мІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

нАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «лЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНОРМАЦІЙНИХ ТЕХНОЛОГІЙ

|  |
| --- |
|  |



ЗВІТ

Про виконання лабораторної роботи № 7

З дисципліни

«Спеціалізовані мови програмування»

Виконала:

студентка гр. ІТ-31

Катерина ПОПОВА

Прийняв:

Сергій ЩЕРБАК

Львів – 2023

**Тема роботи :** Робота з API та веб-сервісами

**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**

import requests

import json

import csv

import re

import logging

from prettytable import PrettyTable

logging.basicConfig(filename='book\_catalog.log', level=logging.INFO, format='%(asctime)s - %(levelname)s: %(message)s')

class BookAPIConnector:

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

self.base\_url = 'https://jsonplaceholder.typicode.com/posts'

def fetch\_books(self):

try:

resp = requests.get(self.base\_url)

resp.raise\_for\_status()

return resp.json()

except requests.RequestException as err:

logging.error(f"API call error: {err}")

return []

def create\_book(self, book\_info):

try:

resp = requests.post(self.base\_url, json=book\_info)

resp.raise\_for\_status()

return resp.json()

except requests.RequestException as err:

logging.error(f"API call error: {err}")

return None

class BookCatalog:

def \_\_init\_\_(self, api\_connector):

self.api\_connector = api\_connector

def retrieve\_books(self):

return self.api\_connector.fetch\_books()

def insert\_book(self, name, writer):

book\_info = {'title': name, 'author': writer}

return self.api\_connector.create\_book(book\_info)

def find\_book\_by\_title(self, title):

all\_books = self.retrieve\_books()

return [book for book in all\_books if title.lower() in book['title'].lower()]

class FileStorage:

@staticmethod

def store\_data(data, file\_format):

if file\_format == 'json':

with open('saved\_books.json', 'w') as file:

json.dump(data, file, indent=4)

elif file\_format == 'csv':

with open('saved\_books.csv', 'w', newline='') as file:

csv\_writer = csv.writer(file)

csv\_writer.writerows(data)

elif file\_format == 'txt':

with open('saved\_books.txt', 'w') as file:

file.write(str(data))

class BookLibraryInterface:

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

self.catalog = catalog

def operate(self):

while True:

print("\nChoose an action:")

print("1. View all books")

print("2. Add a new book")

print("3. Look up a book by title")

print("4. Quit")

selection = input("Your choice (1-4): ")

if selection == '1':

books = self.catalog.retrieve\_books()

if books:

book\_table = PrettyTable()

book\_table.field\_names = ["ID", "Title", "Author"]

for book in books:

book\_table.add\_row([book['id'], book['title'], book.get('author', 'N/A')])

print(book\_table)

store\_decision = input("Save these books? (yes/no): ")

if store\_decision.lower() == 'yes':

FileStorage.store\_data(book\_table.get\_string(), 'txt')

print("Books stored in saved\_books.txt")

else:

print("No books found.")

elif selection == '2':

book\_title = input("Book title: ")

book\_author = input("Book author: ")

if self.validate\_entry(r'^[A-Za-z0-9\s]+$', book\_title) and self.validate\_entry(r'^[A-Za-z\s]+$', book\_author):

new\_book = self.catalog.insert\_book(book\_title, book\_author)

if new\_book:

print(f"New book added: {new\_book}")

else:

print("Could not add the book.")

else:

print("Invalid input. Please enter a valid title and author.")

elif selection == '3':

search\_title = input("Book title to search: ")

found\_books = self.catalog.find\_book\_by\_title(search\_title)

print("Books found:", found\_books)

elif selection == '4':

print("Exiting the Book Library Interface.")

break

else:

print("Invalid selection. Please choose a number between 1 and 4.")

@staticmethod

def validate\_entry(pattern, text):

return bool(re.match(pattern, text))

def start\_application():

api\_connector = BookAPIConnector()

catalog = BookCatalog(api\_connector)

interface = BookLibraryInterface(catalog)

interface.operate()

if \_\_name\_\_ == '\_\_main\_\_':

start\_application()

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