Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
БАКАЛАВРА

на тему *Рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент групи КН-20-2* (Група виконавця) _____ (Підпис) *Кирило МУКОМЕЛА* (Ім'я, ПРІЗВИЩЕ)

Керівник: *к.т.н., доц. каф. КН* (Науковий ступінь, посада) _____ (Підпис) *Руслан БАГРІЙ* (Ім'я, ПРІЗВИЩЕ)

Нормоконтроль: *к.т.н., доц. каф. КН* (Науковий ступінь, посада) _____ (Підпис) *Руслан БАГРІЙ* (Ім'я, ПРІЗВИЩЕ)

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ (Підпис) *Олександр БАРМАК* (Ім'я, ПРІЗВИЩЕ)

18 червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор Олександр БАРМАК
«*16*» *лютого* *2024* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв»*

2. Завдання видано студенту *Кирилу МУКОМЕЛІ*
(Ім'я, прізвище)

3. Керівник роботи *доцент кафедри КН Руслан БАГРІЙ*
(посада, ім'я, прізвище)

4. Затверджено наказом університету від «*16*» *лютого* *2024* р. № *8*

5. Дата видачі завдання студенту: «*16*» *лютого* *2024* р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Метою кваліфікаційної роботи є спрощення навігації користувачів у вебсистемі мобільних пристроїв за допомогою методу формування рекомендацій на основі контентного фільтрування. При створенні методу враховуються такі параметри як: покупки клієнтів системи, історія перегляду товарів клієнтів, список бажаного клієнтів та всі товари, що наявні у базі для коректної роботи. Також слід забезпечити правильну роботу функцій додавання, редагування, видалення та перегляду товарів замовлень користувачів, функції додавання товарів у кошик та створення замовлення у роботі системи, одержання виводу роботи методу рекомендацій.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником, складання календарного графіка виконання роботи*	*січень 2024*	*виконано*
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*лютий 2024*	*виконано*
3	*Робота над розділом 1 - Персоналізовані системи рекомендацій для пропозицій товарів*	*березень 2024*	*виконано*
4	*Робота над розділом 2 - Рекомендаційний метод на основі контентного фільтрування*	*квітень 2024*	*виконано*
5	*Робота над розділом 3 - Програмна реалізація вебсистеми з методом формування рекомендацій мобільних пристроїв*	*травень 2024*	*виконано*
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2024*	*виконано*
7	*Попередній захист кваліфікаційної роботи бакалавра*	*червень 2024*	*виконано*
8	*Захист кваліфікаційної роботи*	*червень 2024*	*виконано*

Виконавець: *студент групи КН-20-2* _____ *Кирило МУКОМЕЛА*
Група виконавця | Підпис | Ім'я, ПРІЗВИЩЕ

Керівник: *к.т.н., доц. каф. КН* _____ *Руслан БАГРІЙ*
Науковий ступінь, посада | Підпис | Ім'я, ПРІЗВИЩЕ

Анотація

Тема кваліфікаційної роботи бакалавра: *«Рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-20-2 Кирило МУКОМЕЛА*

Керівник кваліфікаційної роботи бакалавра: *к.т.н., доцент кафедри КН Руслан БАГРІЙ*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
68	13	25	26	3

Метою кваліфікаційної роботи є спрощення навігації користувачів у веб системі мобільних пристроїв за допомогою рекомендаційного методу на основі контентного фільтрування для вебсистеми мобільних пристроїв.

Розроблений рекомендаційний метод може бути використаний для персоналізації контенту у вебсистемах. Це може допомогти користувачам знаходити контент, який відповідає їх потребам, а також може допомогти веб-сайтам покращити досвід користувача.

Практичне використання полягає у тому, що зростання популярності вебсистем та наявність користувачів мобільних пристроїв, що мають обмежений час та ресурси, тому особливо важливо надавати персоналізований контент, який відповідає інтересам користувачів.

Ключові слова: вебсистема, інтелектуальний пошук, контент, персоналізація контенту, рекомендаційний метод.

Виконавець: *студент групи КН-20-2* _____ *Кирило МУКОМЕЛА*

Група виконавця | Підпис | Ім'я, ПРІЗВИЩЕ

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
NN	The K-Nearest Neighbors algorithm
ER	Entity-Relationship
CPU	Central Processing Unit
GPU	Graphic Processing Unit
NFC	Near Field Communication
RAM	Random Access Memory
TFRS	TensorFlow Recommenders
ES6/7	ECMAScript version 6/7
SQL	Structured Query Language
API	Application Programming Interface
js	Javs script
IDE	Integrated Development Environment
MVC	Model – View – Controller architecture
MSCR	Moder – Service – Controller - Router
CRUD	create read update delete
FSD	Feature-Sliced Design
URL	Uniform Resource Locator
RTK	Redux Tool Kit
npm	Node Package Manager

Вступ

У сучасному світі, де обсяг інформації постійно зростає, стає все складніше користувачам знаходити те, що їм дійсно цікаво. Рекомендаційні системи приходять на допомогу, пропонуючи персоналізовані рекомендації контенту на основі інтересів та уподобань користувачів.

Одним із поширених основ для рекомендаційного методу є алгоритм контентного фільтрування. Цей метод ґрунтується на аналізі контенту, який користувач вже вподобав, та на цій основі рекомендує схожий контент.

Метою кваліфікаційної роботи є спрощення навігації користувачів у вебсистемі мобільних пристроїв за допомогою методу формування рекомендацій на основі контентного фільтрування.

Для досягнення поставленої мети необхідно реалізувати виконання наступних задач:

– провести аналіз систем рекомендацій на основі методу контентного фільтрування;

– розробити метод формування рекомендацій мобільних пристроїв на основі методу контентного фільтрування;

– розробити функціональну та інформаційну структуру вебсистеми мобільних пристроїв;

– розробити програмну реалізацію вебсистеми мобільних пристроїв з використанням методу для формування рекомендацій на основі методу контентного фільтрування;

– провести тестування розробленого методу формування рекомендацій мобільних пристроїв

Об'єктом дослідження є процес формування рекомендацій мобільних пристроїв з використанням методу контентного фільтрування.

Предметом дослідження є методи збору та аналізу інформації, методи контентного фільтрування, технології та методи проектування вебсистем мобільних пристроїв.

Практична значимість полягає в тому, що розроблена в рамках даної кваліфікаційної роботи рекомендаційний метод може бути використаний для персоналізації контенту в вебсистемах мобільних пристроїв. Це може допомогти користувачам знаходити контент, який їм цікавий, а також може допомогти веб-сайтам збільшити час, який користувачі проводять на сайті.

Актуальність дослідження полягає у тому, що зростання популярності вебсистем на мобільних пристроях робить контентний фільтрування ще більш актуальним. Користувачі мобільних пристроїв мають обмежений час та ресурси, тому їм особливо важливо отримувати персоналізований контент, який відповідає їхнім інтересам.

Очікувані результати

В результаті виконання кваліфікаційної роботи очікується розроблена та імплементований рекомендаційний метод на основі методу контентного фільтрування для вебсистеми мобільних пристроїв.

Розділ 1 Персоналізовані системи рекомендацій для пропозицій товарів

1.1 Огляд персоналізованих систем рекомендацій

Рекомендаційні системи прийнято вважати інструментом для фільтрації інформації, що надає індивідуально підібрані пропозиції щодо елементів, які найбільше відповідають інтересам конкретного користувача. Такі системи особливо корисні в ситуаціях, коли є необхідність вибрати товар з величезної кількості доступних варіантів, які може запропонувати певна послуга або платформа [1].

З розвитком і зростанням обсягу доступного онлайн-контенту користувачі все більше стикаються з проблемою вибору. Величезна кількість інформації часто призводить до інформаційного перевантаження, що ускладнює процес прийняття рішень. У зв’язку з цим є важливим, щоб веб-платформи впроваджували ефективні рекомендаційні системи. Це дозволяє надавати кожному користувачеві персоналізовані рекомендації щодо елементів, які відповідають їхнім уподобанням та інтересам. Таким чином, системи рекомендацій допомагають підвищити задоволеність користувачів і їхню залученість, сприяючи більш приємному та зручному користуванню платформою.

Зазвичай рекомендації стосуються різноманітних процесів прийняття рішень, таких як вибір продукту для покупки, музики для прослуховування або новин для читання онлайн. Рекомендаційні системи знаходять застосування у багатьох сферах, причому загальновизнаними прикладами є генератори списків відтворення для відео- та музичних сервісів, рекомендації товарів для інтернет-магазинів або контенту для платформ соціальних мереж, а також рекомендації веб-контенту. Ці системи можуть працювати з одним типом вхідних даних, наприклад, музикою, або обробляти кілька видів даних як усередині однієї платформи, так і між різними платформами, як-от новини, книги та пошукові запити. Існують також популярні рекомендаційні системи, що спеціалізуються на певних темах, таких як вибір ресторанів або онлайн-знайомства. Крім того,

рекомендаційні системи були розроблені для таких галузей, як вивчення наукових статей, пошук експертів і співавторів, а також для надання фінансових послуг[2].

З розвитком інтернету речей з'явилося багато компаній, що пропонують величезну кількість послуг і товарів, що підвищило потребу в ефективних системах рекомендацій для залучення користувачів. До прикладу:

– YouTube: Щохвилини користувачі завантажують 500 годин відео, тому для перегляду всіх відео, завантажених за одну годину, користувачеві знадобилося б 82 роки;

– Spotify: Користувачам доступні понад 80 мільйонів пісень і подкастів;

– Amazon: Платформа пропонує більше 350 мільйонів різних товарів для покупки.

Усі ці платформи застосовують потужні моделі машинного навчання для створення релевантних рекомендацій, що підвищують інтерес користувачів до їхніх послуг і товарів [3].

Існуючі рекомендаційні системи поділяються на два основні типи: системи на основі контентного фільтрування та системи на основі колоборативної фільтрації [4].

Так рекомендаційні системи на основі контентного фільтрування використовують аналіз контенту для того, щоб рекомендувати користувачам нові елементи, які, ймовірно, їм сподобаються. Контентне фільтрування працює шляхом виявлення спільних характеристик між елементами, які користувач вже оцінив, та новими елементами, з якими він ще не знайомий [6].

Перевагами таких систем є [7]:

– прозорість: Користувачі бачать, чому їм рекомендуються певні елементи;

– ефективність: Ці системи можуть бути дуже ефективними для рекомендації нових елементів, які схожі на ті, які користувачеві вже подобаються;

– масштабованість: Ці системи легко масштабувати для роботи з великими каталогами елементів.

Проте ці системи мають свої недоліки. Вони не можуть працювати в режимі "холодного старту", тобто потребують певного часу для збирання інформації про користувача, щоб почати давати точні рекомендації. Також такі системи схильні створювати так звану "інформаційну бульбашку". Якщо користувач довго цікавиться певною темою, система починає рекомендувати лише контент з цієї сфери, ігноруючи інші, пов'язані теми. [8].

Іншим типом рекомендаційних систем є системи на основі колоборативної фільтрації. Ці системи генерують рекомендації, ґрунтуючись на поведінці та вподобаннях інших користувачів. Вони припускають, що користувачі з подібними смаками в минулому, ймовірно, матимуть схожі смаки й у майбутньому [9].

Переваги цих систем включають [10]:

– відсутність потреби в явній інформації про вподобання користувачів: Вони можуть генерувати рекомендації навіть без наявності відгуків від користувачів;

– подібні системи можуть створювати дуже персоналізовані рекомендації, ґрунтуючись на поведінкових даних користувачів;

– здатність ефективно працювати з дуже великими наборами даних про користувачів і продукти.

Однак ці системи мають недоліки, подібні до попереднього типу систем. Без бази відгуків користувачів їм потрібен час для збирання достатньої кількості інформації для надання точних рекомендацій. Крім того, такі системи можуть бути схильні до упередженості, часто рекомендують популярні продукти, які можуть не відповідати індивідуальним вподобанням конкретного користувача [11].

Також існують змішані системи Змішані рекомендаційні системи [12], які поєднують кілька методів для генерації рекомендацій. Ці системи використовують переваги різних методів, щоб згенерувати більш точні й різноманітні рекомендації, ніж будь-який метод, використовуваний окремо.

