МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗВІТ

для лабораторної роботи №7 з

дисципліни

«Спеціалізовані мови програмування»

Виконав:

студент гр. ІТ-32

Паньків Б. В.

Прийняв:

доц. каф. ІСМ

Щербак С.С.

Львів-2023

Мета: Створення консольного об’єктно - орієнтованого додатка з використанням API

Хід виконання:

План роботи

Завдання 1: Вибір провайдера API

Виберіть надійний API, який надає через HTTP необхідні дані для віддаленого зберігання, вивантаження або реалізуйте свій. Для прикладу це може бути jsonplaceholder.org

Завдання 2: Інтеграція API

Виберіть бібліотеку для роботи з API та обробки HTTP запитів (для прикладу це може бути бібліотека Requests). Інтегруйте обраний API в ваш консольний додаток на Python. Ознайомтеся з документацією API та налаштуйте необхідний API-ключ чи облікові дані.

Завдання 3: Введення користувача

Розробіть користувальницький інтерфейс, який дозволяє користувачам візуалізувати всі доступні дані в табличному вигляді та у вигляді списку. Реалізуйте механізм для збору та перевірки введеного даних користувачем.

Завдання 4: Розбір введення користувача

Створіть розбірник для видобування та інтерпретації виразів користувача на основі регулярних виразів, наприклад, для візуалізації дат, телефонів, тощо. Переконайтеся, що розбірник обробляє різні формати введення та надає зворотний зв'язок про помилки.

Завдання 5: Відображення результатів

Реалізуйте логіку для візуалізації даних через API в консолі. Обробляйте відповіді API для отримання даних у вигляді таблиць, списків. Заголовки таблиць, списків мають виділяться кольором та шрифтом, які задається користувачем

Завдання 6: Збереження даних

Реалізуйте можливості збереження даних у чіткому та читабельному форматі JSON, CSV та TXT

Завдання 7: Обробка помилок

Розробіть надійний механізм обробки помилок для керування помилками API, некоректним введенням користувача та іншими можливими проблемами. Надавайте інформативні повідомлення про помилки.

Завдання 8: Ведення історії обчислень

Включіть функцію, яка реєструє запити користувача, включаючи введені запити та відповідні результати. Дозвольте користувачам переглядати та рецензувати історію своїх запитів.

Завдання 9: Юніт-тести

Напишіть юніт-тести для перевірки функціональності вашого додатку. Тестуйте різні операції, граничні випадки та сценарії помилок.

Код:

Config.py

```
X_RapidAPI_Key="72fd5307f1msh400099b75a5b4b0p14abf2jsnb900f93a0c64"
X_RapidAPI_Host="instagram130.p.rapidapi.com"

get_personal_profile="https://instagram130.p.rapidapi.com/account-info"
```

Service.py

```
import json
import pyfiglet
from colorama import Fore
import requests
from prettytable import PrettyTable
import regex
import config as config

fonts = {index: font for index, font in enumerate(sorted(pyfiglet.FigletFont.getFonts()))}

class UserProfileService:
    def get_personal_profile(self, username: str):
        if not username or not isinstance(username, str) or not regex.match(
            "^[\w](?!.*?\.{2})[\w.]{1,28}[\w]$", username
        ):
            raise ValueError("Invalid username format!")

        query_params = {"username": username}
        headers = {
            "X-RapidAPI-Key": config.X_RapidAPI_Key,
```

```
        "X-RapidAPI-Host": config.X_RapidAPI_Host
    }

    response = requests.get(config.get_personal_profile, headers=headers, params=query_params)

    if response.status_code != 200:
        error_message = response.json().get('message', 'Unknown error occurred!')
        raise ValueError(f"Error occurred! {error_message}")
    else:
        return response.json()



class ProfileTableDisplayer:

    def display_profile(self, json_data: str):
        data = json.loads(json_data)
        table = PrettyTable()
        table.field_names = ["Attribute", "Value"]

        allowed_keys = {
            "id", "biography", "full_name", "is_business_account", "category_name", "is_private",
"username"
        }

        for key, value in data.items():
            if key in allowed_keys:
                table.add_row([f"{Fore.YELLOW + key + Fore.RESET}", value])

        return table.get_string()
```

tests.py

```
import unittest
```

```
from unittest.mock import patch
from service import UserProfileService

class TestUserProfileService(unittest.TestCase):

    @patch('service.requests.get')
    def test_get_personal_profile_success(self, mock_get):
        username = "apple"
        expected_response = {"id": "5821462185"}
        mock_get.return_value.status_code = 200
        mock_get.return_value.json.return_value = expected_response

        user_service = UserProfileService()
        result = user_service.get_personal_profile(username)

        self.assertEqual(result, expected_response)

    @patch('service.requests.get')
    def test_get_personal_profile_error_response(self, mock_get):
        username = "panffffiv"
        error_message = "User not found"
        mock_get.return_value.status_code = 404
        mock_get.return_value.json.return_value = {"message": error_message}

        user_service = UserProfileService()

        with self.assertRaises(ValueError) as context:
            user_service.get_personal_profile(username)

        self.assertIn(error_message, str(context.exception))
```

```
if __name__ == '__main__':
    unittest.main()
```

