Шифр «Fisher»

Наукова робота на конкурс за напрямом:
Інформаційні технології

на тему:
«Інформаційна технологія підбору навчальних відеоматеріалів»

ЗМІСТ

ВСТУП ..... …..3
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР МЕТОДІВ ДОСЛІДЖЕННЯ..... 4
1.1 Аналіз методів фільтрації даних в рекомендаційних інформаційних системах 4
1.2 Застосування методу колаборативної фільтрації для підбору навчальних відеоматеріалів 8
1.3 Алгоритм підбору навчальних відеоматеріалів 12
2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ПІДБОРУ НАВЧАЛЬНИХ ВІДЕОМАТЕРІАЛІВ 13
2.1 Структурно-функціональне моделювання інформаційної технології підбору навчальних відеоматеріалів 13
2.2 Моделювання роботи інформаційної системи підбору навчальних відеоматеріалів 17
2.3 Логічне проектування моделі бази даних..... 19
3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДБОРУ НАВЧАЛЬНИХ ВІДЕОМАТЕРІАЛІВ 20
ВИСНОВКИ..... 27
СПИСОК ЛІТЕРАТУРИ..... 28
ДОДАТОК А. ІНСТРУКЦІЯ КОРИСТУВАЧА..... 30
ДОДАТОК Б. ІНСТРУКЦІЯ АДМІНІСТРАТОРА..... . 35
ДОДАТОК В. ЛІСТИНГ ПРОГРАМНОГО КОДУ**ОШИБКА! ЗАКЛАДКА НЕ ОПРЕД**

ВСТУП

Актуальність роботи. В мережі Інтернет навчальні відеоматеріали пропонується потоковими онлайн сервісами, навчальними платформами та файловими хостингами. Найбільшим ресурсом для публікації відео в мережі Інтернет є YouTube. Кількість відеоматеріалів, розміщених на даному ресурсі постійно зростає, тому з'являється потреба в якісній фільтрації навчального контенту під час пошуку відео за певною тематикою. Користувачі витрачають багато часу на ручний пошук та фільтрацію цього контенту для задоволення своїх потреб та критеріїв підбору. Як результат з'являється необхідність у створенні інформаційної системи, яка б містила в собі рекомендаційну логіку, що значно полегшить процес пошуку та фільтрації відео-контенту.

Мета роботи: розробити інформаційну технологію для підбору навчальних відеоматеріалів.

Об'єкт дослідження – процес знаходження рекомендованих навчальних відеоматеріалів.

Предмет дослідження – інформаційна технологія підбору навчальних відеоматеріалів.

Теоретичною основою дослідження є методи фільтрації даних, методологія системного аналізу, методи функціонального моделювання інформаційних систем, методологія проектування баз даних.

Наукова новизна роботи полягає в розробці інформаційної технології підбору навчальних відеоматеріалів, за рахунок чого удосконалено процес пошуку навчальних відеоматеріалів.

Практичне значення роботи полягає у створенні веб-додатку, в якому реалізовано рекомендаційну логіку підбору навчальних відеоматеріалів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР МЕТОДІВ ДОСЛІДЖЕННЯ

1.1 Аналіз методів фільтрації даних в рекомендаційних інформаційних системах

Рекомендаційні системи є одним із найбільш популярних додатків інтелектуального аналізу даних і машинного навчання в сфері інтернет-бізнесу. Вони аналізують поведінку користувачів інтернет-сервісу, після чого дають кількісну та якісну оцінку вподобання користувачем того чи іншого об’єкту. Об’єктами рекомендацій можуть бути товари в інтернет-магазині, перелік розділів веб-сайту, медіа-контент або інші споживачі веб-сервісу. Рекомендаційні системи віднайшли широке застосування у таких сферах як електронна комерція, соціальні мережі, веб-додатки тощо, де акцент робиться на користувача даних [**Ошибка! Источник ссылки не найден.**].

Логіка рекомендаційних систем дозволяє виділяти переваги кінцевих користувачів і повертати результати, які будуть цікавими для нього, ґрунтуючись на оцінках інших користувачів і припущеннях самої системи. На відміну від пошукових систем, для того щоб отримати відповідь, рекомендаційна система не вимагає конкретного запиту. Користувачеві пропонується виставити оцінку деяким об’єктам з колекції і на підставі даних оцінок та їх порівняння з оцінками інших користувачів система робить припущення та обчислює можливі рекомендації.

У кожній рекомендаційної системі ми маємо справу з користувачем, якому надається безліч альтернатив, серед яких йому необхідно здійснити вибір. Користувачу може не вистачати досвіду і знань для того, щоб самостійно відфільтрувати варіанти, які не відповідають його потребам. Користувач в певній формі, явній або неявній, надає системі інформацію про свої уподобання

та інтереси, при цьому деякі з них він може навіть не усвідомлювати. Таким чином, рекомендаційна система зображується як система, яка використовує певний тип фільтрації і існуючі відомості про потреби користувачів, для подальшої рекомендації набору альтернатив, які система вважає актуальними для них.

Для розробки рекомендаційної системи використовують 4 типи фільтрації:

- заснована на контенті;
- колаборативна;
- заснована на знаннях;
- гібридна.

Контентна фільтрація формує рекомендації на основі поведінки користувачів. Наприклад, цей підхід може використовувати дані про перегляди статей та блогів (які теми читає користувач та характеристики таких блогів). Якщо деякий користувач читає статті про мову Python або регулярно коментує пости в блогах з тривимірного моделювання, то тематична фільтрація може використовувати цю інформацію для виявлення подібного контенту і пропонувати схожий в якості рекомендованого для цього користувача (статті в блогах про мову Python або тривимірне моделювання). При контентній фільтрації створюються профілі для користувачів та об’єктів в системі. Користувацькі профілі можуть містити певну інформацію або відповіді на деякий набір питань. Щодо профілів об’єктів, вони можуть містити імена акторів та виконавців, назви жанрів, тощо, або ж іншу інформацію в залежності від типу об’єкта. Такий підхід (контентна фільтрація) був використаний у проекті Music Genome Project [2], так музичний аналітик оцінював кожну композицію за безліччю музичних параметрів, які потім можна було використати для виявлення музичних уподобань окремих користувачів [2].

Колаборативна фільтрація видає рекомендації, основані на моделі поведінки попереднього користувача. Маючи у наявності користувацькі

акаунти, система може вести збір інформації про шаблони поведінки авторизованих користувачів протягом багатьох років. Цей підхід дозволяє в подальшому рекомендувати користувачам досить точні матеріали, товари або інші об’єкти, в залежності від тематики веб-ресурсу [3]. По суті, рекомендації базуються на автоматичній співпраці безлічі користувачів і на виділення (методом фільтрації) тих користувачів, які демонструють схожі переваги або шаблони поведінки. виступає в якості повноцінного джерела інформаційного наповнення веб-ресурсу.

Як приклад, коли створюється веб-сайт, який пропонує його відвідувачам рекомендації щодо блогів, збирається інформації від великої кількості користувачів, які підписуються та читають блоги. Після чого можна згрупувати таких користувачів по їх інтересам та уподобанням. Наприклад, можна об’єднати в єдину групу тих користувачів, котрі читають блоги на одну й ту саму тематику. За цією інформацією визначаються найпопулярніші блоги серед тих, які читають учасники цієї групи. Після цього можна порекомендувати найпопулярніший блог конкретному користувачеві цієї групи серед тих, на які він ще не встиг підписатися або які він не читає [4].

Колаборативна фільтрація широко використовується в комерційних сервісах і соціальних мережах. Перший сценарій використання – це створення рекомендації щодо цікавої і популярної інформації на основі врахування «голосів» спільноти. Такі сервіси, як Reddit [5] (розважальний новинний онлайновий сервіс) або Digg [6] (соціальний новинний веб-портал) – це типові приклади систем, що використовують алгоритми колаборативної фільтрації.

Щодо рекомендаційних систем, заснованих на знаннях, загалом це системи, в яких для отримання рекомендацій використовуються отримані будь-яким чином знання. Найчастіше такі знання додаються вручну адміністратором веб-сайту або ж власне користувачами. Наприклад, користувачі онлайн-магазину можуть вказувати товари інші схожі товари. На основі цих даних і створюються рекомендації [7]. Прикладами таких систем є різні інтернет магазини Amazon [8], eBay [9] та онлайн-сервіси пошуку товарів.

Гібридні методи фільтрації поєднують підходи колаборативної та контентної фільтрацій. Такі підходи підвищують ефективність (і складність) рекомендаційних систем. Об’єднання результатів колаборативної і контентної фільтрацій потенційно дозволяє підвищити точність рекомендації. Крім того, гібридний підхід може бути корисний, якщо застосування колаборативної фільтрації починається при значній розрідженості даних. Гібридний підхід дозволяє спочатку зважувати результати згідно контентної фільтрації, а потім зміщувати ці ваги у напрямку колаборативної фільтрації (в міру "визрівання" доступного набору даних по конкретному користувачеві) [4].

Відомим прикладом гібридної рекомендаційної системи є Netflix [10], яка поєднує в собі 27 алгоритмів рекомендації.

Для колаборативної фільтрації існує дуже багато об’єктів і користувачів, але досить мало рейтингів, так як користувачі оцінюють не всі об’єкти (матриця рейтингів). Система повинна обчислити інші елементи матриці. Також для нового користувача, у якого ще немає оцінених об’єктів або їх занадто мало для аналізу, складно що-небудь рекомендувати. Те ж саме стосується і об’єктів. Цю проблему по-іншому називають «холодним стартом». Пасивна поведінка користувачів може привести до простою системи. Колаборативна фільтрація не вимагає конкретизованих запитів. Замість цього вона надає оцінки, які спрощують взаємодію користувачів з системою за рахунок простих алгоритмів.

Контентна фільтрація розглядає тільки однотипні об’єкти, що підходить для деяких сайтів, але іноді це невигідно і може привести до зворотного ефекту. Наприклад, користувачеві можна порекомендувати вакансії з однаковими професіями, але різних компаній. Якщо користувач купив смартфон і система почне рекомендувати інші смартфони, то це може відлякати клієнта. Також, для кожного об’єкта потрібно вручну прописувати велику кількість характеристик. Контентна фільтрація гарна тим, що вона вирішує проблеми колаборативної фільтрації і спирається на важливість слова в контексті документа.

Рекомендаційні системи засновані на знаннях важко реалізувати, тому що необхідно з’єднати різні групи об’єктів, щоб система не повторювала помилок

контентної фільтрації. В свою чергу, така система вирішує проблему контентної фільтрації. Вона пропонує різні, взаємопов’язані об’єкти.

Гібридна рекомендаційна система складна тим, що в неї вкладено різні алгоритми з різних методів. Гібридний метод містить в собі багато алгоритмів з різних методів, що надає йому високу точність кінцевих результатів

Для реалізації рекомендаційної інформаційної системі в даній роботі, за основу був взятий метод колаборативної фільтрації. Оскільки функціонал сайту передбачає аудит та збереження рейтингових оцінок відео, які виставляються користувачами сайту, щоб потім формувати набори даних, метод колаборативної фільтрацій типу User-based є найбільш зручним в даному випадку.

1.2 Застосування методу колаборативної фільтрації для підбору навчальних відеоматеріалів

В ході аналізу предметної області, було виконано огляд та аналіз існуючих рекомендаційних методів та обрано метод колаборативної фільтрації, реалізований через сингулярний розклад матриці уподобань відеоматеріалів користувачами системи.

Інформаційна область для систем колаборативної фільтрації складається з користувачів, які виразили переваги для різних відео [11]. Оцінку (перевагу) часто представляють у вигляді триплетів (користувач, відео, оцінка). Часто рекомендаційні системи використовують оціночну шкалу, таку як 0-5 зірок. Множина всіх триплетів оцінок формує розріджену матрицю, яка називається матрицею оцінок [11]. Пари (користувач, відео), в яких користувачі не виставили оцінку відео, є невідомими значеннями цієї матриці, як зображено в табл. 1.1.

Таблиця 1.1 – Приклад розрідженої матриці оцінок відео

	Відео №1	Відео №2	Відео №3
Користувач №1	4	?	2
Користувач №2	1	2	?
Користувач №3	?	5	4

При використанні системи колаборативної фільтрації необхідно вирішити два завдання:

1) спрогнозувати оцінку або перевагу, яке користувач виставить відео; метою прогнозу є заповнення в матриці оцінок відсутніх значень;

2) надати рекомендації, тобто формування ранжованого списку N-елементів для даного користувача.

Сингулярний розклад матриці **X** (матриця рейтингових оцінок користувачів) розміру $m \times n$, яка складена з дійсних або комплексних чисел, буде розкладанням на множники у вигляді USV^T, як приведено у формулі 1.1, де U (матриця «властивостей» користувачів) — матриця розміру $n \times r$ буде дійсною або комплексною унітарною матрицею, S буде $r \times r$ прямокутною діагональною матрицею, і V^T буде дійсною або комплексною унітарною матрицею розміру $m \times r$.

Уніфікований вигляд кожної с матриць приведено у формулі 1.2.

$$X_{[n\ x\ m]} = U_{[n\ x\ r]} S_{[r\ x\ r]} (V_{[m\ x\ r]})^T \tag{1.1}$$

$$\begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mr} \end{pmatrix} \begin{pmatrix} s_{11} & \cdots & s_{1r} \\ \vdots & \ddots & \vdots \\ s_{r1} & \cdots & s_{rr} \end{pmatrix} \begin{pmatrix} v_{11} & \cdots & v_{1r} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{mr} \end{pmatrix} \tag{1.2}$$

Метод сингулярного розкладу матриці був використаний для прогнозування рейтингових оцінок для тих відео, яким користувач не виставив

оцінку, після чого відео з найвищими прогнозованими оцінками пропонуються користувачеві.