Підсумовуючи усе вищесказане, можна зробити висновок, що використання рекомендаційного методу на базі контентного фільтрування

дозволяє значно підвищити наступні важливі як для користувачів так і для бізнесу параметри:

– персоналізацію рекомендацій, пропонуючи користувачам контент, який відповідає їхнім інтересам та потребам;

– ефективність пошуку інформації, зменшуючи час та зусилля, які користувачі витрачають на знаходження цікавих для них матеріалів;

– задоволеність користувачів, надаючи їм доступ до релевантного та корисного контенту;

– лояльність користувачів до платформи, що використовує рекомендаційний метод;

– можливості для монетизації, рекомендуючи користувачам продукти або послуги, які з більшою ймовірністю їм зацікавлять.

1.2 Огляд рекомендаційних системи на основі методу контентного фільтрування

Розглянемо рекомендаційні системи на основі контентного фільтрування, зокрема їх принцип роботи, навчання та оцінку моделей. Ці системи засновані на моделях машинного навчання, які прогнозують оцінки, що їх користувач виставить елементам. Наприклад, якщо користувач регулярно переглядає фільми певного жанру або читає статті на певні теми, система аналізує цей контент і знаходить схожі елементи для рекомендації. Це дозволяє системі точно визначати інтереси користувача і пропонувати йому релевантні варіанти.

На основі цих прогнозів система рекомендує користувачеві елементи з найвищим передбачуваним рейтингом. Важливою складовою є навчання моделей, яке відбувається шляхом аналізу великої кількості даних про вподобання користувачів. Оцінка моделей здійснюється за допомогою метрик, які визначають точність передбачень і загальну ефективність системи. Таким чином, контентне фільтрування забезпечує персоналізований підхід до рекомендацій, покращуючи

користувацький досвід і підвищуючи задоволеність від використання платформи. [13].

Для навчання та оцінки моделей необхідні дані про відгуки користувачів, які бувають двох типів: явні та неявні. Збір та аналіз цих даних дозволяє системі створювати точніші прогнози та рекомендації, забезпечуючи користувачам персоналізований досвід. Це критично важливо для розуміння вподобань та потреб користувачів, адже саме на основі цих даних формується ядро рекомендаційної системи, що постійно вдосконалюється та адаптується під зміни у поведінці користувачів.

Явний зворотний зв'язок — це оцінка, яку користувач чітко й безпосередньо дає, щоб висловити свою думку про продукт. Наприклад, це може бути оцінка за шкалою від 1 до 5 після покупки товару або лайк/дизлайк після перегляду відео. Цей тип зворотного зв'язку дає детальну інформацію про те, наскільки користувачу сподобався товар, але його важко збирати, оскільки більшість людей не пишуть відгуки та не оцінюють кожну свою покупку. Незважаючи на це, явний зворотний зв'язок є надзвичайно цінним, оскільки він надає конкретну і пряму інформацію, яка допомагає точно оцінити вподобання користувачів та вдосконалити модель.

Неявний зворотний зв'язок ґрунтується на припущенні, що взаємодія користувача з елементом свідчить про його вподобання. Прикладами такого зворотного зв'язку є історія покупок або переглядів веб-сторінок, список відтворених пісень тощо. Цей тип даних набагато багатший, але менш детальний та більш "зашумлений". Наприклад, користувач може купити товар, щоб подарувати його комусь іншому. Проте завдяки величезному обсягу доступних даних більшість сучасних систем рекомендацій спираються саме на неявний зворотний зв'язок. Використовуючи складні алгоритми для аналізу цих даних, системи можуть виявляти закономірності та тенденції, що дозволяє їм робити точніші і більш релевантні рекомендації для користувачів [14].

Після збору явних або неявних відгуків формується матриця оцінок користувачів. Ця матриця є ключовим елементом у побудові рекомендаційних

систем, адже вона надає структуру для аналізу даних. У випадку явного зворотного зв'язку кожен запис у матриці є числовим значенням, що відображає оцінку, яку користувач поставив певному елементу, наприклад, кількість зірок, які користувачі поставили фільму. Ці числові значення надають чіткі орієнтири для алгоритмів машинного навчання, які можуть використовувати ці дані для побудови моделей, що точно відображають вподобання користувачів.

Для неявного зворотного зв'язку існують логічні значення, що свідчать про наявність або відсутність взаємодії. Наприклад, чи дивився користувач певний фільм, чи купував товар, чи слухав певну пісню. Ці значення, хоча і не настільки детальні, як числові оцінки, надають багатий обсяг даних, що дозволяє алгоритмам машинного навчання виявляти схеми та закономірності у поведінці користувачів. Важливо зазначити, що матриця оцінок є дуже розрідженою, адже користувачі зазвичай взаємодіють лише з невеликою кількістю елементів з усього доступного контенту. Це створює виклики для рекомендаційних систем, які повинні ефективно працювати з неповними даними і знаходити способи заповнення пропусків для точного прогнозування вподобань користувачів.

Розрідженість матриці оцінок є однією з головних проблем у розробці ефективних рекомендаційних систем. Алгоритми повинні бути здатними працювати з великою кількістю нульових значень, що свідчать про відсутність взаємодії, і все ж таки знаходити значущі зв'язки між користувачами та елементами. Для цього використовуються різноманітні методи, такі як факторизація матриць, колаборативна фільтрація та глибоке навчання. Ці методи дозволяють зменшити розмірність матриці та виділити найважливіші характеристики, які визначають вподобання користувачів.

Додатково, врахування контекстуальних факторів, таких як час, місце або ситуаційні обставини, може значно підвищити точність рекомендацій. Наприклад, користувачі можуть мати різні вподобання у різні періоди року або у різних обставинах. Включення таких змінних у моделі дозволяє рекомендаційному методу бути більш гнучкими та адаптивними до змін у поведінці користувачів. У підсумку, ефективне використання матриці оцінок та врахування додаткових

факторів дозволяє створювати потужні та персоналізовані рекомендаційні системи [15].

Для розробки рекомендаційного методу на основі контентного фільтрування можна використовувати різні технології, які полегшують та пришвидшують цей процес. Однією з найпопулярніших є TensorFlow [16], платформа з відкритим кодом для машинного навчання, розроблена Google. Вона пропонує широкий спектр інструментів для створення, тренування та розгортання моделей машинного навчання, включаючи глибокі нейронні мережі [17]. Використання TensorFlow дозволяє створювати складні моделі, які можуть ефективно обробляти великі обсяги даних та робити точні прогнози. Це особливо важливо для рекомендаційних систем, де якість рекомендацій прямо залежить від точності моделей.

Перевагами цієї технології є велика гнучкість та масштабованість, широкий спектр бібліотек та інструментів, а також велика спільнота, яка постійно розвиває та покращує цю технологію. Зокрема, бібліотека TensorFlow Recommenders пропонує багатий набір функцій і можливостей, що робить її придатною для широкого спектру рекомендаційних завдань. Вона забезпечує інструменти для побудови, тренування та оцінки моделей рекомендаційних систем, що дозволяє швидко впроваджувати нові методи і технології в робочий процес. Завдяки цьому, розробники можуть зосередитися на вдосконаленні моделей та підвищенні якості рекомендацій, не витрачаючи зайвий час на налаштування інфраструктури [18].

Цю бібліотеку використовують для створення рекомендаційних систем у таких великих компаніях, як Netflix, Spotify та YouTube [19]. Наприклад, Netflix використовує TensorFlow для персоналізації рекомендацій фільмів та серіалів, забезпечуючи кожному користувачу унікальний список рекомендацій, заснований на його вподобаннях та історії переглядів. Spotify застосовує TensorFlow для створення персоніфікованих плейлистів, аналізуючи музичні вподобання користувачів та пропонуючи нові треки, які можуть їм сподобатися. YouTube використовує TensorFlow для рекомендації відео користувачам, що дозволяє

платформі пропонувати релевантний контент, який відповідає інтересам кожного глядача, і таким чином, утримувати їх на платформі довше.

Крім того, завдяки масштабованості TensorFlow, ці компанії можуть обробляти величезні обсяги даних у режимі реального часу, забезпечуючи швидкі та точні рекомендації для мільйонів користувачів. Це робить TensorFlow одним з найпотужніших інструментів для створення сучасних рекомендаційних систем, що здатні відповідати на виклики високих навантажень і складних алгоритмів.

Підсумовуючи усе вище сказане, можна зробити висновок, що рекомендаційний метод на основі контентного фільтрування є потужним інструментом для персоналізації досвіду користувачів. Такий метод може допомогти користувачам знаходити цікаві елементи, економити час та отримувати кращий загальний досвід. Завдяки цьому користувачі можуть більш ефективно знаходити контент, який відповідає їхнім інтересам, що підвищує їх задоволення від використання платформи та сприяє збільшенню залученості.

Використання технологій, таких як TensorFlow та TensorFlow Recommenders, може пришвидшити розробку подібних систем та значно покращити фінальний результат, що напряму відобразиться на якості продукту, для якого розробляється даний метод. TensorFlow забезпечує необхідну гнучкість та масштабованість, що дозволяє створювати високоефективні моделі машинного навчання, які можуть обробляти великі обсяги даних та робити точні прогнози. Це, у свою чергу, дозволяє створювати рекомендаційні системи, які можуть адаптуватися до змін у поведінці користувачів та забезпечувати релевантні рекомендації в реальному часі.

Крім того, завдяки широкому спектру інструментів та бібліотек, що пропонуються TensorFlow, розробники можуть швидко впроваджувати нові методи і технології у свої моделі. Це забезпечує безперервне вдосконалення рекомендаційних систем та підвищення їх ефективності. Великі компанії, такі як Netflix, Spotify та YouTube, успішно використовують TensorFlow для створення персоналізованих рекомендацій, що свідчить про потужність та надійність цієї платформи. В результаті, рекомендаційні системи, побудовані на основі

TensorFlow, не лише покращують користувацький досвід, але й сприяють зростанню бізнесу за рахунок підвищення залученості та лояльності користувачів.

Таким чином рекомендаційні системи на основі контентного фільтрування та сучасних технологій, таких як TensorFlow, є ключовим компонентом успішних платформ, що прагнуть надавати користувачам персоналізований та ефективний досвід. Їх використання дозволяє не лише задовольняти потреби користувачів, але й забезпечувати конкурентну перевагу у динамічному світі цифрового контенту.

1.3 Аналіз реалізованих рекомендаційних систем на базі методу контентного фільтрування

Одною з найбільших платформ шо використовує рекомендаційний метод на основі контентного фільтрування є торгова платформа Amazon. Дана платформа використовує контентне фільтрування як один з основних методів рекомендацій продуктів для своїх користувачів (рисунок 1.1). Цей метод ґрунтується на аналізі контенту продуктів, таких як описи, рецензії, технічні характеристики та категорії, щоб знайти подібні продукти, які можуть зацікавити користувача.

