

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«__» _____ 20__ р.

**Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та
технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Вебзастосунок онлайн магазину з реалізацією алгоритму
рекомендації товарів»**

Виконав:

студент IV курсу, групи ІС-92

Чечет Микита Олексійович _____

Керівник:

Доцент кафедри ІСТ, к.ф.-м.н

Жереб Костянтин Анатолійович _____

Рецензент:

Посада, науковий ступінь, вчене звання

Прізвище, ім'я, по батькові _____

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«__» _____ 20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Чечету Микиті Олексійовичу

1. Тема проєкту «Вебзастосунок онлайн магазину з реалізацією алгоритму рекомендації товарів», керівник проєкту Жереб Костянтин Анатолійович, к.ф.-м.н., затверджені наказом по університету від «31» травня 2023 р. №2101-с.
2. Термін подання студентом проєкту: 12 червня 2023 року.
3. Вихідні дані до проєкту: мова програмування Javascript, мова програмування TypeScript, платформа Angular, середовище розробки JetBrains IntelliJ IDEA.
4. Зміст пояснювальної записки: проаналізувати існуючі алгоритми рекомендацій, вибрати технології розробки та визначити особливості реалізації, описати структури та елементи програмного забезпечення, розробити керівництво користувача, провести тестування.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників): діаграма послідовностей, діаграма станів, діаграма компонентів, діаграма потоків даних.

Календарний план

з/п	Назва етапів виконання проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	11.02.23	
2	Аналіз наявних рекомендаційних систем у вигляді компоненту в існуючих рішеннях	17.04.23 – 23.04.23	
3	Аналіз алгоритмів фільтрації та способів знаходження схожих елементів	24.04.23 – 30.04.23	
4	Проектування програмного забезпечення	01.05.23 – 07.05.23	
5	Розробка програмного продукту	08.05.23 – 14.05.23	
6	Впровадження системи та тестування продукту	15.05.23 – 21.05.23	
7	Оформлення текстової документації	22.05.23 – 06.06.23	

Студент

Микита ЧЕЧЕТ

Керівник

Костянтин ЖЕРЕБ

АНОТАЦІЯ

Чечет М. О. Вебзастосунок онлайн магазину з реалізацією алгоритму рекомендації товарів. КПІ ім. Ігоря Сікорського, Київ, 2023

Проект містить 4 розділи, 70 сторінок, 19 рисунків, 2 таблиці, 4 кресленики, 25 джерел.

Ключові слова: Angular, collaborative-filtering, content-based filtering, SVD++, machine learning, pattern, singular value decomposition, stochastic gradient descent, рекомендаційні системи.

Об'єктом дослідження є веб застосунок онлайн магазину.

Метою розробки є реалізація покращеної варіації алгоритму рекомендації товарів на основі сингулярного розкладу матриць та побудова сучасного веб застосунку на основі фреймворку Ангуляр для підвищення комерційної спроможності магазину. Для реалізації системи було використано: мова програмування JavaScript та Typescript, бібліотеки RxJS, NgRx, Angular Material, фреймворк Angular, середовище розробки IntelliJ IDEA.

У дипломному проєкті було реалізовано систему машинного навчання на базі покращеного алгоритму сингулярного розкладу матриць SVD++, яка на основі вже наданих оцінок користувачів прогнозує наступний товар для рекомендації.

Дана розробка в подальшому може бути розвинена до повноцінного веб магазину із повним набором функціоналу та використана у комерційній діяльності.

SUMMARY

Chechet M. O. Web application of an online store with the implementation of the product recommendation algorithm. KPI named after Igor Sikorskyi, Kyiv, 2023

The project contains 4 chapters, 72 pages, 19 figures, 2 tables, 4 drawings, 25 sources.

Keywords: Angular, collaborative-filtering, content-based filtering, SVD++, machine learning, pattern, singular value decomposition, stochastic gradient descent, recommender systems.

The object of development is a web application of an online store with the implementation of a product recommendation algorithm.

The purpose of the development is the implementation of an improved variation of the algorithm for recommending goods based on the singular matrix distribution and the construction of a modern web application based on the Angular framework to increase the commercial capacity of the store. The following were used to implement the system: JavaScript and Typescript programming languages, RxJS, NgRx, Angular Material, Angular framework, IntelliJ IDEA development environment.

The diploma project implemented a machine learning system based on the improved SVD++ matrix singular decomposition algorithm, which predicts the next product for recommendation based on already provided user ratings.

In the future, this development can be developed into a full-fledged web store with a full set of functions and used in commercial activities.

**Пояснювальна записка
до дипломного проєкту
на тему: «Вебзастосунок онлайн магазину з
реалізацією алгоритму рекомендації товарів»**

Київ – 2023 року

ЗМІСТ

ВСТУП	5
1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ	8
1.1 Види систем рекомендації	9
1.1.1 Колаборативна фільтрація	9
1.1.2 Заснований на контенті	13
1.1.3 Факторизація матриці.....	15
1.1.4 Гібридні системи.....	16
1.2 Порівняння систем рекомендацій	18
1.2.1 Переваги та недоліки колаборативної фільтрації.....	18
1.2.2 Переваги та недоліки алгоритмів, що базуються на контенті	19
1.2.3 Переваги та недоліки алгоритмів факторизації матриць.....	19
1.2.4 Переваги та недоліки гібридних систем.....	20
Висновки до розділу	21
2 СТРУКТУРА СИСТЕМИ.....	23
2.1 Структура системи.....	23
2.2 Вибір мови програмування	27
2.2.1 JavaScript.....	27
2.2.2 Typescript	27
2.3 Обґрунтування вибору інструментів	28
2.3.1 Node.js	29
2.3.2 TypeScript.....	30
2.3.3 Angular	30

					ІС92.290БАК.004 ПЗ			
		№ докум.	Підпис	Дата	<i>Вебзастосунок онлайн магазину із реалізацією алгоритму рекомендації товарів</i> Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.	Чечет М.О.					Т	2	72
Керівн.	Жереб К.А.					КПП ім. Ігоря Сікорського Група ІС-92		
В.								

2.3.4 RxJs.....	32
2.3.5 NgRx.....	33
2.3.6 Angular Material.....	34
2.4 Вибір середовища розробки	35
2.5 Інформаційне забезпечення	36
2.5.1 Вхідні дані	36
2.5.2 Вихідні дані	37
2.5.3 Структура масивів інформації.....	37
Висновки до розділу	37
3 РЕАЛІЗАЦІЯ СИСТЕМИ.....	38
3.1 Змістовна постановка задачі.....	38
3.2 Математична постановка задачі.....	38
3.2.1 Оновлення базових предикторів (biases):.....	41
3.2.2 Оновлення факторних предикторів:	41
3.3 Обґрунтування методів розв'язання.....	42
3.4 Специфікація функцій застосунку	43
Висновки до розділу	44
4 ДОСЛІДЖЕННЯ ТА ЕКСПЕРИМЕНТИ.....	45
4.1 Керівництво користувача.....	45
4.2 Випробування програмного продукту.....	50
4.2.1 Мета випробувань.....	50
4.2.2 Результати випробувань.....	51
4.2.3 Навантажувальне тестування та аналіз швидкодії.....	53
Висновок до розділу	55

ВИСНОВКИ	56
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	58

ВСТУП

Ми живемо в епоху інформаційних систем та технологій, де кожного дня здавалося б вже дуже звичні та прості речі переходять на новий етап, безперервно покращуючи рівень якості нашого повсякденного життя. Не виключенням є і Світова Мережа Інтернет, яка дає нам невичерпні можливості не тільки для самореалізації, но і для комфортизації повсякденного життя. Одним із типовіших прикладів є онлайн магазини, що економлять наш час на здійснення покупок.

Швидкість розвитку веб-технологій сьогодні є неабияким фактором, що випереджає час. Це надає підприємствам та бізнесам унікальну можливість зайняти лідерські позиції на ринку шляхом розробки якісного веб-продукту, зокрема онлайн-магазину. Щоб успішно конкурувати з іншими гравцями, важливо мати на увазі декілька аспектів.

Перш за все, швидкість завантаження веб-сторінок грає критичну роль у веб-продуктах. Користувачі сучасного інтернету очікують миттєвого доступу до інформації та зручності використання. Якщо веб-сайт або онлайн-магазин повільно завантажується, це може спричинити втрату користувачів і зниження конкурентоспроможності. Тому розробка ефективних архітектур, використання кешування, оптимізація зображень та інші техніки прискорення завантаження сторінок є надзвичайно важливими.

В сучасному цифровому світі, якісна веб-розробка та новітні підходи до конкурентної спроможності онлайн сервісів стали надзвичайно важливими. Зокрема, алгоритми рекомендації товарів займають центральне місце в ефективному функціонуванні багатьох електронних комерційних платформ.

Розробка імплементації алгоритмів рекомендацій відіграє вирішальну роль у забезпеченні персоналізованого досвіду користувачів. Вона базується на зборі, аналізі та інтерпретації великого обсягу даних, що зібрані від користувачів у процесі їхньої взаємодії з платформою. На основі цих даних алгоритми рекомендацій можуть зрозуміти індивідуальні інтереси та вподобання кожного користувача, і надати персоналізовані рекомендації товарів.

					ІС92.290БАК.004 ПЗ	Арк.
						5
Зм.	Лист	№ докум.	Підпис	Дата		

Однією з головних переваг якісної веб-розробки є здатність збирати та обробляти дані у режимі реального часу. Це дозволяє платформам швидко реагувати на зміни в поведінці користувачів, а також забезпечити актуальні рекомендації, що підтримують високий рівень задоволеності клієнтів.

Новітні підходи до конкурентної спроможності онлайн сервісів включають в себе застосування машинного навчання та штучного інтелекту для покращення алгоритмів рекомендацій. Це дозволяє збільшити точність та ефективність рекомендацій, а також підвищити рівень задоволеності користувачів.

Завдяки якісній веб-розробці та новітнім підходам до алгоритмів рекомендацій, платформи електронної комерції можуть отримати ряд вагомих переваг. По-перше, вони забезпечують збільшення продажів, оскільки персоналізовані рекомендації збільшують шанси того, що користувач придбає товари, які відповідають його потребам та інтересам. По-друге, такі рекомендації збільшують лояльність клієнтів, спонукаючи їх повертатися на платформу знову і знову. Нарешті, це також сприяє збільшенню конкурентоспроможності платформи, оскільки вона може пропонувати унікальний та персоналізований досвід для кожного користувача.

Загалом, якісна сучасна веб-розробка та новітні підходи до алгоритмів рекомендацій товарів грають важливу роль у забезпеченні успіху онлайн сервісів. Вони сприяють покращенню взаємодії з користувачами, збільшенню продажів та підвищенню конкурентоспроможності платформ. Ці фактори є вирішальними для успіху електронної комерції в сучасному цифровому світі.

Але будь яка сфера діяльності що пов'язана із грошима — завжди мала і буде мати конкуренцію, тому якщо користувач вирішив відкрити свій онлайн магазин по продажу будь чого — треба завжди бути на голову вище інших конкурентів щоб мати добрий прибуток. Саме це я і взяв за основу свого дипломного проєкту — реалізація сучасного веб застосунку із використанням новітніших front—end технологій та патернів проектування із використанням machine—learning алгоритму рекомендації товарів користувача на основі його вподобань.

Актуальність проблеми обумовлена високим рівнем конкуренції на сучасному онлайн ринку. Та сам таке поєднання функціоналу веб розробки та математичного алгоритму рекомендації товарів здатне значно підвищити ефективність онлайн магазину за рахунок придбання користувачами додаткової кількості товарів.

Об'єктом дослідження є веб застосунок онлайн магазину.

Предметом дослідження є реалізація алгоритму рекомендації товарів користувачу.

Метою розробки є реалізація покращеної варіації алгоритму рекомендації товарів на основі сингулярного розкладу матриць та побудова сучасного веб застосунку на основі фреймворку Ангуляр для підвищення комерційної спроможності магазину. Для досягнення мети, розглядається завдання розробки рекомендаційної системи, яка базується на принципах колаборативної фільтрації, де необхідно розв'язати такі задачі:

— розробити відповідний алгоритм на основі отриманих даних із урахуванням переваг та недоліків існуючих рішень;

— дослідити існуючі системи рекомендацій та провести порівняльний аналіз;

— спроектувати веб застосунок та реалізувати у ньому використання алгоритму;

— протестувати створений застосунок та переконатись у коректній роботі алгоритму.

Також дана розробка має значне практичне призначення, бо включає у себе повноцінну розробку веб магазину, який можна використовувати у комерційній діяльності.

Дипломний проєкт складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 25 найменувань. Графічна частина включає 4 кресленики формату А3. Загальний обсяг 61 сторінка.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Історія рекомендаційних систем сягає коріння в далекі часи, але вони набули широкого розповсюдження та значної еволюції з появою Інтернету та розвитком комп'ютерних технологій. Надалі розглянемо важливі етапи історії рекомендаційних систем.

Перші згадки про рекомендаційні системи можна знайти у великих бібліотеках та каталогах, де бібліотекари використовували рекомендації для підбору книг для своїх читачів. У таких системах використовувалися прості алгоритми, засновані на категоріях та тематиках.

У 1990-х роках з появою Інтернету рекомендаційні системи стали використовувати аналіз контенту для рекомендацій [1]. Це означало, що система аналізувала властивості самого об'єкта (наприклад, ключові слова, жанри, автори) та порівнювала їх з іншими об'єктами для надання рекомендацій.

У кінці 1990-х років стали популярними алгоритми колаборативної фільтрації [2]. Ці алгоритми базувалися на аналізі історій взаємодії користувачів з об'єктами (наприклад, оцінки, покупки) для знаходження схожості між користувачами або між об'єктами. На основі цієї схожості система рекомендувала нові об'єкти, що сподобалися іншим користувачам зі схожими вподобаннями.

