МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»

Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА

за темою

«Розробка та створення сайту кінотеатру»

Студент 4 курсу групи КН-20К

Ягода Даніла Владиславович

(прізвище, ім'я та по-батькові студента)

(підпис студента)

10.06.2024

(дата)

Керівник дипломної роботи

кандидат фіз.-мат. наук

(посада керівника)

Кириченко Віктор Вікторович

(прізвище, ім'я та по-батькові керівника)

До захисту

(резолюція «До захисту»)

(підпис викладача)

Київ, 2024 рік

СКОРОЧЕННЯ

Django - це веб-фреймворк на мові програмування Python, призначений для швидкої розробки веб-додатків.

API - інтерфейс програмування застосунків. Це набір правил та механізмів, які дозволяють різним програмам взаємодіяти одна з одною.

ORM - об'єктно-реляційне відображення. Це технологія, яка дозволяє використовувати об'єктно-орієнтований підхід до роботи з базами даних.

CRUD - створення, читання, оновлення та видалення. Це основні операції, які можна виконувати з даними в базі даних.

URL - рядок символів, що вказує на місцезнаходження ресурсу в Інтернеті або в мережі.

HTML - мова розмітки гіпертексту. Використовується для створення структури веб-сторінок.

CSS - каскадні таблиці стилів. Використовується для визначення зовнішнього вигляду веб-сторінок.

JavaScript - мова програмування, яка використовується для створення інтерактивних елементів на веб-сторінках.

SQL - мова структурованих запитів. Використовується для взаємодії з базами даних.

JWT - маркер доступу JSON. Використовується для автентифікації та передачі даних між сторонами у вигляді JSON-об'єктів.

API-ключ - унікальний ідентифікатор, який використовується для доступу до веб-служби або API.

ORM-модель - це клас, який представляє таблицю бази даних у Django ORM.

CRUD-операції - основні операції, які можна виконувати з даними в системі: створення, читання, оновлення та видалення.

URL-адреса - адреса в Інтернеті, яка вказує на конкретний ресурс, наприклад, веб-сторінку або зображення.

HTML-сторінка сторінка в Інтернеті, створена з використанням мови розмітки HTML.

CSS-стилі - файл або фрагмент коду, який визначає зовнішній вигляд елементів HTML на сторінці.

JavaScript-скрипт - код, написаний мовою програмування JavaScript, який виконується на сторінці веб-браузера.

SQL-запит - запит, написаний мовою структурованих запитів, який виконується на базі даних.

JWT-маркер - спосіб автентифікації та передачі даних між сторонами у вигляді JSON-об'єктів.

API-ключ - унікальний ідентифікатор, який використовується для доступу до веб-служби або API.

ЗМІСТ

ВСТУП

У сучасному цифровому світі розвиток інтернет-технологій розширює можливості розваг та розваг. Однією з найбільш перспективних та захоплюючих областей є онлайн-розваги, зокрема веб-кінотеатри. Створення веб-кінотеатру - це захоплива та складна задача, що передбачає інтеграцію різноманітних технологій, вивчення ринкових тенденцій та врахування потреб сучасної аудиторії.

Ця дипломна робота спрямована на розгляд процесу створення та розвитку веб-кінотеатру як важливого аспекту сучасної інтернет-розваги. Досліджуючи цю тему, ми прагнемо проаналізувати стратегії розробки та впровадження веб-кінотеатру, визначити ключові технологічні рішення, які впливають на його функціональність та взаємодію з користувачами, а також дослідити практичні аспекти управління контентом та забезпечення безпеки веб-платформи.

Основні питання, які будуть висвітлені в цій дипломній роботі, включають в себе аналіз ринкових тенденцій у сфері онлайн-розваг, стратегії платформенної розробки веб-кінотеатру, питання ліцензування та авторських прав, впровадження функціоналу інтерактивного взаємодії користувачів, а також підходи до забезпечення безпеки та конфіденційності даних.

Даний дослід покликаний внести внесок у розуміння та розвиток онлайн-розваг, допомогти підприємцям та розробникам у створенні успішних веб-кінотеатрів, а також надати практичні рекомендації щодо оптимізації функціоналу та управління веб-платформою з метою забезпечення високої якості та задоволення потреб сучасної аудиторії.

Основною метою дослідження є створення веб-кінотеатру, який надає користувачам можливість зручного та доступного перегляду різноманітного відеоконтенту через Інтернет. Крім того, мета включає в себе розробку функціональності та інтерфейсу, що забезпечують комфортну взаємодію користувачів з платформою.

Завданням є завдання та реалізація веб-кінотеатру з метою створення онлайн-платформи для перегляду та споживання відеоконтенту.

Об'єктом дослідження є процес створення та розвитку веб-кінотеатру як онлайн-платформи для перегляду фільмів, серіалів та іншого відеоконтенту. Включаючи в себе аналіз ринкових тенденцій, вибір технологічних рішень, розробку функціональності та інтерфейсу, забезпечення безпеки та конфіденційності даних, а також маркетинг та просування платформи.

РОЗДІЛ 1. АНАЛІЗ ІСНУ.ЧОЇ ІНФОРМАЦІЇ ЩОДО РОЗРОБКИ ОНЛАЙН КІНОТЕАТРУ

1.1. Порівняльний аналіз існуючої інформації та рішень

Перегляд існуючих веб-кінотеатрів та платформ для онлайн-перегляду відеоконтенту є важливим етапом у визначенні тенденцій та стандартів в даній сфері. Цей аналіз дозволяє отримати уявлення про різноманіття доступних сервісів, їхні основні особливості та переваги. Шляхом дослідження різних платформ користувач може визначити, які функції та можливості є найбільш важливими для нього особисто або для цільової аудиторії. Такий огляд також допомагає зрозуміти конкурентне середовище та визначити можливі ніші для нового веб-кінотеатру. Результати цього аналізу стануть важливою основою для розробки стратегії та функціональності власної платформи.

Опишимо декілька онлайн кінотеатрів до яких входить.

1. Netflix
2. Amazon Prime Video
3. Disney+
4. Hulu
5. HBO Max

Опишимо кожну з цих площадок більш детально.

Netflix - це один з найбільших та найвідоміших онлайн-кінотеатрів у світі. Заснований у 1997 році в США, Netflix перетворився на лідера у галузі стрімінгового відео та став відомим завдяки своєму вражаючому каталозі фільмів, серіалів та оригінального контенту.

Netflix має широкий вибір фільмів, серіалів, документальних фільмів та оригінального контенту у різних жанрах. Від класичних кінострічок до нових релізів, платформа пропонує різноманітність для всіх смаків.

Також, Netflix активно інвестує у власний оригінальний контент, включаючи серіали, фільми, комедійні шоу, документальні фільми та багато

іншого. Такі серіали, як "Stranger Things", "The Crown" та "Money Heist", стали справжніми хітами.

Платформа пропонує модель підписки, що дозволяє користувачам отримати необмежений доступ до вмісту за фіксовану щомісячну плату. Це дозволяє користувачам дивитися вміст без реклами та обмежень.

Netflix використовує алгоритми рекомендацій, щоб пропонувати користувачам вміст, який відповідає їхнім інтересам та передпочтенням. Це дозволяє зробити досвід перегляду більш індивідуалізованим та задоволенням.

Також, Netflix доступний на різних пристроях, включаючи смартфони, планшети, телевізори, консолі та комп'ютери, що дозволяє користувачам дивитися вміст в будь-який час та в будь-якому місці.

Amazon Prime Video - це частина платформи Amazon Prime, яка пропонує великий вибір відеоконтенту, включаючи фільми, серіали, оригінальний контент та багато іншого. Платформа була запущена в 2006 році і стала одним із найбільших конкурентів на ринку стрімінгового відео.

Amazon Prime Video пропонує широкий вибір відеоконтенту у різних жанрах, включаючи фільми, серіали, документальні фільми, мультфільми та оригінальний контент від Amazon Studios.

Платформа активно інвестує у власний оригінальний контент, включаючи такі хіти, як "The Marvelous Mrs. Maisel", "The Boys", "Fleabag" та інші. Цей унікальний контент привертає увагу аудиторії та відрізняє Amazon Prime Video від інших платформ.

Amazon Prime Video входить в склад платформи Amazon Prime, яка пропонує різноманітні переваги, включаючи безкоштовну доставку, музичний стрімінг, електронні книги та багато іншого, за фіксовану щорічну плату.

Платформа доступна на різних пристроях, включаючи смартфони, планшети, телевізори, консолі та комп'ютери, що дозволяє користувачам дивитися вміст в будь-який час та в будь-якому місці.

Amazon Prime Video використовує алгоритми рекомендацій, щоб пропонувати користувачам вміст, який відповідає їхнім інтересам та передпочтенням.

Disney+ - це стрімінгова платформа від компанії Disney, яка була запущена в листопаді 2019 року. Disney+ став місцем, де фанати можуть отримати доступ до всесвіту Disney, включаючи фільми, серіали, мультфільми, документальні фільми та багато іншого.

Disney+ пропонує широкий вибір вмісту від Disney, Pixar, Marvel, Star Wars та National Geographic. Це включає в себе класичні анімаційні фільми, нові релізи, серіали, оригінальний контент та багато іншого.

Також, Disney+ активно інвестує у власний оригінальний контент, включаючи серіали, фільми та документальні фільми, які доступні лише на цій платформі. Деякі з найбільш популярних оригінальних шоу включають "The Mandalorian", "WandaVision", "The Falcon and the Winter Soldier" та інші.

Платформа пропонує можливість стрімінгу в високій якості, включаючи 4K Ultra HD та HDR, де це доступно. Крім того, Disney+ дозволяє користувачам завантажувати контент для перегляду офлайн на різних пристроях.

Disney+ доступний на різних пристроях, включаючи смартфони, планшети, телевізори, консолі та комп'ютери.

Також, Disney+ зорієнтований на сімейний аудиторію, тому багато контенту націлено на дітей та дорослих одночасно, забезпечуючи розважальний досвід для всієї родини.

Hulu - це американська стрімінгова платформа, яка пропонує широкий вибір відеоконтенту, включаючи фільми, серіали, оригінальний контент та багато іншого. Hulu був запущений у 2007 році і став одним з популярних сервісів для стрімінгового відео у США.

Hulu пропонує великий каталог вмісту в різних жанрах, включаючи фільми, серіали, документальні фільми, анімаційні шоу та оригінальний контент.

Також, Hulu має партнерства з такими мережами, як ABC, NBC, FOX, FX, HBO, Showtime та іншими, що дозволяє платформі пропонувати широкий вибір відомого вмісту.

