МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій
Кафедра радіоелектронних і комп’ютерних систем

Допустити до захисту

Завідувач кафедри
_____ доцент Оленич І. Б.
«____»_____ 2023 р.

Курсова робота

Магістр
(освітній ступінь)

Принципи розробки рекомендаційних систем.

Виконав:
Студент V курсу групи ФЕІ-54
спеціальності 121 – Комп’ютерні науки

_____ Паночко І. В.

Науковий керівник:
_____ Бойко Я. В.
«____»_____ 2023 р.

Львів — 2023

АНОТАЦІЯ

Курсова робота студента групи ФеІ-54 Паночка Ігоря на тему «Принципи розробки рекомендаційних систем».

Ця курсова робота присвячена вивченню та дослідженню принципів розробки рекомендаційних систем з використанням підходу спільної фільтрації і бібліотеки surprise. Робота охоплює теоретичний аналіз рекомендаційних систем, зосереджуючись на спільній фільтрації як ефективному методі рекомендацій.

У роботі будуть розглянуті основні концепції і підходи до спільного фільтрування, включаючи базові алгоритми, як метод найближчих сусідів та факторизацію матриць. Буде детально розглянуто реалізацію цих методів з використанням бібліотеки surprise, яка забезпечує потужні інструменти для розробки рекомендаційних систем.

Результати дослідження можуть служити основою для розробки та вдосконалення рекомендаційних систем з використанням бібліотеки surprise та спільної фільтрації, що має значний потенціал для покращення персоналізованого відбору контенту та покупок для користувачів.

ANNOTATION

Course work of the student of FeI-54 group Panochka Ihor on the topic "Principles of development of recommender systems".

This course work is devoted to the study and research of the principles of development of recommender systems using the joint filtering approach and the surprise library. The work covers a theoretical analysis of recommender systems, focusing on collaborative filtering as an effective recommendation method.

The paper will cover the basic concepts and approaches to collaborative filtering, including basic algorithms such as the nearest neighbor method and matrix factorization. The implementation of these methods will be discussed in detail using the surprise library, which provides powerful tools for developing recommender systems.

The results of the study can serve as a basis for the development and improvement of recommender systems using the surprise library and collaborative filtering, which has significant potential to improve personalized content selection and shopping for users.

ЗМІСТ

1

33

44

1 Огляд технологій рекомендаційних систем55

9

1111

1.3 Багатокритеріальні рекомендаційні системи133

1.4 Гібридні рекомендаційні системи.....14

2 ТЕХНОЛОГІЇ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ПРОЕКТУ1414

2.1 Інфраструктура Java144

2.2. Технології Java1616

2.3 Інфраструктура Python……..20

2.4 Технології Python……..22

2.5 Бібліотека Surprise……23

3 Результати роботи та їх обговорення……..25

Висновки……31

Список ЛІТЕРАТУРИ292

ВСТУП

Рекомендаційні системи -- це автоматизовані комп'ютерні програми, які відповідають на запити користувачів у різних контекстах. Такі системи є повсюдними і стали невід'ємною частиною нашого повсякденного життя. Приклади включають рекомендації продуктів користувачам на сайтах, таких як Amazon, рекомендації вмісту користувачам, які відвідують веб-сайт, такі як Yahoo!, рекомендації фільмів користувачам на сайтах типу Netflix, рекомендації у виборі роботи користувачам на сайті LinkedIn тощо. Алгоритми зіставлення побудовані з використанням великих обсягів високочастотних даних, отриманих від минулих взаємодій користувача із різними сервісами. Алгоритми є статистичними за характером і включають проблеми в таких областях, як послідовні процеси прийняття рішень, моделювання взаємодій з дуже багатовимірними категоріальними даними і розробка методів опрацювання масштабованої статистичної інформації. Нові методології в цій сфері вимагають тісної співпраці між вченими в галузі комп'ютерних наук, машинного навчання, статистиками, експертами з оптимізації, системними експертами, і, звичайно, експертами в предметній області. Це одне з найбільш захоплюючих застосувань великих даних. Статистично, система рекомендацій є багатовимірним послідовним процесом, і не менш важливо вивчати такі питання, як розробка експериментів, а також розробка складних статистичних моделей.

Повномасштабні рекомендаційні системи, як правило, потребують для своєї побудови значних обсягів вхідних даних, а також значних обчислювальних ресурсів, які не завжди є доступними. Завданням дипломної роботи є розробка власної спеціалізованої рекомендаційної системи, яка, не володіючи всім функціоналом типових рекомендаційних систем, легка в реалізації, масштабованості та підтримці.

1 ОГЛЯД ТЕХНОЛОГІЙ РЕКОМЕНДАЦІЙНИХ СИТЕМ

Рекомендаційні системи змінили способи взаємодії неживих веб-сайтів зі своїми користувачами. Замість надання статистичної інформації, коли користувачі шукають і, можливо, купують продукти, рекомендаційні системи збільшують ступінь інтерактивності для розширення послуг користувачеві можливостей. Рекомендаційні системи формують рекомендації незалежно для кожного конкретного користувача на основі його минулих покупок і пошуків, а також на основі поведінки інших користувачів. У цій статті описуються рекомендаційні системи і реалізовані в них алгоритми.

Базові підходи

У більшості рекомендаційних систем застосовується один з двох базових підходів: спільне фільтрування (collaborative filtering) і фільтрація на основі вмісту (content-based filtering). Існують також і інші підходи (в тому числі гібридні).

Алгоритми, що використовуються рекомендаційними системами

Відомий підхід, застосований в конкурсі Netflix prize , наочно демонструє, що в рекомендаційних механізмах можуть бути використані самі різні алгоритми. Отримувані результати можуть відрізнятися в залежності від проблеми, для вирішення якої спроектований конкретний алгоритм, і від відносин, які присутні в даних. Багато з цих алгоритмів прийшли з області машинного навчання (яка, в свою чергу, являє собою підобласть штучного інтелекту), яка займається алгоритмами для навчання, прогнозування і прийняття рішень.

Кореляція Пірсона

Подібність між двома користувачами (і їх атрибутами, такими як статті, прочитані в колекції блогів) може бути точно обчислено за допомогою т. Зв. *кореляції Пірсона* . Цей алгоритм вимірює лінійну залежність між двома змінними (або користувачами) як функцію їх атрибутів. Однак він не обчислює цей параметр по всій сукупності користувачів. Цю сукупність необхідно попередньо відфільтрувати до *близьких елементів* на основі високорівневих показників подібності, таких як читання подібних блогів.

Кореляція Пірсона, яка широко застосовується в дослідницькій діяльності, є досить популярним алгоритмом в сфері спільної фільтрації.

Алгоритми кластеризації.

Алгоритми кластеризації - це різновид "спонтанного навчання" (unsupervised learning), що дозволяє виявити структуру в рядах на перший погляд випадкових (або немаркованих) даних. У загальному випадку такий алгоритм базується на виявленні подібності між елементами (наприклад, між читачами блогу) за допомогою обчислення їх відстані від інших елементів в просторі ознак (feature space) (ознакою в просторі ознак може, наприклад, кількість прочитаних статей в наборі блогів). Кількість незалежних ознак визначає розмірність простору ознак. Якщо елементи "близькі" один до одного, то їх можна об'єднати в один кластер.

Існує безліч алгоритмів кластеризації. Найпростішим з них є алгоритм *k* -середній (*k* -means), який розділяє елементи на *k* кластерів. Спочатку елементи розподіляються по цим кластерам в довільному порядку. Потім для кожного кластера обчислюється *центр мас* (або просто *центр*) Як функція його членів. Після цього перевіряється відстань кожного члена кластера від центру цього кластера. Якщо за результатами цієї перевірки член виявляється ближче до іншого кластеру, то він переміщається в цей кластер. Після перевірки всіх відстаней для всіх членів центри кластерів обчислюються заново. При досягненні

стабільного стану (в процесі чергової ітерації члени не транспортувалися) набір вважається кластеризований належним чином, і алгоритм зупиняється.