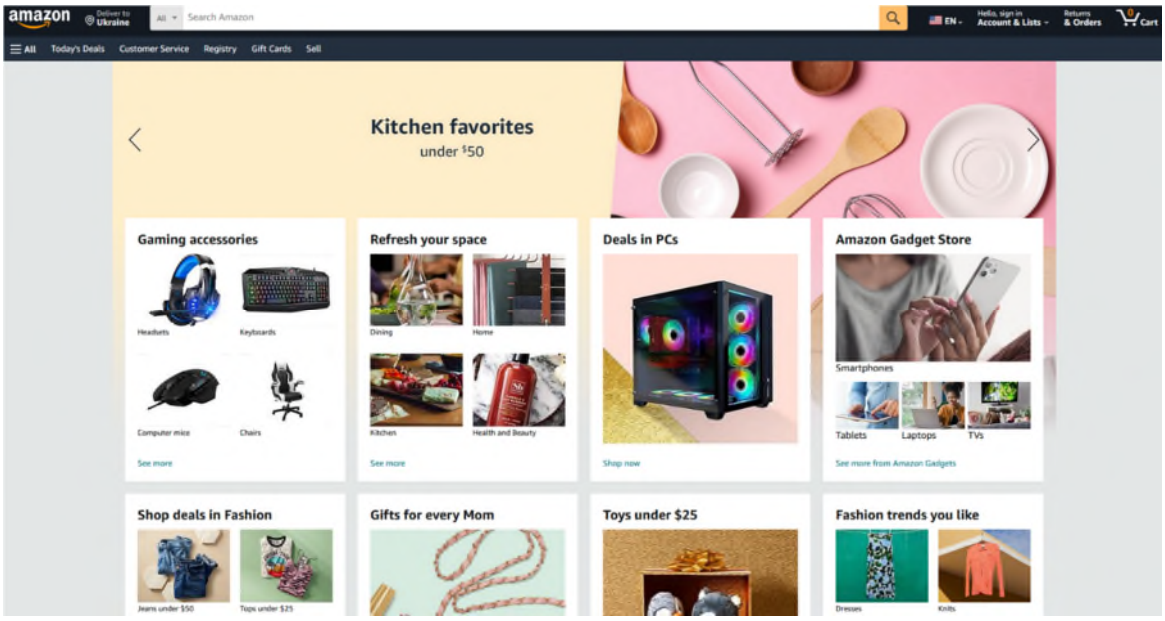


Рисунок 1.1 – Головна сторінка з пропозиціями, відфільтрованими за допомогою контентного фільтрування [20]

Так, Amazon використовує контентне фільтрування на основі таких речей як:

– Amazon аналізує історію покупок користувача, щоб визначити, які типи продуктів його цікавлять. На основі цієї інформації платформа рекомендує подібні продукти, які користувач міг пропустити за звичайного пошуку;

– при перегляді товарів, вебсистема аналізує опис продукту, рецензії та технічні характеристики, щоб рекомендувати подібні продукти;

– платформа аналізує текст відгуку, що залишив користувач, щоб рекомендувати інші продукти, які можуть його зацікавити.

Вигодою використання контентного фільтрування на платформі Amazon, є те, що рекомендації мають високу точність адже ж персоналізованими під кожного користувача. Проте, контентне фільтрування може рідко рекомендувати нові продукти та категорії, оскільки дане фільтрування базується на аналізі існуючих даних. Це може вплинути на те, як користувачі отримують інформацію про нові товари. Крім того, контентне фільтрування може бути упередженим, якщо дані, на яких воно базується, неповні або неточні. Це може призводити до неправильних рекомендацій, які не відповідають інтересам користувача.

Схожий до Amazon метод контентного фільтрування є також на агрегаторі літератури Goodreads (рисунок 1.2). Дана платформа використовує контентне фільтрування як один з основних методів рекомендацій книг для своїх користувачів. Цей метод ґрунтується на аналізі контенту книг, таких як описи, рецензії, жанри та рейтинги, щоб знайти подібні книги, які можуть зацікавити користувача.

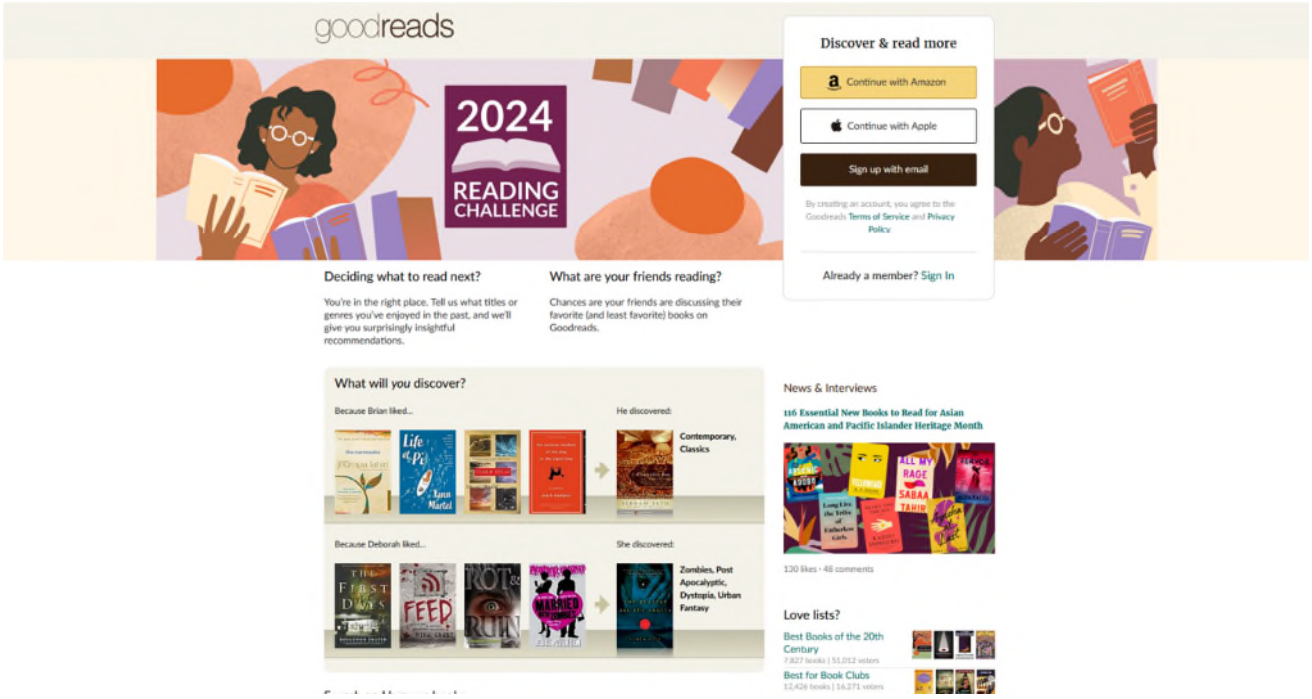


Рисунок 1.2 – Головна сторінка ресурсу Goodreads [21]

Роботу методу фільтрування на даній платформі можна описати наступними кроками:

– першим йде аналіз книг, які вже були оцінені користувачем. Система аналізує дані книги, щоб з'ясувати уподобання користувача. Вона розглядає різні аспекти цих книг, такі як жанр, теми, автори тощо;

– наступним є створення профілю інтересів користувача. На основі проаналізованих на попередньому кроці даних формується профіль інтересів. Цей профіль включає в себе інформацію про те, які типи книг цікавлять користувача, які автори та теми йому подобаються;

– далі система порівнює створений профіль інтересів з характеристиками інших книг на Goodreads. Вона оцінює, наскільки схожі інтереси з іншими користувачами, які прочитали ці книги;

– на основі цього порівняння система надає рекомендації книг. Ці рекомендації можуть бути основані на подібності до книг, які вже подобаються користувачеві, або на інших читацьких звичках користувачів зі схожими інтересами;

– після того, як користувач прочитав книгу, яку йому було рекомендовано, він можете оцінити її і залишити відгук. Це дозволяє системі удосконалювати рекомендації.

Хоча платформи Amazon і Goodreads надають різні типи послуг, обидві вони стикаються з подібними проблемами в роботі рекомендаційних систем, які використовують контентне фільтрування. Цей метод рекомендацій має певний недолік – він погано справляється з новими категоріями та предметами. Причина полягає в тому, що для нових категорій просто не вистачає даних, щоб алгоритм міг точно прогнозувати інтереси користувачів. Отже, навіть при великій кількості прочитаних книг користувачами, рекомендації можуть бути обмеженими і не завжди відображати їхні справжні вподобання.

Крім того, контентне фільтрування часто демонструє упередженість на основі попередніх інтересів користувачів. Якщо людина раніше цікавилася певним типом книг, алгоритм продовжуватиме пропонувати лише схожі варіанти, навіть якщо користувач може бути зацікавлений у чомусь новому. Це призводить до ситуації, коли рекомендаційний метод фактично обмежує можливості для відкриття нових жанрів або авторів, що могли б зацікавити користувача. Така обмеженість рекомендаційного механізму може знизити загальне задоволення від використання платформи.

Важливо також зазначити, що проблеми контентного фільтрування впливають на здатність платформи підтримувати інтерес користувачів до нових та маловідомих книг. Коли рекомендації зосереджуються лише на відомих і популярних предметах, користувачі можуть втратити шанс відкрити для себе нові та цікаві твори. Це стає особливо проблематичним для авторів і видавців, які прагнуть привернути увагу до своїх нових робіт, але не можуть конкурувати з усталеними популярними категоріями. В результаті, для покращення ефективності рекомендаційних систем важливо враховувати не лише історію користувачів, але й намагатися пропонувати різноманітні та нові варіанти, щоб розширити їхній читацький досвід.

1.4 Мета та завдання кваліфікаційної роботи

Метою даної роботи є спрощення навігації користувачів у вебсистемі мобільних пристроїв за допомогою методу формування рекомендацій на основі контентного фільтрування.

Для досягнення поставленої мети необхідно реалізувати виконання наступних задач:

– провести аналіз систем рекомендацій на основі методу контентного фільтрування;

– розробити метод формування рекомендацій мобільних пристроїв на основі методу контентного фільтрування;

– розробити функціональну та інформаційну структуру вебсистеми мобільних пристроїв;

– розробити програмну реалізацію вебсистеми мобільних пристроїв з використанням методу для формування рекомендацій на основі методу контентного фільтрування.

– провести тестування розробленого методу формування рекомендацій мобільних пристроїв

Розділ 2 Рекомендаційний метод на основі контентного фільтрування

2.1 Загальна схема роботи рекомендаційного методу на основі контентного фільтрування

Враховуючи особливості роботи рекомендаційного методу на основі контентного фільтрування, детально опишемо схему роботи. Так, даний метод складається з наступних кроків: Збір даних, Побудова профілю користувача, Аналіз відповідності контенту, Ранжування рекомендацій, Фільтрація рекомендацій, Оновлення профілю користувача, Представлення рекомендацій. Представимо роботу даного методу у схематичному вигляді (рис. 2.1)

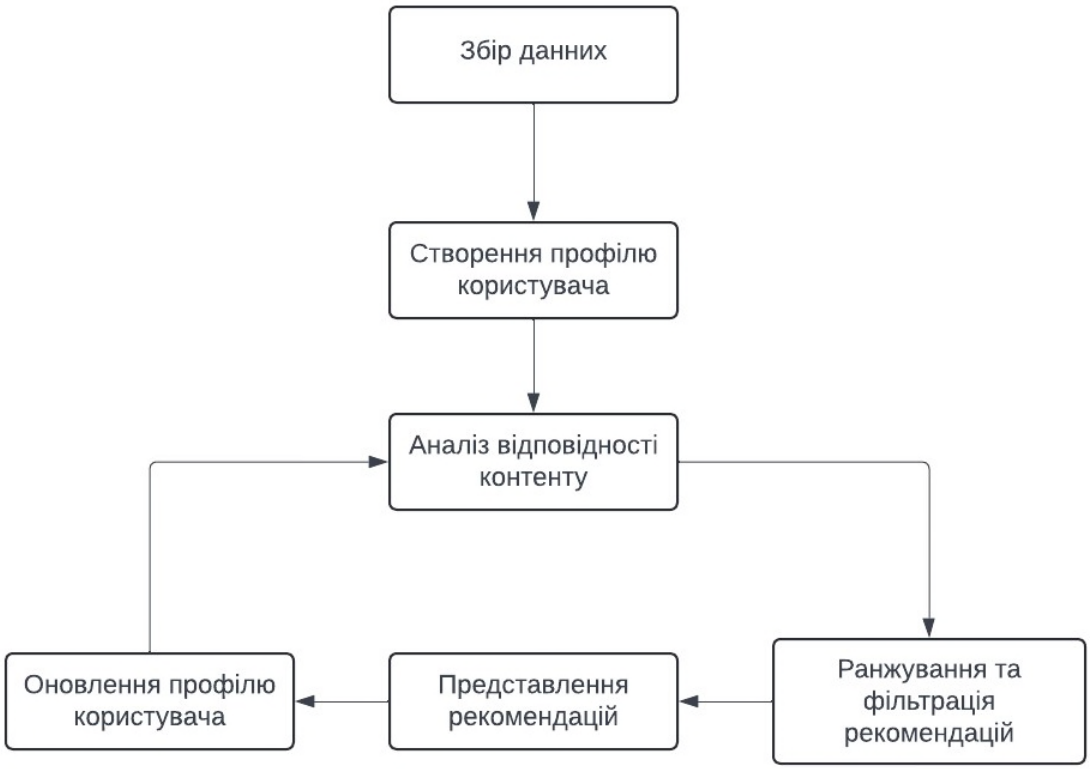


Рисунок 2.1 – Загальна схема роботи рекомендаційного методу на основі контентного фільтрування.

