Final Exam: **Project Outline with Docker for 100 Points**

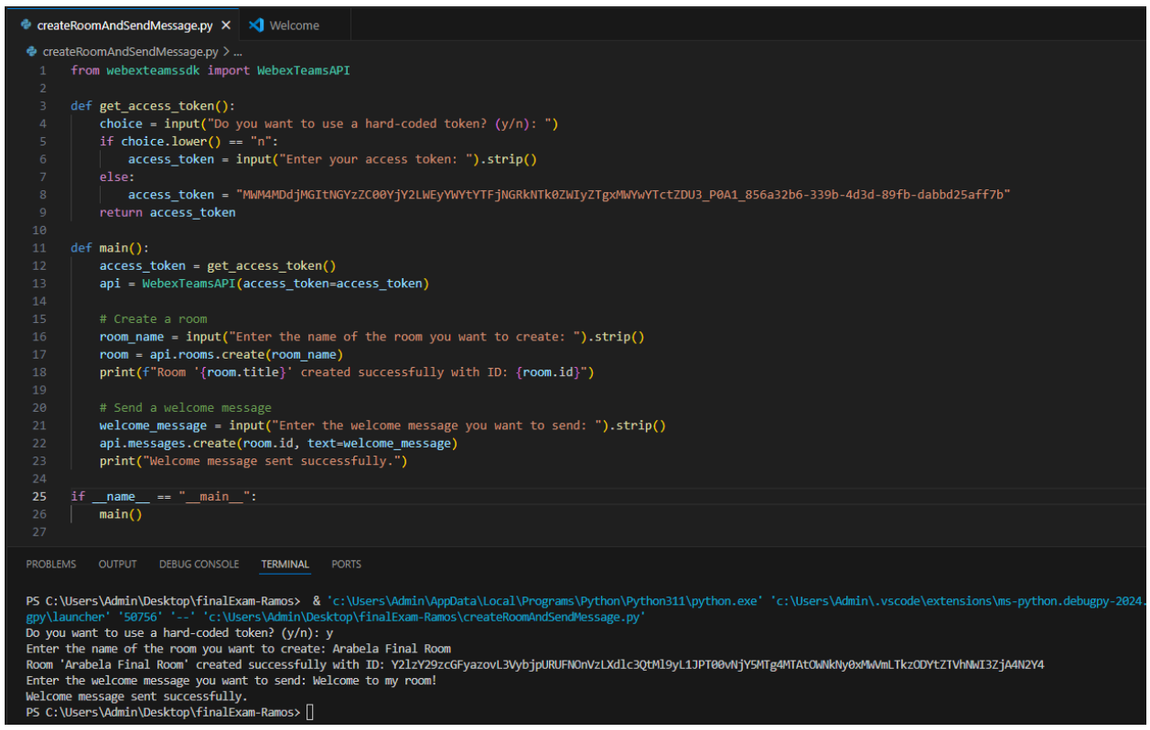
Total Structure and Points

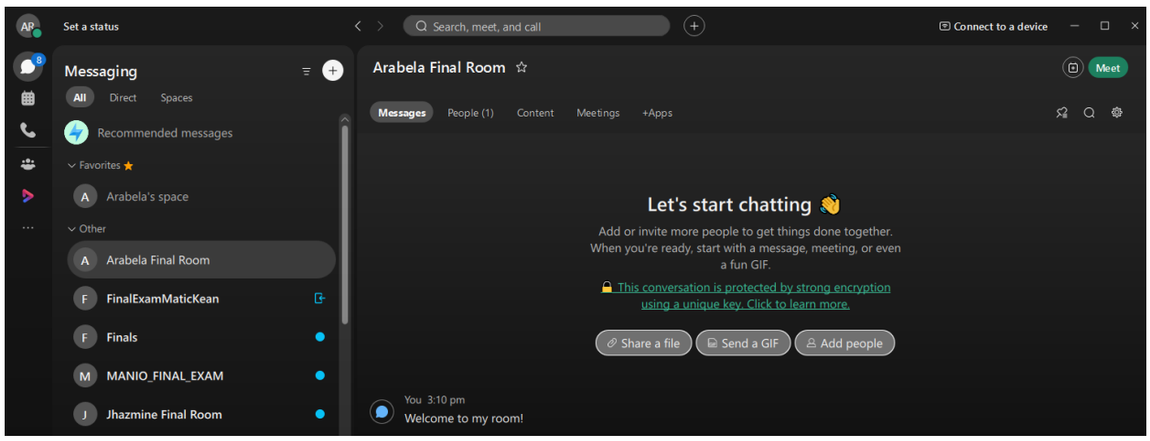
1. **Part 1: Webex Teams API Integration (30 Points)**
2. **Part 2: Faker Data Generation (20 Points)**
3. **Part 3: Docker Setup and Integration (20 Points)**
4. **Part 4: GitHub Documentation and Submission (30 Points)**

Detailed Breakdown

Part 1: Webex Teams API Integration (30 Points)

1. **Create a Room and Send a Message (10 Points)**
   - Create a script that uses webexteamssdk to create a room in Webex and send a welcome message.
   - Take user inputs for the room title and the message content.





