МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Лабораторна робота №7

з дисципліни

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

на тему

РОБОТА З API ТА ВЕБ-СЕРВІСОМ

Виконала:

ст. гр. ІТ-21сп

Кобленц Д.А.

Прийняв:

Щербак С.С.

Львів-2023

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

**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 програми:**  
import requests

from colorama import Fore

from colorama import Style

class APIApp:

def \_\_init\_\_(self, api\_client, reader\_writer, validator, history, error\_handler, results\_display, input\_parser, data\_saver):

self.api\_client = api\_client

self.reader\_writer = reader\_writer

self.validator = validator

self.history = history

self.error\_handler = error\_handler

self.results\_display = results\_display

self.input\_parser = input\_parser

self.data\_saver = data\_saver

def get\_data\_from\_api(self):

return self.api\_client.get\_data()

def get\_user\_input(self):

user\_input = self.reader\_writer.read\_input(f"Enter email: ")

return user\_input

def run(self, data\_type, user\_input):

try:

# Task 1: Selecting the API provider

data\_from\_api = self.api\_client.get\_data()

# Checking the validity of the user's input

if self.validator.validate\_email(user\_input):

print("Email is valid.")

response = requests.get("https://jsonplaceholder.typicode.com/users?email=" + user\_input)

self.history.add\_to\_history(user\_input, response.json())

# Task 5: Displaying the results

self.results\_display.display\_data(response.json(), Fore.BLUE, Style.BRIGHT)

else:

print("Invalid email format. Please enter the email in the correct format.")

self.data\_saver.save\_data(data\_from\_api, 'json')

except Exception as e:

print(f"Error: {e}")

class APIClient:  
 """  
 The APIClient class interacts with an API to get data.  
 """  
  
 def \_\_init\_\_(self, api\_url):  
 """  
 Initializes an APIClient object with an API URL.  
 """  
 self.api\_url = api\_url  
  
 def get\_data(self):  
 """  
 Gets data from the API.  
 """  
 response = requests.get(self.api\_url)  
 return response.json()

class MenuLab7:  
 """  
 A class to manage the menu for Lab7.  
 """  
  
 def \_\_init\_\_(self, api\_app, validator, reader\_writer):  
 """  
 Initializes the MenuLab7 with an API app, a validator, and a reader/writer.  
  
 Args:  
 api\_app: The API app to use for data retrieval.  
 validator: The validator to use for input validation.  
 reader\_writer: The reader/writer to use for file operations.  
 """  
 self.api\_app = api\_app  
 self.validator = validator  
 self.reader\_writer = reader\_writer  
 log\_file\_path = os.path.join('..', 'Data', 'Lab9', 'app.log')  
 logging.basicConfig(filename=log\_file\_path, filemode='w',  
 format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')  
 logging.getLogger().setLevel(logging.INFO)  
  
 def display\_options(self):  
 """  
 Displays the menu options to the user.  
 """  
 current\_file\_directory = os.path.dirname(os.path.abspath(\_\_file\_\_))  
 relative\_config\_path = os.path.join('..', '..', 'Config', 'init.json')  
 config\_path = os.path.abspath(os.path.join(current\_file\_directory, relative\_config\_path))  
  
 with open(config\_path, 'r') as file:  
 config = json.load(file)  
  
 menu\_config = config['menu7']  
 menu\_builder = MenuBuilder(menu\_config)  
  
 menu\_builder.display()  
  
 def run(self):  
 """  
 Runs the menu, allowing the user to choose options and interact with the API app.  
 """  
 while True:  
 self.display\_options()  
 choice = self.validator.validate\_input("Choose an option (1/2/3/4/5): ", ['1', '2', '3', '4', '5'])  
 logging.info(f'User chose option: {choice}')  
 if choice == '1':  
 valid\_data = False  
 data = None  
 while not valid\_data:  
 data\_type, user\_input = self.api\_app.get\_user\_input()  
 if data\_type == 'phone':  
 if self.validator.validate\_phone(user\_input):  
 valid\_data = True  
 elif data\_type == 'email':  
 if self.validator.validate\_email(user\_input):  
 valid\_data = True  
 elif data\_type == 'credit card':  
 if self.validator.validate\_credit\_card(user\_input):  
 valid\_data = True  
 elif data\_type == 'date':  
 if self.validator.validate\_date(user\_input):  
 valid\_data = True  
 if valid\_data:  
 self.api\_app.run(data\_type, user\_input)  
 elif choice == '2':  
 self.api\_app.history.view\_history()  
 elif choice == '3':  
 display\_choice = self.validator.validate\_input("Choose a data display format (list/table): ",  
 ['list', 'table'])  
 data\_from\_api = self.api\_app.get\_data\_from\_api()  
 self.api\_app.history.add\_to\_history('Display data', data\_from\_api)  
 if display\_choice == 'list':  
 print(data\_from\_api)  
 else:  
 self.api\_app.results\_display.display\_data(data\_from\_api, Fore.BLUE, Style.BRIGHT)  
 elif choice == '4':  
 save\_choice = self.validator.validate\_input("Do you want to save the data? (yes/no): ", ['yes', 'no'])  
 if save\_choice == 'yes':  
 format\_choice = self.validator.validate\_input(  
 "Choose a format to save the data (json/csv/txt): ", ['json', 'csv', 'txt'])  
 self.api\_app.data\_saver.save\_data(self.api\_app.get\_data\_from\_api(), format\_choice)  
 print(f"Data saved in {format\_choice} format.")  
 elif choice == '5':  
 print("Goodbye!")  
 logging.info('User exited.')  
 break  
 else:  
 print("Invalid choice. Please try again.")

На рисунку 1 зображено результат програми.

**Рис. 1** Результат виконання програми(запит)

**Висновок:** я створив проект, який надає цінний досвід роботи з API, дизайну користувацького інтерфейсу, валідації введення, обробки помилок та тестування.