



Assignment Cover– be sure to keep a copy of all work submitted

To be completed by student – PLEASE PRINT CLEARLY

Name: MUHAMMAD IDHAM BIN MUHAMMAD ZAINI		
ID Number: AM2207011654		
Lecturer SIR MOHD AKMAL BIN AZMER	Lab group / Tutorial group / Tutor (if applicable)	
Course and Course Code CC101 – SWC2373	Submission Date: 10 NOVEMBER 2023	
Assignment No. / Title EMERGING TECHNOLOGY – API ASSIGNMENT	Extension & Late submission: Allowed / <u>Disallowed</u>	
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		
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 piece of work is my/our own. I/we consent to appropriate storage of our work for checking to ensure that there is no plagiarism/ academic cheating. Signature:		
This section may be used for feedback or other information		

Contents

1. Introduction.....	3
1.1 What is Webex?.....	3
1.2 What is Webex API?	3
1.3 How can third party apps can be developed with Webex API?	3
1.4 How can we develop app from Webex API?.....	4
2. Objectives.....	5
3. Literature Review:.....	6
3.1 Description of Software and Programming Language:	6
3.2 Dependencies:.....	6
3.3 Application Architecture:.....	6
3.4 Connection to Webex API:	6
4. Development of the application	7
5. Testing of the apps	11
6. Conclusion	14
7. Reference	15

1. Introduction

Webex is a platform developed by Cisco that transforms the way people collaborate and communicate. It provides features like video conferencing, online meetings, and team messaging, making it an essential tool for today's remote work and team collaboration. On the other hand, Webex API opens a world of possibilities for developers. It's a set of tools and protocols that allow third-party applications to seamlessly integrate with Webex services. This means you can create custom apps that leverage Webex's features, such as managing meetings, sending messages, and accessing user data. In this report, we'll explore how to develop these third-party apps, from registration on the Cisco Developer Portal to secure authentication and the actual code development process.

1.1 What is Webex?

Webex are created by Cisco and serves as a highly popular cloud-based platform for communication and teamwork. It provides an extensive range of services, encompassing video conferencing, virtual meetings, team chats, file sharing, and more. The essence of Webex lies in simplifying communication and collaboration, allowing individuals and teams to seamlessly connect and work together, regardless of their physical location. It's a vital asset for remote work, virtual meetings, and enhancing collaborative efforts among teams.

1.2 What is Webex API?

Webex offers a bunch of Application Programming Interfaces (APIs) that empower developers to craft third-party applications capable of interacting with Webex services. These APIs give developers the freedom to seamlessly incorporate Webex features into their own applications, taking the user experience up a notch and expanding the platform's capabilities.

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

Creating third-party applications with Webex API revolves around tapping into the provided APIs to unlock a wide array of Webex platform features and functions. This apps from the capability to create and manage meetings, interact with chat messages, access user

information, and beyond. These applications can web-based, mobile, or desktop apps, catering to the developer's specific choices and needs.

1.4 How can we develop app from Webex API?

To develop an application using Webex API, you need to follow a series of steps:

1. **Access Webex API:** You begin by securing access to the Webex API through application registration on the Cisco Developer Portal.
2. **User Authentication:** The next step is to authenticate your application, enable to interact with Webex resources on behalf of users.
3. **API Integration:** Consult the API documentation to grasp the available endpoints and their functionalities, understanding how they can be utilized in your application.
4. **Development:** Create your application by writing code in a programming language. Like Python, JavaScript, depending on the platform you're targeting.
5. **Testing:** Validate your application's functionality by testing it within a Webex sandbox environment to ensure it behaves as intended.
6. **Deployment:** Once satisfied with the testing, deploy your application on your chosen platform, making it accessible to users.
7. **User Management:** Implement user management features, permissions, and roles as necessary to govern user interactions effectively.
8. **Continual Improvement:** Always maintaining and updating your application as new features are added or changes are made to the Webex API.