Приклад розкладу матриці приведено у формулах 1.3, 1.4., де рядки матриці визначають користувачів системи, стовбці матриці визначають відеоматеріали, а числові значення – оцінку користувача виставлену певному відео.

$$A = \begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{matrix} \quad (1.3)$$

Після виконання сингулярного розкладу матриці A, отримуємо наступні матриці U, S, V^T, як приведено у формулах 1.4-1.6:

$$U = \begin{matrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{matrix} \quad (1.4)$$

$$S = \begin{matrix} 12.4 & 0 \\ 0 & 9.5 \end{matrix} \quad (1.5)$$

$$V^T = \begin{matrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & -0.12 & -0.69 & -0.69 \end{matrix} \quad (1.6)$$

Результатом множення матриць U, S, V^T є матриця B, приведена у формулі 1.7:

$$B = \begin{matrix} 0.92 & 0.95 & 0.92 & 0.01 & 0.01 \\ 2.91 & 3.01 & 2.91 & -0.01 & -0.01 \\ 3.90 & 4.04 & 3.90 & 0.01 & 0.01 \\ 4.82 & 5.00 & 4.82 & 0.03 & 0.03 \\ 0.70 & 0.53 & 0.70 & 4.11 & 4.11 \\ -0.69 & 1.34 & -0.69 & 4.78 & 4.78 \\ 0.32 & 0.23 & 0.32 & 2.01 & 2.01 \end{matrix} \quad (1.7)$$

Порівнюючи матриці A та B, можна сказати, що різниця між їх елементами дуже мала, іншими словами, результат множення остаточної трьох матриць B (після SVD) майже однаковий з початковою матрицею A (до SVD).

В рамках даної роботи було використано реалізацію методу сингулярного розкладу матриці за допомогою функції svds() бібліотеки SciPy [12]. На вхід svds() надходить розріджена матриця рейтингових оцінок відео, зроблених користувачами системи. Під розрідженістю матриці мається на увазі той факт, що серед великої кількості відеоматеріалів кожен користувач виставив рейтинг (оцінку) лише невеликій кількості відео, тому більшість значень в матриці є порожніми. Після обробки вхідних даних, функція svds() повертає матрицю з прогнозованими апроксимованими оцінками по кожному відео, заповнюючи порожні місця в матриці. Після цього, маючи дані прогнозованих оцінок, залежно від користувача (авторизованого на сайті), який послав запит на обчислення рекомендацій, з результуючого набору формується сортована вибірка ідентифікаторів відео (від найкращих до найгірших оцінок) для даного користувача, підготовлюється список з URL-адресами цих відео і вони відображаються на сторінці рекомендацій на веб-інтерфейсі.

1.3 Алгоритм підбору навчальних відеоматеріалів

Підбір навчальних відеоматеріалів здійснюється за наступним алгоритмом.

1. Користувачі виставляють оцінку (1-5) переглянутому відео.
2. З виставлених оцінок формується набір даних, який подається на вхід рекомендаційному алгоритму.
3. На виході з алгоритму повертається матриці прогнозованих оцінок
4. Виконується вибірка пар відео-оцінка для конкретного користувача, який зробив запит на знаходження рекомендованих відео.
5. Вибірка з п.4 сортується у порядку зменшення прогнозованої рейтингової оцінки, після чого система відображає 10 відео з найкращими оцінками

Запропонований алгоритм використовує метод колаборативної фільтрації, оскільки для знаходження невідомих переваг користувача використовуються відомі переваги, тобто оцінки, інших користувачів в системі.

Для якісною роботи алгоритму, система повинна мати у наявності набір даних у вигляді «користувач» – «оцінка» для більшості користувачів в системі

Даний набір формується динамічно у процесі перегляду користувачем певного відео та виставлення оцінки.

Після формування набору даних, система має готові дані, які подаються на вхід рекомендаційному алгоритму.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ПІДБОРУ НАВЧАЛЬНИХ ВІДЕОМАТЕРІАЛІВ

2.1 Структурно-функціональне моделювання інформаційної технології підбору навчальних відеоматеріалів

Для опису роботи системи та візуалізації логічних відношень між роботами, які відбуваються всередині системи, було створено функціональну модель за допомогою методології IDEF0.

На рис. 2.1 зображена контекстна діаграма «Підбір навчальних відеоматеріалів», яка представляє собою найвищий рівень абстракції для даного завдання. Створення діаграми відбувалося з точки зору архітектора системи з метою аналізу процесу вибору рекомендованих відеоматеріалів та подальшого вдосконалення інформаційної системи.

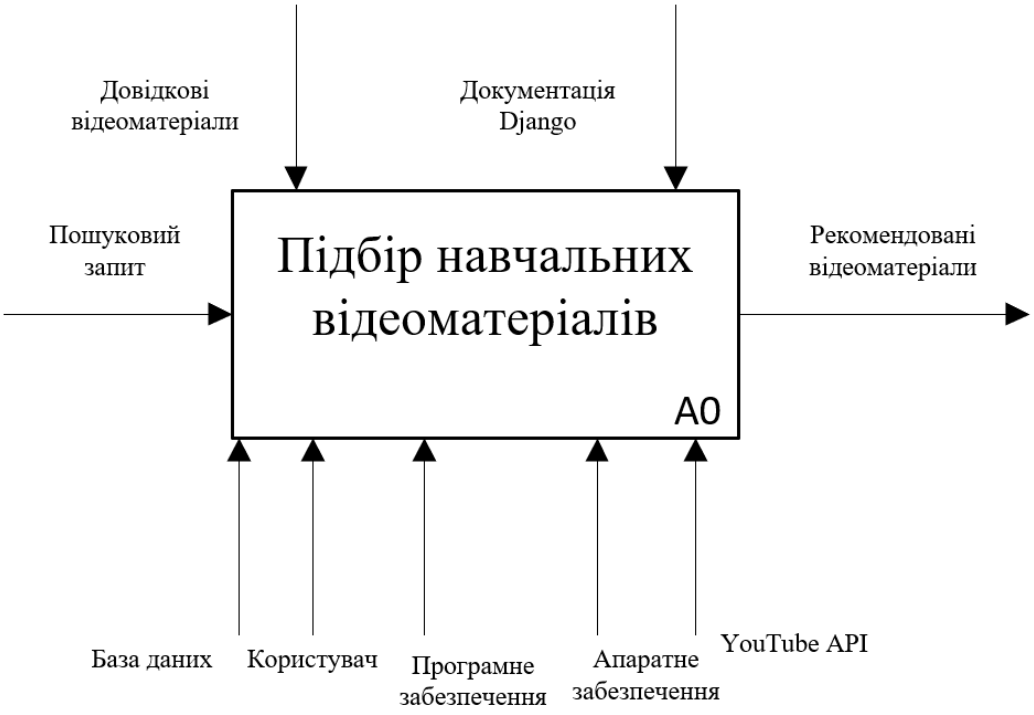

Рисунок 2.1 – Контекстна діаграма в нотації IDEF0

Після того, як система була описана в цілому, вона розбивається на великі фрагменти. Цей процес називається функціональної декомпозицією, а діаграми, які описують кожен фрагмент та взаємодію цих фрагментів між собою, називаються діаграмами декомпозиції.

Діаграма декомпозиції контекстного представлення моделі, як набір основних робіт системи, зображена на рис. 2.2. На вхід системи подається пошуковий запит користувача, відеоматеріали на шукану тематику витягуються із сервісу YouTube, після чого оцінюються користувачем системи. Отримані рейтингові оцінки зберігаються до бази даних, після чого формується та оброблюється набір даних з оцінками усіх користувачів системи та обчислюється набір рекомендованих відео для користувача, який зробив запит на відображення персональних рекомендацій.

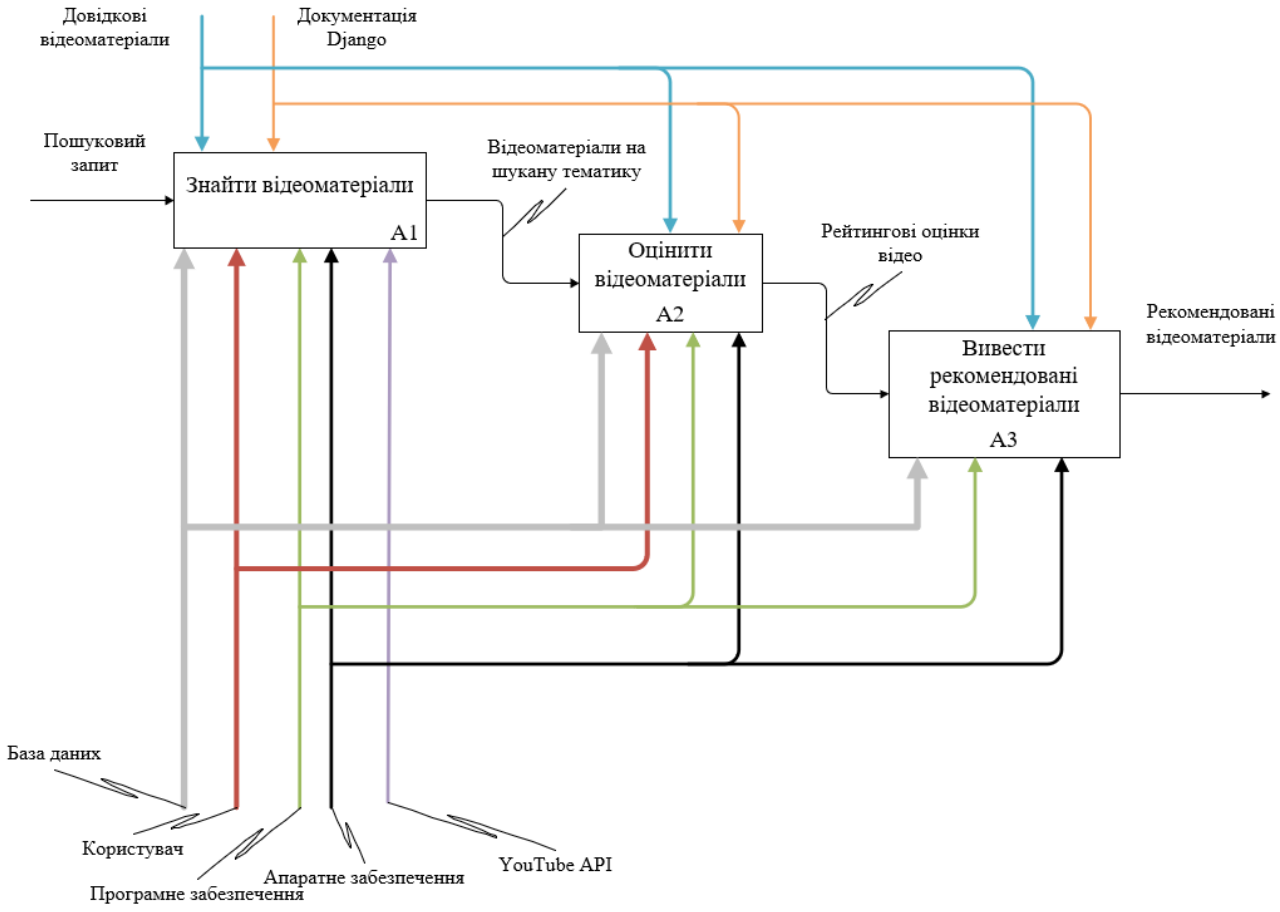

Рисунок 2.2 – Діаграма декомпозиції у нотації IDEF0

Для більш детального представлення складу кожної із основних робіт, виконуваних в рамках інформаційної системи, було виконано їх декомпозицію, приклад якої приведено на рис.2.3-2.5.

На рис. 2.3 приведена декомпозиція процесу «Знайти відеоматеріали». В рамках даного процесу виконується обробка пошукового запиту користувача, надсилається API запит search().list() до сервісу YouTube, відповідь на який повертається у вигляді списку ідентифікаторів відеоматеріалів, які відповідають шуканій тематиці, після чого відбувається рендеринг цих відео на сторінці додатку.