З поширенням Інтернету та соціальних мереж рекомендаційні системи стали більш персоналізованими. Вони використовують дані про користувачів, такі як інтереси, поведінка, демографічні дані, для покращення якості рекомендацій [3]. Такі системи здатні пропонувати індивідуально підходящі об'єкти, враховуючи унікальні вподобання кожного користувача.

Сучасні рекомендаційні системи часто поєднують різні підходи та алгоритми для отримання кращих результатів. Гібридні системи можуть використовувати комбінацію колаборативної фільтрації, фільтрації за змістом та інших методів для надання більш точних та різноманітних рекомендацій.

1.1 Види систем рекомендації

Існує кілька основних алгоритмів рекомендації товарів, які використовуються для забезпечення персоналізованого досвіду користувачів. Розглянемо детальніше кожен з них:

1.1.1 Колаборативна фільтрація

Цей алгоритм базується на зборі та аналізі інформації про вподобання користувачів. Він шукає схожість між користувачами або між товарами на основі їхніх історій покупок або оцінок. Після цього алгоритм може рекомендувати товари, які були популярними серед схожих користувачів або які подібні до тих, що сподобалися в минулому.

Такі алгоритми базуються на ідеї "подібність привертає" [4]. Вони дозволяють виявляти схожі вподобання між користувачами або між об'єктами для надання рекомендацій. При використанні цих алгоритмів важливо враховувати проблеми, такі як холодний старт (коли немає достатньо даних про нових користувачів або товари) та проблеми масштабованості при великій кількості користувачів та товарів.

Якщо розглядати алгоритми колаборативної фільтрації більш детально, то можна виділити наступні найпоширеніші реалізації.

1.1.1.1 User-based колаборативна фільтрація

Цей алгоритм спирається на схожість між користувачами для генерації рекомендацій. Система аналізує історію взаємодії користувачів з об'єктами (наприклад, оцінками товарів) та шукає користувачів, які мають схожі смаки або вподобання. Наступною кроком є знаходження товарів, які сподобалися цим схожим користувачам, але не сподобалися цільовому користувачу, і рекомендація їх останньому. Даний принцип можна побачити на рисунку 1.1 [5].

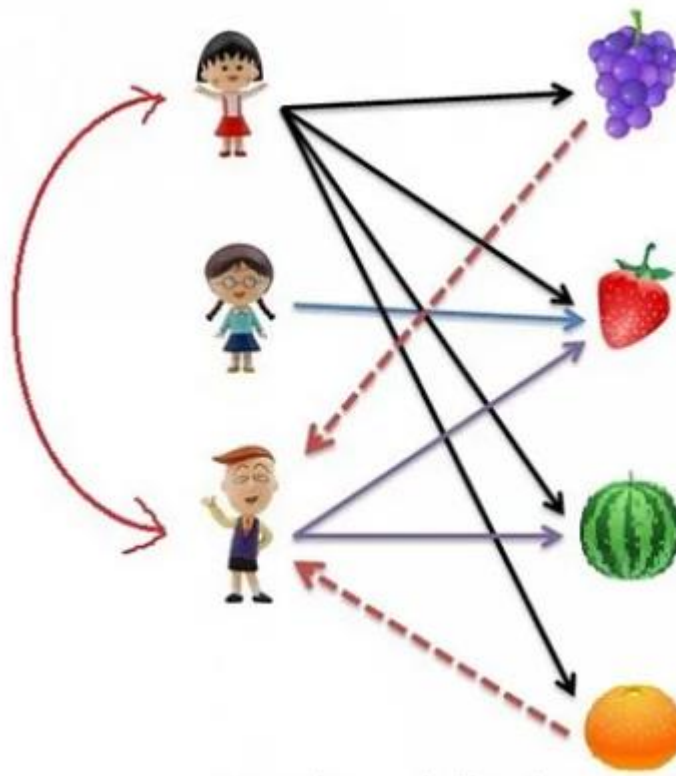


Рисунок 1.1 – Принцип роботи User-Based колаборативної фільтрації [1]

1.1.1.2 Item-based колаборативна фільтрація

У цьому випадку аналізується схожість між об'єктами (товарами). Система аналізує взаємодію користувачів з об'єктами та виявляє схожість між об'єктами на основі вподобань користувачів.

Головною відмінністю від user-based колаборативної фільтрації є те, що для аналізу рекомендації використовуються дані про товари, а не про користувачів.

Далі система може рекомендувати користувачу інші об'єкти, які схожі на ті, що йому сподобалися раніше.

Даний принцип можна побачити на рисунку 1.2 [5].

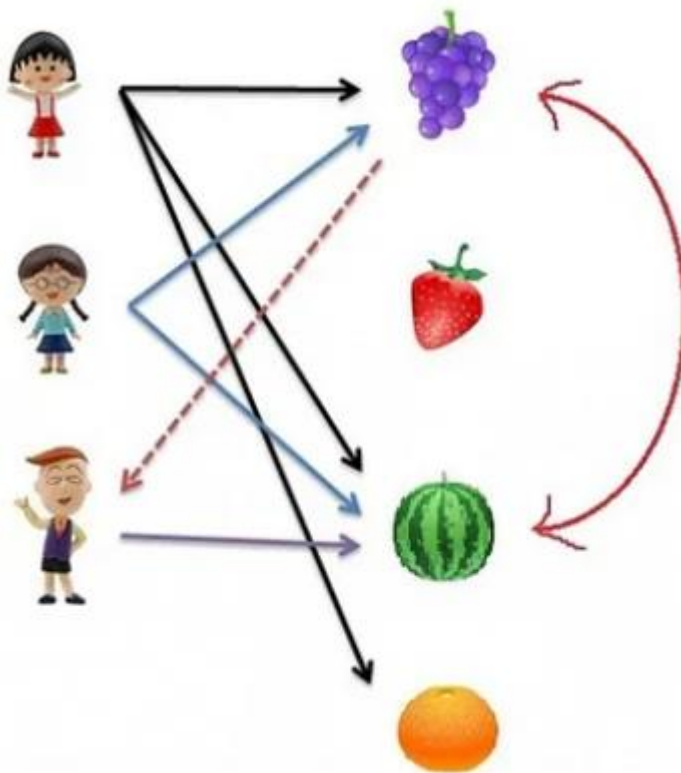


Рисунок 1.2 – Принцип роботи item-based колаборативної фільтрації

1.1.1.3 Singular Value Decomposition (SVD)

Це метод розкладання матриці, який використовується для знаходження складових, що визначають вподобання користувачів та характеристики товарів. В процесі розкладання матриці оцінок користувачів і товарів на менші матриці, можна отримати складові, які репрезентують уявлення про вподобання користувачів та характеристики товарів. За допомогою цих складових можна рекомендувати користувачеві нові товари. Приклад даних матриць зображено на рисунку 1.3 [6].

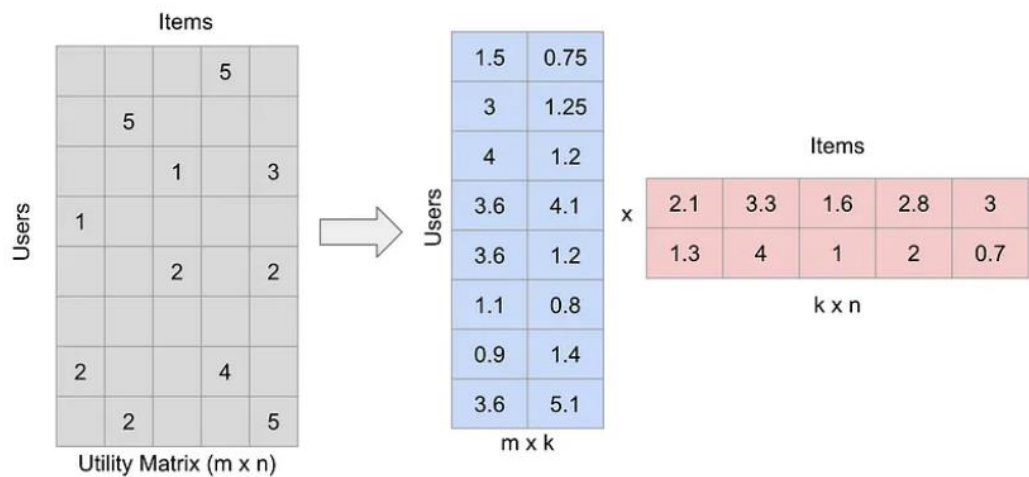


Рисунок 1.3 – Принцип роботи Singular Value Decomposition колаборативної фільтрації

1.1.1.4 Гібридна колаборативна фільтрація

Цей підхід поєднує різні аспекти user-based та item-based колаборативної фільтрації, а також може включати інші методи рекомендацій, що зображено на рисунку 1.4 [7]. Шляхом комбінування різних алгоритмів із використанням вагових коефіцієнтів можна отримати більш точні та персоналізовані рекомендації.

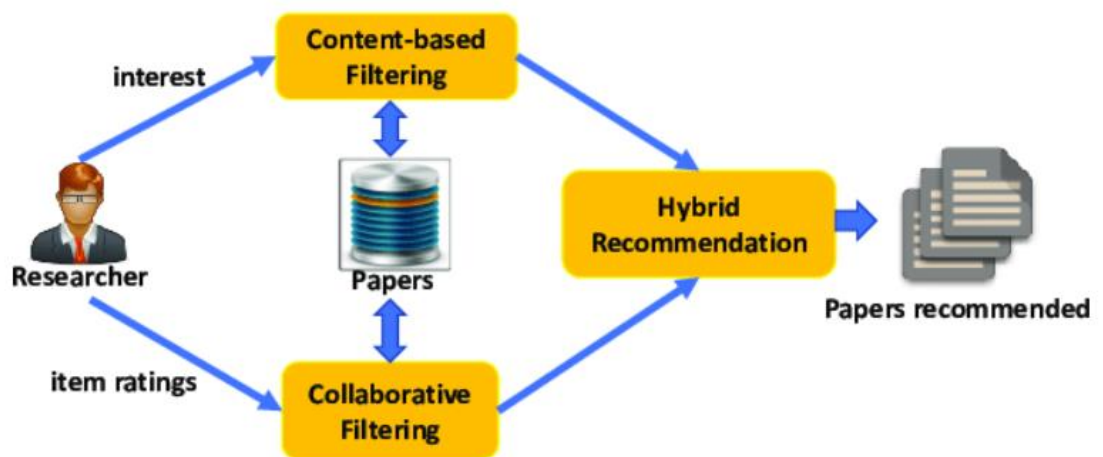


Рисунок 1.4 – Принцип роботи гібридної колаборативної фільтрації

1.1.2 Заснований на контенті

Цей підхід використовує аналіз характеристик товарів для рекомендацій. Алгоритм аналізує описи, ключові слова, категорії або інші характеристики товарів, а потім знаходить схожість між ними. На основі цієї схожості алгоритм рекомендує користувачеві товари, які мають схожі характеристики з тими, що йому сподобалися раніше.

1.1.2.1 Фільтрація за контентом (Content-Based Filtering)

Цей підхід аналізує властивості та характеристики самого об'єкта (товару) для здійснення рекомендацій. Наприклад, у випадку онлайн магазину це можуть бути ключові слова, опис, категорії, жанри або інші метадані товару. Система будує профіль кожного користувача на основі його взаємодії з попередніми товарами і рекомендує інші товари зі схожими властивостями або контентом. Візуалізацію даного підходу зображено на рисунку 1.5 [8].



Рисунок 1.5 – Принцип роботи content-based filtering алгоритму

1.1.2.2 Тематична модель (Topic Modeling)

Цей алгоритм використовується для аналізу текстового контенту, такого як описи товарів або відгуки користувачів, і виявлення тематик, що присутні в цьому контенті. Зазвичай використовується метод Latent Dirichlet Allocation (LDA), який може виділити семантичні теми, що присутні в тексті, принцип роботи якого зображено на рисунку 1.6 [9]. На основі цих тем система може рекомендувати товари, що відповідають інтересам користувача, пов'язані з цими темами.

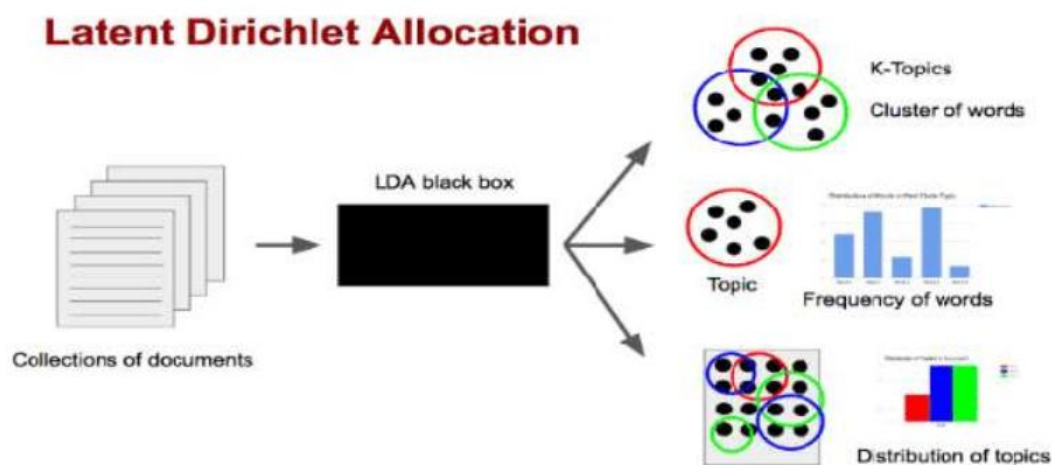


Рисунок 1.6 – Принцип роботи topic modeling (LDA) алгоритму

1.1.2.3 Векторні представлення (Vector Representations)

Цей метод використовується для перетворення об'єктів та користувачів у векторні представлення у просторі ознак. Зазвичай використовуються методи векторної моделі термів (Term Frequency-Inverse Document Frequency, TF-IDF) або word embeddings, такі як Word2Vec або GloVe, для представлення контенту. За допомогою цих векторних представлень можна обчислити схожість між об'єктами та рекомендувати подібні товари.