Платформа розробляє власний оригінальний контент, включаючи серіали та документальні фільми. Деякі з найбільш популярних оригінальних шоу включають "The Handmaid's Tale", "Castle Rock", "The Act" та інші.

Hulu пропонує підписку з різними планами, які дозволяють користувачам отримати доступ до вмісту без реклами або з обмеженим обсягом реклами за фіксовану щомісячну плату.

Також Hulu доступний на різних пристроях, включаючи смартфони, планшети, телевізори, консолі та комп'ютери.

HBO Max - це стрімінгова платформа, що належить WarnerMedia, яка була запущена у 2020 році. HBO Max є додатковою послугою від HBO, яка розширює доступність контенту, включаючи оригінальний вміст HBO, вміст від інших мереж та студій, а також ексклюзивний оригінальний контент від HBO Max.

Платформа має доступ до всього оригінального вмісту HBO, включаючи такі популярні шоу, як "Game of Thrones", "Westworld", "Succession" та багато інших.

Поміж оригінального вмісту HBO, HBO Max також пропонує великий вибір вмісту від інших мереж та студій, таких як Warner Bros., DC, Turner Classic Movies, Cartoon Network та інші.

Платформа розробляє власний оригінальний контент, включаючи серіали, фільми та документальні фільми, які доступні лише на HBO Max. Деякі з найбільш популярних оригінальних шоу включають "Lovecraft Country", "Raised by Wolves", "The Flight Attendant" та інші.

HBO Max пропонує різні підписки, включаючи підписку з рекламою та без реклами, а також доступ через інші пакети послуг WarnerMedia.

Також, HBO Max доступний на різних пристроях, включаючи смартфони, планшети, телевізори, консолі та комп'ютери.

Один із ключових аспектів для користувачів - можливість швидко знаходити вміст. Кожен з конкурентів має свою систему пошуку, яка може включати фільтрацію за жанрами, акторами, режисерами тощо.

Оцінка розмаїтості та обсягу вмісту, що пропонується. Це включає як останні новинки, так і класичні фільми та серіали, а також можливість доступу до ексклюзивного вмісту.

Аналіз різних моделей підписки, їх вартості, умов та можливостей. Це включає в себе розгляд доступу до преміум контенту, наявність реклами, можливість зміни або скасування підписки тощо.

Оцінка якості стрімінгу, наявність підтримки HD, 4K, HDR, а також можливості завантаження контенту для перегляду офлайн, перегляду на різних пристроях тощо.

Аналіз системи рекомендацій, яка адаптується під індивідуальні вподобання користувача, щоб пропонувати йому вміст, який йому може сподобатися.

Опишимо функціональні можливості кожного з додатків.

1. Netflix

- Можливість шукати вміст за назвою фільму, серіалу, актора чи режисера.
- Широкий вибір вмісту в різних жанрах, включаючи оригінальний контент Netflix.
- Модель підписки без реклами, яка дозволяє користувачам отримати необмежений доступ до контенту за фіксовану щомісячну плату.
- Підтримка відтворення в різних якостях, включаючи HD та 4K, а також можливість завантаження контенту для перегляду офлайн.
- Система рекомендацій, яка адаптується до вподобань користувача на основі його історії перегляду.

2. Amazon Prime Video

- Можливість шукати вміст за ключовими словами, жанрами, акторами тощо.

- Широкий вибір вмісту включаючи оригінальний контент Amazon.
- Модель підписки, що включає в себе не лише стрімінговий сервіс, а й інші переваги Amazon Prime.
- Підтримка відтворення в HD та Ultra HD, можливість завантаження контенту для перегляду офлайн.
- Система рекомендацій, яка адаптується до індивідуальних вподобань користувача.

3. Disney+

- Можливість пошуку вмісту за назвою, жанром, персонажем тощо.
- Широкий вибір вмісту включаючи класичні анімаційні фільми та оригінальний контент Disney.
- Модель підписки для отримання необмеженого доступу до контенту.
- Підтримка відтворення в HD та 4K, можливість завантаження контенту для перегляду офлайн.
- Рекомендації вмісту, які враховують історію перегляду та вподобання користувача.

4. Hulu

- Можливість шукати вміст за назвою, акторами, жанрами тощо.
- Широкий вибір вмісту включаючи оригінальний контент Hulu та інші шоу.
- Модель підписки з різними планами, включаючи доступ з рекламою та без.
- Підтримка відтворення в HD, можливість завантаження контенту для перегляду офлайн.
- Рекомендації вмісту, які адаптовані під індивідуальні вподобання користувача.

5. HBO Max

- Можливість шукати вміст за назвою, акторами, жанрами тощо.

- Широкий вибір вмісту, включаючи оригінальний контент HBO та інших мереж та студій.
- Модель підписки, яка включає доступ до всього вмісту за фіксовану щомісячну плату.
- Підтримка відтворення в HD, 4K та HDR, можливість завантаження контенту для перегляду офлайн.
- Рекомендації вмісту, які адаптовані під індивідуальні вподобання користувача.

Ці функціональні можливості допомагають користувачам знаходити, отримувати доступ та насолоджуватися вмістом відповідно до їхніх уподобань та потреб.

Нижче наведена таблиця порівняння функцій основних стрімінгових платформ.

Таблиця 1.1 – Порівняння функцій

Функції	Netflix	Amazon Prime Video	Disney+	Hulu	HBO Max
Пошук	Так	Так	Так	Так	Так
Каталог фільмів та серіалів	Широкий	Широкий	Широкий	Широкий	Широкий
Можливість підписки	Так	Так	Так	Так	Так
Функції відтворення та стрімінгу	HD, 4K, HDR, офлайн	HD, 4K, офлайн	HD, 4K, офлайн	HD, офлайн	HD, 4K, HDR, офлайн
Персоналізовані рекомендації	Так	Так	Так	Так	Так

Для залучення та утримання аудиторії стрімінгові платформи активно використовують різноманітні стратегії маркетингу та просування.

Багато конкурентів використовують широкомасштабні рекламні кампанії на соціальних мережах, на телебаченні, в Інтернеті та на вулицях для привертання уваги нових абонентів.

Співпраця з виробниками фільмів та серіалів, а також створення оригінального контенту спільно з відомими режисерами та акторами допомагає платформам привертати аудиторію за допомогою ексклюзивного контенту.

Залучення нових абонентів за допомогою промо-кодів, знижок на підписку, безкоштовних пробних періодів та інших спеціальних пропозицій.

Використання алгоритмів машинного навчання для аналізу вподобань користувачів і відповідне налаштування рекомендаційного вмісту та реклами.

Створення унікального та цікавого контенту, включаючи відео, статті, подкасти тощо, що дозволяє платформам взаємодіяти з аудиторією та залучати нових користувачів.

Партнерство з подіями, фестивалями, спортивними змаганнями та іншими подіями для просування свого бренду та привертання уваги аудиторії.

Аналіз цих стратегій допоможе отримати більш глибоке розуміння того, як конкуренти просувають свої продукти та як можна оптимізувати власні маркетингові зусилля для залучення та утримання аудиторії.

Одним із ключових аспектів стрімінгових платформ є їхня модель монетизації, яка визначає, як вони отримують прибуток від користувачів. Розглянемо основні моделі монетизації, що використовуються конкурентами.

1. Підписка (Subscription)

- Netflix. Модель підписки Netflix надає користувачам необмежений доступ до вмісту за фіксовану щомісячну плату.
- Amazon Prime Video. Абоненти Amazon Prime мають доступ до Prime Video безкоштовно як частина свого підписку на Amazon Prime, або можуть підписатися окремо на Prime Video.
- Disney+. Disney+ пропонує підписку, яка дає користувачам доступ до всього вмісту за щомісячну або річну плату.

- Hulu. Hulu пропонує різні плани підписки, включаючи версію з рекламою та без неї.

- HBO Max. Підписка на HBO Max дає користувачам доступ до всього вмісту за фіксовану щомісячну плату.

2. Покупка окремих фільмів або серіалів (Transactional)

- Крім підписки, деякі платформи, наприклад, Amazon Prime Video та Google Play Movies, дозволяють користувачам придбати окремі фільми або серіали за визначену ціну.

3. Реклама (Advertising)

- Hulu. Одна з моделей Hulu - безкоштовна версія з рекламою, де користувачі отримують доступ до вмісту безкоштовно, але переглядають рекламу.

- Peacock. Сервіс Peacock використовує модель монетизації "Freemium", де базова версія з рекламою доступна безкоштовно, але користувачі можуть оновити свій план на платний, щоб отримати доступ до додаткового вмісту та перегляду без реклами.

4. Гібридні моделі

- YouTube Premium. Платформа YouTube пропонує YouTube Premium, яка включає в себе безрекламний доступ до вмісту YouTube разом з підпискою на Google Play Music.

Аналіз цих моделей монетизації дозволяє краще розуміти стратегії платформ у залученні та утриманні аудиторії, а також їхні переваги та недоліки.

Аналіз моделей монетизації, що використовуються стрімінговими платформами, виявив різноманітні підходи до отримання прибутку від користувачів.

Підписка залишається однією з найпоширеніших моделей, яка забезпечує користувачам необмежений доступ до вмісту за фіксовану щомісячну або річну плату. Цей підхід дозволяє платформам залучати

користувачів за рахунок широкого вибору контенту та зручної системи оплати.

Крім того, моделі монетизації, що базуються на покупці окремих фільмів або серіалів, дозволяють користувачам отримати доступ до конкретного вмісту за визначену ціну, що може бути вигідним для тих, хто не хоче підписуватися на повний план.

Рекламна модель монетизації також залишається популярною, особливо в безкоштовних версіях платформ, де користувачі мають можливість перегляду контенту за рахунок рекламних відеороликів.

Загалом, різноманітність моделей монетизації дозволяє стрімінговим платформам пристосовуватися до різних потреб та вподобань користувачів, залучати нових абонентів та забезпечувати стабільний дохід.

1.2. Постановка завдання на проектування.

Веб-кінотеатр повинен виконувати ряд основних функцій, які забезпечать користувачам зручний та цікавий досвід перегляду відеоконтенту.

Користувачам повинна бути надана можливість шукати фільми та серіали за різними критеріями, такими як назва, жанр, рік випуску, актори, режисери тощо. Також важливо мати можливість фільтрувати результати пошуку за різними параметрами.

Користувачі повинні мати можливість переглядати фільми та серіали без перерви на рекламу, якщо вони користуються платним або преміум-планом. Підтримка відтворення в різних якостях (наприклад, HD, Full HD, 4K) також є важливою для забезпечення якісного перегляду контенту.