2. Objectives

- **Obtaining Webex Token:** First, we need to get our Webex token. This token is like the key to accessing the Webex API, allowing us to talk to it on behalf of the user.
- **Creating a User-Friendly Interface:** Develop application that's user-friendly. It'll kindly ask the user for their Webex token and present them with clear menu options for different things they can do.
- **Checking the Connection:** Users should be able to test the connection to the Webex server. Display with a "successful" message if everything's working smoothly.
- **Displaying User Info:** Allow user to display their Webex information, things like their Name, Nickname, and Email addresses.
- **Listing Available Rooms:** Provide the capability to list of user rooms in Webex, showing stuff like the Room ID, Room Title, date created, and last activity.
- **Creating New Rooms:** Enable users to have the option to make a new room in Webex. They'll give it a name, and when it's created, we'll display an acknowledgement upon successful room creation.
- **Sending Messages to Rooms:** Allow users to choose a room from their list and send message to that room. Once the message sent, display an acknowledgement upon successful room creation.
- **User Interaction and Flow:** Ensure the user's interaction flow by provide clear menu choices, guide them through their selected choices, and offering the ability to go back to the main menu.
- **Handling Errors:** Implement error-handling procedures to resolve possible complications, such as unsuccessful API requests or inaccurate user input, and provide users with informative error messages.

These objectives provide an insight into task required to creating a tool that empowers users to troubleshoot their Webex experience while ensuring it remains user-friendly and readily available to assist when things deviate from the plan.

3. Literature Review:

3.1 Description of Software and Programming Language:

For this assignment, we've opted for Python as our programming language. Python is versatile and comes with a wealth of libraries that make it a great fit for working with the Webex API. To create a user-friendly web-based interface, we're using Flask, which is a web framework that complements Python nicely.

3.2 Dependencies:

Our application relies on a few key Python libraries. We use 'requests' to handle the nitty-gritty of sending and receiving HTTP requests with the Webex API. And, for that all-important user authentication, we're using OAuth libraries to keep things secure. These dependencies are crucial for ensuring a smooth interaction with the Webex platform.

3.3 Application Architecture:

Picture our application as the friendly go-between connecting you and the Webex API. When you click a button or make a request in the app, it takes your input, talks to the Webex API through the magic of HTTP requests, and then handles what comes back. It's essentially the interpreter between you and the Webex API, making sure everything runs smoothly.

3.4 Connection to Webex API:

To get to the Webex API, our app follows a secure process. First, it asks you for your Webex token and uses that to authentication. This token is like your special key that lets the app access your Webex stuff. With that access token, the app can do things like test the

connection, get your details user information, list your rooms, create new rooms, and send messages to a room.

This setup ensures that our app acts as a bridge between you and the Webex API, giving you a friendly way to use Webex services while keeping everything safe and authenticated.

4. Development of the application

Figure 4.1: This is the access token, like a special key. It's used to verify your identity when you interact with the Webex system. Think of it as a digital password that lets you access features in Webex.

```
access_token = 'OTFkNmY1ZmEtMzBhNi00YzYxLTkxMjgtYjQwN2E5MGxNDFjZDBkZmI0NzEtNTEz_P0A1_d753f5ed-fc6f-453b-b80d-9eed175c690b'
```

Figure 4.1:

Figure 4.2: The application is ability to test the connection with the Webex server (Option 0). It sends a request to the Webex API, asking for user information using an HTTP GET request. To prove who you are, it attaches your Webex token in the Authorization header. If the response from the server has a status code of 200, it means the connection is good to go. If not, it shows an error message.

```
# Function to test connection with Webex server
def test_connection():
    url = 'https://webexapis.com/v1/people/me'
    headers = {
        'Authorization': 'Bearer {}'.format(access_token)
    }

    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        print("Connection with Webex server is successful.")
    else:
        print("Connection with Webex server failed.")

    input("Press Enter to return to the menu...")
```