2. **Add Multiple Participants to the Room (10 Points)**
   - Extend the script to include a function that reads participant email addresses from a predefined list and adds them to the room.

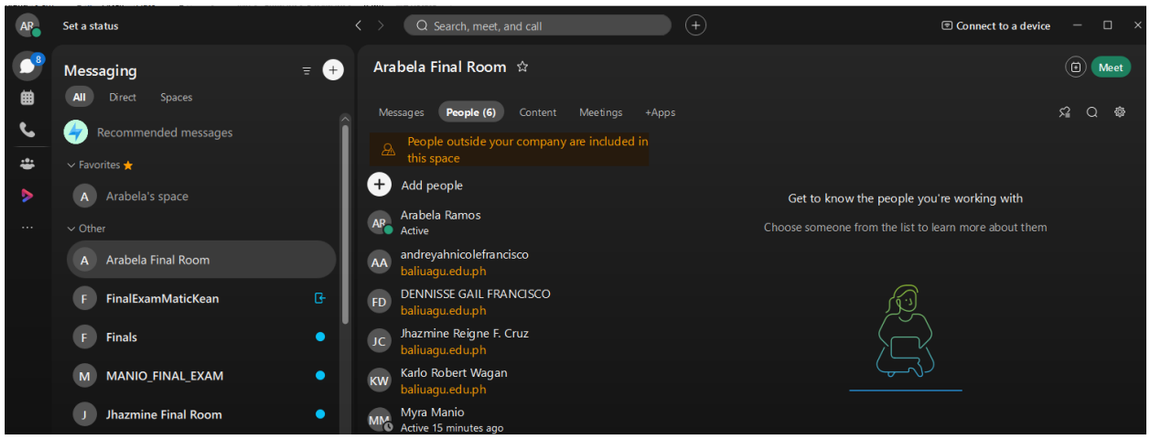
createRoomAndSendMessage.py 3 | addMultipleParticipant.py | webexTeam

webexTeamsAPI.py > main

```
67  def main():
77      # Send a welcome message
78      send_welcome_message(api, room_id)
79
80      # Add multiple participants
81      while True:
82          person_email = input("Enter participant's email (or type 'done' to finish): ").strip()
83          if person_email.lower() == "done":
84              break
85          add_participant_to_room(room_id, person_email, access_token)
86
87      # List all room messages
88      list_room_messages(api, room_id)
89
90      # Option to delete a specific message
91      delete_choice = input("\nDo you want to delete a message? (y/n): ").strip()
92      if delete_choice.lower() == 'y':
```