Веб-кінотеатр повинен надати користувачам можливість створити обліковий запис для зберігання персоналізованих налаштувань, історії перегляду, списку улюблених фільмів тощо. Авторизація також дозволяє контролювати доступ до платного вмісту та підтримувати безпеку користувача.

Система рекомендацій, яка аналізує історію перегляду та вподобання користувача, допомагає рекомендувати новий вміст, який відповідає їхнім інтересам. Персоналізовані списки рекомендацій, а також функції "Продовжити перегляд" і "На основі вашого перегляду" допомагають залучити користувача до нового вмісту.

Можливість користувачів залишати відгуки та оцінювати фільми та серіали дозволяє іншим користувачам зробити кращий вибір при виборі контенту для перегляду.

Деякі платформи надають можливість завантажувати вміст для перегляду офлайн, що дозволяє користувачам дивитися фільми та серіали без доступу до Інтернету, наприклад, під час подорожі або в дорозі.

Визначення можливостей пошуку та фільтрації вмісту є важливим аспектом для забезпечення зручності та ефективності користування веб-кінотеатром.

Користувачі повинні мати можливість шукати вміст за його назвою. Це дозволяє швидко знайти конкретний фільм або серіал.

Вміст може бути розділений за жанрами, такими як комедія, драма, фантастика, жахи тощо. Користувачі можуть шукати вміст відповідно до їхніх вподобань.

Користувачі можуть шукати вміст, в якому брали участь певні актори або режисери.

Можливість фільтрувати вміст за рейтингом користувачів або критиків допомагає зробити кращий вибір при виборі фільму або серіалу для перегляду.

Користувачі можуть шукати вміст за роками випуску, що дозволяє їм відшукати класичні фільми або найновіші прем'єри.

Можливість шукати вміст за тривалістю дозволяє користувачам відшукати фільми або серіали, які відповідають їхній доступній кількості часу для перегляду.

Користувачі можуть шукати вміст за мовою або країною виробництва, що дозволяє їм знайти фільми або серіали з конкретних регіонів або мов.

Можливість фільтрувати вміст за популярністю серед користувачів або за датою випуску допомагає знайти актуальний та популярний контент.

Забезпечення різноманітних можливостей пошуку та фільтрації допомагає користувачам знаходити та вибирати вміст, який відповідає їхнім вподобанням та потребам.

Встановлення вимог до процесу реєстрації та авторизації користувачів є важливим кроком для забезпечення безпеки та зручності використання веб-кінотеатру.

Наведемо деякі вимоги, які можна врахувати.

1. Реєстрація

- Необхідно визначити обов'язкові поля для реєстрації, такі як електронна адреса, пароль, ім'я користувача тощо.
- Система повинна перевіряти унікальність електронних адрес, щоб кожен користувач мав унікальний обліковий запис.
- Користувачі повинні підтверджувати свій пароль для підтвердження його правильності та захисту від помилкового введення.

2. Авторизація

- Система повинна забезпечувати захист від несанкціонованого доступу до облікового запису користувача шляхом використання безпечних методів авторизації, таких як двофакторна аутентифікація.
- Сесії користувача повинні бути належно керовані, з можливістю виходу з облікового запису та автоматичним завершенням сесії після тривалої неактивності.

3. Безпека даних

- Паролі користувачів повинні зберігатися в зашифрованому вигляді для запобігання їхнього неправильного використання у разі витоку даних.
- Усі дані, передані між користувачем та сервером, повинні бути зашифровані для захисту від перехоплення даних сторонніми особами.

4. Скасування облікового запису

- Користувачі повинні мати можливість скасувати свій обліковий запис у будь-який момент, якщо вони більше не хочуть користуватися сервісом.

5. Політика конфіденційності та Угоди користувача

- Сервіс повинен надати користувачам доступ до політики конфіденційності та угоди користувача, яка визначає правила користування та обробки їхніх особистих даних.

Врахування цих вимог допоможе забезпечити безпеку та зручність процесу реєстрації та авторизації користувачів у веб-кінотеатрі.

Вибір технологій для розробки веб-кінотеатру залежить від різних факторів, таких як функціональні вимоги, потужність та масштаб проекту, доступність ресурсів та знання команди розробників.

Для розробки клієнтської частини веб-кінотеатру можна використовувати JavaScript-фреймворки, такі як React.js, Vue.js або Angular. Ці фреймворки дозволяють розробляти інтерактивні та динамічні інтерфейси користувача.

На рівні серверної частини можна використовувати мови програмування, такі як Node.js (з використанням фреймворків Express.js або Nest.js), Python (з Django або Flask), Ruby (з Ruby on Rails), або Java (з Spring Framework).

Для зберігання даних про фільми, користувачів, сесій тощо можна використовувати реляційні бази даних, такі як MySQL, PostgreSQL або SQLite.

Для зберігання великого обсягу неструктурованих даних, таких як відеофайли, можна розглянути використання об'єктно-орієнтованих баз даних, таких як MongoDB або CouchDB.

Для розгортання та керування веб-кінотеатром можна використовувати хмарні платформи, такі як Amazon Web Services (AWS), Google Cloud Platform (GCP) або Microsoft Azure. Вони надають широкий спектр послуг, таких як віртуальні сервери, зберігання даних, CDN, моніторинг та безпека.

Для реалізації стрімінгової функціональності можна використовувати медіа-сервери, такі як Wowza Streaming Engine, nginx-rtmp або Helix Universal Server. Крім того, можна розглянути інтеграцію зі стрімінговими сервісами, такими як Amazon CloudFront або Akamai.

Для управління версіями коду та співпраці розробників можна використовувати системи контролю версій, такі як Git з платформами GitHub або GitLab.

Для автоматизованого тестування функціональності та виявлення помилок можна використовувати фреймворки для тестування, такі як Jest (для JavaScript) або PyTest (для Python).

Встановлення вимог до інтерфейсу користувача та його адаптивності є важливим етапом у розробці веб-кінотеатру, оскільки це визначає зручність та ефективність взаємодії користувача з платформою на різних пристроях і розмірах екранів.

Інтерфейс повинен бути розроблений з урахуванням принципів респонсивного дизайну, що дозволить автоматично адаптувати його до різних розмірів екранів, від маленьких мобільних пристроїв до великих десктопних моніторів.

Інтерфейс повинен бути оптимізований для мобільних пристроїв, з урахуванням обмеженого простору екрану та особливостей взаємодії, таких як торцеве меню, великі кнопки для торцевих пристроїв тощо.

Навігаційні елементи повинні бути легкодоступними та інтуїтивно зрозумілими для користувачів. Меню, пошукові поля та фільтри повинні бути легко знаходити та використовувати на будь-якому пристрої.

Графічні елементи, відео-прев'ю та постери повинні бути оптимізовані для швидкого завантаження та високої якості на різних пристроях і роздільних здатностях екранів.

Кнопки та елементи керування для відтворення, паузи, перемотування та інших функцій повинні бути зручно розміщені та легко натискатися на різних пристроях, включаючи сенсорні екрани.

Інтерфейс повинен бути оптимізований для роботи на різних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge тощо, забезпечуючи однакову функціональність та відображення на всіх платформах.

Інтерфейс повинен відповідати стандартам доступності для людей з обмеженими можливостями, забезпечуючи можливість використання платформи людьми з різними потребами та здібностями.

Перевірка і відладка інтерфейсу повинні включати тестування на різних пристроях і емуляторах, щоб переконатися в правильному відображенні та функціонуванні на різних платформах і пристроях.

Вимоги до інтерфейсу користувача та його адаптивності допоможуть забезпечити зручну та ефективну взаємодію користувачів з веб-кінотеатром на будь-якому пристрої та в будь-який час.

Визначення вимог до безпеки та захисту даних користувачів є критичним для забезпечення конфіденційності, цілісності та доступності їхніх особистих даних у веб-кінотеатрі.

Всі дані, що передаються між клієнтом та сервером, повинні бути зашифровані за допомогою протоколів шифрування, таких як SSL/TLS. Це забезпечить конфіденційність даних під час їхньої передачі через Інтернет.

Система повинна використовувати безпечні методи аутентифікації, такі як хешування паролів та використання токенів або сеансових ключів для авторизації користувачів.

Всі дані, що вводяться користувачами, повинні бути належним чином перевірені та очищені перед тим, як вони будуть використані в запитах до бази даних, щоб уникнути SQL-ін'єкцій та інших атак.

Система повинна мати механізми моніторингу та реагування на потенційні вторгнення, такі як виявлення аномальної активності та блокування доступу до системи в разі підозрілих дій.

Регулярне резервне копіювання даних і належне зберігання резервних копій допомагає забезпечити можливість відновлення даних у разі їх втрати або пошкодження.

Веб-кінотеатр повинен мати чітку політику конфіденційності та обробки даних, яка визначає, які дані збираються, як вони використовуються і кому вони доступні, а також як користувачі можуть контролювати свої дані.

Система повинна мати заходи захисту від розподілених атак на відмову в обслуговуванні (DDoS) та інших видів кібератак, що можуть призвести до перерв у роботі сервісу.

Збір та аналіз журналів дій користувачів допомагає виявляти підозрілу активність та вчасно реагувати на можливі загрози безпеці.

Врахування цих вимог допоможе забезпечити високий рівень захищеності даних користувачів та забезпечити безпеку їхнього взаємодії з веб-кінотеатром.

Врахування потреб користувачів щодо доступності та зручності використання веб-кінотеатру відіграє важливу роль у створенні успішного та конкурентоспроможного сервісу.

Веб-кінотеатр повинен мати респонсивний дизайн, який адаптується до різних типів пристроїв (комп'ютери, планшети, мобільні телефони), розмірів екранів та орієнтацій (горизонтальна або вертикальна). Це забезпечить зручне використання веб-кінотеатру на будь-якому пристрої.

Інтерфейс веб-кінотеатру повинен бути легким у використанні та інтуїтивно зрозумілим для користувачів. Основні функції, такі як пошук вмісту, навігація по категоріях та відтворення відео, повинні бути доступні без зайвих зусиль.

Веб-кінотеатр може надавати користувачам можливість налаштувати свій вміст відповідно до їхніх інтересів та вподобань. Це може включати рекомендації фільмів на основі перегляду, обрані жанри або режисерів, а також створення списків улюблених фільмів.

Користувачі оцінюють швидкий та безперервний доступ до вмісту без зайвої затримки. Оптимізація швидкості завантаження сторінок та відео, а також інші технічні аспекти, що впливають на швидкість роботи веб-сайту, є важливими.