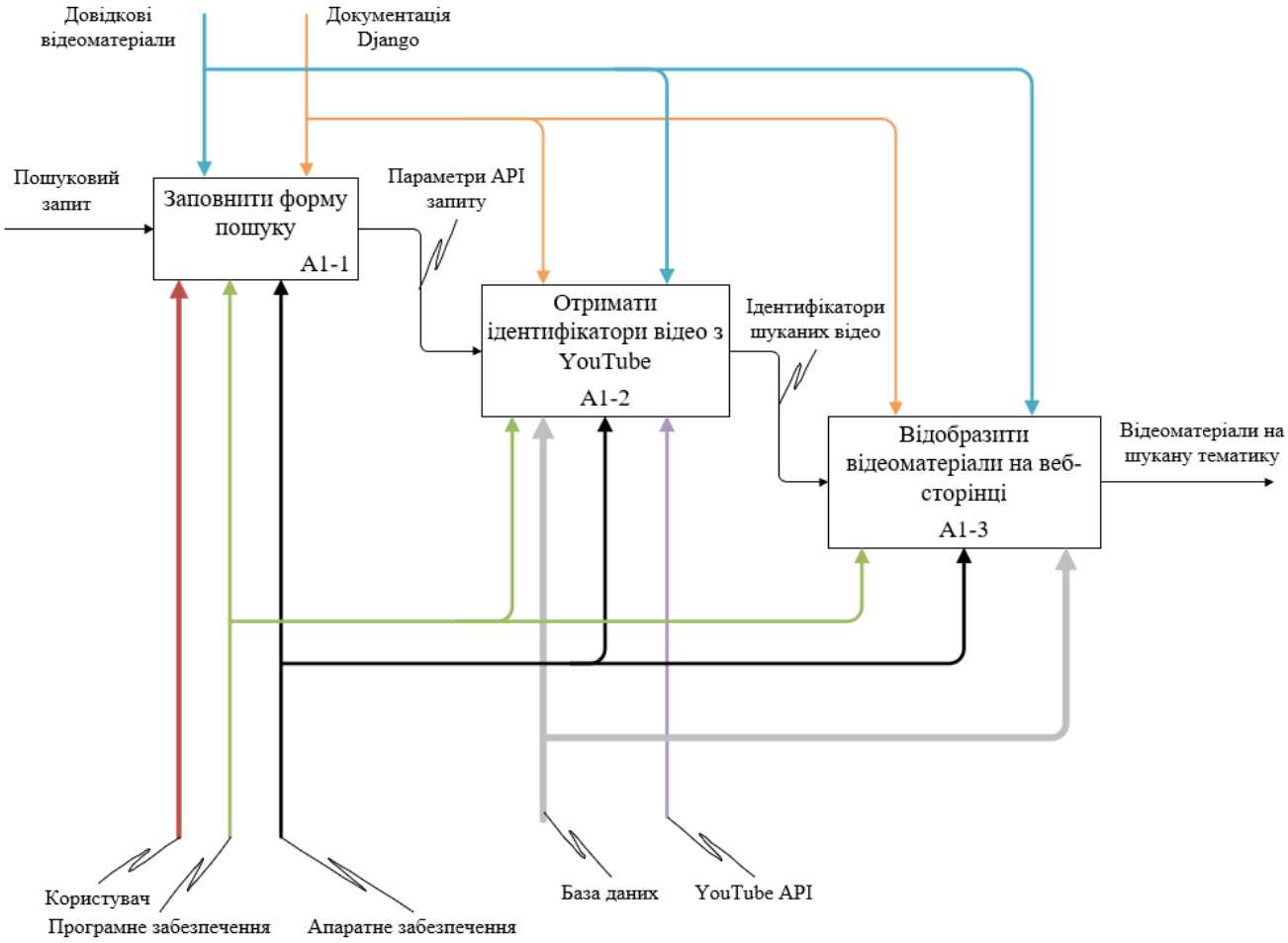

Рисунок 2.3 – Декомпозиція процесу «Знайти відеоматеріали»

На рис.2.4 зображена декомпозиція процесу «Оцінити відеоматеріали».

Користувач проставляє рейтингову оцінку відео, після чого вона заноситься до бази даних MySQL. Потім система виконує оновлення рейтингу відео та відображає зміни в статистиці рейтингу на сторінці додатку.

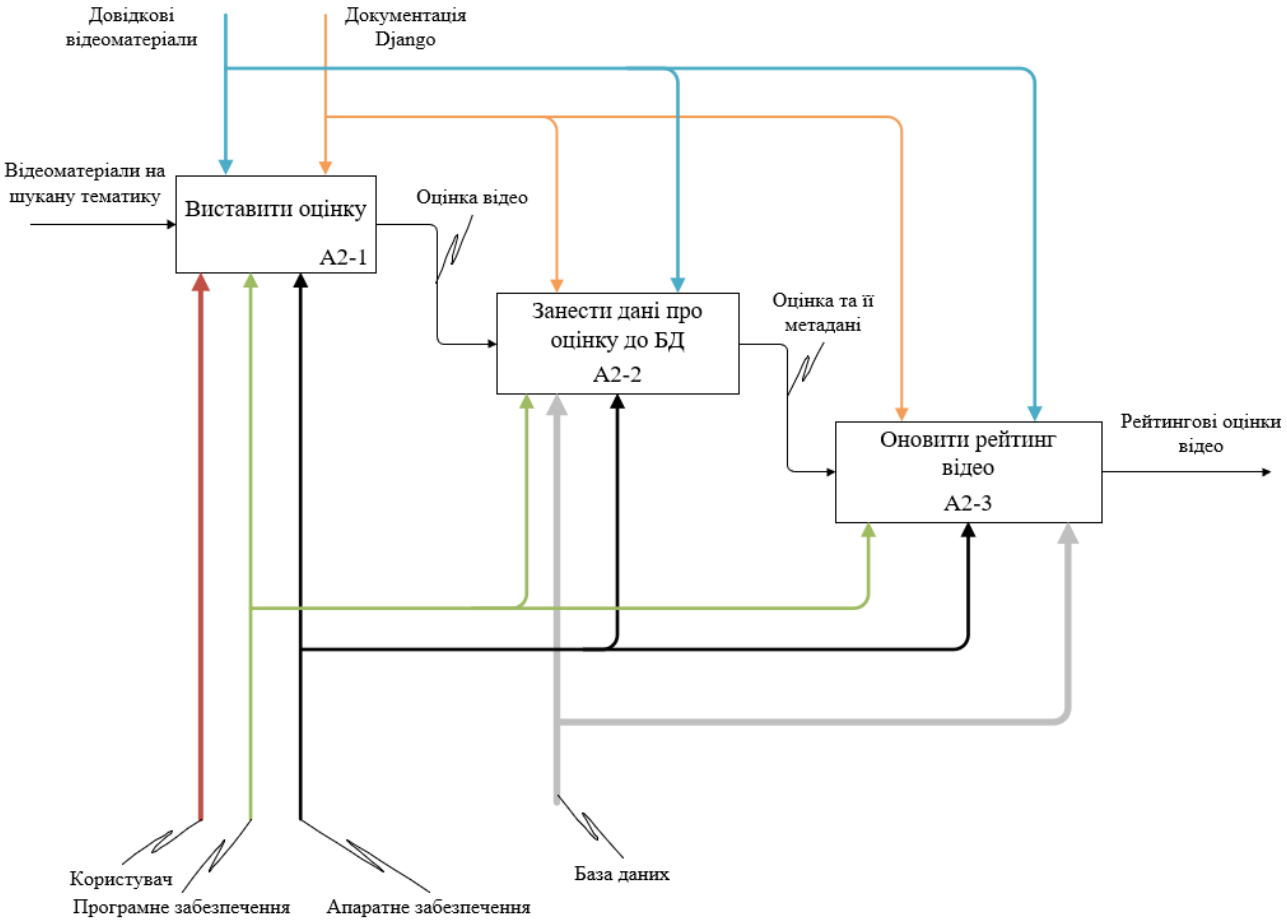

Рисунок 2.4 – Декомпозиція процесу «Оцінити відеоматеріали»

На рис. 2.5 приведена декомпозиція процесу «Вивести рекомендовані відеоматеріали». Базуючись на присутніх в базі даних оцінках відео, система формує набір даних у вигляді прямокутної матриці, де рядками виступають користувачі системи, а стовбцями є відповідні відео. Пересіченням рядка та стовпця є рейтингова оцінка, яку конкретний користувач виставив конкретному відео. Сформований набір даних передається на вхід рекомендаційному методу, який реалізований шляхом сингулярного розкладу матриці. На виході алгоритму формується матриця прогнозованих оцінок для конкретного користувача (того, який зробив запит на пошук рекомендацій), виконується обробка даної матриці, після чого рекомендовані до перегляду відео

відображаються на сторінці.

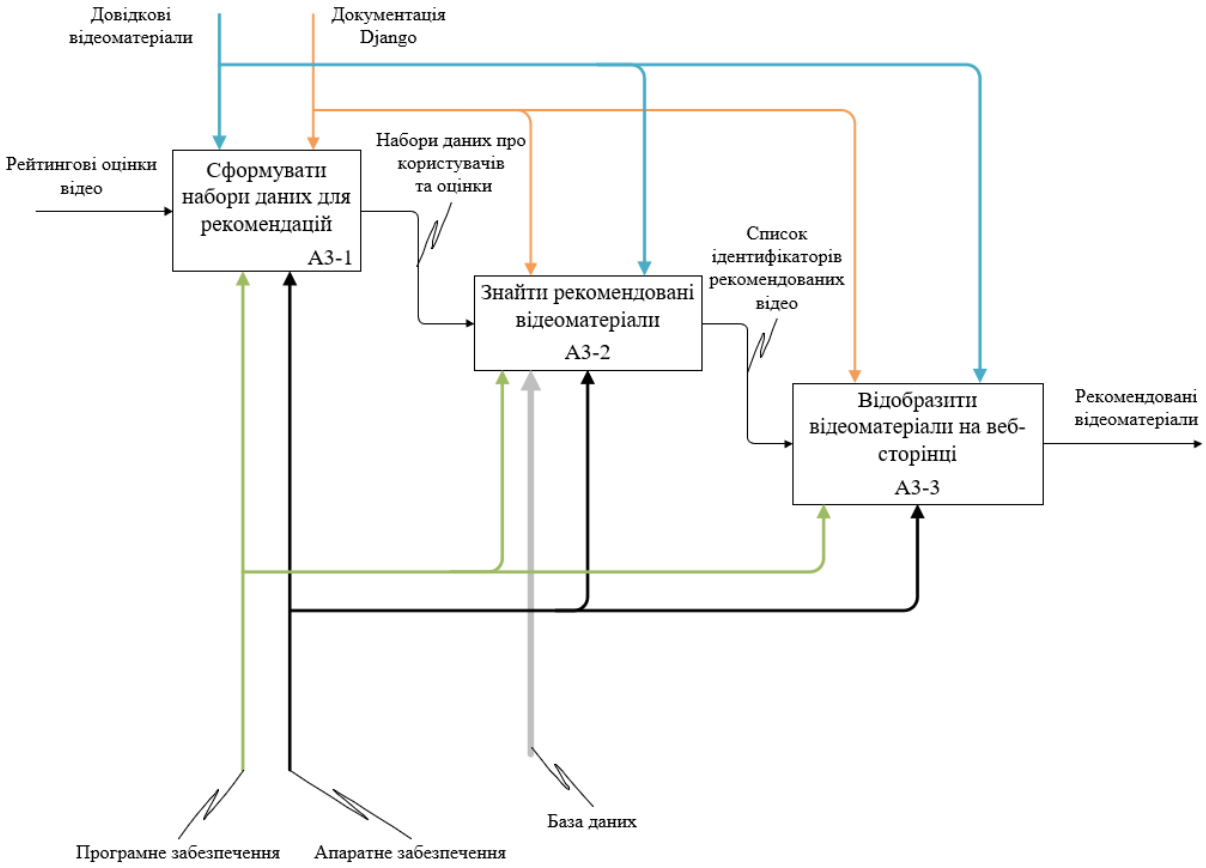

Рисунок 2.5 – Декомпозиція процесу «Вивести рекомендовані відеоматеріали»

2.2 Моделювання роботи інформаційної системи підбору навчальних відеоматеріалів

Діаграми потоків даних (DFD) представляють ієрархію функціональних процесів, пов'язаних між собою потоками даних. Мета такого уявлення – продемонструвати, як кожен процес перетворює свої вхідні дані у вихідні, а також визначити відносини між цими процесами.

Побудова DFD-моделі базується на принципі декомпозиції. На рис. 2.6 зображена контекстна діаграма «Підбір навчальних відеоматеріалів».

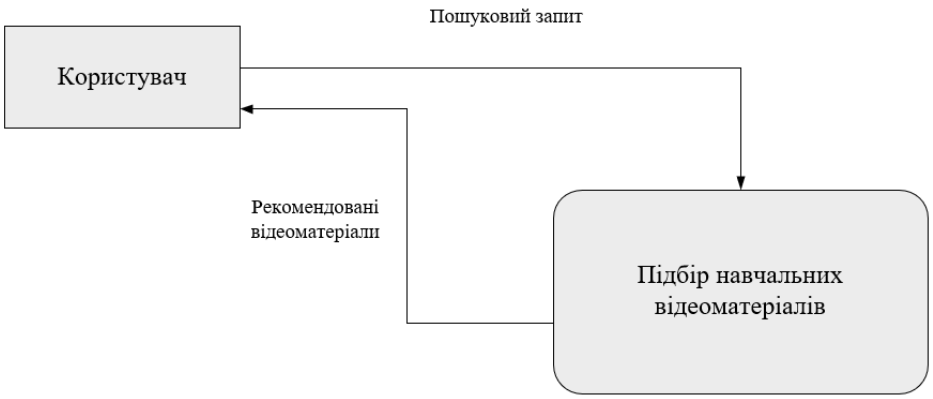

Рисунок 2.6 – Контекстна діаграма в нотації DFD

Декомпозиція підсистеми «Підбір навчальних відеоматеріалів», представленої на контекстній діаграмі, приведена на рис.2.7.

