

JAN HENDRIX B. NAVALES

BSIT 3rd Year

API Midterm Exam

Midterm Exam: Python API Integration and Data Generation

Total Points: 50

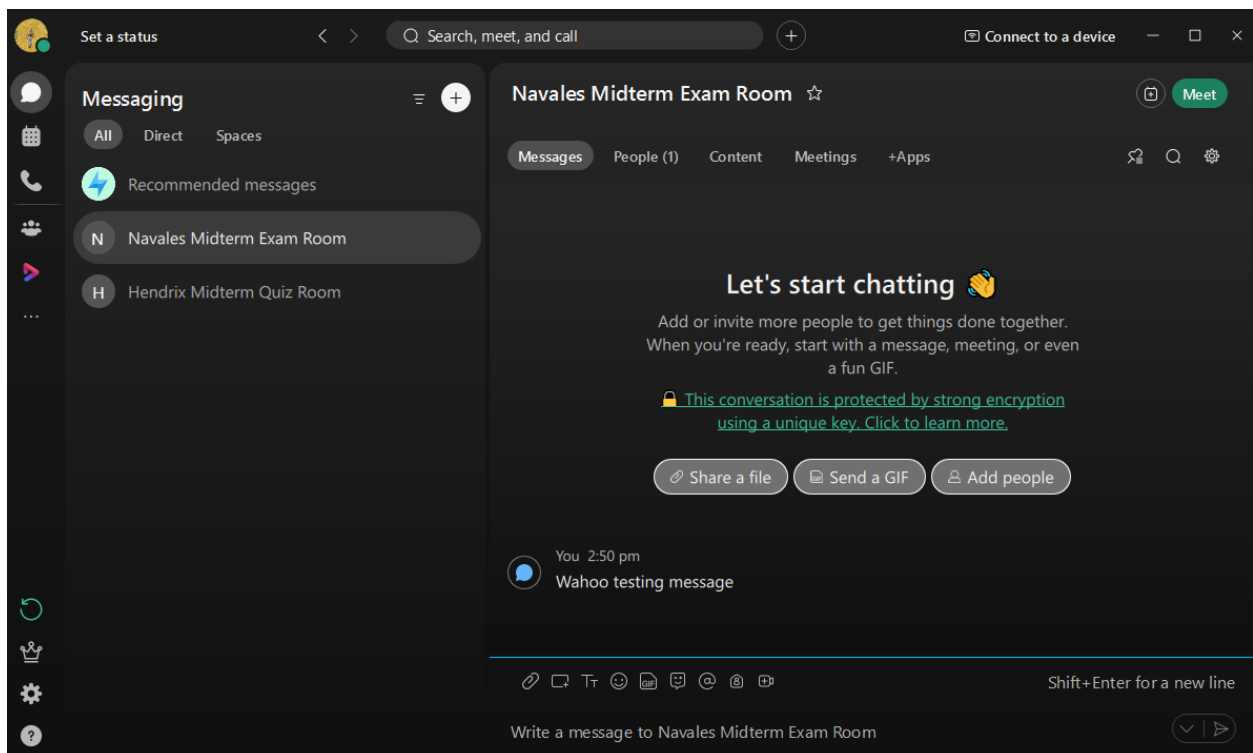
Part 1: Webex Teams API (30 Points)