1.1.3 Факторизація матриці

Засновані на факторизації матриць алгоритми є ефективними і широко використовуються в рекомендаційних системах. Вони здатні розкрити складові, що визначають взаємодію між користувачами та об'єктами, що дозволяє здійснювати точні та персоналізовані рекомендації. Основні алгоритми засновані на факторизації матриць включають Singular Value Decomposition (SVD) та Matrix Factorization (MF).

1.1.3.1 Singular Value Decomposition (SVD)

SVD є одним з найпопулярніших методів факторизації матриць. Він використовується для розкладання матриці рейтингів користувачів і об'єктів на менші матриці. За допомогою SVD можна отримати складові, які представляють уявлення про вподобання користувачів та характеристики об'єктів. Ці складові можуть бути використані для рекомендацій шляхом передбачення оцінок для невідомих взаємодій. Приклад розкладу матриць зображено на рисунку 1.7 [10].

$$\begin{array}{c} \text{Original} \\ \text{Matrix} \end{array} \quad \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} = \begin{array}{c} \text{Eigenvectors} \\ \text{Matrix} \end{array} \begin{bmatrix} -1 & -1 \\ 2 & 1 \end{bmatrix} \begin{array}{c} \text{Eigenvalues} \\ \text{Matrix} \end{array} \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix} \begin{array}{c} \text{Inverse of} \\ \text{Eigenvectors} \\ \text{Matrix} \end{array} \begin{bmatrix} 1 & 1 \\ -2 & -1 \end{bmatrix}$$

Рисунок 1.7 – Принцип роботи SVD алгоритму

1.1.3.2 Matrix Factorization (MF)

Matrix Factorization є варіантом факторизації матриць, який часто використовується в рекомендаційних системах. У цьому підході матриця рейтингів користувачів і об'єктів розкладається на дві менші матриці – матрицю користувачів та матрицю об'єктів, що зображені на рисунку 1.8 [11]. Ці матриці представляють уявлення про вподобання користувачів і характеристики об'єктів. За допомогою факторизованих матриць можна здійснювати передбачення рейтингів та рекомендацій для користувачів.

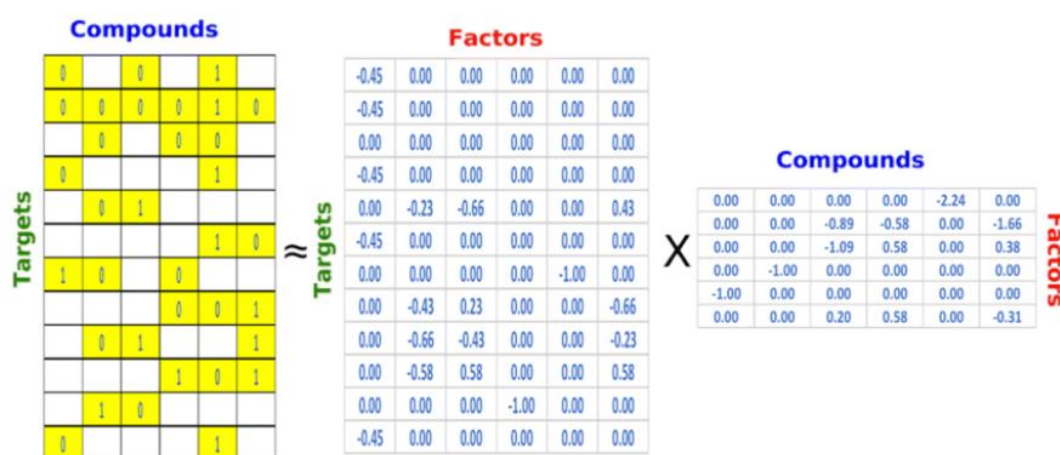


Рисунок 1.8 – Принцип роботи matrix factorization алгоритму

1.1.4 Гібридні системи

Гібридні системи рекомендації товарів поєднують різні підходи та алгоритми для надання більш точних та персоналізованих рекомендацій користувачам. Ці системи комбінують сильні сторони різних підходів і стають все більш популярними в сфері електронної комерції та онлайн-сервісів.

Основна ідея гібридних систем полягає в тому, що вони використовують комбінацію різних типів рекомендацій, таких як колаборативна фільтрація, заснована на контенті, рекомендації на основі популярності, демографічні фактори тощо. Таким чином, система може враховувати різні аспекти користувачів,

предметів та контексту, що дозволяє побудувати більш гнучку та ефективну систему рекомендацій.

Одним з підходів до побудови гібридних систем є комбінування результатів різних алгоритмів. Наприклад, можна використовувати колаборативну фільтрацію для знаходження схожих користувачів або предметів, а потім поєднати її з рекомендаціями на основі контенту, щоб отримати більш точні рекомендації. Іншим підходом є використання моделей машинного навчання для прогнозування рейтингів або відповідей на питання, які потім комбінуються з іншими методами рекомендацій.

Гібридні системи також можуть використовувати контекстну інформацію, таку як геолокація, часові аспекти, поведінкові дані користувачів та інші фактори, щоб адаптувати рекомендації до конкретного контексту або ситуації.

Переваги гібридних систем рекомендацій полягають у їхній здатності до надання більш точних та персоналізованих рекомендацій, оскільки вони комбінують різні джерела інформації та підходи. Вони можуть враховувати як історичні дані, так і контекстуальні фактори, що дозволяє враховувати чи змінюються потреби та уподобання користувачів. Крім того, гібридні системи можуть бути більш стійкими до проблем, таких як холодний старт, коли недостатньо даних для надання рекомендацій новим користувачам або предметам.

Однак, розробка та підтримка гібридних систем можуть бути складнішими та вимагати більше ресурсів порівняно зі стандартними підходами. Крім того, вибір та налаштування відповідних алгоритмів та методів може бути нетривіальним завданням.

У підсумку, гібридні системи рекомендацій товарів мають переваги у тому, що вони забезпечують більш точні та персоналізовані рекомендації, адаптуються до різних контекстів та враховують різноманітні аспекти користувачів та предметів. Водночас, їхнє впровадження та підтримка можуть бути складними завданнями, які вимагають відповідного аналізу та налаштування алгоритмів.

1.2 Порівняння систем рекомендацій

На основі представлених видів та алгоритмів рекомендації товарів користувачу, порівняємо їхні особливості, переваги та недоліки в порівнянні один з одним.

1.2.1 Переваги та недоліки колаборативної фільтрації

Переваги колаборативної фільтрації:

— здатність до рекомендацій відсутніх об'єктів: Колаборативна фільтрація може рекомендувати об'єкти, про які немає детальної інформації або контенту, використовуючи схожість між користувачами або об'єктами на основі їх історії взаємодії.

— виявлення складних залежностей: Колаборативна фільтрація може виявити складні залежності між користувачами та об'єктами, які можуть бути складні для виявлення алгоритмами, що базуються на контенті;

— популярність інформації: Колаборативна фільтрація використовує вже наявні дані про взаємодію користувачів з об'єктами, тому не вимагає додаткової інформації про контент або характеристики об'єктів;

Недоліки колаборативної фільтрації:

— проблема холодного старту: Колаборативна фільтрація може мати проблему при рекомендаціях для нових користувачів або нових об'єктів, оскільки відсутня історія взаємодії;

— проблема розрідженої матриці: У великих рекомендаційних системах матриця взаємодії між користувачами та об'єктами може бути розрідженою, що ускладнює точність рекомендацій;

— проблема рекомендацій вузького кола: Колаборативна фільтрація може спричиняти рекомендації, що обмежуються вузьким колом об'єктів, оскільки базується на історії взаємодії користувачів;

1.2.2 Переваги та недоліки алгоритмів, що базуються на контенті

Переваги алгоритмів, що базуються на контенті:

— урахування особистих вподобань: Алгоритми, що базуються на контенті, здатні брати до уваги особисті вподобання користувачів, використовуючи характеристики об'єктів;

— проблема холодного старту: Алгоритми, що базуються на контенті, можуть пропонувати рекомендації навіть для нових користувачів або об'єктів, оскільки вони враховують характеристики об'єктів;

— ефективність: Алгоритми, що базуються на контенті, можуть бути ефективними для систем з обмеженим обсягом даних;

Недоліки алгоритмів, що базуються на контенті:

— обмежена схожість: Алгоритми, що базуються на контенті, можуть бути обмежені у виявленні складних залежностей між користувачами та об'єктами, оскільки вони враховують лише характеристики об'єктів;

— проблема нових тенденцій: Алгоритми, що базуються на контенті, можуть бути менш ефективними в оновлюваних середовищах, оскільки не враховують оновлення у вподобаннях користувачів;

1.2.3 Переваги та недоліки алгоритмів факторизації матриць

Переваги алгоритмів факторизації матриць:

— робота з розрідженими матрицями: Факторизація матриць дозволяє працювати з розрідженими матрицями, зменшуючи проблему розрідженості та забезпечуючи точні рекомендації;

— здатність до виявлення складних залежностей: Факторизація матриць може виявляти складні залежності між користувачами та об'єктами, шляхом виявлення прихованих факторів;

— персоналізовані рекомендації: Факторизація матриць дозволяє створювати персоналізовані рекомендації, враховуючи індивідуальні вподобання користувачів;

Недоліки алгоритмів факторизації матриць:

— обчислювальна складність: Факторизація матриць може бути обчислювально витратною, особливо для великих та розріджених матриць, що може призвести до затримок у генерації рекомендацій;

— проблема схожості об'єктів: Факторизація матриць може мати проблему у врахуванні схожості між об'єктами, оскільки залежності базуються на внутрішніх факторах;

1.2.4 Переваги та недоліки гібридних систем

Переваги гібридних систем рекомендацій товарів:

— краща точність рекомендацій: Гібридні системи комбінують різні підходи, що дозволяє отримувати більш точні рекомендації. Вони використовують інформацію з різних джерел, таких як колаборативна фільтрація, контент, демографічні дані та інші фактори, що покращує якість рекомендацій;

— більша персоналізація: Гібридні системи враховують індивідуальні вподобання та характеристики користувачів, що дозволяє надавати персоналізовані рекомендації. Вони враховують контекстуальні фактори, історію взаємодії та інші аспекти, що покращує відповідність рекомендацій індивідуальним потребам користувачів;

— більш широкий охоплюючий спектр рекомендацій: Гібридні системи здатні надавати рекомендації не тільки на основі подібності користувачів або предметів, але й на основі інших факторів, таких як популярність, демографічні дані, контекст тощо. Це дозволяє надавати рекомендації, які враховують різні аспекти та потреби користувачів;

Недоліки гібридних систем рекомендацій товарів:

— складність розробки та підтримки: Розробка гібридних систем може бути складним завданням, оскільки вони вимагають комбінації різних алгоритмів та методів. Налаштування та підтримка таких систем можуть також бути витратними з точки зору часу та ресурсів;

— проблема холодного старту: Гібридні системи можуть стикатись з проблемою холодного старту, коли недостатньо даних про нових користувачів або предмети для надання точних рекомендацій. В таких випадках потрібно використовувати додаткові стратегії, наприклад, популярні або контекстуальні рекомендації;

— вимоги до обчислювальних ресурсів: Гібридні системи можуть бути вимогливими до обчислювальних ресурсів, оскільки вони використовують комбінацію різних алгоритмів та методів. Обчислювальна складність може бути проблемою для великих обсягів даних або в умовах обмежених ресурсів;

Усе ж таки, гібридні системи рекомендацій товарів мають багато переваг, які перевищують їхні недоліки, оскільки вони надають точніші та персоналізовані рекомендації, а також враховують різноманітні аспекти користувачів та предметів. Хоча розробка та підтримка гібридних систем можуть бути вимогливими, але вони можуть забезпечити велику цінність та поліпшити користувацький досвід.

Висновки до розділу

У даному розділі були розглянуті найпопулярніші види та типи систем рекомендації товарів, а саме: алгоритми колаборативної фільтрації, алгоритму заснованому на контенті, факторизації матриць та був проведений порівняльний аналіз між всіма існуючими аналогами за існуючими недоліками та перевагами.

Алгоритми колаборативної фільтрації шукають схожість між користувачами або між товарами на основі їхніх історій покупок або оцінок. Із основних недоліків можна виділити проблему холодного старту: Колаборативна фільтрація може мати проблему при рекомендаціях для нових користувачів або нових об'єктів. Водночас найбільшою перевагою даного підходу є виявлення складних залежностей:

колаборативна фільтрація може виявити складні залежності між користувачами та об'єктами, які можуть бути складні для виявлення

На відміну від колаборативної фільтрації, алгоритми засновані на контенті не мають серед недоліків проблему холодного старту, оскільки вони можуть пропонувати рекомендації навіть для нових користувачів або об'єктів, бо вони враховують характеристики об'єктів. Але водночас такий підхід має свій головний недолік – він може бути обмежений у виявленні складних залежностей між користувачами та об'єктами, оскільки вони враховують лише характеристики об'єктів.

Також серед розглянутих методів був алгоритм сингулярного розкладу матриць (SVD), найбільшою перевагою якого є факторизація матриць, яка дозволяє працювати з розрідженими матрицями, зменшуючи проблему розрідженості та забезпечуючи точні рекомендації. Але як і в усіх методах, даний підхід також має свій недолік, а саме обчислювальна складність: Факторизація матриць може бути обчислювально витратною, особливо для великих та розріджених матриць, що може призвести до затримок у генерації рекомендацій.

Також був проведений порівняльний аналіз усіх існуючих систем та були описані їхні головні недоліки та переваги в порівнянні один з одним.

