1. Logowanie/Rejestracja użytkownika (UserController):

a. Logowanie użytkownika (POST /user/login)

1. Scenariusz: Sprawdzenie, czy użytkownik może się zalogować z poprawnymi danymi.

Dane wejściowe: login i hasło

Oczekiwany rezultat: Status: 200 OK

2. Scenariusz: Logowanie użytkownika z niepoprawnym hasłem

Dane wejściowe:

```
{
  "login": "testuser",
  "password": "wrongpassword"
}
```

Oczekiwany rezultat: Zwrócenie statusu 401 Unauthorized

3. Scenariusz: Logowanie użytkownika z nieistniejącym loginem

Dane wejściowe:

```
{
  "login": "nonexistentuser",
  "password": "passworrrd"
}
```

Oczekiwany rezultat: Status 404 (Not Found).

b. Rejestracja użytkownika (POST /user/register)

1. Sprawdzenie, czy nowy użytkownik może się zarejestrować, poprawne dane

Dane wejściowe:

```
{
  "login": "newuser3",
  "password": "newpassword124"
}
```

Oczekiwany rezultat: Status: 201 Created

2. Scenariusz: Rejestracja użytkownika z istniejącym loginem

Dane wejściowe:

{

"login": "existinguser",

"password": "password123"

}

Oczekiwany rezultat: Status: 409 Conflict

2. Zarządzanie tytułami (książkami) (TitlesController):

a. Pobieranie tytułów (GET /titles/)

1. Scenariusz : Sprawdzenie, czy użytkownik może pobrać listę dostępnych tytułów.

Dane wejściowe:

Parametr “userId”: 399

Oczekiwany rezultat: Status: 200 OK

2. Scenariusz: Pobranie listy tytułów dla nieistniejącego użytkownika

Dane wejściowe:

Parametr “userId”: 999999 (nieistniejący użytkownik)

Oczekiwany rezultat: Status: 404 (Not Found), brak listy tytułów.

b. Dodawanie tytułu (POST /titles/)

1. Scenariusz: Sprawdzenie, czy użytkownik może dodać nowy tytuł.

Dane wejściowe:

```
{
    "userId": 411,
    "id": 413,
    "title": "New Book Title",
    "author": "New Author",
    "year": 2020
}
```

Oczekiwany rezultat: Status: 201 Created

2. Scenariusz: Dodanie tytułu bez wymaganych danych (jednego z pól np. tytułu)

Dane wejściowe:

```
{
    "userId": 411,
    "id": 413,
    "author": "New Author",
    "year": 2020
}
```


Oczekiwany rezultat: Status 400 (Bad Request).

c. Aktualizacja istniejącego tytułu (PUT /titles/)

1. Scenariusz: Sprawdzenie czy użytkownik może edytować istniejący tytuł

Dane wejściowe: Dane wejściowe:


```
{
        "userId": 124,
        "id": 1,
        "title": "Updated Book Title",
        "author": "Updated Author",
        "year": 2021
}
```


Oczekiwany rezultat: Status 200 OK, zaktualizowany tytuł.

2. Scenariusz: Aktualizacja nieistniejącego tytułu

Dane wejściowe:


```
{
  "userId": 124,
  "id": 9999,
  "title": "Nonexistent Book",
  "author": "Unknown Author",
  "year": 2022
}
```


Oczekiwany rezultat: Status 404 (Not Found).

d. Usuwanie tytułu (DELETE /titles/)

1. Scenariusz: Sprawdzenie, czy użytkownik może pomyślnie usunąć tytuł.

Dane wejściowe:

Parametr id: 3

Parametr userId: 124

Oczekiwany rezultat: Status: 204 No Content

2. Scenariusz: Sprawdzenie, czy użytkownik może usunąć nieistniejący tytuł

Dane wejściowe: Parametry:

id: 9999 (nieistniejący tytuł)

userId: 124

Oczekiwany rezultat: Status HTTP 404 (Not Found).

3. Zarządzanie egzemplarzami książek (ItemsController)

a. Pobranie listy egzemplarzy (GET /items/)

1. Scenariusz: Pobranie listy egzemplarzy danego tytułu dla zalogowanego użytkownika

Dane wejściowe:

Parametr userId: 124

Parametr titleId: 1

Oczekiwany rezultat: Status: 200 OK

b. Dodanie nowego egzemplarza (POST /items/)

1. Scenariusz: Dodanie nowego egzemplarza książki do katalogu.

Dane wejściowe:

```
{
  „userId”: 124
  "titleId": 2,
  "purchaseDate": "2024-11-09",
}
```

Oczekiwany rezultat: Status: 201 Created

c. Aktualizacja egzemplarza (PUT /items/)

1. Scenariusz: Aktualizacja szczegółów istniejącego egzemplarza

Dane wejściowe:


```
{
  "userId": 124,
  "id": 10,
  "purchaseDate": "2024-11-08"
}
```


Oczekiwany rezultat: Status: 200 OK

d. Usunięcie egzemplarza (DELETE /items/)

1. Scenariusz: Pomyślne usunięcie egzemplarza książki

Dane wejściowe:

Parametr userId: 124

Parametr id: 12

Oczekiwany rezultat: Status: 204 No Content

4. Zarządzanie wypożyczeniami (RentsController)

a. Pobranie listy wypożyczeń (GET /rents/)

1. Scenariusz: Pobranie listy wypożyczeni danego egzemplarza dla użytkownika

Dane wejściowe:

Parametr userId: 124

Parametr itemId: 10

Oczekiwany rezultat: Status 200 OK

b. Dodanie nowego wypożyczenia (POST /rents/)

1. Scenariusz: Utworzenie nowego wypożyczenia dla egzemplarza książki

Dane wejściowe:


```
{
  "userId": 423,
  "itemId": 425,
  "customerName": "Elton John",
  "rentDate": "2024-11-01"
```

```
}
```

Oczekiwany rezultat: Status 201 Created

c. Aktualizacja wypożyczenia (PUT /rents/)

1. Scenariusz: Pomyślna aktualizacja wypożczenia

Dane wejściowe:

```
{
  "userId": 423,
  "id": 427,
  "customerName": "Elton John",
  "rentDate": "2024-10-31",
  "expirationDate": "2024-11-02"
}
```

Oczekiwany rezultat: Status 200 OK

d. Usunięcie wypożyczenia (DELETE /rents/)

1. Scenariusz: Pomyślne usunięcie wypożyczenia

Dane wejściowe:

Parametr userId: 124

Parametr id: 2

Oczekiwany rezultat: Status: 204 No Content