ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра системного аналізу

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «**ІНФОРМАЦІЙНА СИСТЕМА АНАЛІЗУ ТА РЕКОМЕНДУВАННЯ КІНОФІЛЬМІВ ДЛЯ ГЛЯДАЧІВ**»

Виконав: студент 4 курсу, групи САД–41

спеціальності

124 Системний аналіз
(шифр і назва спеціальності)

Дубенець В.В.
(прізвище та ініціали)

Керівник Гордієнко Т.Б.
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра Системного аналізу

Ступінь вищої освіти – «Бакалавр»

Спеціальність – 124 Системний аналіз

ЗАТВЕРДЖУЮ
Завідувач кафедри
Системного аналізу
Т.Б. Гордієнко

«___» _____2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Дубенець Владислав Володимирович

(прізвище, ім'я, по-батькові)

1. Тема роботи: «Інформаційна технологія аналізу та рекомендування кінофільмів для глядачів»

Керівник роботи: Годнієнко Тетяна Богданівна д.т.н., професор

(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «__» _____2023 року №__

2.Строк подання студентом роботи:_____

3.Вхідні дані до роботи: інформація про вподобання користувача, інформація про рекомендаційні системи, науково-технічна література, пов'язана з рекомендаційними системами

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): огляд рекомендаційних систем, постановка задачі та вибір інструментів для реалізації, реалізація системи для рекомендацій кінофільмів

5.Перелік графічного матеріалу: презентація

6.Дата видачі завдання: «23» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання	23.02.2023	виконано
2	Підбір науково-технічної літератури	1.03.2023 – 5.03.2023	виконано
3	Аналіз предметної області	6.03.2023 – 15.03.2023	виконано
4	Достіждення стратегій створення рекомендацій	16.03.2023 – 29.03.2023	виконано
5	Розробка Telegram боту з пошуку схожих кінофільмів	1.04.2023 – 15.05.2023	виконано
6	Тестування розробленої програми	15.05.2023 – 16.05.2023	виконано
7	Вступ, висновки, реферат	16.05.2023 – 17.05.2023	виконано
8	Оформлення пояснювальної записки	17.05.2023	виконано
9	Попередній захист		

Дата видачі завдання: «23» лютого 2023

Студент_____ Дубенець В.В._____
(підпис) (прізвище та ініціали)

Керівник роботи_____ Гордієнко Т.Б_____
(підпис) (прізвище та ініціали)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
ПОДАННЯ

ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ ЩОДО ЗАХИСТУ

БАКАЛАВРСЬКОЇ РОБОТИ

Направляється студент Дубенець В.В._____ до захисту бакалаврської роботи за
(прізвище та ініціали)
спеціальністю 124 Системний аналіз _
(шифр та назва спеціальності)

на тему: «Інформаційна технологія аналізу та рекомендування кінофільмів для глядачів».

Бакалаврська робота і рецензія додаються.
Директор інституту _____ __Кравченко В.І._____
(підпис) (прізвище та ініціали)

Довідка про успішність

Дубенець В.В. за період навчання в Навчально-науковому інституті
(прізвище та ініціали)
телекомунікацій з 2019 року до 2023 рік повністю виконав навчальний план за спеціальністю з таким розподілом оцінок за:
національною шкалою: відмінно ____%, добре ____%, задовільно ____%;
шкалою ECTS: A ____%; B ____%; C ____%; D ____%; E ____%.
Методист факультету_____ _____
(підпис) (прізвище та ініціали)

Висновок керівника бакалаврської роботи

Студент Дубенець Владислав Володимирович

Все це дозволяє оцінити виконану бакалаврську роботу студента Дубенця В.В. на оцінку «_____» та присвоїти йому кваліфікацію бакалавр з системного аналізу.

Керівник роботи _____ Гордієнко Т.Б
(підпис)
«__» _____2023 року

Висновок кафедри про бакалаврську роботу

Бакалаврську роботу розглянуто. Студент Дубенець В.В. допускається до захисту даної роботи в Державній екзаменаційній комісії.
Завідувач кафедри Системного аналізу

_____ _Гордієнко Т.Б._____
(Підпис) (прізвище та ініціали)

«__» _____ 2023 року

Реферат

Текстова частина бакалаврської роботи: 59 с., 19 рис., 1 дод., 19 джерел.

СИСТЕМИ РЕКОМЕНДАЦІЙ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, КОНТЕНТНА ФІЛЬТРАЦІЯ, МЕТОДИ РЕКОМЕНДАЦІЙ, ПОКАЗНИКИ ПОДІБНОСТІ,ЗБІР ДАНИХ, PHP, TELEGRAM БОТ, API.

Об’єктом дослідження є процес надання рекомендацій щодо кінофільмів для гладачів.

Предметом дослідження є методи формування рекомендацій на основі колаборативної, контентної та гібридної фільтрації.

Мета роботи – аналіз та рекомендування кінофільмів для глядачів.

В процесі виконання дослідження було проаналізовано загальний стан рекомендаційних систем. Також було проведено детальний аналіз різних методів побудови таких систем. Були досліджені проблеми, які присутні в систмах рекомендацій, та розглянуті різні методи боротьби з ними.

На основі результатів виконаних досліджень розроблено програму, у вигляді telegtam боту, яка рекомендує кінофільми для глядачів на основі їх вподобань, проведено перевірку на вдосконалення систем.

Зміст

Вступ 0
1 Огляд рекомендаційних систем 1
1.1 Поняття рекомендаційної системи 1
1.2 Методи побудови рекомендацій 5
1.2.1 Фільтрація на основі контенту 6
1.2.2 Колаборативна фільтрація 9
1.3 Проблеми рекомендаційних систем та їх можливі рішення 11
1.4 Аналіз існуючих систем 14
2 Постановка задачі та вибір інструментів для реалізації 18
2.1 Постановка задачі 18
2.2 Функціонал та вимоги до програми 18
2.3 Вибір програмного та технічного забезпечення 19
3 Реалізація системи для рекомендацій кінофільмів 26
3.1 Підготовка до початку роботи 26
3.1.1 Створення боту в Telegram 27
3.1.2 Розробка інтерфейсу користувача 28
3.1.3 Розробка основної частини 33
3.2 Результат тестування програми 36
Висновки 1
Перелік джерел посилань 1

Вступ

В сучасному світі інформаційні технології відіграють важливу роль в розвитку різних галузей. Кіноіндустрія не є винятком і також не стоїть на місці. Вона активно використовує різноманітні технології для різних цілей, зокрема для поліпшення якості та ефективності своєї роботи. Одним з найперспективніших напрямків використання інформаційної технології в кіноіндустрії є створення персоналізованих рекомендацій. З розвитком технологій та зростанням кількості веб-платформ для перегляду фільмів та серіалів, користувачам потрібно мати можливість знайти не тільки якісний контент, а й контент, який відповідає їхнім особистим уподобанням та інтересам.

Кожна людина, рано чи пізно, стикається із ситуацією коли хоче подивитись цікавий фільм, мультфільм або серіал, проте самостійно в неї не виходить знайти нічого, що б її зацікавило. Вона витрачає дуже багато часу на пошуки чогось, як їй здається, вартого уваги, однак, в процесі перегляду розуміє, що це зовсім не те, чого вона очікувала.

В результаті декілька годин даремно витраченого часу, відчуття незадоволення, та навіть, можливо, зіпсований вечір. Саме для того, щоб уникнути схожих ситуацій, створюються системи рекомендацій, які допомагають знайти максимально схожий фільм до уподобань конкретної людини.

Рекомендації – це один із найбільш важливих аспектів кіноіндустрії для глядачів. Існують різні підходи до формування рекомендацій, від традиційного методу пропонування новинок в кінотеатрах до використання інформаційних технологій та аналізу даних про глядацькі вподобання.

Така технологія базується на обробці даних користувачів, їхніх уподобаннях, історії перегляду та відгуках, а також на аналізі великої кількості інформації про фільми та серіали, таких як жанр, акторський склад, режисери,

рейтинг, рецензії, відгуки інших глядачів, та ще багато різних параметрів. Все для того, щоб надавати персоналізовані рекомендації щодо кінофільмів, які можуть зацікавити потенційного глядача.

Завдяки аналізу відгуків, рейтингів та інших даних про користувачів, можливо розуміти їхні вподобання та рекомендувати фільми, які їм сподобаються. Це допомагає покращити якість вибору фільмів для глядачів, а також збільшує імовірність того, що користувачі будуть задоволені після перегляду.

Крім того, інформаційна технологія може бути корисною для кінокомпаній, веб-платформ та в цілому допомогти кіноіндустрії збільшити прибуток і покращити взаємодію з глядачами, залучати нових та зберігати старих. Рекомендації можуть допомогти зацікавити нових глядачів у кінофільмах, які вони, можливо, не знайшли б самостійно. А також, технології аналізу можуть допомогти студіям краще зрозуміти своїх глядачів та розробляти більш успішні стратегії маркетингу та продажу. Більш того, ця технологія може мати широкі застосування в інших галузях, наприклад, у рекомендаціях музики, книг, продуктів харчування, тощо.

Однак, важливо пам’ятати, що такі технології аналізу мають свої обмеження та не можуть гарантувати 100% точності. Іноді можуть виникати ситуації, коли рекомендації будуть не точними, або не відповідати індивідуальним вподобанням користувача. Тому важливо постійно вдосконалювати технології та включати нові фактори в аналіз.

У цій роботі буде досліджуватися інформаційна технологія аналізу та рекомендування кінофільмів, щоб зрозуміти, як вона працює, які методи та алгоритми використовуються для збору та обробки даних, а також які фактори впливають на ефективність таких систем. Метою буде створення ефективної системи надання рекомендацій кінофільмів для глядачів

1 Огляд рекомендаційних систем

1.1 Поняття рекомендаційної системи

З розвитком комерційних сайтів, з’явилась проблема вибору. З плином часу пропозицій стає все більше, а це значить що людям, які не мають особистого досвіду чи певної компетенції в оцінці того чи іншого виду товару, проблематично підібрати собі найбільш підходящу пропозицію. Саме для вирішення цієї проблеми і були створені системи рекомендацій.

Рекомендаційна система представляє собою програмний алгоритм, який повинен будувати для користувача персоналізовані списки рекомендацій товарів чи послуг [1]. Такі системи, зазвичай, базуються на історії взаємодії людини з платформою, аналізом його поведінки, взаємодії з іншими користувачами, її вподобаннями і так далі [3].

Програми рекомендацій значно покращили спосіб взаємодії між сервісом та відвідувачем, адже до появи таких технологій, сайти надавали лише статичну інформацію, тобто однакову для абсолютно всіх користувачів. Тепер же, завдяки рекомендаційним системам, інформація яка надається користувачам, стає більш персоналізованою, яка опирається на конкретні потреби і вподобання конкретної людини. Це значною мірою підвищує якість взаємодії відвідувачів з платформою, та забезпечує краще задоволення користувачів.

Взагалі, рекомендаційні технології досить успішно застосовують у різних сферах, наприклад:

- Електронна комерція. На сьогодні це один із найактуальніших варіантів застосування систем для рекомендації. Завдяки ним людина може вибрати саме той товар, із величезного каталогу, який найбільше задовольнить її [2].

- Медіа-індустрія. Через непомірно великий вибір фільмів, серіалів, телешоу, музики, книг та статей, людям досить важко знайти щось схоже до їх уподобань. В таких випадках рекомендаційні списки допомагають сильно скоротити час на пошуки, та з більшим шансом користувач залишиться задоволеним після перегляду фільму або прочитання книги чи статті.
- Соціальні мережі. Списки можливих знайомих, певних груп та спільнот на основі інтересів та попередніх взаємодій користувача.
- Туризм та готельний бізнес. Рекомендації місця для відпочинку та екскурсій, ресторанів, готелів, а також інших послуг для подорожуючих
- Фінансові послуги. Рекомендації інвестицій або кредитних продуктів на основі фінансових транзакцій та інших факторів.

Загалом, існує декілька основних види рекомендаційних систем [6]:

- Колаборативна фільтрація (Collaborative filtering) [4];
- Метод контентної фільтрації (Content-based filtering).

Найпопулярніший вид системи рекомендацій це колаборативна фільтрація. Логіка колаборативної фільтраці зображена на рис. 1.1.

Стратегія колаборативної фільтрації базується на аналізі інформації про інших користувачів. Алгоритм підбору контенту працює на машинному навчанні. За останні роки якість таких алгоритмів збільшилася. Система визначає всю відому інформацію про користувача, далі знаходить інших людей з подібними інтересами, збирає інформацію про це, які продукти користуються популярністю у них, та вже на основі отриманих даних підготовлює список речей які, ймовірно, зацікавлять конкретного відвідувача.