Обчислення відстані між двома об'єктами може бути важким для візуалізації. Один з поширених методів вирішення цього завдання полягає в тому, щоб розглядати кожен член кластера як багатовимірний вектор і обчислювати для нього т.зв. евклідова відстань.

Існує безліч інших різновидів кластеризації, в тому числі теорія адаптивного резонансу (Adaptive Resonance Theory), нечітка кластеризація методом С-середніх (Fuzzy C-means), імовірнісна кластеризація за допомогою ЕМ-алгоритму (Expectation-Maximization) і т. Д.

інші алгоритми

У рекомендаційних механізмах може застосовуватися безліч алгоритмів (а якщо враховувати варіації, то їх ще більше). Перерахую деякі успішно застосовуються алгоритми.

•**Баєсові мережі довіри (Bayesian Belief Nets)** - візуально можуть бути представлені як орієнтований ациклічний граф, ребра якого являють пов'язані ймовірності змінних.

•**Ланцюги Маркова (Markov chains)** - засновані на такому ж підході, як у Баєсових мережах довіри, але вирішують проблему вироблення рекомендації як послідовну оптимізацію, а не як просте прогнозування.

•**Класифікація за методом Роккі (Rocchio classification)** (заснована на векторній моделі) - використовує відгуки про релевантності елементів для підвищення точності рекомендацій.

Проблеми рекомендаційних систем.

Можливості збору даних, які надає Інтернет, суттєво спростили використання "мудрості натовпу" за допомогою спільної фільтрації. З іншого боку, величезна кількість доступних даних ускладнює реалізацію цієї можливості. Наприклад, поведінка деяких користувачів цілком піддається моделюванню, однак інші користувачі не демонструють типової поведінки. Наявність таких користувачів може призводити до зміщення результатів рекомендаційної системи і до зниження її ефективності. Крім того, користувачі можуть задіяти рекомендаційну систему для підвищення перевагу одного продукту щодо іншого продукту - наприклад, за допомогою відправки позитивних відгуків про одному продукті і негативних відгуків про його конкурентів. Хороша рекомендаційна система зобов'язана справлятися з цими проблемами.

Ще одна проблема, властива великим рекомендаційним системам, пов'язана з масштабністю. Традиційні алгоритми добре працюють з порівняно невеликими обсягами даних, однак з ростом цих наборів отримання результатів на колишньому рівні якості за допомогою традиційних алгоритмів може стати проблематичним.

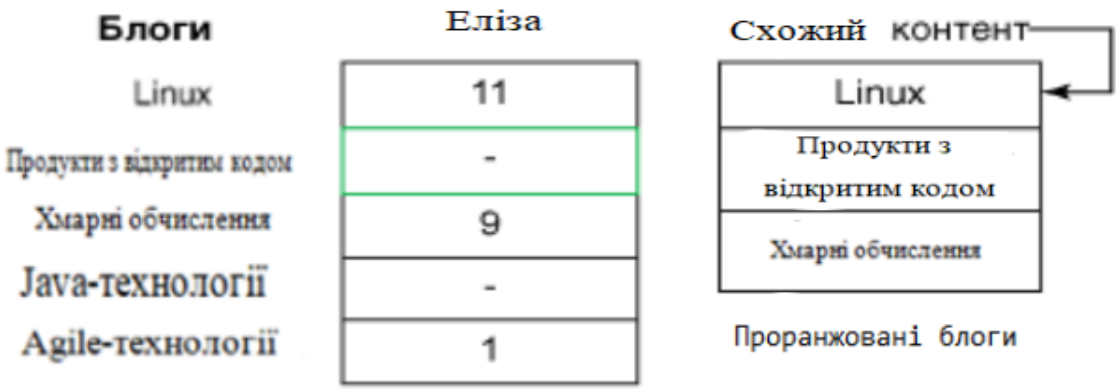
І, нарешті, певні проблеми породжує необхідність дотримання конфіденційності. Рекомендаційні алгоритми здатні розпізнавати такі закономірності, про існування яких люди можуть навіть не підозрювати. Нещодавно приклад такої ситуації мав місце у великій компанії, яка змогла обчислювати індекс прогнозування вагітності на основі купівельних переваг. Після отримання цільових рекламних оголошень батько дочки з подивом дізнався про її вагітність. Система, що прогнозує цієї компанії виявилася настільки точною, що змогла передбачити очікувану дату пологів у майбутньої мами на основі придбаних нею товарів.

1.1 Спільна фільтрація.

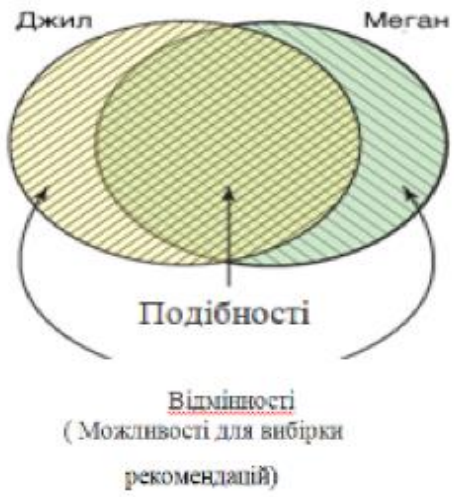
Спільна фільтрація виробляє рекомендації, засновані на моделі попередньої поведінки користувача. Ця модель може бути побудована виключно на основі поведінки даного користувача або - що більш ефективно - з урахуванням поведінки інших користувачів з подібними характеристиками. У тих випадках, коли спільна фільтрація бере до уваги реакцію інших користувачів, вона використовує знання про групу (group knowledge) для вироблення рекомендацій на основі подібності користувачів. По суті рекомендації базуються на автоматичному співпраці безлічі користувачів і на виділення (методом фільтрації) тих користувачів, які демонструють схожі переваги або шаблони поведінки. Як приклад припустимо, що ви створюєте веб-сайт, щоб пропонувати його відвідувачам рекомендації щодо блогів. На основі інформації від багатьох користувачів, які підписуються на блоги і читають їх, ви можете згрупувати цих користувачів по їх перевагам. Наприклад, ви можете об'єднати в одну групу користувачів, які читають кілька одних і тих же блогів. За цією інформацією ви ідентифікуєте найпопулярніші блоги серед тих, які читають учасники цієї групи. Потім - конкретного користувача цієї групи - ви рекомендуєте найпопулярніший блог з тих, на які він ще не підписаний і які він не читає. У таблиці на рис. 1 рядка відповідають набору блогів, а стовпці - користувачам. Осередок на перетині рядка блогу і шпальти користувача містить кількість статей, прочитаних цим користувачем в цьому блозі. Кластеризація користувачів на основі читацьких звичок (наприклад, за допомогою алгоритму *найближчого сусіда*) дозволяє виділити два кластери, кожен з яких містить по два користувача. Зверніть увагу на схожість читацьких звичок у членів кожного кластера: Кластер 1 утворюють учасники з іменами Марк і Еліза, кожен з яких прочитав по кілька статей по тематиці "Linux" і за тематикою "Хмарні обчислення". Кластер 2 утворюють учасники з іменами Меган і Джил, кожен з яких прочитав по кілька статей по тематиці "Java-технології" і за тематикою "