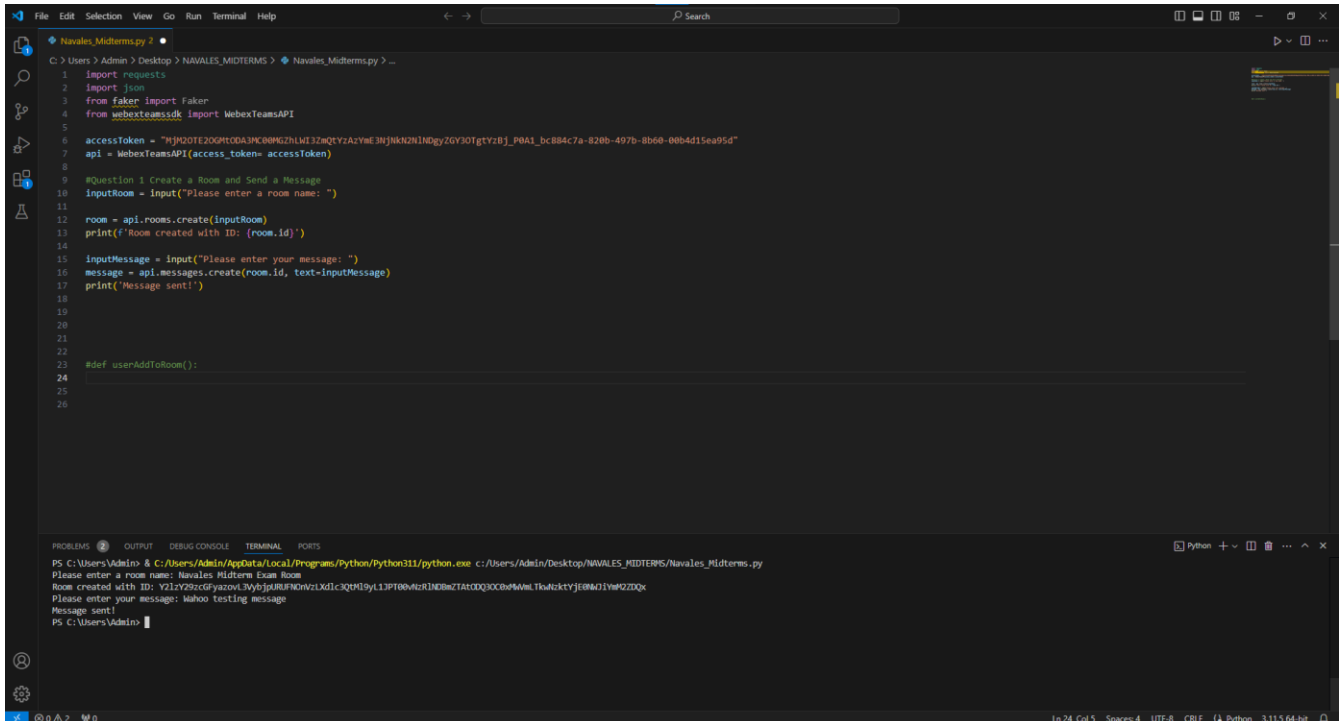
Question 1: Create a Room and Send a Message (10 pts)

Write a Python script that creates a room in Webex and sends a welcome message to the room. Make sure to use the `webex teams sdk` library. Provide the room name and message as user inputs (using `input()`).

Requirements:

- Create a Webex room with the title provided by the user.
- Send a message to the room that the user also provides.





```
File Edit Selection View Go Run Terminal Help
C:\Users\Admin\Desktop> NAVALES_MIDTERMS > Navales_Midterms.py ...
1 import requests
2 import json
3 from faker import Faker
4 from webxteamssdk import WebexTeamsAPI
5
6 accessToken = "HjMQ0TEZ0GHCOA3MC0BMGZHLMI32mQYzAZyME3HjNKNZHIHQyZ0Y30TgtYzBj_P0A1_bc884c7a-820b-497b-8b0b-0b04d15ea95d"
7 api = WebexTeamsAPI(accessToken= accessToken)
8
9 #Question 1: Create a Room and Send a Message
10 inputRoom = input("Please enter a room name: ")
11
12 room = api.rooms.create(inputRoom)
13 print(f'Room created with ID: {room.id}')
14
15 inputMessage = input("Please enter your message: ")
16 message = api.messages.create(room.id, text=inputMessage)
17 print("Message sent!")
18
19
20
21
22
23 #def userAddToRoom():
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Admin> & C:\Users\Admin\AppData\Local\Programs\Python\Python311\python.exe c:\Users\Admin\Desktop\NAVALES_MIDTERMS\Navales_Midterms.py

Please enter a room name: Navales Midterm Exam Room

Room created with ID: Y21Y292cGfyzovL3ybJpUR8H0nvzLxd1c3Q9K3yL13P10bWzR1ND0bzTAT0XQ30C0bWMLT6wKtYJj0B4G1YwZ0Xx

Please enter your message: Mahoo testing message

Message sent!

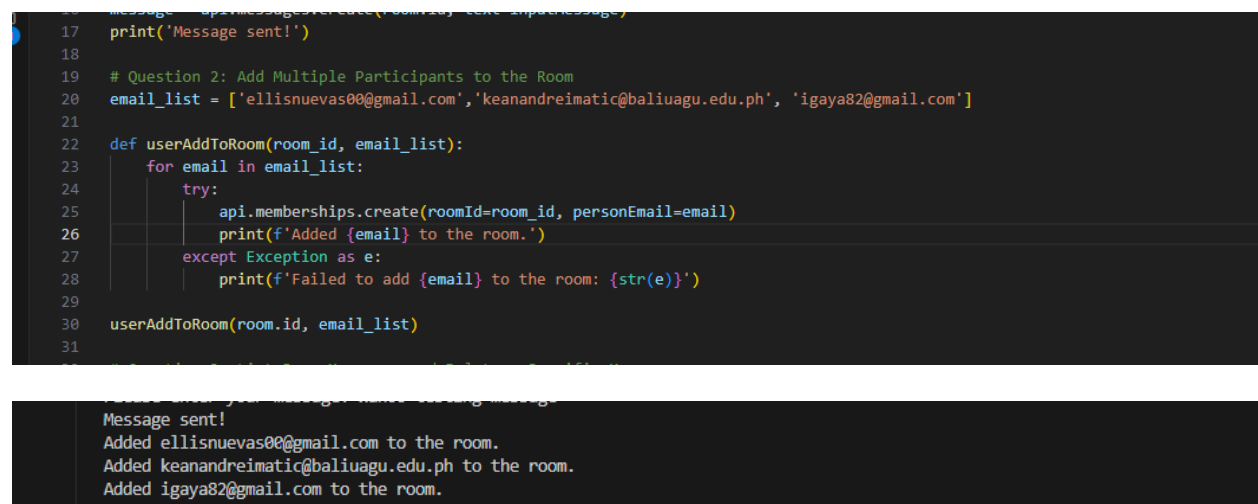
PS C:\Users\Admin>

Question 2: Add Multiple Participants to the Room (10 pts)

After creating the room, write a Python function to add multiple participants (using their email addresses) to the room. The emails should be read from a list of at least three addresses.

Requirements:

- Use a list to store three participants' email addresses.
- Loop through the list and add each participant to the room using the Webex API.



