Sprawozdanie z projektu: Aplikacja do zarządzania wypożyczalnią filmów online

# 1. Cel projektu

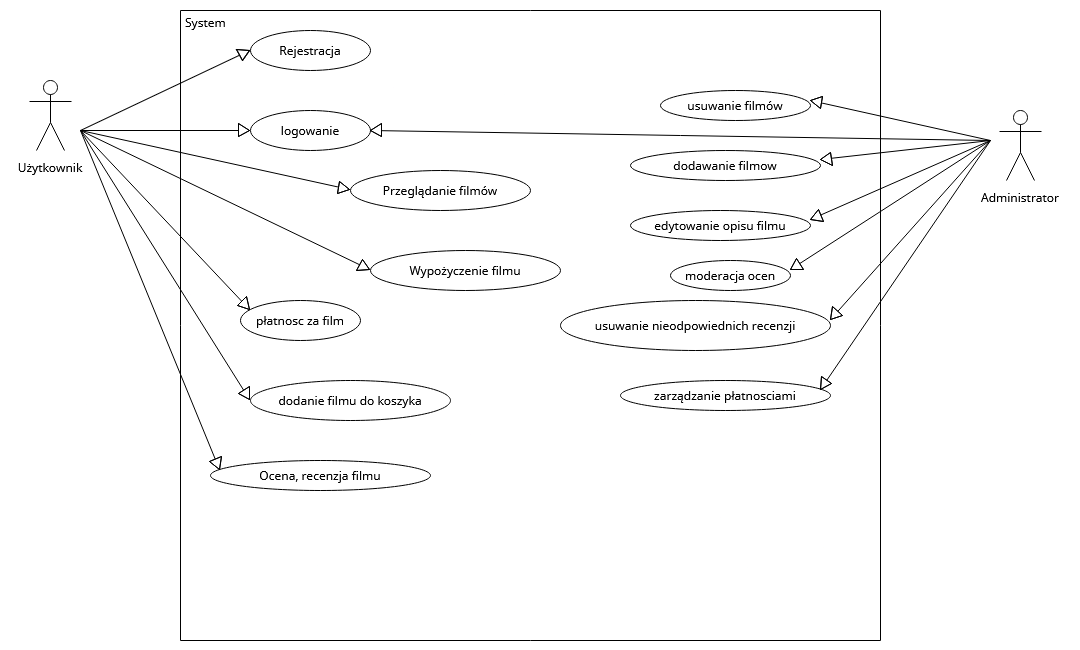
Celem projektu jest stworzenie aplikacji internetowej umożliwiającej użytkownikom przeglądanie, wypożyczanie oraz zarządzanie filmami online. Aplikacja ma umożliwiać rejestrację i logowanie użytkowników, dodawanie filmów, przeszukiwanie katalogu, wypożyczanie filmów oraz dodawanie opinii.

# 2. Główne funkcjonalności

- Rejestracja i logowanie użytkowników  
- Dodawanie i edytowanie filmów  
- Wyszukiwanie filmów  
- Wypożyczanie filmów  
- Wystawianie recenzji  
- Obsługa płatności

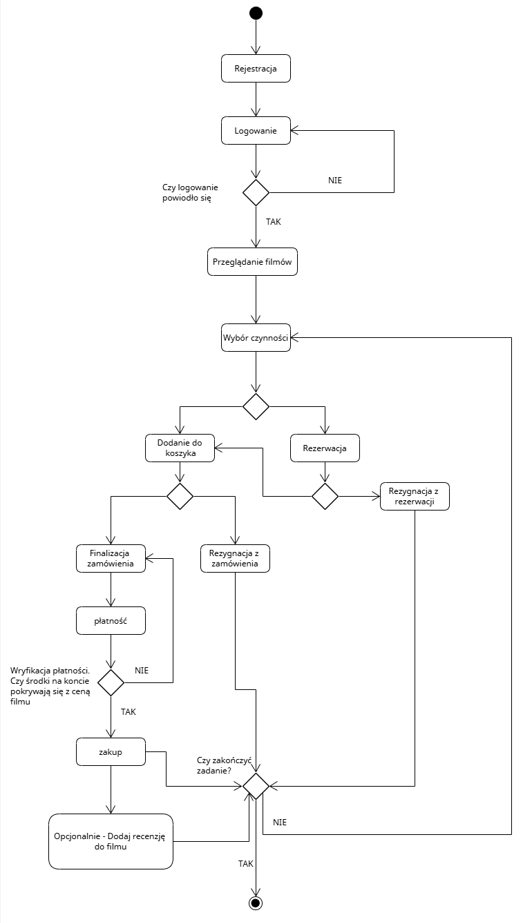
# 3. Diagramy UML

W ramach projektu wykonano następujące diagramy UML:  
- Diagram przypadków użycia:



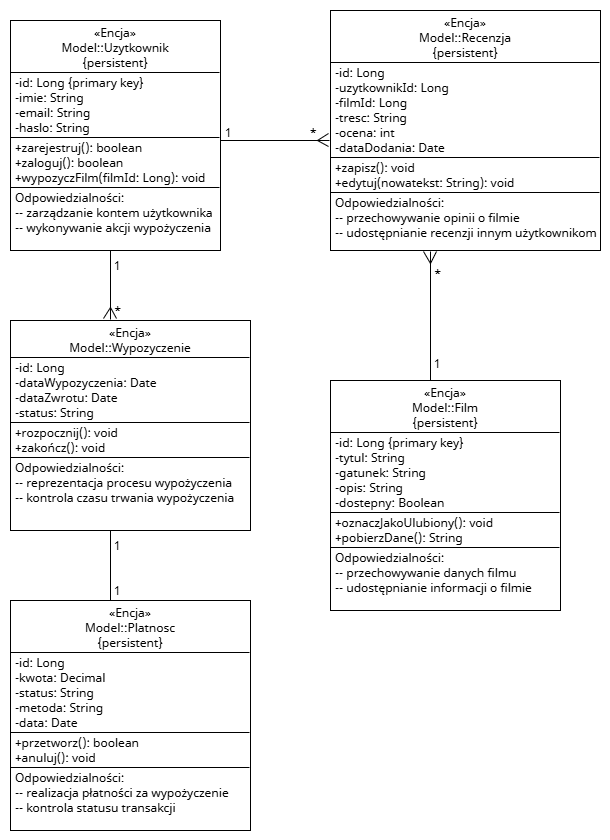
Rysunek 1 przedstawiający diagram przypadków użycia.

- Diagram czynności:



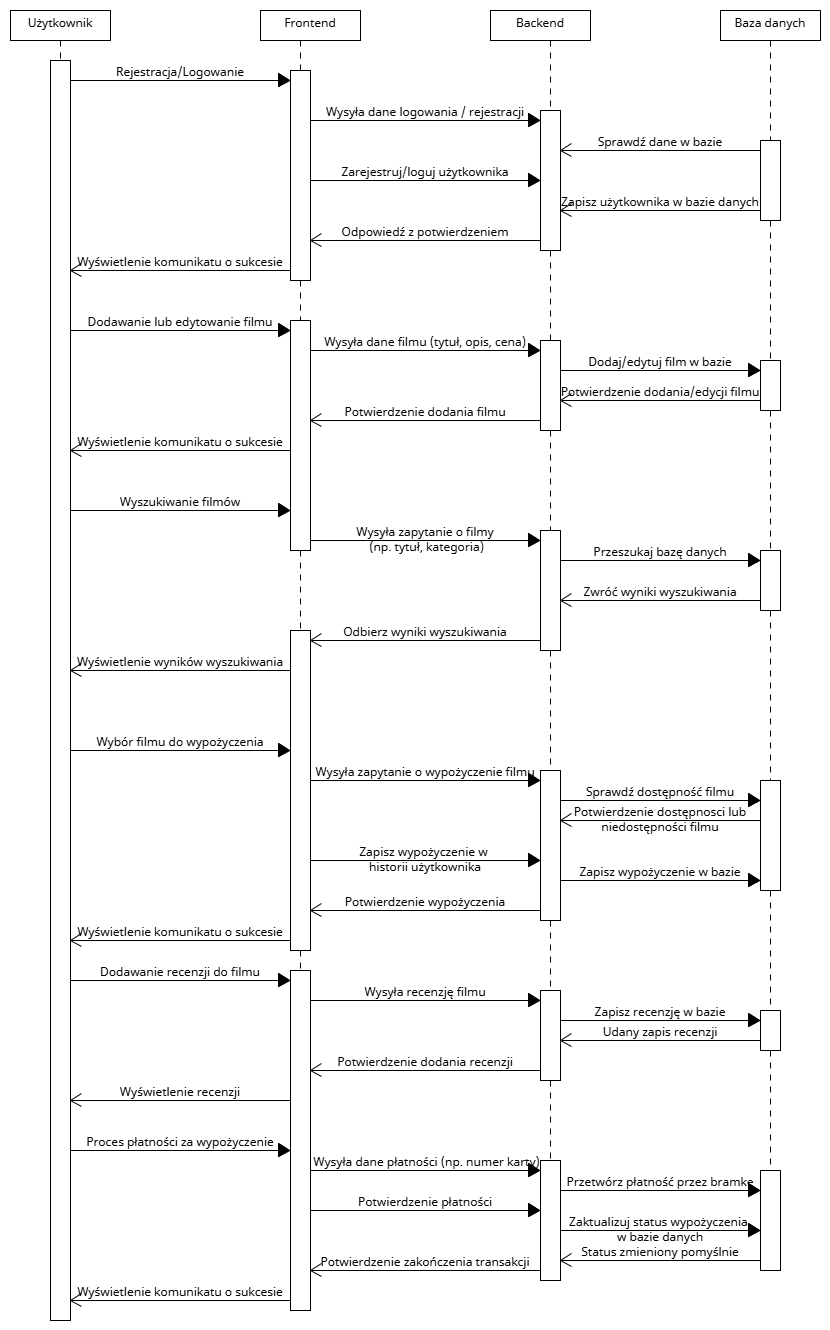
Rysunek 2 przedstawiający diagram czynności.

