# UNIVERSITI POLY-TECH MALAYSIA

<table>
<tr><td colspan="3">Name(s): Muhammad ‘Arif Naqyyuddin Bin Abd Kahar</td></tr>
<tr><td colspan="3">ID Number(s): AM2207011677</td></tr>
<tr><td colspan="2">Lecturer : Mohd Akmal Bin Mohd Azmer</td><td>Lab group / Tutorial group / Tutor (if applicable)</td></tr>
<tr><td colspan="2">Course and Course Code :<br>SWC2373</td><td>Submission Date:<br>10th November 2023</td></tr>
<tr><td colspan="2">Assignment No. / Title :<br><br>API Assignment</td><td>Extension & Late submission:<br>Allowed / Disallowed</td></tr>
<tr><td>Assignment type:<br>Individual</td><td>% of Assignment Mark</td><td>Returning Date:</td></tr>
<tr><td colspan="3">Penalties:<br>1. 10% of the original mark will be deducted for every one-week period after the submission date<br>2. No work will be accepted after two weeks of the deadline<br>3. If you were unable to submit the coursework on time due to extenuating circumstances you may be eligible for an extension<br>4. Extension will not exceed one week</td></tr>
<tr><td colspan="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 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.<br><br>Signature(s):<br><br>Full Name: Muhammad ‘Arif Naqyyuddin Bin Abd Kahar</td></tr>
<tr><td colspan="3">This section may be used for feedback or other information</td></tr>
</table>

# Introduction

Cisco Webex is a collaboration platform that enables individuals and teams to digitally meet, collaborate, and communicate. Webex provides a collection of APIs (Application Programming Interfaces) that developers may use to integrate and enhance Webex capabilities into their own applications.

Third-party developers can use the Webex API to create apps that interact with Webex services such as retrieving user information, creating and managing rooms, and sending messages. This task entails creating a troubleshooting tool utilizing the Webex API that allows users to test connections, display user information, list rooms, create rooms, and send messages.

## Objective

The primary objectives of this assignment are as follows:

1. To test the connection with webex server. If successful, it will display acknowledgement. The user will have the option to go back to the menu.

2. To display his/her information such as Displayed Name, Nickname, Emails. The user will prompt back to the menu.

3. To display a list of FIVE(5) rooms. The information that will be displayed on each room are, Room ID, Room Title, date created and last activity. The user will prompt back to the menu.

4. To Create a room. The user will be prompted with the title of the room so that it can create a new room in webex. The user will prompt back to the menu. An acknowledgement will be shown if the room is able to create.

5. To Send message to a room. When chosen this option, it will display FIVE(5) rooms. The user will choose which room it would like to send messages to. The user then will type a message in the room. Upon successful message sent, it will acknowledge that the message has been sent. The user also has the option of going back to the menu page.

# Literature Review

**Cisco Webex and Webex API:**

Cisco Webex stands out as a top-tier collaboration platform, seamlessly woven into the fabric of contemporary business communication and virtual collaboration. It brings together a suite of tools, encompassing video conferencing, messaging, file sharing, and various collaborative features. Functioning as a cloud-based service, Webex empowers organizations to foster team connections and facilitate communication effortlessly, regardless of geographical boundaries.

The Webex API acts as a gateway, inviting developers to elevate the functionality of the Webex platform and seamlessly integrate it into their own applications. This API opens doors for third-party app development, allowing developers to leverage the robust capabilities of Webex within their unique solutions. By tapping into the Webex API, developers gain entry to a diverse set of endpoints, offering capabilities ranging from user and space management to retrieving messages and programmatically initiating meetings.

**Programming Language and Dependencies:**

For this project, we've opted for Python as the programming language of choice. Python's popularity in the development community stems from its readability, versatility, and extensive library support. Notably, Python's simplicity facilitates rapid development and easy maintenance, making it an ideal selection for projects with diverse requirements.

