МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра ІСМ



ЗВІТ

Про виконання лабораторної роботи №7

«Робота з API та веб-сервісами»

З дисципліни

«Спеціалізовані мови програмування»

Студентки групи РІ – 31

Нащецької Яни Вікторівни

Прийняв викладач

Щербак С.С.

**Лабораторна робота № 7. Робота з API та веб-сервісами.**

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

**План 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.

**Основний код програми:**

# src/api\_provider.py

import requests

import sys

import os

sys.path.append(os.path.join(os.path.dirname(\_\_file\_\_), 'src'))

class ApiProvider:

    BASE\_URL = "https://jsonplaceholder.typicode.com"

    def fetch\_data(self, endpoint):

        try:

            response = requests.get(f"{self.BASE\_URL}/{endpoint}")

            response.raise\_for\_status()

            return response.json()

        except requests.RequestException as e:

            raise Exception(f"Error fetching data from API: {e}")

# src/api\_repository.py

import requests

class APIRepository:

    def \_\_init\_\_(self, base\_url):

        self.base\_url = base\_url

    def get\_data(self, endpoint):

        url = f"{self.base\_url}/{endpoint}"

        response = requests.get(url)

        response.raise\_for\_status()

        return response.json()

# src/data\_parser.py

import re

class DataParser:

    def parse\_dates(self, text):

        return re.findall(r'\d{4}-\d{2}-\d{2}', text)

    def parse\_phones(self, text):

        return re.findall(r'\+?\d[\d\s-]{8,14}\d', text)

    def parse\_command(self, command\_str: str):

        # A simple implementation of command parsing

        if command\_str.lower() == "get posts":

            return "get posts"

        elif command\_str.lower() == "get users":

            return "get users"

        elif command\_str.lower() == "exit":

            return "exit"

        else:

            return None

def main():

    Logger.log("Лабораторна 7 запущена")

    api\_provider = ApiProvider()

    repository = DataRepository(api\_provider)

    unit\_of\_work = UnitOfWork(repository)

    ui = UserInterface()

    parser = DataParser()

    saver = DataSaver()

    error\_handler = ErrorHandler()

    logger = HistoryLogger()

    print("Welcome to the data management app!")

    while True:

        user\_input = input("\nEnter command ('get posts', 'get users', 'exit'): ")

        command = parser.parse\_command(user\_input)

        if command is None:

            print("Invalid command. Please try again.")

            continue

        try:

            if command == "exit":

                print("Exiting the program.")

                break

            elif command == "get posts":

                unit\_of\_work.load\_data("posts")

                data = unit\_of\_work.commit()

                ui.display\_table(data)

                logger.log\_request(command, data)

                saver.save\_as\_json(data)

                saver.save\_as\_csv(data)

            elif command == "get users":

                unit\_of\_work.load\_data("users")

                data = unit\_of\_work.commit()

                ui.display\_table(data)

                logger.log\_request(command, data)

                saver.save\_as\_json(data)

                saver.save\_as\_csv(data)

            else:

                print("Unknown command.")

        except Exception as e:

            error\_handler.handle\_error(e)

**Висновок:** За допомогою цієї лабораторної роботи я створила проект, який надає досвід роботи з API, дизайну користувацького інтерфейсу, валідації введення, обробки помилок та тестування.