Kacper Grabalski 2ID12A
Krzysztof Giemza 2ID12A
Projekt Języki Skryptowe
Temat: Lokalna sieć wymiany książek

Dr inż. Dariusz Michalski. Formularz samooceny do projektu z języków skryptowych

Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
1	UI	JEST		☐		
		Wprowadzanie danych	user_interface.py Kod: def add_user(users): name = input("Imię i nazwisko: ") user_id = max([u.user_id for u in users], default=0) + 1 users.append(User(name, user_id))	☐		2

		Wyświetlanie danych	user_interface.py
			Kod:
			def search_books(books):
			print("\nZnalezione książki:")
			for book in found_books:
			print(f"{book.title} - {book.author} ({book.genre})")
				☐		2

		Zmiana danych	user_interface.py kod: book_a.owner_id = user_b_id book_b.owner_id = user_a_id	☐		2
		Wyszukiwanie danych	user_interface.py Kod: found_books = list(filter(lambda b: term in b.title.lower() or term in b.author.lower() or term in b.genre.lower(), books))	☐		2
		Przedstawienie wyników	visualization.py kod: plt.bar(genre_count.keys(), genre_count.values()) plt.title("Rozkład książek według gatunków") plt.savefig("books_genre_chart.png")	☐		2
2	Podstawy	Zmienne	main.py kod: users, books = load_data()	☐		2
		typy danych	Stringi, inty, listy, słowniki np. lista w user_interface.py def display_stats(books): genres = [b.genre for b in books]	☐		2
		komentarze	main.py # Wczytanie danych przy starcie aplikacji	☐		1
		operatory	storage.py kod: if book_data['book_id'] == book_id_to_find: # ...	☐		1,5
		Instrukcje warunkowe (if, elif, else)	user_interface.py kod: choice = input("Wybierz opcję: ")	☐		3

			if choice == "1":
			add_user(users, books)
			elif choice == "2":
			add_book(users, books)
			elif choice == "3":			

		Instrukcje iteracyjne				
		for	user_interface.py kod: for book in found_books: print(f"{book.title} - {book.author} ({book.genre})")	☐		2
		while	user_interface.py kod: while True: print("\n=== Lokalna Sieć Wymiany Książek ===") print("1. Dodaj użytkownika") print("2. Dodaj książkę") print("3. Wyszukaj książki") print("4. Wymień książkę") print("5. Wyświetl statystyki i użytkowników") print("6. Wyszukaj użytkownika") print("7. Wyjście") choice = input("Wybierz opcję: ")	☐		2
		Operacje wejścia (input)	user_interface.py kod: choice = input("Wybierz opcję: ")	☐		1,5
		Operacje wyjścia (print)	main.py kod; print("Dane zostały zapisane. Do widzenia!")	☐		1,5

		Funkcje z parametrami i wartościami zwracanymi	user_interface.py
			kod:
			def search_books(books):
			term = input("Szukaj (tytuł/autor/gatunek): ").lower()
			found_books = list(filter(lambda b: term in b.title.lower()
			or term in b.author.lower()
			or term in b.genre.lower(), books))
			
			if not found_books:
			print("Brak wyników!")
			return
			
			print("\nZnalezione książki:")
			for book in found_books:
			print(f"{book.title} - {book.author} ({book.genre})")
				☐		2

		Funkcje rekurencyjne	utils.py
			kod:
			def recursive_factorial(n):
			if n == 0:
			return 1
			return n * recursive_factorial(n-1)
				☐		3

		Funkcje przyjmujące inne funkcje jako	search_books.py funkcja fillter kod: def search_books(books): term = input("Szukaj	☐		3

		argumenty	(tytuł/autor/gatunek): ").lower() found_books = list(filter(lambda b: term in b.title.lower() or term in b.author.lower() or term in b.genre.lower(), books))			

		Dekoratory	utils.py
			kod:
			def log_activity(func):
			def wrapper(*args, **kwargs):
			print(f"Wywołano: {func.__name__}")
			return func(*args, **kwargs)
			return wrapper
				☐		1,5

3	Kontenery	Użycie listy	user.py kod: self.books_owned = books_owned if books_owned else []	☐		2
		Użycie słownika	user_interface.py kod: genre_count = {genre: genres.count(genre) for genre in unique_genres}	☐		2
		Użycie zbioru	user_interface.py kod: unique_genres = set(genres)	☐		1,5
		Użycie krotki	storage.py kod: return users, books	☐		1,5

4	Przestrzenie nazw	Zastosowano zmienne lokalne	user_interface.py
			kod:
			def add_user(users):
			name = input("Imię i nazwisko: ") # Zmienna lokalna 'name'
			user_id = max([u.user_id for u in users], default=0) + 1 # Zmienna lokalna 'user_id'
			users.append(User(name, user_id))
				☐		1,5

		Zastosowano zmienne globalne	storage.py kod: DATA_FILE = "library_data.json"	☐		1,5
		Zastosowano zakresy funkcji	utils.py (funkcja zagnieżdżona) kod: def log_activity(func): def wrapper(*args, **kwargs): # Funkcja wewnętrzna # ... return wrapper	☐		1,5

		Zastosowano zakresy klas	user.py
			kod:
			class User:
			def __init__(self, name, user_id): # Metoda klasy
			self.name = name # Atrybut instancji
				☐		1,5