Деякі користувачі можуть цінувати можливість завантажувати вміст для перегляду офлайн у тих випадках, коли вони не мають доступу до Інтернету. Функція завантаження контенту для перегляду офлайн може позитивно вплинути на користувальницький досвід.

Для користувачів з обмеженими можливостями, такими як візуальні чи слухові обмеження, важливо мати доступність до вмісту. Підтримка адаптивних технологій, таких як екранні читачі, може покращити доступність веб-кінотеатру для всіх користувачів.

Врахування цих аспектів допоможе забезпечити, що веб-кінотеатр буде доступним та зручним у використанні для широкого кола користувачів, що позитивно позначиться на їхньому задоволенні та відданості сервісу.

Встановлення конкретних цілей допомагає зорієнтуватися в роботі та забезпечити успішний розвиток веб-кінотеатру.

Опишимо декілька конкретних цілей, які можна досягти за допомогою створення веб-кінотеатру.

1. Збільшити кількість користувачів, які використовують веб-кінотеатр для перегляду фільмів і серіалів.

2. Підвищити кількість відвідувачів, які реєструються на платформі або підписуються на платну підписку.

3. Забезпечити регулярне оновлення та розширення бібліотеки вмісту для задоволення різноманітних смаків аудиторії.

4. Збільшити час, який користувачі проводять на веб-кінотеатрі, шляхом підвищення зацікавленості та привабливості контенту.

5. Зменшити кількість користувачів, які покидають веб-кінотеатр без перегляду контенту або реєстрації.

6. Забезпечити стабільний потік доходів через підписки, покупку окремих фільмів або рекламу.

7. Забезпечити зручну та привабливу платформу для користувачів з легким пошуком, навігацією та переглядом вмісту.

8. Створення веб-кінотеатру, який отримає позитивні відгуки та рекомендації від користувачів та індустрії.

Ці конкретні цілі допоможуть визначити стратегію розвитку веб-кінотеатру та забезпечити його успішну реалізацію.

Отже, для досягнення поставлених цілей для веб-кінотеатру потрібно сформулювати основні завдання, які варто виконати.

1. Залучення аудиторії

- Провести маркетингові кампанії для просування веб-кінотеатру на соціальних мережах, в медіа та інших каналах.
- Розробити стратегію приваблення нових користувачів через рекламу, акції та знижки.

2. Збільшення конверсії

- Оптимізувати процес реєстрації та підписки для зниження бар'єрів для користувачів.
- Запровадити привабливі пропозиції для нових користувачів, такі як безкоштовний пробний період або ексклюзивний вміст.

3. Покращення залучення контенту

- Встановити партнерські відносини з кіностудіями та виробниками контенту для регулярного оновлення бібліотеки вмісту.
- Аналізувати популярність та відгуки користувачів, щоб визначити найбільш привабливий вміст для додавання на платформу.

4. Збільшення затримки на сайті

- Підвищити якість рекомендаційного двигуна для особистого підбору контенту для кожного користувача.
- Забезпечити можливість створення списків улюблених фільмів та серіалів для подальшого перегляду.

5. Зниження відсотку відмов

- Оптимізувати швидкість завантаження сторінок та відео для зменшення часу очікування користувачів.
- Покращити навігацію та пошук, щоб користувачі легко знаходили те, що їх цікавить.

6. Монетизація

- Розробити різні моделі монетизації, такі як підписки, покупка окремих фільмів та реклама.
- Аналізувати результативність кожної моделі та оптимізувати їх для забезпечення максимальних прибутків.

7. Покращення користувацького досвіду

- Провести дослідження користувачів та впровадити їхні рекомендації для поліпшення користувацького досвіду.
- Постійно вдосконалювати інтерфейс та функціонал веб-кінотеатру на основі зворотного зв'язку користувачів.

8. Збільшення репутації

- Забезпечити високу якість вмісту та сервісу для задоволення потреб користувачів.
- Активно взаємодіяти з аудиторією через соціальні мережі, форуми та інші канали для отримання відгуків та підтримки.

Ці завдання спрямовані на досягнення визначених цілей та забезпечення успішного розвитку веб-кінотеатру.

РОЗДІЛ 2. ПРОЕКТНІ І ТЕХНІЧНІ РІШЕННЯ. ВИДИ ЗАБЕЗПЕЧЕННЯ

2.1. Інформаційне забезпечення.

Для розроблення бази даних вашого веб-кінотеатру, я рекомендую використовувати реляційну базу даних, таку як MySQL, оскільки вона добре підходять для зберігання структурованої інформації та має широкі можливості для оптимізації та масштабування.

Напишимо основні можливі таблиці та їхні поля для зберігання інформації про фільми, користувачів та рейтинги.

1. Таблиця "Фільми".
- id (унікальний ідентифікатор фільму)
- назва фільму
- рік випуску
- жанр
- тривалість
- режисер
- опис
- посилання на зображення постера тощо.

2. Таблиця "Користувачі".
- id (унікальний ідентифікатор користувача)
- ім'я
- прізвище
- електронна пошта
- пароль (зашифрований)
- дата реєстрації
- інформація про підписку (якщо користувач підписаний на платний план)

3. Таблиця "Рейтинги".

- id (унікальний ідентифікатор рейтингу)
- id фільму (зовнішній ключ, пов'язує з таблицею "Фільми")
- id користувача (зовнішній ключ, пов'язує з таблицею "Користувачі")
- рейтинг (від 1 до 5)
- коментар (необов'язково)

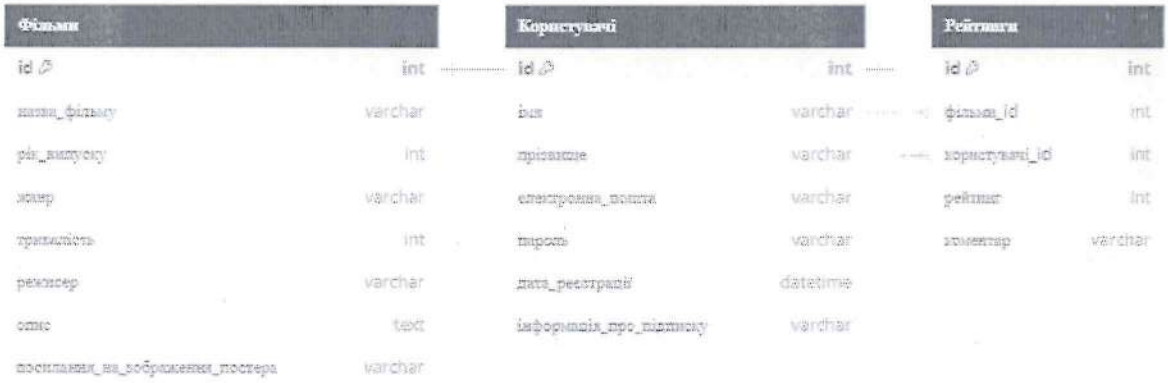

Рис.2.1 – База даних

Реалізація системи аутентифікації та авторизації користувачів є критично важливою для забезпечення безпеки даних у вашому веб-кінотеатрі.

Паролі користувачів повинні бути збережені в базі даних у захешованому вигляді, використовуючи сильний алгоритм хешування, такий як bcrypt або SHA-256. Це дозволить уникнути зламу паролів в разі витоку даних.

Після успішної аутентифікації користувача генерується унікальний маркер сесії, який зберігається на стороні клієнта (у куках або локальному сховищі) і використовується для авторизації користувача під час кожного запиту.

Кожен запит від користувача повинен бути перевірений на наявність дійсного маркера сесії та відповідних прав доступу до ресурсу. Це забезпечить тільки авторизованим користувачам доступ до захищених областей.

Використання HTTPS для забезпечення шифрування даних між клієнтом і сервером допоможе уникнути атак на перехоплення сесії.

Встановлення обмеження на кількість невдалих спроб авторизації перед блокуванням акаунту на певний період часу може запобігти спробам перебору паролів.

Надання користувачам можливості безпечного виходу з системи, що включає видалення маркера сесії та очищення будь-яких збережених даних аутентифікації на клієнтському боці.

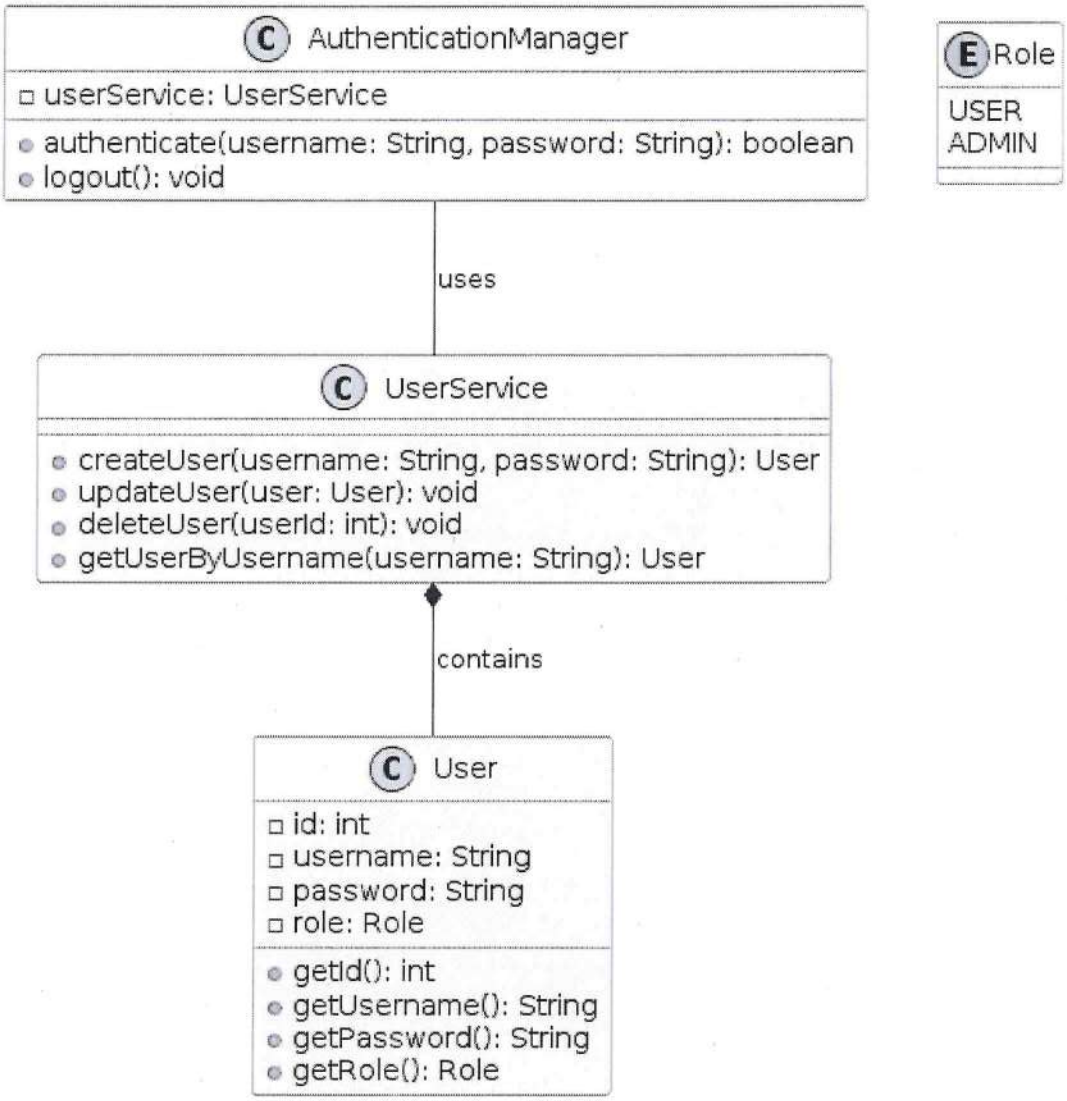

Рис.2.2 - Реалізація системи аутентифікації та авторизації користувачів

Реалізація цих заходів дозволить забезпечити ефективну систему аутентифікації та авторизації, яка забезпечить безпеку даних у вашому веб-кінотеатрі.