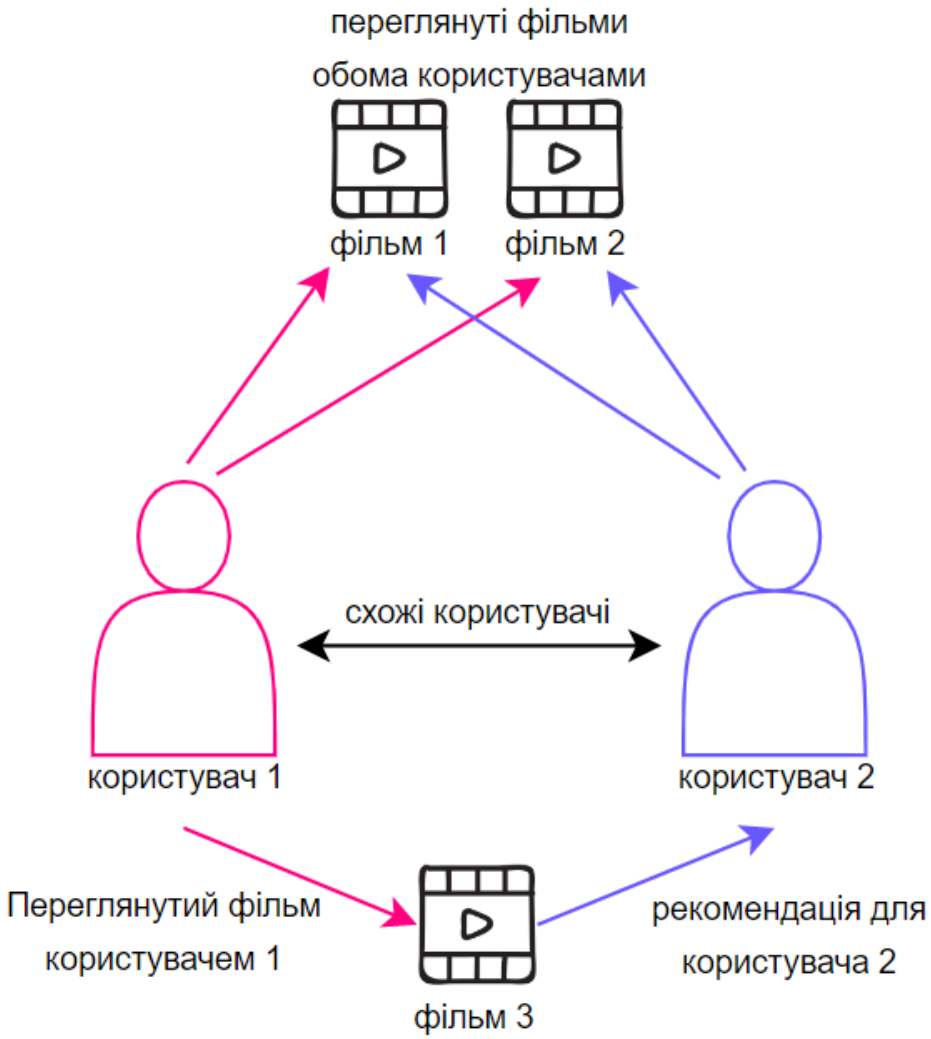

Рис. 1.1 – Логіка колаборативної фільтрації

Метод контентної фільтрації – це одна із основних стратегій рекомендаційних систем, що базується на аналізі продукту та вподобаннях конкретного користувача (рис. 1.2). Цей метод використовується для рекомендації продукту з чітко визначеними характеристиками, як музика, фільми, книги або товари в інтернет магазинах. Система співставляє характеристики товару з вподобаннями людини, та на цій основі шукає контент, який буде схожий по своїм характеристикам на вже куплений товар або переглянутий фільм.

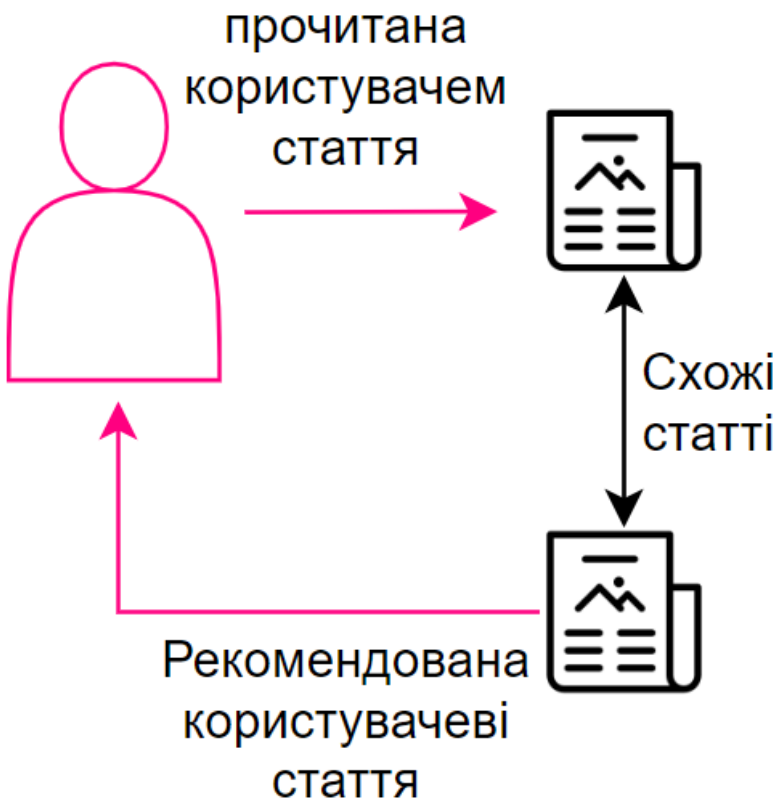

Рис. 1.2 – Логіка контентної фільтрації

Як приклад систем рекомендацій можна навести різні сервіси для перегляду фільмів і серіалів такі як Netflix, Disney+, Amazon Prime Video, Apple TV. Або сервіси для перегляду відео, прослуховування музики та для читання книг: YouTube, Spotify, Scribd, тощо.

У процесі роботи рекомендаційні системи збирають дані про користувачів, використовуючи поєднання явних і неявних методів.

Приклад явного збору даних:

- Оцінка запропонованих товарів або послуг за диференційованою шкалою;
- Ранжування груп об'єктів від найкращого до найгіршого;
- Вибір кращого з двох запропонованих об'єктів;
- користувачу пропонують створити список його улюблених об'єктів.

Приклади неявного збору даних

- спостереження за тим, що користувач оглядає в інтернет-магазинах або базах даних іншого типу;
- ведення записів про поведінку користувача онлайн;
- відстеження вмісту комп'ютера користувача;

1.2 Методи побудови рекомендацій

Існує декілька методів на яких будуються систем рекомендацій, більшість з яких мають спільні риси, які базуються на основних алгоритмах та дозволяють їх класифікувати (рис. 1.3).

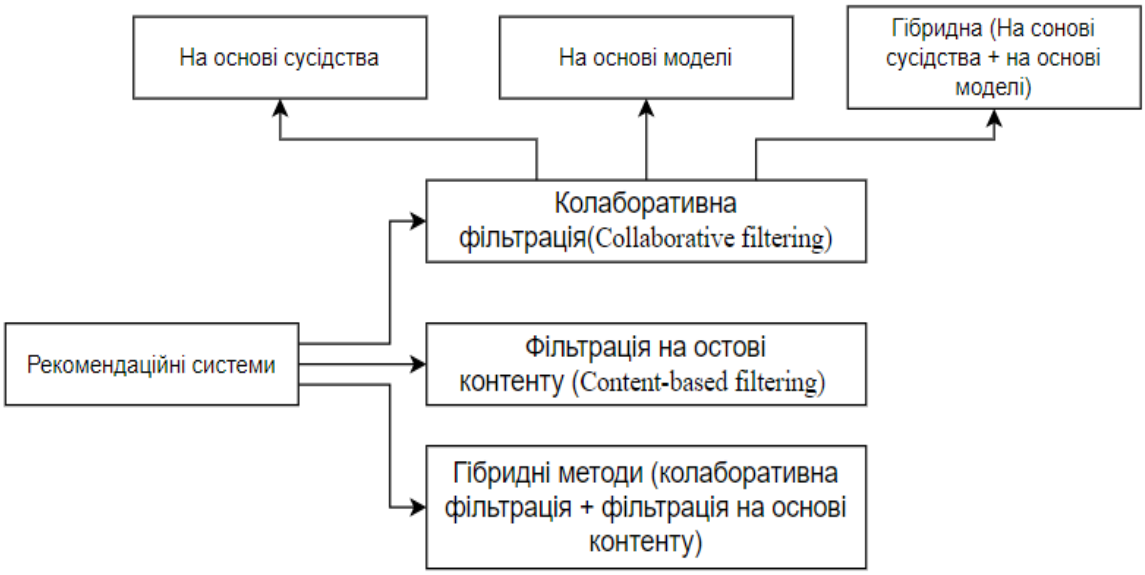

Рис. 1.3 – Основні методи побудови рекомендаційних систем

Одним із способів класифікації рекомендаційних систем, є відбір даних про користувачів. Зазвичай використовується контентна фільтрація, та колаборативна фільтрація, хоча на практиці частіше зустрічається гібридні методи, які поєднують переваги зазначених методів. Розглянемо їх детальніше.

1.2.1 Фільтрація на основі контенту

Це метод фільтрації, який використовується для прогнозування того контенту, який може бути цікавим для користувача на його попередній історії взаємодії з системою та характеристиках контенту[5].

Процес розпочинається на зі збору інформації про вміст (наприклад фільми, книги, музика, тощо), який може бути рекомендований. Ця інформація може включати такі характеристики, як назва, опис, категорії, автори і так далі. Потім, створюється профіль користувача, який містить дані про його минулі взаємодії з системою та його вподобання щодо характеристик контенту.

Далі, використовуючи алгоритми машинного навчання, система порівнює характеристики контенту з профілем користувача, та визначає, які елементи контенту найбільше відповідають вподобанням конкретного користувача. Наприклад, якщо людина переглянула багато фільмів жахів, то система буде рекомендувати їй інші фільми цього ж жанру.

Один з найпоширеніших методів фільтрації на основі вмісту – це використання векторних представлень, таких як TF-IDF (Term Frequency-Inverse Document Frequency) [10] та Word2Vec [11]. За допомогою цих методів кожен елемент контенту та профіль користувача можуть бути представлені у вигляді векторів. Далі, застосовуючи метод обчислення відстаней, система порівнює вектори та визначає, які елементи контенту найбільше відповідають профілю користувача.

TF-IDF це статистична техніка в обробці природньої мови, яка використовується для визначення ваги термінів в документах та колекціях документів. В системах рекомендацій ця техніка використовується для визначення схожості між користувачем та об’єктом, а також для ранжування рекомендацій за їх вагою.

TF (term frequency — частота слова) — відображає кількість входжень конкретного слова в документ. Це означає, що яко термін з’являється більше разів, то його значення в TF-IDF буде вище.

$$\text{TF} = \frac{n_i}{\sum_k n_k} \tag{1.1}$$

Де n_i – число входжень влова в документ, а в знаменнику — загальна кількість слів в документі

IDF (inverse document frequency — обернена частота документа) — відображає наскільки часто конкретний термін з’являється в інших документах в колекції. IDF обчислюється як логарифм відношення загальної кількості документів в колекції до кількості документів в яких з’являється термін. Іншими словами, IDF визначає рідкість терміну у колекції документів.

$$\text{IDF} = \log \frac{|D|}{|(d_i \supset t_i)|} \tag{1.2}$$

Де $|D|$ — кількість документів колекції, $|(d_i \supset t_i)|$ — кількість документів в яких зустрічається слово t_i коли $n_i \neq 0$.

Вибір основи логарифму у формулі не має значення, адже зміна основи призведе до зміни ваги кожного слова на постійний множник, тобто вагове співвідношення залишиться незмінним.

Іншими словами, показник TF-IDF це добуток двох множників TF та IDF. Чим вище значення TF-IDF для певного терміну в документі, тим вища вага цього терміну в цьому документі.

$$TF - IDF = TF \cdot IDF \tag{1.3}$$

Word2Vec — це алгоритм машинного навчання, який використовується для отримання векторних подань слів, на основі контексту, в якому вони зустрічаються у тексті.

Основна ідея алгоритму полягає в тому, що слова які часто зустрічаються в однакових контекстах, повинні мати подібні векторні

подання. Наприклад, якщо слова «кіт» і «собака» зазвичай з’являються в одному і тому ж контексті (мій кіт і моя собака полюбляють гратися з м’ячем), то їх векторні подання повинні бути близьку один до одного в просторі векторів.

Word2Vec має два основні підходи: CBOW (Continuous Bag-of-Words) та Skip-gram. CBOW намагається передбачити поточне слово на основі конспекту, тоді як Skip-gram — навпаки, спробує передбачити контекст на основі поточного слова.

Алгоритм працює наступним чином: спочатку будується словник слів, які зустрічаються в тексті, та відображення кожного слова на унікальний ідентифікатор (наприклад, номер). Далі для кожного слова будується векторний простір фіксованої розмірності. Початкові значення векторів можуть бути різними.

Далі, для кожного слова в тексті будується його «контекст» - множина слів, які знаходяться в певній відстані від нього. Наприклад, якщо брати відстань 2, то контекстом для слова «кіт» можуть бути слова «Мій», «і», «моя», «собака», «полюбляють», «гратися», «з», «м’ячем» - тобто всі слова, які знаходяться на відстані не більш як 2.

Контексти для кожного слова використовуються для тренування моделі. Кожен контекст вважається «запитом», а вектор слова – «відповіддю». Мета полягає в тому, щоб знайти відповідність між запитом та відповіддю, тобто знайти такі векторні подання слів, щоб вони мали подібні відстані в просторі векторів.

Інший метод фільтрації на основі вмісту – це використання рекомендаційних систем, які використовують наліз контенту та знання про смисл контенту для зрозуміння того, що користувач шукає. Ці системи можуть використовувати алгоритми, які використовують глибоке навчання, щоб

зрозуміти контент та пропонувати рекомендації, які найбільше відповідають інтересам користувача.

Узагальнюючи, фільтрація на основі вмісту - це метод фільтрації, який використовує аналіз контенту, щоб надати рекомендації користувачеві. Цей метод може бути корисним, особливо якщо немає даних про інших користувачів або для нового контенту.

1.2.2 Колаборативна фільтрація

Метод колаборативної фільтрації – це техніка аналізу даних, що дозволяє прогнозувати сподівану оцінку користувача для продукту на основі історії його оцінок та історії оцінок інших користувачів з подібними смаками[9].

Основна ідея полягає в тому, що, якщо два користувачі оцінюють різні продукти однаково, то ймовірність того, що одному користувачу сподобається продукт, сподобався іншому – висока.

Розрізняють декілька видів колаборативної фільтрації: на основі моделі, на основі сусідства та гібридний тип.

Колаборативна фільтрація на основі сусідства. Цей алгоритм обчислює схожість двох користувачів, або інших об’єктів, на основі цього складає прогноз для користувача, приймаючи середнє зважене всіх оцінок. Обчислення схожості між об’єктами або користувачами займає важливе місце в цьому підході. Багаторазові заходи, такі як кореляція Пірсона[14], або схожість заснована на скалярному добутку, використовується для цього.

$$\text{simil}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r_x})(r_{y,i} - \overline{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r_x})^2 \sum_{i \in I_{xy}} (r_{x,i} - \overline{r_x})^2}} \quad (1.4)$$

Так визначається схожість двох користувачів X та Y, через кореляцію Пірсона.

Там, I_{xy} – це набір елементів, оцінених як користувачем x, так і користувачем y.