Малюнок 1. Простий приклад спільної фільтрації

Кількість статей, прочитаних даним користувачем



Тепер ви зможете ідентифікувати певні відмінності в рамках кожного кластера і сформувати значущі рекомендації. У кластері 1 Марк прочитав 10 статей з блогу по продуктам з відкритим вихідним кодом, а Еліза не прочитали жодної такої статті; Еліза прочитала одну статтю в блозі по Agile-технологій, а Марк не прочитав жодної такої статті. Таким чином, виходячи з рис.1, Елізі можна порекомендувати блог по продуктам з відкритим вихідним кодом. Для Марка неможливо сформувати ніяких рекомендації, оскільки невелика різниця між ним і Елізою за кількістю прочитань блогу по Agile-технологій з великою ймовірністю буде відфільтрована. У кластері 2 Джил прочитала три статті в блозі по продуктам з відкритим вихідним кодом, а Меган не прочитали жодної такої статті; Меган прочитала 3 статті в блозі по Linux, а Джил не прочитали жодної такої статті. Таким чином, для учасників кластера 2 можна сформувати наступні дві рекомендації: учаснику Джил рекомендується блог по Linux, а учаснику Меган - блог по продуктам з відкритим вихідним кодом. Інший спосіб розгляду цих відносини заснований на їх схожість і відмінності, як ілюструє діаграма Венна на рис. 2. Схожість визначають, за якими ознаками (за допомогою відповідного алгоритму) слід групувати користувачів, що мають схожі інтереси. Відмінності це можливості, які можуть бути використані для вироблення рекомендацій - наприклад, за допомогою застосування фільтра популярності.



Малюнок 2. Схожість і відмінності, які використовуються в спільній фільтрації

Хоча на рис. 2 показана спрощена картина (яка до того ж страждає від розрідженості даних через використання лише двох зразків), таке уявлення досить зручно.

1.2 Фільтрація на основі вмісту.

Тематична фільтрація формує рекомендацію на основі поведінки користувача. Наприклад, цей підхід може використовувати ретроспективну інформацію про перегляди (які блоги читає користувач і характеристики цих блогів). Якщо який-небудь користувач зазвичай читає статті про Linux або регулярно залишає коментарі в блогах з проектування програмного забезпечення, то тематична фільтрація може використовувати цю ретроспективну інформацію для виявлення подібного контенту і пропозиції такого контенту як рекомендованого для цього користувача (статті в блогах по Linux або в інших блогах з проектування програмного забезпечення). Цей контент може бути визначений в ручному режимі або витягнутий автоматично на основі інших методів подібності.

Повернемося до рис.1 і розглянемо користувача з ім'ям Еліза. Припустимо, що ви застосовуєте ранжування блогів, яке вказує, що користувачі, які читали статті по Linux, можуть зацікавитися хмарними обчисленнями і продуктами з відкритим вихідним кодом. В цьому випадку ви зможете з легкістю порекомендувати Елізе -

на основі її поточних читацьких звичок - блог по продуктам з відкритим вихідним кодом. Цей підхід, проілюстрований на рис. 3, спирається виключно на контент, до якого звертається один користувач, а не на реакцію інших користувачів в системі.

Малюнок 3. Ранжируваний список відмінностей, який використовується в тематичній фільтрації

Діаграму Венна (див. рис.2 можна застосувати і в цьому випадку: якщо одна сторона - це користувач Еліза, а інша - це ранжируваних набір подібних блогів, то подібності ігноруються (оскільки Еліза вже прочитала відповідні блоги), а ранжирування відмінності створюють можливості для вироблення рекомендацій.

1.3 Багатокритеріальні рекомендаційні системи

Зрозуміло, що видача всіх цих рекомендацій оператору для вибору якогось варіанту для прийняття рішення, не має ніякого сенсу, так як оператору буде необхідно постійно аналізувати співвідношення трьох показників якості та двох критеріїв обмеження. При цьому покращення одного показника, як правило, викликає погіршення іншого. Також необхідно врахувати, що для дифузійної станції цукрового заводу характерною ознакою є використання різних одиниць і різних шкал для показників якості роботи дифузійної станції. Крім того, наявні безрозмірні показники роботи, наприклад, переміщення стружки в приладі. У зв'язку з цим необхідно провести нормалізацію показників роботи дифузійної станції. Тому була використана формула повної нормалізації критеріїв:

$$f_i(x) = \frac{\overline{f_i(x)} - \min\limits_{x \in X} \overline{f_i(x)}}{\max\limits_{x \in X} \overline{f_i(x)} - \min\limits_{x \in X} \overline{f_i(x)}} \qquad (1)$$

В якості критеріїв були використані показники роботи дифузійної станції. Дана нормалізація відображає вихідні значення показника якості роботи дифузійної станції для наведених вище рекомендацій. Умовою вирішення даної задачі є знаходження таких рішень в процесі вибору рекомендації (з врахуванням множини показників), при яких показники роботи дифузійної станції матимуть максимальне значення:

$$\langle f_1(x), f_2(x), \ldots, f_i(x) \rangle \to \max_{x \in X}. \qquad (3)$$

На рис.3 наведене графічне представлення нормалізованих показників роботи для вибору рекомендацій .На практиці, для дифузійної станції неможливо знайти таке рішення, при якому всі показники роботи одночасно матимуть максимальне значення.

1.4 Гібридні рекомендаційні системи.

Гібридні підходи , які поєднують спільну фільтрацію та фільтрацію на основі вмісту, також підвищують ефективність (і складність) рекомендаційних систем. Простий приклад гібридної системи міг би використовувати підходи, показані на рис.1 і на рис.3 об'єднання результатів спільної фільтрації та фільтрації на основі вмісту потенційно дозволяє підвищити точність рекомендації. Крім того, гібридний підхід може бути корисний, якщо застосування спільної фільтрації починається при значній розрідженості даних (т. Н. *Холодний старт*). Гібридний підхід дозволяє спочатку зважувати результати згідно фільтрації на основі змісту, а потім зміщувати ці ваги у напрямку до спільної фільтрації (в міру "визрівання" доступного набору даних по конкретному користувачеві).

2 СЕРЕДОВИЩЕ РОЗРОБКИ

2.1 Інфраструктура Java

Мова Java розроблений так, щоб забезпечувати максимальну переносимість. Багато аспектів Java визначаються відразу для всіх можливих реалізацій. Наприклад, тип int завжди повинен являти собою 32-розрядний ціле зі знаком з доповненням по модулю 2. У багатьох мовах програмування точні визначення типів є долею конкретної реалізації; на рівні мови даються лише загальні гарантії, такі як мінімальний діапазон чисел даного типу або можливість системного запиту, що дозволяє визначити діапазон на цій платформі.

У мові Java такі вимоги просунуті аж до рівня машинної мови, на який транслюється текст програм. Оригінальний текст програми на Java компілюється в *байт-код* , що виконується на *віртуальній машині Java* . Байт-код є універсальною мовою, і саме він інтерпретується віртуальною машиною в кожній системі, яка підтримує Java. Зрозуміло, віртуальна машина Java може бути реалізована і на апаратному рівні - тобто з використанням спеціалізованих

мікросхем. Це ніяк не впливає на переносимість байт-коду і є всього лише одним з видів реалізації віртуальної машини.

Віртуальна машина присвоює кожному з додатком власний *контекст часу виконання (runtime)* , який одночасно ізолює додатки один від одного і забезпечує безпеку роботи. Менеджер безпеки кожного контексту часу виконання визначає, які можливості доступні даному додатку. Наприклад, менеджер безпеки може заборонити додатком читання або запис на локальний диск або обмежити мережеві з'єднання строго визначеними комп'ютерами.