					ІС92.290БАК.004 ПЗ	Арк.
						22
Зм.	Лист	№ докум.	Підпис	Дата		

2 СТРУКТУРА СИСТЕМИ

Метою цього дипломного проєкту є розробка базового онлайн магазину із рекомендацією товарів користувача на основі його вподобань. Даний застосунок буде розроблений на основі сучасного front-end фреймворку Angular, в якому будуть реалізовані всі сучасні патерни побудови веб застосунку.

Така реалізація дозволить власнику не тільки заощадити гроші на подальшому розвитку проєкту, що включає у себе масштабування, розширення функціоналу та створення нових функцій, а й збільшити свої прибутки за рахунок реалізації алгоритму рекомендації товарів.

2.1 Структура системи

Дана система буде включати у себе два окремих модулі, де перший – це front-end застосунок, який представляє сам інтерфейс онлайн магазину із можливістю придбання товарів, та другий – це модуль самого алгоритму рекомендації, який буде включений до застосунку

Front-end застосунок буде розділений на окремі модулі, такі як shared, core та app.

Головна мета shared модулю це групування усіх компонентів та директив що перевикористовуються у всьому застосунку. Такий підхід чітко відповідає означенню патерна DRY (don't repeat yourself) [12], бо весь функціонал що перевикористовується буде написаний лише один раз та не буде повторюватися. Такий модуль в свою чергу буде містити такі компоненти, як:

- директиви;
- моделі об'єктів;
- конфігураційні файли;
- компоненти сторінок;
- сервіси, що перевикористовуються;

Структура shared модулю зображена на рисунку 2.1.

```
|-- shared
  |-- [+] components
  |-- [+] directives
  |-- [+] pipes
  |-- [+] models
  |-- [+] configs
```

Рисунок 2.1 – Структура shared модулю

Core модуль був створений для інкапсуляції усієї базової логіки що потрібна для коректної роботи застосунку, наприклад API (Application Programming Interface) сервіси, які відповідають за взаємодію із сервером, інтерсептори (англ. Interceptors), які є посередниками між відправкою та отриманням запитів з та до сервера. Структура даного модулю зображена на рисунку 2.2.

```
|-- core
  |-- [+] components
    |-- [+] footer
    |-- [+] header
  |-- [+] guards
  |-- [+] interceptors
  |-- [+] mocks
  |-- [+] services
  |-- [+] authentication
  |-- core.module.ts
```

Рисунок 2.2 – Структура core модулю

App модуль в свою чергу включає у себе усі окремі компоненти застосунку, такі як сторінки, приклад яких можна побачити на рисунку 2.3. Вони ж в свою чергу також входять до окремих модулів для того щоб реалізувати патерн лінивого

підвантаження (lazy-loading) [13]. Суть даного підходу полягає у підвантажуванні та обробці тільки тих складових застосунку, до яких безпосередньо перейшов користувач. Така реалізація дозволяє суттєво зменшити час першого завантаження застосунку, бо замість обробки усього проекту буде підвантажена лише менша частина, потрібна лише для початкової сторінки.

```
|-- appointment
  |-- [+] components
  |-- [+] models
  |-- [+] services
  |-- appointment-routing.module.ts
  |-- appointment.module.ts
```

Рисунок 2.3 – Структура базового сторінкового модулю

Структура даних у проєкті буде реалізована на основі сучасного патерну реактивного програмування (reactive programming [14]), який включає у себе використання окремої бібліотеки NgRx [15]. У такому підході усі дані які прив'язані до конкретних сторінок або модулів будуть згруповані у відповідні об'єкти (states). Діаграма роботи даного підходу зображена на рисунку 2.4.

Для двонапрявленого доступу до даних будуть використовуватися спеціальні об'єкти actions, які інкапсулюють у собі спеціальні команди для взаємодії із об'єктом даних. Такий підхід є реалізацією патерну команда (command) [16].

За модифікування об'єктів даних відповідають спеціальні функції (reducers), які містять логіку модифікації даних на основі вхідних команд (actions). Така реалізація є прикладом використання патерну State.

В більшості випадків об'єкти даних можуть буди доволі місткими, включаючи у себе багато інших внутрішніх об'єктів, що може стати проблемою у роботі з ними. Саме для цього використовуються спеціальні вибіркові функції (selectors), які повертають конкретні частини складеного об'єкту, що значно спрощує роботу із даними.

За реалізацію побічних ефектів застосунку, таких як відправка запитів на сервер, будуть використані спеціальні сервіси які називаються ефектами (effects). Ці сервіси підписуються на усі вхідні команди (actions) та виконують відповідну логіку залежачи від команди. За такий функціонал відповідає патерн споглядач (observer) [16].

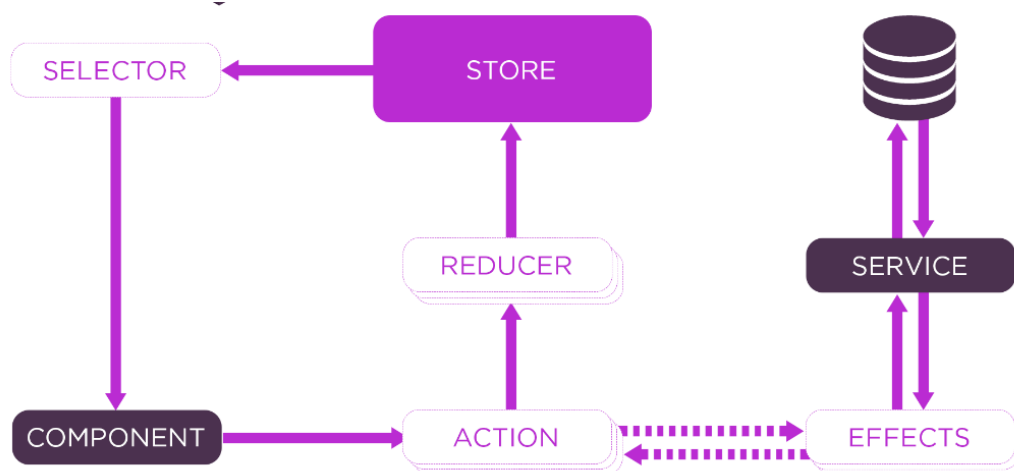


Рисунок 2.4 – Структура взаємодії усіх компонентів NgRx

Також структура окремих сторінкових модулів буде включати у себе структурний патерн контейнерних та презентаційних компонентів (container and presentational components), який полягає у сепарації складових на контейнерні компоненти, які містять у собі всю логіку сторінки, та презентаційних, які відповідають лише за представлення даних. Такий розподіл значно підвищує читабельність коду та допомагає краще масштабувати застосунок.

За презентативну стилістичну основу застосунку буде використаний препроцесор стилів SASS (Syntactically Awesome Style Sheets) [17], який є надбудовою над стандартною мовою стилів CSS (Cascading Style Sheets). Даний препроцесор дозволяє не тільки значно швидше розробляти продукт за допомогою вбудованого функціоналу, а й значно збільшити розуміння та читабельність коду.

Як один із пунктів реалізації патерну прогресивного веб застосунку (PWA) буде реалізований механізм service worker, який дозволяє кешувати дані та запити на сервер та реалізує функціонал режиму без доступу до мережі інтернет (offline

mode) [18]. Така функціональність є досить ефективною, бо допомагає покращити досвід користувача (user experience) шляхом підвищення швидкості обробки даних.

2.2 Вибір мови програмування

JavaScript та TypeScript є двома мовами програмування, які широко використовуються для розробки веб-застосунків [19]. Із найбільших переваг даних мов можна виділити наступні пункти:

2.2.1 JavaScript

— універсальність: JavaScript є стандартною мовою програмування для веб-розробки. Всі сучасні браузері підтримують виконання JavaScript, що дозволяє створювати веб-застосунки, що працюють на будь-якому пристрої з доступом до Інтернету;

— клієнтська та серверна розробка: JavaScript може бути використаний як на клієнтській, так і на серверній стороні розробки. На клієнтській стороні він використовується для створення інтерактивних елементів веб-сторінок, взаємодії з користувачем та виконання різних операцій у браузері. На серверній стороні, з використанням платформи, такої як Node.js, JavaScript дозволяє створювати потужні та масштабовані серверні додатки;

— багата екосистема: JavaScript має велику та активну екосистему з багатьма бібліотеками та фреймворками, які допомагають веб-розробникам прискорити розробку та покращити функціональність веб-застосунків. Такі бібліотеки, як React, Vue.js, та фреймворки, такі як Angular, дозволяють будувати складні та потужні веб-додатки;

2.2.2 Typescript

— статична типізація: TypeScript є розширенням JavaScript, яке додає статичну типізацію. Він дозволяє оголошувати типи змінних, параметрів та

					IC92.290БАК.004 ПЗ	Арк.
						27
Зм.	Лист	№ докум.	Підпис	Дата		

функцій, що полегшує відлагодження, зменшує кількість помилок та поліпшує роботу над великими проектами;

— покращена підтримка IDE: TypeScript надає більшу підтримку для інтегрованих середовищ розробки (IDE), таких як Visual Studio Code. Завдяки статичній типізації, IDE може надати більше підказок, автодоповнень та перевірок під час розробки, що сприяє покращенню продуктивності розробника;

— об'єктно-орієнтована парадигма: TypeScript підтримує об'єктно-орієнтовану парадигму програмування, включаючи класи, спадкування, інтерфейси та багато інших концепцій, що дозволяють писати більш структурований та модульний код;

— екосистема JavaScript: TypeScript побудований на основі JavaScript, що означає, що ви можете використовувати існуючі бібліотеки та фреймворки JavaScript у вашому TypeScript-проекті. Ви також можете використовувати @types пакети, що надають типові оголошення для зовнішніх бібліотек JavaScript;

Усі front-end застосунки працюють у середовищі браузера, який використовує мову JavaScript, то відповідною мовою для написання застосунку була обрана її строго типізована версія Typescript. Ця мова має перевагу над звичайною JavaScript, бо має функціонал строгої типізації, що значно покращує та пришвидшує процес розробки застосунку шляхом впровадження конкретних типів для змінних та функцій, що в свою чергу, значно зменшує ризик помилки у програмуванні та дозволяє сучасним середовищам розробки використовувати такий функціонал, як авто-заповнення полів та пре-валідація виразів.

2.3 Обґрунтування вибору інструментів

Для повноцінної та ефективної розробки програмного забезпечення були використані відповідні бібліотеки та набори інструментів, що значно пришвидшують, покращують та економлять час на розробку застосунку. Далі наведено перелік використаних бібліотек та технологій.

					ІС92.290БАК.004 ПЗ	Арк.
						28
Зм.	Лист	№ докум.	Підпис	Дата		

2.3.1 Node.js

Як середовище для взаємодії із front-end застосунком було використане середовище NodeJS, бо Angular має вбудований командний інтерфейс CLI (Command Line Interface), який дозволяє швидко та ефективно взаємодіяти із застосунком.

Node.js є потужним серверним середовищем, заснованим на JavaScript, яке має кілька переваг у порівнянні з іншими технологіями розробки веб-додатків [20].

Однією з ключових переваг Node.js є його висока продуктивність. Використання однопотокової та неблокуючої моделі введення/виведення (I/O) дозволяє досягти вражаючої швидкості обробки запитів. Це особливо важливо для додатків, які потребують обробки багатьох одночасних з'єднань або роботи в реальному часі, таких як чати, потокове відео, ігри тощо.

Більш того, Node.js використовує мову JavaScript як свою основну мову розробки. Це дозволяє розробникам використовувати одну мову як на фронтенді, так і на бекенді, спрощуючи процес розробки та забезпечуючи більшу переносимість коду. Крім того, велика кількість доступних пакетів та модулів у публічному репозиторії NPM (Node Package Manager) робить розробку швидшою та зручнішою.

Також Node.js може легко масштабуватись горизонтально, додавати нові вузли до серверного кластера, щоб розподілити навантаження та забезпечити високу доступність. Крім того, використання асинхронного програмування дозволяє ефективно використовувати обмежені ресурси сервера та запобігати блокуванню додатків під час виконання довгих операцій вводу/виводу.

Також завдяки своїй асинхронній природі та можливості обробляти події в режимі реального часу, Node.js ідеально підходить для створення додатків, які потребують активної взаємодії з користувачами в режимі реального часу, наприклад, веб застосунки. Дане середовище підтримує реалізацію веб-сокет з'єднання, що в свою чергу дозволяє реалізовувати відповідні системи із роботою у реальному часі.

2.3.2 TypeScript

Однією з найбільших переваг TypeScript є можливість використовувати статичну типізацію. Це робить можливим оголосити типи для змінних, параметрів функцій та повернутих значень, що дозволяє виявляти помилки на етапі розробки. Це сприяє покращенню надійності коду, полегшує рефакторинг та підвищує зрозумілість програмного коду.

Також TypeScript підтримує об'єктно-орієнтований підхід до програмування, дозволяючи використовувати класи, спадкування, інтерфейси, абстрактні класи та багато інших конструкцій. Це спрощує організацію коду, забезпечує більшу модульність та повторне використання.

Більш того, TypeScript базується на стандартах ECMAScript і постійно оновлюється, щоб підтримувати нові функції та можливості, які вводяться в JavaScript. Це означає, що розробник може використовувати сучасні конструкції мови JavaScript, такі як стрілкові функції, розширені об'єктні літерали, асинхронні функції та ітератори, навіть якщо цільова платформа не підтримує їх безпосередньо.

Слід також зазначити, що TypeScript надає потужні інструменти розробки, які полегшують процес розробки і підвищують продуктивність. Наприклад, він має інтегровану систему автодоповнення коду, перевірку типів під час компіляції, можливість виявлення помилок та підказок у редакторі коду. Це допомагає зменшити кількість помилок, полегшує роботу з великими проектами та поліпшує рефакторинг.

