

---

*Przedmiot: Inżynieria Oprogramowania*

*Zadanie: Projekt aplikacji*

*Temat: System biblioteczny*

*Wykonat: Piotr Zakrzewski 11972, IV semestr, INF-NST*

*Data: 27.05.2025*

*Link do repozytorium GitHub:*

*[https://github.com/shrimp050/system\\_biblioteczny](https://github.com/shrimp050/system_biblioteczny)*

---

## Cel projektu

Celem projektu było stworzenie prostego systemu zarządzania biblioteką, który umożliwia:

- dodawanie, usuwanie i edytowanie książek,
- wypożyczanie, zwracanie i rezerwowanie książek,
- wyszukiwanie książek,
- logowanie i rejestrowanie się użytkowników,
- tworzenie i modyfikowanie kont użytkowników,
- testowanie funkcjonalności.

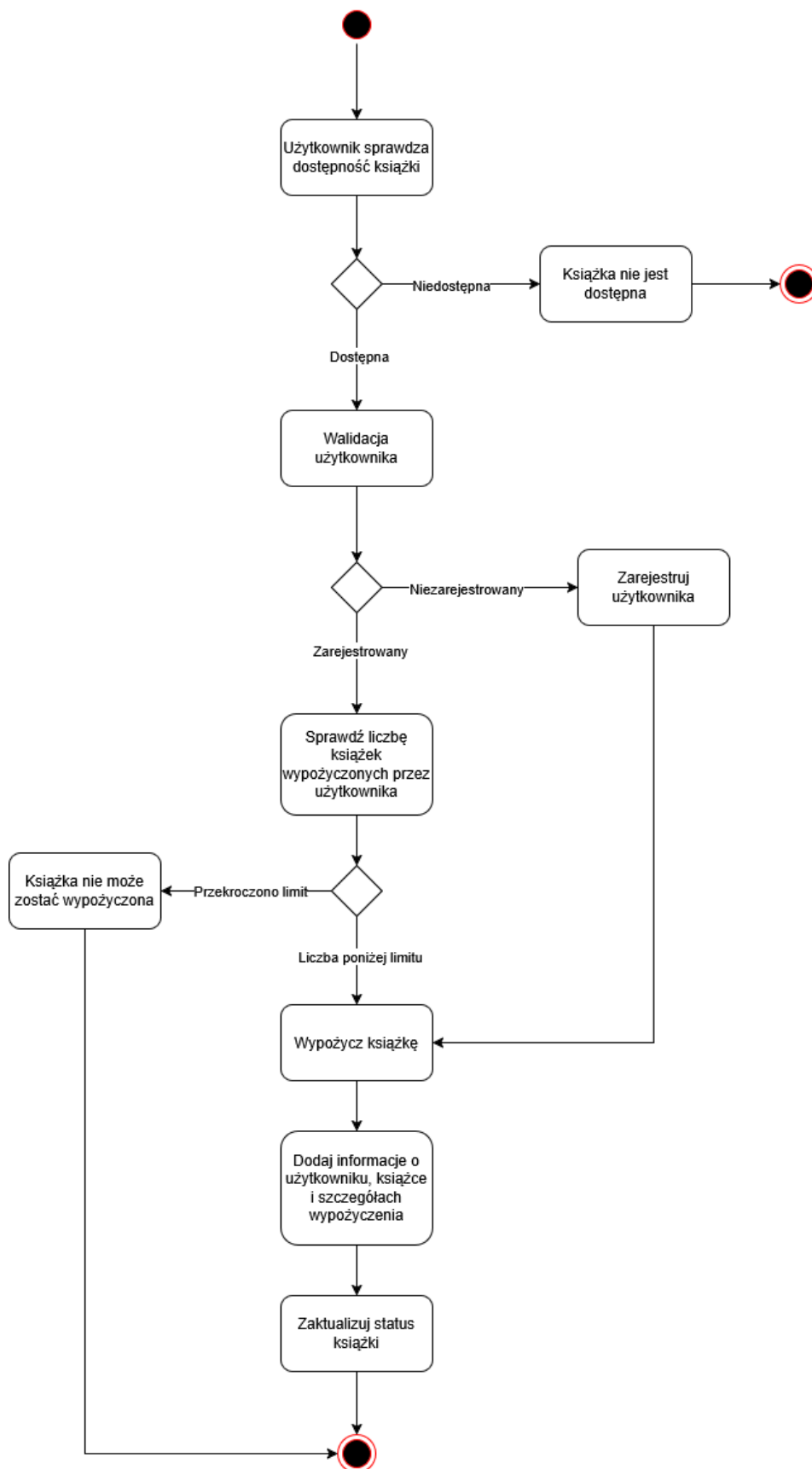
Przykładowe funkcje ilustrujące wybrane elementy funkcjonalności oraz odpowiadające im testy zostały zaimplementowane w języku Python.