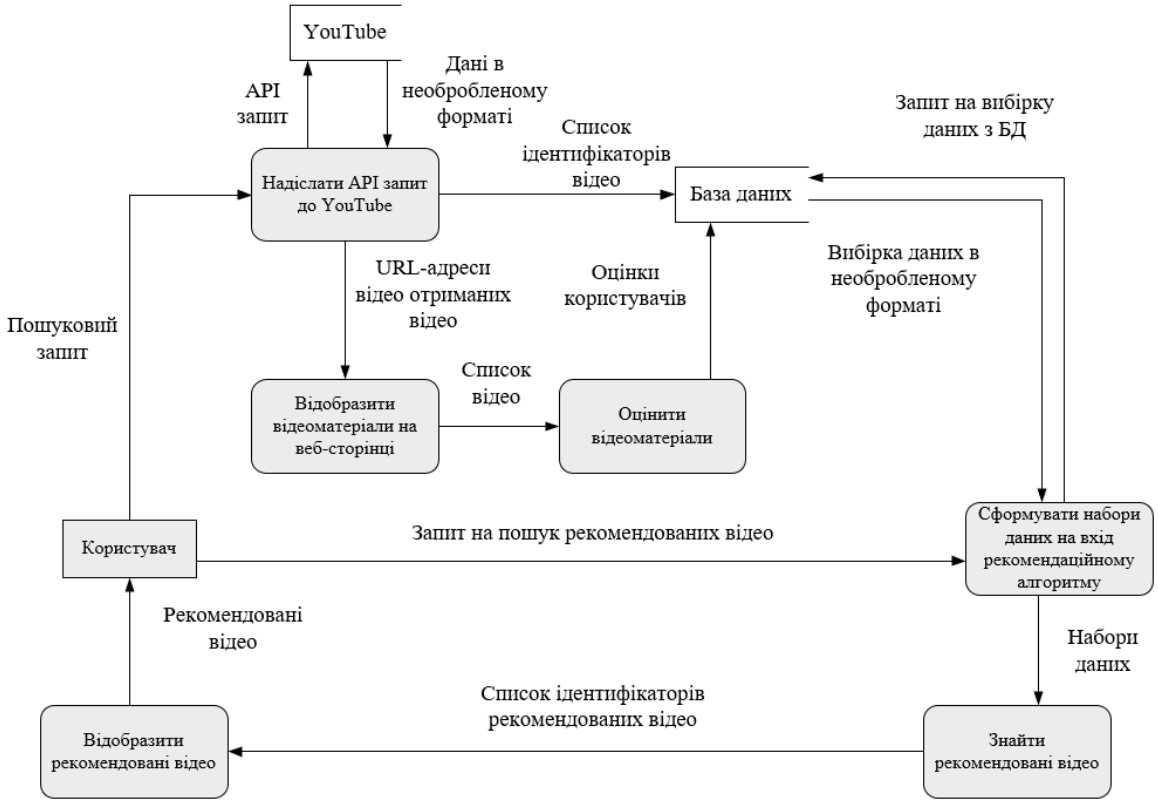

Рисунок 2.7 – Діаграма декомпозиції у нотації DFD

На рис.2.7 представлені взаємовідносини між процесами, які виконується в системі та шлях обробки вхідних даних у вихідні, починаючи з пошукового запиту користувача і закінчуючи списком відео рекомендацій.

2.3 Логічне проектування моделі бази даних

Логічне проектування [13] − це розроблення структур зберігання, методів доступу й логічної структури системи баз даних без прив'язки до конкретної системи управління базами даних (СКБД).

Для збереження даних було створено наступні таблиці:

- star_ratings_rating – інформація про оцінку відео об'єкту
- star_ratings_userrating – інформація про користувача, який зробив оцінку відео об'єкту
- auth_user – інформація про користувачів системи
- videorecapp_video – інформація про відеоматеріали

На рис.3.8 приведена логічна модель бази даних.

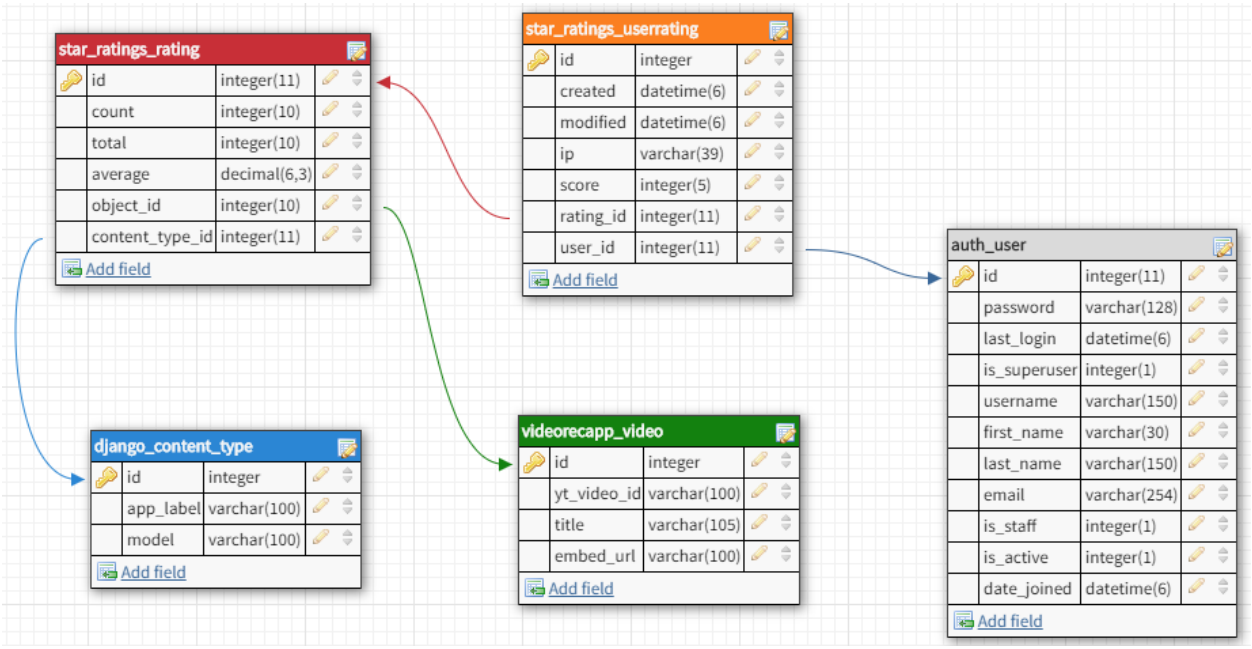

Рисунок 3.8 – Логічна модель бази даних

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДБОРУ НАВЧАЛЬНИХ ВІДЕОМАТЕРІАЛІВ

На рис.3.1 представлена архітектура створеного веб-додатку у вигляді UML діаграми компонентів [14, 15].

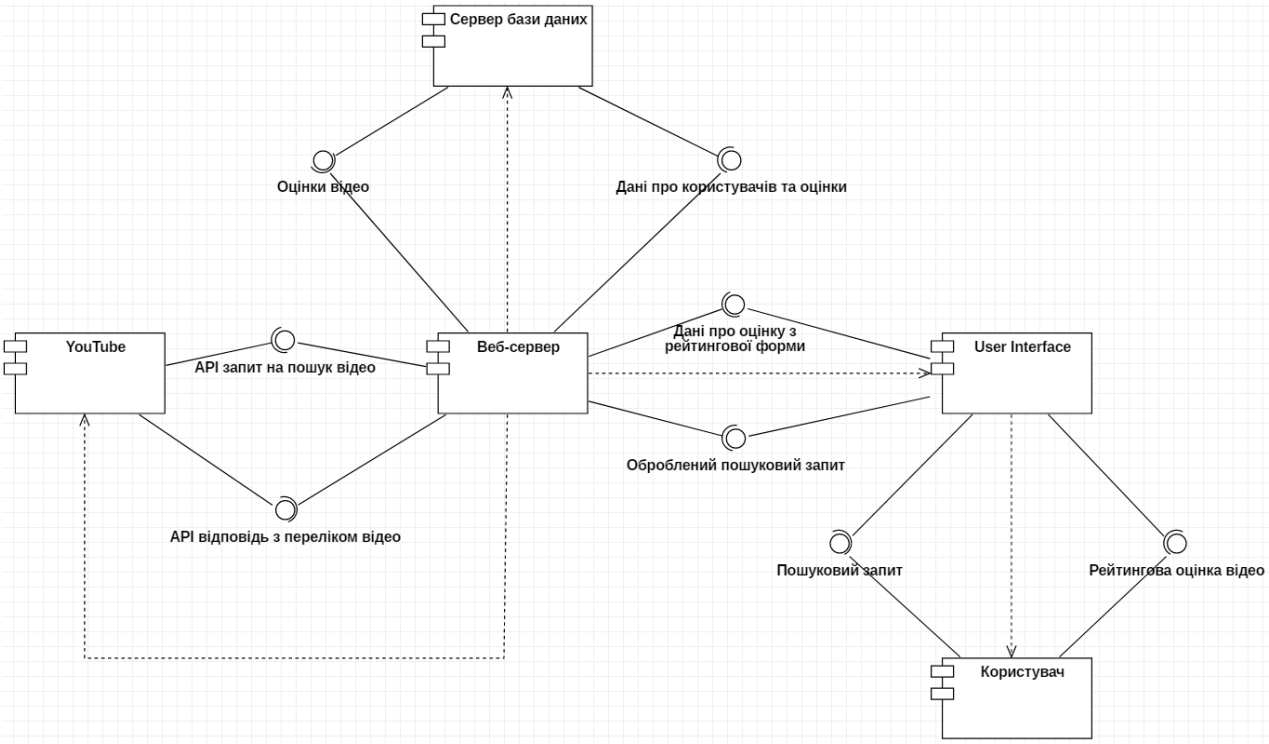

Рисунок 3.1 – UML діаграма компонентів інформаційної системи

Відеоматеріали витягуються з базі сервісу YouTube за допомогою API запитів. Дані по відео та рейтингам зберігаються до БД MySQL. Набір даних у вигляді списку відео та оцінок для кожного користувача формується на основі даних в БД.. Набір рекомендованих відео обчислюється за допомогою методу колаборативної фільтрації та та відображається користувачеві на веб-інтерфейсі

Програмна реалізація рекомендаційної інформаційної системи відбувалась у середовищі PyCharm. База даних з конфігураціями розташовувалась на локальному MySQL сервері. Робота з базою даних

проводилась через вбудований веб інтерфейс адміністратора Django та консольну утиліту manage.py.

Веб-фреймворк Django підтримує MVC [16] (Model-View-Controller) патерн, але його реалізація дещо відрізняється від класичного MVC. Основна відмінність двох моделей полягає в тому, що сам Django має вбудовану частину Controller (програмний код, який контролює взаємодію між Model і View), залишаючи нам Template. Template – це файл HTML, в якому також використовується Django Template Language.

На рис.3.2 приведено схему взаємодії компонентів проекту Django.

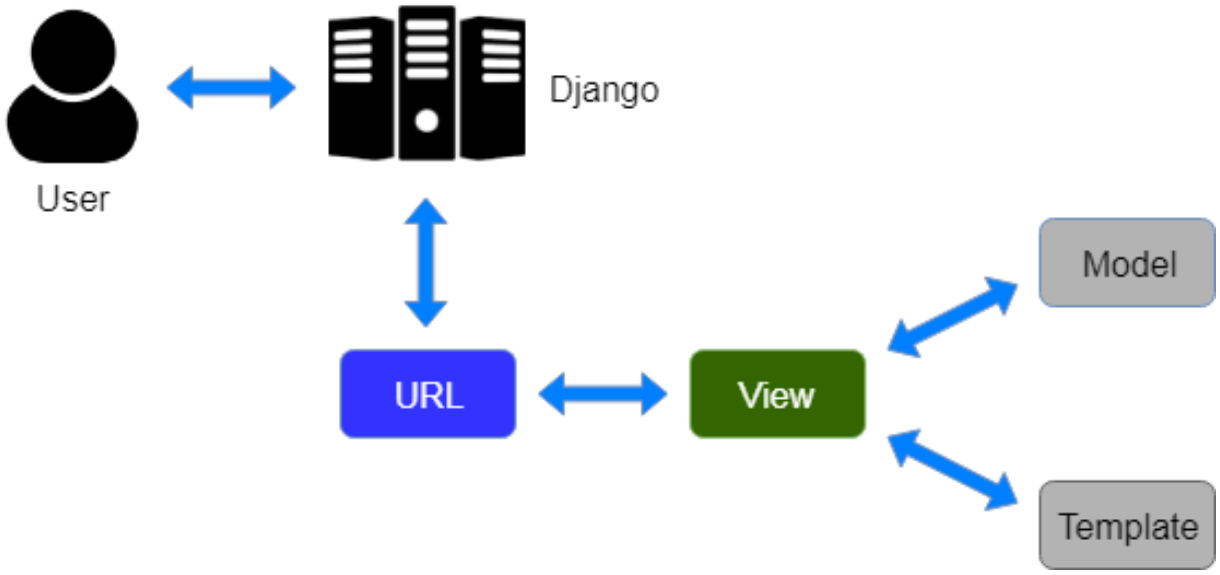

Рисунок 1.2 – Взаємодія компонентів MVT патерну

Структура проекту приведена на рис.3.3 та має наступний вигляд:

- recsys – загальна директорія проекту Django
- recsys/settings.py – файл з налаштування проекту Django (шлях до директорій проекту, опис підключення до бази даних і.т.д.)
- recsys/urls.py – декларації URL для даного проекту та їх представлення їх відповідності до view файлів
- recsys/wsgi.py – модуль налаштувань WSGI сервера для розгортки веб-додатку

- static – директорія для статичних файлів (CSS стилі, JS файли, зображення)
- templates – директорія з переліком HTML шаблонів веб-сторінок додатку
- videorecapp – директорія додатку Django
- videorecapp/admin.py – файл для реєстрації моделей для подальшого редагування на веб-інтерфейсі адміністратора
- videorecapp/apps.py – перелік встановлених додатків
- videorecapp/forms.py – створені користувацькі форми для введення та обробки даних
- videorecapp/matrix_factorization.py – реалізація методу матричної факторизації через алгоритм сингулярного розподілу
- videorecapp/models.py – файл з моделями проекту (відповідають таблицям в базі даних)
- videorecapp/prepare_datasets.py – логіка підготовки наборів даних для подальшої обробки рекомендаційним алгоритмом
- videorecapp/urls.py - декларації URL для даного додатку та їх представлення їх відповідності до view файлів
- videorecapp/views.py – перелік views (функцій для обробки веб-запитів) та логіка обробки даних отриманих з шаблонів
- videorecapp/yt_actions.py – логіка взаємодії з YouTube через YouTube Data API, отримання переліку відео на задану тему
- manage.py - утиліта командного рядка, яка дозволяє взаємодіяти з проектом, базою даних та веб-сервером

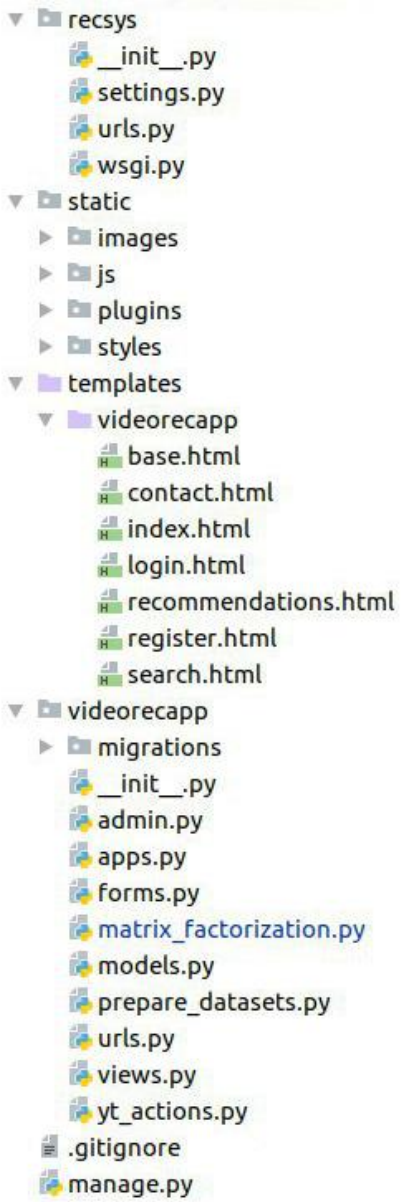

Рисунок 3.3 – Структура додатку Django в середовищі PyCharm

У розробленій рекомендаційній інформаційній системі відбуваються наступні процеси:

- реєстрація користувачів;
- авторизація користувачів;
- взаємодія з YouTube API для витягнення відео-контенту на обрану користувачем тематику;
- відображення відео-контенту на веб-сторінці додатку;
- оцінка відео-контенту користувачем;

- формування набору даних для обробки рекомендаційним алгоритмом;
- визначення актуальних відеоматеріалів та їх представлення користувачам.

