UNIVERSITI POLY-TECH MALAYSIA
UPTM

Name: **NUR SYAZWANI BINTI SYAHRIL**		
Id Number: **AM2207011636**		
Lecturer Name: **MOHD AKMAL BIN MOHD AZMER**		Section No.: **01**
Course Name and Course Code: **EMERGING TECHNOLOGIES – SWC 2373**		Submission Date: **10 NOVEMBER 2023**
Assignment Title: **WEBEX API (REPORT)**		**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): [illegible]

Name: **NUR SYAZWANI BINTI SYAHRIL**

This section may be used for feedback or other information

TABLE OF CONTENTS

1. INTRODUCTION

Webex is an open online platform for collaboration that aims at improving how people communicate and collaborate. Performing as a central location for online meetings, webinars and video conferences, Webex connects distances and unites people regardless of where they are physically located. Webex has become an essential instrument for productivity and remote communication because of its features, which include real-time collaboration, easy sharing if content and a user-friendly interface. It's ability to adapt to a wide range of professionals' contexts makes it an essential instrument for both individuals and business, promoting productive and successful collaboration in the digital era.

Moreover, developers have a variety of options due to the Webex API, which enables them to integrate and customize Webex features to produce solutions that are specifically designed to meet enterprise requirements.

1.1 What is Webex?

Webex, developed by Cisco is a product suite for video conferring and collaboration that combines Webex Meetings, Webex Teams and Webex Devices. This cloud-based platform merges Cisco's Web conferencing and team collaboration apps, rebranded in April 2018. It simplifies online meetings, team messaging and file sharing, serving both small groups and large enterprises. Users can join video conferences on desktop or mobile with additional features on other versions. As a leading unifies communications platform, Webex streamlines collaboration across businesses.

1.2 What is Webex API?

Webex API is like a toolbox for developers, letting them add Webex features to their own apps. It can use to handle contacts, calendars and even record meetings. The API uses OAuth and JWT for security, supporting REST APIs and Webhooks. This means developers can create custom apps and expand what Webex can do. In essence, the Webex APU facilities a seamless and programmable collaboration, allowing external apps to integrate smoothly for personalized solutions in meetings, messaging and collaboration.

1.3 How can third-party apps be developed with Webex API?

In Webex, third-party apps are like extra tools created by outside developers. These apps use a special set of instructions called the Webex API to connect and work with Webex features. Instead of just using what Webex offers, these external apps bring their own unique functions, making the Webex experience more customized and adaptable to different needs. It can be anything from special meeting tools to chat helpers or smart assistants, all created by developers to add extra stuff to Webex.

Here are a few ways third-party apps can be developed using the Webex APIs :

Messaging Integration: Leverage Webex Teams API to add messaging capabilities to apps, enabling functions like sending and receiving messages, creating spaces and managing users for collaboration apps.

Embedding Meetings: With the Meetings Embed API, embed Webex meetings into web and mobile apps seamlessly using iframes for a tightly integrated experience.

Virtual Assistants & Chatbots: Use Webex Devices API to create virtual assistants, chatbots and other AI driven experiences with natural language capabilities.

Workflow Automation: Automate processes like recoding meetings, file transmission and follow-ups based on triggers and action using Webex APIs.

Analytics & Reporting: Utilize the Analytics API to gain insights into usage, allowing developers to create custom reports and dashboards for metrics.

1.4 How can we develop apps from Webex API?

Here are the main steps to develop applications with the Webex APIs:

Get Developer Account - Sign up on Webex's developer portal to get access to API documentation, SDKs and other developer resources.

Understand the APIs – Review the documentation for the Webex APIs they want to use. This provides details on endpoints, parameters and authentication.

Get API Credentials – The user needs to register their app in the developer portal to obtain credentials like client ID, client secret required for API access.

Choose a SDK – Webex provide SDKs in languages like JAVA, JavaScript. Pick an appropriate SDK for faster app development.

Make API Calls – In user app code, use the SDK to make API calls. Initialize the SDK client, set headers, parameters and call endpoints to achieve required functionality.

Handle Responses – The user must parse JSON responses from Webex APIs to handle success scenarios and error appropriately.

Test & Debug – The user should thoroughly test their code end-to-end and debug via logging, tracing to fix issues.

Publish the Integration – Once tested, the user can publish the app in the Webex App Hub to distribute it to clients.

Developing apps from the Webex API allowing user to create personalized solutions that meet specific their needs. This flexibility fosters innovation allowing users to create more personalized and effective collaboration experience. Integration with the Webex ensures compatibility and the support if the user community provides valuable collaboration. Moreover, potential business opportunities arise as user craft impactful and unique applications to address specific needs in their respective domains.