5	Moduły i pakiety	Projekt podzielony na moduły (import, __init__)		☐		2

Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
		Własne pakiety/funkcje pomocnicze w osobnych plikach .py	Struktura plików: main.py storage.py user_interface.py user.py book.py utils.py visualization.py	☐		2

6	Obsługa błędów	Obsługa wyjątków (try, except, finally)	storage.py
			kod:
			try:
			  with open(DATA_FILE, 'r') as f:
			  data = json.load(f)
			except (FileNotFoundError, json.JSONDecodeError):
			  return [], []
				☐		2

		Użycie assert do testów i walidacji	test_units.py
			kod:
			def test_book_creation(self):
			  book = Book("Python Basics", "A. Smith", 1, "Programming", 101)
			  self.assertEqual(book.title, "Python Basics")
				☐		1,5

7	Łańcuchy znaków	Operacje na stringach (m.in. formatowanie, dzielenie, wyszukiwanie)	user_interface.py kod: term = input("Szukaj (tytuł/autor/gatunek): ").lower() if term in b.title.lower() or term in b.author.lower()	☐		2

8	Obsługa plików	Odczyt z plików .txt, .csv, .json, .xml
		(min. 1)
			storage.py
			kod:
			# Zapis
			with open(DATA_FILE, 'w') as f:
			  json.dump(data, f, indent=2)
			
			# Odczyt
			with open(DATA_FILE, 'r') as f:
			  data = json.load(f)
				☐		2

		Zapis do plików .txt, .csv, .json, .xml (min. 1)	visualization.py kod: plt.savefig("books_genre_chart.png")	☐		2

9	OOP	Klasy	user.py i book.py
			kod:
			# user.py
			class User:
			  # …
			
			# book.py
			class Book:
			  # ...
				☐		2

		Metody	user.py kod: def add_book(self, book_id): self.books_owned.append(book_id)	☐		2
		Konstruktory	book.py kod: def __init__(self, title, author, book_id, genre, owner_id): self.title = title	☐		2

			self.author = author # ...			
		Dziedziczenie		☐		2
10	Programowanie funkcyjne	map		☐		1,5
		filter	user_interface.py kod: found_books = list(filter(lambda b: term in b.title.lower(), books))	☐		1,5
		lambda	user_interface.py kod: list(filter(lambda b: term in b.title.lower(), books))	☐		1,5

		reduce	storage.py
			kod:
			from functools import reduce
			# Przykładowe użycie reduce do sumowania
			total_books = reduce(lambda x, y: x + len(y.books_owned), users, 0)
				☐		1,5

11	Wizualizacja danych	Wygenerowano wykres (np. matplotlib, seaborn)	visualization.py kod: plt.bar(genre_count.keys(), genre_count.values()) plt.savefig("books_genre_chart.png")	☐		2
		Zapisano wykres do pliku graficznego (.png lub .jpg)	visualization.py kod: plt.bar(genre_count.keys(), genre_count.values()) plt.savefig("books_genre_chart.png")	☐		1,5

T12	Testowanie	Testy jednostkowe (assert, unittest, pytest)	test_units.py
			kod:
			class TestLibrary(unittest.TestCase):
			def test_book_creation(self):
			book = Book("Python Basics", "A. Smith", 1, "Programming", 101)
			self.assertEqual(book.title, "Python Basics")
				☐		1,5

		Testy funkcjonalne	test_functional.py kod: @patch('builtins.input', side_effect=['7']) def test_main_menu_exit(self, mock_input): # ...	☐		1,5
		Testy Integracyjne	test_integration.py kod: def test_exchange_flow(self):	☐		1,5
		Testy graniczne / błędne dane	test_boundary.py kod: def test_large_data(self): # Test z 1000 użytkowników i 10000 książek	☐		1,5
		Testy wydajności (np. czas wykonania, timeit)	test_performance.py kod: save_time = timeit.timeit("test_save()", number=10)	☐		1,5

		Testy pamięci memory_profiler	test_memory.py kod: @profile def memory_test(): # Test użycia pamięci	☐		1,5
		Test jakości kodu (flake8, pylint)		☐		1,5
13	Wersjonowanie	Repozytorium GIT	https://github.com/DeskaV2/Projekt_Skryptowe.git	☐		1
		Historia commitów		☐		1

Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
		Link do GitHub	https://github.com/DeskaV2/Projekt_Skryptowe.git	☐		1
		Opis commitów		☐		1
14	Dokumentacja	Plik README.md (cel, autorzy, uruchamianie)	Jest utworzony i może zostać otwarty po pobraniu repozytorium	☐		1,5
		Przykładowe dane wejściowe i wyjściowe	Wybierz opcję: 1 Imię i nazwisko: Jan Kowalski Utworzono użytkownika: Jan Kowalski (ID: 1)	☐		2
		Diagram klas lub struktura modułów	. ├── main.py # Główny punkt wejścia ├── storage.py # Obsługa danych (JSON) ├── user_interface.py # Interfejs użytkownika ├── user.py # Model użytkownika ├── book.py # Model książki ├── utils.py # Funkcje pomocnicze ├── visualization.py # Generowanie wykresów ├── test_unit.py # Testy jednostkowe ├── test_fun.py # Testy funkcjonalne ├── test_integ.py # Testy integracyjne ├── test_gran.py # Testy graniczne ├── test_wyda.py # Testy wydajności ├── test_pam.py # Testy pamięci └── library_data.json # Baza danych (automatycznie tworzona)	☐		2
			SUMA			