PROBLEMS 3 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
PS C:\Users\Admin\Desktop\finalExam-Ramos> & 'c:\Users\Admin\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debugpy-2024.
gpy\launcher' '51007' '--' 'c:\Users\Admin\Desktop\finalExam-Ramos\webexTeamsAPI.py'
Do you want to use a hard-coded token? (y/n): y
Using Access Token: ZTNmYjkzYW...
Enter the name of the room you want to create: Arabela Final Room
Room 'Arabela Final Room' created successfully with ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNjFiZTFjMjAtOWNkOS0xMWVmLTg3MmMtODE1OWIwMDgzNWNi
Enter the welcome message you want to send: Welcome to my room!
Welcome message sent successfully.
Enter participant's email (or type 'done' to finish): myraangelamanic@baliuagu.edu.ph
Participant myraangelamanic@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): karlorobertwagan@baliuagu.edu.ph
Participant karlorobertwagan@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): jhazminereignecruz@baliuagu.edu.ph
Participant jhazminereignecruz@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): andreyahnicolefrancisco@baliuagu.edu.ph
Participant andreyahnicolefrancisco@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): dennissegailfrancisco@baliuagu.edu.ph
Participant dennissegailfrancisco@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): done
```



3. **List Room Messages and Delete a Specific Message (10 Points)**
   - Create a function to list all messages in the room with their message IDs.
   - Allow the user to delete a message by providing its message ID.

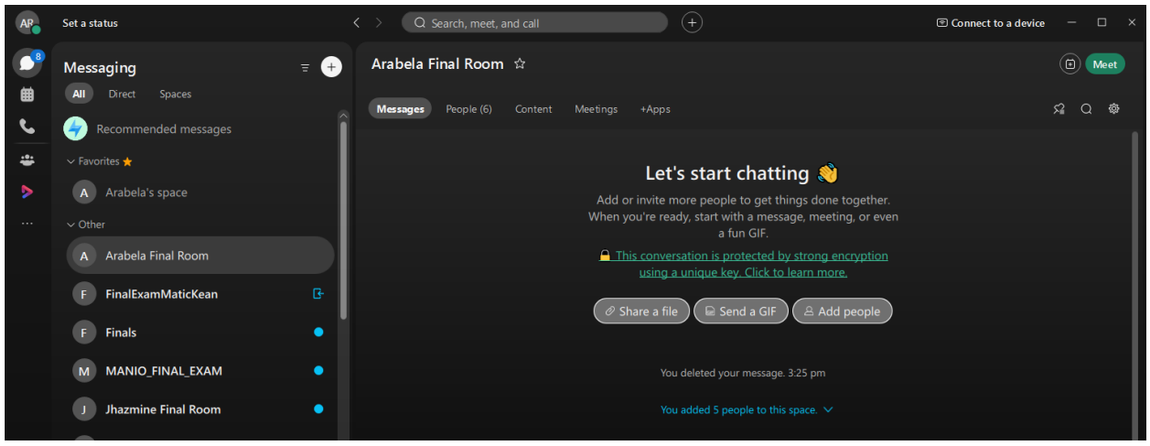
createRoomAndSendMessage.py 3 | addMultipleParticipant.py | webexTeamsAPI.py

webexTeamsAPI.py > main

```
67  def main():
85          add_participant_to_room(room_id, person_email, access_token)
86
87      # List all room messages
88      list_room_messages(api, room_id)
89
90      # Option to delete a specific message
91      delete_choice = input("\nDo you want to delete a message? (y/n): ").strip()
92      if delete_choice.lower() == 'y':
93          message_id = input("Enter the message ID to delete: ").strip()
94          delete_message(api, message_id)
95
96  if __name__ == "__main__":
97      main()
98
```

PROBLEMS 3 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
PS C:\Users\Admin\Desktop\finalExam-Ramos> & 'c:\Users\Admin\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debugpy-2024.
gpy\launcher' '51007' '--' 'c:\Users\Admin\Desktop\finalExam-Ramos\webexTeamsAPI.py'
Do you want to use a hard-coded token? (y/n): y
Using Access Token: ZTNmYjkzYW...
Enter the name of the room you want to create: Arabela Final Room
Room 'Arabela Final Room' created successfully with ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vNjFiZTFjMjAtOWNkOS0xMWVmLTg3MmMtODE1OWIwMDgzNWNi
Enter the welcome message you want to send: Welcome to my room!
Welcome message sent successfully.
Enter participant's email (or type 'done' to finish): myraangelamanic@baliuagu.edu.ph
Participant myraangelamanic@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): karlorobertwagan@baliuagu.edu.ph
Participant karlorobertwagan@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): jhazminereignecruz@baliuagu.edu.ph
Participant jhazminereignecruz@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): andreyahnicolefrancisco@baliuagu.edu.ph
Participant andreyahnicolefrancisco@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): dennissegailfrancisco@baliuagu.edu.ph
Participant dennissegailfrancisco@baliuagu.edu.ph added successfully.
Enter participant's email (or type 'done' to finish): done

List of messages in the room:
Message ID: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL01FU1NBR0UvNmIwMDkwMTAtOWNkOS0xMWVmLTkxNWEtNDc2YzdmYTg1MTU4, Text: Welcome to my room!

Do you want to delete a message? (y/n): y
Enter the message ID to delete: Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL01FU1NBR0UvNmIwMDkwMTAtOWNkOS0xMWVmLTkxNWEtNDc2YzdmYTg1MTU4
Message with ID Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL01FU1NBR0UvNmIwMDkwMTAtOWNkOS0xMWVmLTkxNWEtNDc2YzdmYTg1MTU4 has been deleted.
PS C:\Users\Admin\Desktop\finalExam-Ramos>
```