- Diagram klas:



Rysunek 3 przedstawiający diagram klas.

- Diagram sekwencji:



Rysunek 4 przedstawiający diagram sekwencji.

Diagramy zostały przygotowane w edytorze UMLetino i zapisane w formatach .uxf oraz .png oraz umieszczone na platformie GitHub.

# 4. Testy jednostkowe

Dla każdej funkcji aplikacji przygotowano co najmniej 4 testy jednostkowe z użyciem biblioteki pytest. Testowane funkcje obejmują dodawanie filmów, wyszukiwanie filmów, wypożyczanie filmów, obsługę recenzji oraz płatności.

**-Dodawanie filmów**

Kod źródłowy pliku test\_addmovie.py:

import pytest

def add\_movie(filmy, tytul, gatunek, rezyser):

film = {

'tytul': tytul,

'gatunek': gatunek,

'rezyser': rezyser

}

filmy.append(film)

return filmy

def test\_add\_movie\_valid():

filmy = []

wynik = add\_movie(filmy, 'Inception', 'Sci-Fi', 'Christopher Nolan')

assert len(wynik) == 1

assert wynik[0]['tytul'] == 'Inception'

def test\_add\_movie\_multiple():

filmy = [{'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}]

wynik = add\_movie(filmy, 'Interstellar', 'Sci-Fi', 'Christopher Nolan')

assert len(wynik) == 2

assert wynik[1]['tytul'] == 'Interstellar'

def test\_add\_movie\_invalid():

filmy = []

wynik = add\_movie(filmy, '', 'Drama', 'Christopher Nolan')

assert len(wynik) == 1

assert wynik[0]['tytul'] == ''

def test\_add\_movie\_empty\_list():

filmy = []

wynik = add\_movie(filmy, 'Titanic', 'Romance', 'James Cameron')

assert wynik == [{'tytul': 'Titanic', 'gatunek': 'Romance', 'rezyser': 'James Cameron'}]

if \_\_name\_\_ == "\_\_main\_\_":

pytest.main()

**-Wyszukiwanie filmów**

Kod źródłowy pliku test\_searchmovie.py:

import pytest

def search\_movie(filmy, tytul):

for film in filmy:

if film['tytul'].lower() == tytul.lower():

return film

return None

def test\_search\_movie\_found():

filmy = [{'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}]

wynik = search\_movie(filmy, 'Inception')

assert wynik is not None

assert wynik['tytul'] == 'Inception'

def test\_search\_movie\_not\_found():

filmy = [{'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}]

wynik = search\_movie(filmy, 'Titanic')

assert wynik is None

def test\_search\_movie\_case\_insensitive():

filmy = [{'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}]

wynik = search\_movie(filmy, 'inception')

assert wynik is not None

assert wynik['tytul'] == 'Inception'

def test\_search\_movie\_empty\_list():

filmy = []

wynik = search\_movie(filmy, 'Inception')

assert wynik is None

if \_\_name\_\_ == "\_\_main\_\_":

pytest.main()