Деталізуємо кожен крок даного алгоритму:

Першим кроком являється збір відомостей про вміст який будуть рекомендуватися користувачам. У контексті даної роботи збираються данні про мобільні пристрої та їх похідні. Ці дані включають в себе такі характеристики, як

бренд, модель, дата виходу, наявність аудіо виходу, ємкість батареї, наявність та версію Bluetooth, модель процесору, ціна, тощо.

Далі здійснюється побудова профілю користувача. Кожному користувачеві призначається профіль, який відображає його вподобання на основі вмісту, який він споживав раніше.

Наступним є аналіз відповідності контенту: на цьому етапі для кожного об'єкта вмісту розраховується ступінь відповідності до профілю користувача. Для цього можуть бути використані такі методи, як векторне представлення, алгоритми кластеризації або нейронні мережі. Основна ціль даного етапу полягає в тому, щоб визначити, наскільки об'єкт відповідає вподобанням користувача на основі характеристик об'єкта і профілю користувача.

Після обчислення ступеня відповідності для кожного об'єкта, вмістові об'єкти ранжуються в порядку спадання відповідності. Чим вище ступінь відповідності, тим вище позиція об'єкта в списку рекомендацій.

Також є можливою фільтрація рекомендацій для покращення релевантності. Так, можуть бути відфільтровані об'єкти, які користувач вже споживав, або ті, які мають низький рейтинг серед інших користувачів.

Далі користувачеві виводиться список рекомендованих об'єктів, який може бути представлений у вигляді списку, каруселі, чи будь-якого іншого інтерфейсу.

Після того, як користувач переглянув нові позиції або взаємодіяв з рекомендованим вмістом, його профіль може бути оновлений, базуючись на результатах його дій. Так поставивши низьку оцінку предмету, шо мав високе співпадіння з його профілем, оцінка співпадіння даного предмет та йому подібних знизиться.

2.2 Етапи роботи алгоритму контентного фільтрування

Контентне фільтрування є методом рекомендацій, який використовує атрибути як предметів, так і користувачів для визначення того, які предмети

ймовірно сподобаються користувачеві. Ініціалізація даного алгоритму проходить наступним чином:

Збір даних. Для початку роботи алгоритм потребує підготовлених даних, які можна отримати використавши готові датасети чи поєднавши готові невеликі набори даних. На цьому кроці дані приводяться до нормального стану, а саме відфільтровуються нерелевантні та дубльовані записи, нормуються значення, перетворення категоріальні значення на числові. Також на даному етапі набір даних розділяється на тренувальний та валідаційний набори, для уникнення перенавчання рекомендаційної моделі.

Створення векторного представлення даних. Таким чином утворюється n-мірний масив даних, де записи утворюють n-мірний кластер. До прикладу провівши дані дії над датасетом елементів, що мають лиш три характеристики, ми отримаємо трьох мірний масив, у візуалізації якого можна буде побачити розташовані поруч записи, сукупність яких буде утворювати кластер. Приклад подібної кластеризації наведено на рисунку 2.2:

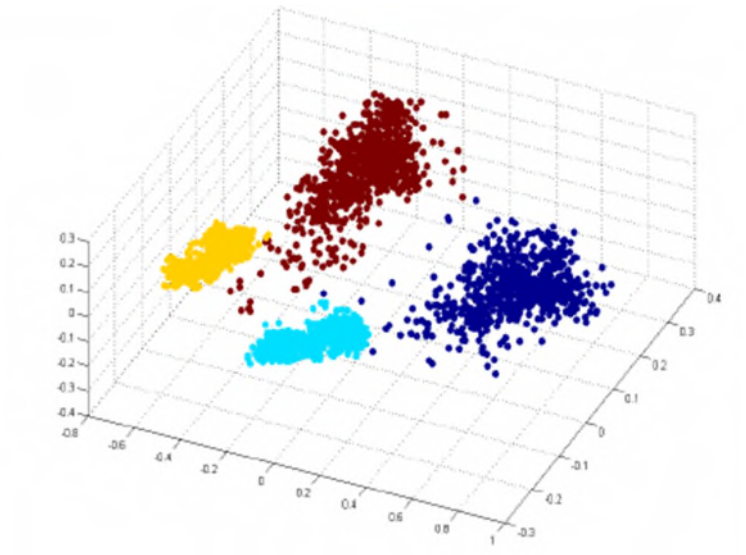


Рисунок 2.2 – Приклад кластеризованих даних у 3 вимірному просторі [27]

Створення метрики схожості. Після отримання векторизованих даних, виконується оцінка схожості між записами. Оцінку схожості між елементами можна виконати за допомогою наступних методів

– Cosine similarity: Вимірює кутовий збіг між двома векторами.

– Jaccard similarity: Вимірює частку спільних елементів між двома множинами.

– Pearson correlation: Вимірює лінійну залежність між двома векторами.

– Minkowski distance: Вимірює відстань між двома векторами в n-вимірному просторі.

Таким чином отримується оцінка схожості для кожної пари записів, що буде використовуватися в наступних кроках

Останнім кроком роботи методу є процес створення рекомендацій. Він ґрунтується на раніше обчислених векторних представленнях як користувачів, так і предметів, а також на метриках схожості для виявлення предметів, які ймовірно сподобаються користувачеві. Це можна зробити за допомогою таких методів, як:

– Найближчі сусіди (NN);

– Співвідношення схожості;

– Факторизація матриці.

Також додатковим кроком буде персоналізація та ранжування отриманих рекомендацій, для того, щоб показувати користувачеві спочатку найбільш відповідні результати. На даному етапі використовується історія взаємодій користувача, контекст та отримані демографічні дані.

Варто зазначити, що при додаванні нових даних в початковий датасет, весь процес ініціалізації потрібно повторити спочатку.

2.3 Функціональна структура інформаційної системи

Виходячи з особливості функціонування , вебсистема має дві фази функціонування. Так перша фаза починається зі старту користування окремим користувачем системою та триває короткий час, під час якого збирає данні про користувача. Під час цієї фази система тільки збирає та опрацьовує данні, які необхідні для рекомендаційного методу щоб розпочати створення рекомендацій. Як тільки було зібрано достатньо інформації про користувача та його дії, система, що імплементує рекомендаційний метод починає надавати рекомендації, що

відображаються на кінцевому інтерфейсі, з яким взаємодіє користувач. Наведемо загальну схему функціонування системи від початку її використання користувачем (рисунок 2.3)

Рисунок 2.3 – Загальна схема функціонування системи

Базуючись на поданій схемі функціонування та вимогах до інформаційної системи, описаних в пункті 1.5, можемо сформувати перелік дій користувача, що мають бути оброблені системою. Так до цих дій входять наступні:

- перегляд списку продукції з визначеними фільтрами;
- перегляд списку продукції у межах вибраної категорії;
- перегляд окремого продукту;
- оцінка продукту;
- написання рецензії на продукт;
- добавлення продукту в кошик\список бажаного;

– замовлення продукту.

Кожна з наведених дій має бути оброблена системою таким чином, щоб вплинути на формування або оновлення профілю користувача, що їх виконав. Наведемо узагальнену схему взаємодії користувача з системою через призму вищеописаних дій (рисунок 2.4):

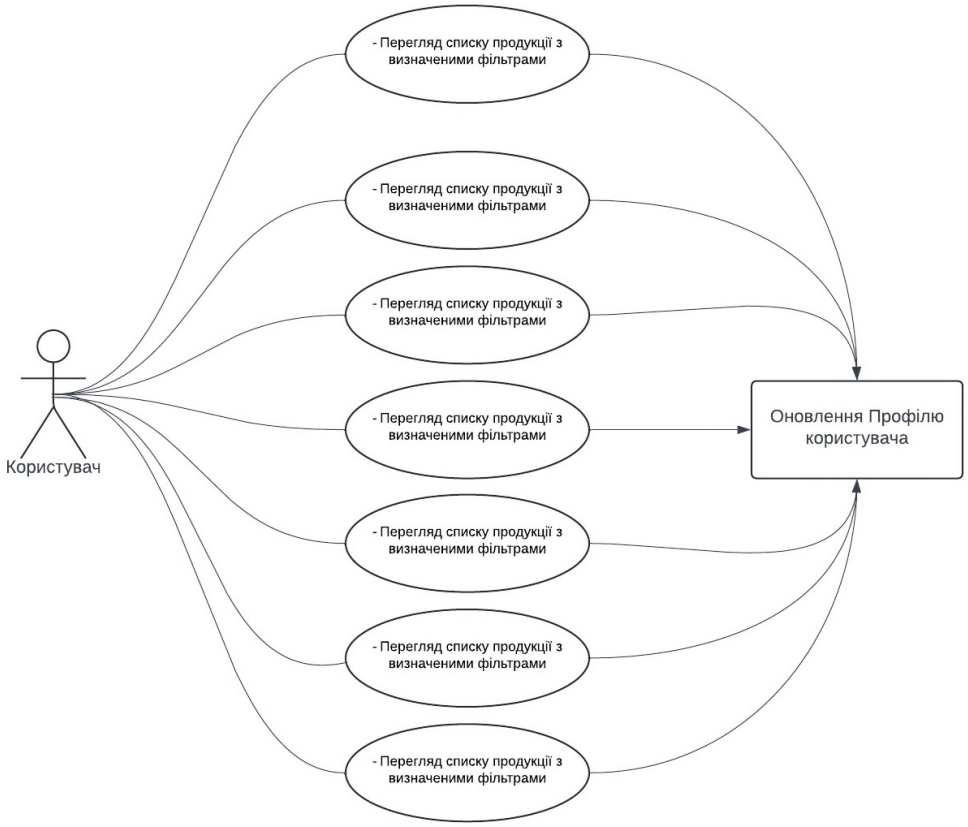


Рисунок 2.4 Діаграма взаємодії користувача з системою

Опишемо кожен прецедент, а саме наслідки, які виникають:

Прецедент №1.

Діюча особа: Користувач.

Дія: Перегляд списку продукції з визначеними фільтрами.

Результат:

– Якщо профіль користувача сформований, здійснюється його оновлення, при якому виставлені фільтри користувача незначно збільшують вагу.

– Якщо профіль ще не створений, здійснюється створення профілю та вибрані фільтри записуються до нього з базовою вагою.

Прецедент №2.

Діюча особа: Користувач.

Дія: Перегляд списку продукції у межах вибраної категорії.

Результат:

– Якщо профіль користувача сформований, здійснюється його оновлення, при якому вибрана категорія незначно збільшують вагу.

– Якщо профіль ще не створений, здійснюється створення профілю та вибрана категорія записується до нього з базовою вагою.

Прецедент №3.

Діюча особа: Користувач.

Дія: Перегляд окремого продукту.

Результат:

– Профіль користувача оновлюється, так, шо категорії та параметри, до яких належав переглянутий продукт незначно збільшують свою вагу.

Прецедент №4.

Діюча особа: Користувач.

– Дія: Оцінка продукту.

Результат:

– Профіль користувача оновлюється, так, шо категорії та параметри, до яких належав оцінений продукт збільшують свою вагу в залежності від того, яка оцінка була виставлена. Так 5 зірок значно збільшують вагу параметрів, коли 1 зірка значно зменшує вагу параметрів.

Прецедент №5.

Діюча особа: Користувач.

– Дія: Написання рецензії на продукт.

Результат:

– Профіль користувача оновлюється, так, шо категорії та параметри, до яких належав оцінений продукт незначно збільшують свою вагу.

Прецедент №6.

Діюча особа: Користувач.

Дія: Добавлення продукту в кошик\список бажаного.

Результат:

– Профіль користувача оновлюється, так, шо категорії та параметри, до яких належав оцінений продукт збільшують свою вагу.

Прецедент №7.

Діюча особа: Користувач.

Дія: Замовлення продукту.

Результат:

– Профіль користувача оновлюється, так, шо категорії та параметри, до яких належав оцінений продукт значно збільшують свою вагу.

