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 :
SWC2373</td><td>Submission Date:
10th November 2023</td></tr>
<tr><td colspan="2">Assignment No. / Title :

API Assignment</td><td>Extension & Late submission:
Allowed / Disallowed</td></tr>
<tr><td>Assignment type:
Individual</td><td>% of Assignment Mark</td><td>Returning Date:</td></tr>
<tr><td colspan="3">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</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.

Signature(s):

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 allows individuals and teams to meet, collaborate, and communicate virtually. Webex provides a set of APIs (Application Programming Interfaces) that enable developers to integrate and extend the functionality of Webex into their own applications.

The Webex API allows third-party developers to create applications that can interact with Webex services, such as retrieving user information, creating and managing rooms, and sending messages. This assignment involves developing a troubleshooting tool using the Webex API, providing users with the ability 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 develop a troubleshooting tool using the Webex API to check user details during conferencing sessions.

2. To implement a user-friendly interface with options to test the connection, display user information, list rooms, create rooms, and send messages.

3. To conduct thorough testing to ensure the functionality and reliability of the application.

4. To provide a comprehensive report detailing the development process, testing procedures, and outcomes.

Literature Review

Cisco Webex and Webex API:

Cisco Webex is a leading collaboration platform that has become integral to modern business communication and virtual collaboration. It offers a suite of tools encompassing video conferencing, messaging, file sharing, and other collaborative features. As a cloud-based service, Webex empowers organizations to connect teams and facilitate communication seamlessly, irrespective of geographical locations.

The Webex API serves as a gateway for developers to extend the functionality of the Webex platform and integrate it into their own applications. This API opens up a wealth of possibilities for third-party app development, enabling developers to harness the power of Webex within their custom solutions. By interfacing with the Webex API, developers gain access to a rich set of endpoints that provide capabilities ranging from managing users and spaces to retrieving messages and initiating meetings programmatically.

Programming Language and Dependencies:

The chosen programming language for this project is Python. Python's popularity in the development community is attributed to its readability, versatility, and extensive set of libraries. In particular, Python's simplicity allows for rapid development and easy maintenance, making it an ideal choice for projects with varied requirements.

A critical dependency in this project is the requests library. This library simplifies the process of making HTTP requests, a fundamental aspect when interacting with Webex API endpoints. By abstracting away the complexities of HTTP communication, requests facilitate a smooth integration with the Webex API, enabling the application to seamlessly send and receive data.

Architecture and Connection to Webex API:

The architecture of the application follows a command-line interface (CLI) paradigm. This decision prioritizes simplicity and ease of use, ensuring that users can interact with the troubleshooting tool without the need for a complex graphical interface. The CLI architecture is complemented by 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, obtained from the user during the initialization of the tool, is included in the headers of each HTTP request. This method of authentication adheres to security best practices, ensuring that the tool has 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
```

```
------------------------------
Room ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtM19yL1JPT00vMTg0YzFlNjAtN2YzNS0xMWVlLTllZWMtYjMwMWJkNTc5Y2Rh
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...4
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): 4
1. Kacak
2. hensem
Choose a room to send a message to (1-5): 2
Enter the message to send: Terlalu kacak
Message sent 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): 5
Exiting the program. Goodbye!
```

Conclusion

In conclusion, the development of the WebexTroubleshootingTool represents a successful endeavor in leveraging the Webex API to enhance user experiences within the context of conferencing sessions. This comprehensive Python application provides users with a versatile set of functionalities, ranging from testing the connection to the Webex server to creating rooms and sending messages.

The implementation adheres to best practices in software development, embracing modularity, error handling, and an intuitive command-line interface. The codebase showcases the effective utilization of the `requests` library for seamless communication with the Webex API, ensuring that users can effortlessly navigate through the application's features.

Through the testing phase, the application demonstrated resilience in handling various scenarios, from successful API calls to gracefully managing errors and invalid inputs. The structured testing approach ensured that each component of the application, from connection testing to room creation and message sending, was rigorously evaluated.

The development process provided insights into the seamless integration of third-party APIs, emphasizing the importance of understanding API documentation and designing applications with user interactions at the forefront. The code's reliance on the Webex API opens the door to potential future enhancements and additional features as the Webex platform evolves.

The deployment of the application to the student's GitHub repository not only showcases proficiency in version control but also provides a valuable resource for future reference and collaboration. The transparent and well-documented nature of the code, coupled with the comprehensive report, ensures that both technical and non-technical stakeholders can grasp the application's purpose, functionality, and development intricacies.

In essence, the WebexTroubleshootingTool stands as a testament to the student's ability to apply theoretical knowledge to practical scenarios, demonstrating proficiency in Python programming, API integration, and software development best practices. As technology continues to evolve, this project serves as a foundational stepping stone in the journey of creating sophisticated and impactful applications that meet the needs of modern collaborative environments.

Reference

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

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