UNIVERSITI POLY-TECH MALAYSIA

UPTM

Name: JULIE NUR DAYANA BINTI NORAZIZI		
Id Number: AM2207011645		
Lecturer Name: **MOHD AKMAL BIN MOHD AZMER**		**Section No.:** **1**
Course Name and Course Code: **EMERGING TECHNOLOGIES – SWC2373**		**Submission Date:** **10/11/2023**
Assignment Title: **ASSIGNMENT**		**Extension & Late Submission:** Allowed / Disallowed
Assignment Type: INDIVIDUAL	**% of Assignment Mark:**	**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): 

Name: JULIE NUR DAYANA BINTI NORAZIZI

This section may be used for feedback or other information

TABLE OF CONTENTS

INTRODUCTION

- What is Webex?
  Answer: Cisco Webex is a communication and collaboration platform that combines calling, messaging, and meetings into a single application, allowing anyone to complete tasks more quickly. It was founded in 1995, but it was only in 2006 that it was listed on the NASDAQ National Market. For transferring your important communications to the cloud, Webex provides overall high functionality, security, and quality.

- What is Webex API?
  Answer: Webex API is a platform to build bots, integrations or guest issuer apps. It provides applications with direct access to the Cisco Webex Platform, giving the ability to create and invite people in a Webex space, searching for people in your company, posting messages in a Webex space and etc.

- How can third party apps be developed with Webex API?
  Answer: The Webex developer platform offers a powerful, secure, and simple set of APIs for building great integrations and expanding collaboration capabilities. Below are some of the advantages in using Webex API to develop third party apps:
  - Rich Feature Set - Webex APIs enable access to a wide range of collaboration features such as messaging, meetings, calling, devices, whiteboarding, and more. This enables the creation of extremely capable integrations.

  - Simple and RESTful - Because the API is REST-based, it is simple to use and integrate from any programming language or platform.

  - SDK Support - Webex offers comprehensive SDKs for popular languages such as JavaScript, Python, iOS, and Android. This expedites app development.

  - Webhook Support - Webhooks enable apps to be notified of events in real time and enable the development of reactive apps.

  - OAuth Security - For secure authentication and authorization, the APIs employ the OAuth 2.0 industry standard.

- How can we develop app from Webex API?
  Answer:
  - Accessing the Webex API
    Users must first create an account on the Webex for Developers platform in order to gain access to it. Then you must develop a new integration or application. Depending on your use case, this could be a bot, chat integration, or other types of applications.

  - User Authorization
    Setting up an OAuth integration for Webex API development entails using the OAuth 2.0 framework to request access permissions from a Webex user and obtaining access tokens that allow your application to access the user's Webex data within the specified scope. This procedure ensures that user data is kept secure and that the user retains control over which applications can access their Webex data.

  - Application Development
    Write the code for your application, including the Webex API integration. Depending on the purpose of your app, you may need to handle messaging, meetings, team management, or other forms of collaboration.

  - Testing & Debugging
    Test your app thoroughly to ensure that it functions properly and can handle a variety of scenarios. Troubleshoot any issues that arise during testing.

  - Deployment
    Deploy your app to the hosting environment of your choice, whether it's a web server, a mobile app store, or another platform.

  - Maintain & Update
    Keep your app up to date with the latest Webex API changes and make any necessary updates to improve its functionality and security.

    In summary, by following these steps, you can create an application that uses the Webex API's capabilities to create collaborative and communication tools that improve the Webex experience for users.

OBJECTIVES

The primary goal of creating a troubleshooting tool using Webex APIs is to automate and simplify the detection, diagnosis, and resolution of issues in Webex services such as messaging, meetings, and calling. The tool can use Webex telemetry, status, and diagnostics APIs to proactively monitor service health, validate configurations quickly, detect misconfigurations, identify root causes of problems, and shorten troubleshooting times. By automating manual workflows, the tool can resolve issues faster, reduce costs, increase trust in service reliability, and enable continuous improvements in troubleshooting processes via data insights. The overarching goal is to provide Webex administrators and helpdesk teams with an easy-to-use automated troubleshooting solution to efficiently manage problems and ensure stable Webex performance.