2. OBJECTIVES

Here are the objectives that are required to create a troubleshooting tool to investigate with Webex API:

Webex Token: Obtain a Webex token to authenticate and authorize API requests on behalf of the user serving as a key element for accessing Webex services.

Connection Testing: Enable users to test connection with the Webex server within the application. Provide a confirmation message upon successful connection, ensuring the server's correct functionality.

User Information Display: Implement functionality to retrieve and display user details from the Webex API including Displayed Name, Nickname and Emails.

Room Listing Feature: Allow users to see a list of rooms they have in their Webex account. Show details like room.id, room.title, when it was created and the last activity.

Room Creation Option: Let users make a new room in Webex. Ask them to give the room a name and confirm when them room is successfully created.

Messaging Room Functionality: Help users pick a room and send message in it. Confirm when the message is sent successfully and make sure the process is easy for users.

User Interaction Flow: Make sure users can easily move through the app. Show them clear choices, guide them through their chosen actions and let them go back to the main menu if needed.

Effective Error Handling: If something goes wrong, handle it well. If the app can't communicate with Webex of the user puts in the wrong info, show a helpful and clear message. Make sure users know what's happening and what to do next.

These objectives describe the tasks needed to create a complete troubleshooting tool for Webex. This requirement will empower users to review their information during conference sessions and carry out different tasks within the Webex system.

3. LITERATURE REVIEW

3.1 Description of Software and Programming Language

Python: Phyton is a versatile, high-level programming language known for its readability and simplicity. It is widely used for various applications including web development and automation. In this assignment, we use Phyton 3 which is the latest version of the Phyton.

Webexteamssdk (Webex API Phyton SDK): Officially Python SDK for the Webex API. It simplifies the process of interacting with Webex services by providing a Pythonic Interface for making API calls and handling authentication.

Requests Library: Designed for simplifying HTTP requests. It provides straightforward API for making various types of HTTP requests, such as, GET, POST, PUT and more. This library abstracts the complexities of working with HTTP, making it easy for developers to interact with web services and APIs.

Webbrowser: Used to automatically open a web browser. This feature is handy for users who want to log in into their Webex account and grant permission to the application. By utilizing this module, developers can make the process smooth and user-friendly ensuring an easy and secure way for users to access Webex services.

3.2 Dependencies

Dependencies leans on the Webex API, acting as the tool's link to the Webex universe. Then, the user steps in their Webex token number, unlocking the tools special abilities. If the tool uses a web browser, it relies on it being present and working smoothly. Lastly, the tool's way of interact to the user, known as the user interface, needs specific components to function without a hitch. So, dependencies are like the trusty buddies the tool relies on for a successful mission.

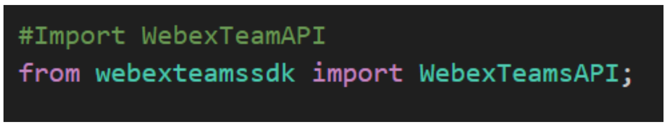
3.3 Architecture between Webex API and Application

The architecture of the troubleshooting tool involves a client application interacting with the Webex API to perform various tasks. The application prompts the user to input their Webex token for authentication. A menu driven interface offers options such as testing the connection, displaying user information, listing rooms, creating rooms and sending messages. Each option corresponds to specific Webex API endpoints, facilitating communication between the application and the Webex severs. The user's input triggers HTTP requests to the API and the application processes the responses and displaying relevant information. This architecture ensures a seamless and user-friendly experience for troubleshooting Webex-related issues.

3.4 How application is connected to Webex API

The application is connected to the Webex API through the integration of HTTP requests to specific endpoints, utilizing the Webex token or authentication. The Webex API is accessed for functionalities such as connecting testing, displaying user information, listing rooms, creating new rooms and sending messages. The application dynamically forms requests for each option, parsing API responses to present relevant information to the user. The secure handling to the Webex token ensures authorized communication and the successful execution for each option during testing server as evidence of the effective of the effective connection between the application and the Webex API, facilitating real-time troubleshooting of user details during conferring sessions.