Figure 4.2

Figure 4.3: This part is the functionality for display your user information (Option 1). It's like sending a request to the Webex API, asking for your details using an HTTP GET request. Then, it shows your displayed name, nickname, and your email addresses. If something goes wrong with the request, it tells you with an error message.

```
# Function to get user details
def get_user_details():
    url = 'https://webexapis.com/v1/people/me'
    headers = {
        'Authorization': 'Bearer {}'.format(access_token),
        'Content-Type': 'application/json'
    }

    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        user_data = response.json()
        print("User Information: ")
        print(f"Display Name: {user_data['displayName']}")
        print(f"Nickname: {user_data['nickName']}")
        print(f"Emails: {' , '.join(user_data['emails'])}")
    else:
        print("Failed to fetch user details.")

    input("Press Enter to return to the menu...")
```

Figure 4.3

Figure 4.4: This part is the functionality for display your list rooms (Option 2). It's like asking the Webex API for details about your rooms with an HTTP GET request. Then, it shows you info about the first five rooms, like their ID, title, creation date, and last activity. If something goes wrong with the request, it tells you with an error message.

```
# Function to list rooms
def list_rooms():
    url = 'https://webexapis.com/v1/rooms'
    headers = {
        'Authorization': 'Bearer {}'.format(access_token),
        'Content-Type': 'application/json'
    }
```



```

response = requests.get(url, headers=headers)
if response.status_code == 200:
    rooms = response.json()
    print("\nList of Rooms:")
    for i, room in enumerate(rooms['items']):
        print(f"Room {i + 1}:")
        print(f"Room ID: {room['id']}")
        print(f"Room Title: {room['title']}")
        print(f>Date Created: {room['created']}")
        print(f>Last Activity: {room['lastActivity']}")
        print()
    else:
        print("Failed to fetch room details.")

input("Press Enter to return to the menu...")

```

Figure 4.4

Figure 4.5: This part is the functionality for creating a new room (Option 3). It asks you to type in the name you want for the room. Then, it sends that name as part of the data in a message to the Webex API using an HTTP POST request. If the room is made successfully, it will display the group in your Webex application but if something goes wrong, it tells you with an error message.

```

# Function to create a room
def create_room():
    url = 'https://webexapis.com/v1/rooms'
    title = input("Enter the title for the new room: ")
    headers = {
        'Authorization': 'Bearer {}'.format(access_token),
        'Content-Type': 'application/json'
    }
    data = {
        "title": title
    }

    response = requests.post(url, json=data, headers=headers)
    if response.status_code == 200:
        room_data = response.json()
        print("Room created successfully!")
        print(f"Room ID: {room_data['id']}")
        print(f"Room Title: {room_data['title']}")
    else:
        print("Failed to create the room.")

    input("Press Enter to return to the menu...")

```

Figure 4.5

Figure 4.6: This part is the functionality about sending messages to a room (Option 4). First, it gets the list of rooms and shows them to you. You pick a room by entering a number. Then, you type in the message you want to send. The application sends it up that message and sends it off to the Webex API using an HTTP POST request. If it goes through, it displays, "Message sent successfully!" But if something doesn't quite work, it tells you with an error message.

```
# Function to send a message to a room
def send_message():
    url = 'https://webexapis.com/v1/messages'
    headers = {
        'Authorization': 'Bearer {}'.format(access_token),
        'Content-Type': 'application/json'
    }

    response = requests.get('https://webexapis.com/v1/rooms', headers=headers)
    if response.status_code == 200:
        rooms = response.json()
        print("\nList of Rooms:")
        for i, room in enumerate(rooms['items']):
            print(f"Room {i + 1}:")
            print(f"Room ID: {room['id']}")
            print(f"Room Title: {room['title']}")
            print()

        room_index = int(input("Enter the room number to send a message (1-5): ")) - 1
        if room_index < 0 or room_index >= len(rooms['items']):
            print("Invalid room selection.")
            return

        room_id = rooms['items'][room_index]['id']
        message = input("Enter the message to send: ")
        data = {
            "roomId": room_id,
            "text": message
        }

    response = requests.post(url, json=data, headers=headers)
    if response.status_code == 200:
        print("Message sent successfully!")
    else:
        print("Failed to send the message.")

input("Press Enter to return to the menu...")
```

