1. **Wstęp do dokumentacji**

Aplikacja do rekomendacji filmów ma na celu dostarczanie użytkownikom spersonalizowanych rekomendacji filmowych na podstawie ich preferencji i aktywności w serwisie. Umożliwia również przeglądanie oraz wyszukiwanie filmów, a zalogowani użytkownicy mogą dodawać swoje recenzje i komentować recenzje innych użytkowników.

Aplikacja zarządza bazą danych filmów i użytkowników, zapewniając pełne wsparcie dla rejestracji, logowania oraz autoryzacji. System śledzi aktywność użytkowników, aby dostarczać lepsze rekomendacje filmowe.

**Główne funkcje aplikacji:**

* Rekomendacja filmów na podstawie preferencji użytkowników.
* Przeglądanie i wyszukiwanie filmów w bazie danych.
* Rejestracja, logowanie i autoryzacja użytkowników.
* Możliwość dodawania recenzji filmów przez zalogowanych użytkowników.
* Komentowanie recenzji innych użytkowników.

**Technologie:**

* **Framework**: Django – obsługuje backend aplikacji, zarządza bazą danych, uwierzytelnianiem i widokami.
* **Baza danych**: PostgreSQL – system zarządzania relacyjną bazą danych, który przechowuje dane o filmach, użytkownikach, recenzjach i komentarzach.

1. **Architektura systemu**

A diagram of a computer

Description automatically generated

Aplikacja do rekomendacji filmów korzysta z wzorca architektonicznego **Model-View-Template (MVT)**, charakterystycznego dla frameworka Django. Wzorzec ten jest podobny do MVC, ale różni się głównie tym, że Django zarządza częścią kontrolera, a logika znajduje się w widokach (Views).

**2.1. Główne komponenty architektury MVT**

W architekturze MVT w Django, aplikacja jest podzielona na trzy główne warstwy:

* **Model**: Odpowiada za interakcję z bazą danych. Modele są klasami Pythona, które definiują strukturę tabel w bazie danych oraz ich relacje.
* **Widok (View)**: Przetwarza żądania użytkowników, pobiera dane z modeli i przekazuje je do szablonów w celu wyświetlenia odpowiednich treści.
* **Szablon (Template)**: Odpowiada za warstwę prezentacji. Generuje dynamiczne strony HTML na podstawie danych przekazanych przez widoki.

**2.2. Podział na aplikacje Django**

Projekt jest podzielony na kilka aplikacji, z których każda obsługuje konkretne funkcjonalności:

1. **movie\_app**:
   * Odpowiada za przeglądanie i wyszukiwanie filmów oraz generowanie rekomendacji filmowych dla użytkowników.
   * Definiuje modele filmów oraz logikę rekomendacyjną.
2. **user\_app**:
   * Zajmuje się rejestracją, logowaniem, zarządzaniem profilami użytkowników oraz autoryzacją.
   * Przechowuje dane użytkowników oraz zarządza sesjami.
3. **review\_app**:
   * Odpowiada za dodawanie recenzji do filmów oraz komentowanie recenzji innych użytkowników.
   * Zawiera modele recenzji i komentarzy oraz logikę zarządzającą tą funkcjonalnością.
4. **recommendation\_app:**
   * Odpowiada za tworzenie rekomendacji dla użytkowników
   * Zawiera model rekomendacji

**2.3. Baza danych**

Aplikacja korzysta z bazy danych **PostgreSQL** do przechowywania danych związanych z filmami, użytkownikami, recenzjami i komentarzami. Struktura bazy danych jest odzwierciedleniem modeli Django:

* **Tabela filmów**: Zawiera dane o filmach (np. tytuł, reżyser, gatunek, ocena).
* **Tabela użytkowników**: Przechowuje dane o użytkownikach, w tym informacje o ich kontach i ustawieniach.
* **Tabela recenzji**: Przechowuje recenzje filmów napisane przez użytkowników.
* **Tabela komentarzy**: Przechowuje komentarze do recenzji filmów.

**2.4. Rekomendacje filmów**

Celem algorytmu rekomendacji jest dobranie zbioru filmów, które prawdopodobnie przypadną do gustu wskazanemu użytkownikowi. Algorytm ten polega na rekomendacjach wstawianych przez innych użytkowników oraz na ogólnej zależności gatunków filmów (jeśli użytkownikowi podoba się film kryminalny, to prawdopodobnie spodobają mu się inne filmy kryminalne).

Algorytm rekomendacji na początku pobiera ostatnio wystawione przez użytkownika pozytywne recenzje. Pozytywna recenzja jest definiowana jako recenzja, której ocena jest wyższa niż wskazany próg - próg ten wybrano na ocenę 4. Po pobraniu pozytywnych recenzji użytkownika, wywołany zostaje algorytm wyszukiwania filmów podobnych dla każdego pozytywnie ocenionego filmu.

Algorytm wyszukiwania podobnych filmów składa się z dwóch podalgorytmów. Pierwszy z nich, zwany *rekomendacjami inteligentnymi*, sprawdza ostatnie pozytywne recenzje wybranego filmu. Następnie sprawdza on, jakie inne filmy zostały pozytywnie ocenione przez użytkowników, którzy te opinie wystawili, i zwraca odnalezione filmy. Druga część algorytmu, zwana *rekomendacjami gatunkowymi*, sprawdza gatunki otrzymanego filmu i zwraca najpopularniejsze filmy z tych gatunków. Popularność filmu jest definiowana pod kątem oceny filmu, gdzie filmy o ocenie wyższej niż 4 są uznawane jako popularne. Otrzymane w ten sposób rekomendacje filmów są sortowane tak, aby najpierw pojawiły się filmy z algorytmu inteligentnego, a dopiero następnie pojawiały się filmy z rekomendacji gatunkowych.