Створення API для взаємодії з фронтендом та іншими додатками дозволить забезпечити взаємодію вашого веб-кінотеатру з іншими системами чи мобільними додатками. Ось загальний опис можливого API.

1. Отримання списку фільмів.

- Метод. GET
- URL-шлях. /api/films
- Опис. Запит для отримання списку всіх фільмів у вашому кінотеатрі.

2. Отримання деталей конкретного фільму.

- Метод. GET
- URL-шлях. /api/films/{film_id}
- Опис. Запит для отримання деталей про конкретний фільм за його ідентифікатором.

3. Аутентифікація користувача.

- Метод. POST
- URL-шлях. /api/auth/login
- Опис. Запит для аутентифікації користувача за його ім'ям користувача та паролем.

4. Отримання даних про користувача.

- Метод. GET
- URL-шлях. /api/auth/user
- Опис. Запит для отримання даних про авторизованого користувача.

5. Встановлення рейтингу фільму.

- Метод. POST
- URL-шлях. /api/ratings
- Опис. Запит для встановлення рейтингу фільму користувачем.

6. Створення нового користувача.

- Метод. POST
- URL-шлях. /api/users

- Опис. Запит для створення нового користувача в системі.

7. Пошук фільмів за різними критеріями.

- Метод. GET
- URL-шлях. /api/search?q={query}&genre={genre}&year={year}
- Опис. Запит для пошуку фільмів за різними критеріями, такими як загальний пошук, жанр чи рік випуску.

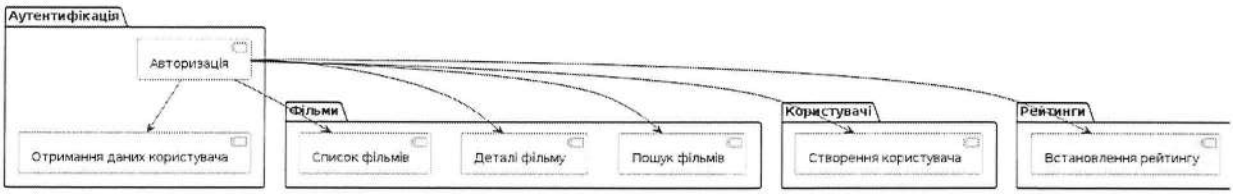

Рис 2.3. - Візуальне представлення структури API для вашого веб-кінотеатру

Це представлення включає компоненти, такі як робота з фільмами, аутентифікація користувачів, робота з користувачами та встановлення рейтингів фільмів. Кожен компонент має свої взаємодіючі ендпоінти, які надають можливість взаємодії з ним через API. Це важливий крок для розробки та реалізації взаємодії між фронтендом та серверною частиною вашого додатку, а також для підтримки роботи з іншими зовнішніми додатками.

2.2. Математичне забезпечення

Для персоналізованих рекомендацій у онлайн кінотеатрах використовуються різні алгоритми, що базуються на методах машинного навчання та аналізу даних.

Колаборативний фільтр, алгоритм використовує спільну інформацію про активність користувачів та рейтинги для рекомендації контенту. Він може бути реалізований як популярність, колаборативна фільтрація за спільністю (SVD), або методи засновані на зміщенні.

Контент-базовані методи, алгоритми аналізують властивості контенту (наприклад, жанр, актори, режисер) та порівнюють їх зі смаками та вподобаннями користувачів для рекомендацій.

Гібридні моделі, комбінація різних підходів, таких як колаборативний фільтр та контент-базовані методи. Гібридні системи можуть бути більш ефективними, оскільки вони поєднують переваги кожного методу та компенсують їхні недоліки.

Асоціативні правила, алгоритми використовують асоціативні правила для виявлення зв'язків між різними елементами контенту та рекомендацію відповідного контенту на основі цих зв'язків.

Методи засновані на зміщенні, алгоритми враховують відмінності в оцінках користувачів та елементів контенту та використовують їх для генерації рекомендацій.

Математичні методи грають важливу роль у аналізі та обробці даних про фільми та користувачів у онлайн кінотеатрах.

Використання методів статистики дозволяє аналізувати розподіл даних про фільми та користувачів, виявляти відмінності між групами користувачів та розуміти поведінку користувачів.

Методи лінійної алгебри, такі як сингулярний розклад (SVD), використовуються для аналізу матриць рейтингів фільмів користувачами, а також для рекомендацій та прогнозування предпочтень.

Оптимізаційні алгоритми використовуються для підбору параметрів моделей рекомендацій, щоб максимізувати точність та релевантність рекомендацій.

Вони дозволяють групувати користувачів та фільми на основі їхніх характеристик або взаємодій, що полегшує аналіз та рекомендації.

Методи машинного навчання, такі як нейронні мережі, дерева рішень, метод опорних векторів тощо, використовуються для розробки складних моделей рекомендацій та аналізу даних.

Вона використовується для моделювання ймовірності подій, таких як імовірність подобання фільму користувачеві або імовірність того, що користувач здійснить певну дію.

Ці математичні методи допомагають зрозуміти структуру даних, виявляти закономірності та здійснювати аналіз, що полегшує прийняття рішень щодо рекомендацій та вдосконалення користувацького досвіду у веб-кінотеатрі.

Розглянемо метод колаборативного фільтрування, який є одним з найпоширеніших у системах рекомендацій.

Починається зі збору даних про взаємодію користувачів з контентом. Це може бути історія перегляду фільмів, оцінки, відгуки тощо.

На основі зібраних даних створюється матриця, де рядки відповідають користувачам, стовпці - фільмам, а елементи - рейтингам, які користувачі присвоюють фільмам.

Визначаємо метод обчислення схожості між користувачами або фільмами. Зазвичай використовуються косинусна схожість або кореляція Пірсона.

Застосовуємо обраний метод для обчислення схожості між користувачами або фільмами на основі їхніх відгуків.

Для кожного користувача або фільму вибираємо N найбільш схожих елементів. Це може бути N схожих користувачів для даного користувача або N схожих фільмів для даного фільму.

На основі вибраних схожих елементів формуємо рекомендації для користувачів. Наприклад, якщо користувач подібний до іншого користувача, рекомендуємо йому фільми, які сподобалися цьому іншому користувачеві.

Після того, як рекомендації згенеровано, їх можна оцінити за допомогою метрик ефективності, таких як точність, покриття тощо. Якщо потрібно, модель можна оптимізувати, змінюючи метод схожості або параметри алгоритму.

Після успішного тестування рекомендаційна система впроваджується в продукт. Потім слід постійно моніторити її роботу та вносити виправлення для забезпечення оптимального функціонування.

Розрахунок схожості між користувачами:

a. Кореляція Пірсона (Pearson correlation)

$$r_{xy} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

Де:

r_{xy} - коефіцієнт кореляції між користувачами x та y,

xi, yi - рейтинги, які користувачі x та y поставили спільним фільмам,

$\bar{x}, \bar{y}$ - середні рейтинги користувачів x та y.

b. Косинусна схожість (Cosine similarity):

$$\text{similarity}(u, v) = \frac{\sum_i r_{ui} \cdot r_{vi}}{\sqrt{\sum_i r_{ui}^2} \cdot \sqrt{\sum_i r_{vi}^2}}$$

Де:

rui, rvi - рейтинги, які користувачі u та v поставили фільму i.

Після розрахунку схожості між користувачами можна вибрати N найбільш схожих користувачів для кожного користувача та рекомендувати їм фільми, які сподобалися цим користувачам, але не були переглянуті поточним.

Після генерації рекомендацій їх ефективність можна оцінити за допомогою метрик, таких як точність, покриття тощо. Змінюючи метод обчислення схожості або параметри алгоритму, можна оптимізувати модель для поліпшення рекомендацій.

2.3. Програмне забезпечення

Опишимо код розробленого додатку

```
#!/usr/bin/env python
import os
import sys
```

```
def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'movie_review.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Цей код є типовим для запуску Django-додатка з командного рядка. Давайте розберемо кожен рядок коду:

#!/usr/bin/env python: Цей рядок, відомий як "шебанг", вказує операційній системі, яку інтерпретатор Python використовувати для виконання скрипту. У цьому випадку використовується інтерпретатор Python, який може бути знайдений в поточному середовищі.

import os: Цей оператор імпортує модуль os, який надає функції для взаємодії з операційною системою.

import sys: Цей оператор імпортує модуль sys, який надає доступ до змінних та функцій, пов'язаних з інтерпретатором Python.

def main():: Цей рядок починає оголошення функції з іменем main.

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'movie_review.settings'): Цей рядок встановлює значення змінної середовища DJANGO_SETTINGS_MODULE на 'movie_review.settings'. Це потрібно для налагодження Django, щоб він знав, які налаштування використовувати.

try:: Цей рядок починає блок спроби виконати наступні рядки коду.

from django.core.management import execute_from_command_line: Цей рядок імпортує функцію execute_from_command_line з модуля django.core.management, яка використовується для запуску Django-додатка з командного рядка.

except ImportError as exc:: Цей рядок починає блок обробки винятків для випадку, якщо виникає помилка імпорту.

raise ImportError() from exc: Цей рядок піднімає новий виняток типу ImportError, якщо виникає помилка імпорту.

execute_from_command_line(sys.argv): Цей рядок викликає функцію execute_from_command_line з аргументами командного рядка sys.argv, які передаються в програму.

if __name__ == '__main__':: Цей рядок перевіряє, чи виконується скрипт безпосередньо (а не імпортується з іншого модуля).

main(): Цей рядок викликає функцію main, щоб запустити основний код програми, який був оголошений раніше.