Підхід, заснований на скалярному добутку визначає скалярний добуток між двома користувачами x і y, як:

$$\text{simil}(x, y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}} \quad (1.5)$$

Заснований на користувачеві алгоритм топ-Н рекомендації використовує засновану на подібності векторну модель для визначення К — більшості подібних користувачів до активного користувача. Після того, як знайдені найбільш схожі користувачі, їх відповідні матриці агрегуються для визначення рекомендованого набору елементів. Популярний метод, знаходження схожих користувачів — Locality-sensitive hashing, який реалізує механізм пошуку найближчих сусідів у лінійному часі.

Переваги цього підходу включають в себе: очікуваність результатів, що є важливим аспектом рекомендаційних систем; просте створення і використання; просте полегшення нових даних; добра масштабованість зі співавторами рейтингових пунктів.

Колаборативна фільтрациія на основі моделі. Цей метод надає рекомендації використовуючи параметри статичних моделей для оцінок користувачів, які побудовані з використанням різних методів, таких як метод баєсівських мереж, кластеризація та латентно-семантичні моделі, такі як сингулярний розклад, ймовірнісний латентно-семантичний аналіз, прихований розподіл Дирихле і марковський процес вирішення на основі моделей розробляються з використанням інтелектуального аналізу даних та алгоритмів машинного навчання щоб знайти закономірності. Кількість параметрів у моделі може бути скорочена за допомогою методу головних компонентів.

Цей підхід є більш складним і точним, оскільки він розкриває латентні фактори, які пояснюють спостережувані оцінки. Він також краще обробляє розріджені матриці, що дозволяє обробляти великі набори даних з більшою масштабованістю.

Гібридний підхід[5], що об’єднує в собі підходи засновані на методі та сусідстві, є найбільш поширеним при розробці рекомендаційних систем для комерційних сайтів. Цей підхід допомагає подолати обмеження початкового підходу заснованого на сусідстві і поліпшити якість прогнозів. Крім того, гібридний підхід допомагає подолати проблему розрідженості даних та втрати інформації. Проте, реалізація та використання цього підходу є складним завданням та вимагає багато витрат.

1.3 Проблеми рекомендаційних систем та їх можливі рішення

Не слід забувати і про недоліки в системах рекомендацій. Так, в колаборативному методі фільтрації існує проблема холодного старту[8]. Така проблема в системах рекомендацій виникає, коли система не може надати релевантних рекомендацій новим користувачам або новим товарам чи послугам, які ще не мають історії взаємодії в системі. Це з тим, що системи рекомендацій зазвичай використовують історичні дані взаємодії користувачів із товарами або послугами для прогнозування їх переваг. Коли новий користувач приєднується до системи, система не має інформації про його переваги, і тому вона не може дати точні рекомендації. Це ж стосується і нових товарів і послуг, які не мають історії взаємодії з користувачем або іншими товарами в системі. Проблема холодного старту може суттєво вплинути на якість рекомендацій та призвести до того, що користувачі будуть розчаровані в платформі та перестануть її використовувати. Щоб вирішити цю проблему, існують різні методи, такі як використання контекстної інформації про нових користувачів або нові товари чи послуги, а також використання інших джерел даних, таких як профілі соціальних мереж[8] і т.д.

Проблема синонімії[3] в системах рекомендацій полягає в тому, що іноді користувач може використовувати синоніми термінів, що означають одне й те ж поняття, і в результаті система може пропустити релевантні рекомендації. Наприклад, якщо користувач шукає рекомендації для книг про космос, він може використовувати слова «космос», «всесвіт» або «космічний простір» замість терміну «космос». Якщо система рекомендацій не здатна зв'язати ці терміни, вона може пропустити релевантні рекомендації, що може призвести до незадоволення користувача. Ще одна проблема пов'язана з тим, що деякі синоніми можуть мати різний контекст або забарвлення. Наприклад, слова «герой» і «переможець» можуть мати різне забарвлення, тому система рекомендацій може запропонувати книги, які не відповідають смислу запиту користувача. Для розв'язання проблеми синонімії в системах рекомендацій можуть використовувати методи обробки природньої мови, щоб зрозуміти значення запитів користувачів та враховувати контекст і забарвлення синонімів. Також можуть використовуватися алгоритми машинного навчання для автоматичного вивчення семантичних зв'язків між словами та рекомендацій.

Ще одна важлива проблема, це проблема шахрайства[7]. Рекомендаційні системи стали сильно впливати на продажі та прибуток, з тих пір як стали широко використовуватися в комерційних сайтах. Зловмисники можуть спотворювати рейтинг певних товарів чи послуг, або навпаки, додавати хороші оцінки для своїх товарів. Це все призводить до того, що недобросовісні постачальники товарів та послуг намагаються шахрайськими методами поліпшити продажі своїх товарів, та погіршити їх у конкурентів. Для запобігання таким діям, можуть бути використані алгоритми машинного навчання, щоб автоматично виявляти фальшиві профілі та відгуки. Також можуть бути додані нові правила та стандарти, які контролюють маркетингові та рекламні компанії, щоб запобігти використанню рекомендаційних систем для сприяння шахрайським діям.

Розрідженість даних. Одна з найпоширеніших проблем, які зустрічають при використанні систем рекомендацій. Вона виникає, коли у базі даних немає інформацій про користувачів або предмети, що робить неможливим надання точних рекомендацій. Таке може статися у новій системі яка просто не стигла зібрати всі дані про користувачів, або коли люди не користуються системою рекомендацій, не ставлять оцінки товарам і т.д. Також розрідженість даних посилює проблему холодного старту.

Щоб вирішити проблему, можна збільшити кількість даних у базі даних, наприклад, шляхом залучення нових користувачів, однак це не гарантує сильного ефекту. Також можна залучити алгоритми машинного навчання, які дозволять зменшити вплив розрідженості на рекомендації. Використання додаткової інформації про користувачів або товари, якщо є така можливість.

Фільтрація на основі вмісту також має свої недоліки, такі як обмежена різноманітність рекомендацій. Це означає, що система може рекомендувати тільки те, що схоже на те, що користувач уже споживав. Крім того, цей метод може не враховувати контекст користувача та може приводити до підводних каменів, таких як «фільтр пузиря», коли користувачі отримують рекомендації, які відповідають їхнім вподобанням, але не дають їм нового досвіду або не збагачують їхній кругозір.

Наприклад, якщо користувач завжди переглядає тільки футбольні матчі, то фільтрація на основі вмісту може рекомендувати тільки інші футбольні матчі, але не буде рекомендувати матчі інших видів спорту або інші типи відео контенту.

Також, фільтрація на основі вмісту може бути недостатньо точною, якщо відбір продуктів залежить від складної системи тегів або ключових слів. Ще, якщо компонент, який рекомендується, має багато характеристик, що змінюються з часом (наприклад, фільм з різних років), то фільтрація може бути менш точною.

Щоб запобігти цим проблемам, рекомендаційні системи можуть використовувати комбінацію фільтрації на основі вмісту та інших методів фільтрації, таких як фільтрація на основі поведінки користувача. Це може допомогти забезпечити різноманітність рекомендацій та враховувати контекст користувача.

1.4 Аналіз існуючих систем

Існує багато ресурсів які використовують рекомендаційні системи. Розглянемо деякі з них.

Наприклад, сервіс для перегляду відео – YouTube. Він показує ролики користувачам на основі їх реакцій. Такий зворотній зв’язок може бути явною (кліки, лайки та дизлайки, підписки, тощо), та неявною (час перегляду відео). Так, сьогодні YouTube приховує дизлайки від глядачів та авторів, однак і досі використовує їх показники у своїх системах і алгоритмах.

Весь процес рекомендацій проходить в три етапи:

Етап перший. Використовуючи дані глядачів, алгоритм автоматично створює профілі користувачів. Системи аналізують історію пошуку кожного окремого глядача, алгоритми зпівставляють відео на основі історії користувачів та аналогічного контенту який вони дивились раніше.

Етап другий. Групуються користувачі зі схожими смаками, для того аби заповнити відсутні дані про те, що може сподобатись або не сподобатись окремим глядачам. Алгоритми аналізують схожі групи людей використовуючи демографічні дані, наприклад.

Етап третій. Тут відбувається ранжування роликів які, потенційно, можуть бути цікаві глядачу. Воно уже включає в себе аналіз метаданих відео (назву, опис, теги), а також показник ефективності кожного окремого відео, наприклад показник залучуваної аудиторії.

Алгоритм віддає перевагу відео з більшою залучуваністю, тобто з більшою кількістю переглядів, лайків, коментарів і поширень, зіставляючи подібні вибірки роликів для глядачів. Але для того, щоб пропозиції були настільки ж різні як і уподобання кожного користувача, вводиться деяка випадковість. Ця випадковість хороша з двох причин: по-перше, це заохочу дивитись різні по тематиці та змісту відео. По-друге, вона рекомендує глядачам відео з менш популярних каналів, саме через це шанс у маленьких каналів стати популярними збільшується. Згодом, YouTube використовує цей профіль, щоб вирішити які рекомендації робити і на далі. Алгоритм спочатку генерує підходящі ролики, а потім ранжує їх в залежності від показника ефективності. Схема системи рекомендацій YouTube зображена на рис. 1.4.

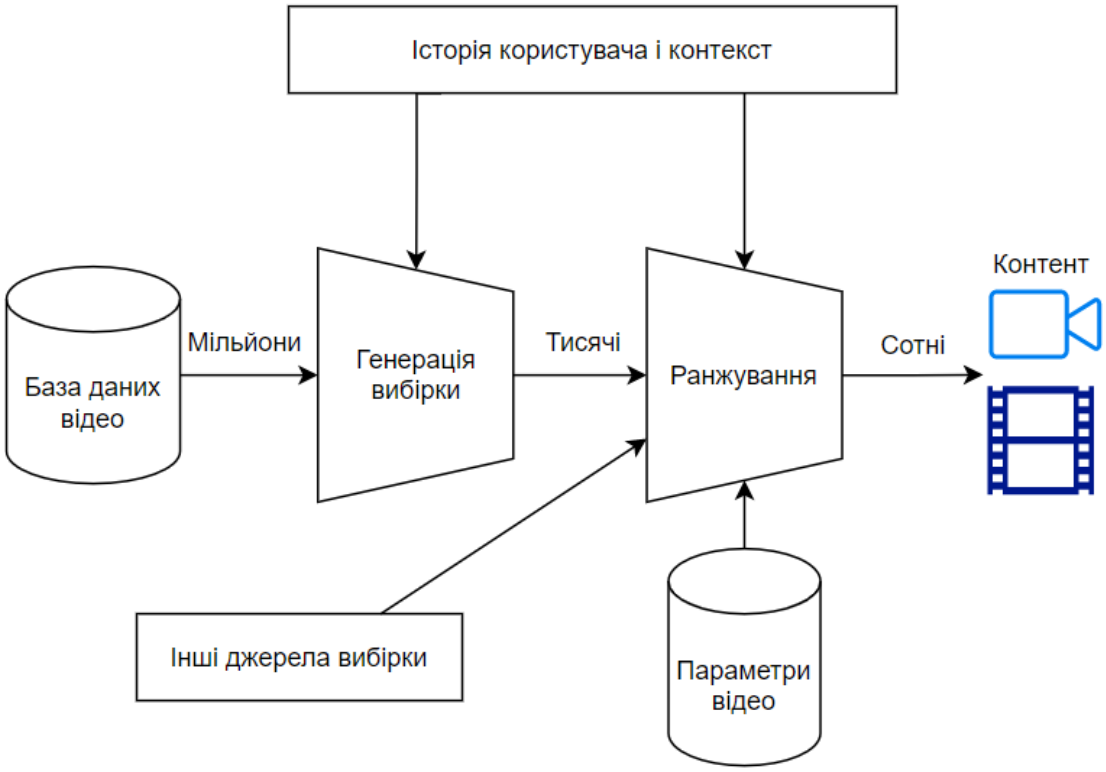

Рис 1.4 – Схема системи рекомендацій YouTube

Або ж Netflix [12], який використовує технології штучного інтелекту та машинного навчання, щоб показувати своїм глядачам персоналізовані

трейлери фільмів і телешоу відповідно до їхніх уподобань. Компанія розробила власний алгоритм під назвою Cinematch, який успішно зіставляв вміст у 75% випадків [15]. Успіх визначався як рекомендація фільму, який отримав рейтинг у межах 0,5 балів від рейтингу фільму, використаного для рекомендації.

Cinematch працював на основі кількох факторів: по-перше, самі фільми сортувалися за жанром, роком випуску, режисерами, акторами тощо. По-друге, враховувалася історія рейтингів окремих користувачів, включаючи фільми, які вони дивилися та стояли в черзі. І по-третє, враховувався загальний рейтинг усіх користувачів Netflix.

Ця система допомогла уникнути банальностей, коли вам сподобалося «Кримінальне чтиво», вам може сподобатися «Шалені пси». Cinematch створював набагато більш віддалені зв’язки на основі оцінок інших користувачів і часто давав несподівані результати.

Ось як це працювало: скажімо, ви дали високу оцінку «Кримінальному чтиву». Cinematch знаходить інших людей, які також дали високу оцінку «Кримінальному чтиву», а потім визначає, які інші фільми ці люди також високо оцінили. Наприклад, він може виявити, що кілька людей також дали високу оцінку «Бейб: поросятко в місті». Cinematch розраховує ймовірність того, що вам може сподобатися «Бейб», виходячи з кількості людей, які поставили йому високу оцінку. Алгоритм продовжується, з’єднуючи все більше несхожі фільми. Cinematch рекомендував класику, блокбастери та незалежні фільми з однаковим успіхом, а не лише популярні фільми.

Звичайно, система побудована на сухій математиці. Однак, на відміну від інших подібних систем, які обмежуються популярними назвами та жанрами, Cinematch імітує щось на зразок виноградної лози, ланцюжок живих рекомендацій від людей зі схожими смаками.

Однак 75% виявилося недостатньо для Netflix. Тому в 2006 році компанія запустила конкурс[13] на нову систему рекомендацій з призом у мільйон доларів. Команда-переможець повинна була підвищити точність рекомендацій щонайменше на 10%. Це призвело до створення нового алгоритму під назвою The Ensemble, який поєднував різні методи машинного навчання для отримання ще більш персоналізованих рекомендацій.

Загалом Cinematch та його наступники революціонізували спосіб споживання медіа та представили нам новий контент, який ми могли б пропустити. За допомогою штучного інтелекту та машинного навчання ці системи змінили індустрію розваг і зробили наше враження від перегляду приємнішим та персоналізованим.