A pivotal dependency in this project is the requests library. This library streamlines the process of making HTTP requests, a foundational element when interacting with Webex API endpoints. By abstracting away the intricacies of HTTP communication, requests simplify integration with the Webex API, enabling the application to seamlessly send and receive data.

**Architecture and Connection to Webex API:**

The application's architecture adheres to a command-line interface (CLI) paradigm, prioritizing simplicity and user-friendliness. This design choice ensures that users can engage with the troubleshooting tool without the need for a complex graphical interface. Complementing the CLI architecture is the modular design of the code, with each functionality encapsulated within methods of the WebexTroubleshootingTool class. This modular approach enhances code maintainability and readability.

The connection to the Webex API is established through the use of an API token. The token, acquired from the user during the tool's initialization, is included in the headers of each HTTP request. This authentication method aligns with security best practices, ensuring that the tool possesses the necessary permissions to interact with the user's Webex environment.

# Development Of The Application

### Class Definition

import requests

class WebexTroubleshootingTool:

The code starts by importing the `requests` library, which is commonly used for making HTTP requests. Then, a class named `WebexTroubleshootingTool` is defined.

### Constructor Method (`__init__`)

```
    def __init__(self):
        self.base_url = "https://webexapis.com/v1"
        self.headers = {"Authorization": f"Bearer {input('Enter Webex Token: ')}"}
```

The `__init__` method initializes the class. It sets the `base_url` variable to the Webex API's base URL and prompts the user to input their Webex API token. The token is then included in the `headers` dictionary, which will be used in subsequent API requests.

### Methods for Different Options

Testing Connection (`test_connection`)

```
    def test_connection(self):
        try:
            response = requests.get(f"{self.base_url}/people/me", headers=self.headers)
            response.raise_for_status()
            print("Connection successful!")
        except requests.exceptions.HTTPError as err:
            print(f"Error: {err}")
        input("Press Enter to go back to the menu...")
```

The `test_connection` method sends a GET request to the `/people/me` endpoint to check the connection to the Webex server. If the request is successful (status code 2xx), it prints "Connection successful!" Otherwise, it catches any HTTP error and prints an error message.

Displaying User Information (`display_user_info`)

```
    def display_user_info(self):
        response = requests.get(f"{self.base_url}/people/me", headers=self.headers)
        user_info = response.json()
        print(f"Display Name: {user_info.get('displayName', 'N/A')}")
        print(f"Nickname: {user_info.get('nickName', 'N/A')}")
        print(f"Emails: {', '.join(user_info.get('emails', []))}")
        input("Press Enter to go back to the menu...")
```

The `display_user_info` method retrieves the user's information using a GET request to `/people/me` and prints the display name, nickname, and emails. It waits for the user to press Enter before returning to the main menu.

Listing Rooms (`list_rooms`)

```
    def list_rooms(self):
        response = requests.get(f"{self.base_url}/rooms", headers=self.headers)
        rooms = response.json().get("items", [])
        for room in rooms[:5]:
            print(f"Room ID: {room.get('id', 'N/A')}")
            print(f"Room Title: {room.get('title', 'N/A')}")
            print(f"Date Created: {room.get('created', 'N/A')}")
            print(f"Last Activity: {room.get('lastActivity', 'N/A')}")
            print("-" * 30)
        input("Press Enter to go back to the menu...")
```

The `list_rooms` method retrieves a list of rooms using a GET request to `/rooms` and prints information for the first five rooms, including room ID, title, date created, and last activity.

Creating a Room (`create_room`)

```
    def create_room(self):
        room_title = input("Enter the title for the new room: ")
        data = {"title": room_title}
        response = requests.post(f"{self.base_url}/rooms", headers=self.headers, json=data)
        if response.status_code == 200:
            print("Room created successfully!")
        else:
            print(f"Error: {response.status_code}")
        input("Press Enter to go back to the menu...")
```

The `create_room` method prompts the user to enter a title for a new room. It then sends a POST request to `/rooms` with the provided title. If successful, it prints "Room created successfully!" Otherwise, it prints an error message.