```
from django.shortcuts import render,get_object_or_404
from django.shortcuts import render ,redirect
from django.contrib.auth.models import User
from django.contrib import messages
from user.forms import SignupForm,EditProfileForm
from django.contrib.auth.decorators import login_required
from .models import Profile
from django.template import loader
from django.http import HttpResponse



def Signup(request):
    if request.method == 'POST':
```

```
        form = SignupForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            email = form.cleaned_data.get('email')
            first_name = form.cleaned_data.get('first_name')
            last_name = form.cleaned_data.get('last_name')
            password = form.cleaned_data.get('password')
            User.objects.create_user(username=username, email=email,
first_name=first_name, last_name=last_name, password=password)
            return redirect('login')
    else:
        form = SignupForm()

    context = {
        'form': form,
    }

    return render(request, 'user/register.html', context)
```

```
@login_required
def EditProfile(request):
    user = request.user.id
    profile = Profile.objects.get(user__id=user)

    if request.method == 'POST':
        form = EditProfileForm(request.POST, request.FILES)
        if form.is_valid():
            profile.picture = form.cleaned_data.get('picture')
            profile.first_name = form.cleaned_data.get('first_name')
```

```
            profile.last_name = form.cleaned_data.get('last_name')
            profile.location = form.cleaned_data.get('location')
            profile.url = form.cleaned_data.get('url')
            profile.profile_info = form.cleaned_data.get('profile_info')
            profile.save()
            return redirect('profile',username=profile.user.username)
    else:
        form = EditProfileForm()

    context = {
        'form': form,
    }

    return render(request, 'user/edit_profile.html', context)

def UserProfile(request,username):
    user=get_object_or_404(User,username=username)
    profile=Profile.objects.get(user=user)
    movies_watched = profile.watched.filter(Type='movie')
    series_watched = profile.watched.filter(Type='series')
    watchlist=profile.to_watch.all().order_by("-imdbRating")

    context={
        'profile':profile,
        'movies_watched':movies_watched,
        'series_watched':series_watched,
        'watchlist':watchlist,
    }
    template=loader.get_template('user/user_profile.html')
    return HttpResponse(template.render(context,request))
```

1. Signup
    - У цій функції ми спочатку перевіряємо, чи метод запиту - POST. Якщо так, то ми створюємо форму реєстрації з отриманими даними.
    - Якщо форма є валідною, ми витягуємо дані з форми (ім'я, електронна пошта, тощо) та створюємо нового користувача за допомогою User.objects.create_user.
    - Після успішного створення користувача ми перенаправляємо його на сторінку входу.
2. EditProfile
    - У цій функції спочатку ми отримуємо профіль користувача за його id.
    - Якщо метод запиту - POST, то ми створюємо форму редагування профілю з отриманими даними.
    - Якщо форма є валідною, ми оновлюємо дані профілю користувача за допомогою даних з форми.
    - Після успішного оновлення профілю ми перенаправляємо користувача на його власну сторінку профілю.
3. UserProfile
    - У цій функції ми отримуємо користувача за його ім'ям користувача.
    - За допомогою get_object_or_404 ми знаходимо відповідний об'єкт користувача в базі даних або видаємо помилку 404, якщо користувача не знайдено.
    - Після цього ми отримуємо профіль користувача та список його переглянутих фільмів і серіалів.
    - В кінці ми передаємо ці дані в шаблон і відображаємо сторінку профілю користувача за допомогою методу render.

```
4. from django.urls import path
   from .views import EditProfile,Signup
   from django.contrib.auth import views as userViews
   from user.views import UserProfile

   urlpatterns = [
      path('login/',
   userViews.LoginView.as_view(template_name='user/login.html',next_page
   ='landing'), name='login'),
      path('logout/', userViews.LogoutView.as_view(next_page='landing'),
   name='logout'),
      path('profile/<username>/', UserProfile, name='profile'),
      path('profile/edit', EditProfile, name='edit-profile'),
      path('signup/', Signup, name='register'),
   ]
```

Цей фрагмент коду відповідає за налаштування маршрутів (URL) в Django-додатку.

from django.urls import path: Цей рядок імпортує функцію path з модуля django.urls. Функція path використовується для визначення шаблонів маршрутів у Django-додатку.

from .views import EditProfile, Signup: Цей рядок імпортує функції EditProfile та Signup з модуля views поточного пакету (.views). Ці функції використовуються для обробки запитів на редагування профілю та реєстрацію користувачів відповідно.

from django.contrib.auth import views as userViews: Цей рядок імпортує функції автентифікації користувачів з модуля views пакету django.contrib.auth. Ці функції використовуються для обробки стандартних дій, таких як вхід в систему (LoginView) та вихід з системи (LogoutView).

from user.views import UserProfile: Цей рядок імпортує функцію UserProfile з модуля views пакету user. Ця функція використовується для відображення профілю користувача.

urlpatterns: Цей рядок визначає список маршрутів (URL) для додатку. Кожен елемент у цьому списку - це об'єкт path, який визначає шаблон URL для певної сторінки та функцію, яка буде виконана при обробці цього URL.

path('login/', userViews.LoginView.as_view(template_name='user/login.html',next_page='landing'), name='login'): Цей рядок встановлює маршрут для сторінки входу в систему. При обробці цього URL викликається клас LoginView з пакету django.contrib.auth.views, який відображає сторінку входу в систему.

path('logout/', userViews.LogoutView.as_view(next_page='landing'), name='logout'): Цей рядок встановлює маршрут для виходу з системи. При обробці цього URL викликається клас LogoutView з пакету django.contrib.auth.views, який виконує вихід з системи.

path('profile/<username>/', UserProfile, name='profile'): Цей рядок встановлює маршрут для сторінки профілю користувача. При обробці цього URL викликається функція UserProfile, яка відображає профіль користувача.

path('profile/edit', EditProfile, name='edit-profile'): Цей рядок встановлює маршрут для сторінки редагування профілю користувача. При обробці цього URL викликається функція EditProfile, яка дозволяє користувачеві редагувати свій профіль.

path('signup/', Signup, name='register'): Цей рядок встановлює маршрут для сторінки реєстрації нового користувача. При обробці цього URL викликається функція Signup, яка відображає сторінку реєстрації.

```
from PIL import Image
from django.conf import settings
import os
from cloudinary.models import CloudinaryField


# Create your models here.
def user_directory_path(instance,filename):
    profile_pic_name = 'user/user_{0}/profile.jpg'.format(instance.user.id)
    full_path = os.path.join(settings.MEDIA_ROOT, profile_pic_name)
    if os.path.exists(full_path):
        os.remove(full_path)
    return profile_pic_name

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,
related_name='profile')
    first_name = models.CharField(max_length=50, null=True, blank=True)
    last_name = models.CharField(max_length=50, null=True, blank=True)
    location = models.CharField(max_length=50, null=True, blank=True)
    url = models.CharField(max_length=80, null=True, blank=True)
    profile_info = models.TextField(max_length=150, null=True, blank=True)
    created = models.DateField(auto_now_add=True)
    to_watch = models.ManyToManyField(Movie, related_name='towatch')
    watched = models.ManyToManyField(Movie, related_name='watched')
    picture = CloudinaryField('image')

    def save(self, *args, **kwargs):
        super().save(*args, **kwargs)
        SIZE = 250, 250
```

```
#     if self.picture:
#         pic = Image.open(self.picture.path)
#         pic.thumbnail(SIZE, Image.LANCZOS)
#         pic.save(self.picture.path)

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

def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

def save_user_profile(sender, instance, **kwargs):
    instance.profile.save()


post_save.connect(create_user_profile, sender=User)
post_save.connect(save_user_profile, sender=User)
```

Цей фрагмент коду містить моделі для профілів користувачів у Django-додатку.

import:

from django.db import models: Цей рядок імпортує класи моделей Django, які використовуються для визначення структури бази даних.

Models:

class Profile(models.Model):: Цей клас визначає модель профілю користувача. Він має такі поля, як ім'я, прізвище, місцезнаходження, URL, інформація про профіль, дата створення тощо.

user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='profile'): Поле user встановлює зв'язок "один до одного" з

моделлю User Django, яке визначає вбудовану модель користувача. При видаленні користувача його профіль також видаляється.

to_watch = models.ManyToManyField(Movie, related_name='towatch'): Поле to_watch встановлює зв'язок "багато до багатьох" з моделлю Movie, яке визначає фільми, які користувач планує подивитися.

watched = models.ManyToManyField(Movie, related_name='watched'): Поле watched встановлює зв'язок "багато до багатьох" з моделлю Movie, яке визначає фільми, які користувач вже переглянув.

picture = CloudinaryField('image'): Поле picture зберігає зображення профілю користувача за допомогою CloudinaryField, що дозволяє зберігати зображення в хмарі та автоматично оптимізувати його розмір.

Функції:

def user_directory_path(instance, filename):: Ця функція визначає шлях для зберігання зображення профілю користувача на сервері. Вона приймає екземпляр моделі та ім'я файлу та повертає шлях збереження.

Сигнали:

post_save.connect(create_user_profile, sender=User): Цей рядок підключає функцію create_user_profile до сигналу post_save моделі User. Ця функція автоматично створює профіль користувача при створенні нового користувача.

post_save.connect(save_user_profile, sender=User): Цей рядок підключає функцію save_user_profile до сигналу post_save моделі User. Ця функція автоматично зберігає профіль користувача при збереженні користувача.