У сукупності всі ці кошти роблять мову Java повністю крос-платформною і надають схему безпеки для виконання переданого по мережі коду на різних рівнях довіри (trust levels). Оригінальний текст Java, скомпільований в байт-код Java, може виконуватися на будь-якому комп'ютері, де є віртуальна машина Java. Код може виконуватися на відповідному рівні захисту, щоб запобігти випадковому або зловмисне пошкодження системи. рівень довіри

регулюється в залежності від джерела байт-коду - байт-код на локальному диску або в

захищеної мережі користується великою довірою, ніж байт-код, отриманий з віддаленого комп'ютера, невідомо навіть де розташованого.

Java містить ряд інших можливостей, які коротко згадуються тут і розглядаються в наступних розділах:

Потоки: Java володіє вбудованою підтримкою потоків для створення багатопоточних додатків. Для синхронізації паралельного доступу до об'єктів і даних класу використовується блокування на рівні об'єктів і на рівні класів.

Класи загального призначення: до складу Java входять класи, що представляють багато примітивних типи даних (такі, як Integer, Double і Boolean), а також клас Class для роботи з різними типами класів.

Допоміжні класи і інтерфейси: Java містить пакет java.util з безліччю корисних класів - таких, як BitSet, Vector, Stack і Date. Більш докладно про допоміжних класах

2.2. Технології Java

Spring MVC

Фреймворк **Spring MVC** забезпечує архітектуру паттерна Model - View - Controller (Модель - Відображення (далі - Вид) - Контролер) за допомогою слабо пов'язаних готових компонентів. Патерн MVC розділяє аспекти додатки (логіку введення, бізнес-логіку і логіку UI), забезпечуючи при цьому вільну зв'язок між ними.

- **Model** (Модель) інкапсулює (об'єднує) дані програми, в цілому вони будуть складатися з POJO («Старих добрих Java-об'єктів», або бінів).
- **View** (Відображення, Вид) відповідає за відображення даних Моделі, - як правило, генеруючи HTML, які ми бачимо в своєму браузері.
- **Controller** (Контролер) обробляє запит користувача, створює відповідну Модель і передає її для відображення в Вид.

DispatcherServlet

Вся логіка роботи Spring MVC побудована навколо DispatcherServlet , який приймає і обробляє всі HTTP-запити (з UI) і відповіді на них.

Після отримання HTTP-запиту DispatcherServlet звертається до інтерфейсу **HandlerMapping** , який визначає, який Контролер повинен бути викликаний, після чого, відправляє запит в потрібний Контролер.

- Контролер приймає запит і викликає відповідний службовий метод, заснований на GET або POST. Викликаний метод визначає дані Моделі,

засновані на певній бізнес-логікою і повертає в DispatcherServlet ім'я Віда (View).

●За допомогою інтерфейсу **ViewResolver** DispatcherServlet визначає, який Вид потрібно використовувати на підставі отриманого імені.

●Після того, як Вид (View) створений, DispatcherServlet відправляє дані Моделі у вигляді атрибутів в Вид, який в кінцевому підсумку відображається в браузері.

Всі вищезгадані компоненти, а саме, HandlerMapping , Controller і ViewResolver , є частинами інтерфейсу WebApplicationContext наслідує ApplicationContext , з деякими додатковими особливостями, необхідними для створення web-додатків.

Визначення Контролера

DispatcherServlet відправляє запит контролерам для виконання певних функцій. Анотація @Controllerannotation вказує, що конкретний клас є контролером. Анотація @RequestMapping використовується для мапінг (зв'язування) з URL для всього класу або для конкретного методу обробника.

Анотація @Controller визначає клас як Контролер Spring MVC. У першому випадку, @RequestMapping вказує, що всі методи в даному контролери відносяться до URL-адресою "/ hello". Наступна анотація @RequestMapping (method = RequestMethod.GET) використовується для оголошення методу **printHello ()** як дефолтного методу для обробки HTTP-запитів GET (в даному контролер). Ви можете визначити будь-який інший метод як обробник всіх POST-запитів з даного URL-адресою.

Hibernate

Hibernate — засіб відображення між об'єктами та реляційними структурами (object-relational mapping, ORM) для платформи Java. Hibernate є вільним програмним забезпеченням, яке поширюється на умовах GNU Lesser General Public License. Hibernate надає легкий для використання каркас (фреймворк) для відображення між об'єктно-орієнтованою моделлю даних і традиційною реляційною базою даних.

Метою Hibernate є звільнення розробника від значних типових завдань із програмування взаємодії з базою даних. Розробник може використовувати Hibernate як при розробці з нуля, так і для вже існуючої бази даних.

Hibernate піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних SQL), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних.

Hibernate забезпечує використання SQL-подібної мови Hibernate Query Language (HQL), яка дозволяє виконувати SQL-подібні запити, записані поряд з об'єктами даних Hibernate. *Запити критеріїв* надаються як об'єктно-орієнтована альтернатива до HQL.

JPA

JPA (Java Persistence API) це специфікація Java EE і Java SE, що описує систему управління збереженням java об'єктів в таблиці реляційних баз даних в зручному вигляді. Сама Java не містить реалізації JPA, проте існує багато реалізацій даної специфікації від різних компаній (відкритих і немає). Це не єдиний спосіб збереження java об'єктів в бази даних (ORM систем), але один з найпопулярніших в Java світі.

Spring Security

Spring Security це Java / JavaEE framework, що надає механізми побудови систем аутентифікації та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework. Проект був розпочатий Беном Алексом (Ben Alex) в кінці 2003 року під ім'ям «Acegi Security», перший реліз вийшов в 2004 році. Згодом проект був поглинений Spring'ом і став його офіційним дочірнім проектом. Вперше публічно представлений під новим ім'ям Spring Security 2.0.0 в квітні 2008 року. Ключові об'єкти контексту Spring Security:

SecurityContextHolder, в ньому міститься інформація про поточний контексті безпеки програми, який включає в себе детальну інформацію про користувача (Principal) працює в даний час з додатком. За замовчуванням SecurityContextHolder іспользуетThreadLocal для зберігання такої інформації, що означає, що контекст безпеки завжди доступний для методів виконуються в тому ж самому потоці. Для того що б змінити стратегію зберігання цієї інформації можна скористатися статичним методом класу SecurityContextHolder.setStrategyName (String strategy). Більш докладно SecurityContextHolder

- SecurityContext, містить об'єкт Authentication і в разі необхідності інформацію системи безпеки, пов'язану із запитом від користувача.
- Authentication представляє користувача (Principal) з точки зору Spring Security.
- GrantedAuthority відображає дозволу видані користувачу в масштабі всього програми, такі дозволи (як правило називаються «ролі»), наприклад ROLE_ANONYMOUS, ROLE_USER, ROLE_ADMIN.
- UserDetails надає необхідну інформацію для побудови об'єкта Authentication з DAO об'єктів додатки або інших джерел даних системи безпеки. Об'єкт UserDetailscодержіт ім'я користувача, пароль, прапори: isAccountNonExpired,

isAccountNonLocked, isCredentialsNonExpired, isEnabled і Collection - прав (ролей) користувача.

- UserDetailsService, використовується щоб створити UserDetails об'єкт шляхом реалізації єдиного методу цього інтерфейсу

аутентифікація

(1) Користувачеві буде запропоновано увійти в систему надавши ім'я (логін або email) і пароль. Ім'я користувача і пароль об'єднуються в екземпляр класу UsernamePasswordAuthenticationToken (екземпляр інтерфейсу Authentication) після чого він передається примірнику AuthenticationManager для перевірки.
(2) У випадку якщо пароль не збігається з назвою користувача буде викинуто виключення BadCredentialsException з повідомленням "Bad Credentials".
(3) Якщо аутентифікація пройшла успішно повертає повністю заповнений примірник Authentication.
(4) Для користувача встановлюється контекст безпеки шляхом виклику методу SecurityContextHolder.getContext (). SetAuthentication (...), куди передається об'єкт який повернув AuthenticationManager.