## Diagramy UML

W celu zobrazowania struktury oraz działania aplikacji bibliotecznej, opracowano zestaw diagramów UML. Diagramy te pomagają w zrozumieniu architektury systemu, przepływu danych oraz interakcji pomiędzy użytkownikiem a aplikacją. Użyto następujących typów diagramów:

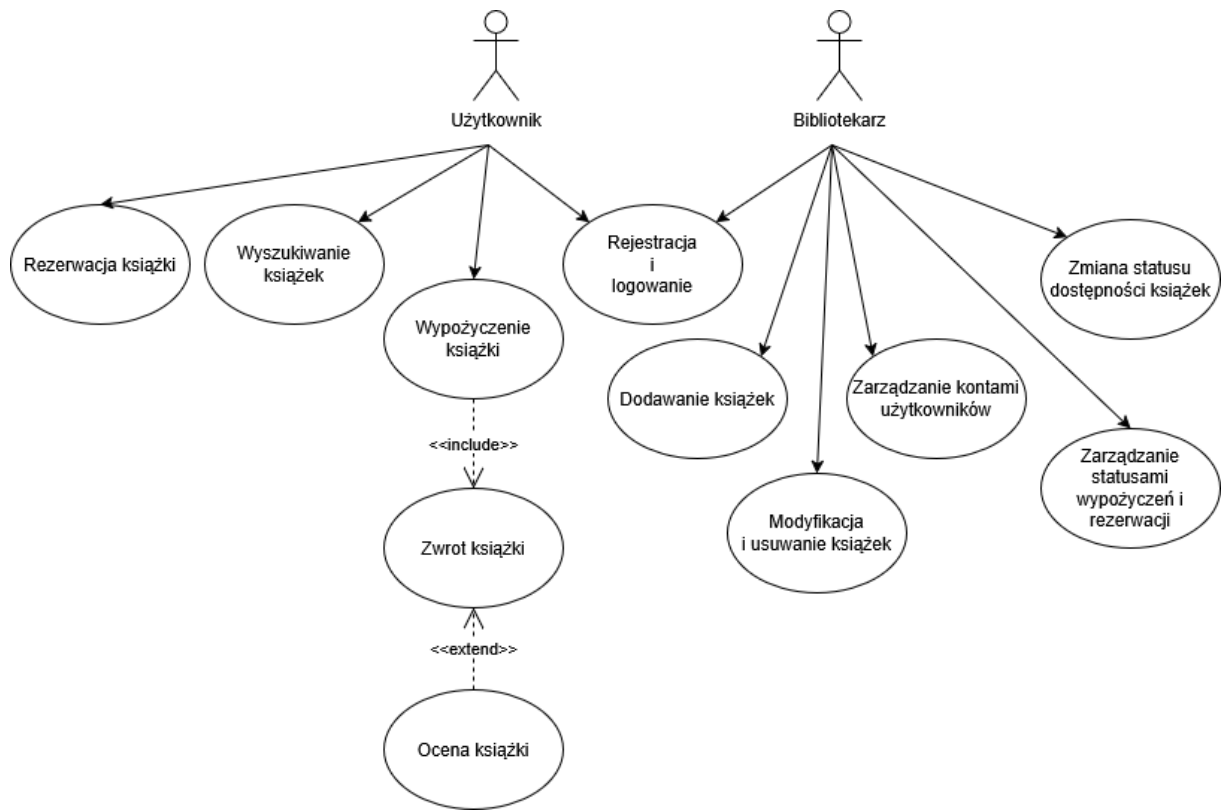
- **Diagram przypadków użycia** - przedstawia główne funkcjonalności systemu z punktu widzenia użytkownika
- **Diagram czynności** - obrazuje przebieg operacji takich jak wypożyczenie lub zwrot książki, uwzględniając możliwe warunki i decyzje.
- **Diagram klas** - prezentuje strukturę danych w systemie, klasy oraz ich atrybuty i relacje.
- **Diagram sekwencji** - pokazuje szczegółowy przebieg interakcji między obiektami w czasie dla wybranych przypadków użycia.