Sending a Message to a Room (`send_message`)

```
    def send_message(self):
        response = requests.get(f"{self.base_url}/rooms", headers=self.headers)
        rooms = response.json().get("items", [])[:5]
        for i, room in enumerate(rooms):
            print(f"{i + 1}. {room.get('title', 'N/A')}")
        room_index = int(input("Choose a room to send a message to (1-5): ")) - 1
        room_id = rooms[room_index].get('id', None)
        if room_id:
            message = input("Enter the message to send: ")
            data = {"roomId": room_id, "text": message}
                    response = requests.post(f"{self.base_url}/messages", headers=self.headers,
json=data)
            if response.status_code == 200:
                print("Message sent successfully!")
            else:
                print(f"Error: {response.status_code}")
        else:
            print("Invalid room selection.")
        input("Press Enter to go back to the menu...")
```

The `send_message` method retrieves a list of rooms, displays them, and prompts the user to choose a room to send a message to. It then prompts for the message content and sends a POST request to `/messages` to send the message to the selected room. It provides feedback on the success or failure of the message sending process.

**Main Menu Execution (`main_menu`)**

```
    def main_menu(self):
        while True:
            print("Main Menu:")
            print("0. Test Connection")
            print("1. Display User Info")
            print("2. List Rooms")
            print("3. Create Room")
            print("4. Send Message to Room")
            print("5. Exit")
            choice = input("Enter your choice (0-5): ")

            if choice == "0":
                self.test_connection()
            elif choice == "1":
                self.display_user_info()
            elif choice == "2":
                self.list_rooms()
            elif choice == "3":
                self.create_room()
            elif choice == "4":
                self.send_message()
            elif choice == "5":
                print("Exiting the program. Goodbye!")
                break
            else:
                print("Invalid choice. Please try again.")
```

The `main_menu` method serves as the main control loop of the application. It continuously displays the menu options, takes user input, and calls the respective methods based on the chosen option. The loop continues until the user chooses to exit the program.

**Running the Application (`if __name__ == "__main__":`)**

```
if __name__ == "__main__":
    tool = WebexTroubleshootingTool()
    tool.main_menu()
```

This block of code checks if the script is being run as the main program (not imported as a module). If so, it creates an instance of the `WebexTroubleshootingTool` class and invokes the `main_menu` method, initiating the execution of the application.