2.3. Інфраструктура Python

У веб-розробці Python займає важливе місце, використовуючи свою потужну інфраструктуру для створення веб-додатків та розширень. У цьому розділі ми розглянемо основні компоненти та інструменти, які стануть незамінними при розробці веб-додатків з використанням Python.

Веб-сервери:

Веб-сервер є одним з основних компонентів інфраструктури Python для веб-розробки. Існують різні веб-сервери, які можна використовувати, такі як Gunicorn, uWSGI, CherryPy, Tornado та багато інших. Вони забезпечують взаємодію між веб-додатком і Інтернетом, обробляючи запити від клієнтів та надсилаючи їх до програми-додатку Python.

Фреймворки:

Фреймворки є невід'ємною частиною веб-розробки на Python. Вони надають зручні та потужні інструменти для швидкої розробки веб-додатків. Найпопулярніші фреймворки Python для веб-розробки включають Django, Flask та Pyramid. Вони забезпечують структуру та організацію проекту, реалізацію маршрутизації, обробку запитів та багато іншого.

Шаблонізатори:

Шаблонізатори дозволяють вбудовувати динамічний контент у статичні HTML-сторінки. Вони допомагають розділити логіку від представлення та спрощують розробку. Декілька популярних шаблонізаторів Python включають Jinja2, Django Templates та Mako.

Бази даних:

Бази даних є необхідними для зберігання та управління даними у веб-додатках. Python має багато інструментів та бібліотек для роботи з різними типами баз даних, такими як PostgreSQL, MySQL, SQLite та іншими. SQLAlchemy є однією з популярних ORM (Object-Relational Mapping) бібліотек, яка спрощує взаємодію з базами даних.

Управління залежностями:

У веб-розробці Python використовуються залежності - зовнішні бібліотеки та модулі, які використовуються в проекті. Для керування цими залежностями можна використовувати інструменти, такі як pip, poetry або conda. Вони дозволяють легко встановлювати, оновлювати та управляти залежностями проекту.

Тестування:

Тестування є важливою частиною розробки веб-додатків. Python має широкий вибір фреймворків для тестування, таких як unittest, pytest, nose і майже всі популярні фреймворки веб-розробки мають свої власні засоби для тестування.

2.4 Технології Python

Веб-розробка та штучний інтелект є двома сильно зростаючими галузями, і використання Python у цих сферах стає все більш популярним. У цьому пункті курсової роботи ми розглянемо, як можна використовувати технології Python для веб-розробки та штучного інтелекту зокрема для рекомендаційних систем.

Фреймворки для веб-розробки

Python має кілька потужних фреймворків для веб-розробки, які дозволяють створювати швидкі та масштабовані веб-додатки. Два найпопулярніші фреймворки - Django та Flask - надають розробникам зручні інструменти для створення веб-додатків різного рівня складності. Вони мають вбудовану підтримку для маршрутизації, роботи з базами даних, шаблонів та багато іншого, що робить їх ідеальними виборами для веб-розробки.

Бібліотеки для рекомендаційних систем

Python також має ряд потужних бібліотек для створення рекомендаційних систем. Дві найвідоміші бібліотеки - Surprise та LightFM - надають розробникам можливість легко реалізувати різні алгоритми рекомендаційних систем, таких як колаборативне фільтрування та рекомендації на основі контенту. Ці бібліотеки забезпечують зручний інтерфейс для завантаження даних, побудови моделей та отримання рекомендацій.

Машинне навчання та глибоке навчання

Python є популярним вибором для реалізації алгоритмів машинного навчання та глибокого навчання. Бібліотеки, такі як scikit-learn, TensorFlow та PyTorch,

надають розробникам різноманітні інструменти для створення та навчання моделей машинного навчання та нейронних мереж. Це дозволяє розробникам створювати складні рекомендаційні системи, які враховують індивідуальні уподобання користувачів та здатні покращувати свою продуктивність з часом.

Технології Python веб-розробки та штучного інтелекту стали незамінними інструментами для реалізації рекомендаційних систем. Використання фреймворків для веб-розробки, таких як Django та Flask, дозволяє розробникам створювати потужні веб-додатки, а бібліотеки для рекомендаційних систем допомагають побудувати ефективні та персоналізовані рекомендації для користувачів. Машинне навчання та глибоке навчання з використанням Python додають можливості для створення більш складних та точних рекомендаційних систем. Застосування цих технологій в сфері рекомендаційних систем може значно покращити користувальницький досвід та підвищити ефективність бізнесу.

2.5 Бібліотека Surprise

Бібліотека Surprise є потужним інструментом для реалізації рекомендаційних систем з використанням спільного фільтрування в середовищі Python. Вона надає зручний інтерфейс та високорівневі методи для побудови та оцінки різних алгоритмів рекомендацій.

Основні особливості бібліотеки Surprise:

Реалізація різних алгоритмів: Surprise має вбудовану підтримку для різних алгоритмів рекомендацій, включаючи спільне фільтрування, яке базується на спільних уподобаннях користувачів або товарів. Вона включає такі алгоритми, як SVD (Singular Value Decomposition), KNN (k-Nearest Neighbors), а також більш складні моделі, такі як множинний факторизований розподіл (Multiple Factorization Machines) та багато інших.

Зручний інтерфейс: Surprise пропонує простий і зрозумілий інтерфейс для завантаження та обробки даних. Вона надає можливість завантажувати дані з різних форматів, включаючи CSV, Pandas DataFrame або навіть звичайний текстовий файл. Крім того, Surprise надає методи для розбиття даних на тренувальний і тестовий набори, а також для оцінки та порівняння різних моделей.

Налаштування параметрів: Surprise дозволяє розробникам налаштовувати параметри різних алгоритмів рекомендацій. Це включає вибір кількості факторів для факторизованих моделей, налаштування метрик схожості для KNN-підходу та інші параметри, які впливають на якість рекомендацій.

Оцінка та порівняння моделей: За допомогою Surprise можна легко оцінити різні моделі рекомендацій за допомогою різних метрик, таких як середня квадратична помилка (Mean Squared Error), середня абсолютна помилка (Mean Absolute Error) та інші. Це дозволяє порівняти різні моделі та вибрати найкращу для конкретного завдання рекомендацій.