## Diagram czynności



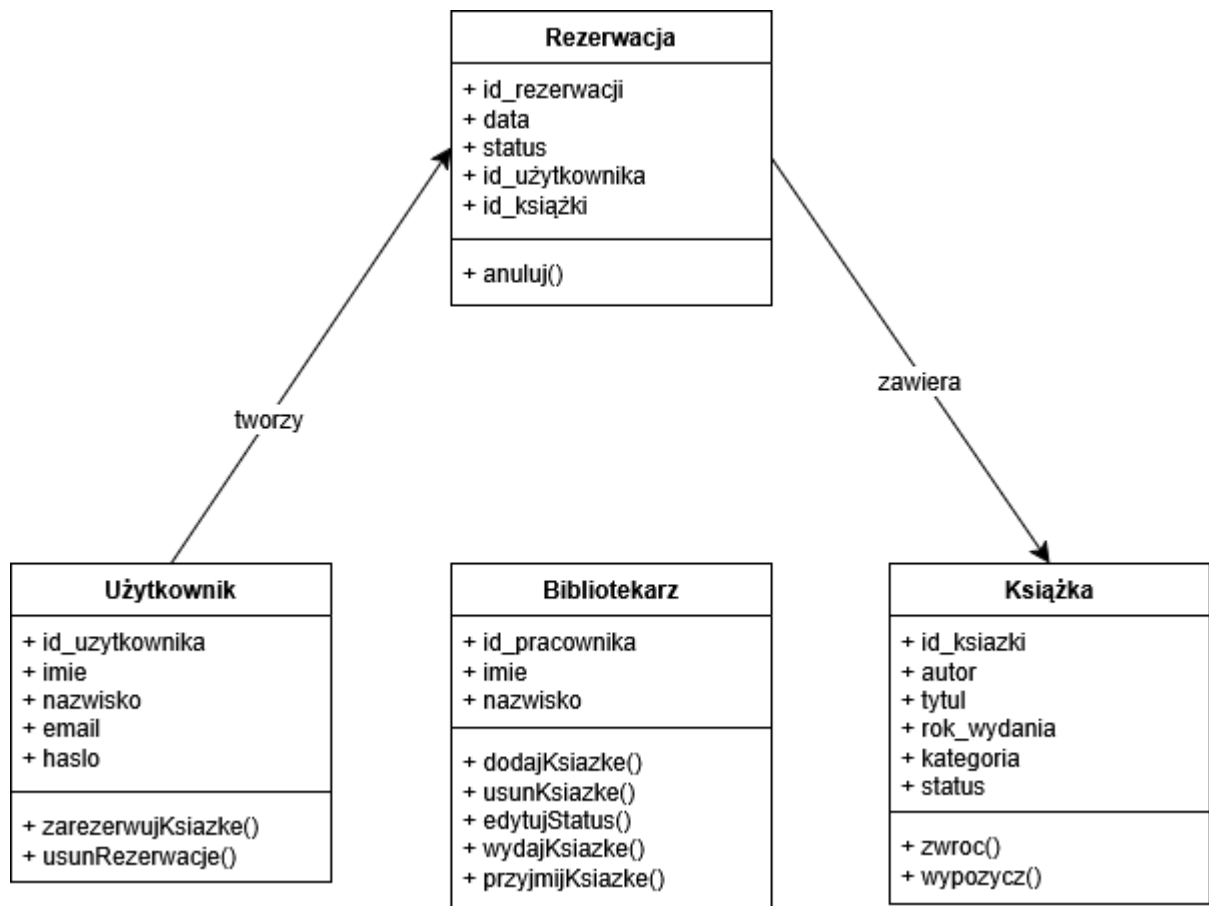
Rysunek 1 Diagram czynności dla funkcji wypożyczenia książki