Runner.py

```
from service import ProfileTableDisplayer, UserProfileService
from utility import FileProcessor
import json


def main():
    json_file_path = "./files/result.json"
    history = []
    successful_result = False
    json_data = {}
    user_profile_info = None
    user_service = UserProfileService()
    display_on_table = ProfileTableDisplayer()

    while True:
        print("Choose an option:")
        print("1. Display data of a personal profile")
        print("2. Show history")
        print("3. Save data into a file")
        print("0. Exit")

        option = input("Your choice: ")

        match option:
            case "1":
                username = input("Enter username: ")
                try:
                    json_data = user_service.get_personal_profile(username)
```

```
            print("Choose an option:")
            print("1. Display data in a table")
            print("2. Display data in JSON format")

            inner_option = input("Your choice: ")

            match inner_option:
                case "1":
                    user_profile_info = display_on_table.display_profile(json.dumps(json_data))
                case "2":
                    user_profile_info = json.dumps(json_data, indent=4)
                case _:
                    print("Invalid option. Enter again!")

            print(user_profile_info)
            history.append(
                f"Data of a personal profile where username is {username}:\n{user_profile_info}")
            successful_result = True
        except ValueError as e:
            print(e)
            successful_result = False

    case "2":
        if not history:
            print("No history!")
        else:
            for counter, item in enumerate(history, 1):
                print(f"{counter}: {item}")

    case "3":
        if history and successful_result:
```

```
            print("Choose an option in order to save into a file:")
            print("1. Save into a txt file")
            print("2. Save into a JSON file")
            print("3. Save into a CSV file")

            inner_option = input("Your choice: ")

            match inner_option:
                case "1":
                    FileProcessor.write_into_file("./files/result.txt", user_profile_info)
                case "2":
                    FileProcessor.write_into_json(json_file_path, json_data)
                case "3":
                    FileProcessor.write_into_csv("./files/result.csv", json_data)
        else:
            print("No data to save!")

    case "0":
        exit(0)

    case _:
        print("Invalid option. Enter again!")


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

На рис. 1 зображено знімок екрану із середовища розробки.

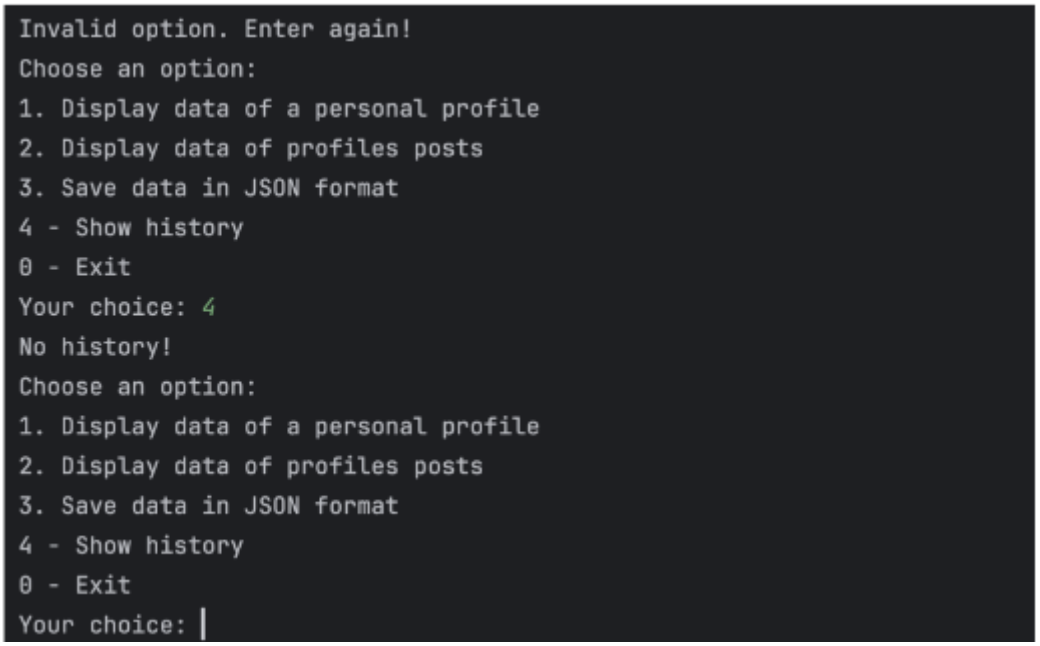

Рис. 1 Виконання програми

Посилання на GitHub-репозиторій із кодом: https://github.com/BOHDAN1329/SMP

Висновки: Я створив проєкт, який надав мені цінний досвід роботи з API, дизайну користувацького інтерфейсу,валідації введення, обрробки помилок та ткстування