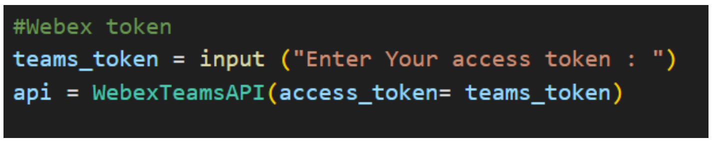
4. DEVELOPMENT OF THE APPLICATION



```
#Import WebexTeamAPI
from webexteamssdk import WebexTeamsAPI;
```

Figure 4.1

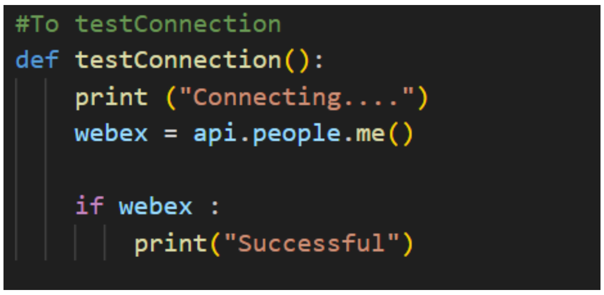
- Figure 4.1, the code imports the WebexTeamsAPI class from the webexteamssdk library. This class is like a tool that provides the necessary functions for the user Python script to communicate with the Webex API. It acts as a connection, making it easy for user script and the Webex collaboration platform to understand each other.



```
#Webex token
teams_token = input ("Enter Your access token : ")
api = WebexTeamsAPI(access_token= teams_token)
```

Figure 4.2

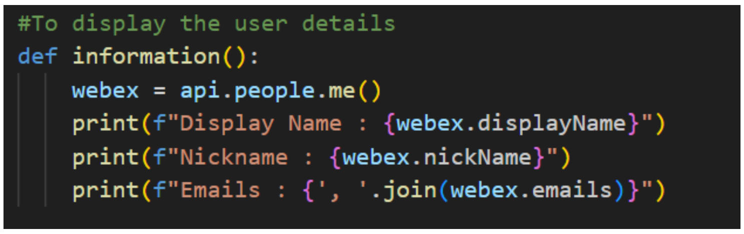
- In Figure 4.2, the code requests the user to input their Webex access token, which is like a special key needed to use Webex API. Once the user enters the token using the input function, the code creates a connection to the Webex API by making a WebexTeamsAPI object with the token. This connection is essential for the Python script to communicate with the Webex API.



```python
#To testConnection
def testConnection():
    print ("Connecting....")
    webex = api.people.me()

    if webex :
        print("Successful")
```

Figure 4.3

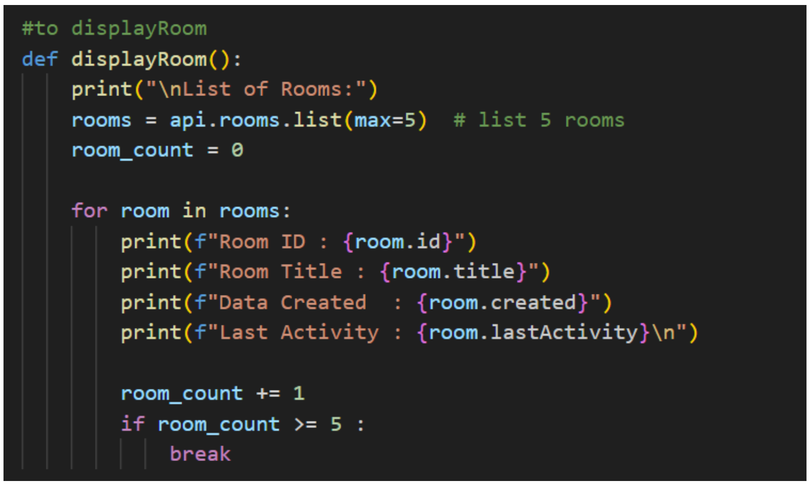
- The testConnection function which corresponds to option 0, checks if the Webex API is working and can be reached. It starts by showing a message saying **"Connecting…."** And the uses **api.people.me()** to try fetching information about the user who logged in. If this works, it prints a message to confirm that the connection is successful. Essentially, this function helps make sure the application can communicate to the Webex API correctly in the beginning.



```python
#To display the user details
def information():
    webex = api.people.me()
    print(f"Display Name : {webex.displayName}")
    print(f"Nickname : {webex.nickName}")
    print(f"Emails : {', '.join(webex.emails)}")
```

Figure 4.4

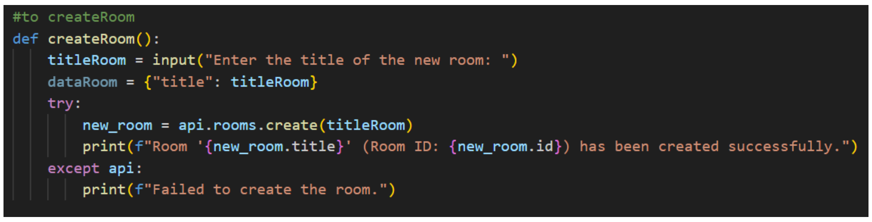
- def information() gets the user's details by using me() method and shows their display name, nickname and emails.



```python
#to displayRoom
def displayRoom():
    print("\nList of Rooms:")
    rooms = api.rooms.list(max=5)  # list 5 rooms
    room_count = 0

    for room in rooms:
        print(f"Room ID : {room.id}")
        print(f"Room Title : {room.title}")
        print(f"Data Created  : {room.created}")
        print(f"Last Activity : {room.lastActivity}\n")

        room_count += 1
        if room_count >= 5 :
            break
```