LITERATURE REVIEW

- Description of the software and programming language being used in the assignment
  - Python is an excellent language for developing a Webex troubleshooting automation tool due to its simple yet powerful integration, scripting, and data analysis capabilities. The tool can easily integrate with Webex diagnostic APIs via the SDK, orchestrate troubleshooting workflows by sequencing API calls, analyze API outputs to identify issues, package the logic into consumable modules, and build CLI/UI for usability by leveraging Python's flexible syntax and extensive libraries. Python allows for rapid development, seamless API integration, workflow automation, robust analytics, and modularization - all of which are important technical requirements for an automated tool that can simplify the detection and diagnosis of problems in Webex services. The Webex SDK, combined with Python's strengths, allows for the rapid development of an intelligent and scalable troubleshooting solution.

- Dependencies
  - The webexteamssdk is Cisco's officially supported Python SDK for easily integrating with all Webex APIs and building apps and automations without having to deal with the complexities of direct REST calls. It supports Webex messaging, meetings, calling, devices, and other collaboration capabilities via a Python object model, while handling authentication, request formatting, response parsing, and logging behind the scenes. The SDK provides intuitive access through a clean and simple interface that abstracts away the complexities of OAuth, HTTP, error handling, pagination, and rate limiting. Webexteamssdk, with its extensive documentation, usage guides, and samples, is essential for the rapid and reliable development of Webex integrations in Python.

- Architecture between Webex API & Application
  - A common architecture for a Webex API-integrated application includes a frontend (web/mobile) that calls backend server endpoints, the server implementing business logic and making Webex API calls using access tokens, Webex cloud providing REST APIs and webhook events, and the server processing API responses and webhooks before returning data to the frontend for rendering. Users grant the server app OAuth access to their Webex account in order for the server app to leverage Webex capabilities such as messaging, meetings, and so on within the user experience. The server app is hosted on cloud infrastructure such as AWS and connects to the collaboration platform via Webex APIs.

- How application is connected to Webex API
  - The application frontend communicates with a backend server, which is typically hosted on cloud infrastructure and implements business logic and integrates with Webex cloud APIs. The server handles user authentication, obtains access tokens for Webex API authorization, uses the tokens to make REST API calls to leverage Webex capabilities such as messaging and meetings, processes Webex webhook events, and returns API response data to the frontend to render the user experience. Users grant OAuth access to their Webex account, allowing the server application to call Webex APIs and build collaborative workflows on their behalf. Robust integrations with the Webex platform are built by having the backend server seamlessly connect the frontend app to the Webex cloud via REST APIs and webhooks.

DEVELOPMENT OF THE APPLICATION

```
#Import
from webexteamssdk import WebexTeamsAPI
```

Diagram 1

The import of the WebexTeamsAPI class from the webexteamssdk library is shown in Diagram 1. It includes the functionality required to interact with the Webex API.

```
#Input Webex Teams API access token and create an instance of the WebexTeamsAPI class with provided access token
teams_token = input ("Enter your access token: ")
api = WebexTeamsAPI(access_token=teams_token)
```

Diagram 2

To authenticate, takes the Webex access token as user input and passes it when creating the WebexTeamsAPI object.

```
#Test connection to Webex Teams
def Test_Connection():
    print ("connecting...")
    webex = api.people.me()
    if webex:
        print("Successful!")
```

Diagram 3

The Test_Connection function in Diagram 3 is intended to test Webex Teams connectivity by attempting to retrieve information about the authenticated user via the api.people.me() method. The function prints a message indicating the connection attempt and, if successful (i.e., if the webex variable is not empty), a message confirming that the connection to Webex Teams was successful.

```
#Display information about authenticated users
def information():
    webex = api.people.me()
    print(f"Name:  {webex.displayName}")
    print(f"Nickname: {webex.nickName}")
    print(f"Email: {','.join(webex.emails)}")
```