3. Результати роботи та їх обговорення

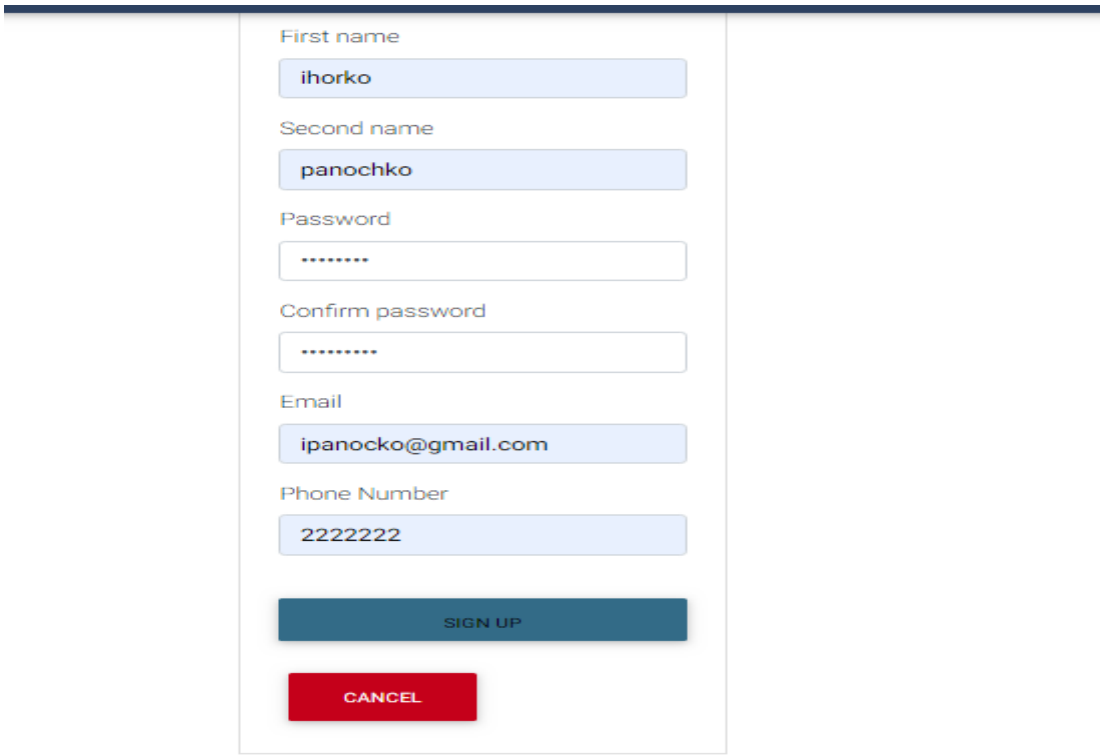


Рис.3.1 Форма для реєстрації користувача.

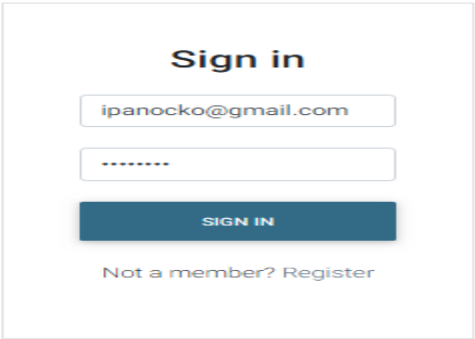


Рис 3.2 Форма для логування користувача.

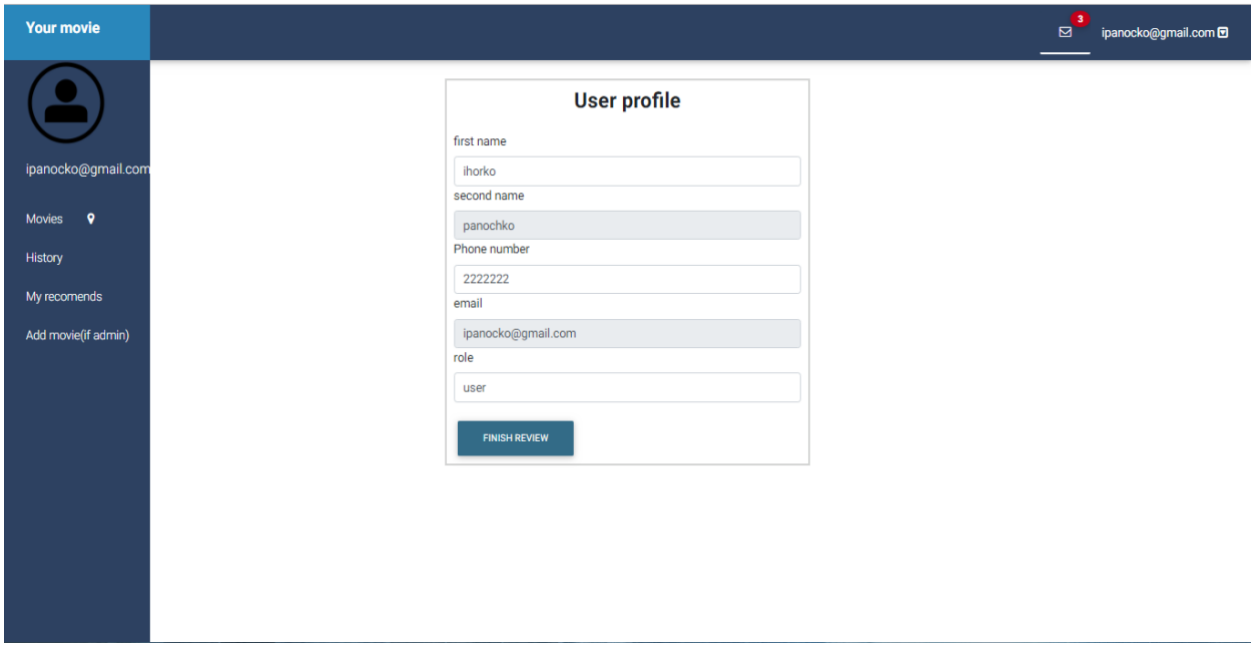


Рис 3.3 Сторінка профілю користувача.

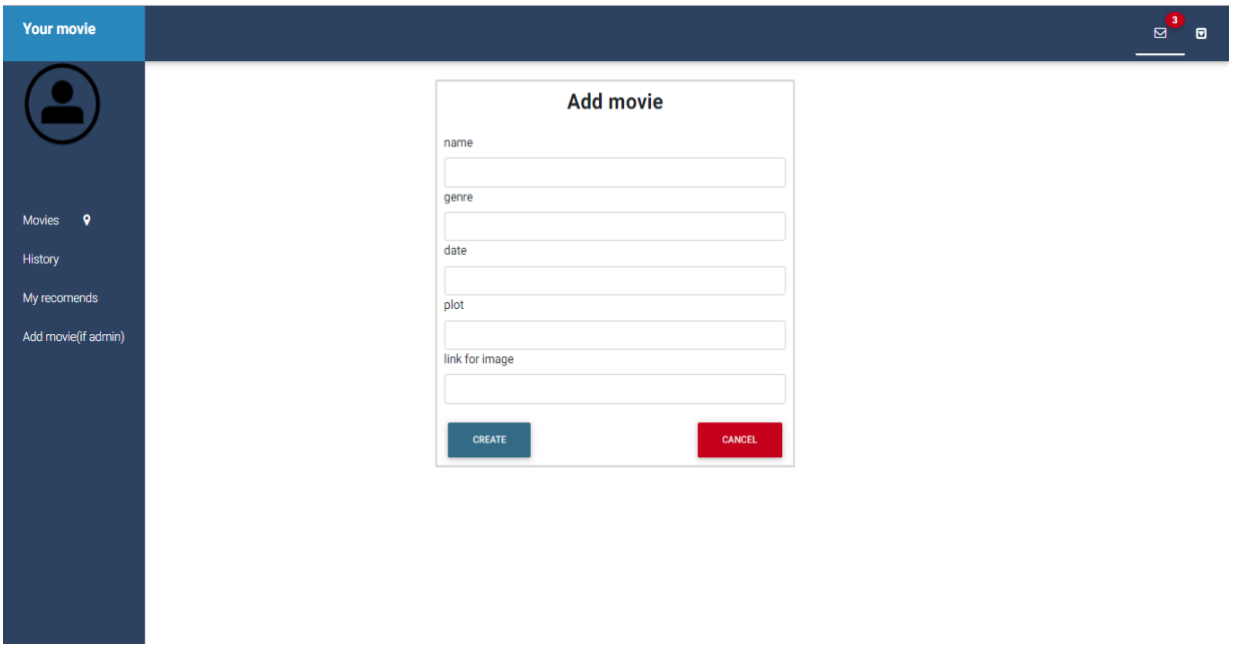


Рис 3.4 Сторінка для додавання фільмів в БД.