2 Постановка задачі та вибір інструментів для реалізації

2.1 Постановка задачі

Метою роботи є:

- Огляд існуючих систем для рекомендацій
- Аналіз якості надання рекомендацій в подібних системах
- Створення власної програми у вигляді Telegram бота, яка буде надавати якісні рекомендації кінофільмів.

2.2 Функціонал та вимоги до програми

Програма має бути максимально проста у використанні та розумінні, при цьому мати досить значний список функцій для комфортного і ефективного користування.

Вимоги:

- Інтуїтивно зрозумілий інтерфейс для швидкої навігації по меню програми. Так користувач зможе швидко знайти необхідні опції та функції, не витрачаючи багато часу. Адже завантаженість інтерфейсу може призвести до погіршення користувацького досвіду, заплутаності та незручностей у використанні програми. Також, одним із важливих аспектів зручного інтерфейсу є задоволення користувача в процесі використання. Це допомагає покращити продуктивність роботи та утримати людину від переходу до конкурентів.
- Доступ до програми з будь-якого пристрою та в будь-якому місці. Взагалі, в наш час, коли в кожної людини є доступ до інтернету та декілька гаджетів, це є базовою вимогою.
- Програма повинна працювати швидко, правильно, та без різного роду помилок, щоб забезпечити найкращий досвід роботи. Це також є стандартною вимогою.

Основний функціонал програми:

- Знаходження схожих фільмів на той, що вкаже користувач.

- Можливість пошуку конкретного фільму, серіалу або мультфільму по назві, а також по жанру і року виходу.

Додатковий функціонал програми:

- Після того, як користувач знайде підходящий фільм для перегляду, при натисканні кнопки «Переглянути», його буде перекидати на сторінку з цим фільмом одного із онлайн кінотеатрів.

- Авто-рекомендації. Час від часу бот буде надсилати повідомлення з Підбіркою фільмів, на основі списку обраних фільмів користувача.
- Можливість додавати вподобані фільми до списку обраного.
- Список найкращих фільмів та серіалів за весь час.
- Список популярних фільмів і серіалів, який буде постійно оновлюватись.
- Відстеження виходу новинок.

Для кращої взаємодії з ботом, можна реалізувати додаткові функції, такі як наприклад профіль користувача в якому можна буде переглянути та налаштувати різні параметри (вказати улюблений жанр, як приклад). Кнопка для виводу різної інформації стосовно того, як користуватись програмою, на що вона взагалі здатна, для чого створена та інші відомості, розділ з відповідями на різного роду важливі запитання, які можуть виникнути у користувача.

2.3 Вибір програмного та технічного забезпечення

Для реалізації системи є багато зручних рішень, однак вибір пав на Telegram бота. Причина такого вибору полягає у простоті створення боту та використанні, а також популярності Telegram.

Telegram не просто месенджер, він є універсальним набором функцій, який більше підходить для соціальних мереж або Інтернету. Цей кросплатформовий продукт обміну контентом надає можливості для миттєвого обміну різними видами контенту, включаючи звичайні повідомлення, дзвінки та мультимедіа. Одна з головних причин, чому Telegram цінується, полягає у його безпеці, що забезпечується наскрізним

шифруванням, яке захищає дані користувача від перехоплення зловмисниками. У Telegram ви можете спілкуватися як в особистих, так і в групових чатах, знаходити потрібні канали по темі, що вас цікавить, бути в курсі останніх подій, ставати блогером і автором унікального контенту, переглядати фільми, грати в ігри, слухати музику, створювати ботів для бізнесу або особистого використання, та багато іншого.

Месенджер має аудиторію в більш ніж 500 млн. користувачів, це означає, що можна залучити широку аудиторію не обмежуючи себе географічними межами. Що до зручності використання, то телеграм бот знаходиться в месенджері, який дуже багато людей уже використовує щодня. Крім того, бота можна додати в свій особистий чат з друзями або рідними, що дозволяє всім учасникам одночасно бачити всю інформацію про фільм, та швидше обрати щось для перегляду. Також, користувачі можуть отримувати повідомлення з рекомендаціями прямо у месенджер, що збільшує імовірність їх перегляду. Боти можуть працювати на будь-якому пристрої з Telegram, включаючи смартфони, планшети та персональні комп'ютери. Найкраще те, що людині не потрібно нічого встановлювати і забивати і так невелику кількість пам'яті на пристрої. Достатньо надіслати боту повідомлення та почати ним користуватись.

Ось ще декілька переваг телеграм ботів:

- Бот працює цілодобово, перестане функціонувати лише у випадку аварії на сервері, що стається надзвичайно рідко.
- Не потрібно чекати відповіді довгі декілька хвилин, а іноді навіть годин і днів. Телеграм бот відповідає моментально.
- Відсутні всілякі вимоги до потужності пристрою, адже для роботи боту використовуються потужності сторонні сервери.
- Достатньо високий рівень безпеки.
- Телеграм бот дуже зручний та легкий у використанні і при цьому може бути надзвичайно функціональним.

Мовою програмування було обрано PHP (Hypertext Preprocessor) – C-подібна скриптова мова загального призначення, яка активно застосовується в розробці веб-додатків. А якщо точніше, на CodeIgniter 3 – MVC (Model-View-Controller) фреймворк з відкритим вихідним кодом написаний на мові програмування PHP.

До переваг CodeIgniter можна віднести:

- Швидкість. Це дозволяє створювати високопродуктивні програми.
- Модульність. Фреймворк підтримує модульну архітектуру, що дозволяє розробникам швидко розширювати функціональність додатків.
- Безпека. CodeIgniter має вбудовані функції для захисту від SQL-ін'єкцій та інших типів атак.
- CodeIgniter має простий і зрозумілий синтаксис, що дозволяє розробникам швидко створювати функціональний код.

Також додатково було використано Telegram API PHP для більш швидкого проектування інтерфейсу, та інших функцій які полегшують роботу.

Основна причина розробки на цій мові програмування була зумовлена тим, що вона дає змогу розгорнути API або веб-додаток без значних затруднень. Крім того, в майбутньому планується розвивати проект, та створити веб-сайт, який також буде написаний на PHP і завдяки цьому, буде можливо інтегрувати бота без всіляких труднощів.

Також потрібно підібрати ресурс, з якого будуть підгружатись усі фільми та серіали до нашого боту за допомогою API ключа. Вибір пав на сервіс перегляду кінофільмів «Кинопоиск». Так як на будь яких інших онлайн кінотеатрах та інформаційних сайтах про кіно не було передбаченого API для вивантаження даних, було прийнято рішення використовувати саме цей ресурс. Щоб отримати його, достатньо зробити пошуковий запит в Google, або перейти до спеціально створеного Telegram боту @ kinopoiskdev_bot в якому можна обрати тариф для отримання API ключа.

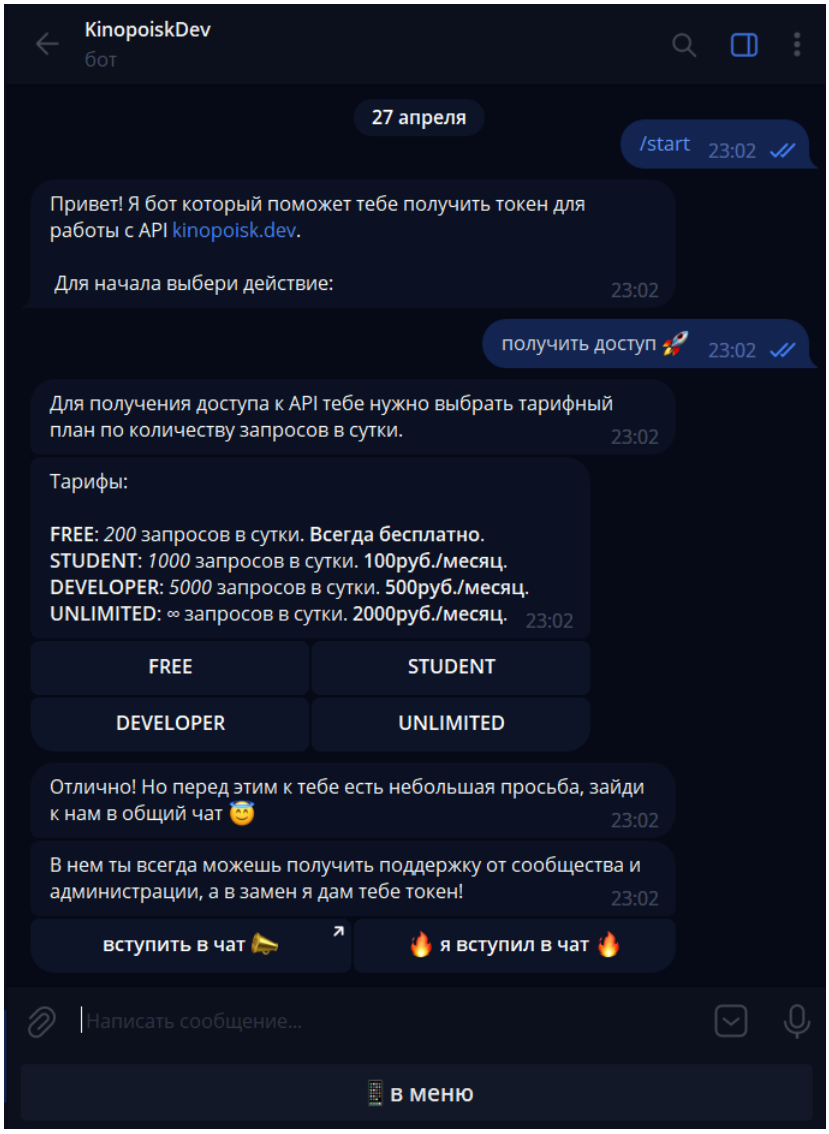

Рис. 2.1 – Процес отримання API ключа за допомогою Telegram боту від веб-сайту «Кинопоиск»

В якості бази даних будемо використовувати MySQL – система управління базами даних (СУБД). MySQL підтримує мову програмування SQL (Structured Query Language), яка була створе спеціально для роботи з базами даних. Взагалі, SQL – це універсальна мова, яка підтримує всі системи управління базами даних, тому, знаючи її, ви зможете працювати з будь-якими СУБД.

MySQL є надійним і безкоштовним, з відкритим кодом. Програма використовує властивості ACID (atomicity – атомарність, consistency – узгодженість, isolation – ізольованість, durability – надійність) для забезпечення цілісності транзакцій між кількома механізмами зберігання. Завдяки цьому розробники можуть легко створювати високо масштабовані веб-додатки баз даних з використанням інтерфейсу SQL, не маючи особливих знань про базову архітектуру системи чи механізми зберігання.

MySQL є однією з найбільш популярних систем баз даних, яку використовують сьогодні, і має велику та активну спільноту користувачів. Це означає, що доступна велика кількість інформації щодо використання MySQL для розробки як веб-, так і хмарних програм. З цих причин MySQL є чудовим вибором для розробки додатків баз даних.

Ще декілька переваг СУБД MySQL:

- Команда підтримки MySQL гарантує високоякісне обслуговування клієнтів за допомогою електронної пошти або онлайн-чату, доступного 24 години на добу, 365 днів на рік. Більше того, на їх веб-сайті є велика база знань, яка містить інформацію про різні теми, пов'язані з програмним забезпеченням та специфічні помилки, які можуть виникнути під час встановлення.
- MySQL має багато корисних функцій, таких як реплікація, підзапити та тригери, які не зустрічаються в інших програмних забезпеченнях для баз даних.
- Для роботи MySQL не потрібно багато системних ресурсів.
- Ще одна важлива функція MySQL полягає в можливості створення та зміни представлення, яка є унікальною для цієї бази даних і не зустрічається в інших програмах.

Вибір середовища розробки. Насправді писати програмний код можна в будь-якому текстовому редакторі. Однак, більшість інтегрованих середовищ розробки (Integrated Drive Electronics або IDE) включають в себе

набагато більше функцій ніж звичайний текстовий редактор. Нижче будуть наведені деякі переваги, завдяки яким розробники обирають саме їх:

- IDE містить набір інструментів, які допомагають розробникам писати код швидше та ефективніше. Наприклад, IDE може надавати автодоповнення коду, підсвічувати ключові слова, підсвічування помилок, також автоматично викликає компілятор і запускає програму при натисненні певної кнопки, або комбінацію клавіш.
- Є підтримка великої кількості додаткових інструментів: системи збирання, системи багтрекінга, системи месседжингу, систему управління контролем версій, тощо.
- Дозволяє розробникам працювати з однаковими інструментами та налаштуваннями, що допомагає уникнути проблем з сумісністю коду між різними розробниками.
- Допомагає організувати проект та структурувати код, що дозволяє розробникам легше орієнтуватися у проекті та швидше знайти необхідну ділянку коду.
- Дозволяє використовувати різні мови програмування. Більшість IDE підтримують різні мови програмування, що дозволяє розробникам працювати з різними мовами в одному інтерфейсі.
- Може бути корисною для розробників, які працюють у команді, оскільки вона дозволяє зберігати та керувати версіями коду, а також дозволяє легко ділитися проектом та взаємодіяти з іншими розробниками.

Для написання коду буде використовуватись PhpStorm - комерційна крос-платформне інтегроване середовище розробки для PHP. PhpStorm являє собою інтелектуальний редактор з можливостями аналізу коду на льоту, запобігання помилкам у коді та автоматизованими засобами рефакторингу для PHP.

PhpStorm має багатий набір функцій, таких як підсвічування коду, розширене форматування, перевірка на помилки під час написання коду та автодоповнення. PhpStorm підтримує багато версій PHP та їх

функціональність, включаючи нові можливості, такі як простір імен, замикання та синтаксис коротких масивів. Інструмент також має підтримку стандартів оформлення коду та PHPDoc. Крім того, PhpStorm має ряд інших корисних функцій, таких як детектор дублювання коду, підтримка редагування шаблонів Smarty та MVC-представлення для Symfony2 та Yii фреймворків.

3 РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ РЕКОМЕНДАЦІЙ КІНОФІЛЬМІВ

3.1 Підготовка до початку роботи

Для початку потрібно встановити інструменти які потрібні для розробки, або ж просто полегшить роботу.

Спершу встановимо сам Telegram та зареєструємось в ньому. Зробити це дуже просто, знадобиться лише номер телефону, на який прийде код підтвердження після вводу його в спеціальне поле. Перед цим же, потрібно вибрати країну, а після вводу перевірочного коду ввести ім’я та фамілію. Готово, аккаунт створено.