Figure 4.5

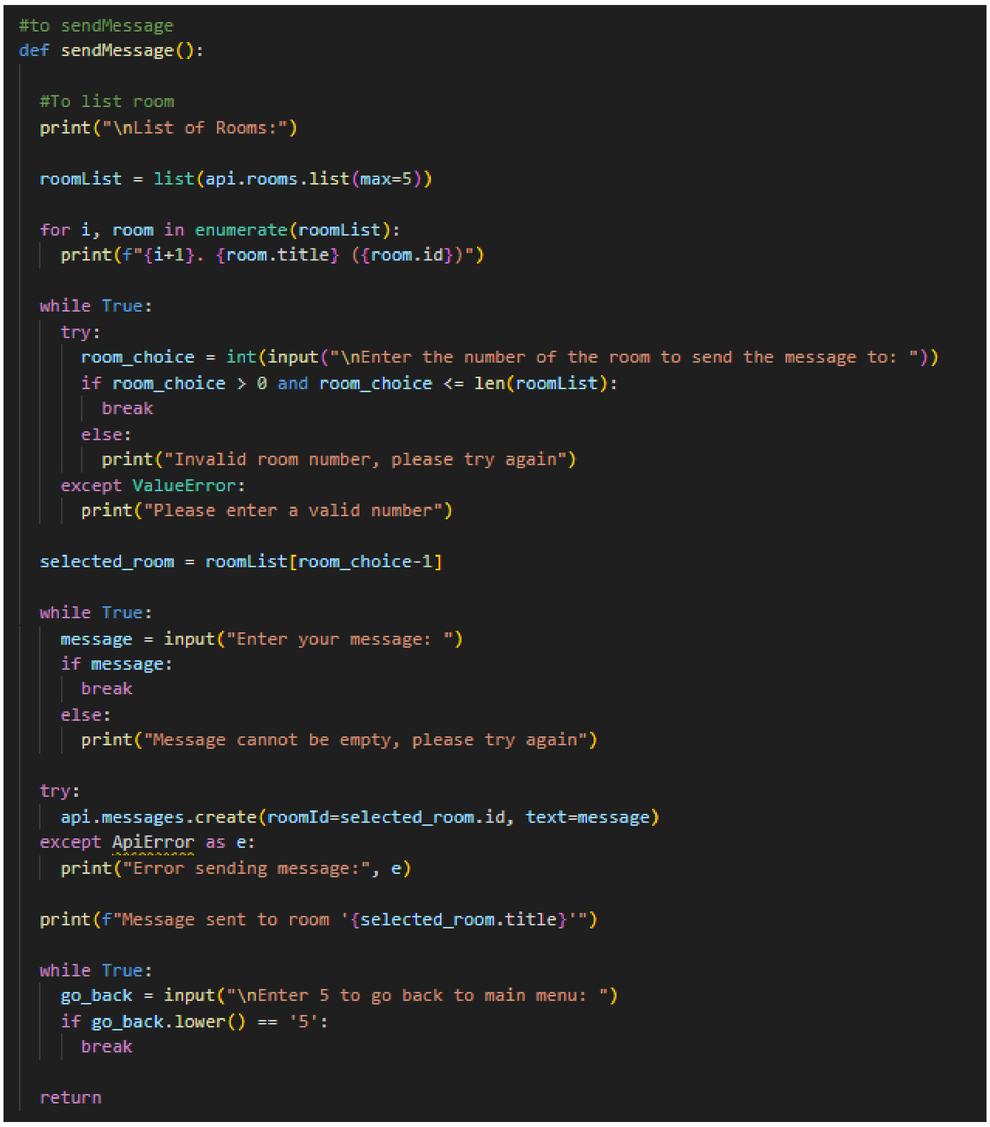
- In Figure 4.5, the displayRoom function uses the **listRooms** function to get a list of rooms. After that, it goes through each room and prints information like Room ID, Room Title, Date Created and Last Activity for maximum of 5 rooms.



```python
#to createRoom
def createRoom():
    titleRoom = input("Enter the title of the new room: ")
    dataRoom = {"title": titleRoom}
    try:
        new_room = api.rooms.create(titleRoom)
        print(f"Room '{new_room.title}' (Room ID: {new_room.id}) has been created successfully.")
    except api:
        print(f"Failed to create the room.")
```