## Diagram przypadków użycia



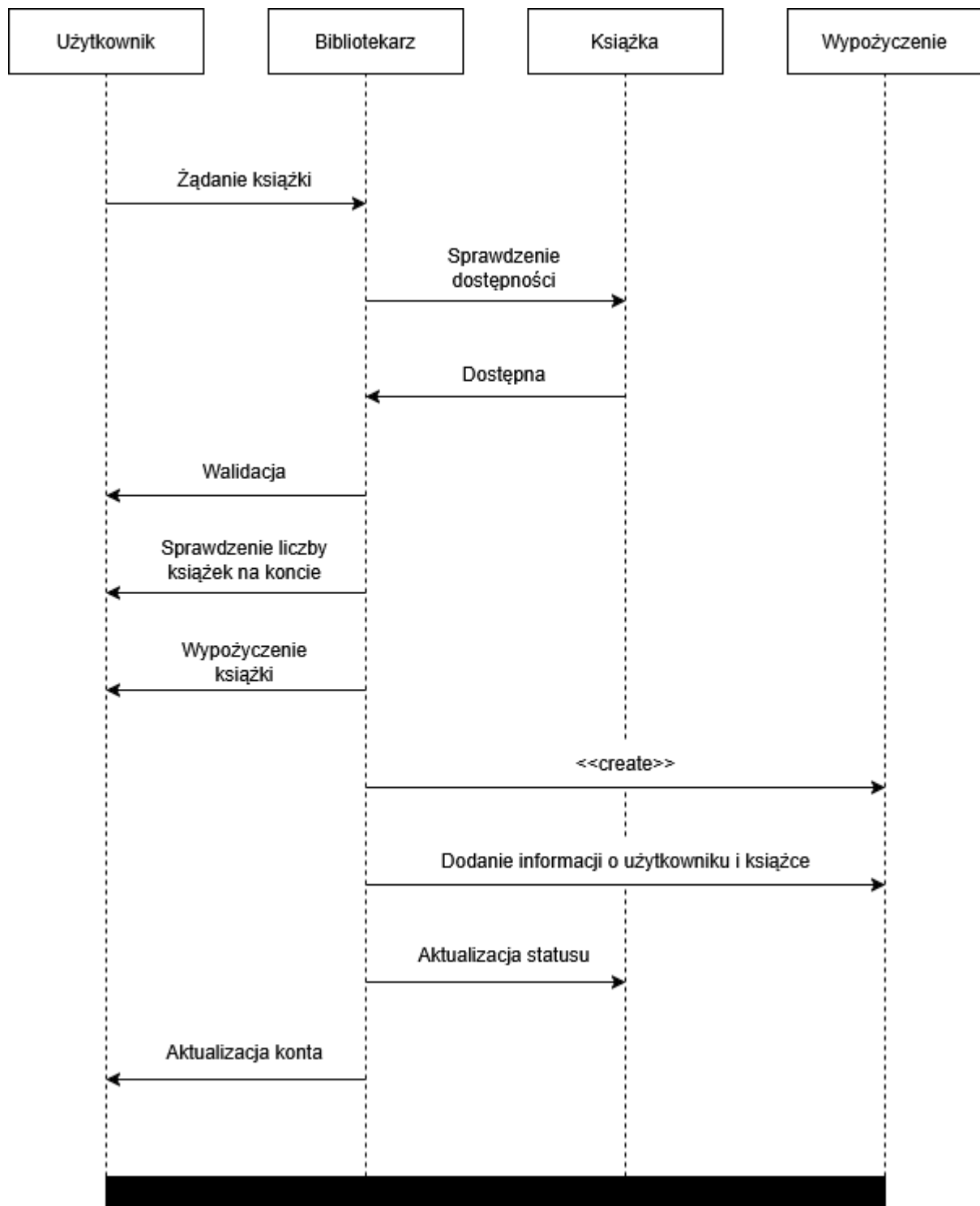
Rysunek 2 Diagram przypadków użycia z aktorami Użytkownik oraz Bibliotekarz