- **Wypożyczanie filmów**

Kod źródłowy pliku test\_rentmovie.py:

import pytest

def rent\_movie(uzytkownik, film):

if 'wypozyczone\_filmy' not in uzytkownik:

uzytkownik['wypozyczone\_filmy'] = []

uzytkownik['wypozyczone\_filmy'].append(film)

return uzytkownik

def test\_rent\_movie\_new\_user():

uzytkownik = {}

film = {'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}

wynik = rent\_movie(uzytkownik, film)

assert 'wypozyczone\_filmy' in wynik

assert len(wynik['wypozyczone\_filmy']) == 1

assert wynik['wypozyczone\_filmy'][0]['tytul'] == 'Inception'

def test\_rent\_movie\_existing\_user():

uzytkownik = {'wypozyczone\_filmy': [{'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}]}

film = {'tytul': 'Interstellar', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}

wynik = rent\_movie(uzytkownik, film)

assert len(wynik['wypozyczone\_filmy']) == 2

assert wynik['wypozyczone\_filmy'][1]['tytul'] == 'Interstellar'

def test\_rent\_movie\_duplicate():

uzytkownik = {'wypozyczone\_filmy': [{'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}]}

film = {'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}

wynik = rent\_movie(uzytkownik, film)

assert len(wynik['wypozyczone\_filmy']) == 2

assert wynik['wypozyczone\_filmy'][1]['tytul'] == 'Inception'

def test\_rent\_movie\_multiple\_users():

uzytkownik\_1 = {'wypozyczone\_filmy': [{'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}]}

uzytkownik\_2 = {'wypozyczone\_filmy': []}

film = {'tytul': 'Interstellar', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}

wynik\_1 = rent\_movie(uzytkownik\_1, film)

wynik\_2 = rent\_movie(uzytkownik\_2, film)

assert len(wynik\_1['wypozyczone\_filmy']) == 2

assert len(wynik\_2['wypozyczone\_filmy']) == 1

assert wynik\_2['wypozyczone\_filmy'][0]['tytul'] == 'Interstellar'

if \_\_name\_\_ == "\_\_main\_\_":

pytest.main()

**- Obsługa recenzji**

Kod źródłowy pliku test\_review.py:

import pytest

def leave\_review(film, recenzja):

if 'recenzje' not in film:

film['recenzje'] = []

film['recenzje'].append(recenzja)

return film

def test\_leave\_review\_new\_review():

film = {'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}

wynik = leave\_review(film, 'Amazing movie!')

assert 'recenzje' in wynik

assert len(wynik['recenzje']) == 1

assert wynik['recenzje'][0] == 'Amazing movie!'

def test\_leave\_review\_multiple\_reviews():

film = {'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan', 'recenzje': ['Great!']}

wynik = leave\_review(film, 'Amazing movie!')

assert len(wynik['recenzje']) == 2

assert wynik['recenzje'][1] == 'Amazing movie!'

def test\_leave\_review\_duplicate\_review():

film = {'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan', 'recenzje': ['Great!']}

wynik = leave\_review(film, 'Great!')

assert len(wynik['recenzje']) == 2

assert wynik['recenzje'][1] == 'Great!'

def test\_leave\_review\_empty\_review():

film = {'tytul': 'Inception', 'gatunek': 'Sci-Fi', 'rezyser': 'Christopher Nolan'}

wynik = leave\_review(film, '')

assert len(wynik['recenzje']) == 1

assert wynik['recenzje'][0] == ''

if \_\_name\_\_ == "\_\_main\_\_":

pytest.main()

**- Proces płatności**

Kod źródłowy pliku test\_processpayment.py:

import pytest

def process\_payment(suma\_platnosci):

if suma\_platnosci <= 0:

raise ValueError("Kwota płatności musi być większa od zera.")

return True

def test\_process\_payment\_valid():

wynik = process\_payment(100)

assert wynik is True

def test\_process\_payment\_zero():

with pytest.raises(ValueError):

process\_payment(0)

def test\_process\_payment\_negative():

with pytest.raises(ValueError):

process\_payment(-50)

def test\_process\_payment\_large\_amount():

wynik = process\_payment(1000000)

assert wynik is True

if \_\_name\_\_ == "\_\_main\_\_":

pytest.main()

# 5. Podział obowiązków

Projekt realizowany był zespołowo. Podział obowiązków przedstawia się następująco:  
  
Konrad Borawski – funkcji programu wraz z testami do funkcji

Miłosz Dąbkowski – diagramy UML, sprawozdanie

Kamil Karwowski – diagramy UML

# 6. Repozytorium

Repozytorium projektu znajduje się na platformie GitHub, było ono aktualizowane po każdych zajęciach.

Link do repozytorium:

<https://github.com/Konrad-B1/Projekt-aplikacja-do-zarz-dzania-wypozyczalni-film-w-online.git>