2.3.3 Angular

Angular – це потужний фреймворк для розробки веб – застосунків, який пропонує кілька переваг, що роблять його лідером серед конкурентних фреймворків. Він є потужним фреймворком, який надає розробникам багато можливостей для створення високоякісних та масштабованих веб-застосунків. Він

поєднує в собі потужний функціонал, продуктивність, широку спільноту та підтримку, роблячи його привабливим вибором для веб-розробки.

Даний фреймворк розроблений з урахуванням масштабованості проєктів. Він пропонує потужну систему модульності та інструменти для організації коду, що дозволяє легко розширювати та підтримувати додатки навіть у разі зростання їх обсягу. Компонентна архітектура Angular допомагає розділити функціональність на незалежні блоки, що полегшує роботу багатьох розробників над проєктом.

Більш того, Angular має вбудовану підтримку для реактивного програмування за допомогою RxJS. Це дозволяє створювати реактивні компоненти, які автоматично оновлюються при зміні даних, що полегшує створення додатків із real-time функціоналом, таких як чати, потокові оновлення тощо. Крім того, RxJS надає потужні оператори та інструменти для роботи з асинхронними подіями та даними.

Також слід зазначити, що Angular пропонує багатий функціонал для розробки веб-застосунків. Він має вбудовану систему маршрутизації для навігації між сторінками, підтримку форм та валідацію даних, можливості міжкомпонентного спілкування, вбудовану підтримку інтернаціоналізації та локалізації, а також можливості для тестування додатків. Це дозволяє розробникам швидко створювати повнофункціональні застосунки з різноманітними потребами.

Окрім цього Angular має велику спільноту розробників, яка активно допомагає одне одному, ділиться знаннями та ресурсами. Є багато документації, статей, курсів та плагінів, що спрощують процес вивчення та використання фреймворку. Крім того, Angular підтримується компанією Google, що забезпечує стабільність, оновлення та активну підтримку фреймворку.

Також Angular може використовуватися для розробки як веб-застосунків, так і мобільних додатків з використанням фреймворку Ionic. Це дозволяє створювати універсальні рішення для різних платформ з використанням знайомих інструментів та практик.

Більш того, Angular офіційно підтримується командою розробників Google, що значно полегшує відповідну підтримку застосунків.

					IS92.290БАК.004 ПЗ	Арк.
						31
Зм.	Лист	№ докум.	Підпис	Дата		

2.3.4 RxJs

RxJS (Reactive Extensions for JavaScript) – це бібліотека, яка пропонує реактивне програмування для JavaScript, зокрема для веб-додатків [21]. Вона надає потужний інструментарій для роботи з асинхронними подіями і дозволяє елегантно керувати потоками даних. Із головних переваг даної бібліотеки можна виділити наступні пункти:

— асинхронність і події: Веб-застосунки часто працюють з асинхронними запитами на сервер, подіями від користувача, таймерами та іншими асинхронними подіями. RxJS надає потужні оператори для маніпулювання потоками подій, зручність у роботі з асинхронністю та можливість комбінувати, фільтрувати і перетворювати дані з різних джерел;

— код злагодження: За допомогою RxJS можна зменшити кількість вкладених зворотних викликів (callback hell) та спростити асинхронний код. Ви можете легко компонувати і комбінувати різні асинхронні операції, що робить код більш зрозумілим та підтримуваним;

— обробка помилок: RxJS надає механізми для елегантної обробки помилок. За допомогою операторів, таких як `catch`, `retry` і `throwError`, ви можете легко контролювати і обробляти помилки в асинхронному коді;

— кешування і пакетування даних: RxJS дозволяє ефективно працювати з даними, зокрема кешувати, фільтрувати, трансформувати і комбінувати їх. Ви можете визначити потоки даних, які оновлюються автоматично при зміні вхідних даних, і автоматично оновлювати відображення;

— крос-платформеність: RxJS є мультиплатформеною бібліотекою, що підтримує не лише веб, але й інші платформи, такі як мобільні додатки на базі React Native, desktop-додатки, Node.js та інші;

В цілому, використання RxJS у веб-застосунку дозволяє створювати більш ефективний, зрозумілий і легко підтримуваний асинхронний код, спрощує роботу з подіями та потоками даних, та забезпечує багато інших корисних функцій для розробки веб-додатків.

2.3.5 NgRx

Бібліотека NgRx є реалізацією архітектурного шаблону Redux для Angular-додатків. Вона надає потужний інструментарій для управління станом додатка та реактивного програмування. Ось кілька причин, чому використовувати NgRx у веб-застосунку:

— централізоване управління станом: NgRx дозволяє зберігати стан додатка в одному централізованому місці, відомому як "store". Це полегшує відлагодження, тестування і керування станом додатка, особливо в складних веб-застосунках з багатьма компонентами та взаємодією між ними;

— передбачуваність та відновлення стану: NgRx пропонує незмінний підхід до зміни стану додатка, де кожна зміна стану є чистою функцією, що виконується за допомогою "reducers". Це забезпечує передбачуваність стану і полегшує відновлення стану після помилок або перезавантаження сторінки;

— взаємодія з асинхронними даними: NgRx дозволяє легко керувати асинхронними операціями, такими як запити на сервер, за допомогою "effects". Ви можете керувати потоками даних, обробляти помилки та виконувати додаткові дії на базі відповідей сервера;

— строга типізація та інструменти розробки: NgRx працює добре зі строгою типізацією Angular і надає потужні інструменти розробки, такі як NgRx DevTools, що дозволяють відстежувати та аналізувати зміни стану додатка в режимі реального часу;

— масштабованість та розширюваність: NgRx забезпечує модульну структуру для управління станом, що дозволяє легко розширювати функціональність та масштабувати додаток. Ви можете створювати нові "actions", "reducers" та "effects" для кожної функціональності додатку, що дозволяє підтримувати чистоту коду і його легку розбивку на модулі;

Загалом, використання NgRx у веб-застосунку дозволяє створювати добре організований, передбачуваний та легко масштабований код, забезпечує ефективне

управління станом та асинхронними даними, а також надає потужні інструменти розробки для полегшення роботи зі станом додатка.

2.3.6 Angular Material

Angular Material – це набір готових компонентів та рекомендацій для розробки інтерфейсу користувача веб-додатків з використанням фреймворку Angular [22]. Із головних переваг використання даної бібліотеки можна виділити наступні пункти:

— матеріальний дизайн: Angular Material надає реалізацію матеріального дизайну, який є популярним стандартом для створення сучасних та естетично залучних інтерфейсів. Ви можете використовувати готові стилі, компоненти та анімацію, щоб швидко створити привабливий вигляд для вашого веб-додатку;

— готові компоненти: Angular Material має велику кількість готових компонентів, таких як кнопки, карточки, форми, таблиці, меню, діалогові вікна тощо. Ви можете використовувати ці компоненти безпосередньо у вашому коді, що дозволяє зекономити час на їх розробці та роботу над деталями;

— адаптивний та респонсивний дизайн: Компоненти Angular Material розроблені з урахуванням адаптивного та респонсивного дизайну. Вони автоматично пристосовуються до розміру екрану та пропонують оптимальний досвід користувача як на десктопах, так і на мобільних пристроях;

— уніфікований стиль: Використання Angular Material дозволяє забезпечити уніфікований стиль для всього вашого веб-додатку. Ви можете легко керувати темами, кольорами, типографією та іншими аспектами дизайну, щоб забезпечити консистентний вигляд і почуття в усьому додатку;

— легка інтеграція з Angular: Angular Material розроблено як розширення для Angular і працює добре з цим фреймворком. Ви можете легко встановити Angular Material, додати необхідні модулі та компоненти до вашого Angular-додатку та розпочати використовувати їх без зайвих зусиль;

					ІС92.290БАК.004 ПЗ	Арк.
						34
Зм.	Лист	№ докум.	Підпис	Дата		

Загалом це найкраща бібліотека готових компонентів для Angular, бо вона офіційно підтримується командою розроблення Angular.

2.4 Вибір середовища розробки

Сучасні середовища розробки пропонують широкий функціонал для максимізації ефективності розробки програмного забезпечення та мінімізації часових витрат.

Серед більшості існуючих середовищ розробки практично немає різниці, бо майже всі вони мають достатньо вбудованого функціоналу для комфортної роботи.

Я обрав середовище IntelliJ IDEA [23], бо вже мав досвід роботи із ним та в порівнянні з іншими середовищами, як VSCode або Notepad воно не поступає у потужності та зручності вбудованого функціоналу.

Загалом, IntelliJ IDEA є потужним інтегрованим середовищем розробки, яке надає розробникам широкий функціонал, зручні інструменти, широку підтримку мов програмування та велику спільноту. Це робить його популярним вибором серед розробників для розробки програмного забезпечення на різних платформах.

Із переваг даного середовища можна виділити те, що IntelliJ IDEA надає широкий набір інструментів та функціональності, які полегшують розробку програмного забезпечення. Він має потужні можливості автодоповнення коду, перехід до визначень функцій та класів, рефакторинг, розуміння контексту, розширені можливості пошуку та заміни, вбудовану систему керування версіями та багато іншого. Все це допомагає прискорити процес розробки та покращити продуктивність розробників.

Також IntelliJ IDEA має вбудовані інструменти, які спрощують розробку програмного забезпечення. Він надає вбудовану систему збирання та пакування проектів, інструменти для автоматичного тестування, профілюingu та аналізу продуктивності, інтегровану підтримку баз даних, серверів додатків та веб-сервісів, інструменти для розробки графічного інтерфейсу та багато іншого. Це дозволяє працювати з повним циклом розробки у межах одного інструменту.

					ІС92.290БАК.004 ПЗ	Арк.
						35
Зм.	Лист	№ докум.	Підпис	Дата		

2.5 Інформаційне забезпечення

Для розуміння принципів роботи програмного застосунку слід навести інформаційний контекст роботи програми, такий як вхідні та вихідні дані та структури даних.

2.5.1 Вхідні дані

Вхідними даними алгоритму рекомендації товарів є двовимірний масив, де рядки – це користувачі, а стовпці – товари. Приклад вхідних даних зображений на рисунку 2.5. Значенням елементів вхідної матриці є оцінка яку надав користувач відповідному товару, яка вимірюється від 0 (користувач не надав оцінки) до 10 (максимальна оцінка).

```
const ratings = [  
  [1, 0, 10, 0, 0],  
  [2, 1, 9, 1, 0],  
  [1, 1, 9, 1, 0],  
  [10, 1, 1, 1, 9],  
  [10, 1, 2, 3, 10],  
];
```

Рисунок. 2.5 – Приклад вхідних даних алгоритму

Також як додаткові параметри, до алгоритма входять коефіцієнт швидкості навчання (learning rate) та параметр регуляризації (regularization parameter).

Шкала навчання вимірюється числом від 0 до 1 та позначає ступінь навчання алгоритму: чим більше значення, тим швидше алгоритм корегує свої фактори та відповідно швидше закінчує процес навчання. Збільшення коефіцієнту зменшує часові витрати на навчання, але водночас і зменшується точність кореляції. Тому бажано знаходити баланс між цими двома факторами.

Параметр регуляризації використовується для контролю за перенавчанням моделі та поліпшення загальної її здатності до узагальнення. Цей параметр

допомагає уникнути занадто високої варіації або надмірної сліпої адаптації моделі до навчальних даних. Він контролює баланс між точністю прогнозів на навчальних даних та загальною здатністю моделі до узагальнення на нові дані. Завдяки регуляризації можна досягти кращої стійкості та знизити ризик перенавчання, що може виникнути при наявності обмеженої кількості навчальних даних.

2.5.2 Вихідні дані

Вихідними даними алгоритму є передбачувана оцінка певного товару конкретним користувачем.

2.5.3 Структура масивів інформації

Вхідна інформація алгоритму представлена у вигляді двовимірного масиву, де рядки – це користувачі, колонки – товари, а значення – оцінка відповідного користувача до продукту.

Висновки до розділу

В даному розділі були розглянуті всі аспекти побудови структури front-end застосунку із переліком використаних патернів проектування та концепцій архітектурної побудови застосунку.

Також були детально описані набори використаних інструментів та сторонніх бібліотек для мінімізації витрат часу на розробку програмного забезпечення. Серед представлених мов та середовищ програмування були наведені переваги та недоліки, які в свою чергу визначили фінальні обрані набори інструментів. Було наведено інформаційне забезпечення застосунку для більш детального ознайомлення із принципами роботи застосунку.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Змістовна постановка задачі

Сучасний веб ринок – це доволі розвинута екосистема різних застосунків, де кожний власник бореться за повернення уваги клієнтів і, відповідно, за максимізацію прибутку. Якщо десятиліття тому користувачеві було достатньо працюючого функціоналу веб сайту, то зараз зайві мілісекунди очікування завантаження сторінки можуть і зовсім підштовхнути покинути веб застосунок.

Із розвитком та популяризацією веб технологій, таких як онлайн магазини та сервіси, рівень конкуренції досяг досить високого рівня і власникам продуктів потрібно винаходити нові, більш ефективніші методи для привернення уваги користувачів.

Саме тому в наш час, де власники веб застосунків борються за кожного клієнта досить важливим фактором є імплементація алгоритму рекомендації товарів, який збільшить потенційний прибуток за рахунок збільшення вірогідності придбання додаткового товару користувачем.

3.2 Математична постановка задачі

Математична задача буде полягати у створенні алгоритму покращеної версії сингулярного розкладу матриць (Singular Value Decomposition++).

Існує багато реалізацій покращених алгоритмів сингулярного розкладу матриць, наприклад із використанням нейронних мереж або комбінуванням із іншими існуючими алгоритмами [24]. Але я вирішив реалізувати свій покращений алгоритм на основі машинного навчання та з використанням додаткових факторів, таких як user/item biases, які відповідають за тенденції користувачів та товарів.