Figure 4.6

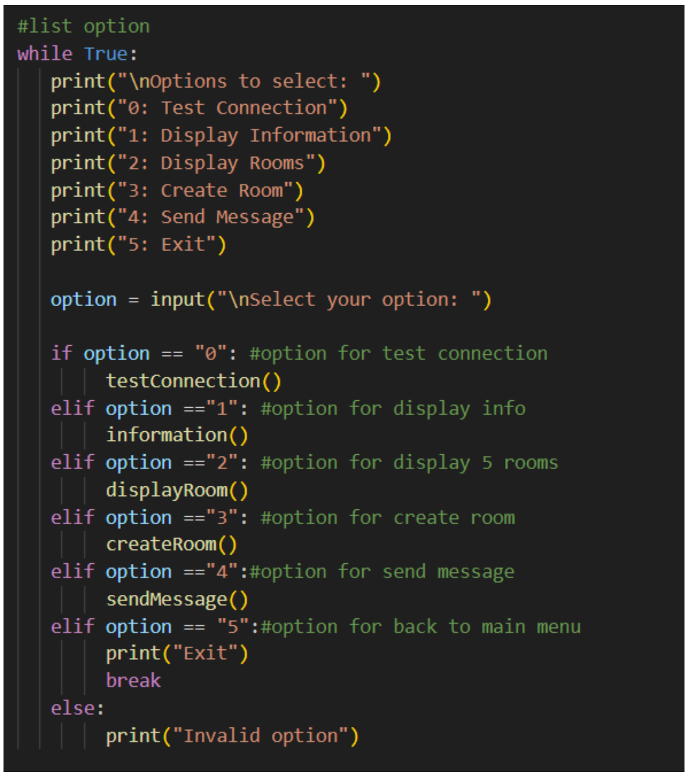
- The createRoom function asks the user to input in the title for a new room. The it tries to make the room using the **createRoom()** method. If its success, the program shows a message saying the room was created successfully, mentioning the room's title and ID. But if there's a problem, it prints out an error message.



```python
#to sendMessage
def sendMessage():

    #To list room
    print("\nList of Rooms:")

    roomList = list(api.rooms.list(max=5))

    for i, room in enumerate(roomList):
        print(f"{i+1}. {room.title} ({room.id})")

    while True:
        try:
            room_choice = int(input("\nEnter the number of the room to send the message to: "))
            if room_choice > 0 and room_choice <= len(roomList):
                break
            else:
                print("Invalid room number, please try again")
        except ValueError:
            print("Please enter a valid number")

    selected_room = roomList[room_choice-1]

    while True:
        message = input("Enter your message: ")
        if message:
            break
        else:
            print("Message cannot be empty, please try again")

    try:
        api.messages.create(roomId=selected_room.id, text=message)
    except ApiError as e:
        print("Error sending message:", e)

    print(f"Message sent to room '{selected_room.title}'")

    while True:
        go_back = input("\nEnter 5 to go back to main menu: ")
        if go_back.lower() == '5':
            break

    return
```

Figure 4.7

- The listRooms function gets a list rooms with a maximum of 5 using the **room.list()** method. The **sendMessage** function lets the user pick a room, type a message and send it. It uses the **listRooms** method to get the rooms, checks if the user's input is valid and then uses the **message.created()** methos to send the message. It also deals with errors like wrong room numbers or input values.

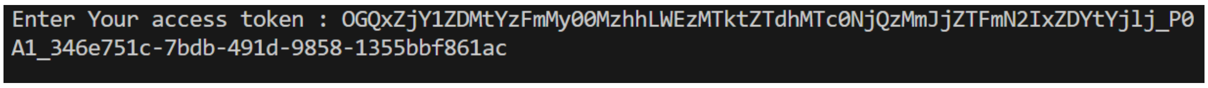


```
#list option
while True:
    print("\nOptions to select: ")
    print("0: Test Connection")
    print("1: Display Information")
    print("2: Display Rooms")
    print("3: Create Room")
    print("4: Send Message")
    print("5: Exit")

    option = input("\nSelect your option: ")

    if option == "0": #option for test connection
        testConnection()
    elif option =="1": #option for display info
        information()
    elif option =="2": #option for display 5 rooms
        displayRoom()
    elif option =="3": #option for create room
        createRoom()
    elif option =="4":#option for send message
        sendMessage()
    elif option == "5":#option for back to main menu
        print("Exit")
        break
    else:
        print("Invalid option")
```