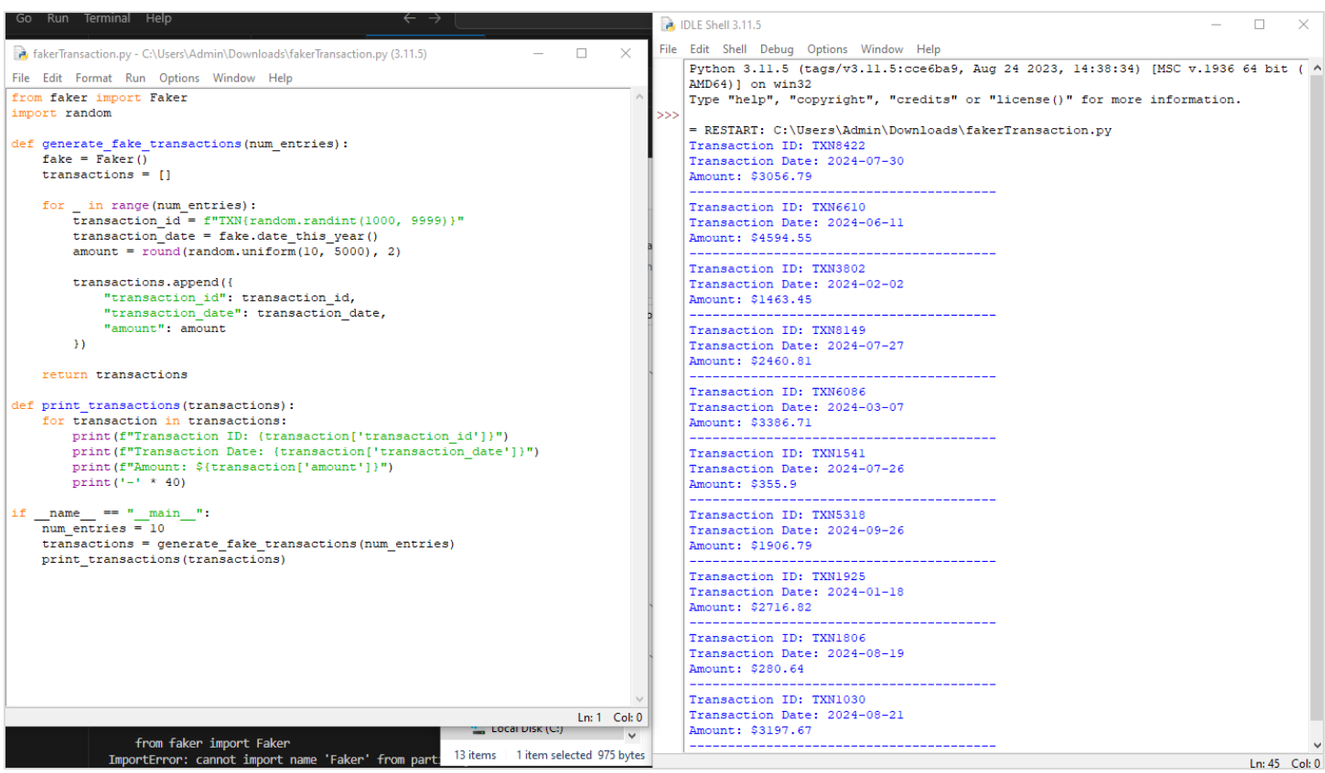
Part 2: Faker Data Generation (20 Points)