2.4 Проектна архітектура системи та взаємозв’язок компонентів

Виходячи з описаних дій, що мають бути оброблені системою та вимогах до інформаційної системи, описаних в пункті 1.4, можемо узагальнити архітектуру інформаційної системи, описати компоненти та їх взаємодію. Таким чином, інформаційна система складаються з трьох модулей, що взаємопов’язані між собою: модуль рекомендаційного методу, бекенд модуль та фронтенд модуль.

Модуль рекомендаційного методу відповідає за аналіз даних, створення та оновлення профілей користувачів та створення персоналізованих рекомендацій. Даний модуль приймає дані про дії користувача, перелік продукції та після обробки отриманих даних може надати рекомендації для конкретного користувача.

Бекенд модуль відповідає за преведення отриманих даних від фронтенд модулю до вигляду, що може спожити модуль рекомендаційного методу та збереження цих даних у базі даних. Узагальнюючи, бекенд модуль виступає посередником між фронтенд модулем та модулем рекомендаційного методу.

Фронтенд модуль відповідає за збір даних та надання зрозумілого інтерфейсу для користувача.

Підсумовуючи все вищесказане, наведемо узагальнюючу схему функціонування модулей системи (рисунок 2.5)

Рисунок 2.5 – Узагальнюючу схему функціонування модулей системи

2.5 Інформаційна структура системи

Базуючись описаній функціональності системи та вимогах до інформаційної системи, можемо розробимо структуру бази даних, що відповідає цілям дослідження.

Для створення бази даних вебсистеми для мобільних пристроїв, потрібно спочатку визначити основні сутності, які існують у цій області. Кожна з цих сутностей має свій набір властивостей, які описують їх характеристики. Також важливо встановити зв'язки між цими сутностями та визначити їх ступінь взаємозв'язку. Так на рисунку 2.3 подано ER-діаграму, що представляє усі вищеописані елементи.

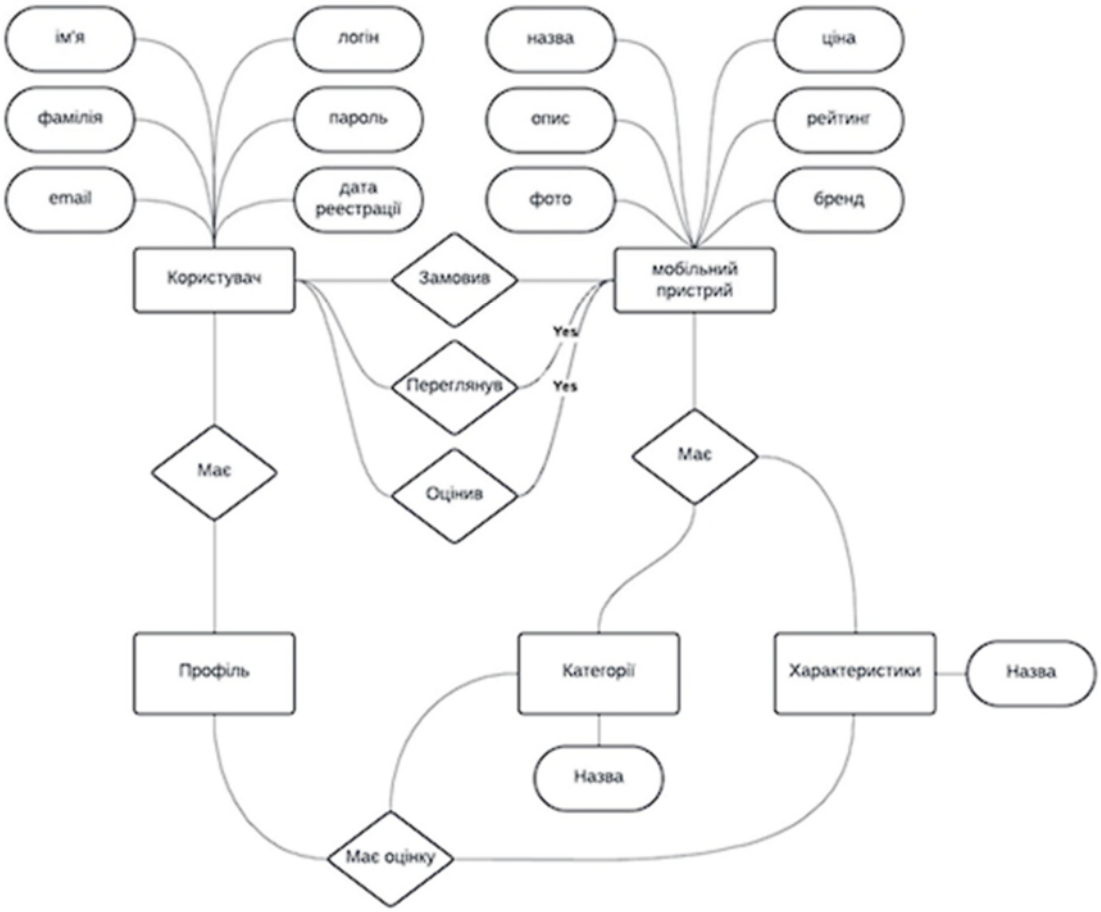


Рисунок 2.6 – ER-діаграма бази даних вебсистеми мобільних пристроїв

З ER-діаграми була виведена даталогічна модель бази даних для мобільної вебсистеми (рис. 2.7). Ця модель описує всі сутності з ER-діаграми у вигляді таблиць, встановлюючи зв'язки та їх потужність. Додано також допоміжні таблиці, необхідні для коректної роботи системи. База даних містить наступні таблиці:

1. Користувачі - зберігає дані про користувачів системи.
2. Мобільні пристрої - зберігає дані про мобільні пристрої, які використовуються в системі.
3. Категорії - зберігає дані про категорії контенту,
4. Характеристики - зберігає дані про характеристики контенту,
5. Групи - зберігає дані про групи користувачів,
6. Типи вмісту - зберігає дані про типи контенту, доступного в системі,
7. Дозволи - зберігає дані про дозволи користувачів на доступ до контенту,

8. Admin_log - зберігає журнал дій адміністраторів системи,

Службові таблиці використовуються для реалізації зв'язків типу "багато-до-багатьох". Кожна таблиця призначена для зберігання інформації про відповідну сутність.

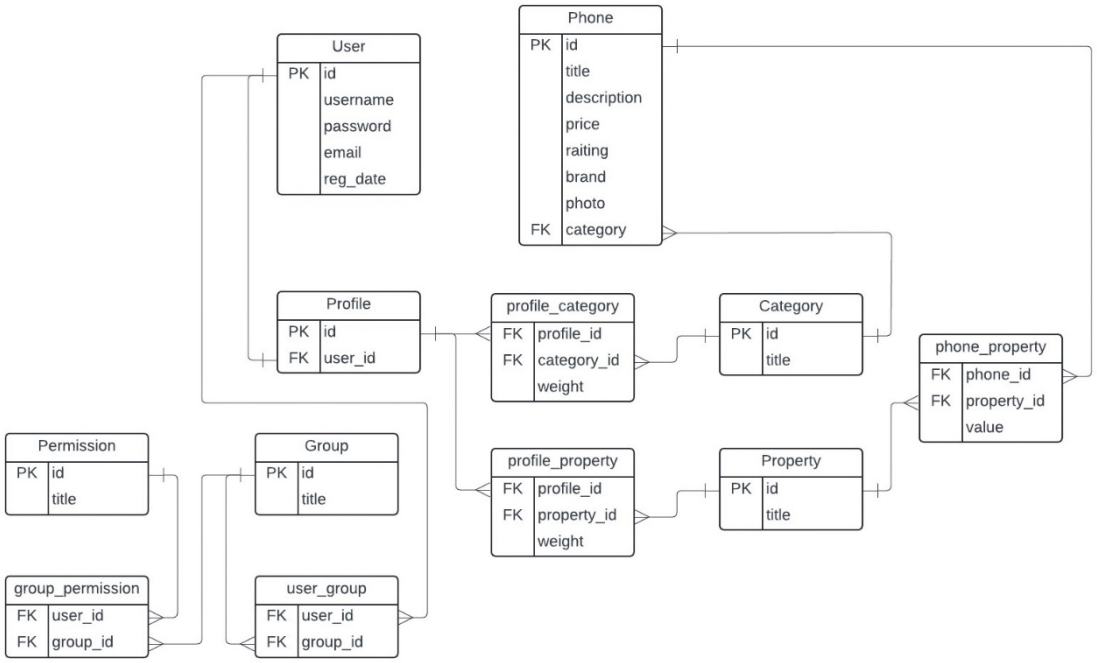


Рисунок 2.7 – Даталогічна модель бази даних вебсистеми мобільних пристроїв

Детально опишемо основні таблиці, подані на даталогічній моделі.

Таблиця «User» призначена для збереження даних про користувачів. Атрибутами таблиці є: id, username, email, password, reg_date. Детальний опис вказаних атрибутів наведено в таблиці 2.1

Таблиця 2.1 – Атрибути таблиці «Users»

№	Назва	Тип даних	Опис
1	id	bigint(serial)	Первинний ключ. Порядковий номер користувача.
2	username	varchar(150)	Логін користувача.
3	email	varchar(254)	Email користувача
4	password	varchar(128)	Пароль користувача. Зашифрований за допомогою

			алгоритму PBKDF2 з хешем SHA256.
5	Reg_date	timestamp	Часова марка, що показує дату та час регістрації користувача.

Таблиця «Group» призначена для збереження даних про групи користувачів. Атрибутами таблиці є: id, title. Детальний опис вказаних атрибутів наведено в таблиці 2.2

Таблиця 2.2 – Атрибути таблиці «Group»

№	Назва	Тип даних	Опис
1	id	bigint(serial)	Первинний ключ. Порядковий номер групи користувачів.
2	title	varchar(150)	Назва групи

Таблиця «Permission» призначена для збереження даних про рівні дозволів, такі як адмін, менеджер, користувач, анонім. Атрибутами таблиці є: id, title. Детальний опис вказаних атрибутів наведено в таблиці 2.3

Таблиця 2.3 – Атрибути таблиці «Permission»

№	Назва	Тип даних	Опис
1	id	bigint(serial)	Первинний ключ. Порядковий номер рівня дозволу.
2	title	varchar(150)	Назва рівня доступу

Таблиця «Phone» призначена для збереження даних про мобільні пристрої. Атрибутами таблиці є: id, title, description, price, raiting, brand, photo, categoty_id. Детальний опис вказаних атрибутів наведено в таблиці 2.4

Таблиця 2.4 – Атрибути таблиці «Phone»

№	Назва	Тип даних	Опис

1	id	bigint(serial)	Первинний ключ. Порядковий номер запису.
2	title	varchar(100)	Назва мобільного пристрою
4	description	text	Опис продукту
5	photo	varchar(512)	URL посилання на головне фото продукту
6	price	integer	Ціна мобільного пристрою. Враховується модулем рекомендації
7	raiting	Integer	Рейтинг мобільного пристрою. Визначається користувачами. Враховується модулем рекомендації
8	brand	Varchar(100)	Бренд мобільного пристрою. Враховується модулем рекомендації
9	category_id	bigint	Вторинний ключ, що вказує на запис в таблиці «Category», що відповідає категорії даного мобільного пристрою

Таблиця «Category» призначена для збереження даних про категорії мобільних пристроїв, такі як адмін, менеджер, користувач, анонім. Атрибутами таблиці є: id, title. Детальний опис вказаних атрибутів наведено в таблиці 2.5

Таблиця 2.5 – Атрибути таблиці «Category»

№	**Назва**	**Тип даних**	**Опис**
1	id	bigint(serial)	Первинний ключ. Порядковий номер категорії.
2	title	varchar(150)	Назва категорії.

Таблиця «Property» призначена для збереження даних про характеристики мобільних пристроїв, що використовуються в аналізі та створенні рекомендацій. Атрибутами таблиці є: id, title. Детальний опис вказаних атрибутів наведено в таблиці 2.6

Таблиця 2.6 – Атрибути таблиці «Property»

№	Назва	Тип даних	Опис
1	id	bigint(serial)	Первинний ключ. Порядковий номер характеристики.
2	title	varchar(150)	Назва характеристики.