```
17 print('Message sent!')
18
19 # Question 2: Add Multiple Participants to the Room
20 email_list = ['ellisnuevas00@gmail.com', 'keanandreimatic@baliuagu.edu.ph', 'igaya82@gmail.com']
21
22 def userAddToRoom(room_id, email_list):
23     for email in email_list:
24         try:
25             api.memberships.create(roomId=room_id, personEmail=email)
26             print(f'Added {email} to the room.')
27         except Exception as e:
28             print(f'Failed to add {email} to the room: {str(e)}')
29
30 userAddToRoom(room.id, email_list)
31
```

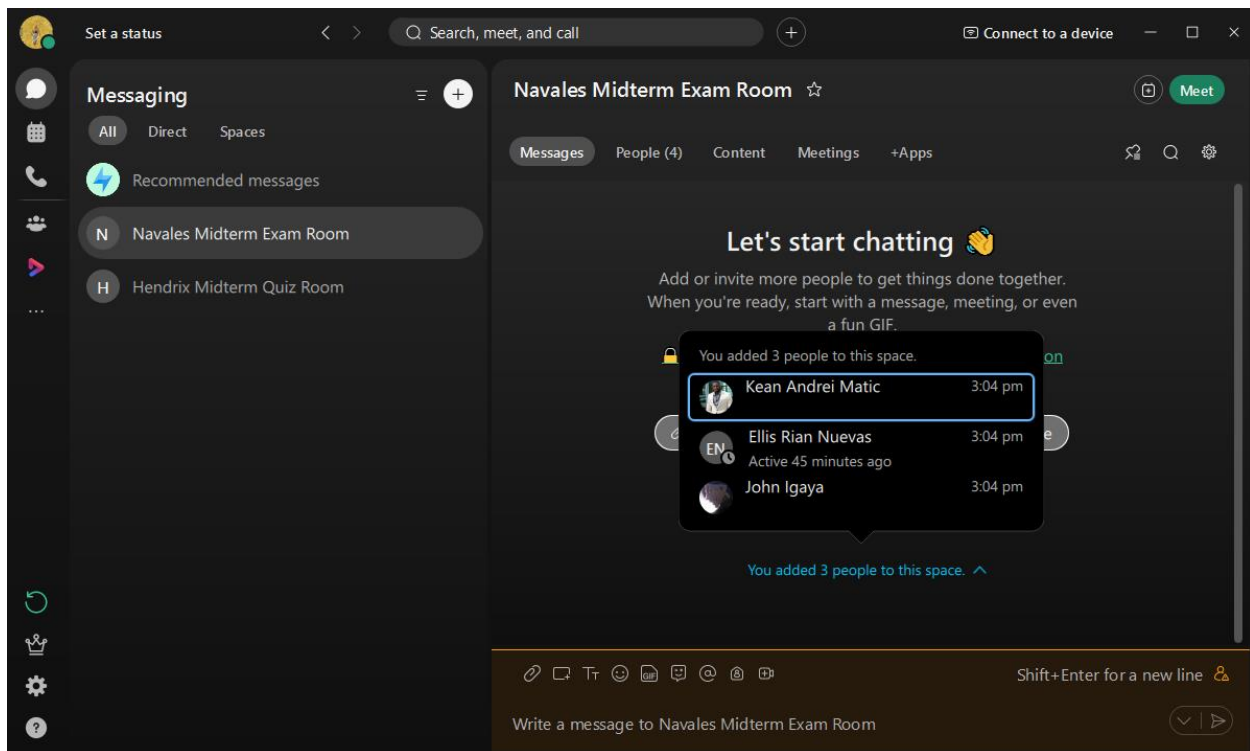
Message sent!

Added ellisnuevas00@gmail.com to the room.

Added keanandreimatic@baliuagu.edu.ph to the room.

Added igaya82@gmail.com to the room.

Listing all members in the room:



Question 3: List Room Messages and Delete a Specific Message (10 pts)

Extend your previous script to list all the messages in the room. Then, allow the user to delete a specific message by providing its message ID.

Requirements:

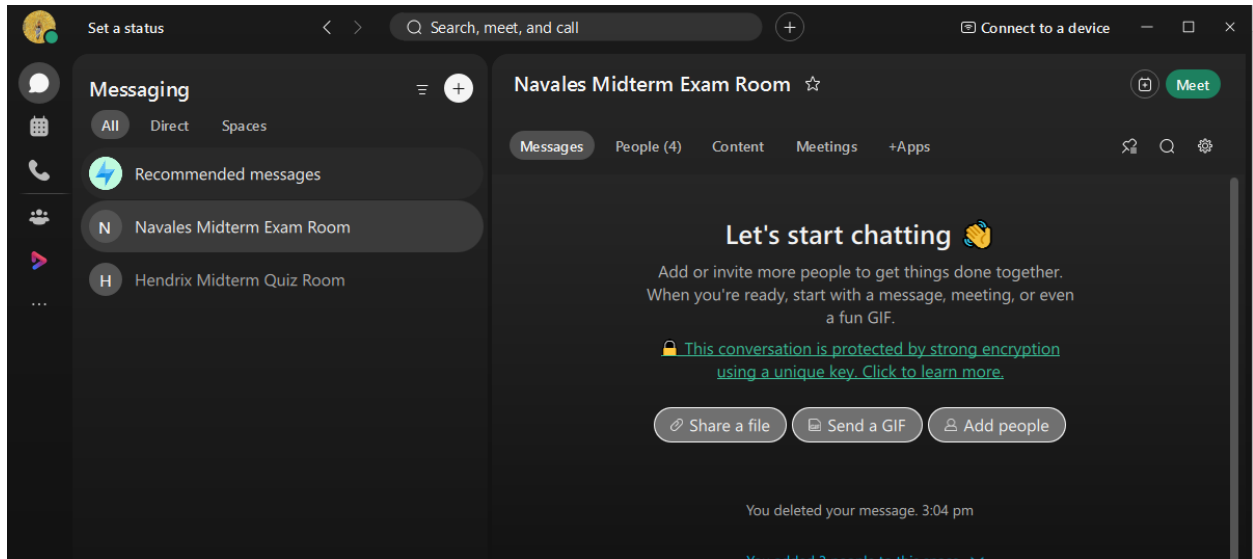
- Display all messages with their message IDs.
- Prompt the user to input a message ID to delete.
- Use the Webex API to delete the selected message.