Суть базового SVD методу полягає у декомпозиції матриці вхідних даних розмірності $m \times n$ оцінок товарів користувача на дві окремі матриці явних залежностей спільних рис між користувачами, розмірності $m \times m$ та товарами за відповідними ознаками, розмірності $n \times n$. Значення цих матриць відповідають за

подібність між відповідними користувачами та товарами за конкретними факторами. Змістовна суть цих факторів невідома, це можуть бути такі фактори як подібність до сфери інтересів, наприклад спорт або музика.

Також слід зазначити що після розкладу базової матриці ми ще отримаємо третю, додаткову матрицю рівня впливу цих факторів. Це матриця розмірності $m \times n$ де значення елемента відповідає за вагу відповідного фактора подібності. Як показує досвід, найперші фактори завжди мають більшу вагу в порівнянні з останніми.

Після розкладу вхідної матриці у базовому алгоритмі SVD треба спочатку знайти найбільш подібних користувачів до користувача, для якого ми шукаємо рекомендації, а потім знайти відповідні товари, які найбільш подібні до відповідних знайдених користувачів.

Знайти подібних користувачів можна представивши дані матриці подібності користувачів як координати точок, де сама точка – це користувач. А потім за допомогою Евклідової відстані знайти тих користувачів, відстань від шуканого користувача до яких є найменшою. Але проблема такого підходу полягає у розмірності самих матриць, бо рахування відстані між точками у, наприклад, 1000-мірному просторі буде займати багато часу. Більш того, фактори у матриці впливу розташовані за спаданням ефективності, і відкидання крайніх факторів не відобразиться на фінальному результаті. Саме тому, як оптимізаційне рішення, береться тільки три перших фактора подібності, щоб було просто та швидко представити точки у 3-мірному просторі.

Після того як були знайдені найбільш подібні користувачі до шуканого, аналогічним методом шукаються подібні товари.

Але такого підходу недостатньо для ефективної роботи алгоритму. Також як додаткова міра ефективності алгоритму будуть реалізовані вектори прихильностей до інших користувачів та товарів (user biases/item biases). Їхня суть полягає у виявленні типової поведінки користувачів щодо оцінювання товарів, наприклад коли користувач завжди виставляє позитивні/негативні відгуки. Так само і до

товарів — коли якийсь товар майже всі користувачі оцінюють позитивно або негативно.

Для реалізації машинного навчання буде реалізована система оцінки роботи алгоритму та рахування середньоквадратичної помилки/відхилення від номінального значення вже відомої оцінки за якийсь товар (1).

$$MSE = \frac{\sum (predictions[i] - target[i])^2}{n}, \quad (3.1)$$

де MSE – середньоквадратичне відхилення;

n – кількість елементів;

$predictions[i]$ – прогнозована оцінка товару i ;

$target[i]$ – фактична оцінка товару i ;

Саме навчання буде реалізоване за допомогою стохастичного градієнтного спуску. У алгоритмі SVD++ (Singular Value Decomposition++) стохастичний градієнтний спуск використовується для оновлення параметрів моделі на кожному кроці навчання. Процес навчання полягає в пошуку оптимальних значень параметрів, щоб мінімізувати функцію втрати.

Стохастичний градієнтний спуск у алгоритмі SVD++ дозволяє ефективно навчати модель на великих обсягах даних, оскільки він працює з випадковими підмножинами даних, замість використання усіх навчальних прикладів. Це дозволяє швидше знаходити локальні мінімуми функції втрати та покращувати точність моделі з кожним оновленням параметрів. Однак, через стохастичний характер оновлення, процес може мати більшу варіацію та може знадобитися додатковий контроль за процесом навчання, такий як налаштування швидкості навчання.

Для роботи даного алгоритму треба використати формули оновлення базових та факторних предикторів.

У стохастичному градієнтному спуску (SGD) у алгоритмі SVD++ формули для оновлення параметрів моделі виглядають наступним чином:

					IC92.290БАК.004 ПЗ	Арк.
						40
Зм.	Лист	№ докум.	Підпис	Дата		

3.2.1 Оновлення базових предикторів (biases)

Для користувача u :

$$b_u[u] = b_u[u] + learningRate * (err - lambda * b_u[u]), \quad (3.2)$$

де $b_u[u]$ – значення предиктора;

$learningRate$ – швидкість навчання, яка визначає, наскільки великими кроками оновлюються параметри моделі;

err – відхилення від фактичного результату

$lambda$ – параметр регуляризації, який контролює вагу регуляризації при оновленні параметрів;

Для товару v :

$$b_v[v] = b_v[v] + learningRate * (err - lambda * b_v[v]), \quad (3.3)$$

де $b_v[v]$ – значення предиктора;

$learningRate$ – швидкість навчання, яка визначає, наскільки великими кроками оновлюються параметри моделі;

err – відхилення від фактичного результату

$lambda$ – параметр регуляризації, який контролює вагу регуляризації при оновленні параметрів;

3.2.2 Оновлення факторних предикторів:

Для користувача u , товару v та кожного фактору f :

$$u_f[u][f] = u_f[u][f] + learningRate * (err * v_f[v][f] - lambda * u_f[u][f]), \quad (3.4)$$

де $u_f[u][f]$ – значення предиктора користувача u фактора f ;

$v_f[v][f]$ – значення предиктора товару v фактора f ;

Для товару v , користувача u , та кожного фактору f :

$$v_f[v][f] = v_f[v][f] + learningRate * (err * u_f[u][f] - lambda * v_f[v][f]), \quad (3.5)$$

де $v_f[v][f]$ – значення предиктора товару v фактора f ;

$u_f[u][f]$ – значення предиктора товару u фактора f ;

3.3 Обґрунтування методів розв'язання

Розроблений мною алгоритм SVD++ заснований на базовому алгоритмі сингулярного розкладу матриць (SVD), але із додатковими покращеннями, такими як впровадження базових предикторів користувачів та товарів (user/item biases) та використання алгоритму машинного навчання (machine learning) в основі якого лежить циклічне оновлення факторів за допомогою стохастичного градієнтного спуску допоки середньоквадратична помилка не буде мінімальною.

У алгоритмі SVD++ user biases (користувацькі зміщення) та item biases (зміщення предметів) використовуються для моделювання індивідуальних вподобань користувачів та властивостей предметів, які не враховуються безпосередньо у взаємодії між користувачами та предметами.

User biases (користувацькі зміщення) представляють собою відхилення або "зміщення" від загального середнього рейтингу, яке може бути спричинене індивідуальними смаками або уподобаннями користувачів. Наприклад, деякі користувачі можуть загалом більше схильні ставити високі рейтинги, тоді як інші можуть бути більш критичними та ставити нижчі рейтинги. User biases дозволяють врахувати ці відмінності в оцінках користувачів і забезпечити більш індивідуалізовані рекомендації.

Item biases (зміщення продуктів) враховують особливості та характеристики самого продукту, які можуть впливати на рейтинги, незалежно від конкретного користувача. Наприклад, деякі продукти можуть бути загалом більш популярними або краще оціненими, що може викликати вищі рейтинги взагалі. Item biases дозволяють враховувати ці властивості предметів та відрізнити їх від інших на основі цих характеристик.

Використання user biases та item biases у алгоритмі SVD++ дозволяє моделювати індивідуальні уподобання користувачів та властивості предметів, що дозволяє розширити спектр аналізу та отримати більше унікальної інформації щодо користувачів та товарів. Ці зміщення додаються до загального прогнозу рейтингу щоб врахувати індивідуальні відмінності та особливості.

3.4 Специфікація функцій застосунку

Наступна таблиця 3.1 містить опис функцій класу алгоритму.

Таблиця 3.1 – Специфікації функцій застосунку

Функція	Опис
train	Тренує саму модель, ітераційно змінюючи відповідні предиктори допоки середньоквадратична похибка не буде мінімальною
predict	Повертає спрогнозоване значення оцінки товару користувачем на основі натренованих предикторів
dotProduct	Допоміжна функція для знаходження скалярного добутку двох векторів

Висновки до розділу

У даному розділі були розглянуті змістовні та математичні постановки задач, у яких були розібрані усі етапи реалізації та роботи покращеного алгоритму розкладу матриць SVD++.

Були роз'яснені принципи роботи базових предикторів (biases) та факторних предикторів (factors).

Був розглянутий основний процес тренування предикторів на основі стохастичного градієнтного спуску допоки середньоквадратична похибка не буде мінімальною.

Було наведено обґрунтування методів розв'язання та наведена специфікація функцій застосунку

					ІС92.290БАК.004 ПЗ	Арк.
						44
Зм.	Лист	№ докум.	Підпис	Дата		

4 ДОСЛІДЖЕННЯ ТА ЕКСПЕРИМЕНТИ

4.1 Керівництво користувача

Для початку роботи програми потрібно мати встановлене середовище NodeJS та відкрити проєкт у будь якому середовищі розробки, наприклад IntelliJIDEA.

Після відкриття проєкту треба лише прописати команду “ng serve” у командному рядку (CLI) та перейти до браузера за посиланням <http://localhost:4200>.

По замовчуванню якщо користувач не авторизований, то першою сторінку веб застосунку буде аутентикаційна сторінка (login page), де користувачу на вибір буде запропоновано або увійти до сайту, або зареєструватись. Дана сторінка зображена на рисунку 4.1.

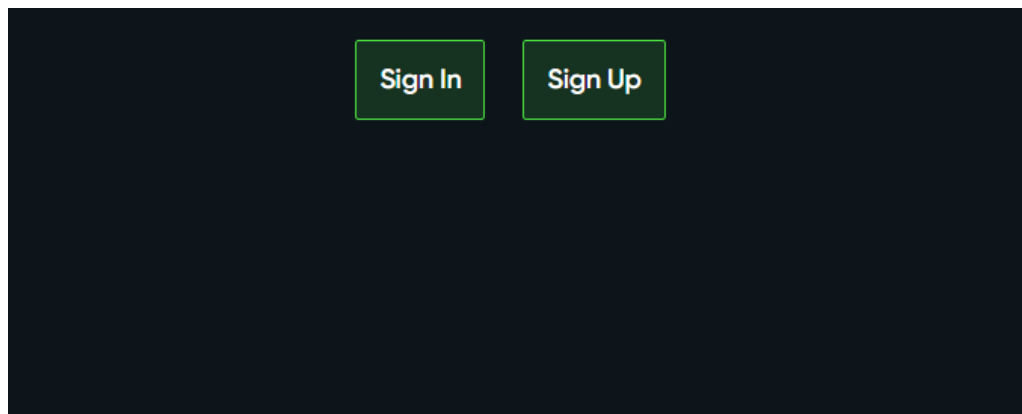


Рисунок 4.1 – Аутентикаційна сторінка застосунку

Якщо користувач натисне кнопку “увійти” (Sign In), то з’явиться відповідне модальне вікно для вводу особистих даних. Дане вікно має відповідну html форму для заповнення особистих даних користувача, а саме його електронна скринька та пароль. Також на цьому модальному вікні реалізовані кнопки для закриття вікна та відправки відповідного запиту на сервер.

Кнопка для аутентифікації буде заблокована до тих пір, поки користувач не введе усі обов’язкові поля для заповнення.

Після успішного запиту на сервер, користувач буде автоматично перенесений на головну сторінку веб-застосунку.

Дане вікно зображене на рисунку 4.2.

					IC92.290БАК.004 ПЗ	Арк.
						45
Зм.	Лист	№ докум.	Підпис	Дата		

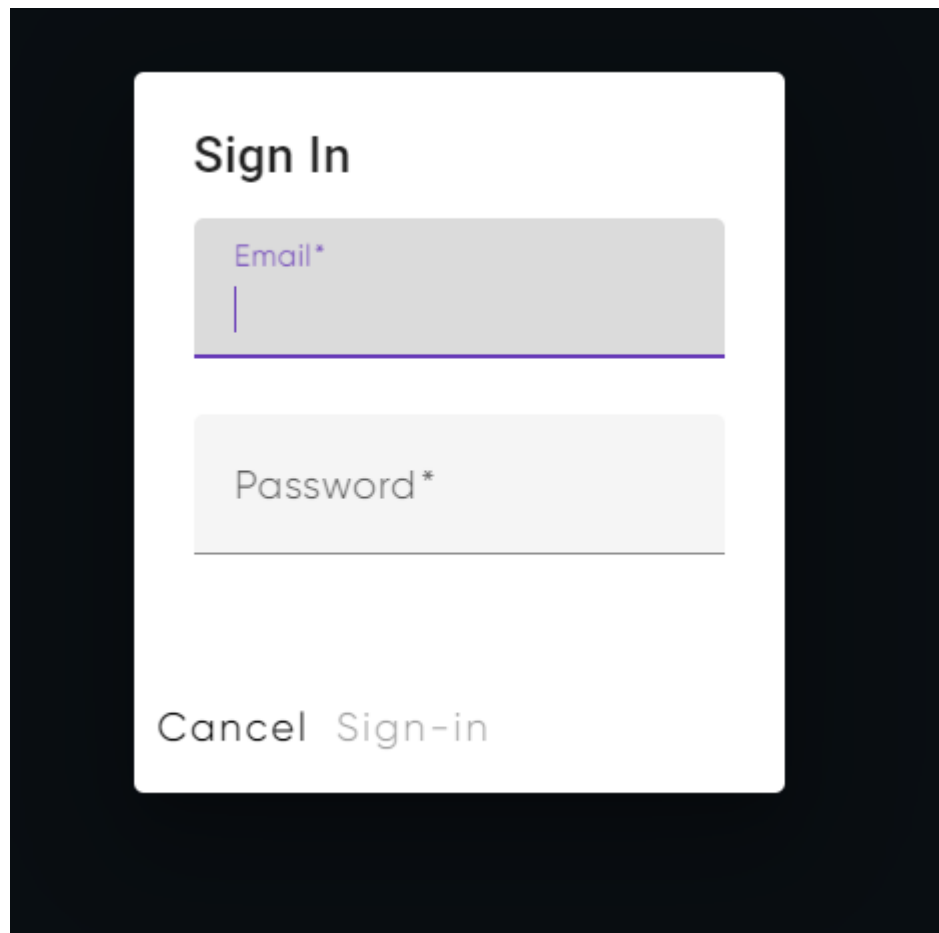


Рисунок 4.2 – Модальне вікно для входу в особистий аккаунт

Якщо користувач натисне кнопку “zareestruватися” (Sign Up), то з’явиться відповідне модальне вікно для вводу особистих даних. Дане вікно має відповідну html форму для заповнення особистих даних користувача, а саме його електронна скринька, ім’я, новий пароль та повторне введення паролю. Також на цьому модальному вікні реалізовані кнопки для закриття вікна та відправки відповідного запиту на сервер.