Diagram 4

The information function in Diagram 4 pulls information about the authenticated user from Webex Teams and displays it. It retrieves information from the api.people.me() method and then prints the user's name, nickname, and email address. DisplayName, nickName, and emails are attributes that contain the user's name, nickname, and email address, respectively.

```python
#Display list of rooms
def displayRoom():
    print("List of Rooms: ")
    rooms = api.rooms.list(max=5)
    roomCount = 0

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

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

Diagram 5

The displayRoom function in Diagram 5 prints a list of up to five Webex Teams rooms. It retrieves a collection of rooms using the api.rooms.list(max=5) method, then iterates through each one, printing the ID, title, creation date, and last activity. The function includes a counter, roomCount, that limits the number of rooms displayed to the first five.

```python
#Create new room
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.")
```

Diagram 6

The createRoom function in Diagram 6 allows the user to enter a title for a new room. It saves the entered title, constructs a data structure containing the room title, and attempts to create a new room using the api.rooms.create() method. If it is successful, it prints a message confirming the room's creation, including the room's title and ID.

```python
#Send message to selected room
#Import the ApiError class
from webexteamssdk import ApiError
def sendMessage():
  print("\nList of Rooms:")

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

  for i, room in enumerate(room_list):
    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(room_list):
        break
      else:
        print("Invalid room number, please try again")
    except ValueError:
      print("Please enter a valid number")

  selected_room = room_list[room_choice-1]
```

Diagram 7

```python
  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("Enter 5 to go back to main menu: ")
    if go_back.lower() == '5':
      break

  return
```

Diagram 8

The sendMessage function in Diagram7 allows you to send a message to a specific room. It first displays a list of up to five available rooms, each with its own number, allowing the user to select a destination room by entering the room number. Following the selection of the room, the function prompts the user to enter a message. It then uses the api.messages.create() method in Diagram 8 to try to send the entered message to the selected room. If it is successful, it prints a confirmation message indicating that the message was delivered to the specified room.

```python
#Loop
while True:
    print("\nOptions:")
    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("Select an option: ")