```
from django import forms
from django.contrib.auth.models import User
from django.core.exceptions import ValidationError
from user.models import Profile


def UniqueUser(value):
    if User.objects.filter(username__iexact=value).exists():
```

```
        raise ValidationError('User with this this username already exists.')

def UniqueEmail(value):
    if User.objects.filter(email__iexact=value).exists():
        raise ValidationError('User with this email already exists.')


class SignupForm(forms.ModelForm):
    username = forms.CharField(widget=forms.TextInput(), max_length=30,
required=True)
    email = forms.CharField(widget=forms.EmailInput(), max_length=100,
required=True)
    first_name = forms.CharField(widget=forms.TextInput(), max_length=50,
required=True)
    last_name = forms.CharField(widget=forms.TextInput(), max_length=50,
required=True)
    password = forms.CharField(widget=forms.PasswordInput())
    confirm_password = forms.CharField(widget=forms.PasswordInput(),
required=True, label='Confirm your password.')

    class Meta:
        model = User
        fields = ('username', 'email', 'first_name','last_name', 'password')

    def __init__(self, *args, **kwargs):
        super(SignupForm, self).__init__(*args, **kwargs)
        self.fields['username'].validators.append(UniqueUser)
        self.fields['email'].validators.append(UniqueEmail)

    def clean(self):
```

```
        super(SignupForm, self).clean()
        password = self.cleaned_data.get('password')
        confirm_password = self.cleaned_data.get('confirm_password')

        if password != confirm_password:
            self._errors['password'] = self.error_class(['Password do not match. Try
again'])
        return self.cleaned_data

class EditProfileForm(forms.ModelForm):
    picture = forms.ImageField(required=False)
    first_name = forms.CharField(widget=forms.TextInput(), max_length=50,
required=False)
    last_name = forms.CharField(widget=forms.TextInput(), max_length=50,
required=False)
    location = forms.CharField(widget=forms.TextInput(), max_length=25,
required=False)
    url = forms.CharField(widget=forms.TextInput(), max_length=100,
required=False)
    profile_info = forms.CharField(widget=forms.TextInput(), max_length=150,
required=False)

    class Meta:
        model = Profile
        fields = ('picture', 'first_name', 'last_name', 'location', 'url', 'profile_info')
```

Цей фрагмент коду містить форми для реєстрації користувачів та редагування профілю.

def UniqueUser(value): Ця функція виконує перевірку, чи існує користувач з таким же ім'ям користувача.

def UniqueEmail(value): Ця функція виконує перевірку, чи існує користувач з такою ж електронною поштою.

Встановлюються поля для введення username, email, first_name, last_name, password та confirm_password.

Визначається клас Meta, який вказує на модель User та поля, які потрібно включити в форму.

В функції clean виконується перевірка того, чи співпадають введені пароль та підтвердження пароля.

Форма редагування профілю (EditProfileForm):

Встановлюються поля для введення зображення профілю, імені, прізвища, місцезнаходження, URL та інформації про профіль.

Визначається клас Meta, який вказує на модель Profile та поля, які потрібно включити в форму.

Ці форми використовуються для взаємодії з користувачами на веб-сайті, де вони можуть реєструватися та редагувати свої профілі. Вони також містять валідацію даних, щоб гарантувати їх правильність та цілісність.

```
from .forms import RateForm
from django.shortcuts import render, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.template import loader
from django.http import HttpResponse, HttpResponseRedirect
import requests
from .models import Movie, Genre, Rating,Review
from django.utils.text import slugify
from user.models import Profile
from django.contrib.auth.models import User
from django.urls import reverse
from django.db.models import Avg
```

```
# Create your views here.
def landing(request):
    if Movie.objects.all().exists():
        genre_list=Genre.objects.all()
    else:
        genre_list=[]
    query=request.GET.get('q')
    if query:
        url='https://www.omdbapi.com/?apikey=c9161d22&s='+query
        response=requests.get(url)
        movie_data=response.json()
        context={
            'query':query,
            'movie_data':movie_data,
        }
        template=loader.get_template('search.html')
        return HttpResponse(template.render(context,request))
    context={'genre_list':genre_list}
    return render(request,'landing.html',context)


@login_required(login_url='login')
def movieDetails(request,imdb_id):
    if Movie.objects.filter(imdbID=imdb_id).exists():
        movie_data=Movie.objects.get(imdbID=imdb_id)
        reviews = Review.objects.filter(movie=movie_data)
        if Review.objects.filter(movie=movie_data,user=request.user).exists():
            reviewed=True
        else:
            reviewed=False
        reviews_avg = reviews.aggregate(Avg('rate'))
```

```
    if Review.objects.filter(movie=movie_data).exists():
        reviews_count = reviews.count()
    else:
        reviews_count=0
    our_db=True
else:
    url='http://www.omdbapi.com/?apikey=c9161d22&i='+imdb_id
    response=requests.get(url, timeout=3)
    movie_data=response.json()

    #inject
    rating_objs=[]
    genre_objs=[]

    #genre
    genre_list=list(movie_data['Genre'].replace(" ","").split(','))
    for genre in genre_list:
        genre_slug=slugify(genre)
        g,created=Genre.objects.get_or_create(title=genre,slug=genre_slug)
        genre_objs.append(g)

    #Rate
    for rate in movie_data['Ratings']:

r,created=Rating.objects.get_or_create(source=rate['Source'],rating=rate['Value'])
        rating_objs.append(r)

    #Language
    language_list=[x.strip() for x in movie_data['Language'].split(',')]
    if language_list[0] == 'Hindi':
```

```
            genre_slug=slugify("Bollywood")
            g,created=Genre.objects.get_or_create(title="Bollywood",slug=genre_slug)
            genre_objs.append(g)
        elif language_list[0] == 'English':
            genre_slug=slugify("Hollywood")
            g,created=Genre.objects.get_or_create(title="Hollywood",slug=genre_slug)
            genre_objs.append(g)
        elif language_list[0] == 'Tamil':
            genre_slug=slugify("Kollywood")
            g,created=Genre.objects.get_or_create(title="Kollywood",slug=genre_slug)
            genre_objs.append(g)
        elif  language_list[0] == 'Telugu':
            genre_slug=slugify("Tollywood")
            g,created=Genre.objects.get_or_create(title="Tollywood",slug=genre_slug)
            genre_objs.append(g)
        elif language_list[0] == 'Kannada':
            genre_slug=slugify("Sandalwood")

g,created=Genre.objects.get_or_create(title="Sandalwood",slug=genre_slug)
            genre_objs.append(g)
        else:
            genre_slug=slugify("Others")
            g,created=Genre.objects.get_or_create(title="Others",slug=genre_slug)
            genre_objs.append(g)

        if language_list[0] == 'Japanese' and genre_list[0] == 'Animation' and
movie_data['Type'] == 'series':
            genre_slug=slugify("Anime")
            g,created=Genre.objects.get_or_create(title="Anime",slug=genre_slug)
            genre_objs.append(g)
```

```
 m,created=Movie.objects.get_or_create(
    Title=movie_data['Title'],
    Year=movie_data['Year'],
    Rated=movie_data['Rated'],
    Released=movie_data['Released'],
    Runtime=movie_data['Runtime'],
    Actors=movie_data['Actors'],
    Director=movie_data['Director'],
    Writer=movie_data['Writer'],
    Plot=movie_data['Plot'],
    Language=movie_data['Language'],
    Country=movie_data['Country'],
    Metascore=movie_data['Metascore'],
   imdbRating=movie_data['imdbRating'],
    Awards=movie_data['Awards'],
    Poster_url=movie_data['Poster'],
    Type=movie_data['Type'],
    imdbID=movie_data['imdbID'],
)
m.Genre.set(genre_objs)
m.Ratings.set(rating_objs)
m.save()
our_db=False
reviewed=False
movie_data_2=Movie.objects.get(imdbID=imdb_id)
reviews = Review.objects.filter(movie=movie_data_2)
reviews_avg='N/A'
reviews_count=0
```

```
    context={
        'movie_data':movie_data,
        'our_db':our_db,
        'reviews':reviews,
        'reviews_avg':reviews_avg,
        'reviews_count':reviews_count,
        'reviewed':reviewed,
    }
    template=loader.get_template('MovieDetailsPage.html')
    return HttpResponse(template.render(context,request))

def genres(request,genre_slug):
    genre=get_object_or_404(Genre,slug=genre_slug)
    movie_data=Movie.objects.filter(Genre=genre).order_by("-imdbRating")
    context={
        'genre_slug':genre_slug,
        'movie_data':movie_data,
    }
    template=loader.get_template('genre.html')
    return HttpResponse(template.render(context,request))

def Rate(request, imdb_id):
    movie = Movie.objects.get(imdbID=imdb_id)
    user = request.user
    if request.method == 'POST':
        form = RateForm(request.POST)
        if form.is_valid():
            rate = form.save(commit=False)
            rate.user = user
            rate.movie = movie
```

```
            rate.save()
            return HttpResponseRedirect (reverse('movie-details', args=[imdb_id]))
    else:
        form = RateForm()
    template = loader.get_template('rate.html')
    context = {
        'form': form,
        'movie': movie,
    }
    return HttpResponse(template.render(context, request))

def watchlist(request,imdb_id):
    movie=Movie.objects.get(imdbID=imdb_id)
    user=request.user
    profile=Profile.objects.get(user=user)
    profile.to_watch.add(movie)
    return HttpResponseRedirect(reverse('movie-details',args=[imdb_id]))

def watched_movies(request,imdb_id):
    movie=Movie.objects.get(imdbID=imdb_id)
    user=request.user
    profile=Profile.objects.get(user=user)
    if profile.to_watch.filter(imdbID=imdb_id).exists():
        profile.to_watch.remove(movie)
        profile.watched.add(movie)
    else:
        profile.watched.add(movie)
    return HttpResponseRedirect(reverse('movie-details',args=[imdb_id]))
```

Цей фрагмент коду містить кілька важливих функцій та логіку для відображення даних фільмів та їх взаємодії з користувачами. Давайте розглянемо кожну частину:

from .forms import RateForm: Імпорт форми для оцінки фільмів.

Імпорт необхідних класів та бібліотек для роботи з Django та зовнішніми API.

Функція landing, відображає головну сторінку веб-сайту, де користувач може шукати фільми за жанром або за ключовими словами.

Функція movieDetails відображає деталі фільму, включаючи його загальну інформацію, оцінки, огляди тощо.

Зберігає інформацію про фільм у базі даних, якщо вона ще не існує.

Функції genres відображає фільми за певним жанром.

Функція Rate дозволяє користувачам оцінювати фільм та залишати відгуки.

Функції watchlist та watched_movies додають фільм до списку переглянутих або списку для перегляду користувача.

Ці функції відображають та обробляють дані про фільми, а також забезпечують взаємодію користувачів зі сторінками фільмів на веб-сайті. Це включає введення оцінок, додавання до списків перегляду тощо.