Кнопка для реєстрації буде заблокована до тих пір, поки користувач не введе усі обов’язкові поля для заповнення.

Після успішного запиту на сервер, користувач буде автоматично перенесений на головну сторінку веб-застосунку.

Дане вікно зображене на рисунку 4.3.

					ІС92.290БАК.004 ПЗ	Арк.
						46
Зм.	Лист	№ докум.	Підпис	Дата		

Рисунок 4.3 – Модальне вікно для реєстрації

Допоки відповідні обов’язкові поля не заповнені, кнопка реєстрації/входу буде заблокована. Але якщо ввести всі дані правильно – то відповідна кнопка стане активною.

Для прикладу заповнемо відповідну форму наступними даними:

- test6@re.re, як електронна скринька користувача;
- Tiguran-Rizzrak, як ім’я користувача;
- 12341234, як пароль;
- 12341234, як повторний пароль;

Після заповнення всіх потрібних даних, можемо побачити що відповідна кнопка стала активною.

Приклад заповнених даних зображено на рисунку 4.4.

Sign Up

Email*
test6@re.re

Name*
Tiguran-Rizzrak

Password*
.....

Repeat password*
.....

Cancel Sign-up

Рисунок 4.4 – Правильно заповнені дані модального вікна

Після того як користувач натисне відповідну кнопку, при умові успішного запиту його має перевести на головну сторінку із списком доступних товарів для покупки.

При заході на головну сторінку, застосунок автоматично надає перелік усіх доступних товарів із зображенням рисунку, ціни та назви відповідного товару.

Також кожний товар має кнопку для придбання, при натисканні на яку при умові успішної покупки користувачу буде показане модальне вікно із пропозицією залишити особистий відгук про придбаний товар.

Вигляд даної сторінки можна побачити на рисунку 4.5.

					IC92.290БАК.004 ПЗ	Арк.
						48
Зм.	Лист	№ докум.	Підпис	Дата		

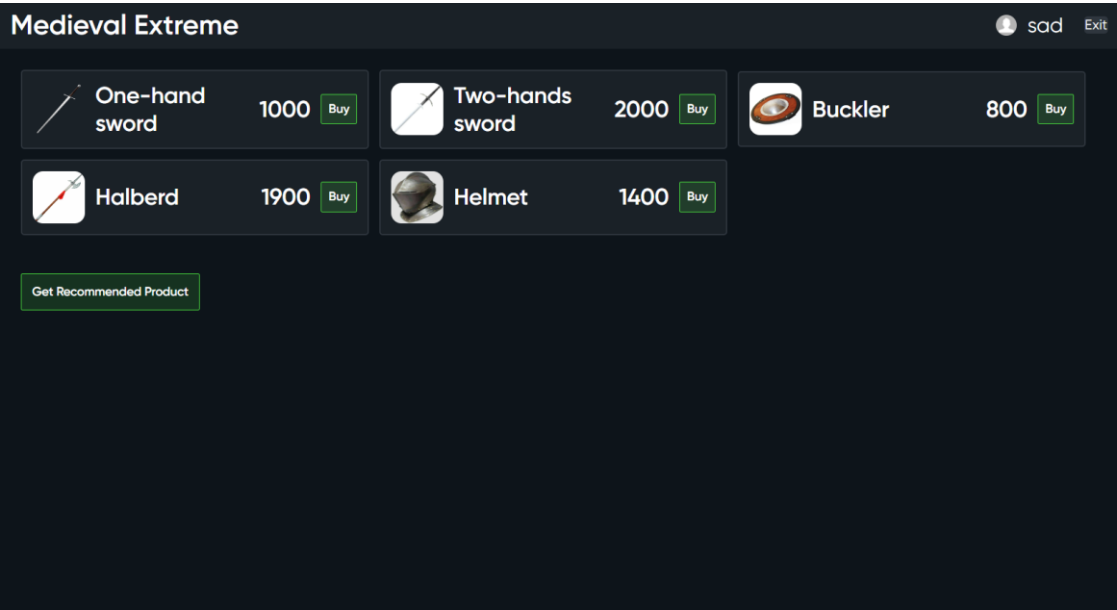


Рисунок 4.5 – Головна сторінка застосунку

На головній сторінці користувач може бачити список усіх наявних товарів у магазині. При натисканні на кнопку “Придбати” (Buy), після того як товар був придбаний буде з’являтися модальне вікно для оцінки придбаного товару. Дане вікно було розроблено для того щоб була можливість збирати дані для роботи алгоритму. Приклад даного модального вікна зображений на рисунку 4.6.

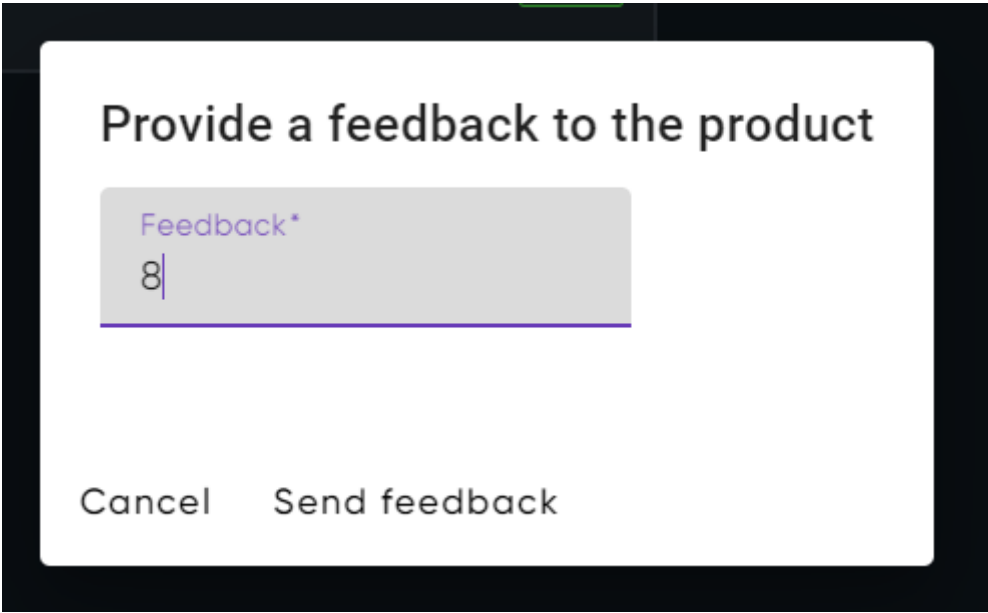


Рисунок 4.6 – Модальне вікно для оцінки придбаного товару

Після успішного підтвердження фідбеку про товар – модальне вікно зачиняється.

Також у верхній частині головної сторінки є кнопка для виходу із магазину (Exit), при натисканні на яку видалиться сесія користувача та сторінка перенаправиться на аутентикацію.

При натисканні на кнопку “Отримати рекомендацію” (Get recommendation) – запусниться виконання коду самого алгоритму, який на основі існуючих оцінок усіх користувачів поверне та відповідно відобразить рекомендований товар на головній сторінці. Даний функціонал зображений на рисунку 4.7.

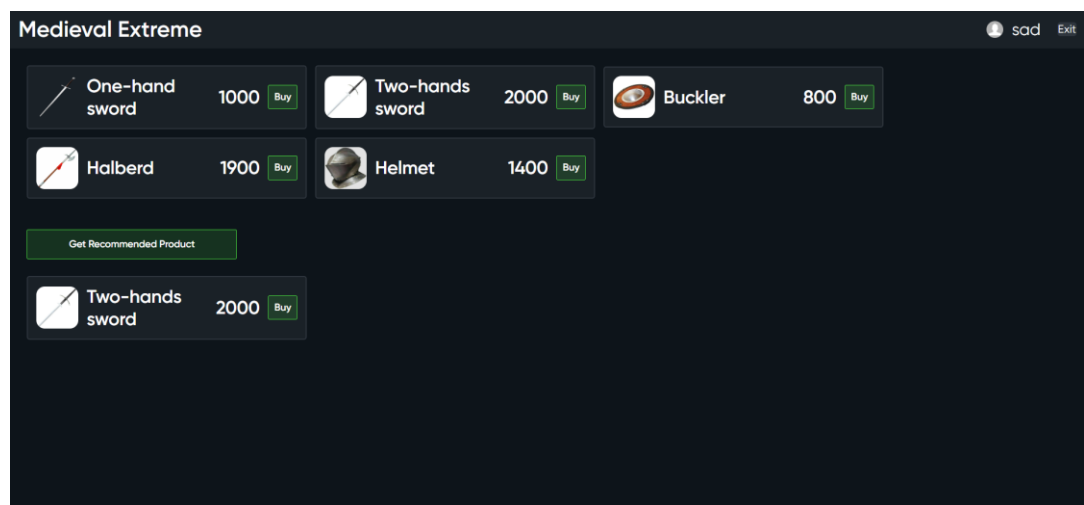


Рисунок 4.7 – Відображення рекомендованого товару після натискання на відповідну кнопку

4.2 Випробування програмного продукту

4.2.1 Мета випробувань

Метою випробувань є впевнення у правильній роботі алгоритму. Для наявної демонстрації обсяг випробувальних даних буде невеликим та інтуїтивним, щоб можна було евристично зрозуміти чи правильно спрацював алгоритм рекомендації товарів. Для цього буде використаний датасет із п'ятьма користувачами та товарами.

					ІС92.290БАК.004 ПЗ	Арк.
						50
Зм.	Лист	№ докум.	Підпис	Дата		

4.2.2 Результати випробувань

Для тестування алгоритму візьмемо наступний датасет, що зображений на рисунку 4.8.

	Item1	Item2	Item3	Item4	Item5
User1	9	0	3	0	0
User2	10	9	3	8	5
User3	10	8	2	10	5
User4	1	2	8	2	9
User5	2	1	9	3	10

Рисунок 4.8 – Приклад тестувального датасету

На цьому прикладі слід виділити наступне: тестувальний користувач для якого ми будемо шукати товари для рекомендації – це User1. Цей користувач вже придбав перший та третій товари, відповідно ми будемо шукати спрогнозовані оцінки для решти товарів та алгоритм має вибрати найбільш привабливий з них.

Із вхідних даних ми можемо підкреслити наступне: User2 та User3 дали дуже схожі оцінки для товарів, які також придбав шуканий користувач, а саме:

- перший товар це 10 та 10 до 9;
- третій товар це 2 та 3 до 3;

А користувачі User4 та User5 навпаки, практично протилежні:

— перший товар це 1 та 2 до 9;

— третій товар це 8 та 9 до 3;

Узагальнення даної інформації представлено на рисунку 4.9.

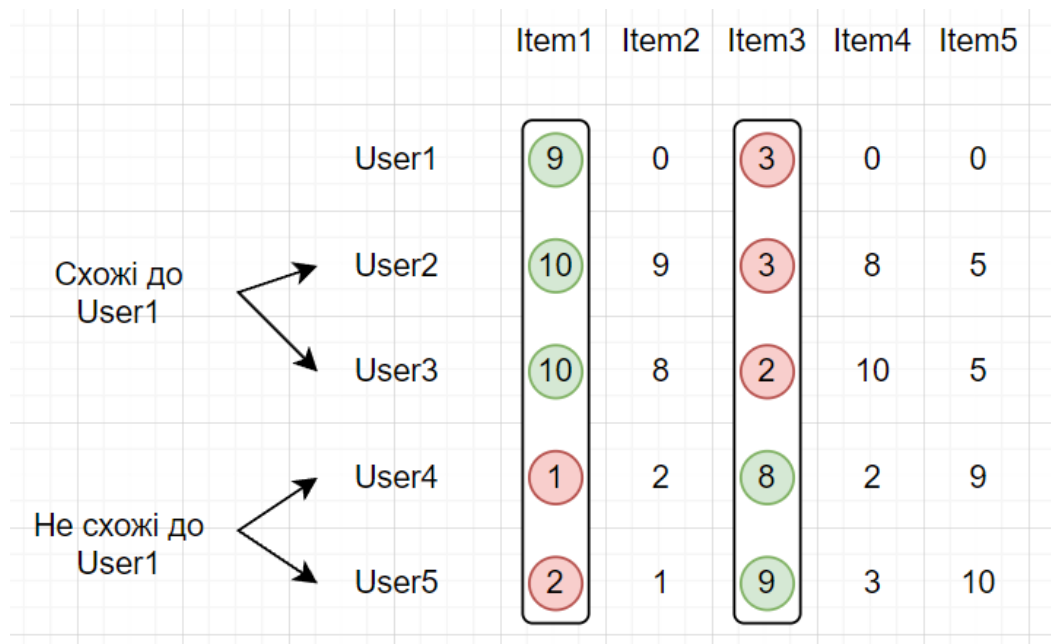


Рисунок 4.9 – Розділення користувачів за подібністю

Із цих даних можемо зробити наступний висновок – оскільки SVD++ алгоритм базується на collaborative-filtering підході, тому він буде відштовхуватися від подібних користувачів. А серед оцінок User2 та User3 найбільші мають товари Item2 та Item4, де Item4 навіть трохи більше.