Також варто описати призначення деяких допоміжних таблиць. Так таблиця group_permission відповідає за відображення взаємозв’язків між групами користувачів та рівнями дозволів. Таблиця user_group відображає належність певного користувача до групи або кількох груп. Таблиця phone_property відображає наявність певної характеристики у певного мобільного пристрою та яке значення цієї характеристики має мобільний пристрій. Таблиця profile_category відображає те, з якою категорією взаємодіяв користувач та яку вагу певна категорія має при створенні рекомендацій. Таблиця profile_property є схожою до таблиці profile_category. Ця таблиця відображає те, які характеристики мобільних пристроїв подобаються чи не подобаються певному користувачеві.

2.6 Підготовка робочих вхідних даних для системи

Враховуючи особливості роботи рекомендаційного модулю, необхідно підготувати датасет, за допомогою якого буде проведено навчання модулю. Враховуючи особливості предметної області, було вибрано датасет Dataset_Cell_Phones_Model_Brand, який розміщений на ресурсі back4app. Даний датасет має понад 8 тисяч мобільних пристроїв з великою кількістю характеристик. В перелік характеристик мобільних пристроїв входять наступні:

- Brand: Бренд мобільного телефону.
- Model: Модель мобільного телефону.
- Announced: Дата анонсу телефону.
- Audio_jack: Наявність аудіороз'єму (так/ні).
- Battery: Параметри батареї (ємність, тип і т.д.).

- Bluetooth: Версія Bluetooth.
- CPU: Процесор.
- Chipset: Чіпсет.
- Colors: Доступні кольори.
- Dimensions: Розміри телефону.
- Display_resolution: Роздільна здатність екрану.
- Display_size: Розмір екрану.
- Display_type: Тип екрану (наприклад, IPS LCD, AMOLED).
- EDGE: Підтримка EDGE (так/ні).
- FourG: Підтримка 4G (так/ні).
- GPRS: Підтримка GPRS (так/ні).
- GPS: Наявність GPS (так/ні).
- GPU: Графічний процесор.
- Internal_memory: Внутрішня пам'ять.
- Loud_speaker: Наявність динаміка (так/ні).
- Memory_card: Підтримка карти пам'яті (так/ні).
- NFC: Підтримка NFC (так/ні).
- Network: Мережеві стандарти (наприклад, GSM, CDMA).
- Network_Speed: Швидкість передачі даних в мережі.
- Operating_System: Операційна система.
- Primary_camera: Основна камера (мегапікселі).
- RAM: Оперативна пам'ять.
- Radio: Наявність радіоприймача (так/ні).
- SIM: Типи SIM-карт.
- Secondary_camera: Фронтальна камера (мегапікселі).
- Sensors: Доступні сенсори (наприклад, акселерометр, гіроскоп).
- Status: Статус телефону (наприклад, анонсований, випущений).
- ThreeG: Підтримка 3G (так/ні).
- TwoG: Підтримка 2G (так/ні).

– USB: Версія USB.

– WLAN: Підтримка Wi-Fi (так/ні).

Проте для коректної генерації рекомендацій, було обрано лише наступні параметри: Brand, Model, Announced, Battery, Bluetooth, CPU, GPU, Colors, Display_resolution, Display_type, Internal_memory, NFC, Primary_camera, Secondary_camera, RAM.

Ці параметри були обрані для навчання рекомендаційного методу через їхню важливість для користувачів і вплив на прийняття рішення. Бренд і модель є ключовими характеристиками, що визначають довіру і якість, тоді як технічні характеристики, такі як процесор, графічний процесор, оперативна пам'ять і внутрішня пам'ять, визначають продуктивність пристрою. Мультимедійні можливості, зокрема основна і фронтальна камери, важливі для тих, хто активно використовує смартфон для фотографії і відеозйомки. Дисплей, зокрема його роздільна здатність і тип, впливають на якість зображення та загальні враження від використання. Також враховуються характеристики підключення, як-от Bluetooth і NFC, що забезпечують сумісність з іншими пристроями. Батарея визначає тривалість роботи пристрою без підзарядки, а дата анонсу показує, наскільки сучасною є модель. Всі ці параметри разом дозволяють створити точні та персоналізовані рекомендації для користувачів.

Виходячи з особливості алгоритму роботу методу контентного фільтрування, опишемо те, як данні перетворюються та нормуються перед стартом методу контентного фільтрування. Так для навчання рекомендаційного методу на основі контентного фільтрування кожне з обраних полів проходить процес нормалізації, щоб забезпечити коректне порівняння та обробку даних. Перш за все, текстові поля, такі як Brand, Model, і Colors, перетворюються на числові вектори за допомогою техніки кодування embedding, щоб представити їх у зручному для машинного навчання вигляді. Технічні характеристики, такі як CPU, GPU, RAM, Internal_memory, Battery, Primary_camera, Secondary_camera, Display_resolution, і Display_type, нормалізуються за допомогою стандартизації, що дозволяє привести всі числові значення до одного діапазону.

Щодо категоріальних параметрів, таких як Bluetooth і NFC, вони кодуються за допомогою бінарного кодування (0 або 1), що вказує на наявність або відсутність цієї функції. Поле Announced переводиться у числовий формат, що представляє кількість днів з моменту випуску до сьогоднішнього дня, після чого також нормалізується для узгодження з іншими числовими полями. Такий підхід забезпечує рівномірність і порівнянність даних, що дозволяє алгоритму ефективно знаходити подібності між різними смартфонами на основі їхніх характеристик.

Таким чином один з записів з підготовленого датасету буде виглядати наступним чином (в таблиці не врахована векторизація строкових значень, через неможливість наглядного наведення подібних даних):

Таблиця 2.7 – Приклад підготовленого запису

Параметр	Нормалізоване значення
Id	8b5d03c6-90cf-4d02-b69f-c8ae2090c1f5
Brand	OnePlus
Model	_5
Announced	2565
Battery	Non-removable Li-Po 3300 mAh battery
Bluetooth	1
CPU	Octa-core (4x2.45 GHz Kryo & 4x1.9 GHz Kryo)
GPU	Adreno 540
Colors	Midnight Black| Slate Gray
Display_resolution	5.5
Display_type	Optic AMOLED capacitive touchscreen 16M colors
Internal_memory	64
NFC	1

Primary_camera	Dual 16 MP| f/1.7| 24mm| EIS (gyro) + 20 MP| f/2.6| 36mm| phase detection autofocus| 1.6x optical zoom| dual-LED flash|
Secondary_camera	16 MP| f/2.0| 20mm| EIS (gyro)| 1.0 m pixel size| 1080p| Auto HDR
RAM	8

2.7 Висновки до розділу 2

У даному розділі було розглянуто та описано рекомендаційний метод на основі контентного фільтрування, етапи та особливості його функціонування. Описаний метод включає наступні кроки.

1. Збір даних про мобільні пристрої та їх характеристики.
2. Побудова профілю користувача для кожного користувача, який відображає його вподобання на основі раніше споживаного контенту.
3. Для кожного об'єкта контенту розраховується ступінь відповідності профілю користувача.
4. Об'єкти контенту ранжуються в порядку спадання відповідності.
5. Фільтрація рекомендацій для покращення релевантності.
6. Представлення рекомендацій.
7. Після того, як користувач переглянув рекомендації, його профіль може бути оновлений.
8. Алгоритм роботи контентного фільтрування включає наступні кроки:
9. Підготовка даних, нормалізація, поділ на тренувальний та валідаційний набори.
10. Створення векторного представлення даних, а саме перетворення даних у n-мірний масив.
11. Створення метрики схожості.

12.Виявлення предметів, які ймовірно сподобаються користувачеві.

13.Персоналізація та ранжування рекомендацій, вдосконалення отриманих рекомендацій з урахуванням історії взаємодій, контексту та демографічних даних.

Також було описано загальну архітектуру системи, модулі, з яких система складається та їх взаємодію. Так система складається з трьох основних компонентів: модуля рекомендацій, бекенд модуля та фронтенд модуля.

Модуль рекомендацій реалізує метод контентного фільтрування для прогнозування, які мобільні пристрої сподобаються користувачеві. Модель навчається на наборі даних, що містить інформацію про мобільні пристрої.

Бекенд модуль відповідає за обробку запитів від фронтенд модуля, забезпечення даних модулю рекомендацій та н.

Фронтенд модуль забезпечує інтерфейс користувача для системи.

Для навчання рекомендаційного модуля використовується датасет Dataset_Cell_Phones_Model_Brand. З датасету обираються **та нормуються** лише наступні параметри: Brand, Model, Announced, Battery, Bluetooth, CPU, GPU, Colors, Display_resolution, Display_type, Internal_memory, NFC, Primary_camera, Secondary_camera, RAM.

Нормування даних проходить за наступними правилами:

- текстові поля кодуються за допомогою embedding;
- технічні характеристики нормалізуються за допомогою стандартизації;
- категоріальні параметри кодуються за допомогою бінарного кодування;
- поле Announced переводиться у числовий формат і нормалізується.

Розділ 3 Програмна реалізація вебсистеми з використанням рекомендаційного методу на основі контентного фільтрування для мобільних пристроїв

3.1 Визначення засобів створення вебсистеми мобільних пристроїв

Виходячи з описаної архітектури інформаційної системи, для кожного з модулей будуть використані відповідні фреймворки та бібліотеки, що пришвидшать розробку визначеної системи.

Для реалізації рекомендаційного модуля було використано бібліотеку TensorFlow Recommenders (TFRS), що розроблена саме для розробки рекомендаційних систем. TFRS допомагає з усім робочим процесом побудови системи рекомендацій: від підготовки даних до формулювання моделі, навчання, оцінювання та розгортання. Бібліотека має наступні переваги:

– TFRS побудовано на Keras, що забезпечує плавний процес навчання та дозволяє будувати складні моделі;

– підтримка великої кількості алгоритмів та реалізація абсолютної більшості алгоритмів рекомендацій;

– бібліотека дозволяє враховувати інформацію про користувачів, елементи та контекст у моделях рекомендацій.

Основною мовою програмування був обраний JavaScript з використанням надбудови TypeScript через його гнучкість, потужність та широке розповсюдження у веб-розробці. Використання TypeScript додає ряд переваг, які роблять розробку більш ефективною та надійною. Основні такі переваги:

– TypeScript додає статичну типізацію до JavaScript, що дозволяє виявляти багато помилок на етапі компіляції, ще до виконання коду. Це значно підвищує надійність і передбачуваність коду, зменшуючи кількість багів у готовому продукті;

– статична типізація також покращує можливості навігації по коду та його рефакторингу. Інструменти розробки, такі як WebStorm, можуть використовувати

типову інформацію для більш ефективного автозаповнення, пошуку використань змінних і функцій, а також для безпечного перейменування елементів коду;

– TypeScript підтримує всі сучасні можливості JavaScript (ES6/ES7 та новіші), а також додає власні розширення. Це дозволяє розробникам використовувати новітні можливості мови, забезпечуючи при цьому сумісність зі старими браузерами;

– модульність і повторне використання коду: TypeScript спрощує створення модульного коду, що легко підтримується і повторно використовується. Завдяки інтерфейсам та модулюванню коду, розробка великих проектів стає більш структурованою та менш схильною до помилок;

– TypeScript має велику і активну спільноту, а також підтримується компанією Microsoft. Це забезпечує наявність великої кількості ресурсів, бібліотек і інструментів, що спрощують розробку та інтеграцію TypeScript у проекти;

– багато популярних бібліотек і фреймворків (таких як React, Angular, Vue.js) мають відмінну підтримку TypeScript, що полегшує інтеграцію і використання цих технологій у проектах.

Для модулю рекомендаційного методу буде використано бібліотека TensorFlow Recommenders, що призначена саме для розробки рекомендаційних систем. Дана бібліотеки допомагає з усім робочим процесом побудови системи рекомендацій: від підготовки даних до формулювання моделі, навчання, оцінювання та розгортання. Бібліотека має наступні переваги:

– TFRS побудовано на Keras, тож має плавний процес навчання та дозволяє будувати складні моделі;

– бібліотека реалізує абсолютну більшість алгоритмів рекомендацій.