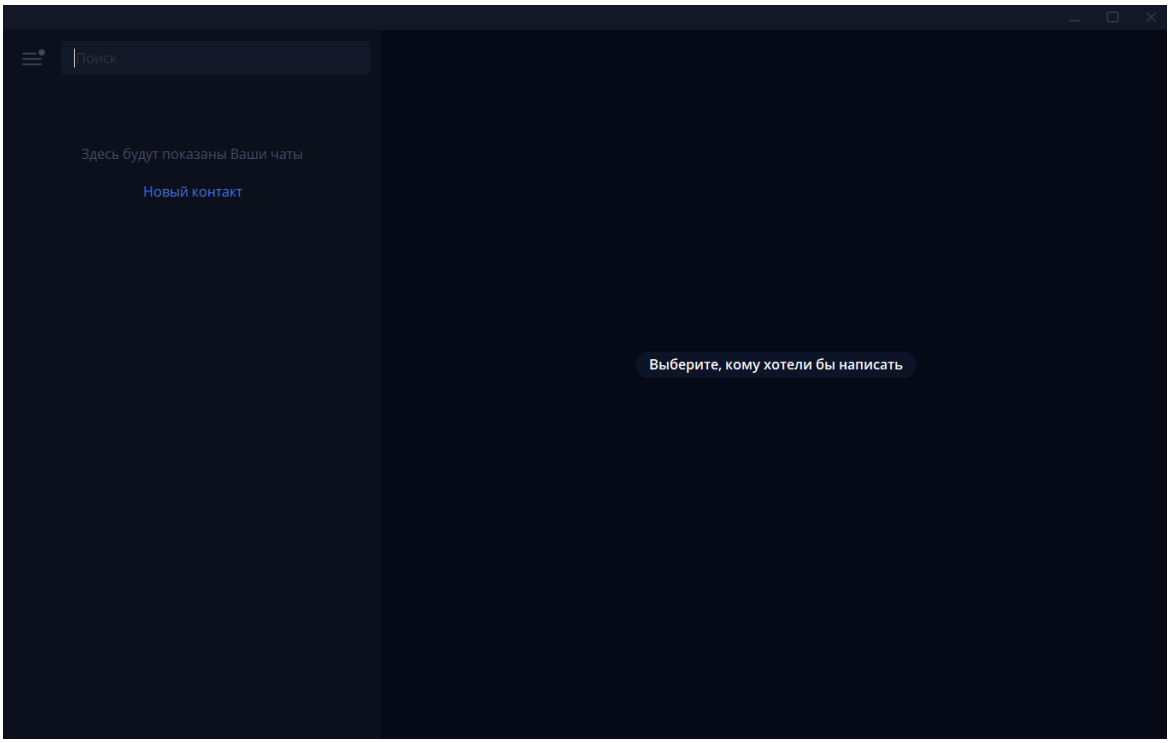

Рис. 3.1 – Інтерфейс нового акаунту в Telegram

Для встановлення CodeIgniter 3 достатньо перейти на сайт розробника, завантажити архів, після чого розпакувати його в зручне місце, та завантажити файли і папки CodeIgniter 3 на свій сервер.

3.1.1 Створення боту в Telegram

Щоб почати розробку, потрібно створити нового бота. Для цього потрібно знайти спеціального бота в Telegram, який називається BotFather. Це офіційний бот в телеграмі, який дозволяє створювати та налаштовувати власних ботів в месенджері.

Процес створення нового, свого власного боту дуже простий. Спочатку потрібно ввести команду /newbot, після чого придумати назву для боту. Назва може містити 64 символи. Однак рекомендується використовувати короткі назви, щоб вони краще поміщалися в повідомленнях користувачів.

Наступний крок – ввести ім’я користувача, тобто те ім’я, за яким можна буде знайти нашого бота. Username може містити 32 символи і повинен закінчуватись позначенням «bot», наприклад test_bot або testBot.

Бот створений, BotFather надсилає повідомлення в якому міститься посилання на щойно створеного бота, а також токен. Токен – це унікальний ключ доступу, ідентифікатор вашого боту, для взаємодії з Telegram API, який дозволяє виконувати різні дії в месенджері. Його важливо зберігати в таємниці і не передавати третім особам. Якщо хтось отримає доступ до токену, то зможе використовувати його для здійснення дій від імені боту, включаючи відправку спаму, підміну повідомлень та інші шкідливі дії. Тому дуже важливо зберігати токен телеграм боту в таємниці, щоб забезпечити безпеку бота та його користувачів. Крім того, рекомендується зберігати токен в безпечному місці, наприклад, у зашифрованому файлі або у змінній середовища, щоб запобігти його витоку. Далі наведений приклад самого токену:

123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11

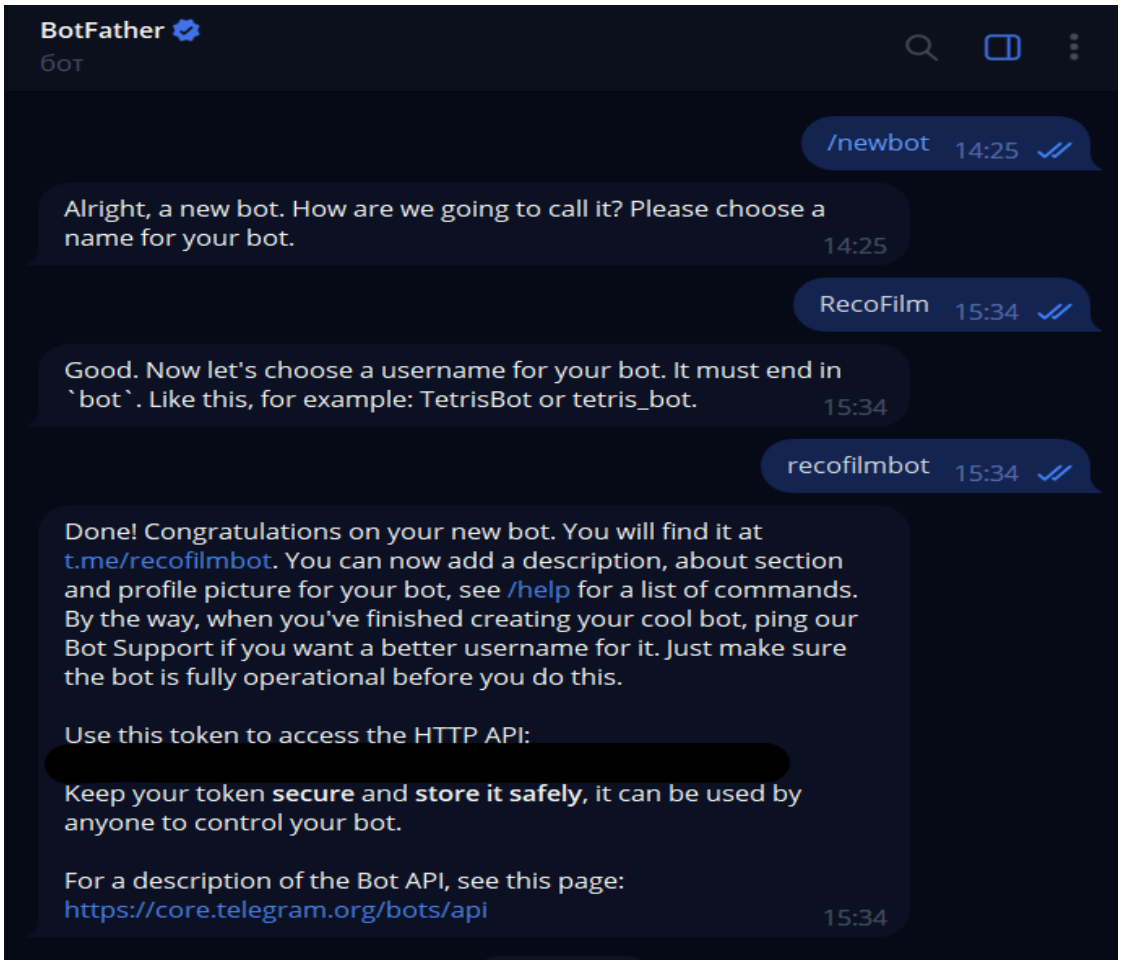

Рис. 3.2 – Процес створення нового боту через офіційний Telegram бот – BotFather

3.1.2 Розробка інтерфейсу користувача

Як вже зазначалося, інтерфейс програми – одна з найважливіших її частин, тому до його розробки слід підходити обережно, важливо не перенавантажувати інтерфейс непотрібними кнопками та іншими елементами.

Було вирішено створити привітання, яке з’являється після запуску боту, а також 4 кнопки в основному меню (клавіатура користувача). Це пошук за запитом, популярне, обране, а також профіль та інфо. Текст прикрашений підходящими по змісту смайликами для кращого візуального сприйняття.

Нижче наведений приклад коду, для генерації першої кнопки в основному меню.

```
$telegram->sendMessage([
'chat_id' => $chat_id,
```

'reply_markup' => $telegram->buildInlineKeyBoard([
[$telegram->buildInlineKeyboardButton(' Пошук за запитом')],
'text' => ' Виберіть параметер пошуку:'

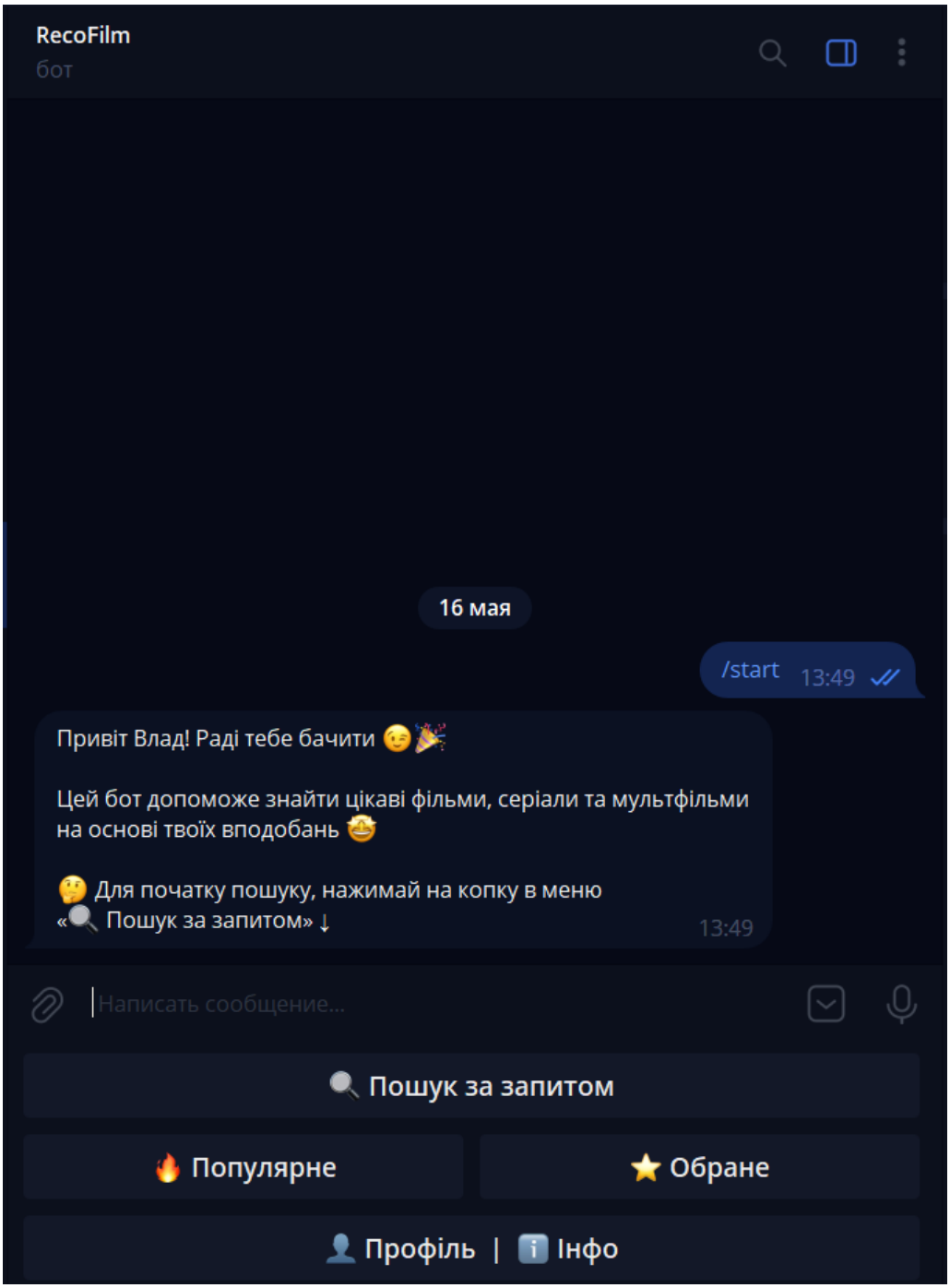

Рис. 3.3 – Інтерфейс користувача (привітання та основне меню)

Окрім клавіатури користувача, існують також callback- та URL-кнопки. Callback-кнопки дозволяють реалізувати навігацію по боту, наприклад,

перелистування сторінки, або перехід до сторінки з інформацією про сам фільм.

Нижче наведений приклад створення callback-кнопок навігації в списку знайдених фільмів

$telegram->buildInlineKeyboardButton((string) ($key + 1), '',

"callback_get_id_chunk_item_$id_search-$current_page-$key"));

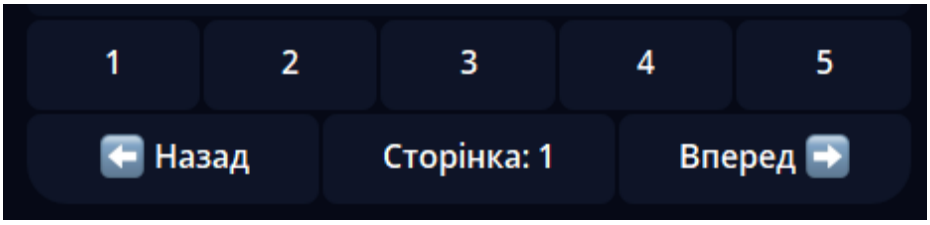

Рис. 3.4 – Інтерфейс користувача (кнопки навігації)

URL-кнопки в свою чергу, перенаправляють користувача на сторонній ресурс при натисканні. В нашому випадку, на ресурс для перегляду фільмів HDRezka.