```

31
32 # Question 3: List Room Messages and Delete a Specific Message
33 def list_room_messages(room_id):
34     messages = api.messages.list(roomId=room_id)
35     for message in messages:
36         print(f'Message ID: {message.id}, Text: {message.text}')
37
38 def deleteMessage(message_id):
39     try:
40         api.messages.delete(message_id)
41         print(f'Message with ID {message_id} deleted.')
42     except Exception as e:
43         print(f'Failed to delete message with ID {message_id}: {str(e)}')
44
45 print('Listing all messages in the room:')
46 list_room_messages(room.id)
47
48 inputMessageId = input("Please enter the message ID to delete: ")
49 deleteMessage(inputMessageId)
50

```

```
Listing all messages in the room:
Message ID: Y21zY29zcGfYazovL3VybJpURUFNbnVzLXd1c3QtM19yL01FU1NBR8UWVjY2YwQzZTAtODQ3YS8xMmMlTlhYmItYjlkOT1kM2VlNmUw, Text: Wahoo testing message
Please enter the message ID to delete: Y21zY29zcGfYazovL3VybJpURUFNbnVzLXd1c3QtM19yL01FU1NBR8UWVjY2YwQzZTAtODQ3YS8xMmMlTlhYmItYjlkOT1kM2VlNmUw
Message with ID Y21zY29zcGfYazovL3VybJpURUFNbnVzLXd1c3QtM19yL01FU1NBR8UWVjY2YwQzZTAtODQ3YS8xMmMlTlhYmItYjlkOT1kM2VlNmUw deleted.
PS C:\Users\Admin> |
```



Part 2: Faker Program (20 Points)

Question 4: Generate Fake User Profiles (10 pts)

Write a Python script that uses the `faker` library to generate 10 fake user profiles. Each profile should include:

- Full name
- Email address
- Job title
- Company

Requirements:

- Generate a list of 10 user profiles.
- Print each profile in a structured format (one line per profile).