1. **Generate Fake User Profiles (10 Points)**
   - Use the faker library to generate 10 fake user profiles, including name, email, job title, and company.
   - Output each profile in a readable, structured format.

```
from faker import Faker
fake = Faker()

def gen_user_profiles():
    output = []
    for i in range(10):
        profile = {
            "id": i,
            "name": fake.name(),
            "email": fake.email(),
            "address": fake.address()
        }
        print(profile)
        output.append(profile)

def main():
    while True:
        print("\nFake Generator")
        print("[a] Generate Random User Profiles")
        print("[q] Quit")
        print()
        selection = input(" >>> ")

        if selection == 'a':
            gen_user_profiles()
        elif selection == 'q':
            print("Quit")
            break
        else:
            print("Invalid option. Please try again.")

if __name__ == "__main__":
    main()
```

```
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Admin\Downloads\fake_data_gen-1.py

Fake Generator
[a] Generate Random User Profiles
[q] Quit

 >>> a
{'id': 0, 'name': 'Aaron Odonnell', 'email': 'michaelwhite@example.com', 'address': '78768 Felicia Run Suite 647\nAlexanderchester, OR 13585'},)
{'id': 1, 'name': 'John Powell', 'email': 'icharles@example.net', 'address': '82678 Gonzalez Divide\nSouth Christina, UT 89990'},)
{'id': 2, 'name': 'Nicole Schultz', 'email': 'cknight@example.com', 'address': '2173 Torres Trace Suite 984\nMichaelmouth, NE 75588'},)
{'id': 3, 'name': 'Amber Brown', 'email': 'williamorozco@example.com', 'address': '2791 Arroyo Glens\nWest Johnstad, KS 50378'},)
{'id': 4, 'name': 'Julie Herring', 'email': 'acarter@example.net', 'address': '688 Nguyen Square Suite 147\nNew Karen, MI 97146'},)
{'id': 5, 'name': 'Vanessa Cook', 'email': 'littledavid@example.org', 'address': '221 Jennifer Island\nWalkertown, VT 48119'},)
{'id': 6, 'name': 'Thomas Davis', 'email': 'bthomas@example.org', 'address': '120 Webb Mountain\nMccarthyhaven, GU 36822'},)
{'id': 7, 'name': 'Ashley Gutierrez', 'email': 'zruiz@example.net', 'address': '9764 Madeline Motorway Apt. 655\nTeresafort, HI 76394'},)
{'id': 8, 'name': 'Howard Meyer MD', 'email': 'batesaimee@example.com', 'address': '95425 Jones Curve Apt. 125\nEast Brandonville, MP 42734'},)
{'id': 9, 'name': 'Timothy Ramsey', 'email': 'ujenkins@example.org', 'address': '1390 Reyes Unions\nValdezchester, SD 08375'},)

Fake Generator
[a] Generate Random User Profiles
[q] Quit

 >>>
```

2. **Generate Fake Transaction Records (10 Points)**
   - Generate 10 fake transaction records, each with a unique ID, transaction date, and amount.
   - Display each transaction in a readable format.