[$telegram->buildInlineKeyboardButton(' Подивитись',

'https://rezka.ag/search/?do=search&subaction=search&q=' .

urlencode($name_film), '')]

При пошуку по назві, ключовому слову, жанру, року виходу, підбірки популярних, схожих кінокартин, а також в списку обраного, користувачу видається список знайдених фільмів з такими параметрами:

- Цифра, з номером якої потрібно натиснути кнопку, аби перейти до сторінки інформації про фільм
- Назва та рік виходу
- Рейтинг по «Кинопоиску»
- Жанр
- Країна виходу
- Тривалість фільму

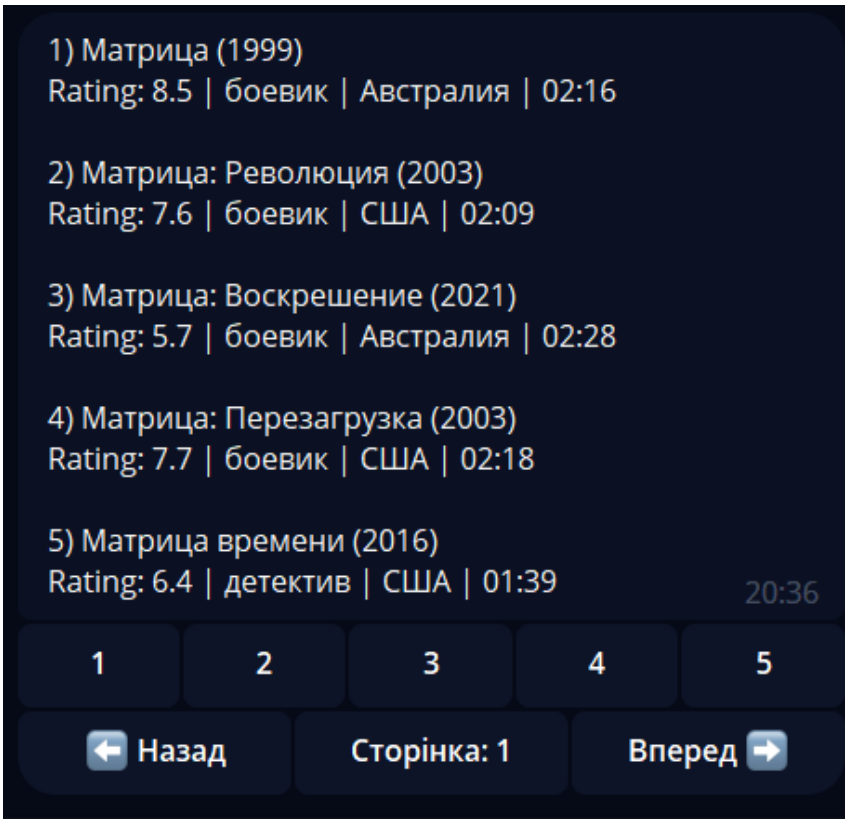

Рис. 3.5 – Інтерфейс користувача (результат пошуку «Матриця»)

Інформаційна сторінка про фільм має наступні дані:

- Обкладинка до фільму
- Назву та рік виходу
- Тривалість фільму
- Рейтинг
- Країна виходу
- Режисер
- Актори
- Опис до фільму

Рис. 3.6 – Інтерфейс користувача (інформаційна сторінка фільму «Матриця»)

Крім цього, в наявності є ще декілька кнопок:

- Подивитись фільм
- Пошук схожих фільмів

- Додати в обране
- Повернутись до списку фільмів

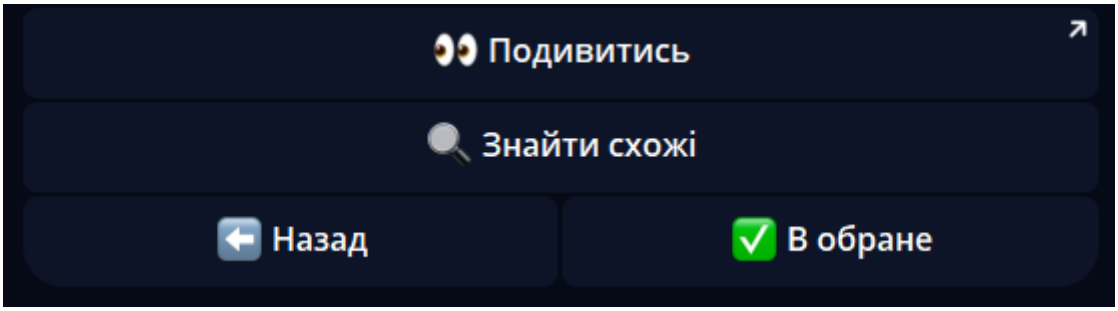

Рис. 3.6 – Інтерфейс користувача (кнопки на інформаційній сторінці фільму)

3.1.3 Розробка основної частини

Основне призначення програми яка розробляється – рекомендування кінофільмів.

В загальному вигляді, все працює наступним чином. Спочатку користувач боту буде надсилати запит з тим параметром пошуку, який вибере. Запит надсилається в контролер для обробки. Якщо необхідно, контролер підвантажує додаткову інформацію з контролеру bot model (бази даних). На основі отриманої інформації, контролер генерую меню і надсилає його http запитом в телеграм, де його отримує кінцевий користувач.

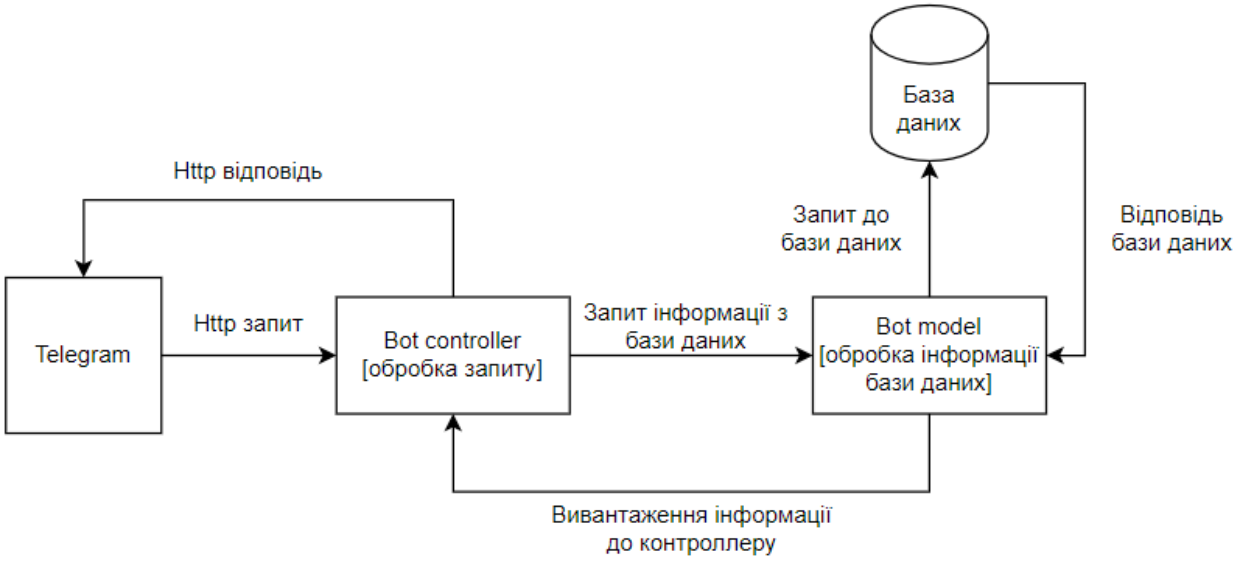

Рис. 3.5 – Загальна схема роботи Telegram боту

Пошук кінофільмів виконується завдяки контролеру. Користувач виконує пошуковий запит який передається в тіло контролера. Далі виконується перевірка на наявність яких-небуть даних, якщо їх немає, то нічого не відбувається. А якщо є, то на основі цих даних, надсилається запит до API «Кинопоиск», для подальшої обробки. Кінцевий результат, у вигляді JSON-масиву, знову отримується контролером, де починається нова перевірка на наявність відповіді. У випадку відсутності відповіді, бот надсилає повідомлення для користувача про те, що немає результатів по пошуковому запиту. В іншому випадку, робота продовжується. Починається перерахунок результатів масиву, а саме сторінок з результатами пошуку. За замовчуванням, від API надається відповідь на одну сторінку з двадцятьма фільмами, а також кількість сторінок загалом. Якщо у загальній відповіді більше ніж одна, тоді всі сторінці підвантажуються у циклі, та додаються до JSON-масиву. Кінцевий JSON-масив вставляється в базу даних, для подальшої обробки в майбутньому. Після чого, на основі масиву генерується меню, та відправляється користувачу.

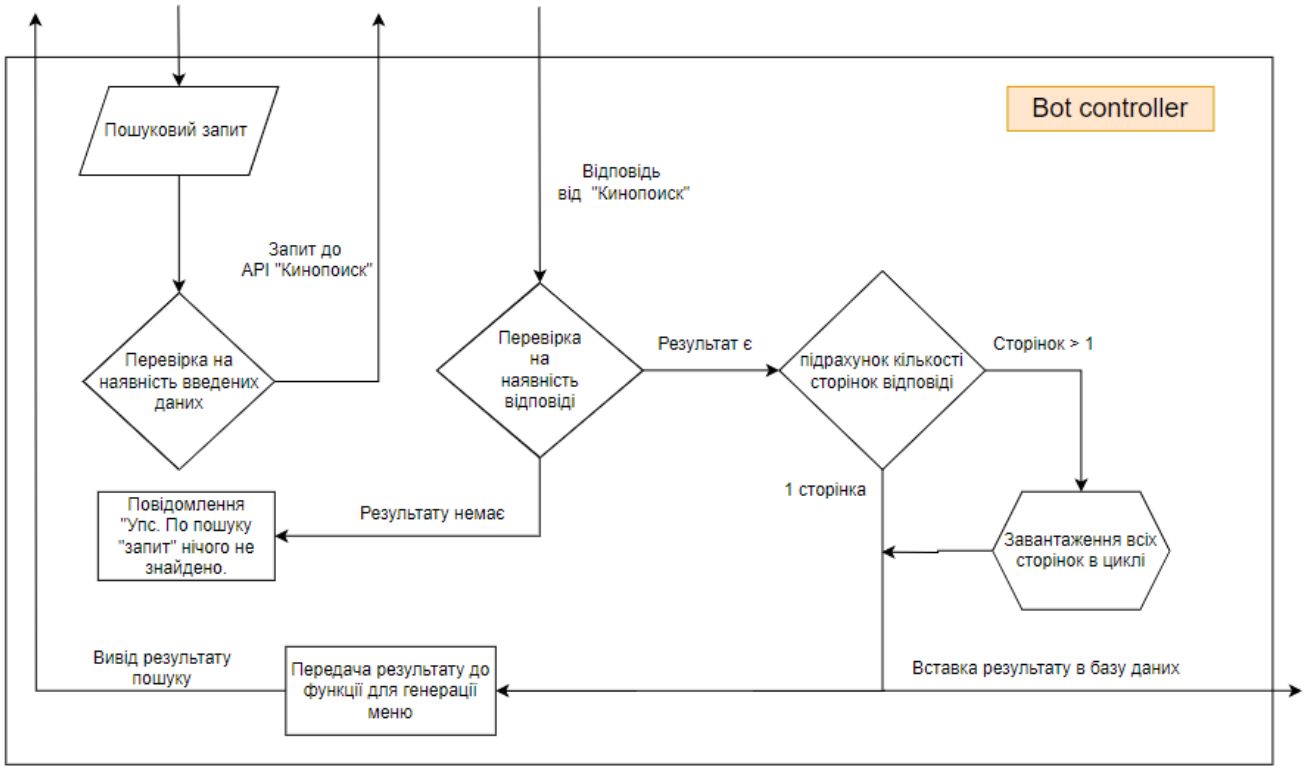

Рис. 3.6 – Алгоритм пошуку кінофільмів

Сам же пошук схожих фільмів працює схожим чином. Замість пошукового запиту в контролер бота, передається callback (зворотній виклик), в якому є інформація, що це саме запит на пошук подібних фільмів, а також id фільму на основі якого буде здійснюватися пошук схожих фільмів. Далі знову йде запит до API «Кинопоиск», обробка запиту, і повертається результат який починає перевірятися контролером на наявність відповіді. У разі її відсутності, бот використовує альтернативні шляхи пошуку схожого контенту. А саме по жанру та ключовому слову. Кінцевий результат має вигляд id фільмів, які потребують підвантаження додаткової інформації через цикл. В кінці, як і при пошуку фільмів, виконується вставка JSON-масиву в базу даних, для подальшої обробки в майбутньому. Після чого, на основі масиву генерується меню, та відправляється користувачу.