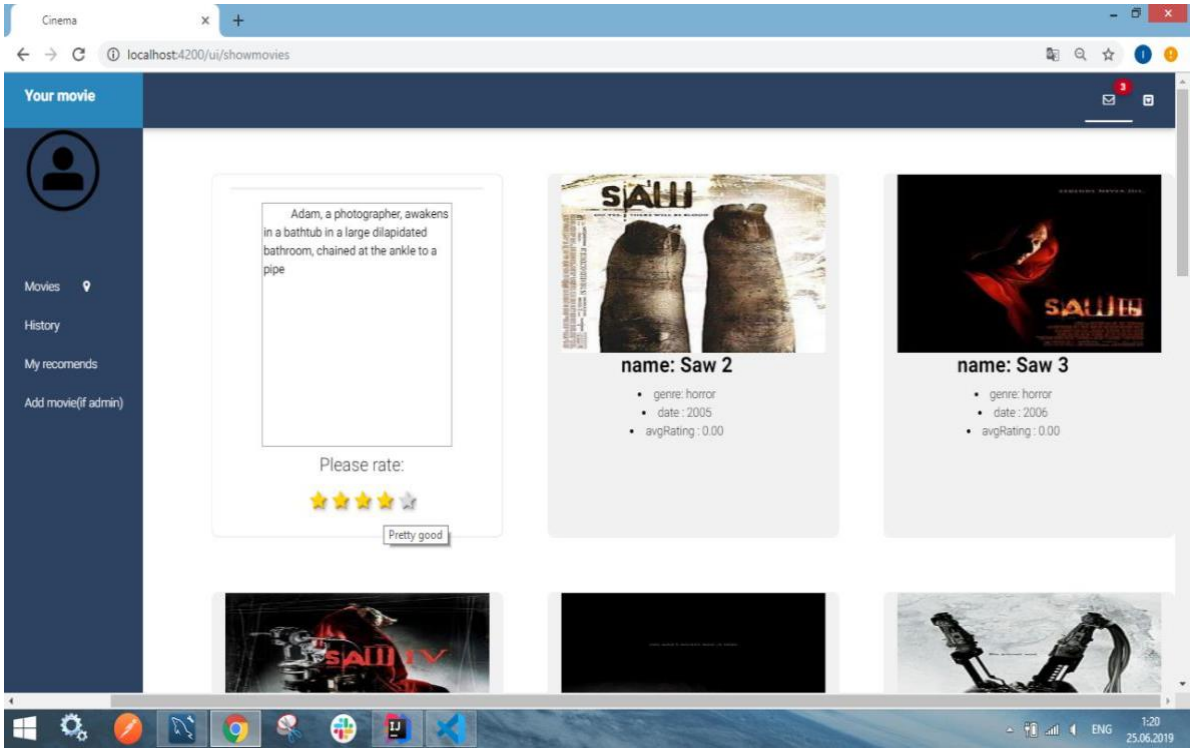


Рис 3.5 Сторінка з можливістю оцінювання.

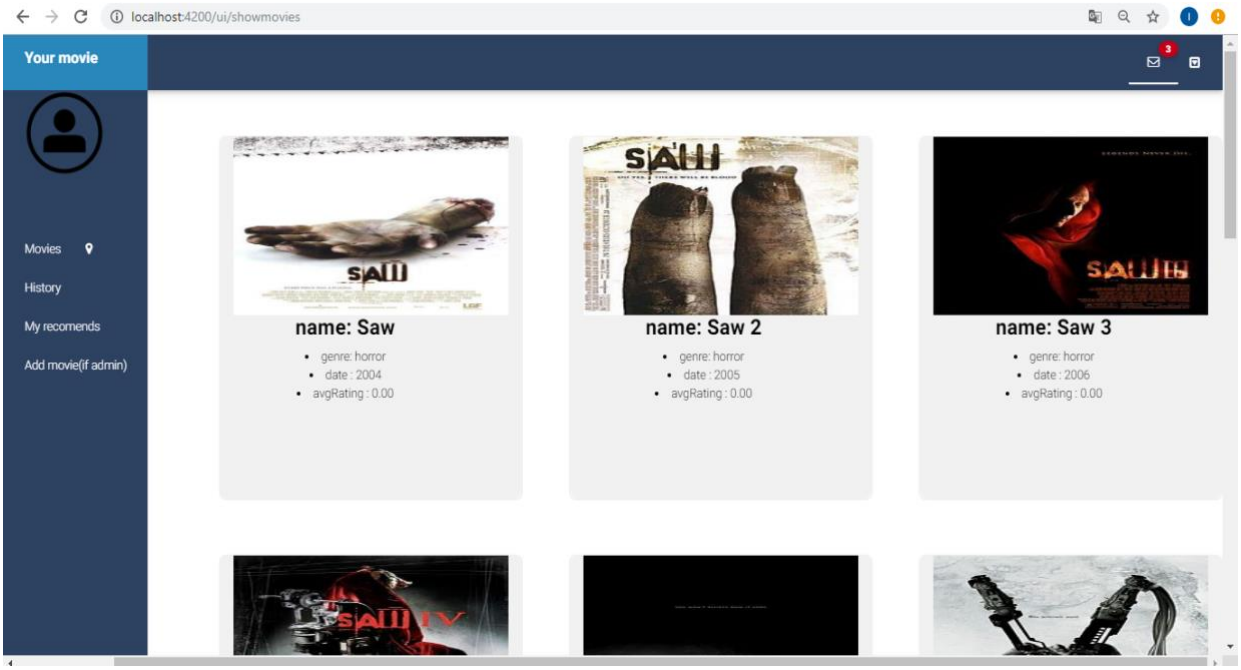


Рис 3.6 Сторінка з списком всіх фільмів.

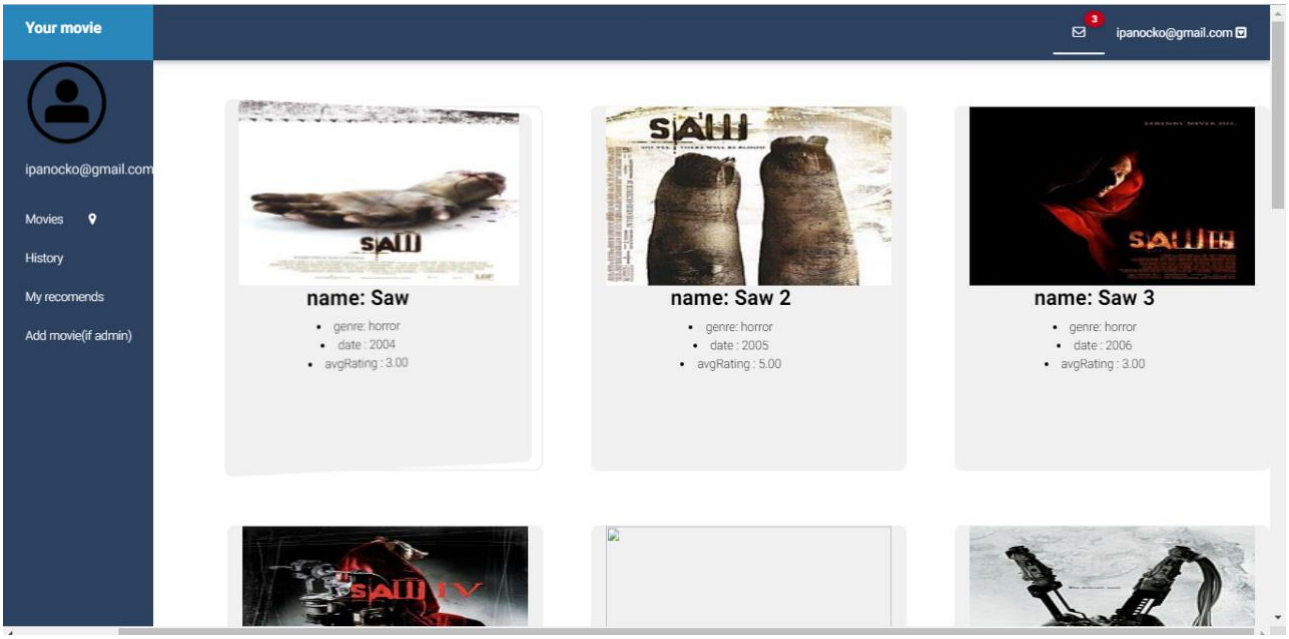


Рис 3.7 Сторінка з списком переглянутих фільмів.