Figure 4.6

Figure 4.7: This part represents the main menu of an application. It continuously displays a list of options, such as testing the connection, viewing user information, listing rooms, creating a room, sending messages, and quitting the application.

```
while True:
    print("\nMain Menu:")
    print("0. Test Connection with Webex Server")
    print("1. Display User Information")
    print("2. List Rooms")
    print("3. Create a Room")
    print("4. Send Message to a Room")
    print("5. Quit")

    choice = input("Select an option (0/1/2/3/4/5): ")

    if choice == "0":
        test_connection()
    elif choice == "1":
        get_user_details()
    elif choice == "2":
        list_rooms()
    elif choice == "3":
        create_room()
    elif choice == "4":
        send_message()
    elif choice == "5":
        print("Exiting the application.")
        break
    else:
        print("Invalid option. Please select a valid option.")
```

Figure 4.7

5. Testing of the apps

To begin with, you need to pick a choice by entering a number. For instance, if you choose "0", the application will check the connection with the Webex server. If everything's good, it says the connection is successful. Then, you can press the "Enter" key to go back to the menu and make more choices. It's a way for you to tell the app what you want it to do.

```
Main Menu:
0. Test Connection with Webex Server
1. Display User Information
2. List Rooms
3. Create a Room
4. Send Message to a Room
5. Quit
Select an option (0/1/2/3/4/5): 0
Connection with Webex server is successful.
Press Enter to return to the menu...
```

Figure 5.1

Furthermore, you pick an option by entering a number. For instance, if you choose "1", the application will display your user information like Name, Nickname, Emails. After that, you can press the "Enter" key to go back to the menu. It's a way for you to find out information about your account in Webex.

```
Main Menu:
0. Test Connection with Webex Server
1. Display User Information
2. List Rooms
3. Create a Room
4. Send Message to a Room
5. Quit
Select an option (0/1/2/3/4/5): 1
User Information:
Display Name: Idham Idham
Nickname: Idham
Emails: muhammadidham168@gmail.com
Press Enter to return to the menu...
```

Figure 5.2

In Addition, you select an option by entering a number. If you choose "2", the application will display a list of your rooms. Once you have got the details like Room ID, Room Tittle, Date Created, and Last Activity. Then, you can press "Enter" to go back to the menu and make more choices. It's like a way to view information about the rooms you have in Webex.

```
Main Menu:
0. Test Connection with Webex Server
1. Display User Information
2. List Rooms
3. Create a Room
4. Send Message to a Room
5. Quit
Select an option (0/1/2/3/4/5): 2

List of Rooms:
Room 1:
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNbnVzLXdlc3QtMl9yL1JPT00vNmY2YmU3MTAtNDg5YS0xMwVlLWE3NTY2YzYzc2ZmNiZDM4
Room Title: Welcome Space
Date Created: 2023-09-01T07:37:42.657Z
Last Activity: 2023-10-09T02:58:14.608Z

Room 2:
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNbnVzLXdlc3QtMl9yL1JPT00vMzdiMzVmZAtNDU0Yi0xMwVlLWFjMDktZGIxNGE4MjMzYzgy
Room Title: Distinguish Gentleman
Date Created: 2023-08-28T02:33:05.443Z
Last Activity: 2023-10-28T03:04:40.515Z

Press Enter to return to the menu...
```

Figure 5.3

Moreover, you select an option by entering a number. If you choose "3," the application asks you to name your new room. Let's say you call it "GGWP". After that, they will create the room and says, "Room created successfully!". Then, you can press "Enter" to go back to the menu and make more choices. It's a way to create a new room in Webex and get details about it.

```
Main Menu:
0. Test Connection with Webex Server
1. Display User Information
2. List Rooms
3. Create a Room
4. Send Message to a Room
5. Quit
Select an option (0/1/2/3/4/5): 3
Enter the title for the new room: GGWP
Room created successfully!
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXd1c3QTMl9yL1JPT00vNDVjNGU4MDAtN2FhYy0xMwV1LWJjMzQTYmYzMmY4YmZiMzI1
Room Title: GGWP
Press Enter to return to the menu...
```