**2.5. Warstwa komunikacji**

* **Widoki Django (Views)**: Przetwarzają żądania HTTP, pobierają dane z modeli i generują odpowiedzi w postaci stron HTML. Widoki kontrolują logikę biznesową aplikacji, np. wyszukiwanie filmów, przetwarzanie recenzji czy rekomendacje.
* **Szablony (Templates)**: Szablony są odpowiedzialne za generowanie dynamicznych stron internetowych. Korzystają z języka szablonów Django, aby wyświetlać dane przekazywane przez widoki (np. listę filmów, recenzje, szczegóły użytkownika).

**2.6. Pliki konfiguracyjne**

* **settings.py**:
  + Zawiera konfigurację całego projektu, w tym ustawienia bazy danych, konfigurację aplikacji, ustawienia statycznych plików, autoryzację oraz inne kluczowe ustawienia aplikacji.

**2.7. Statyczne zasoby i szablony**

* **Pliki statyczne**: Zawierają zasoby takie jak CSS, JavaScript oraz obrazy, które są potrzebne do prawidłowego wyświetlania interfejsu użytkownika.
* **Szablony (HTML)**: Szablony HTML definiują, jak wyglądają strony internetowe generowane przez Django. Są zintegrowane z logiką w widokach i dynamicznie wyświetlają dane.

1. **Endpointy**
   * **Main**
     1. **GET / :** Zwraca stronę główną aplikacji.
     2. **GET /about/ :** Zwraca informację o aplikacji.
     3. **GET /admin :** Zwraca stronę administratora.
   * **Filmy** 
     1. **GET /movies/:** Zwraca listę wszystkich filmów.
     2. **GET /details/{id}/:** Zwraca szczegóły konkretnego filmu.
   * **Recenzje**
     1. **POST /reviews/add\_review/{movie\_id}/ :** Dodaje nową recenzje dla filmu.
     2. **POST /reviews/add\_comment/{review\_id}/ :** Dodaje komentarz do recenzji filmu.
   * **Użytkownicy**
     1. **GET /users/ :** Zwraca listę wszystkich użytkowników.
     2. **GET /users/details/{user\_id}/ :** Zwraca stronę użytkownika.
     3. **POST /users/login/ :** Logowanie użytkownika.
     4. **POST /users/logout/ :** Wylogowanie użytkownika.
     5. **POST /users/register:** Rejestracja nowego użytkownika.

**4. Instalacja i konfiguracja**

W tej sekcji opisano kroki instalacji oraz konfiguracji aplikacji do rekomendacji filmów. Instrukcje dotyczą zarówno uruchomienia aplikacji lokalnie, jak i konfiguracji środowiska produkcyjnego.

**4.1. Wymagania systemowe**

Aby uruchomić aplikację, wymagane są następujące komponenty:

* **Python**: Wersja 3.8 lub wyższa.
* **Django**: Wersja 3.x lub wyższa.
* **PostgreSQL**: Baza danych PostgreSQL do przechowywania danych filmów, recenzji, użytkowników itp.
* **Virtualenv** (opcjonalnie): Umożliwia uruchomienie aplikacji w izolowanym środowisku.

**4.2. Kroki instalacji lokalnej**

**1. Klonowanie repozytorium**

Pobierz kod źródłowy aplikacji, klonując repozytorium:

git clone https://github.com/ProjectManiacc/Movie\_Recommendation\_System.git  
cd Movie\_Recommendation\_System

**2. Tworzenie wirtualnego środowiska**

W celu izolowania zależności aplikacji, zaleca się utworzenie wirtualnego środowiska:

python -m venv venv  
venv\Scripts\activate

**3. Instalacja zależności**

Zainstaluj wszystkie wymagane pakiety na podstawie pliku requirements.txt:

pip install -r requirements.txt

**4. Konfiguracja bazy danych PostgreSQL**

Aplikacja korzysta z PostgreSQL jako głównej bazy danych. Skonfiguruj bazę danych:

* Upewnij się, że PostgreSQL jest zainstalowany i uruchomiony.
* Utwórz bazę danych dla aplikacji i wczytaj plik:
  + movie\_recommendation\_database.sql

**5. Konfiguracja pliku settings.py**

W pliku settings.py dostosuj ustawienia bazy danych:

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.postgresql\_psycopg2',  
 'NAME': 'movie\_recommendation\_db',  
 'USER': 'postgres',  
 'PASSWORD': 'postgres',  
 'HOST': 'localhost',  
 'PORT': '5432',  
 }  
}

**6. Migracja bazy danych**

Zastosuj migracje, aby utworzyć wszystkie potrzebne tabele w bazie danych:

python manage.py migrate

**7. Uruchomienie serwera deweloperskiego**

Uruchom serwer deweloperski, aby sprawdzić działanie aplikacji lokalnie:

python manage.py runserver

Aplikacja będzie dostępna pod adresem http://127.0.0.1:8000/.