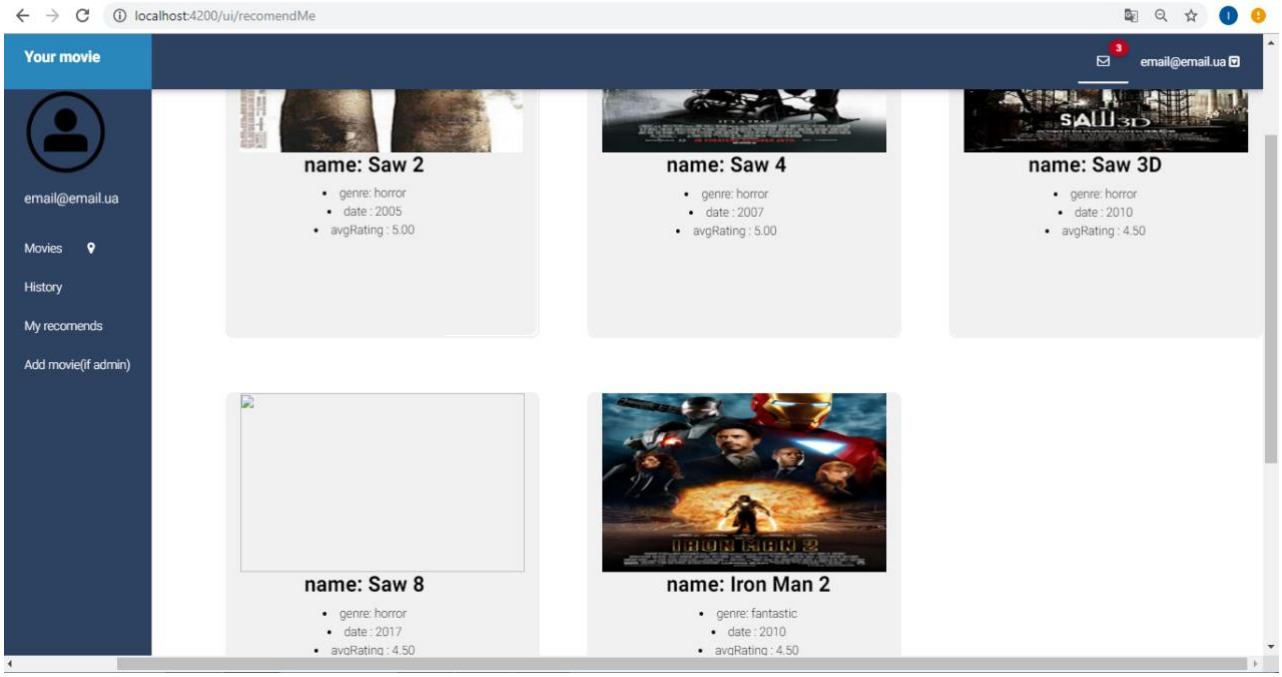


Рис 3.8 Сторінка з рекомендованими фільмами.

```
1 •  SELECT * FROM cinema.movie;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	avg_rating	date	genre	image_link	name	plot
1	0	2004	horror	https://u...	Saw	Ada...
2	0	2005	horror	https://u...	Saw 2	nfor...
3	0	2006	horror	https://u...	Saw 3	Aft...
4	0	2007	horror	https://u...	Saw 4	A w...
NULL	NULL	NULL	NULL	NULL	NULL	NULL

movie 1

Рис 3.10 Відображення фільмів в БД.

Result Grid | Filter Rows: | Edit: | Expo

id	create_date	created_by	item_id	rating	timestamp	user_id
1	NULL	NULL	1	4	964982703	1
2	NULL	NULL	3	4	964981247	1
3	NULL	NULL	6	4	964982224	1
4	NULL	NULL	47	5	964983815	1
5	NULL	NULL	50	5	964982931	1
6	NULL	NULL	70	3	964982400	1
7	NULL	NULL	101	5	964980868	1
8	NULL	NULL	110	4	964982176	1
9	NULL	NULL	151	5	964984041	1
10	NULL	NULL	157	5	964984100	1
11	NULL	NULL	163	5	964983650	1
12	NULL	NULL	216	5	964981208	1
13	NULL	NULL	223	3	964980985	1
14	NULL	NULL	231	5	964981179	1
15	NULL	NULL	235	4	964980908	1
16	NULL	NULL	260	5	964981680	1

Рис 3.11 Відображення оцінок в БД.

```
algo = NMF()
algo.fit(data.build_full_trainset())

my_recs = []
for iid in movies_to_predict:
    my_recs.append((iid, algo.predict(uid=1001, iid=iid).est))

pd.DataFrame(my_recs, columns=['iid', 'predictions']).sort_values('predictions', ascending=False).head(10)
```

Рис 3.12 Фрагмент коду з використанням бібліотеки Surprise для отримання рекомендація для користувача з id:1001

	iid	predictions
547	Once Were Warriors (1994)	4.246192
30	All About Eve (1950)	4.043059
543	On Golden Pond (1981)	3.968875
770	True Romance (1993)	3.850275
399	Jaws (1975)	3.841900
667	Singin' in the Rain (1952)	3.841563
99	Blade Runner (1982)	3.835287
141	Casablanca (1942)	3.815703
661	Shawshank Redemption, The (1994)	3.811728
646	Secret of Roan Inish, The (1994)	3.800933

Рис. 3.13 Результат виконання рекомендаційної системи для користувача, де iid індифікатор фільму, а predictions рівень рекомендації.

ВИСНОВКИ

1. У дипломній роботі проаналізовано розроблені підходи до побудови повнофункціональних рекомендаційних систем і сфери їх застосування.

2. Проведено аналіз різних програмних засобів для створення власних спеціалізованих реалізацій рекомендаційних систем для роботи в галузі ритейлу.

3. В якості прикладу використання спеціалізованої рекомендаційної системи реалізовано проект Інтернет-магазину з продажу мультимедійної продукції з інтеграцією бази даних користувачів. Розроблено алгоритм рекомендаційної системи та імплементовано його на основі технологій Java та Python.

4. Проведено тестування створеної системи, яке показало її придатність для практичного використання.

Список Літератури

1. Spring security: https://habr.com/ru/post/203318/

2.Spring Boot:https://habr.com/ru/post/333756/

3. Creator project :https://start.spring.io/

4. Angular Framework:
https://www.tutorialspoint.com/angular6/angular6_overview.htm

5. Spring MVC: https://habr.com/ru/post/336816/

6.Як працюються рекомендаційні системи:
https://habr.com/ru/company/yandex/blog/241455/

https://habr.com/ru/company/lanit/blog/420499/

https://logic.pdmi.ras.ru/~sergey/teaching/mlstc12/15-recommender.pdf

https://colab.research.google.com/github/singhsidhukuldeep/Recommendation-System/blob/master/Building_Recommender_System_with_Surprise.ipynb#scrollTo=us5FjYjd7Vls

7.Анатомія рекомендаційних систем: https://habr.com/ru/company/lanit/blog/420499/