAtN2U1MC0xMWVlLWJjMzQtYmYzMmY4YWZiMz
I1
   Date Created: 2023-11-08T16:02:40.457Z
   Date of Last Activity: 2023-11-08T16:10:09.356Z

3. Room Name: Syed Amer
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vMWU1NDg2MjAtNzRlNi0xMWVlLWJlYTYtYTMxNDNjMzRiMT
hh
   Date Created: 2023-10-27T16:30:19.522Z
   Date of Last Activity: 2023-10-27T16:33:46.941Z

   Date Created: 2023-10-27T16:14:03.187Z
   Date of Last Activity: 2023-11-08T15:54:27.957Z

5. Room Name: Haikal Iman
   Room Id: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNzk5NGZlZjAtNzRlMy0xMWVlLTg1N2MtZGJlZTE3NjVlYjQ0
   Date Created: 2023-10-27T16:11:24.127Z
   Date of Last Activity: 2023-10-27T16:11:24.127Z

Please enter which number room to send message to: 1
Enter message you want to send: This is a message

GG! Message sent!!!!!

Go to the main menu? (y/n):
```

CONCLUSION

Using the Webex API, I was able to finish the jobs I was given and make a troubleshooting tool. People who use this tool can try connections, see information about other users, see a list of rooms, make new rooms, and send messages. With Webex keys and OAuth 2.0, I've made sure that user authentication is safe. The tool was tested thoroughly to make sure it works well. A report with lots of details about the development process and the choices that were made makes it easy to understand what happened. I used GitHub to keep track of different versions of the code, which made it easy to handle and work together on. The tool appropriately handles errors, making the user experience better. The implementation also includes room management tools, which shows that a good grasp of how Webex rooms work is present. Overall, meeting these goals shows that I understand how to integrate the Webex API and solve problems effectively in application creation.

REFERENCES

1. developer.webex.com. (n.d.). *Getting started with Webex | Webex for Developers*. [online] Available at: https://developer.webex.com/docs.
2. webexteamssdk.readthedocs.io. (n.d.). *webexteamssdk — webexteamssdk 1.6.1.post28.dev0+g22dcb08 documentation*. [online] Available at: https://webexteamssdk.readthedocs.io/en/latest/ [Accessed 10 Nov. 2023].
3. W3Schools (2019). Python Tutorial. [online] W3schools.com. Available at: https://www.w3schools.com/python/.