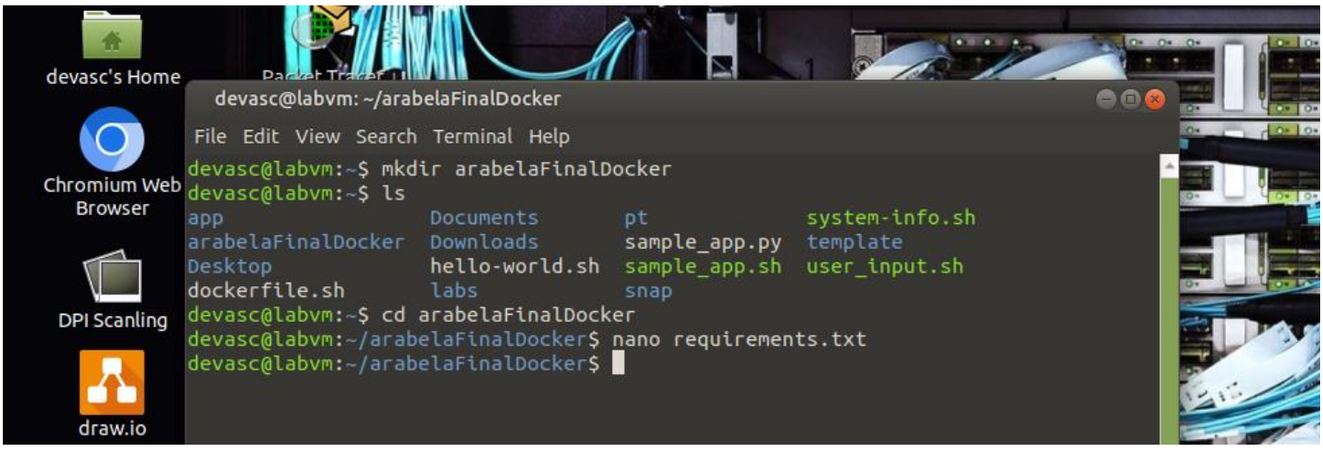


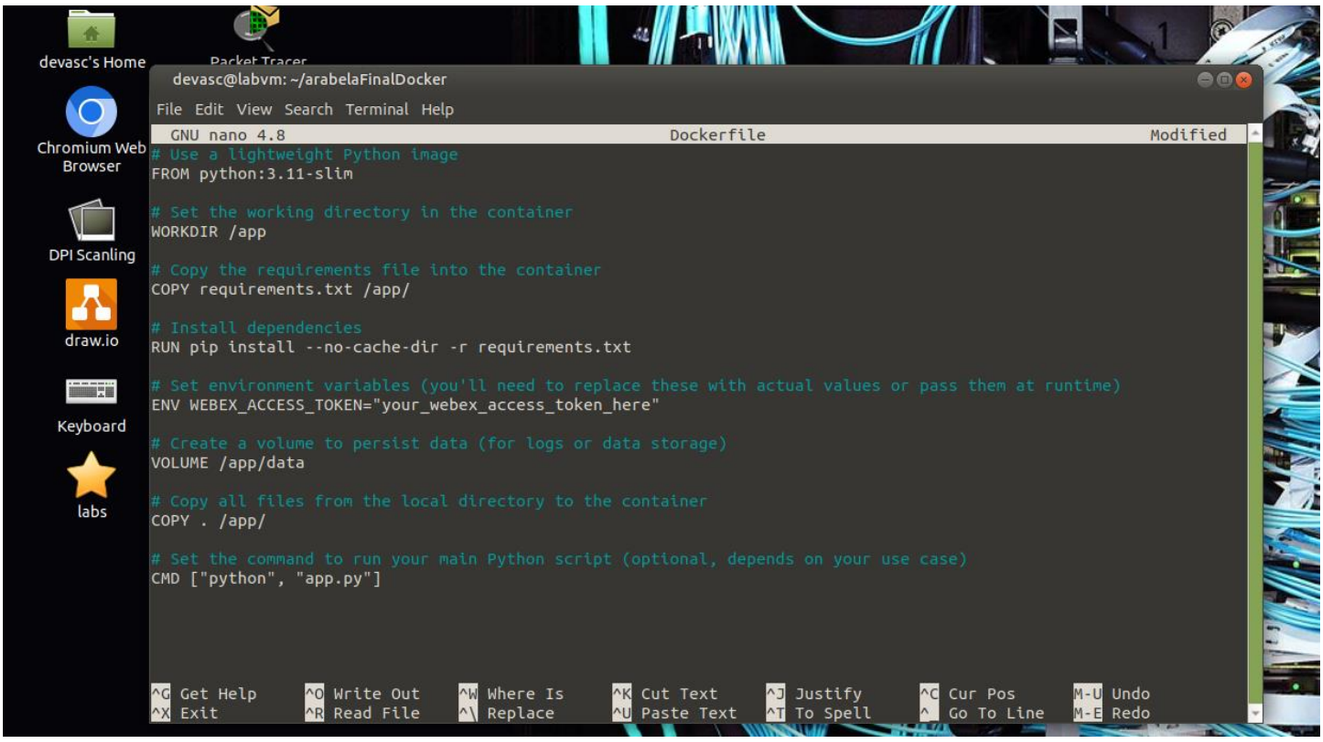
```
from faker import Faker
import random

def generate_fake_transactions(num_entries):
    fake = Faker()
    transactions = []

    for _ in range(num_entries):
        transaction_id = f"TXN{random.randint(1000, 9999)}"
        transaction_date = fake.date_this_year()
        amount = round(random.uniform(10, 5000), 2)

        transactions.append({
            "transaction_id": transaction_id,
            "transaction_date": transaction_date,
            "amount": amount
        })

    return transactions

def print_transactions(transactions):
    for transaction in transactions:
        print(f"Transaction ID: {transaction['transaction_id']}")
        print(f"Transaction Date: {transaction['transaction_date']}")
        print(f"Amount: ${transaction['amount']}")
        print('-' * 40)

if __name__ == "__main__":
    num_entries = 10
    transactions = generate_fake_transactions(num_entries)
    print_transactions(transactions)
```