Figure 4.8

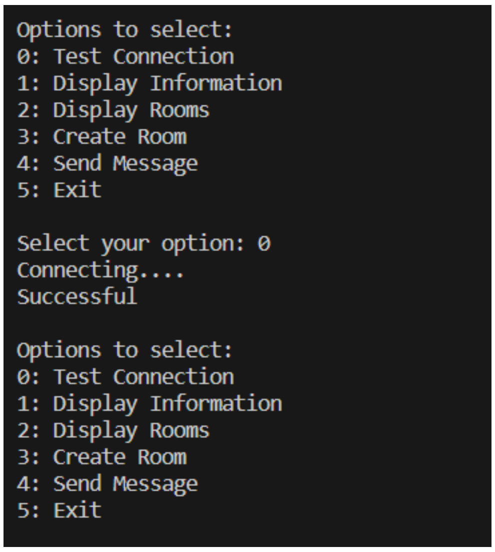
- In Figure 4.8, the script creates a continuous loop that displays a menu of options to the user. Depending on what the user picks, it calls the matching function. It lets users exit by choosing option 5. If someone enters an option that doesn't make sense, it shows a message telling them to pick a valid option.

5. TESTING OF THE APPS



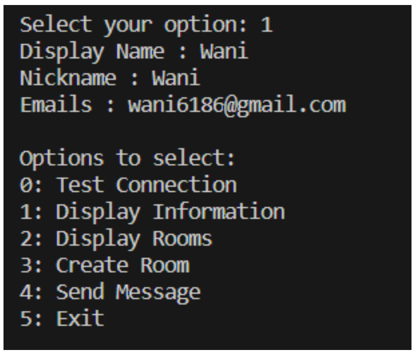
```
Enter Your access token : OGQxZjY1ZDMtYzFmMy00MzhhLWEzMTktZTdhMTc0NjQzMmJjZTFmN2IxZDYtYjlj_P0
A1_346e751c-7bdb-491d-9858-1355bbf861ac
```

- The user is asked to input in their Webex access token.



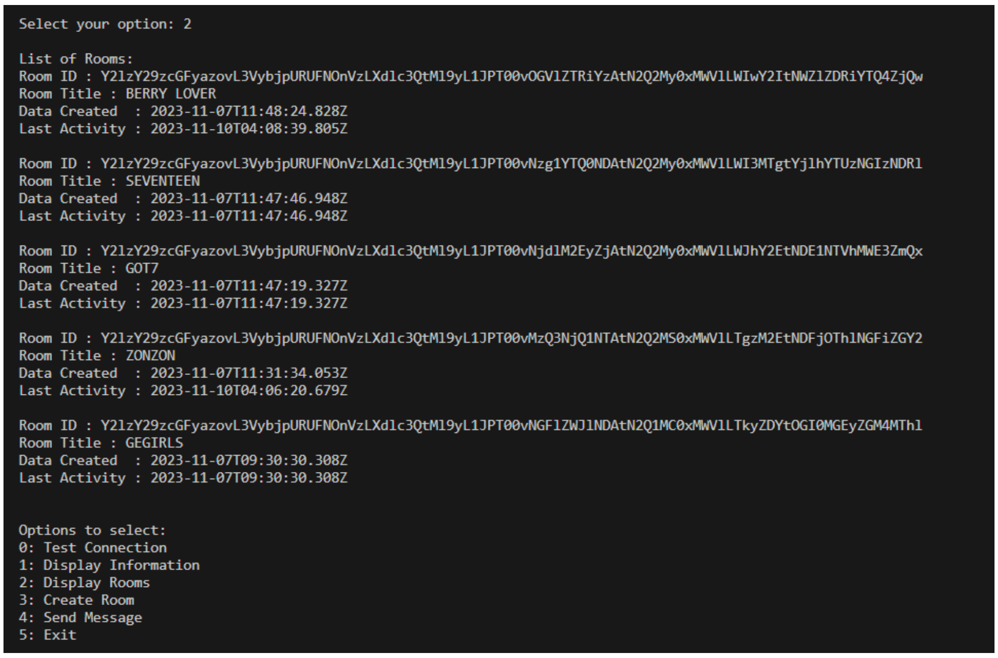
```
Options to select:
0: Test Connection
1: Display Information
2: Display Rooms
3: Create Room
4: Send Message
5: Exit

Select your option: 0
Connecting....
Successful

Options to select:
0: Test Connection
1: Display Information
2: Display Rooms
3: Create Room
4: Send Message
5: Exit
```

- The scripts display a menu with options numbered from 0 to 5.
- Users are given the options to choose from 0 to 5.
- If the user selects 0, the scripts print **"Connecting…."** And then **"Successful",** confirming a successful connection to Webex.
- The user is presented with the option again.