Figure 5.4

Last but not least, you choose an option by entering a number. If you pick "4," the app shows you a list of your rooms. After that, they ask you to pick a room by entering its number (1-5). Let's say you choose "1". It asks you to type a message. After you enter, it sends the message to the room and tells you, "Message sent successfully!" You can press "Enter" to go back to the menu and make more choices. It's a way to send messages within your Webex rooms.

```
Main Menu:
0. Test Connection with Webex Server
1. Display User Information
2. List Rooms
3. Create a Room
4. Send Message to a Room
5. Quit
Select an option (0/1/2/3/4/5): 4

List of Rooms:
Room 1:
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXd1c3QTMl9yL1JPT00vNDVjNGU4MDAtN2FhYy0xMwV1LWJjMzQTYmYzMmY4YmZiMzI1
Room Title: GGWP

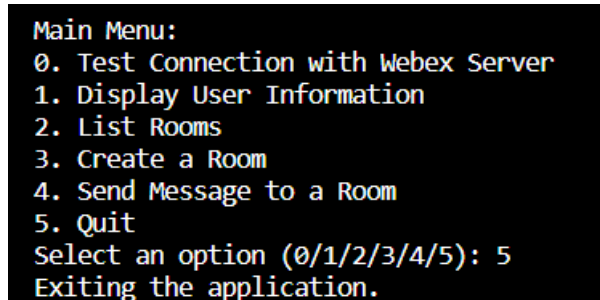
Room 2:
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXd1c3QTMl9yL1JPT00vNmY2YmU3MTAtNDg5YS0xMwV1LWE3NTETyY2YzYzc2ZmNiZDM4
Room Title: Welcome Space

Room 3:
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXd1c3QTMl9yL1JPT00vMzdiMzVmMzAtNDU0Yi0xMwV1LWFjMDktZGIxNGE4MjMzYzgy
Room Title: Distinguish Gentleman

Enter the room number to send a message (1-5): 1
Enter the message to send: FROM THE RIVER TO THE SEE PALESTINE WILL BE FREE!!! #FREE PALESTINE!!!
Message sent successfully!
Press Enter to return to the menu...
```

Figure 5.5

To Sum Up, you choose an option by entering a number. If you select "5," they politely say it's exiting, which means it's closing, and you're done using it. It's a way to leave the app when you're finished.



```
Main Menu:
0. Test Connection with Webex Server
1. Display User Information
2. List Rooms
3. Create a Room
4. Send Message to a Room
5. Quit
Select an option (0/1/2/3/4/5): 5
Exiting the application.
```

Figure 5.6

6. Conclusion

In this assignment, through the development of this troubleshooting tool using the Webex API, the objectives were successfully achieved. The tool effectively empowers users to troubleshoot their Webex experience and access essential features for conferencing sessions. With the accomplishment of these objectives, we've provided a valuable resource for our organization, streamlining the process of checking user details during conferencing sessions and enhancing the overall Webex experience. This project demonstrates the capability of leveraging the Webex API to create practical, user-centric solutions for our organization's needs.

In conclusion, we have successfully built a comprehensive troubleshooting tool that gives users simple access to Webex API features. This tool allows users to oversee and control their Webex account data during conferences, making our organization's operations more efficient. The user-friendly design ensures that users can confidently use the Webex platform for effective collaboration.

7. Reference

1. Webex (n.d.). Webex Meetings Essentials. [online] Webex. Available at: <https://www.webex.com/essentials/meetings.html> [Accessed 1 Nov. 2023].
2. developer.webex.com. (n.d.). Embedded Apps - Developer Guide. [online] Available at: <https://developer.webex.com/docs/embedded-apps-guide> [Accessed 1 Nov. 2023].
3. developer.webex.com. (n.d.). APIs - Getting Started. [online] Available at: <https://developer.webex.com/docs/getting-started>.
4. *Reference - messages* (no date) *Webex for Developers*. Available at: <https://developer.webex.com/docs/api/v1/messages> (Accessed: 04 November 2023).
5. *Reference - rooms* (no date) *Webex for Developers*. Available at: <https://developer.webex.com/docs/api/v1/rooms> (Accessed: 04 November 2023).

Link To Github: <https://github.com/ldhaizs/SWC2373--EMERGING-TECHNOLOGIES>