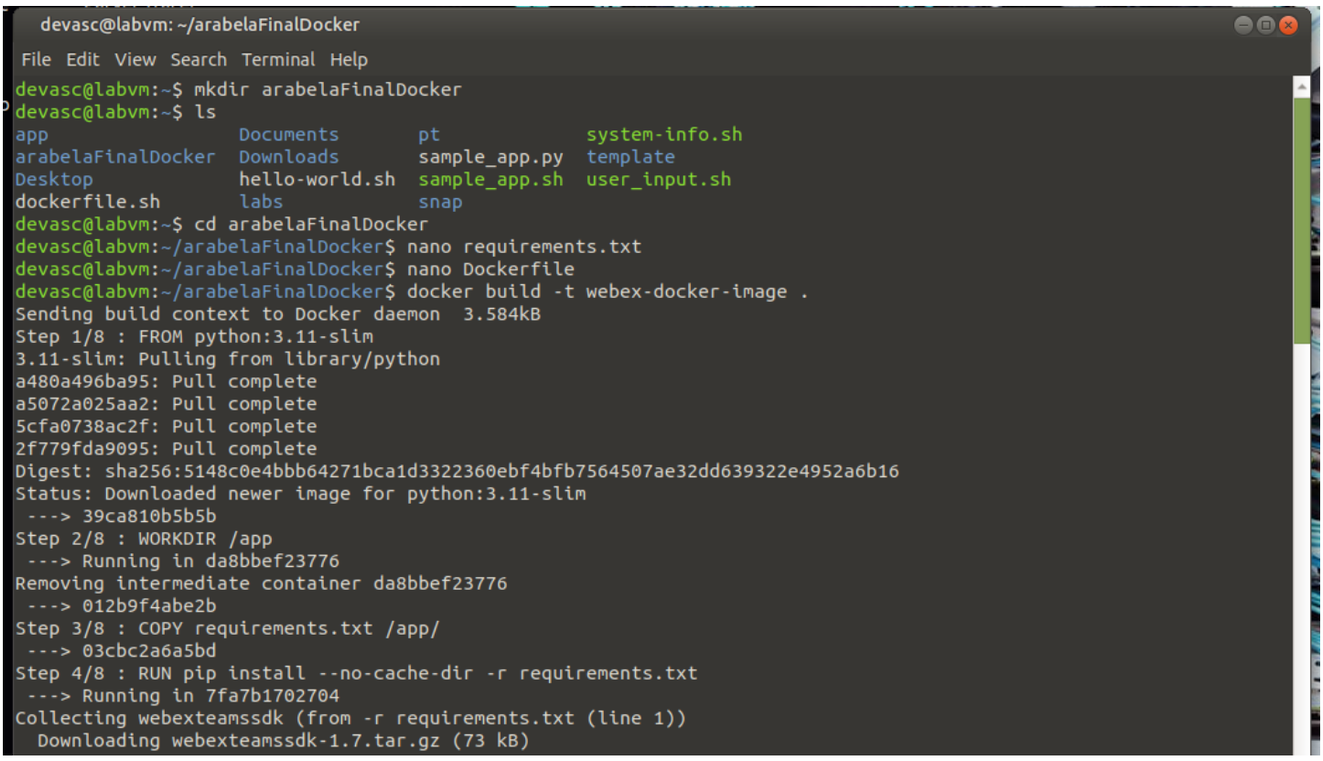
```
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Admin\Downloads\fakerTransaction.py
Transaction ID: TXN8422
Transaction Date: 2024-07-30
Amount: $3056.79
----------------------------------------
Transaction ID: TXN6610
Transaction Date: 2024-06-11
Amount: $4594.55
----------------------------------------
Transaction ID: TXN3802
Transaction Date: 2024-02-02
Amount: $1463.45
----------------------------------------
Transaction ID: TXN8149
Transaction Date: 2024-07-27
Amount: $2460.81
----------------------------------------
Transaction ID: TXN6086
Transaction Date: 2024-03-07
Amount: $3386.71
----------------------------------------
Transaction ID: TXN1541
Transaction Date: 2024-07-26
Amount: $355.9
----------------------------------------
Transaction ID: TXN5318
Transaction Date: 2024-09-26
Amount: $1906.79
----------------------------------------
Transaction ID: TXN1925
Transaction Date: 2024-01-18
Amount: $2716.82
----------------------------------------
Transaction ID: TXN1806
Transaction Date: 2024-08-19
Amount: $280.64
----------------------------------------
Transaction ID: TXN1030
Transaction Date: 2024-08-21
Amount: $3197.67
----------------------------------------
```

Part 3: Docker Setup and Integration (20 Points)

1. **Create a Dockerfile (10 Points)**
    - **Base Image**: Use a lightweight Python image.
    - **Working Directory**: Set a working directory in the container.
    - **Install Dependencies**: Include webexteamssdk and faker in a requirements.txt file, and configure the Dockerfile to install them.
    - **Environment Variables**: Use environment variables to pass sensitive information, like the Webex access token.
    - **Volumes and Data Storage**: Configure volumes to store logs or data persistently outside the container.