В таблиці 3.1 приведено детальний опис основних процесів створеної інформаційної системи.

Таблиця 3.1 – Опис інформаційної системи

Процес	***Інструментарій***	***Результат***
Реєстрація користувачів	Python, функціонал Django, програмний інтерфейс доступу до БД, СКБД MySQL	Збережені реєстраційні дані користувачів
Авторизація користувачів	Python, функціонал Django, програмний інтерфейс доступу до БД, СКБД MySQL	Авторизований доступ до веб-додатку
Взаємодія з YouTube API	Python, функціонал Django, функціонал YouTube Data API	Перелік ідентифікаторів відеоматеріалів на обрану тему
Відображення відео-контенту на веб-сторінці	Python, функціонал Django,	Відображення переліку відео доступних для перегляду та оцінювання
Оцінка відео-контенту	Python, функціонал Django, програмний інтерфейс доступу до БД, СКБД MySQL	Збережені дані про оцінку відео користувачем
Формування набору даних	Python, функціонал Django, програмний інтерфейс доступу до БД, СКБД MySQL	Підготовлений набір даних для обробки рекомендаційним алгоритмом
Визначення актуальних відеоматеріалів	Python, функціонал Django	Перелік рекомендованих відеоматеріалів

Практична реалізація створеної рекомендаційної інформаційної системи дає змогу отримати список рекомендованих навчальних відеоматеріалів для користувача.

Перевагою вибірки відео в розробленій системі є конкретизовані параметри API запиту, який надсилається сервісу YouTube, для пошуку лише тих відео, яку відповідають навчальній тематиці та мають високі рейтинги перегляду

Порівнюючи тестові вибірки відео на тему Python, можно помітити що YouTube надає велику кількість відео й на інші теми, які не відповідають навчальній сфері (рис. 3.4).

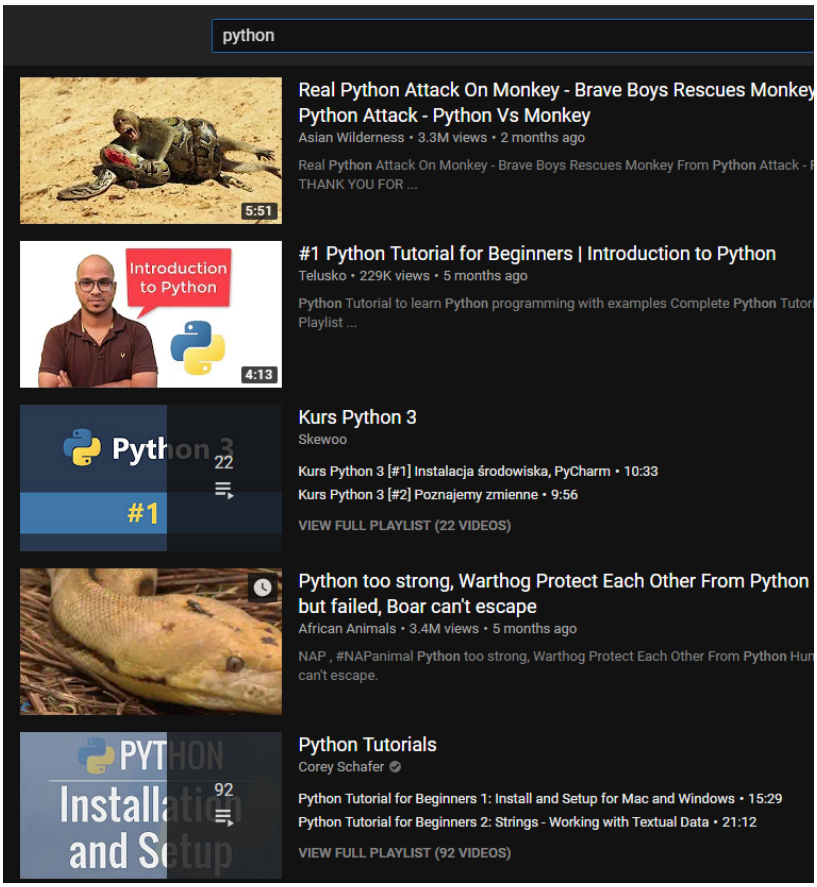

Рисунок 3.4 – Результат пошуку за темою «Python» на YouTube

Оскільки функціонал додатку зосереджений на пошуку лише навчальних відео, це позбавляє рядового користувача від ручної фільтрації знайденого контенту.

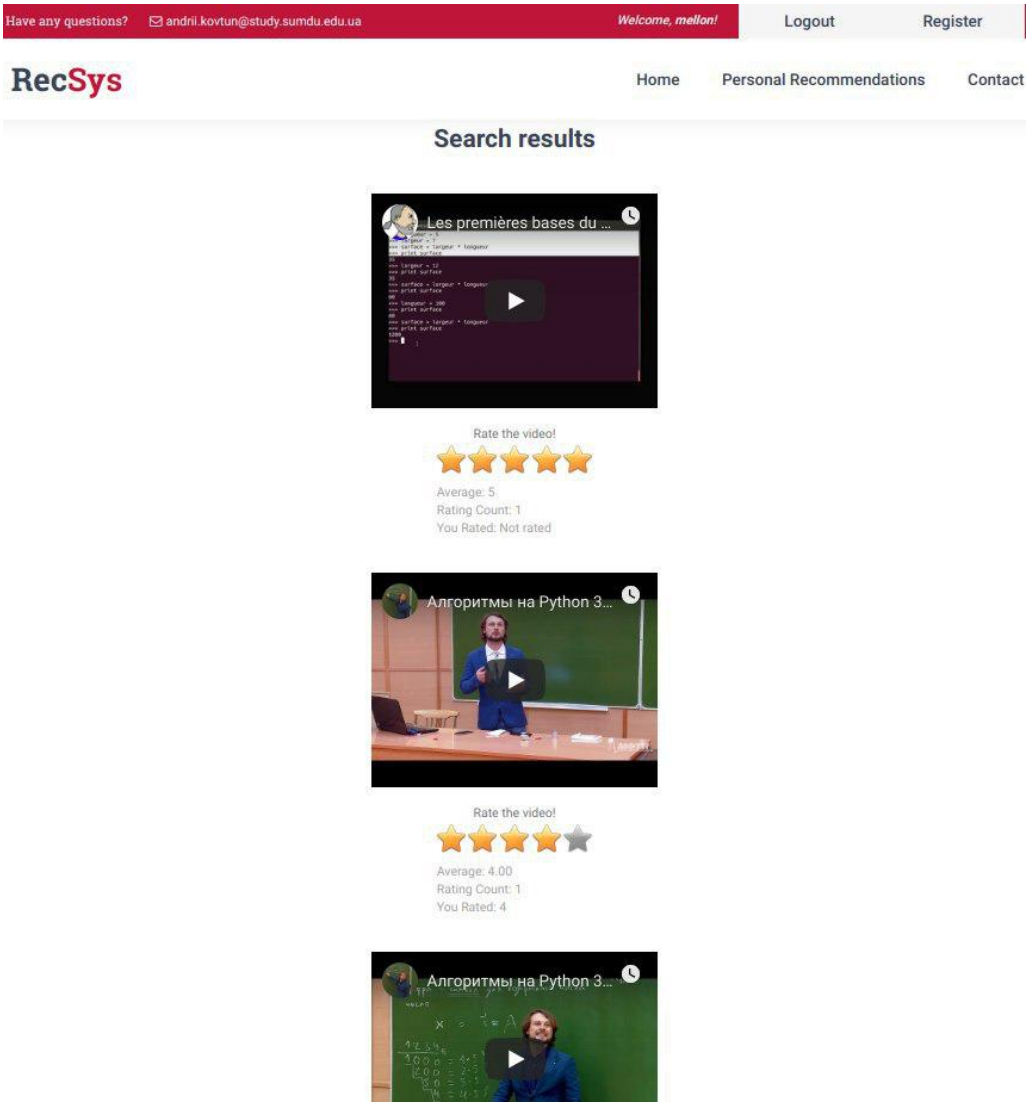

Рисунок 3.4 – Результат пошуку за темою «Python» в розробленій інформаційній системі

ВИСНОВКИ

Рекомендаційна інформаційна система з підбору навчальних відеоматеріалів призначена для знаходження актуальних відеоматеріалів для користувачів системи. Підбір відеоматеріалів базується на методі колаборативної фільтрації та алгоритмі матричної факторизації (декомпозиції), який обробляє дані (рейтингові оцінки відео) користувачів системи та пропонує конкретному користувачу перелік актуальних відео для перегляду.

Рекомендаційна інформаційна система може використовуватися як фізичними особами (учнями, студентами), так і підприємствами малого та середнього масштабу з метою пошуку та фільтрації якісного навчального відео-контенту.

Дана інформаційна система була реалізована як веб-додаток, основним інструментом для її реалізації став веб-фреймворк Django. Доступ до додатку здійснюється через будь-який сучасний веб-браузер, в рамках даної роботи доступ до додатку виконувався через локально налаштований веб-сервер.

Робота додатку відбувається на декількох веб-сторінках, де користувач вводить дані реєстрації, авторизації, запит на пошук відеоматеріалів, виставляє оцінки. Останнім етапом роботи з додатком є виведення переліку рекомендованих відео авторизованому користувачу.

СПИСОК ЛІТЕРАТУРИ

1. Recommender Systems Handbook / R. Francesco, R. Lior, S. Bracha, K. B. Paul. – Dordrecht:Springer, 2015. – 1009 p
2. Music Genome Project [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Music_Genome_Project.
3. Глибовець А. М. Агенти для рекомендацій у колаборативних середовищах / А. М. Глибовець, С. С. Гороховський, А. А. Піка // Наукові праці МДУ ім. Петра Могили. Комп'ютерні технології. – 2010. – Вип. 121, т. 34. – С. 142–151.
4. Building a recommendation engine [Електронний ресурс] – Режим доступу до ресурсу: https://www.packtpub.com/mapt/book/web_development/9781784391911/9/ch09lvl1sec59/building-a-recommendation-engine.
5. Reddit - розважальний, новинний онлайновий сервіс [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Reddit.
6. Digg - соціальний новинний веб-портал [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Digg.
7. Системи засновані на знаннях [Електронний ресурс] – Режим доступу до ресурсу: http://dss-bi.com.ua/System/knowledge-based-systems/?print=pdf.
8. About Amazon [Електронний ресурс] – Режим доступу до ресурсу: https://www.aboutamazon.com/.
9. eBay - онлайн майданчик для проведення аукціонів і торговельний веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/EBay.
10. Netflix - поставщик фильмов и сериалов [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Netflix.
11. Сингулярний розклад матриці [Електронний ресурс] – Режим доступу до ресурсу: https://bit.ly/2RMCNRM.

12. Scipy Lecture Notes [Електронний ресурс] – Режим доступу до ресурсу: https://www.scipy-lectures.org/.
13. GILLENSON M. L. Fundamentals of Database Management Systems / MARK L. GILLENSON., 2011. – (John Wiley & Sons)
14. Component Diagram Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/uml-component-diagram.
15. Dependency in UML [Електронний ресурс] – Режим доступу до ресурсу: https://www.uml-diagrams.org/dependency.html
16. MVC Architecture [Електронний ресурс] – Режим доступу до ресурсу: http://www.tutorialsteacher.com/mvc/mvc-architecture.

ДОДАТОК А

ІНСТРУКЦІЯ КОРИСТУВАЧА

На рис.А.1-А5. приведено приклад роботи з розробленим веб-додатком, починаючи з реєстрації користувача та закінчуючи отриманням рекомендацій. Головна сторінка додатку приведена на рис.А.1. Користувач має можливість авторизуватися, якщо в нього є акаунт в системі (кнопка «Login») або зареєструватися (кнопка «Register»).

