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-СЕРВІСАМИ

Виконав:
ст. гр. ІТ-32
Шоха А.А.
Прийняв:
Щербак С.С.

Львів-2023

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

Завдання на лабораторну роботу

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.

Хід роботи

Код класу що виконує дану лабораторну:

```
from prettytable import PrettyTable
import google_auth_oauthlib.flow
import google.auth.transport.requests
import sys
sys.path.append('/Users/admin/Desktop/lpnu/5 сем/Specialised programming languages/source')
from config.paths_config import GOOGLE_BOOKS_API_CREDENTIALS, GOOGLE_BOOKS_API_OUTPUT
from shared.json_processor import JSONProcessor
class GoogleBooksApiService:
    """A simple Google Books API service class."""
    def __init__(self):
        self.credentials = self.get_credentials()

    def get_credentials(self):
                                                                flow = google_auth_oauthlib.flow.InstalledAppFlow.from_client_secrets_file(
            GOOGLE_BOOKS_API_CREDENTIALS,
            scopes=['https://www.googleapis.com/auth/books']
        )
        flow.run_local_server(port=0)
        return flow.credentials

    def search_books_by_title(self, credentials, title_query):
                                                            session = google.auth.transport.requests.AuthorizedSession(credentials)
        url = "https://www.googleapis.com/books/v1/volumes"
        params = {"q": title_query}
        response = session.get(url, params=params)
        return response.json().get('items', [])
    def search_books_by_author(self, author_query):
                                                            session = google.auth.transport.requests.AuthorizedSession(self.credentials)
        url = "https://www.googleapis.com/books/v1/volumes"
        params = {"q": f"inauthor:{author_query}"}
        response = session.get(url, params=params)
                            JSONProcessor.save_data(GOOGLE_BOOKS_API_OUTPUT, f'books_by_{author_query}', response.json().get('items', []), 'json')
        return response.json().get('items', [])
    def search_books_by_isbn(self, isbn_query):
```

```
                                                                session       =
google.auth.transport.requests.AuthorizedSession(self.credentials)
        url = "https://www.googleapis.com/books/v1/volumes"
        params = {"q": f"isbn:{isbn_query}"}
        response = session.get(url, params=params)
        return response.json().get('items', [])
    def list_new_releases(self, category="fiction", max_results=10):
                                                                session       =
google.auth.transport.requests.AuthorizedSession(self.credentials)
        url = "https://www.googleapis.com/books/v1/volumes"
               params = {"q": f"subject:{category}", "orderBy": "newest",
"maxResults": max_results}
        response = session.get(url, params=params)
        return response.json().get('items', [])
    def get_popular_books_in_category(self, category, max_results=10):
                                                                session       =
google.auth.transport.requests.AuthorizedSession(self.credentials)
        url = "https://www.googleapis.com/books/v1/volumes"
             params = {"q": f"subject:{category}", "orderBy": "relevance",
"maxResults": max_results}
        response = session.get(url, params=params)
        return response.json().get('items', [])
    def display_data(self, data, field_names, entity_name):
        if not data:
            print(f"No {entity_name} data to display")
            return None
        table = PrettyTable()
        table.field_names = field_names
        for idx, item in enumerate(data, start=1):
            book_info = item.get('volumeInfo', {})
            row = [idx]
            for field in field_names[1:]:  # Skip the index column
                if field == "Title":
                    title = book_info.get('title', 'N/A')
                    row.append(title)
                elif field == "Author(s)":
                                authors = ', '.join(book_info.get('authors',
['Unknown']))
                    row.append(authors)
                elif field == "Published Date":
                    published_date = book_info.get('publishedDate', 'N/A')
                    row.append(published_date)
                elif field == "ISBN":
                          isbn = ', '.join([identifier.get('identifier') for
identifier      in      book_info.get('industryIdentifiers',      [])      if
identifier.get('type') in ['ISBN_10', 'ISBN_13']])
                    row.append(isbn or 'N/A')
                else:
                    row.append('N/A')
            table.add_row(row)
        print(f"{entity_name.capitalize()} data:")
        print(table)
```

```
    def display_book_details(self, credentials, title_query):
        books = self.search_books_by_title(title_query)
        field_names = ["#", "Title", "Author(s)", "Page Count", "Categories",
"Language"]
        self.display_data(books, field_names, "book")
    def display_books_by_author(self, credentials, author_query):
        books = self.search_books_by_author(author_query)
        field_names = ["#", "Title", "Author(s)", "Published Date", "ISBN"]
        self.display_data(books, field_names, "books by author")
    def display_books_by_isbn(self, credentials, isbn_query):
        books = self.search_books_by_isbn(isbn_query)
        field_names = ["#", "Title", "Author(s)", "Published Date"]
        self.display_data(books, field_names, "books by ISBN")
    def display_new_releases(self, category="fiction", max_results=10):
        books = self.list_new_releases(category, max_results)
        field_names = ["#", "Title", "Author(s)", "Published Date"]
        self.display_data(books, field_names, "new releases")
    def display_popular_books_in_category(self, category, max_results=10):
        books = self.get_popular_books_in_category(category, max_results)
        field_names = ["#", "Title", "Author(s)", "Published Date"]
        self.display_data(books, field_names, "popular books in category")
```

Приклад роботи програми:

Рис. 1. Результат запиту api

Висновок: під час виконання лабораторної роботи було створено консольний об’єктно - орієнтований додаток з використанням API