Міністерство освіти і науки України

Національний університет «Львівська політехніка»

******

Звіт про виконання лабораторної роботи №7  
з дисципліни  
“Спеціалізовані мови програмування”  
на тему  
“Робота з API та веб-сервісами ”

Виконала:  
студентка групи РІ-31,  
Пятницька Вікторія

Прийняв:

Щербак С.С.

**Мета.** Створення консольного об’єктно - орієнтованого додатка з використанням API та патернів проектування.

**Хід роботи**

Завдання 1: Вибір провайдера API та патернів проектування

Виберіть надійний API, який надає через HTTP необхідні дані для віддаленого зберігання, вивантаження або реалізуйте свій. Для прикладу це може бути jsonplaceholder.org. Крім того, оберіть 2-3 патерна проектування для реалізаціі імплементаціі цієї лабораторноі роботи. Для прикладу, це може бути патерн Unit of Work та Repository

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:**

**./api/api\_config.py:**

class APIConfig:

\_instance = None

def \_\_new\_\_(cls):

if cls.\_instance is None:

cls.\_instance = super().\_\_new\_\_(cls)

cls.\_instance.base\_url = "https://jsonplaceholder.typicode.com"

return cls.\_instance

**./api/app.py:**

from user\_interface import UserInterface

from input\_parser import InputParser

from data\_repository import DataRepository

from colorama import Fore

class App:

def \_\_init\_\_(self):

self.ui = UserInterface()

self.parser = InputParser()

self.repo = DataRepository("users")

self.fetched\_data = None

def run(self):

while True:

print("\n1. Fetch data\n2. Display data\n3. Save data\n4. Show history\n5. Parse input\n6. Exit")

choice = input("Select an option: ").strip()

if choice == "1":

try:

self.fetched\_data = self.repo.get\_data()

self.ui.add\_to\_history("Fetch data", self.fetched\_data)

print("Data fetched successfully.")

except Exception as e:

print(Fore.RED + f"Error: {e}")

elif choice == "2":

if not self.fetched\_data:

print(Fore.RED + "No data to display. Please fetch data first.")

continue

while True:

display\_type = input("Enter display type (table/list): ").strip().lower()

if display\_type in ["table", "list"]:

break

print(Fore.RED + "Invalid input. Please enter 'table' or 'list'.")

while True:

header\_color = input("Enter header color (red, green, blue, yellow, white, cyan, magenta): ").strip().upper()

color\_dict = {

"RED": Fore.RED,

"GREEN": Fore.GREEN,

"BLUE": Fore.BLUE,

"YELLOW": Fore.YELLOW,

"WHITE": Fore.WHITE,

"CYAN": Fore.CYAN,

"MAGENTA": Fore.MAGENTA

}

if header\_color in color\_dict:

header\_color = color\_dict[header\_color]

break

print(Fore.RED + "Invalid input. Please enter a valid color name.")

try:

self.ui.display\_data(self.fetched\_data, display\_type, header\_color)

except Exception as e:

print(Fore.RED + f"Error: {e}")

elif choice == "3":

if not self.fetched\_data:

print(Fore.RED + "No data to save. Please fetch data first.")

continue

while True:

format\_type = input("Enter file format (json/csv/txt): ").strip().lower()

if format\_type in ["json", "csv", "txt"]:

break

print(Fore.RED + "Invalid input. Please enter 'json', 'csv', or 'txt'.")

try:

self.ui.save\_data(self.fetched\_data, format\_type)

except Exception as e:

print(Fore.RED + f"Error: {e}")

elif choice == "4":

self.ui.show\_history()

elif choice == "5":

while True:

user\_input = input("Enter text to parse: ").strip()

if user\_input:

break

print(Fore.RED + "Input cannot be empty. Please enter some text.")

self.parser.parse\_input(user\_input)

elif choice == "6":

print("Exiting the application.")

break

else:

print(Fore.RED + "Invalid choice. Try again.")

**./api/data\_repository.py:**

import requests

from api\_config import APIConfig

class DataRepository:

def \_\_init\_\_(self, resource):

self.api\_url = f"{APIConfig().base\_url}/{resource}"

def get\_data(self):

try:

response = requests.get(self.api\_url)

response.raise\_for\_status()

return response.json()

except requests.RequestException as e:

raise Exception(f"Error: Unable to fetch data - {e}")

**./api/input\_parser.py:**

import re

from colorama import Fore, Style, init

class InputParser:

def \_\_init\_\_(self):

self.patterns = {

"date": re.compile(r'\d{4}-\d{2}-\d{2}'),

"phone": re.compile(r'\+?\d{1,3}[-.\s]?\d{1,3}[-.\s]?\d{4,10}'),

"email": re.compile(r'^[\w\.-]+@[\w\.-]+\.\w{2,3}$'),

"id": re.compile(r'^\d+$'),

"name": re.compile(r'^[A-Za-zА-Яа-я]+$'),

}

def parse\_input(self, user\_input):

detected\_type = None

# Перевірка введених даних

for key, pattern in self.patterns.items():

if pattern.match(user\_input):

detected\_type = key

break

if detected\_type:

print(f"{detected\_type.capitalize()} detected in input.")

else:

print(Fore.RED + "No specific pattern detected or invalid input.")

**Висновки:** внаслідок виконання лабораторної роботи я розробила консольний об’єктно - орієнтований додаток з використанням API та патернів проектування

**GitHub:**[***https://github.com/viksi01/Calculator/tree/lab7***](https://github.com/viksi01/Calculator/tree/lab7)