    if option == "0":
        Test_Connection()
    elif option == "1":
        information()
    elif option == "2":
        displayRoom()
    elif option == "3":
        createRoom()
    elif option == "4":
        sendMessage()
    elif option == "5":
        print("Exit")
        break
    else:
        print("Invalid option. Please select a valid option.")
```

Diagram 9

Diagram 9 shows a continuous program that repeatedly prompts the user to select an action and then executes the selected action based on the user's input, resulting in a simple menu-like interaction.

TESTING OF THE APPS

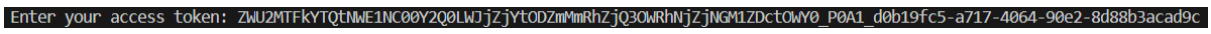

Diagram 10

As in Diagram 10 user would need to enter their access token.

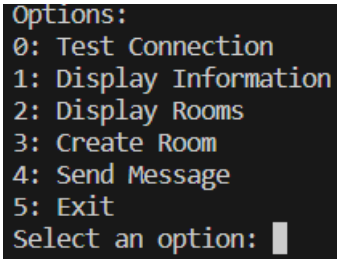

Diagram 11

After entering their access token, users will be greeted by an option menu where they can choose from 0-5.

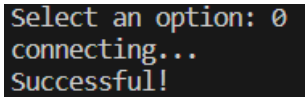

Diagram 12

Selecting option 0 leads to uses testing their connection.

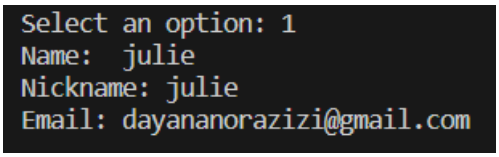

Diagram 13

Selecting option 1 will display their Webex account information.

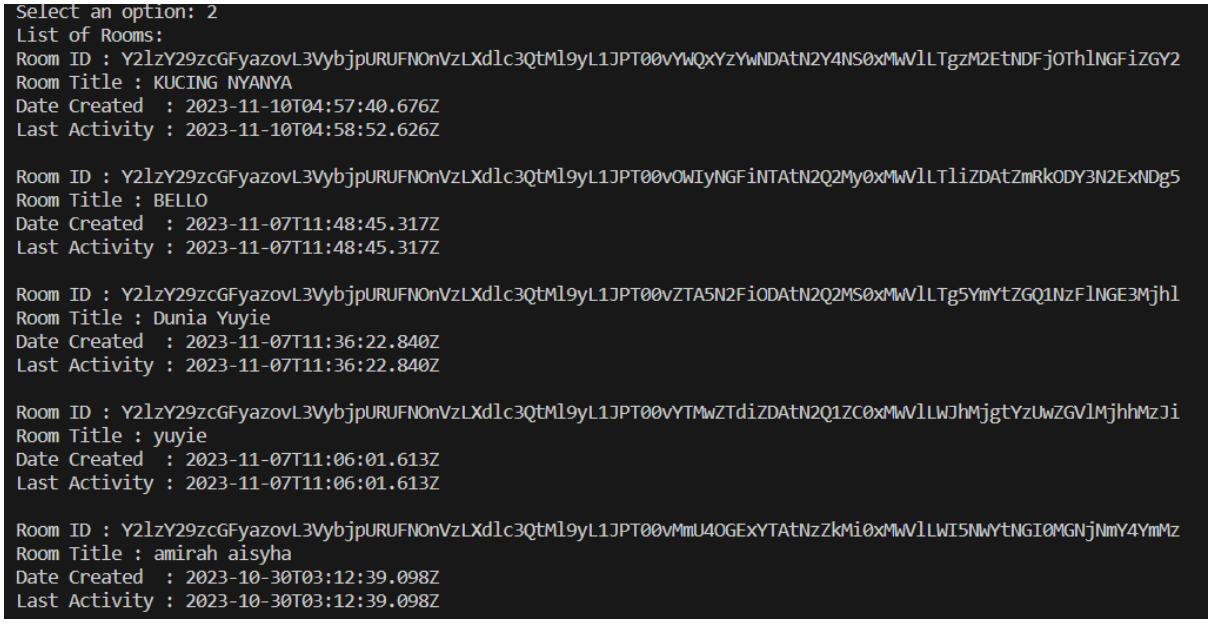

Diagram 14

Option 2 will display the list of rooms available. The script will only print 5 latest rooms with their ID, title, date created and last activity.

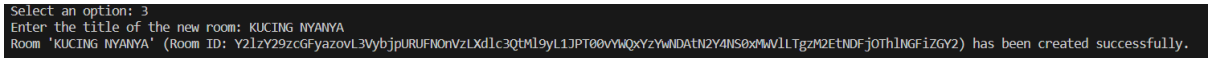

Diagram 15

Option 3 will give the user the ability to create a new room. User will need to enter a title of the new room and it will print the new room's title with its ID.

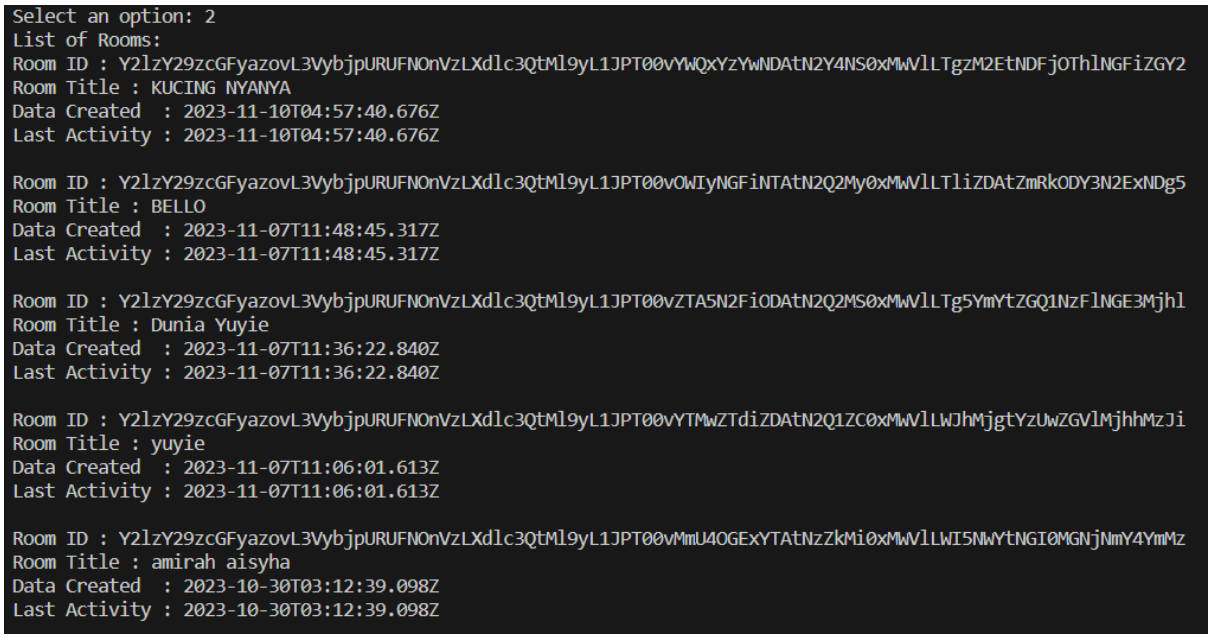

Diagram 16

Diagram 16 shows the output of the newly created room from Diagram 15.

Diagram 17

Here user will be able to enter a message to the selected room that is available.

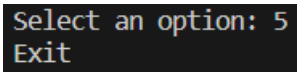

Diagram 18