# Testing Of The Application

```
Main Menu:
0. Test Connection
1. Display User Info
2. List Rooms
3. Create Room
4. Send Message to Room
5. Exit
Enter your choice (0-5): 0
Connection successful!
Press Enter to go back to the menu...
Main Menu:
0. Test Connection
1. Display User Info
2. List Rooms
3. Create Room
4. Send Message to Room
5. Exit
Enter your choice (0-5): 1
Display Name: Muhammad 'Arif Naqyyuddin Abd Kahar
Nickname: Muhammad 'Arif Naqyyuddin
Emails: kl2207011677@student.kuptm.edu.my
Press Enter to go back to the menu...
Main Menu:
0. Test Connection
1. Display User Info
2. List Rooms
3. Create Room
4. Send Message to Room
5. Exit
Enter your choice (0-5): 2
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vMTg0YzFlNjAtN2YzNS0xMWVlLTllZWMtYjMwMWJkNTc5Y2Rh
Room Title: hensem
Date Created: 2023-11-09T19:20:51.270Z
Last Activity: 2023-11-09T19:21:22.393Z
------------------------------
Press Enter to go back to the menu...
Main Menu:
0. Test Connection
1. Display User Info
2. List Rooms
3. Create Room
4. Send Message to Room
5. Exit
Enter your choice (0-5): 3
Enter the title for the new room: Kacak
Room created successfully!
Press Enter to go back to the menu...
Main Menu:
0. Test Connection
1. Display User Info
2. List Rooms
3. Create Room
4. Send Message to Room
5. Exit
Enter your choice (0-5): 2
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vMjcyOGIwMDAtN2YzYi0xMWVlLWJhMjgtYzUwZGVlMjhhMzJi
Room Title: Kacak
Date Created: 2023-11-09T20:04:13.184Z
Last Activity: 2023-11-09T20:04:13.184Z
```

```
Main Menu:
0. Test Connection
1. Display User Info
2. List Rooms
3. Create Room
4. Send Message to Room
5. Exit
Enter your choice (0-5): 2
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vZTdjZWI2ZjAtN2Y0MS0xMWVlLWE2MDMtN2JhNTEzMDhjN2Uy
Room Title: Kuat
Date Created: 2023-11-09T20:52:33.375Z
Last Activity: 2023-11-09T20:52:33.375Z
------------------------------
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vZGQ0MGQ3NDAtN2Y0MS0xMWVlLTlmODAtODNkNTRjMDk5NDJh
Room Title: Berani
Date Created: 2023-11-09T20:52:15.668Z
Last Activity: 2023-11-09T20:52:15.668Z
------------------------------
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vZDFjN2Q0NDAtN2Y0MS0xMWVlLWExZmQtND1kNjRhZjY4N2Rj
Room Title: Gagah
Date Created: 2023-11-09T20:51:56.420Z
Last Activity: 2023-11-09T20:51:56.420Z
------------------------------
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vMjcyOGIwMDAtN2YzYi0xMWVlLWJhMjgtYzUwZGVlMjhhMzJi
Room Title: Kacak
Date Created: 2023-11-09T20:04:13.184Z
Last Activity: 2023-11-09T20:04:13.184Z
------------------------------
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vMTg0YzFlNjAtN2YzNS0xMWVlLTllZWMtYjMwMWJkNTc5Y2Rh
Room Title: hensem
Date Created: 2023-11-09T19:20:51.270Z
Last Activity: 2023-11-09T20:04:41.416Z
------------------------------
Press Enter to go back to the menu...
Main Menu:
0. Test Connection
1. Display User Info
2. List Rooms
3. Create Room
4. Send Message to Room
5. Exit
Enter your choice (0-5): 5
Exiting the program. Goodbye!
```

# Conclusion

In conclusion, the WebexTroubleshootingTool has successfully realized its primary objectives outlined for this assignment. Through rigorous development and testing, the tool now stands as a robust and user-friendly application, providing valuable functionalities that enhance the user experience within the Webex collaboration platform.

Firstly, the objective to test the connection with the Webex server has been achieved. The tool seamlessly verifies the connection, and upon success, graciously acknowledges the user. The intuitive design allows users the flexibility to return to the main menu, ensuring a smooth and responsive interaction.

Secondly, the tool effectively displays personalized user information, including Displayed Name, Nickname, and Emails. This feature enriches user awareness and engagement with their Webex profile, fostering a more personalized and user-centric experience. The option to prompt back to the menu ensures a seamless transition for the user.

Thirdly, the objective to display a list of five rooms, presenting key information such as Room ID, Room Title, date created, and last activity, has been successfully accomplished. This functionality provides users with a quick snapshot of their collaborative spaces, facilitating efficient navigation and management. The user-friendly design ensures a straightforward return to the main menu.

The fourth objective, creating a room, has been realized with the tool prompting the user for the room title and subsequently acknowledging the successful creation. This empowers users to dynamically generate rooms tailored to their needs, contributing to a more flexible and adaptive Webex environment.

Lastly, the tool enables users to send messages to selected rooms, displaying a list of five rooms and allowing users to choose the destination. Upon successful message delivery, the tool acknowledges the user, affirming effective communication. The option to return to the menu provides users with a convenient pathway to continue their interactions.

## Reference

Cisco Webex Developer Portal: https://developer.webex.com/

Python requests library documentation: https://docs.python-requests.org/en/latest/