– бібліотека дозволяє враховувати інформацію про користувачів, елементи та контекст у моделях рекомендацій;

– бібліотека спрощує процес оцінення ефективності моделі та дозволяє розгортати її для обслуговування рекомендацій кінцевим користувачам.

Враховуючи особливості функціонування бекенд модулю інформаційної системи, було обрано наступні компоненти для розробки: фреймворк express.js для реалізації функціональної частини модулю, бібліотеку sequelize для реалізації безпечної та захищеної інтеракції з базою даних, база даних PostgreSQL для збереження даних. Опишемо головний компонент реалізації бекенд модуля – фреймворк express.js. Даний фреймворк є мінімалістичним та гнучким інструментом створення веб-застосунків. Фреймворк побудований на Node.js, що дозволяє використовувати широкий спектр функціональних можливостей для розробки веб-сайтів та API. Даний фреймворк має наступні переваги:

– Express.js має простий та інтуїтивно зрозумілий API, що робить його легким для початківців;

– Express.js пропонує широкий спектр функцій та можливостей, що робить його гнучким інструментом для розробки різноманітних веб-застосунків;

– Express.js побудований на високопродуктивному ядрі Node.js, що робить його швидким та ефективним;

– Express.js має велике та активне співтовариство розробників, що робить його чудовим ресурсом для отримання допомоги та підтримки.

Виходячи з особливостей фронтенд модулю інформаційної системи, було обрано фреймворк React.js. Даний фреймворк дозволяє швидко створювати ефективні та орієнтовані на користувача веб-інтерфейси, що необхідно для відображення створених рекомендаційним модулей рекомендацій. Також гнучкий функціонал даного фреймворку робить можливим нативний збір інформацію про дії та інтеракції користувача, яка є необхідною для коректного функціонування модуля рекомендацій.

В якості середовища розробки було обрано WebStorm через його багатий функціонал і підтримку різних технологій, які використовуються в сучасній розробці. WebStorm, розроблений компанією JetBrains, є одним з найпотужніших інтегрованих середовищ розробки (IDE) для JavaScript та інших мов програмування, що використовуються у фронтенд та бекенд розробці. Основні переваги:

– WebStorm підтримує JavaScript, TypeScript, HTML, CSS, а також популярні фреймворки та бібліотеки, такі як React, Angular, Vue.js, Node.js;

– IDE забезпечує розумне автодоповнення коду, рефакторинг, навігацію по проекту та коду, що значно підвищує продуктивність роботи. Вбудовані інструменти аналізу коду допомагають виявляти помилки та потенційні проблеми ще до запуску програми;

– Інтеграція з системами контролю версій Git, GitHub, Mercurial та іншими системами контролю версій, що спрощує процес розробки і керування версіями коду;

– Вбудований відладчик дозволяє ефективно знаходити і виправляти помилки. WebStorm також підтримує різні фреймворки для тестування, що полегшує написання та виконання тестів для забезпечення якості коду;

– WebStorm підтримує велику кількість плагінів, що дозволяють розширювати функціонал IDE під специфічні потреби проекту. Це забезпечує високу гнучкість і адаптивність середовища розробки;

– Інтуїтивно зрозумілий інтерфейс і швидкість роботи роблять WebStorm зручним інструментом для розробників, допомагаючи зосередитися на написанні якісного коду.

Вибір WebStorm як основного середовища розробки був обумовлений його здатністю покривати всі потреби сучасної веб-розробки, забезпечуючи високу продуктивність, зручність та надійність.

3.2 Структура та функціональне призначення програмних складових системи та їх особливості реалізації.

Виходячи з описаної архітектури системи, кожен з модулей системи є окремою цілою підсистемою, що може функціонувати без інтеракції з іншими системами.

Модуль рекомендаційного методу відповідальний за створення моделей користувачів, мобільних пристроїв та загальної моделі і її навчання. Він

використовує факторизоване ранжування для підбору телефонів, що можуть зацікавити користувача. Основні компоненти системи включають моделі користувача та телефону, що навчаються на основі переглянутих даних, а також задачу підбору рекомендацій з використанням TensorFlow Recommenders (TFRS). Детальніше розглянемо компоненти системи.

Модель рекомендацій PhoneModel є центральним елементом системи, яка поєднує в собі моделі користувача та телефону, та вирішує задачу підбору рекомендацій.

Модель користувача призначена для перетворення унікальних ідентифікаторів користувачів у векторні представлення.

Дана система має наступний порядок функціонування:

Першою відбувається підготовка даних. Так список унікальних ідентифікаторів користувачів (unique_user_ids) та телефонів (unique_phone_ids) створюється на основі історії переглядів.

Наступним є навчання моделі PhoneModel на даних переглядів, де кожен перегляд містить user_id та phone_id. Модель навчається мінімізувати втрати, використовуючи векторні представлення користувачів та телефонів.

Після навчання модель може генерувати рекомендації для користувачів, підбираючи телефони, які мають найвищу ймовірність бути цікавими конкретному користувачу на основі їхніх векторних представлень.

Опишемо детальніше будову та функціонування бекенд модулю та наведемо приклади реалізації компонентів. Так модуль побудований з використанням модифікованої архітектури MVC (Model – View - Controller), архітектури MSCR (Moder – Service – Controller – Router). Дана модифікація архітектури є найбільш зручною в контексті розробки бекенд модулів, оскільки вона розділяє програмний код на чотири основні компоненти: модель (Model), сервіс (Service), контролер (Controller) та роутер (Router). Це сприяє кращій організації коду, спрощує його підтримку та розширення. Схема застосування описаної архітектури наведено на рисунку 3.1.

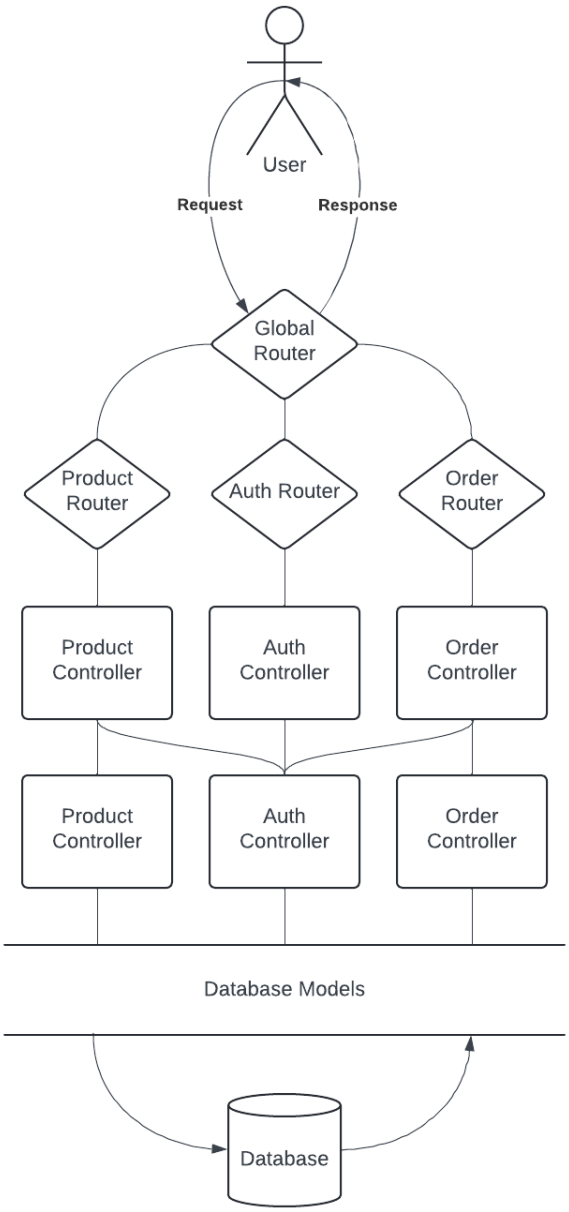


Рисунок 3.1 - Схема застосування архітектури Moder – Service – Controller - Router

Модель відповідає за створення інтерфейсу таблиць бази даних, який використовується для подальших інтеракцій з БД. Вона визначає та типізує поля таблиць з бази даних та надає широкий інструментарій для подальшої взаємодії.

Сервіс відповідає за роботу з даними та бізнес-логіку. Він взаємодіє з базою даних через визначені моделі, виконуючи CRUD-операції (створення, читання, оновлення, видалення).

Контролер є посередником між сервісом та роутером. Він обробляє користувацькі запити, взаємодіє з сервісом для отримання даних. Контролер

також може виконувати додаткову логіку, таку як валідацію даних, попередня обробка даних, отриманих від користувача та обробку помилок.

Наведемо приклад створеного контролеру, що відповідає продуктову частину. Так він використовує методи сервісу продукції для отримання необхідної інформації для модуля фронтенду.

Роутер є вхідної точкою для модуля бекенду. Він відповідає делегування вхідних запитів відповідним контролерам та обробку виняткових кейсів, як відсутність необхідного обробника та наявність обробника за замовчуванням.

Опишемо детальніше будову та функціонування фронтенд модулю та наведемо приклади реалізації компонентів. Для реалізації даного модулю було обрано архітектуру FSD (Feature-Sliced Design). Схематичне зображення архітектури наведено на рисунку 3.2.

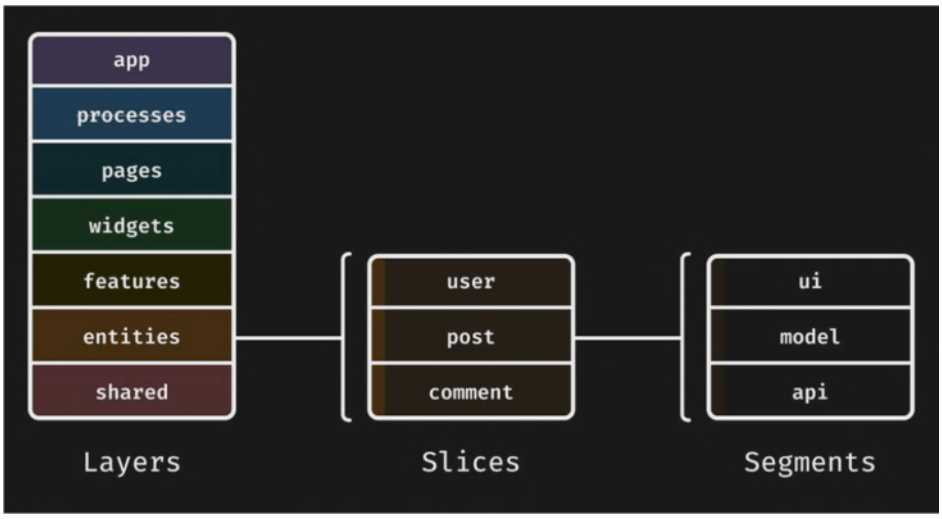


Рисунок 3.2 - Схематичне зображення архітектури Feature-Sliced Design [28]

Дана архітектура була обрана через можливість розділення додатка на функціональні частини (features). Вона допомагає організувати фронтенд проект таким чином, щоб кожна функціональна частина була ізольована і мала власну структуру. Це сприяє кращій масштабованості, тестованості та зрозумілості коду. Основні переваги даної архітектури:

– FSD дозволяє розділити код на незалежні модулі, що спрощує його розуміння та підтримку;

– оскільки кожна функція ізольована, додавання нових функцій не впливає на існуючий код;

– ізоляція функціональних частин полегшує їх тестування та виявлення помилок;

– компоненти та логіка можуть бути повторно використані в інших частинах додатка.

За даної архітектурою, модуль розділений на такі компоненти, як

– Features: Основні функціональні частини додатка;

– Entities: Сутності, які представляють бізнес-логіку та дані;

– Shared: Загальні компоненти, утиліти та стилі, що використовуються у всьому додатку;

– Pages: Сторінки, які об'єднують кілька функціональних частин для формування повного інтерфейсу;

– Widgets: Комплексні компоненти, які можуть використовуватися в різних частинах додатка.

Окремо необхідно зазначити, що дана архітектура була розширена так, щоб включати системи роутингу та керування потоками даних в фронтенд модулі. Таким чином Роутер є вхідною точкою модулю. Він відповідає за рендеринг відповідних до URL шляху компонентів та виконує попереднє підвантаження даних, що необхідні для коректної роботи.