```
55
56 fake = Faker()
57
58 userProfiles = []
59 for _ in range(10):
60     profile = {}
61     'Full name': fake.name(),
62     'Email address': fake.email(),
63     'Job title': fake.job(),
64     'Company': fake.company()
65 }
66 userProfiles.append(profile)
67
68 print(f"Full Name: {profile['Full name']}, Email Address: {profile['Email address']}, Job Title: {profile['Job title']}, Company: {profile['Company']}")
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Admin> & C:/Users/Admin/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Admin/Desktop/NAVALES_MIDTERMS/Navales_Midterms.py
Full Name: Elizabeth Carroll, Email Address: christopher80@example.com, Job Title: Hydrographic surveyor, Company: McIntyre, Strong and Hogan
Full Name: Stephanie Davis, Email Address: odavis@example.com, Job Title: Medical technical officer, Company: Miller-Hill
Full Name: Melissa Saunders, Email Address: rnorris@example.com, Job Title: Programme researcher, broadcasting/film/video, Company: Davis-Phillips
Full Name: Cassandra Hutchinson, Email Address: martinnicola@example.net, Job Title: Acupuncturist, Company: Price PLC
Full Name: Sandra Terry, Email Address: otaylor@example.com, Job Title: Public affairs consultant, Company: Rice, Nelson and Warren
Full Name: Patrick Ramirez, Email Address: eric48@example.net, Job Title: Sports administrator, Company: Lewis, Thompson and Harrell
Full Name: Paul Jackson, Email Address: brownsanne@example.net, Job Title: Building services engineer, Company: Cooper-Drake
Full Name: Brian Serrano, Email Address: tjohnson@example.com, Job Title: Theatre director, Company: Porter-Campbell
Full Name: Patricia Stanley, Email Address: haysalica@example.org, Job Title: Chartered accountant, Company: Galloway-Conway
Full Name: Jeffery Gray, Email Address: dawnbell@example.net, Job Title: Insurance broker, Company: Hancock Group
PS C:\Users\Admin>

Question 5: Generate Fake Transaction Records (10 pts)

Create a script that generates 10 fake transaction records. Each record should include:

- Transaction ID (random alphanumeric string)
- Transaction date (in the format YYYY-MM-DD)
- Amount (random float between 100.00 and 5000.00)

Requirements:

- Use the `Faker` library to generate the data.
- Print each transaction in a readable format.

```
72
73 import random
74 import string
75
76 for _ in range(10):
77     transac_ID = ''.join(random.choices(string.ascii_letters + string.digits, k=10))
78     transac_date = fake.date(pattern="%Y-%m-%d")
79     amount = round(random.uniform(100.00, 5000.00), 2)
80     print(f"Transaction ID: {transac_ID}, Transaction date: {transac_date}, Amount: {amount}")
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Admin> & C:/Users/Admin/AppData/Local/Programs/Python/Python311/python.exe c:/Users/Admin/Desktop/NAVALES_MIDTERMS/Navales_Midterms.py
Transaction ID: rCZidJeb0d, Transaction date: 2802-03-27, Amount: 717.86
Transaction ID: N39tZ9IBVX, Transaction date: 2811-11-25, Amount: 4153.07
Transaction ID: gnHr9kin7K, Transaction date: 2814-02-12, Amount: 3761.5
Transaction ID: 1jDMYxU8x1, Transaction date: 1998-01-18, Amount: 778.09
Transaction ID: vy67Wkl4K5, Transaction date: 1995-08-21, Amount: 3229.29
Transaction ID: RELLi8gMqm, Transaction date: 1972-07-31, Amount: 331.44
Transaction ID: p7404jwzFC, Transaction date: 1979-03-19, Amount: 341.83
Transaction ID: TNFXtCmCM, Transaction date: 2812-12-14, Amount: 2948.55
Transaction ID: tUnjLpLH7e, Transaction date: 1984-04-08, Amount: 2891.41
Transaction ID: pCbCQq4mDE, Transaction date: 1975-02-24, Amount: 687.96
PS C:\Users\Admin>

Follow these steps:

1. Initialize a new Git repository.
2. Add your Python scripts for the Webex API and Faker tasks.
3. Commit the changes with an appropriate commit message.
4. Push the changes to a remote GitHub repository.

Requirements:

- Provide the GitHub link to your repository with the scripts and screenshots included.