```
from django.db import models
from django.utils.text import slugify
import requests
from io import BytesIO
from django.core import files
from django.urls import reverse
from django.contrib.auth.models import User
from cloudinary.models import CloudinaryField



# Create your models here.
```

```
class Genre(models.Model):
    title=models.CharField(max_length=25)
    slug=models.SlugField(null=False,unique=True)

    def get_absolute_url(self):
        return reverse('genres', args=[self.slug])

    def __str__(self):
        return self.title

    def save(self, *args,**kwargs):
        if not self.slug:
            self.title.replace(" ","")
            self.slug=slugify(self.title)
        return super().save(*args,**kwargs)

class Rating(models.Model):
    source=models.CharField(max_length=50)
    rating=models.CharField(max_length=10)

    def __str__(self):
        return self.source

class Movie(models.Model):
    Title=models.CharField(max_length=200)
    Year=models.CharField(max_length=25,blank=True)
    Rated=models.CharField(max_length=10,blank=True)
    Released=models.CharField(max_length=25,blank=True)
    Runtime=models.CharField(max_length=25,blank=True)
    Genre=models.ManyToManyField(Genre,blank=True)
```

```
    Director=models.CharField(max_length=100,blank=True)
    Writer=models.CharField(max_length=300,blank=True)
    Actors=models.CharField(max_length=300,blank=True)
    Plot=models.CharField(max_length=900,blank=True)
    Language=models.CharField(max_length=300,blank=True)
    Country=models.CharField(max_length=100,blank=True)
    Awards=models.CharField(max_length=250,blank=True)
    Poster=CloudinaryField('image')
    Poster_url=models.URLField(blank=True)
    imdbID=models.CharField(max_length=100,blank=True)
    Type=models.CharField(max_length=10,blank=True)
    Ratings = models.ManyToManyField(Rating, blank=True)
    Metascore = models.CharField(max_length=5, blank=True)
    imdbRating = models.CharField(max_length=5, blank=True)

    def __str__(self):
        return self.Title

    def save(self,*args,**kwargs):
        if self.Poster=='' and self.Poster_url != '':
            resp=requests.get(self.Poster_url)
            pb=BytesIO()
            pb.write(resp.content)
            pb.flush()
            file_name=self.Poster_url.split("/")[-1]
            #self.Poster.save(file_name,files.File(pb),save=False)
        return super().save(*args,**kwargs)

RATE_CHOICES = [
    (1, '1- Trash'),
```

```
    (2, '2 Terrible'),
    (3, '3 - OK'),
    (4, '4 Good'),
    (5, '5 Perfect'),
]

class Review (models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    movie = models.ForeignKey(Movie, on_delete=models.CASCADE)
    date = models.DateTimeField(auto_now_add=True)
    text = models.TextField(max_length=300, blank=True)
    rate = models.PositiveSmallIntegerField(choices=RATE_CHOICES)

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

Цей фрагмент коду містить моделі для бази даних, які представляють різні аспекти фільмів та оглядів користувачів.

Модель Genre:

title (назва жанру), slug (узагальнена версія назви жанру для URL).

Метод get_absolute_url: повертає URL для конкретного жанру.

Перевизначений метод save генерує slug на основі назви жанру.

Модель Rating:

source (джерело рейтингу), rating (значення рейтингу).

Модель Movie:

Дані про фільм, такі як назва, рік випуску, рейтинг, режисер, актори тощо.

Зв'язок багато-до-багатьох з Genre та Rating.

Перевизначений метод save: завантажує зображення обкладинки фільму, якщо Poster порожній, а Poster_url не порожній.

Модель Review

Дані про огляд, такі як користувач, фільм, дата, текст та оцінка.

Користується варіантами оцінки з RATE_CHOICES.

Ці моделі визначають структуру даних для зберігання інформації про фільми, жанри, оцінки та огляди користувачів.

Отже розроблений програмний продукт повністю задовільняє потребам та завданням що було поставлено перед розробником.

ВИСНОВКИ

У рамках цієї роботи був успішно розроблений веб-кінотеатр з широким спектром функціональних можливостей, що включають в себе пошук і фільтрацію фільмів, можливість реєстрації та авторизації користувачів, а також персоналізовані рекомендації.

Порівняльний аналіз існуючих веб-кінотеатрів та конкурентів дозволив зрозуміти сильні та слабкі сторони нашого продукту і врахувати їх у процесі розробки.

Розроблений веб-кінотеатр має значний потенціал для залучення аудиторії та розвитку, особливо за умови постійного оновлення контенту, покращення користувацького досвіду та розширення функціональності.

Для подальшого успішного розвитку продукту рекомендується зосередитися на покращенні інтерфейсу користувача, вдосконаленні системи рекомендацій та впровадженні нових стратегій маркетингу для залучення нових користувачів.

Дипломна робота успішно виконала поставлені завдання та вивчила ключові аспекти розробки веб-кінотеатру. Результати роботи вказують на потенціал продукту та його можливість конкурувати на ринку онлайн-відео з існуючими платформами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Django. (2024). Documentation. [Online]. Доступно: https://docs.djangoproject.com/en/3.2/ [Дата доступу: Травень 16, 2024].

2. Django - The Web framework for perfectionists with deadlines. (2024). [Online]. Доступно: https://www.djangoproject.com/ [Дата доступу: Травень 16, 2024].

3. Django REST Framework. (2024). Documentation. [Online]. Доступно: https://www.django-rest-framework.org/ [Дата доступу: Травень 16, 2024].

4. Cloudinary. (2024). Django Integration. [Online]. Доступно: https://cloudinary.com/documentation/django_integration [Дата доступу: Травень 16, 2024].

5. OMDB API. (2024). Official Website. [Online]. Доступно: http://www.omdbapi.com/ [Дата доступу: Травень 16, 2024].

6. Real Python. (2024). Django Tutorials. [Online]. Доступно: https://realpython.com/tutorials/django/ [Дата доступу: Травень 16, 2024].

7. Real Python. (2024). Django REST Framework Tutorials. [Online]. Доступно: https://realpython.com/tutorials/django/rest-framework/ [Дата доступу: Травень 16, 2024].

8. Mozilla Developer Network. (2024). HTML Documentation. [Online]. Доступно: https://developer.mozilla.org/en-US/docs/Web/HTML [Дата доступу: Травень 16, 2024].

9. Mozilla Developer Network. (2024). CSS Documentation. [Online]. Доступно: https://developer.mozilla.org/en-US/docs/Web/CSS [Дата доступу: Травень 16, 2024].

10. Mozilla Developer Network. (2024). JavaScript Documentation. [Online]. Доступно: https://developer.mozilla.org/en-US/docs/Web/JavaScript [Дата доступу: Травень 16, 2024].

11. W3Schools. (2024). SQL Tutorial. [Online]. Доступно: https://www.w3schools.com/sql/ [Дата доступу: Травень 16, 2024].

12. Stack Overflow. (2024). Django Tag Questions. [Online]. Доступно: https://stackoverflow.com/questions/tagged/django [Дата доступу: Травень 16, 2024].

13. GitHub. (2024). Django Repository. [Online]. Доступно: https://github.com/django/django [Дата доступу: Травень 16, 2024].

14. GeeksforGeeks. (2024). Django Tutorial. [Online]. Доступно: https://www.geeksforgeeks.org/django-tutorial/ [Дата доступу: Травень 16, 2024].

15. Towards Data Science. (2024). Building a Movie Recommender with Django. [Online]. Доступно: https://towardsdatascience.com/building-a-movie-recommender-with-django-62a2fd5e8988 [Дата доступу: Травень 16, 2024].

16. DigitalOcean. (2024). How To Build a Django Application. [Online]. Доступно: https://www.digitalocean.com/community/tutorial_series/how-to-build-a-django-application [Дата доступу: Травень 16, 2024].

17. FreeCodeCamp. (2024). Django Web Development Tutorials. [Online]. Доступно: https://www.freecodecamp.org/news/tag/django/ [Дата доступу: Травень 16, 2024].

18. YouTube - Corey Schafer. (2024). Django Tutorials. [Online]. Доступно: https://www.youtube.com/playlist?list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p [Дата доступу: Травень 16, 2024].

19. Medium. (2024). Django Articles. [Online]. Доступно: https://medium.com/topic/django [Дата доступу: Травень 16, 2024].

20. Full Stack Python. (2024). Django. [Online]. Доступно: https://www.fullstackpython.com/django.html [Дата доступу: Травень 16, 2024].

21. TechWithTim. (2024). Django Web Development for Beginners. [Online]. Доступно: https://www.youtube.com/playlist?list=PLzMcBGfZo4-kQkZp-j9PNyKq7Yw5VYjq9 [Дата доступу: Травень 16, 2024].

22. Python.org. (2024). Official Python Documentation. [Online]. Доступно: https://www.python.org/doc/ [Дата доступу: Травень 16, 2024].

23. Codecademy. (2024). Learn Python. [Online]. Доступно: https://www.codecademy.com/learn/learn-python [Дата доступу: Травень 16, 2024].

24. TutorialsPoint. (2024). Python Tutorial. [Online]. Доступно: https://www.tutorialspoint.com/python/index.htm [Дата доступу: Травень 16, 2024].

25. Real Python. (2024). Python Programming Tutorials. [Online]. Доступно: https://realpython.com/tutorials/python/ [Дата доступу: Травень 16, 2024].

26. JetBrains. (2024). PyCharm Documentation. [Online]. Доступно: https://www.jetbrains.com/help/pycharm/quick-start-guide.html [Дата доступу: Травень 16, 2024].

27. GitHub - Django Girls Tutorial. (2024). [Online]. Доступно: https://github.com/DjangoGirls/tutorial [Дата доступу: Травень 16, 2024].

28. GitHub - Will Vincent. (2024). Django for Beginners Repository. [Online]. Доступно: https://github.com/wsvincent/djangoforbeginners [Дата доступу: Травень 16, 2024].

29. GitHub - William S. Vincent. (2024). Django for Professionals Repository. [Online]. Доступно: https://github.com/wsvincent/djangoforprofessionals [Дата доступу: Травень 16, 2024].

30. GitHub - CoreyMSchafer. (2024). Django Blog Project. [Online]. Доступно: https://github.com/CoreyMSchafer/code_snippets/tree/master/Django_Blog [Дата доступу: Травень 16, 2024].

31. GitHub - justdjango. (2024). Movie Rating App Repository. [Online]. Доступно: https://github.com/justdjango/movie-ratings [Дата доступу: Травень 16, 2024].

32. GitHub - justdjango. (2024). E-Commerce App Repository. [Online]. Доступно: https://github.com/justdjango/ecommerce [Дата доступу: Травень 16, 2024].

33. GitHub - Real Python. (2024). Django Repository. [Online]. Доступно: https://github.com/realpython [Дата доступу: Травень 16, 2024].

34. GitHub - TestDrivenIO. (2024). Django Rest Framework Repository. [Online]. Доступно: https://github.com/testdrivenio [Дата доступу: Травень 16, 2024].