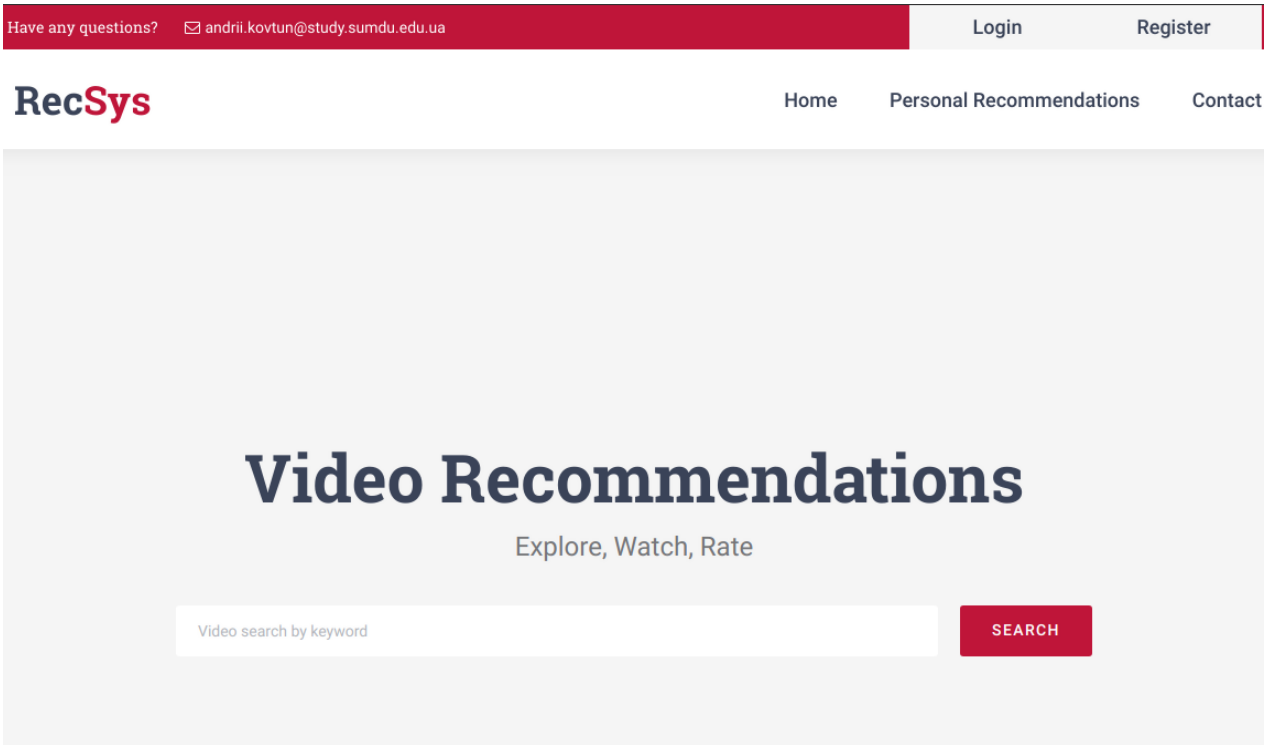

Рисунок А.1 – Головна сторінка додатку

Якщо користувач вирішив зареєструватися в системі, після натиску на кнопку реєстрації відкривається сторінка з реєстраційною формою, як показано нижче на рис. А.2.

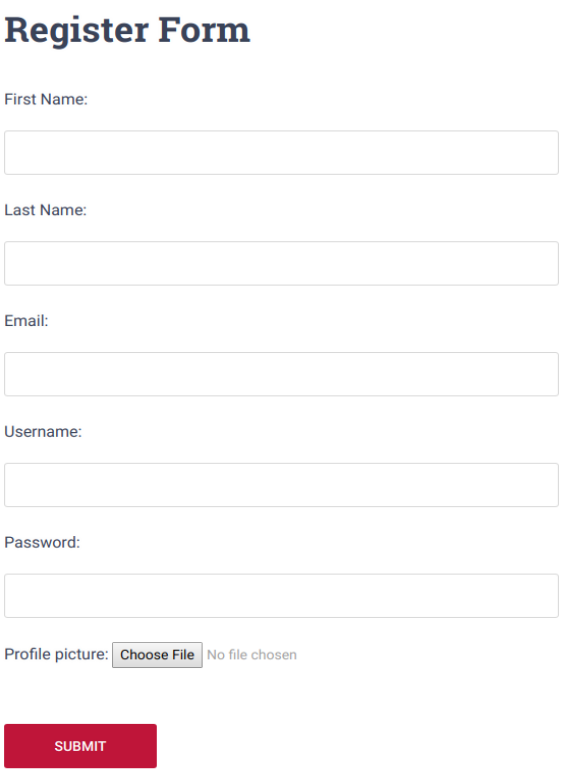

Рисунок А.2 – Форма реєстрації користувача

Якщо користувач має акаунт в системі, він натискає кнопку «Login» на головній сторінці додатку, після чого відкривається сторінка з формою авторизації, як зображено на рис. А.3.

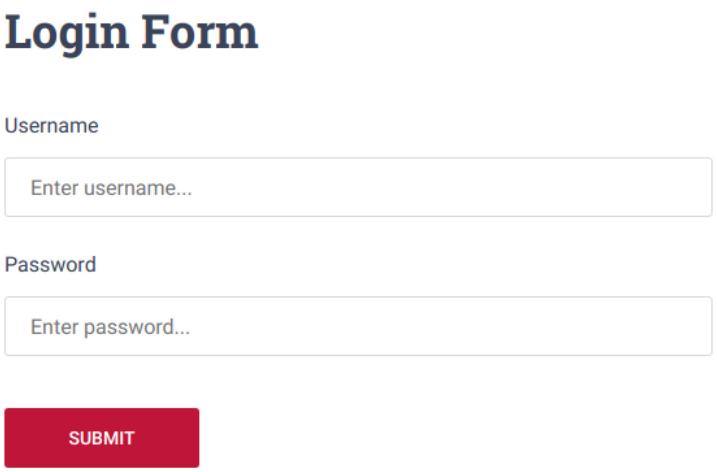

Рисунок А.3 – Форма авторизації користувача

Після успішної авторизації, користувачеві стає доступна для використання форма пошуку відео (рис.А.4).

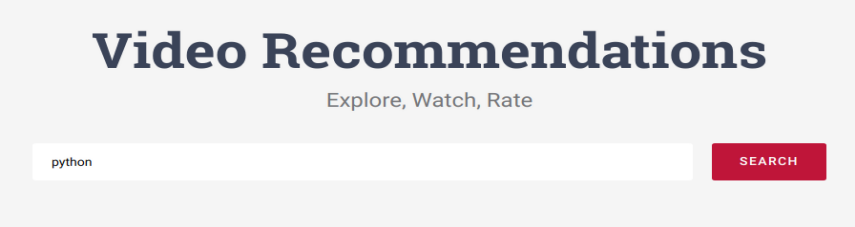

Рисунок А.4 – Використання рядка пошуку для отримання шуканих відеоматеріалів

Користувач вводить рядок запиту та натискає на кнопку «Search». Після чого відображається сторінка з результатами запиту у вигляді списку відео з рейтинговою формою. Користувач має змогу переглянути кожне відео та виставити відповідну оцінку. Приклад сторінки з результатами пошуку приведено на рис.А.5.

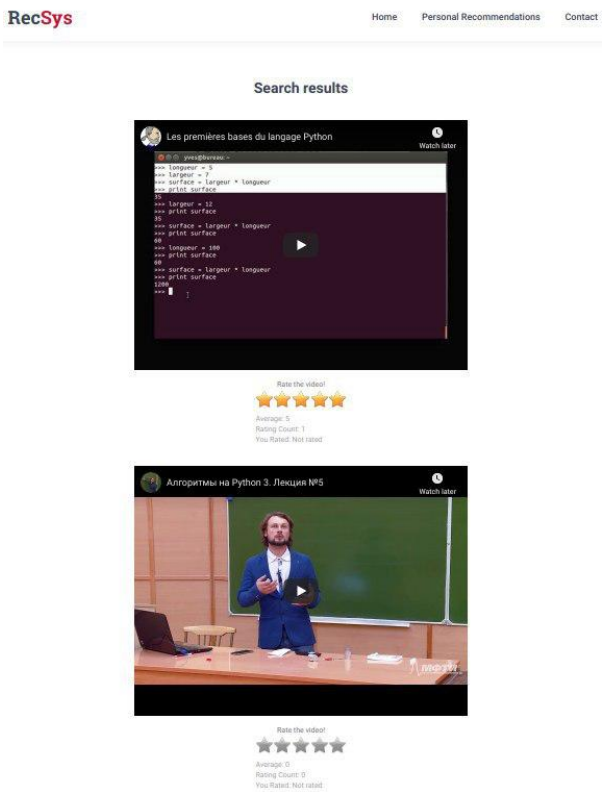

Рисунок А.5 – Результати пошуку відео

Проглянувши та оцінивши низку відеоматеріалів, користувач має змогу переглянути рекомендовані відео. Для цього потрібно натиснути на кнопку «Personal Recommendations», після чого система знаходить відео, які вважає актуальними для даного користувача та відображає нову сторінку з цими відео. Приклад сторінки рекомендацій приведено на рис.А.6.

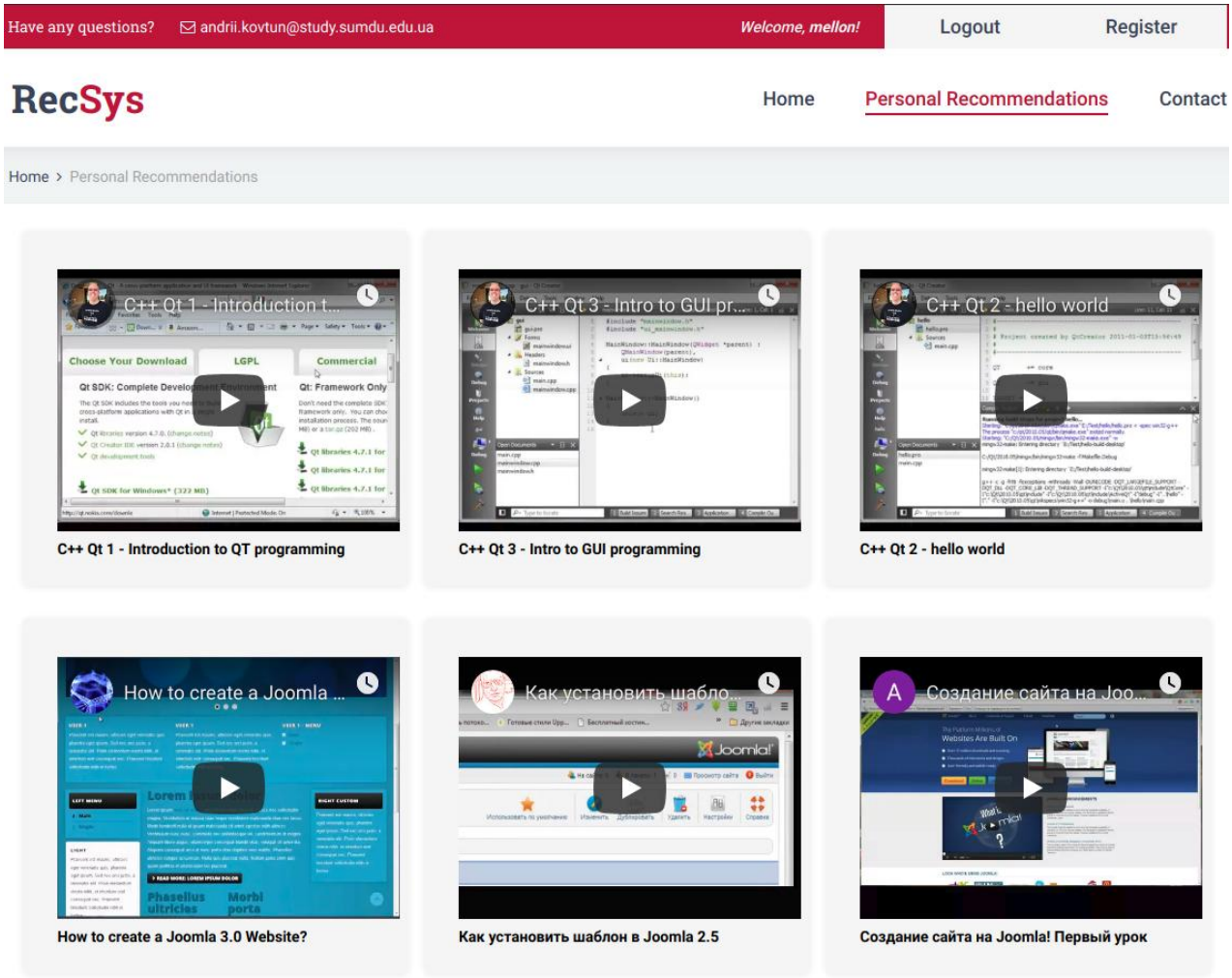

Рисунок А.6 – Сторінка з рекомендаціями

У випадку, коли користувач має бажання зв’язатися з адміністратором сайту (залишити свої коментарі або побажання), користувач має натиснути кнопку «Contact». Після чого відображається сторінка з контактною формою, де користувач повинен заповнити поля з інформацією про себе (ім’я та зворотну адресу, поля «Name» та «Email» відповідно), а також текст повідомлення (поле «Message»). Приклад сторінки з контактною формою приведено на рис.А.7.