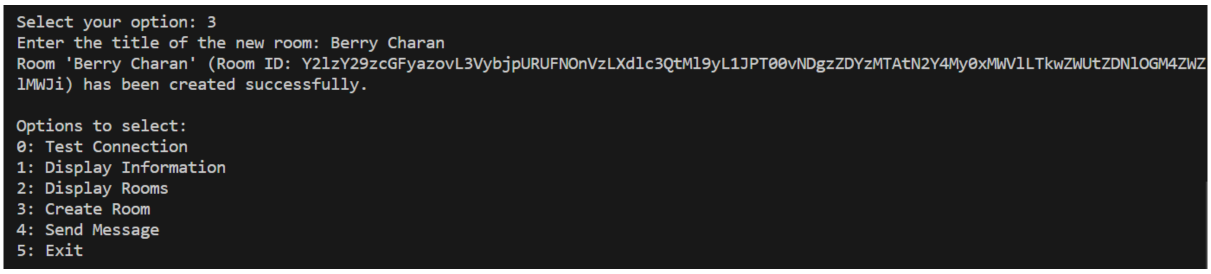
```
Select your option: 1
Display Name : Wani
Nickname : Wani
Emails : wani6186@gmail.com

Options to select:
0: Test Connection
1: Display Information
2: Display Rooms
3: Create Room
4: Send Message
5: Exit
```

- The user chooses option 1 to display their Webex information.
- The script responds by printing details like display name, nickname and emails linked to the user's Webex account.
- After displaying information, the user is presented with the option again, allowing them to make another selection.



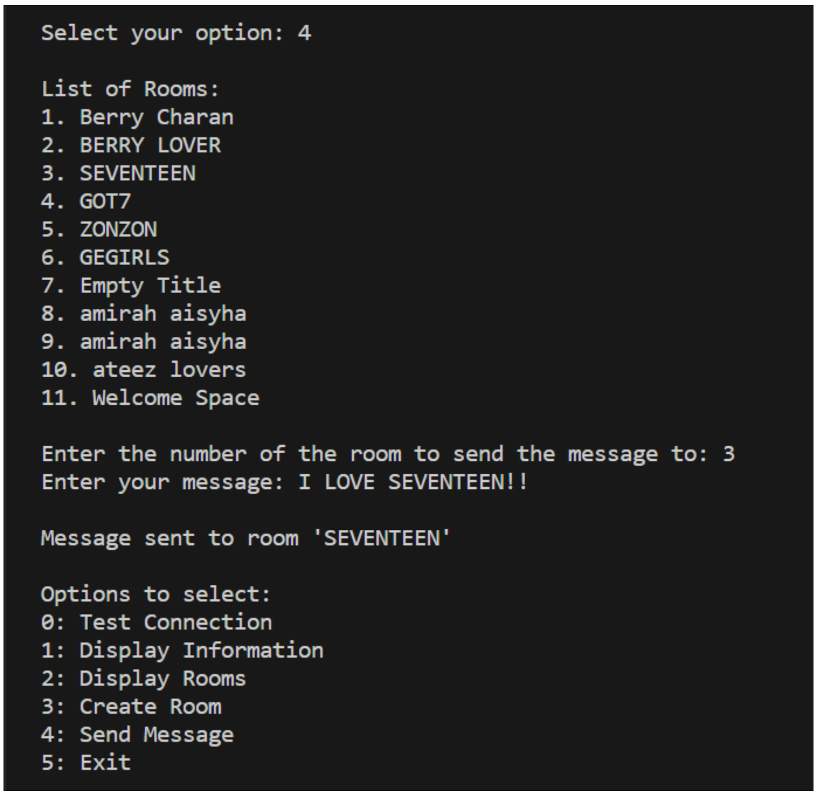
```
Select your option: 2

List of Rooms:
Room ID : Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vOGVlZTRiYzAtN2Q2My0xMWVlLWIwY2ItNWZlZDRiYTQ4ZjQw
Room Title : BERRY LOVER
Data Created  : 2023-11-07T11:48:24.828Z
Last Activity : 2023-11-10T04:08:39.805Z

Room ID : Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNzg1YTQ0NDAtN2Q2My0xMWVlLWI3MTgtYjlhYTUzNGIzNDRl
Room Title : SEVENTEEN
Data Created  : 2023-11-07T11:47:46.948Z
Last Activity : 2023-11-07T11:47:46.948Z

Room ID : Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNjdlM2EyZjAtN2Q2My0xMWVlLWJhY2EtNDE1NTVhMWE3ZmQx
Room Title : GOT7
Data Created  : 2023-11-07T11:47:19.327Z
Last Activity : 2023-11-07T11:47:19.327Z

Room ID : Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vMzQ3NjQ1NTAtN2Q2MS0xMWVlLTgzM2EtNDFjOThlNGFiZGY2
Room Title : ZONZON
Data Created  : 2023-11-07T11:31:34.053Z
Last Activity : 2023-11-10T04:06:20.679Z

Room ID : Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNGFlZWJlNDAtN2Q1MC0xMWVlLTkyZDYtOGI0MGEyZGM4MThl
Room Title : GEGIRLS
Data Created  : 2023-11-07T09:30:30.308Z
Last Activity : 2023-11-07T09:30:30.308Z


Options to select:
0: Test Connection
1: Display Information
2: Display Rooms
3: Create Room
4: Send Message
5: Exit
```