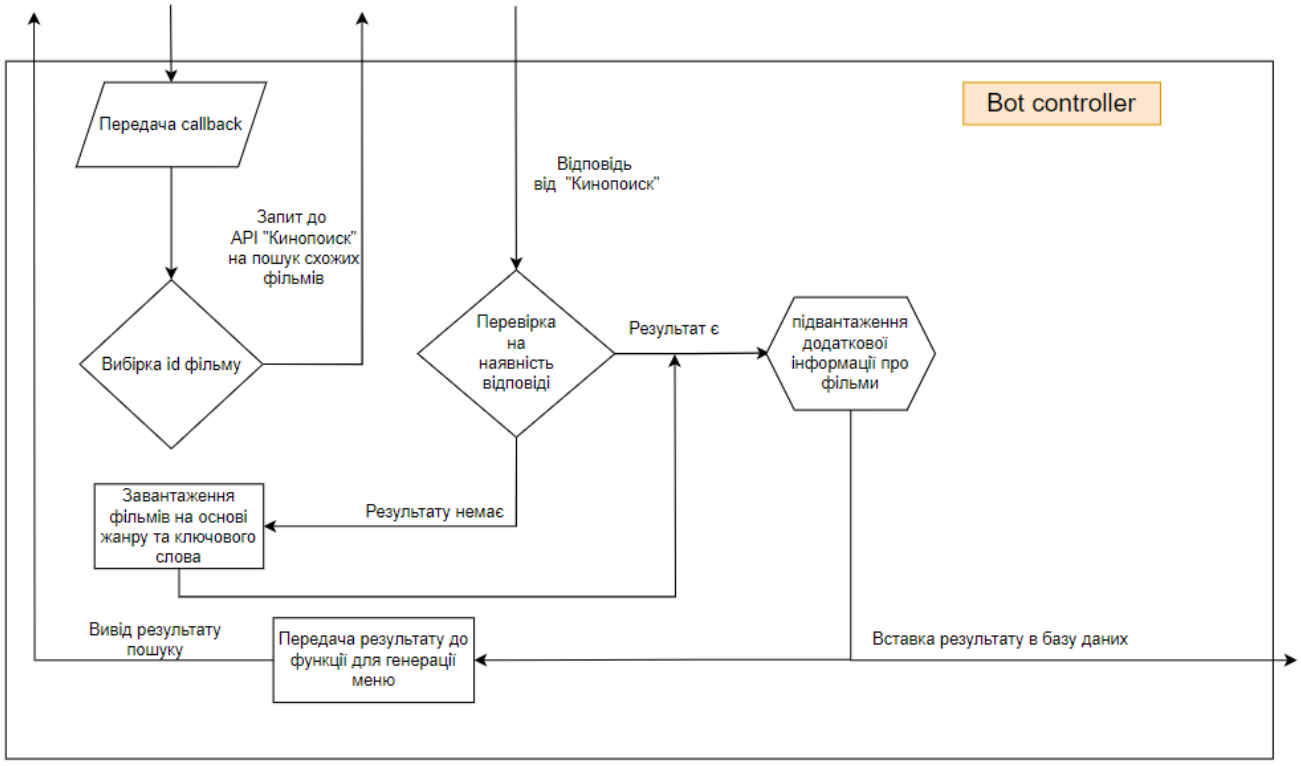

Рис. 3.7 – Алгоритм пошуку схожих кінофільмів

Для роботи боту в автономному режимі, необхідне середовище, в якому бот буде працювати постійно. Для цього потрібен PHP-хостинг, або VPS сервер, на який буде встановлений сам бот.

3.2 Результат тестування програми

Після завершення роботи над ботом, потрібно перевірити його працездатність в реальних умовах.

Для початку роботи потрібно прописати команду /start, після чого з’явиться привітання та головне меню.

Обираючи «пошук за запитом», програма попросить обрати, що саме ви хочете знайти: Фільм по назві, по року виходу або по жанру.

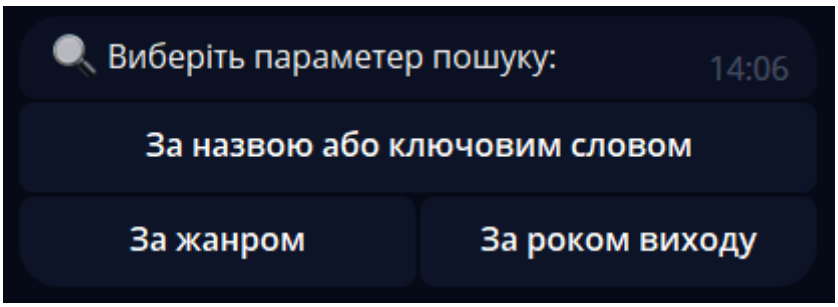

Рис. 3.8 – Вибір параметру пошуку

Для прикладу, оберемо параметр «за роком виходу». Бот попросить ввести рік виходу, фільми якого ми хочемо побачити, напишемо 2023.

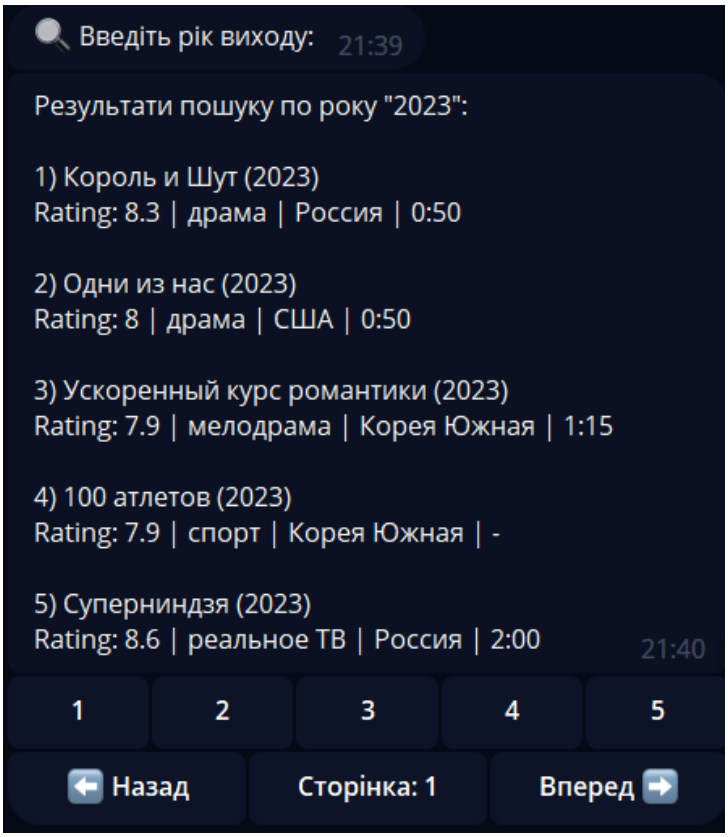

Рис. 3.9 – Результат пошуку за роком виходу, по запиту «2023»

Як бачимо, в списку фільмів є лише ті картини, які вийшли в 2023 році, а отже, програма працює правильно.

Для прикладу пошуку схожих, оберемо фільм «Матриця». Після натискання кнопки «знайти схожі», програма повертає наступний результат:

Рис. 3.10 – Результат пошуку схожих фільмів на «Матриця»

Тестування авторекомендацій. Для початку потрібно увімкнути функцію авторекомендацій в меню – профіль та інфо – налаштувати авторекомендації, адже за замовчуванням вона вимкнена. Потім потрібно дочекатися 19:00 за київським часом і бот автоматично надішле повідомлення з підборкою фільмів та серіалів, основуючись на списку обраного конкретного користувача. Також для коректної роботи авторекомендацій, в обраному потрібно мати не менше трьох кінофільмів. Наприклад, в даному випадку, в списку обраного є наступні картини: Шрек(2001), Кунг-фу Панда(2008) та Пінгвіни мадагаскару (2014), результат наступний:

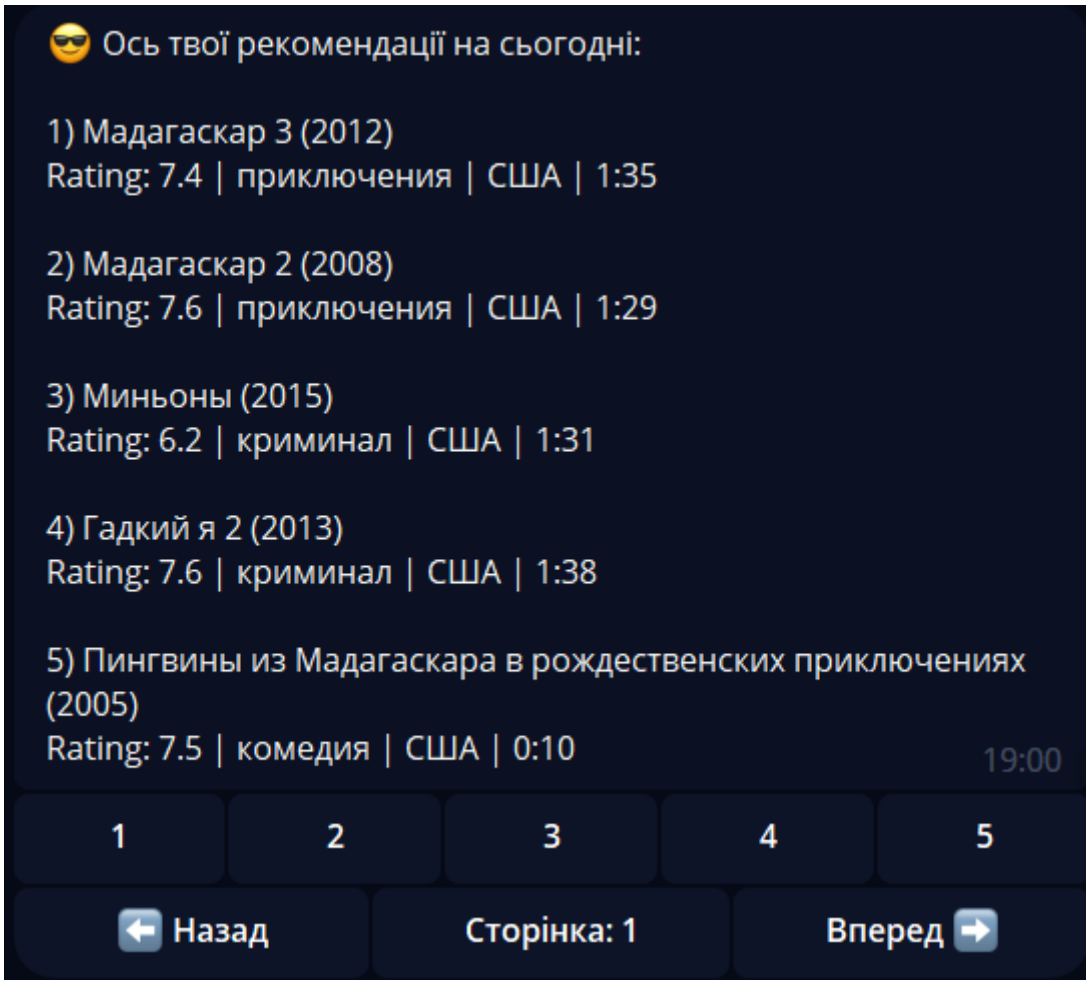

Рис. 3.11 – Результат роботи авторекомендацій

Загалом, програма працює досить швидко, знаходить фільми та серіали за різними запитами, а також допомагає знайти схожі картини на вподобані користувачем, а це означає, що все працює так як треба.

Висновки

Під час написання дипломної роботи, були досліджені системи рекомендацій, їхні види та методи роботи. Визначені переваги та недоліки різних методів систем рекомендацій, їхні проблеми ті можливі способи рішення. Проведений аналіз предметного середовища. У результаті виявлено____

Проведено аналіз існуючих систем рекомендацій, описано як вони працюють, їхні недоліки та переваги. Найкращою системою виявилася_____ або не виявлено____тому запропоновано____

Основним результатом роботи стала розробка спеціальної програми у вигляді телеграм бота, яка дозволяє знайти схожі фільми, базуючись на вподобаннях користувача. Визначені основні вимоги до програм, інтерфейсу користувача. Створено зручний та легкий у використанні користувацький інтерфейс. Описано алгоритм пошуку кінофільмів за допомогою API «Кінопошук», алгоритм пошуку схожих фільмів.

Проведено тестування створеної програми у реальних умовах, щоб переконатись у працездатності боту, відповідності до загальних вимог та наявність проблем, які потрібно виправити.

Запропонована пошукова програма може застосовуватися____ та сприятиме ефективному пошуку_____

Перелік джерел посилань

1. Konstan JA, Riedl J. Рекомендаційні системи: від алгоритмів до користувацького досвіду. Модель користувача. Взаємодія з адаптацією користувача 2012;22:101–23.

2. Schafer JB, Konstan J, Riedl J. Рекомендаційні системи в e-commerce. In: Праці 1-ї ACM конференції з питань електронної комерції; 1999. p. 158–66.

3. Pan C, Li W. Рекомендація дослідницької роботи з аналізом тем. У галузі комп'ютерного дизайну та додатків IEEE 2010;4, pp. V4-264.

4. Pu P, Chen L, Hu R. Орієнтована на користувача система оцінки рекомендаційних систем. В: Матеріали п’ятої конференції ACM щодо рекомендаційних систем (RecSys’11), ACM, New York, NY, USA; 2011. p. 57–164.

5. Rashid AM, Albert I, Cosley D, Lam SK, McNee SM, Konstan JA et al. Знайомство з вами: вивчення нових уподобань користувачів у рекомендаційних системах. Матеріали міжнародної конференції з інтелектуальних користувальницьких інтерфейсів; 2002. p. 127–34.

6. Recommendation System [Електронний ресурс] – Режим доступу до ресурсу: https://www.nvidia.com/en-us/glossary/datascience/recommendation-system/

7. Recommendation systems and machine learning: driving personalization [Електронний ресурс] – Режим доступу до ресурсу: https://www.itransition.com/machine-learning/recommendation-systems

8. P. Lops, M. Gemmis, G. Semeraro Recommender Systems Handbook Springer-Verlag, 2011.

9. Xiaoyuan Su, Khoshgoftaar T. M. A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, 2009.

10. TF-IDF – Term Frequency-Inverse Document Frequency [Електронний ресурс] – Режим доступу до ресурсу: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency

11. Word2Vec Explained [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/word2vec-explained-49c52b4ccb71

12. Recommendations Figuring out how to bring unique joy to each member [Електронний ресурс] – Режим доступу до ресурсу: https://research.netflix.com/research-area/recommendations

13. The Netflix Prize — How Even AI Leaders Can Trip Up [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/the-netflix-prize-how-even-ai-leaders-can-trip-up-5c1f38e95c9f

14. D. Asanov, “Algorithms and Methods in Recommender Systems,” Other Conf., 2011.

15. Netflix’s Recommendation Systems: Entertainment Made for You [Електронний ресурс] – режим доступу: https://illumin.usc.edu/netflixsrecommendation-systems-entertainment-made-for-you/

16. Мелешко Є.В. Дослідження методів побудови рекомендаційних систем в мережі Інтернет / Є.В. Мелешко, Г.С. Семенов В.Д. Хох. // Збірник наукових праць "Системи управління, навігації та зв’язку". Випуск 1(47). – Полтава: ПНТУ ім. Ю. Кондратюка. – 2018. – С. 131-136

17. Moghaddam F., Elahi M. Cold-start solutions for recommendation systems – Bonn, Nordrhein-Westfalen, Germany. – 2019. 35 p.