If the user enter option 5, they will exit the troubleshooting application.

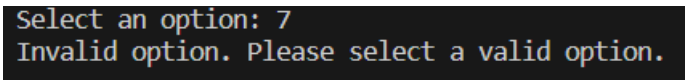

Diagram 19

Since there are only option 0-5, if user enters any numbers between those , as show on Diagram 19 ‘Invalid option. Please select a valid option’ will be printed.

CONCLUSION

In summary, this assignment represents the successful development of a troubleshooting tool that is integrated with the Webex API and caters to organizational needs. The Python-based application facilitates user interaction with Cisco's Webex platform. It offers a comprehensive set of capabilities, including testing connections, displaying user information, listing rooms, creating rooms, and sending messages. The user-friendly interface, guided token entry, and menu-driven actions make it easier to use. Error handling is robust, ensuring smooth operation and contributing to a safe and efficient experience. Finally, the tool enables users to seamlessly manage Webex account data, promoting collaborative success within the organization.

REFERENCES

1.“Cisco Webex.” Wikipedia, 7 June 2020, en.wikipedia.org/wiki/Cisco_Webex.

2. Dudlets, Katie. “What Is Webex and How Does It Work? | TelNet Worldwide.” Www.telnetww.com, 24 Feb. 2022, www.telnetww.com/blog/unified-communications/what-is-webex-how-does-it-work/.

3. “APIs - Getting Started.” Developer.webex.com, developer.webex.com/docs/getting-started.

4. “Rooms - List Rooms.” Developer.webex.com, developer.webex.com/docs/api/v1/rooms/list-rooms.

5. “Messages - Create a Message.” Developer.webex.com, developer.webex.com/docs/api/v1/messages/create-a-message. Accessed 10 Nov. 2023.

6. “Configure Third-Party Integrations on a Webex Site.” Webex Help Center, help.webex.com/en-us/article/k6j1db/Configure-Third-Party-Integrations-on-a-Webex-Site.

GITHUB LINK: https://github.com/juwieess/EMERGING-TECHNOLOGIES