- The user chooses for option 2 to display a list rooms.
- The script responds by printing information about the first 5 rooms, providing details such as Room ID, Room Title, Date Created and Last Activity.
- Following the display of room information, the user is given the option again, enabling them to make another selection.

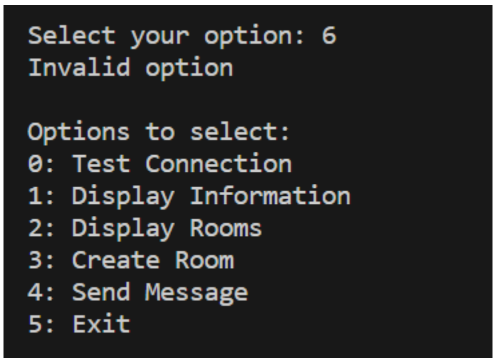


```
Select your option: 3
Enter the title of the new room: Berry Charan
Room 'Berry Charan' (Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNDgzZDYzMTAtN2Y4My0xMWVlLTkwZWUtZDNlOGM4ZWZ
lMWJi) has been created successfully.

Options to select:
0: Test Connection
1: Display Information
2: Display Rooms
3: Create Room
4: Send Message
5: Exit
```

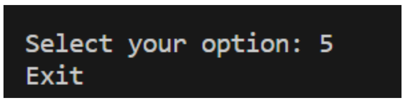
- The user picks option 3 to create a new room
- The script prompts user to input the title for the new room.
- Following the user’s input, the script prints a success message containing the newly created room’s title and ID
- After successful creation, the user is presented with the option again.


```
Select your option: 4

List of Rooms:
1. Berry Charan
2. BERRY LOVER
3. SEVENTEEN
4. GOT7
5. ZONZON
6. GEGIRLS
7. Empty Title
8. amirah aisyha
9. amirah aisyha
10. ateez lovers
11. Welcome Space

Enter the number of the room to send the message to: 3
Enter your message: I LOVE SEVENTEEN!!

Message sent to room 'SEVENTEEN'

Options to select:
0: Test Connection
1: Display Information
2: Display Rooms
3: Create Room
4: Send Message
5: Exit
```

- The user selects option 4 to send message.
- The script presents a list of rooms and asks the user to enter the number of the room where they want to send a message.
- The user inputs the message they want to send.
- After the user inputs the message, the script prints **"Message sent to room 'Room Title' ".**
- Following the successful message sending, the user is given the option again.


```
Select your option: 6
Invalid option

Options to select:
0: Test Connection
1: Display Information
2: Display Rooms
3: Create Room
4: Send Message
5: Exit
```

- The user enters an invalid option "6".
- In response to the invalid option, the script displays the message "Invalid option".
- Following the error message, the user is provided with the option to choose a different option.



```
Select your option: 5
Exit
```

- The user chooses option 5 to exit the application.
- The script responds by printing “Exit”.
- Following the exit message, the loop terminates, bringing an end to the script.

6. CONCLUSION

I’ve successfully built a helpful troubleshooting tool for Webex API, achieving our objectives. I can securely access Webex by obtaining and using tokens. I enable myself to check the server connection, view my information and manage room successfully. Besides, I can list, create and send message in rooms without any hassle. The app ensures a smooth experience with straightforward choices. Robust error handling guides me in case of any issues, making it user-friendly. Overall, I’ve encouraged myself to troubleshoot and interact effortlessly with Webex services during conference sessions.

7. REFERENCE

- Cisco. (n.d.). *APIs - Getting Started*. Developer.webex.com. https://developer.webex.com/docs/getting-started

- S. Gillis, A. (n.d.). *What is Cisco Webex? Everything You Need to Know*. SearchUnifiedCommunications. https://www.techtarget.com/searchunifiedcommunications/definition/Cisco-Webex

- Cisco. (n.d.-b). *Rooms - Create a Room*. Developer.webex.com. Retrieved November 10, 2023, from https://developer.webex.com/docs/api/v1/rooms/create-a-room

- Cisco. (n.d.-b). *Reference - Rooms*. Developer.webex.com. https://developer.webex.com/docs/api/v1/rooms

GITHUB LINK

https://github.com/CharanBerryLover/Wani