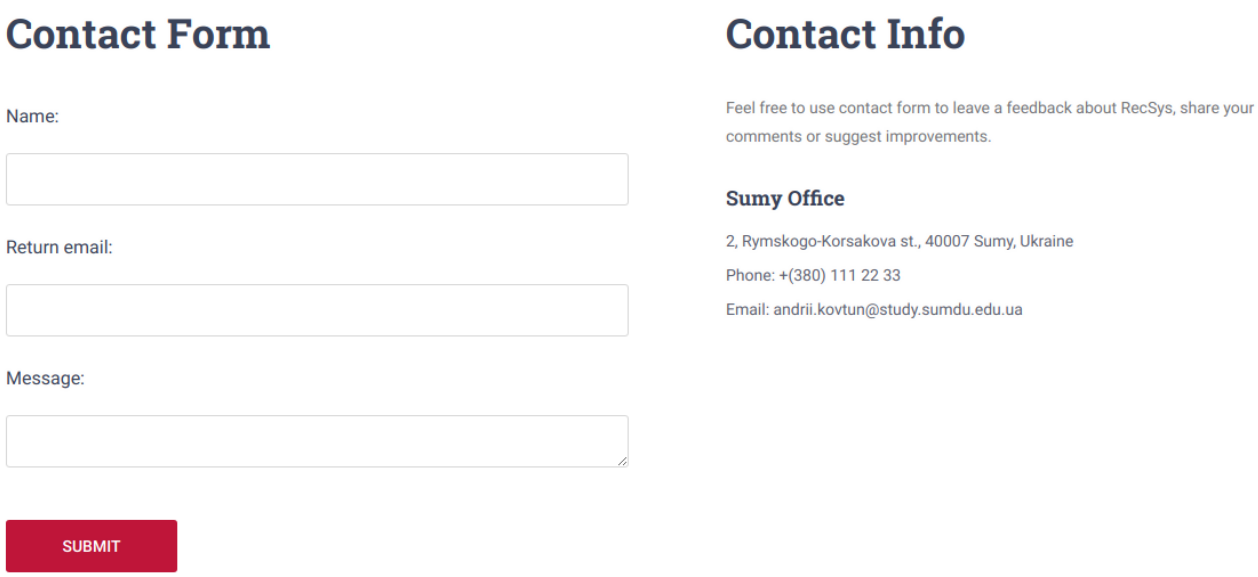

Рисунок А.7 – Контактна форма

ДОДАТОК Б

ІНСТРУКЦІЯ АДМІНІСТРАТОРА

Для комфортної роботи з контентом додатку (базою користувачів, відео та їх рейтингами), адміністратору сайту доступний власний веб-інтерфейс, доступ до якого мають лишу користувачі з правами адміністратора. На рис. Б.1 приведено приклад форми авторизації для доступу до інтерфейсу адміністратора.

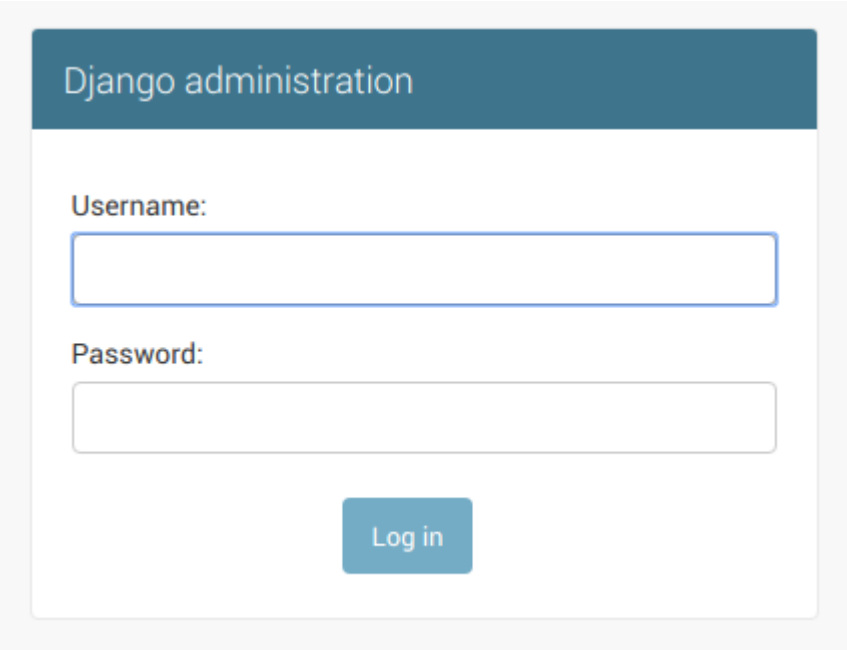

Рисунок Б.1 – Форма авторизації для адміністратора

Після успішної авторизації, відображається сторінка адміністрації сайту, де адміністратор має доступ до таблиць бази даних з можливістю додавання, видалення та редагування даних в цих таблицях. Приклад головної сторінки інтерфейсу адміністратора приведено на рис. Б.2.

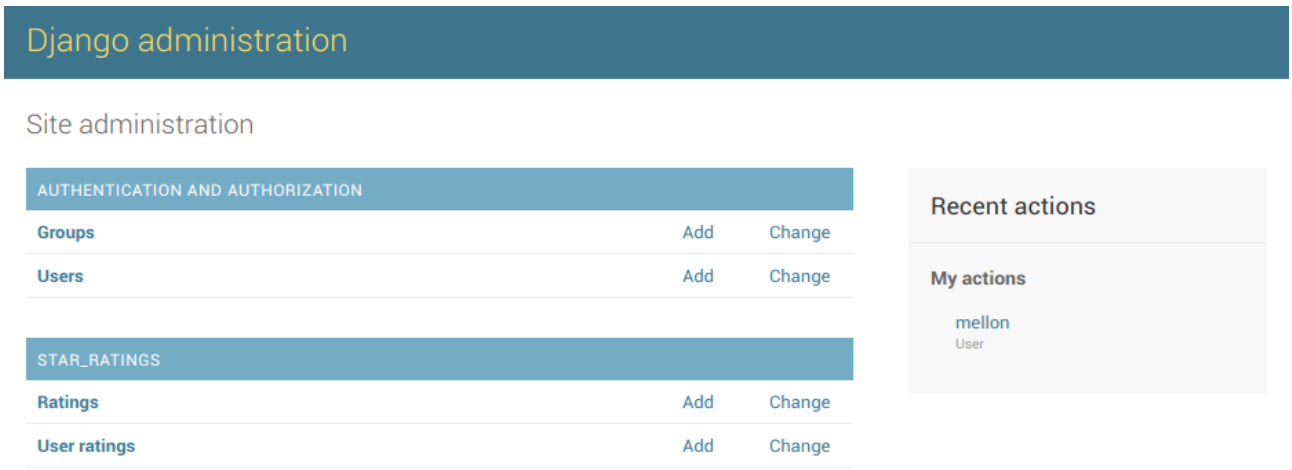

Рисунок Б.2 – Головна сторінка інтерфейсу адміністратора з переліком таблиць

Після вибору таблиці для редагування, відображається сторінка з даною таблицею та даними в ній (рис. Б.3).

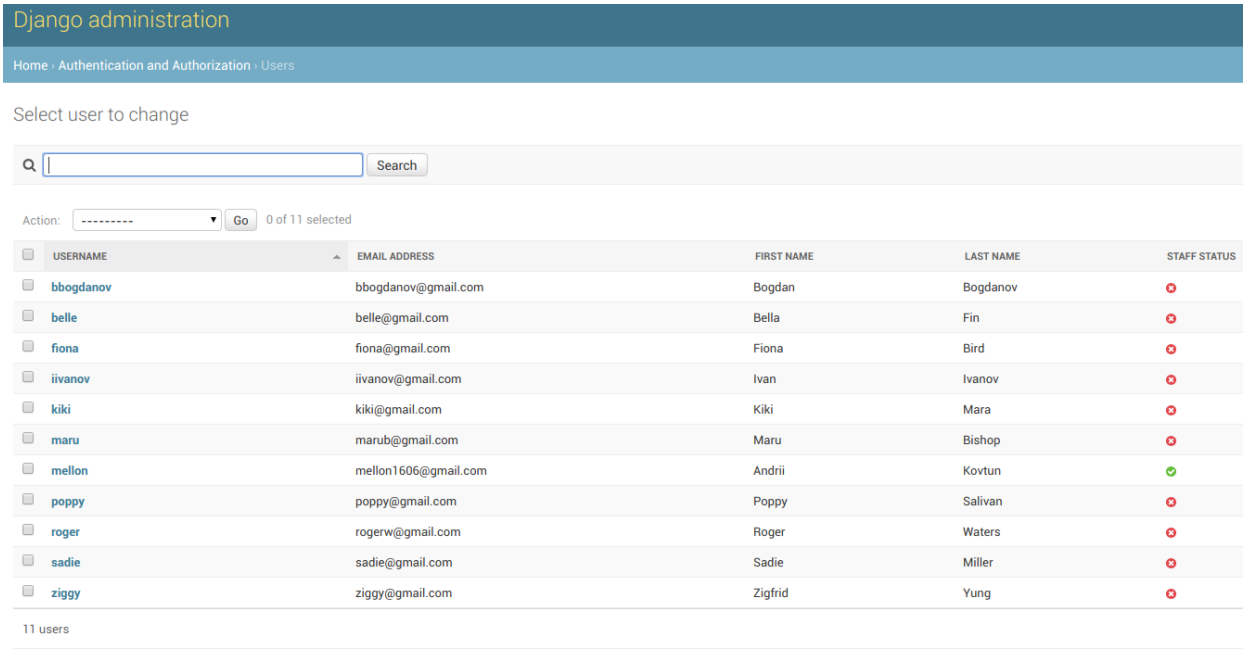

Рисунок Б.3 – Перегляд даних в таблиці Users

Якщо потрібно додати новий запис до таблиці, адміністратор натискає відповідну кнопку («Add User» в наведеному прикладі), після чого відображається форма для заповнення інформації про нового користувача, як зображено на рис. Б.4.

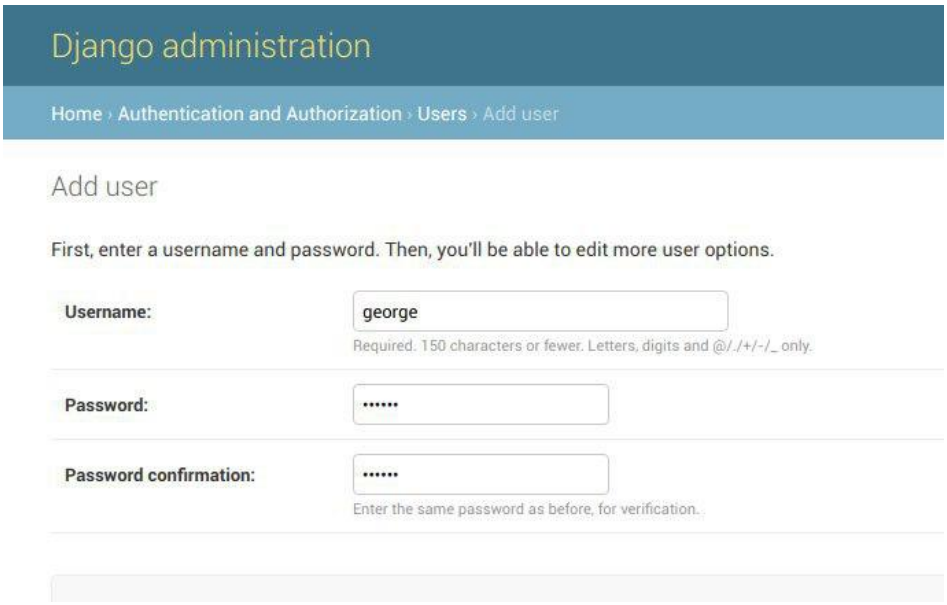

Рисунок Б.5 – Додавання нового користувача через інтерфейс адміністратора

Після введення потрібної інформації, адміністратор натискає на кнопку «Save» та введені дані заносяться до таблиці.

ЛІСТИНГ ПРОГРАМНОГО КОДУ

matrix_factorization.py

```
import pandas as pd
import numpy as np
from scipy.sparse.linalg import svds


def recommend_movies(userID, num_recommendations=5):

    ratings_list = [i.strip().split("::") for i in open('/tmp/recsys_ratings.txt', 'r').readlines()]
    users_list = [i.strip().split("::") for i in open('/tmp/recsys_users.txt', 'r').readlines()]
    movies_list = [i.strip().split("::") for i in open('/tmp/recsys_videos.txt', 'r').readlines()]

    ratings = np.array(ratings_list)
    users = np.array(users_list)
    movies = np.array(movies_list)

    ratings_df = pd.DataFrame(ratings_list, columns=['UserID', 'VideoID', 'Rating'], dtype=int)
    movies_df = pd.DataFrame(movies_list, columns=['VideoID', 'YTID', 'Title'])
    movies_df['VideoID'] = movies_df['VideoID'].apply(pd.to_numeric)

    R_df = ratings_df.pivot(index='UserID', columns='VideoID', values='Rating').fillna(0)
```

```
    R = R_df.values
    user_ratings_mean = np.mean(R, axis=1)
    R_demeaned = R - user_ratings_mean.reshape(-1, 1)

    U, sigma, Vt = svds(R_demeaned)
    sigma = np.diag(sigma)

    all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt) + user_ratings_mean.reshape(-1, 1)
    preds_df = pd.DataFrame(all_user_predicted_ratings, columns=R_df.columns)

    # get and sort the user's predictions
    user_row_number = userID - 1  # UserID starts at 1, not 0
    sorted_user_predictions = preds_df.iloc[user_row_number].sort_values(ascending=False)  # UserID starts at 1

    # get the user's data and merge in the movie information
    user_data = ratings_df[ratings_df.UserID == (userID)]
    user_full = (user_data.merge(movies_df, how='left', left_on='VideoID', right_on='VideoID').
                 sort_values(['Rating'], ascending=False)
                 )

    # here we recommend the most highly-rated predicted movies that user hasn't seen yet
    recommendations = (movies_df[~movies_df['VideoID'].isin(user_full['VideoID'])].
                       merge(pd.DataFrame(sorted_user_predictions).reset_index(), how='left',
                             left_on='VideoID',
                             right_on='VideoID').
```

```
                    rename(columns={user_row_number: 'Predictions'}).
                    sort_values('Predictions', ascending=False).
                    iloc[:num_recommendations, :-1]
                    )

    return user_full, recommendations


already_rated, predictions = recommend_movies(1, 10)
predictions_list = predictions['YTID'].tolist()
result_predictions = []
for vid in predictions_list:
    corrected = vid.strip("")
    result_predictions.append(corrected)
```

models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MaxValueValidator, MinValueValidator

class UserProfileInfo(models.Model):

    user = models.OneToOneField(User, on_delete=models.CASCADE)