## Diagram klas



Rysunek 3 Diagram klas z odpowiednimi zmiennymi i metodami

## Diagram sekwencji



Rysunek 4 Diagram sekwencji - proces wypożyczania książki

## Funkcje i testy

Poniżej przedstawiono 5 przykładowych funkcji programu wraz z 4 testami jednostkowymi dla każdej z nich.

### Główny program

```
def add_book(library, book_id, title, author):
    if book_id in library:
        return "Book already exists"
    library[book_id] = {"title": title, "author": author, "available": True}
    return "Book added"

def search_book(library, title):
    return [book for book in library.values() if title.lower() in book["title"].lower()]

def borrow_book(library, book_id):
    if book_id not in library:
        return "Book not found"
    if not library[book_id]["available"]:
        return "Book not available"
    library[book_id]["available"] = False
    return "Book borrowed"

def return_book(library, book_id):
    if book_id not in library:
        return "Book not found"
    if library[book_id]["available"]:
        return "Book was not borrowed"
    library[book_id]["available"] = True
    return "Book returned"

def remove_book(library, book_id):
    if book_id in library:
        del library[book_id]
        return "Book removed"
    return "Book not found"
```

### Testy jednostkowe

```
from library import add_book, search_book, borrow_book, return_book, remove_book

# Testy add_book
def test_add_book_success():
    lib = {}
    assert add_book(lib, 1, "1984", "Orwell") == "Book added"

def test_add_book_duplicate():
    lib = {1: {"title": "1984", "author": "Orwell", "available": True}}
    assert add_book(lib, 1, "1984", "Orwell") == "Book already exists"

def test_add_book_multiple():
    lib = {}
```

```

add_book(lib, 1, "Book A", "Author A")
add_book(lib, 2, "Book B", "Author B")
assert len(lib) == 2

def test_add_book_data_check():
    lib = {}
    add_book(lib, 3, "Brave New World", "Huxley")
    assert lib[3]["author"] == "Huxley"

# Testy search_book
def test_search_book_found():
    lib = {1: {"title": "1984", "author": "Orwell", "available": True}}
    assert len(search_book(lib, "1984")) == 1

def test_search_book_not_found():
    lib = {1: {"title": "1984", "author": "Orwell", "available": True}}
    assert len(search_book(lib, "Brave")) == 0

def test_search_book_partial_match():
    lib = {1: {"title": "Brave New World", "author": "Huxley", "available": True}}
    assert len(search_book(lib, "brave")) == 1

def test_search_book_case_insensitive():
    lib = {1: {"title": "The Hobbit", "author": "Tolkien", "available": True}}
    assert len(search_book(lib, "hobbit")) == 1

# Testy borrow_book
def test_borrow_book_success():
    lib = {1: {"title": "1984", "author": "Orwell", "available": True}}
    assert borrow_book(lib, 1) == "Book borrowed"

def test_borrow_book_not_found():
    lib = {}
    assert borrow_book(lib, 1) == "Book not found"

def test_borrow_book_unavailable():
    lib = {1: {"title": "1984", "author": "Orwell", "available": False}}
    assert borrow_book(lib, 1) == "Book not available"

def test_borrow_book_flag_changes():
    lib = {1: {"title": "1984", "author": "Orwell", "available": True}}
    borrow_book(lib, 1)
    assert lib[1]["available"] is False

# Testy return_book
def test_return_book_success():
    lib = {1: {"title": "1984", "author": "Orwell", "available": False}}
    assert return_book(lib, 1) == "Book returned"

def test_return_book_not_found():
    lib = {}
    assert return_book(lib, 1) == "Book not found"

def test_return_book_not_borrowed():
    lib = {1: {"title": "1984", "author": "Orwell", "available": True}}
    assert return_book(lib, 1) == "Book was not borrowed"