При запуску алгоритму на цьому датасеті отримаємо наступні прогнозовані оцінки:

— 7.447 для Item2;

— 7.978 для Item4;

— 5.039 для Item5;

На основі цих даних можемо зробити висновок що алгоритм працює правильно.

Також для тестування роботи front-end застосунку створимо такий же датасет, зайдемо від імені першого користувача та натиснемо кнопку “Отримати рекомендацію”.

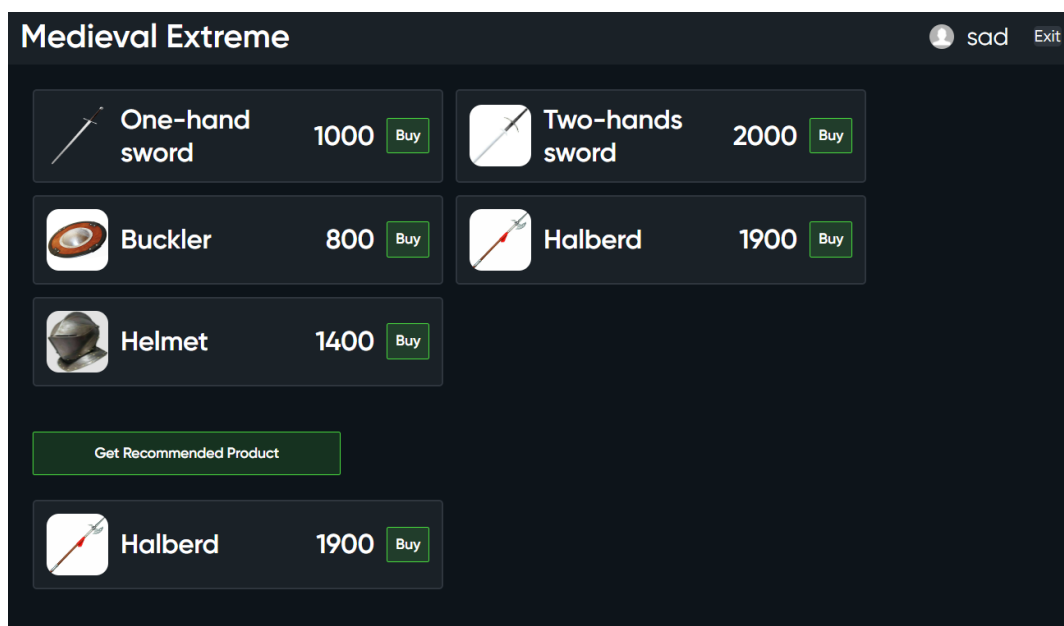


Рисунок 4.10 – Результат роботи застосунку

Як можемо бачити, як рекомендований товар з'явився товар “Halberd”, що є під індексом 4. Це можна побачити на рисунку 4.10. Виходячи з цього можемо зробити висновок що програма працює правильно.

4.2.3 Навантажувальне тестування та аналіз швидкодії

Для аналізу швидкодії розробленого алгоритму було проведено навантажувальне тестування на різних наборах даних, починаючи від 10 тис. оцінок і закінчуючи матрицею на 1 млн. унікальних елементів. На даних наборах даних матриці не розріджені, тобто не мають нулів. Це було зроблено щоб спростити тестування шляхом зменшення розмірності вхідної матриці.

Заміри часу були проведені по три рази для кожного набору даних та був порахований середній час для кожного набору.

Перший етап тестування був проведений на наборі даних у 10 тис. унікальних оцінок серед 100 користувачів та 100 товарів. Результати даного заміру зображені на рисунку 4.11.

```
"C:\Program Files\nodejs\node.exe" C:\Users\OtecValerian\IdeaProjects\rozkladu-algorithms\main.js
1: Time taken for 100 unique users, 100 unique items and 10000 feedbacks: 1.719 s
2: Time taken for 100 unique users, 100 unique items and 10000 feedbacks: 1.738 s
3: Time taken for 100 unique users, 100 unique items and 10000 feedbacks: 1.649 s
The mean time is: 1.702 s

Process finished with exit code 0
```

Рисунок 4.11 – результат тестування на 10 тис. елементах

Другий етап тестування був проведений на наборі даних у 100 тис. унікальних оцінок серед 1000 користувачів та 100 товарів. Результати даного заміру зображені на рисунку 4.12.

```
"C:\Program Files\nodejs\node.exe" C:\Users\OtecValerian\IdeaProjects\rozkladu-algorithms\main.js
1: Time taken for 1000 unique users, 100 unique items and 100000 feedbacks: 2.110 s
2: Time taken for 1000 unique users, 100 unique items and 100000 feedbacks: 1.886 s
3: Time taken for 1000 unique users, 100 unique items and 100000 feedbacks: 2.018 s
The mean time is: 2.005 s

Process finished with exit code 0
```

Рисунок 4.12 – результат тестування на 100 тис. елементах

Третій етап тестування був проведений на наборі даних у 500 тис. унікальних оцінок серед 2000 користувачів та 250 товарів. Результати даного заміру зображені на рисунку 4.12.

```
"C:\Program Files\nodejs\node.exe" C:\Users\OtecValerian\IdeaProjects\rozkladu-algorithms\main.js
1: Time taken for 2000 unique users, 250 unique items and 500000 feedbacks: 5.683 s
2: Time taken for 2000 unique users, 250 unique items and 500000 feedbacks: 5.967 s
3: Time taken for 2000 unique users, 250 unique items and 500000 feedbacks: 5.697 s
The mean time is: 5.782 s

Process finished with exit code 0
```

Рисунок 4.13 – результат тестування на 500 тис. елементах

Четвертий етап тестування був проведений на наборі даних у 1 млн. унікальних оцінок серед 1000 користувачів та 1000 товарів. Результати даного заміру зображені на рисунку 4.14.

```
"C:\Program Files\nodejs\node.exe" C:\Users\OtecValerian\IdeaProjects\rozkladu-alghoritms\main.js
1: Time taken for 1000 unique users, 1000 unique items and 1000000 feedbacks: 53.837 s
2: Time taken for 1000 unique users, 1000 unique items and 1000000 feedbacks: 53.212 s
3: Time taken for 1000 unique users, 1000 unique items and 1000000 feedbacks: 57.334 s
The mean time is: 54.794 s

Process finished with exit code 0
```

Рисунок 4.14 – результат тестування на 1 млн. елементах

Усі отримані результати можна побачити у таблиці 4.1.

Таблиця 4.1 – Результати тестування на швидкодію

№ спроби	Об'єм даних			
	10 тис оцінок	100 тис оцінок	500 тис оцінок	1 млн оцінок
1	1,719 с	2,110 с	5,683 с	53,837 с
2	1,738 с	1,886 с	5,967 с	53,212 с
3	1,649 с	2,018 с	5,697 с	57,334 с
Середнє значення	1,702 с	2,005 с	5,782 с	54,794 с

Висновок до розділу

У даному розділі було представлено базове керівництво користувача із покроковою інструкцією щодо роботи із застосунком.

Було проведено випробування програмного забезпечення на тестових даних. Результат виявився успішним як у алгоритмі, так і у застосунку.

Були представлені результати випробування, а саме результати евристичного тестування на малому наборі даних та навантажувального тестування для більш великих даних.

Із результатів можна побачити що на великих даних час роботи алгоритму досить великий, що сягає майже хвилину. Щоб не зіпсувати модель поведінки сайту, бажано в майбутньому реалізувати websocket з'єднання із сервером, який буде надсилати рекомендовані товари коли алгоритм завершить свою роботу, а на самому сайті в цей час поки не показувати результати.

ВИСНОВКИ

В даній пояснювальній записці до дипломного проєкту було розглянуто більшість найпоширеніших видів алгоритмів рекомендацій товарів, їхні реалізації та математичні моделі. Був проведений порівняльний аналіз усіх існуючих алгоритмів із виділенням унікальних переваг та недоліків.

В першому розділі було проведено аналіз видів систем рекомендацій, а саме: колаборативна фільтрація, алгоритм заснований на контенті та факторизація матриці. Було проведено порівняння між собою усіх видів алгоритмів та виділені ключові недоліки та переваги в порівнянні один із одним.

В другому розділі була представлена структура системи. Був пояснений та обґрунтований вибір мови програмування, середовища розробки та використання сторонніх бібліотек. Була детально описана структура front-end застосунку та перелічені усі сучасні технології розробки разом із використанням патернів архітектурного проектування. Також було представлено інформаційне забезпечення у вигляді опису вхідних та вихідних даних та структури масивів інформації.

У третьому розділі була представлена змістовна та математична постановки задач із детальним поясненням математичного підґрунтя стосовно реалізації самого алгоритму.

У четвертому розділі було проведено дослідження правильності роботи алгоритму та застосунку за допомогою експерименту.

Провівши дану роботу було доведено працездатність даного алгоритму та застосунку, виконані усі пункти поставленого завдання та продемонстровано коректну роботу логіки програми. Метою даної розробки була реалізація алгоритму машинного навчання для отримання рекомендацій товарів із побудовою веб застосунку. Саме таке поєднання сучасних методів веб розробки із математичним фундаментом дозволяє отримати високі комерційні оцінки застосунку із подальшою можливістю масштабованості магазину, бо саме завдяки правильній

архітектурі проєкту – така функціональність як масштабованість чи додавання нових функціональних моделей це лише питання часу.

					ІС92.290БАК.004 ПЗ	Арк.
						57
Зм.	Лист	№ докум.	Підпис	Дата		

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Badrul Sarvar (1998). Item-based Collaborative Filtering Recommendation Algorithms, 3р.
2. Yanush Sarvar (1999). Collaborative Filtering for Implicit Feedback Datasets, 4р.
3. Frank Kane (2021). Building Recommender Systems with Machine Learning and AI: Help people discover new products and content with deep learning, neural networks, and machine learning recommendations, 8-48р.
4. K. Singh, N. Horter (2007). Collaborative Filtering Recommender Systems, 21-44р.
5. Recommendation System - Understanding The Basic Concepts by Shubham Kumar Agrawal, Analytics Vidhya [Електроний ресурс] – Режим доступу до ресурсу:
<https://www.analyticsvidhya.com/blog/2021/07/recommendation-system-understanding-the-basic-concepts/>
6. A Novel Hybrid Deep Neural Network for Predicting Athlete Performance Using Dynamic Brain Waves by Yu-Hung Tsai and MDPI [Електроний ресурс] – Режим доступу до ресурсу:
<https://www.mdpi.com/2227-7390/11/4/903>
7. A hybrid paper recommendation system by Xiangjie Kong and Research Gate [Електроний ресурс] – Режим доступу до ресурсу:
https://www.researchgate.net/figure/A-hybrid-paper-recommendation-system_fig5_330077673
8. Content based and Collaborative Filtering by Lucas Parisi [Електроний ресурс] – Режим доступу до ресурсу:
https://www.youtube.com/watch?v=VGqkHZHuCXg&ab_channel=LucasParisi
9. You will get a Natural Processing Topic Modelling Algorithm by K. Usman and Upwork [Електроний ресурс] – Режим доступу до ресурсу:

<https://www.upwork.com/services/product/development-it-a-natural-language-processing-nlp-topic-modelling-algorithm-1388547297827536896>

10. Foundation Maths: Matrices – Part 5: Inverse Matrix by ChemEngTutor [Електроний ресурс] – Режим доступу до ресурсу:

https://www.youtube.com/watch?v=yzXhagRHIWE&ab_channel=ChemEngTutor

11. Illustration of the matrix factorization method by ResearchGate [Електроний ресурс] – Режим доступу до ресурсу:

https://www.researchgate.net/figure/Illustration-of-the-matrix-factorization-method-In-this-example-the-highlighted-cells_fig8_329128825

12. What is DRY Development? By DigitalOcean [Електроний ресурс] – Режим доступу до ресурсу:

<https://www.digitalocean.com/community/tutorials/what-is-dry-development>

13. Lazy-loading feature modules by Angular documentation [Електроний ресурс] – Режим доступу до ресурсу:

<https://angular.io/guide/lazy-loading-ngmodules>

14. Reactive and Functional Programming Design Patterns by Muhammet Ali Akbay [Електроний ресурс] – Режим доступу до ресурсу:

<https://medium.com/swlh/reactive-and-functional-programming-design-patterns-23da906d2939>

15. NGRX Reactive State for Angular [Електроний ресурс] – Режим доступу до ресурсу:

<https://ngrx.io/>

16. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns, 8-411p.

17. Sass: Syntactically Awesome Style Sheets by SASS documentation [Електроний ресурс] – Режим доступу до ресурсу:

<https://sass-lang.com/>

18. Angular service worker introduction by Angular documentation [Електроний ресурс] – Режим доступу до ресурсу:

<https://angular.io/guide/service-worker-intro>

					IC92.290БАК.004 ПЗ	Арк.
						59
Зм.	Лист	№ докум.	Підпис	Дата		

19. Javascript by official documentation [Електроний ресурс] – Режим доступу до ресурсу:

<https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>

20. NodeJS by official documentation [Електроний ресурс] – Режим доступу до ресурсу:

<https://nodejs.org/en>

21. RxJS by official documentation [Електроний ресурс] – Режим доступу до ресурсу:

<https://rxjs.dev/>

22. Angular Material by official documentation [Електроний ресурс] – Режим доступу до ресурсу:

<https://material.angular.io/>

23. IntelliJ IDEA by official documentation [Електроний ресурс] – Режим доступу до ресурсу:

<https://www.jetbrains.com/>

24. Algorithm for recommendation system by cheungdaven [Електроний ресурс] – Режим доступу до ресурсу:

<https://github.com/cheungdaven/recommendation>