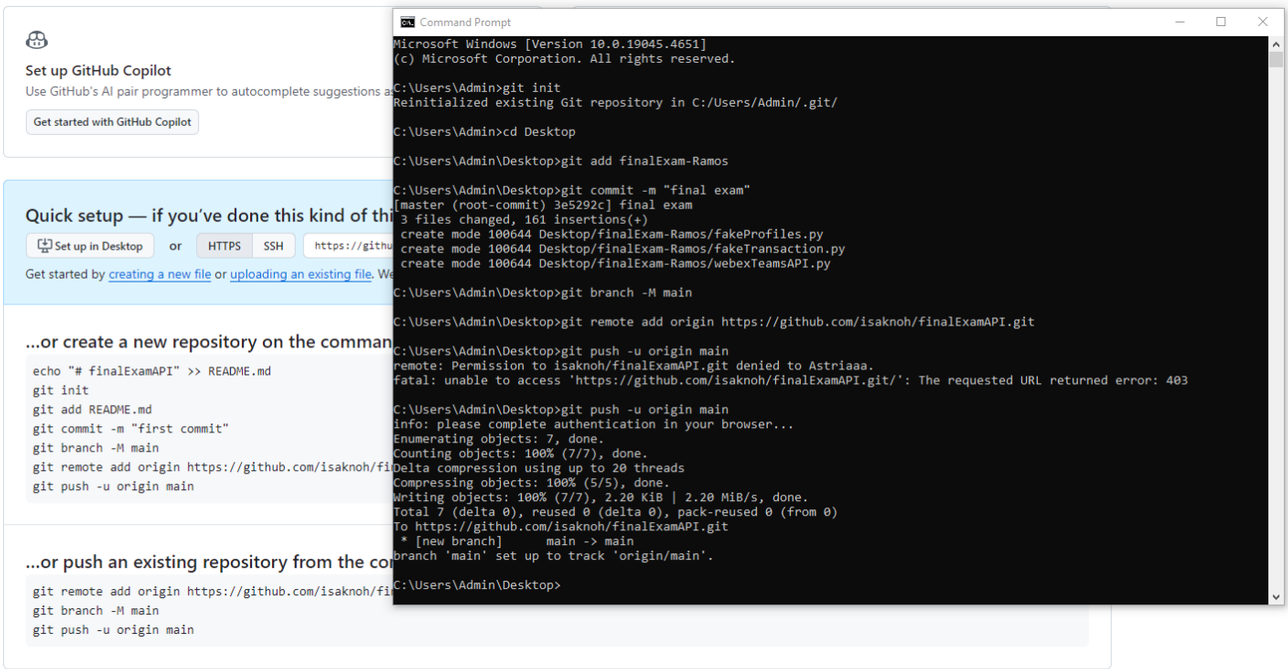




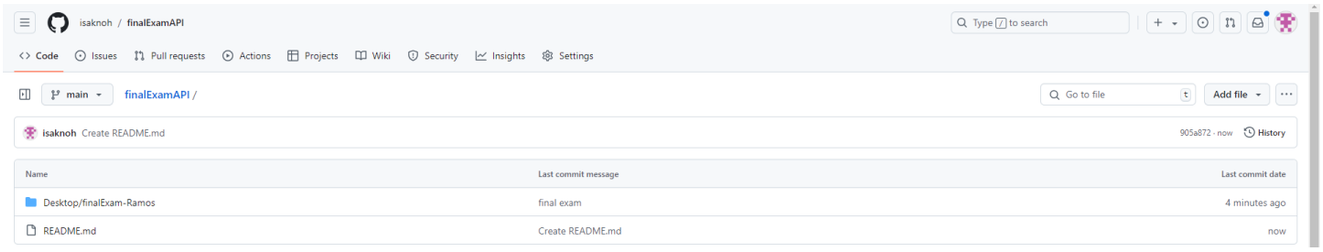
2. **Docker Commands for Running Tasks (10 Points)**
   - Write Docker commands for running each script independently in the container, specifying environment variables and any necessary inputs at runtime.
   - Include clear instructions for building and running the Docker container with both Webex and Faker tasks.

Part 4: GitHub Documentation and Submission (30 Points)

1. **Git Repository and Initial Commit (10 Points)**
   - Initialize a new Git repository, add the necessary project files, and commit the initial setup.



2. **README and Documentation (10 Points)**
   - Write a README.md with clear instructions on how to set up, run, and test the application using Docker.



- Include steps to install Docker (if needed), build the Docker image, and run the container for each task.

3. **Screenshots and Project Submission (10 Points)**
   - Run the scripts and take screenshots to demonstrate the project's functionality, including Docker builds, Webex room creation, message sending, data generation, etc.
   - Include screenshots in a folder in the repository for easy reference.
4. **GitHub Repository Link**
   - Provide a link to the GitHub repository with all scripts, Docker setup, and documentation included for review.

LINK: https://github.com/isaknoh/finalExamAPI