```



```
def test_return_book_flag_changes():
    lib = {1: {"title": "1984", "author": "Orwell", "available": False}}
    return_book(lib, 1)
    assert lib[1]["available"] is True

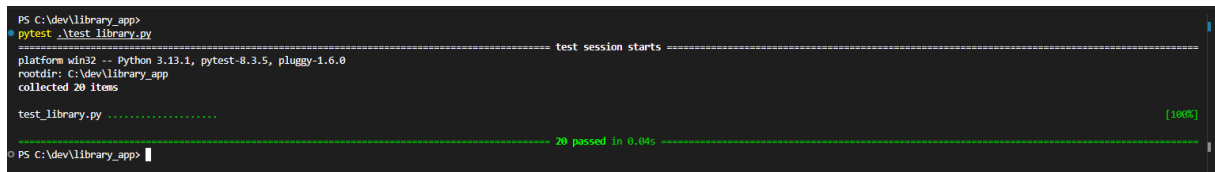
# Testy remove_book
def test_remove_book_success():
    lib = {1: {"title": "1984", "author": "Orwell", "available": True}}
    assert remove_book(lib, 1) == "Book removed"

def test_remove_book_not_found():
    lib = {}
    assert remove_book(lib, 1) == "Book not found"

def test_remove_book_effect():
    lib = {1: {"title": "1984", "author": "Orwell", "available": True}}
    remove_book(lib, 1)
    assert 1 not in lib

def test_remove_book_multiple():
    lib = {
        1: {"title": "Book A", "author": "A", "available": True},
        2: {"title": "Book B", "author": "B", "available": True},
    }
    remove_book(lib, 2)
    assert len(lib) == 1
```

## Działanie testów



```
PS C:\dev\library_app>
pytest .\test_library.py
----- test session starts -----
platform win32 -- Python 3.13.1, pytest-8.3.5, pluggy-1.6.0
rootdir: C:\dev\library_app
collected 20 items

test_library.py ..... [100%]

----- 20 passed in 0.04s -----
PS C:\dev\library_app>
```

Rysunek 5 Zrzut ekranu przedstawiający wynik testów jednostkowych

## Podsumowanie

Zrealizowany projekt stanowi prostą aplikację biblioteczną umożliwiającą podstawowe operacje zarządzania książkami, takie jak dodawanie, usuwanie, wypożyczanie, zwracanie oraz wyszukiwanie pozycji według tytułu. Funkcjonalności zostały zaimplementowane w języku Python i przetestowane przy użyciu testów jednostkowych, co pozwala na weryfikację poprawności działania poszczególnych funkcji.

W ramach projektu przygotowano także zestaw diagramów UML (przypadków użycia, czynności, klas oraz sekwencji), które obrazują strukturę systemu, zależności między komponentami oraz przebieg głównych operacji.

Projekt jest wersjonowany i udostępniony w repozytorium GitHub, co umożliwia śledzenie postępów i wprowadzanych zmian.

Opracowana aplikacja może stanowić podstawę do dalszej rozbudowy, np. o interfejs graficzny, bazę danych czy autoryzację użytkowników. Projekt spełnia założone wymagania i stanowi dobrą bazę edukacyjną do nauki projektowania i testowania oprogramowania.