18. CodeIgniter User Guide [Електронний ресурс] – режим доступу: https://codeigniter.com/userguide3/

19. Чалий С.Ф., Прібильнова І.Б. Ситуаційне представлення споживачів рекомендаційної системи. Матеріали дев’ятої міжнародної науково-технічної конференції. С.3

Додаток А
Лістинг програми

```
function get_film ($id_search, $current_page, $item, $telegram,
$kinopoisk_api_key) {
   //завантаження та обробка фільму на чанки з БД
   $get_search = $this->bot_model->get_search_by_id($id_search);
   $json = json_decode($get_search['json_answer'], true);
   $chunks = array_chunk($json, 5);
   $film = $chunks[$current_page - 1][$item];
   if ($film['filmId']) {
     $staff = $this->kinopoisk_request('/api/v1/staff?filmId=' . $film['filmId'],
$kinopoisk_api_key);
   } elseif ($film['kinopoiskId']) {
     $staff = $this->kinopoisk_request('/api/v1/staff?filmId=' .
$film['kinopoiskId'], $kinopoisk_api_key);
   }
   $default_directors = 2;
   $default_actors = 5;
   $count_directors = 0;
   $count_actors = 0;
   $directors = '';
   $actors = '';
   // перебір режиссерів та акторів
   foreach ($staff as $value) {
     if ($value['professionKey'] == 'DIRECTOR') {
       if ($count_directors < $default_directors) {
         $value['nameRu'] ? $name = $value['nameRu'] : $name =
$value['nameEn'];
         if ($count_directors == $default_directors) {
```

```
            $directors .= $name . '.';
          } else {
            $directors .= $name . ', ';
          }
          $count_directors++;
        }
      }
      if ($value['professionKey'] == 'ACTOR') {
        if ($count_actors < $default_actors) {
          $value['nameRu'] ? $name = $value['nameRu'] : $name =
$value['nameEn'];
          if ($count_actors == $default_actors) {
            $actors .= $name . '.';
          } else {
            $actors .= $name . ', ';
          }
          $count_actors++;
        }
      }
    }
    // перевірка та обробка додаткового тегу для відображення корректної
інформації
    if (strpos($get_search['keyword'], 'similar_id_') !== false or
$get_search['keyword'] == 'favorite' or strpos($get_search['keyword'], 'genre_')
!== false or strpos($get_search['keyword'], 'year_') !== false) {
      $film['nameRu'] ? $name_film = $film['nameRu'] : $name_film =
$film['nameEn'];
      $film['ratingKinopoisk'] != "null" ? $rate = $film['ratingKinopoisk'] : $rate =
'-';
        $film['genres'][0]['genre'] ?? $film['genres'][0]['genre'] = '-';
```

```
        $film['countries'][0]['country'] ?? $film['countries'][0]['country'] = '-';
      if ($film['filmLength']) {
        $length = $this->minutes_to_time($film['filmLength']);
      } else {
        $length = '-';
      }
    } else {
      $film['nameRu'] ? $name_film = $film['nameRu'] : $name_film =
$film['nameEn'];
      $film['rating'] != "null" ? $rate = $film['rating'] : $rate = '-';
        $film['genres'][0]['genre'] ?? $film['genres'][0]['genre'] = '-';
        $film['countries'][0]['country'] ?? $film['countries'][0]['country'] = '-';
      $film['filmLength'] ? $length = $film['filmLength'] : $length = '-';
    }
    if ($film['type'] == 'FILM') {
      $about = 'О фильме:';
    } elseif ($film['type'] == 'TV_SERIES') {
      $about = 'О сериале:';
    } else {
      $about = 'Описание:';
    }
    $film_info['text'] = "$name_film ({$film['year']})\n\nЖанр:
{$film['genres'][0]['genre']}\nВремя: $length\nРейтинг: $rate\nСтрана:
{$film['countries'][0]['country']}\nРежиссер: $directors\nВ ролях:
$actors\n\n$about\n{$film['description']}";
    $film_info['photo'] = $film['posterUrl'];
    $get_profile_favorite_films = $this->bot_model-
>get_favorite_films_by_tg_id($telegram->UserID());
    $favorite_films = json_decode($get_profile_favorite_films, true);
    $key = array_search($film['filmId'], $favorite_films);
```

```
    // генерація inline меню
    if (!empty($key)) {
      $film_info['buttons'] = $telegram->buildInlineKeyBoard([
        [$telegram->buildInlineKeyboardButton(' Подивитись',
'https://rezka.ag/search/?do=search&subaction=search&q=' .
urlencode($name_film), "")],
        [$telegram->buildInlineKeyboardButton(' Знайти схожі', ",
"callback_get_similar_films_by_id_{$film['filmId']}")],
[$telegram->buildInlineKeyboardButton(' Назад', ",
"callback_get_chunk2_$id_search-$current_page"), $telegram-
>buildInlineKeyboardButton(' Видалити з обраного', ",
"callback_delete_favorite_film_to_profile_{$film['filmId']}")],
      ]);
    } else {
      $film_info['buttons'] = $telegram->buildInlineKeyBoard([
        [$telegram->buildInlineKeyboardButton(' Подивитись',
'https://rezka.ag/search/?do=search&subaction=search&q=' .
urlencode($name_film), "")],
        [$telegram->buildInlineKeyboardButton(' Знайти схожі', ",
"callback_get_similar_films_by_id_{$film['filmId']}")],
        [$telegram->buildInlineKeyboardButton(' Назад', ",
"callback_get_chunk2_$id_search-$current_page"), $telegram-
>buildInlineKeyboardButton(' В обране', ",
"callback_set_favorite_film_to_profile_{$film['filmId']}")],
      ]);
    }
    // вставка чекпоінту і оновлення інформації профілю
    $this->bot_model->update_checkpoint_by_tg_id($telegram->UserID(),
json_encode($film_info));
    $stats = $this->bot_model->get_stats_by_tg_id($telegram->UserID());
```

```
    $stats = json_decode($stats, true);
    if (!empty($stats['films_and_count_view'][$name_film])) {
      $stats['films_and_count_view'][$name_film]++;
    } elseif ($stats['films_and_count_view'] == false) {
      $stats['films_and_count_view'] = [$name_film => 1];
    } else {
      $stats['films_and_count_view'] =
array_merge($stats['films_and_count_view'], [$name_film => 1]);
    }
    if (!empty($stats['genres'][$film['genres'][0]['genre']])) {
      $stats['genres'][$film['genres'][0]['genre']]++;
    } elseif ($stats['genres'] == false) {
      $stats['genres'] = [$film['genres'][0]['genre'] => 1];
    } else {
      $stats['genres'] = array_merge($stats['genres'], [$film['genres'][0]['genre'] =>
1]);
    }
    if (!empty($stats['countries'][$film['countries'][0]['country']])) {
      $stats['countries'][$film['countries'][0]['country']]++;
    } elseif ($stats['countries'] == false) {
      $stats['countries'] = [$film['countries'][0]['country'] => 1];
    } else {
      $stats['countries'] = array_merge($stats['countries'],
[$film['countries'][0]['country'] => 1]);
    }
    $stats['total_viewed']++;
    $stats['last_viewed'] = $name_film;
    $this->bot_model->update_stats_by_tg_id($telegram->UserID(),
json_encode($stats));
    return $film_info;  }
```

```
// функція search_menu отримує масив фільмів і генерує текстове представлення та меню кнопок для подальшої обробки та відправки користувачу
  public function search_menu($arr, $current_page, $id_search, $text_search, $telegram, $kinopoisk_api_key) {
    // розшифрування JSON масиву та розбивка на чанки
    $json = json_decode($arr, true);
    $chunks = array_chunk($json, 5);
    $max_page = count($chunks);
    $left = $current_page - 1;
    if ($left != true) $left = $max_page;
    $right = $current_page + 1;
    if ($right > $max_page) $right = 1;
    $pages = [];
    $result = [];
    // перевірка додаткових тегів для вставки додаткової інформації
    if (strpos($text_search, 'similar_id_') !== false or strpos($text_search, 'auto_recommendations_') !== false or $text_search == 'favorite' or strpos($text_search, 'genre_') !== false or strpos($text_search, 'year_') !== false) {
      if (strpos($text_search, 'similar_id_') !== false) {
        $id = substr($text_search, 11);
        $film = $this->kinopoisk_request('/api/v2.2/films/' . $id, $kinopoisk_api_key);
        $film['nameRu'] ? $name_film = $film['nameRu'] : $name_film = $film['nameEn'];
        $result['search_chunk_result'] .= 'Результати пошуку схожі на ' . $name_film . "\n\n";
      } elseif ($text_search == 'favorite') {
        $result['search_chunk_result'] .= 'Список обраного:' . "\n\n";
```

```
    } elseif (strpos($text_search, 'auto_recommendations_') !== false) {
      $result['search_chunk_result'] .= ' Ось твої рекомендації на сьогодні:' .
"\n\n";
    } elseif (strpos($text_search, 'genre_') !== false) {
      $id = substr($text_search, 6);
      $id_and_genres_to_search = $this-
>kinopoisk_request("/api/v2.2/films/filters", $kinopoisk_api_key);
      $result['search_chunk_result'] .= "Результати пошуку по жанру
\"{$id_and_genres_to_search['genres'][$id]['genre']}\":" . "\n\n";
    } elseif (strpos($text_search, 'year_') !== false) {
      $year = substr($text_search, 5);
      $result['search_chunk_result'] .= "Результати пошуку по року
\"{$year}\":" . "\n\n";
    }
    foreach ($chunks[$current_page - 1] as $key => $value) {
      $value['nameRu'] ? $name_film = $value['nameRu'] : $name_film =
$value['nameEn'];
      $value['ratingKinopoisk'] != "null" ? $rate = $value['ratingKinopoisk'] :
$rate = '-';
        $value['genres'][0]['genre'] ?? $value['genres'][0]['genre'] = '-';
        $value['countries'][0]['country'] ?? $value['countries'][0]['country'] = '-';
      if ($value['filmLength']) {
        $length = $this->minutes_to_time($value['filmLength']);
      } else {
        $length = '-';
      }
      $result['search_chunk_result'] .= $key + 1 . ") $name_film
({$value['year']})\nRating: $rate | {$value['genres'][0]['genre']} |
{$value['countries'][0]['country']} | $length\n\n";
```

```
            array_push($pages, $telegram->buildInlineKeyboardButton((string) ($key +
        1), '', "callback_get_id_chunk_item_$id_search-$current_page-$key"));
            }
        } else {
            foreach ($chunks[$current_page - 1] as $key => $value) {
            $value['nameRu'] ? $name_film = $value['nameRu'] : $name_film =
        $value['nameEn'];
            $value['rating'] != "null" ? $rate = $value['rating'] : $rate = '-';
              $value['genres'][0]['genre'] ?? $value['genres'][0]['genre'] = '-';
              $value['countries'][0]['country'] ?? $value['countries'][0]['country'] = '-';
              $value['filmLength'] ?? $value['filmLength'] = '-';
            $result['search_chunk_result'] .= $key + 1 . ") $name_film
        ({$value['year']})\nRating: $rate | {$value['genres'][0]['genre']} |
        {$value['countries'][0]['country']} | {$value['filmLength']}\n\n";
            array_push($pages, $telegram->buildInlineKeyboardButton((string) ($key +
        1), '', "callback_get_id_chunk_item_$id_search-$current_page-$key"));
            }
        }
        // генерація меню
        $result['arr_btn'] = $telegram->buildInlineKeyBoard([
          $pages,
[$telegram->buildInlineKeyboardButton(' Назад', '', "callback_get_chunk_$id_search-
$left"), $telegram->buildInlineKeyboardButton("Сторінка: $current_page", '',
"callback_get_chunk_$id_search-1"), $telegram->buildInlineKeyboardButton('Вперед
', '', "callback_get_chunk_$id_search-$right")],
        ]);
        return $result;
    }
```

```
if (strpos($telegram->Callback_Data(), 'callback_get_similar_films_by_id_') !== false) {
    // вибірка id фільму на основі якого буде здійснений пошук, запит до API на пошук схожих фільмів
    $film_id = substr($telegram->Callback_Data(), 33);
    $original_film = $this->kinopoisk_request('/api/v2.2/films/' . $film_id, $kinopoisk_api_key);
    $items = $this->kinopoisk_request("/api/v2.2/films/$film_id/similars", $kinopoisk_api_key);
    // перевірка на наявність схожих фільмів, якщо загальний об'єм 0, запит змінюється на пошук по жанру та ключовому слову
    if ($items['total'] == 0) {
      $id_and_genres_to_search = $this->kinopoisk_request("/api/v2.2/films/filters", $kinopoisk_api_key);
      foreach ($id_and_genres_to_search['genres'] as $value) {
        if ($value['genre'] == $original_film['genres'][0]['genre']) {
          $id_genre = $value['id'];
        }
      }
      $original_film['nameRu'] ? $name_film = $original_film['nameRu'] : $name_film = $original_film['nameEn'];
      $name_film = explode(' ', $name_film);
      $name_film = urlencode($name_film[array_rand($name_film, 1)]);
      $items = $this->kinopoisk_request("/api/v2.2/films?genres=$id_genre&keyword=$name_film", $kinopoisk_api_key);
    }
    $arr = [];
    $count_similars = 0;
    // завантаження інформації про фільми та вставка в БД
```

```
        foreach ($items['items'] as $item) {
          $item['filmId'] ? $id_film = $item['filmId'] : $id_film =
$item['kinopoiskId'];
          $film = $this->kinopoisk_request('/api/v2.2/films/' . $id_film,
$kinopoisk_api_key);
          if ($count_similars == 15) break;
          array_push($arr, $film);
          $count_similars++;
        }
        $json_array_films = json_encode($arr);
        $current_chunk = 1;
        $id = $this->bot_model->insert_search($telegram->UserID(),
"similar_id_$film_id", $json_array_films);
        // генерація меню функцією search_menu для відправки користувачу
        $search_chunk_result = $this->search_menu($json_array_films,
$current_chunk, $id, "similar_id_$film_id", $telegram, $kinopoisk_api_key);
        $telegram->answerCallbackQuery([
          'callback_query_id' => $telegram->Callback_ID()
        ]);
        $telegram->deleteMessage([
          'chat_id' => $chat_id,
          'message_id' => $telegram->MessageID()
        ]);
        // відправка результату користувачу
        $telegram->sendMessage([
          'chat_id' => $chat_id,
          'reply_markup' => $search_chunk_result['arr_btn'],
          'disable_web_page_preview' => 1,
          'text' => $search_chunk_result['search_chunk_result']
        ]); }
```