    # additional fields
    profile_pic = models.ImageField(upload_to='profile_pics', blank=True)

    def __str__(self):
        return self.user.username
```

```
class Video(models.Model):
    """
    defines a video record retrieved from YouTube:
    """
    yt_video_id = models.CharField(max_length=100, unique=True)
    title = models.CharField(max_length=105, unique=False, null=False)
    embed_url = models.URLField(max_length=100, null=True)
```

forms.py

```
from django import forms
from django.contrib.auth.models import User
from videorecapp.models import UserProfileInfo


class UserForm(forms.ModelForm):


    class Meta:
        model = User
        fields = ('first_name', 'last_name', 'email', 'username', 'password')
        labels = {
            'first_name': 'First Name',
            'last_name': 'Last Name',
            'email': 'Email',
            'username': 'Username',
            'password': 'Password',
        }
        widgets = {
            'first_name': forms.TextInput(attrs={'class': 'comment_input'}),
            'last_name': forms.TextInput(attrs={'class': 'comment_input'}),
            'email': forms.EmailInput(attrs={'class': 'comment_input'}),
            'username': forms.TextInput(attrs={'class': 'comment_input'}),
```

```
            'password': forms.PasswordInput(attrs={'class': 'comment_input'}),
        }

class UserProfileInfoForm(forms.ModelForm):

    class Meta:
        model = UserProfileInfo
        fields = ('profile_pic',)
        labels = {
            'profile_pic': 'Profile picture'
        }

class ContactForm(forms.Form):
    name = forms.CharField(widget=forms.TextInput(attrs={'class': 'comment_input'}), label='Name')
    email = forms.EmailField(widget=forms.EmailInput(attrs={'class': 'comment_input'}), label='Return email')
    message = forms.CharField(widget=forms.Textarea(attrs={'class': 'comment_input'}), label='Message')

class LoginForm(forms.Form):
    username = forms.CharField()
    password = forms.PasswordInput()
```

prepare_datasets.py

```
import os
from django.db import connection

def prepare_datasets():
```

```
filelist = ['/tmp/recsys_ratings.txt', '/tmp/recsys_users.txt', '/tmp/recsys_videos.txt']

prepare_ratings_dataset = """
    SELECT sru.user_id,
      vv.id,
      sru.score
    INTO outfile '/tmp/recsys_ratings.txt'
    FIELDS TERMINATED BY '::'
    OPTIONALLY ENCLOSED BY ""
    LINES TERMINATED BY '\n'
    FROM star_ratings_userrating sru
    JOIN star_ratings_rating srr ON sru.rating_id=srr.id
    JOIN auth_user au ON sru.user_id=au.id
    JOIN videorecapp_video vv ON srr.object_id=vv.id
    ORDER BY sru.user_id;
"""

prepare_users_dataset = """
    SELECT id,
      username
    INTO outfile '/tmp/recsys_users.txt'
    FIELDS TERMINATED BY '::'
    OPTIONALLY ENCLOSED BY ""
    LINES TERMINATED BY '\n'
    FROM auth_user
    ORDER BY id;
"""
prepare_videos_dataset = """
    SELECT id,
      yt_video_id,
```

```
        title
    INTO outfile '/tmp/recsys_videos.txt'
    FIELDS TERMINATED BY '::'
    OPTIONALLY ENCLOSED BY ""
    LINES TERMINATED BY '\n'
    FROM videorecapp_video
    ORDER BY id;
"""

for dataset_file in filelist:
    if os.path.exists(dataset_file):
        os.chmod(dataset_file, 0o777)
        os.remove(dataset_file)
    cursor = connection.cursor()

    # prepare ratings dataset
    cursor.execute(prepare_ratings_dataset)
    result = list(cursor.fetchall())
    ratings_dataset = [list(x)[0] for x in result]

    # prepare users dataset
    cursor.execute(prepare_users_dataset)
    result = list(cursor.fetchall())
    users_dataset = [list(x)[0] for x in result]

    # prepare videos dataset
    cursor.execute(prepare_videos_dataset)
    result = list(cursor.fetchall())
    videos_dataset = [list(x)[0] for x in result]
```

```
    datasets = [ratings_dataset, users_dataset, videos_dataset]
    return datasets

prepare_datasets()
print('DOne')
```

views.py

```
from django.conf import settings
from django.shortcuts import render
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect, HttpResponse
from django.urls import reverse
from django.db import transaction
from django.db import IntegrityError
from django.db import connection
from django.core.mail import send_mail
from videorecapp import yt_actions
from videorecapp.forms import UserForm, UserProfileInfoForm, ContactForm
from videorecapp.models import Video, User
from videorecapp.colaborative_filtering import prepare_dataset, user_recommendations
from star_ratings.models import Rating

def index(request):
    total_users = User.objects.all().count()
    total_videos = Video.objects.all().count()
    total_ratings = Rating.objects.all().count()
    return render(request, 'videorecapp/index.html', {
```

```
        'total_users': total_users,
        'total_videos': total_videos,
        'total_ratings': total_ratings,
    })

def recommendations(request):
    get_user_ids_query = (
        "SELECT id "
        "FROM auth_user "
        "ORDER BY id ASC;"
    )
    get_anonymous_recommendations = ("SELECT title, embed_url "
                    "FROM videorecapp_video v "
                    "INNER JOIN star_ratings_rating r ON v.id=r.object_id "
                    "WHERE r.average=5"
                    )
    if request.user.is_authenticated:
        # create DB connection
        cursor = connection.cursor()
        # get all user ids
        cursor.execute(get_user_ids_query)
        result = list(cursor.fetchall())
        user_ids = [list(x)[0] for x in result]
        dataset = {}
        for user_id in user_ids:
            result = prepare_dataset(user_id)
            user_result = {}
            for key,value in result:
                user_result[key] = value
            dataset[user_id] = user_result
```

```
        recommended_for_user = user_recommendations(dataset, request.user.id)
        if recommended_for_user:
            get_user_recommendations = ("SELECT title, embed_url "
                                        "FROM videorecapp_video v "
                                        "WHERE id IN ({0})"
                                        ).format(','.join(str(video) for video in
recommended_for_user))
            cursor.execute(get_user_recommendations)
            recommended = cursor.fetchall()
            return render(request, 'videorecapp/recommendations.html', {
                'range': range(15),
                'recommended': recommended
            })
        else:
            cursor.execute(get_anonymous_recommendations)
            recommended = cursor.fetchall()
            return render(request, 'videorecapp/recommendations.html', {'range':
range(15), 'recommended': recommended})
    else:
        cursor = connection.cursor()
        cursor.execute(get_anonymous_recommendations)
        recommended = cursor.fetchall()
        return render(request, 'videorecapp/recommendations.html', {'range': range(15),
'recommended': recommended})

def contact(request):
    contact_form = ContactForm()
    return render(request, 'videorecapp/contact.html', {'contact_form': contact_form})

def register(request):
```

```
registered = False

if request.method == 'POST':

    user_form = UserForm(data=request.POST)
    profile_form = UserProfileInfoForm(data=request.POST)

    if user_form.is_valid() and profile_form.is_valid():

        user = user_form.save()
        user.set_password(user.password)
        user.save()

        profile = profile_form.save(commit=False)
        profile.user = user

        if 'profile_pic' in request.FILES:
            profile.profile_pic = request.FILES['profile_pic']

        profile.save()
        registered = True

    else:
        print(user_form.errors, profile_form.errors)

else:
    user_form = UserForm()
    profile_form = UserProfileInfoForm()
```

```
    return render(request, 'videorecapp/register.html',
                  {
                     'user_form': user_form,
                     'profile_form': profile_form,
                     'registered': registered,
                  })

def user_login(request):

    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(username=username, password=password)

        if user:
            if user.is_active:
                login(request, user)
                return HttpResponseRedirect(reverse('videorecapp:index'))
            else:
                return HttpResponse('Account not active')
        else:
            print('*' * 20)
            print('Someone tried to login and failed!')
            print('Username: {0}\nPassword: {0}'.format(username, password))
            print('*' * 20)
            return HttpResponse('Invalid login credentials supplied!')
    else:
        return render(request, 'videorecapp/login.html', {})

@login_required
```

```
def user_logout(request):
    logout(request)
    return HttpResponseRedirect(reverse('videorecapp:index'))

def login_form(request):
    form = UserForm()
    return render(request, 'videorecapp/login.html', {'form': form})

@login_required
def video_list(request):
    if request.method == 'POST':
        search_query = request.POST.get('search_query')
        # second argument for youtube_search is the number of videos to retrieve
        videos = yt_actions.youtube_search(search_query, '15')
        fetched_videos = []
        fetched_video_ids = []
        # create a list of fetched videos as instances of Video model for further DB insert
        for video_id, video_data in videos.items():
            new_video_id = video_id
            fetched_video_ids.append(new_video_id)
            new_video_title = video_data['title']
            new_video_embed_url = video_data['embed_url']
            new_video = Video(yt_video_id=new_video_id, title=new_video_title, embed_url=new_video_embed_url)
            fetched_videos.append(new_video)
        # bulk save of all fetched_videos to DB
        bulk_save(fetched_videos)
        # get the video objects from DB
        videos_to_display = Video.objects.filter(yt_video_id__in=fetched_video_ids)
```

```
    return render(request, 'videorecapp/search.html', {'fetched_videos': videos_to_display})

def send_email(request):
    '''
    if request.method == 'POST':
        contact_form = ContactForm(data=request.POST)
        if contact_form.is_valid():
            subject = 'RecSys: Contact form feedback'
            from_email = settings.EMAIL_HOST_USER
            to_email = ['mellon211195@gmail.com']
            message = contact_form.cleaned_data['message']
            send_mail(subject=subject, from_email=from_email, recipient_list=to_email, message=message, fail_silently=False)
    '''
    return HttpResponseRedirect(reverse('videorecapp:index'))

def bulk_save(queryset):
    for item in queryset:
        try:
            with transaction.atomic():
                item.save()
        except IntegrityError:
            print('Duplicate entry found - {0}'.format(item.yt_video_id))
```

yt_actions.py

```
# youtube interaction logic resides here

import os
```

```
from googleapiclient.discovery import build

DEVELOPER_KEY = os.environ.get('RECSYS_DEV_KEY')
YOUTUBE_API_SERVICE_NAME = 'youtube'
YOUTUBE_API_VERSION = 'v3'

def embed_video(youtube_video_id):
    embed_url = 'http://youtube.com/embed/{0}'.format(youtube_video_id)
    return embed_url

def youtube_search(query, max_videos):
    youtube = build(YOUTUBE_API_SERVICE_NAME,
YOUTUBE_API_VERSION, developerKey=DEVELOPER_KEY)

    ## several parameters are to be specified to get the desired list of videos:
    # q             - the query string
    # part          - snippet
    # maxResults    - number of videos to return
    # type          - video (we don't need playlists or channels here)
    # order         - viewCount (get the most viewed videos)
    # safeSearch    - strict (YouTube will try to exclude all restricted content)
    # videoEmbeddable - true (only retrieve embeddable videos)
    # videoLicense  - creativeCommon (users can reuse videos with this license type)
    # videoCategoryId - id=27 (Education)
    search_response = youtube.search().list(
        q=query,
        part='id,snippet',
        maxResults=max_videos,
        type='video',
        order='viewCount',
```

```
        safeSearch='strict',
        videoEmbeddable='true',
        videoLicense='creativeCommon',
        videoCategoryId='27',
    ).execute()
    videos = {}
    # videos format:
    # videos = {
    #   'id': {
    #           'title': Video title,
    #           'embed_url': URL to use in <iframe> tag in html template
    #       }
    # }
    # add each result to the list of videos
    for search_result in search_response.get('items', []):
        videos[search_result['id']['videoId']] = {}
        videos[search_result['id']['videoId']]['title'] = search_result['snippet']['title']
        videos[search_result['id']['videoId']]['embed_url']                                    =
embed_video(search_result['id']['videoId'])
    return videos
```

АНОТАЦІЯ

Шифр наукової роботи: «Fisher».

Тема роботи: «Інформаційна технологія підбору навчальних відеоматеріалів».

Актуальність роботи зумовлена тим, що кількість відеоматеріалів, розміщених на ресурсі Youtube постійно зростає, тому з’являється потреба в якісній фільтрації навчального контенту під час пошуку відео за певною тематикою.

Мета роботи: розробити інформаційну технологію для підбору навчальних відеоматеріалів.

Об’єкт дослідження – процес знаходження рекомендованих навчальних відеоматеріалів.

Предмет дослідження – інформаційна технологія підбору навчальних відеоматеріалів.

У першому розділі розглянуто актуальність поставленої проблеми, виконано огляд існуючих аналогів, описано метод колаборативної фільтрації застосовно до задачі рекомендації навчальних матеріалів , наведено алгоритм підбору навчальних відеоматеріалів. У другому розділі виконано структурно-функціональний аналіз системи, приведено опис робіт у вигляді діаграми в нотації IDEF0, опис потоків даних в системі у вигляді діаграми в нотації DFD, логічну модель бази даних. У третьому розділі приведено процес програмної реалізації рекомендаційної інформаційної системи, приведено структуру проекту та приклад роботи веб-додатка.

Результатом проведеної роботи є веб-додаток, в якому було реалізовано логіку рекомендаційної моделі навчальних відеоматеріалів. Практичне значення роботи полягає у застосуванні розробленого веб-додатку для пошуку, оцінки та отримання рекомендацій щодо навчальних відеоматеріалів.

Ключові слова: ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, РЕКОМЕНДАЦІЙНА СИСТЕМА, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, МАТРИЧНА ФАКТОРИЗАЦІЯ, PYTHON, DJANGO, ВІДЕОМАТЕРІАЛИ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.