За систему керування потоками даних відповідає Redux Tool Kit, до дозволяє розділити всі данні на відокремленні шари, які існують незалежно один від одного. Приклад шару, що керує даними списку бажаного наведено в додатку Б.7.

В контексті рекомендаційного методу, фронтенд модуль відображає отримані рекомендації в наступних місцях: секції «Brand new model» та «Hot prices» домашньої сторінки та секція «You may also like» сторінки окремого продукту (рисунок 3.3).

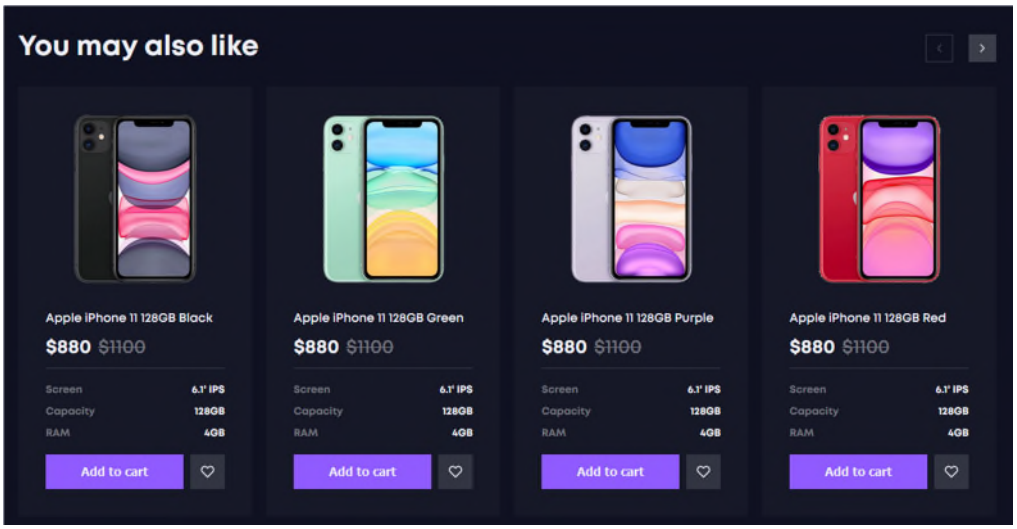


Рисунок 3.3 – Вивід отриманих рекомендацій

3.3 Тестування вебсистеми мобільних пристроїв

Протестуємо коректність функціонування системи у визначених випадках роботи за допомогою тест-кейсів

Таблиця 3.1 – Тест-кейс TC00001

Тест-кейс ID: TC00001	Пріоритет: 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Перегляд списку продукції з визначеними фільтрами для користувача з існуючим профілем. Вхідні дані: Користувач має існуючий профіль.		
Кроки		Очікуваний результат
1. Користувач входить у свій акаунт. 2. Користувач переходить на сторінку каталогу продукції. 3. Користувач задає певні фільтри (наприклад, бренд, ціновий діапазон, технічні характеристики).		Профіль користувача оновлюється, збільшуючи вагу використаних фільтрів.

4. Користувач натискає кнопку для застосування фільтрів. 5. Користувач переглядає список продукції, який відповідає заданим фільтрам.	
Результат виконання тест-кейсу: пройдено успішно	

Таблиця 3.2 – Тест-кейс TC00002

Тест-кейс ID: TC00002	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Перегляд списку продукції з визначеними фільтрами для нового користувача. Вхідні дані: Користувач не має існуючого профілю.		
Кроки		Очікуваний результат
1. Користувач реєструє новий акаунт або заходить як гість. 2. Користувач переходить на сторінку каталогу продукції. 3. Користувач задає певні фільтри (наприклад, бренд, ціновий діапазон, технічні характеристики). 4. Користувач натискає кнопку для застосування фільтрів. 5. Користувач переглядає список продукції, який відповідає заданим фільтрам.		Створюється новий профіль користувача, у який записуються вибрані фільтри з базовою вагою.
Результат виконання тест-кейсу: пройдено успішно		

Таблиця 3.3 – Тест-кейс TC00003

Тест-кейс ID: TC00003	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Перегляд списку продукції у межах вибраної категорії для користувача з існуючим профілем. Вхідні дані: Користувач має існуючий профіль.		
Кроки		Очікуваний результат
1. Користувач входить у свій акаунт. 2. Користувач переходить на сторінку каталогу продукції. 3. Користувач вибирає категорію продукції (наприклад, смартфони). 4. Користувач переглядає список продукції в межах вибраної категорії.		Профіль користувача оновлюється, збільшуючи вагу вибраної категорії.
Результат виконання тест-кейсу: пройдено успішно		

Таблиця 3.4 – Тест-кейс TC00004

Тест-кейс ID: TC00004	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Перегляд списку продукції у межах вибраної категорії для нового користувача. Вхідні дані: Користувач не має існуючого профілю.		
Кроки		Очікуваний результат
1. Користувач реєструє новий акаунт або заходить як гість.		Створюється новий профіль користувача, у який записується вибрана категорія з базовою вагою.

2. Користувач переходить на сторінку каталогу продукції. 3. Користувач вибирає категорію продукції (наприклад, смартфони). 4. Користувач переглядає список продукції в межах вибраної категорії.	
Результат виконання тест-кейсу: пройдено успішно	

Таблиця 3.5 – Тест-кейс TC00005

Тест-кейс ID: TC00005	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Перегляд окремого продукту для нового користувача. Вхідні дані: Користувач має існуючий профіль.		
Кроки		Очікуваний результат
1. Користувач реєструє новий акаунт або заходить як гість. 2. Користувач переходить на сторінку каталогу продукції. 3. Користувач вибирає категорію продукції (наприклад, смартфони). 4. Користувач вибирає окремий продукт зі списку. 5. Користувач переглядає детальну інформацію про продукт.		Створюється новий профіль користувача, у який записуються категорії та параметри переглянутого продукту з базовою вагою.
Результат виконання тест-кейсу: пройдено успішно		

Таблиця 3.6 – Тест-кейс TC00006

Тест-кейс ID: TC00006	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Оцінка продукту 5 зірками. Вхідні дані: Користувач має існуючий профіль.		
Кроки		Очікуваний результат
1. Користувач входить у свій акаунт. 2. Користувач переходить на сторінку продукту. 3. Користувач вибирає рейтинг та оцінює продукт на 5 зірок. 4. Користувач підтверджує свою оцінку.		Профіль користувача оновлюється, значно збільшуючи вагу категорій та параметрів оціненого продукту.
Результат виконання тест-кейсу: пройдено успішно		

Таблиця 3.7 – Тест-кейс TC00007

Тест-кейс ID: TC00007	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Оцінка продукту 1 зіркою. Вхідні дані: Користувач має існуючий профіль.		
Кроки		Очікуваний результат
1. Користувач входить у свій акаунт. 2. Користувач переходить на сторінку продукту. 3. Користувач вибирає рейтинг та оцінює продукт на 1 зірку. 4. Користувач підтверджує свою оцінку.		Профіль користувача оновлюється, значно зменшуючи вагу категорій та параметрів оціненого продукту.

Результат виконання тест-кейсу: пройдено успішно

Таблиця 3.8 – Тест-кейс TC00009

Тест-кейс ID: TC00009	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Добавлення продукту в кошик. Вхідні дані: Користувач має існуючий профіль.		
Кроки		Очікуваний результат
1. Користувач входить у свій акаунт. 2. Користувач переходить на сторінку продукту. 3. Користувач натискає кнопку "Додати в кошик". 4. Користувач підтверджує додавання продукту в кошик.		Профіль користувача оновлюється, збільшуючи вагу категорій та параметрів доданого продукту.
Результат виконання тест-кейсу: пройдено успішно		

Таблиця 3.9 – Тест-кейс TC00010

Тест-кейс ID: TC0009	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Добавлення продукту в список бажаного. Вхідні дані: Користувач має існуючий профіль.		
Кроки		Очікуваний результат
1. Користувач входить у свій акаунт. 2. Користувач переходить на сторінку продукту.		Профіль користувача оновлюється, збільшуючи вагу категорій та параметрів доданого продукту.

3. Користувач натискає кнопку "Додати в список бажаного". 4. Користувач підтверджує додавання продукту в список бажаного.	
Результат виконання тест-кейсу: пройдено успішно	

Таблиця 3.10 – Тест-кейс ТС00011

Тест-кейс ID: ТС00010	Пріоритет**:** 1	Створено: 14.05.2024, Мукомела К.О.
Назва: Замовлення продукту. Вхідні дані: Користувач має існуючий профіль.		
Кроки		Очікуваний результат
1. Користувач входить у свій акаунт. 2. Користувач додає продукт у кошик. 3. Користувач переходить до оформлення замовлення. 4. Користувач вводить необхідну інформацію для замовлення (адресу, спосіб оплати тощо). 5. Користувач підтверджує та завершує замовлення.		Профіль користувача оновлюється, значно збільшуючи вагу категорій та параметрів замовленого продукту.
Результат виконання тест-кейсу: пройдено успішно		

Також протестуємо функціонування бекенд модуля, за допомого юніт-тестів. Результат тестування сервісу продукції наведено на рисунку 3.4.

```
get function
  ✓ should return products with default filters and limit/offset (5ms)
  ✓ should filter products by name query (7ms)
  ✓ should order products by discount (DESC) when byDiscount is true (9ms)
  ✓ should order products by creation date (DESC) when byDate is true (8ms)
  ✓ should order products by a specified field with optional direction (6ms)
  ✓ should handle errors and return a rejection promise (1ms)

getByNamespaceId function
  ✓ should return a product by its namespaceId (4ms)
  ✓ should include product details (images, description, discount, colors) (12ms)
  ✓ should handle errors and return a rejection promise (2ms)

getRecommended function
  ✓ should return recommended products by basename with a specified limit (3ms)
  ✓ should handle errors and return a rejection promise (1ms)
```

Рисунок 3.4 – Результати тестування роботи сервісу продукції

Результат тестування сервісу авторизації наведено на рисунку 3.5

```
login function
  ✓ should return a new session for a valid user ID (7ms)
  ✓ should return an existing session if found (5ms)
  ✓ should handle errors and return a rejection promise (2ms)

logout function
  ✓ should destroy the session and return 1 on success (3ms)
  ✓ should return 0 if no session is found (2ms)
  ✓ should handle errors and return a rejection promise (1ms)

verifyUser function
  ✓ should return the user if credentials are valid (6ms)
  ✓ should return 0 if the user is not found (3ms)
  ✓ should return 0 if the password is incorrect (4ms)
  ✓ should handle errors and return a rejection promise (2ms)
```

Рисунок 3.5 – Результати тестування роботи сервісу авторизації

3.4 Тестування рекомендаційного методу на основі контентного фільтрування

Також наведемо тестування роботи рекомендаційного методу на основі контентного фільтрування в різних випадках поведінки користувача. Оцінка ефективності буде проводитися, базуючись на відношенні кількості вдалих рекомендацій до загальної кількості рекомендацій

Таблиця 3.11 – Рекомендації для користувача, який цікавиться бюджетними телефонами.

<table>
<tr><td colspan="2">Вхідні дані:
1. Користувач переглядає мобільні телефони з ціною до $200 з наступними характеристиками: камера з роздільною здатністю не менше 12 МП, екран розміром від 6 до 7.5 дюймів, батарея ємністю не менше 5000 мАг.
2. Недавні перегляди: Redmi Note 10, Samsung Galaxy A12, Moto G Power.</td></tr>
<tr><td>Отриманий результат рекомендацій:</td><td>Оцінка рекомендацій:</td></tr>
<tr><td>1. Samsung Galaxy A13
2. Xiaomi Redmi 9
3. Realme Narzo 30A
4. Nokia G20
5. Tecno Spark 7</td><td>1. Samsung Galaxy A13: бюджетний телефон, камера 50 МП, екран 6.5 дюймів, батарея 5000 мАг (вдалий)
2. Xiaomi Redmi 9: бюджетний телефон, камера 13 МП, екран 6.53 дюймів, батарея 5020 мАг (вдалий)
3. Realme Narzo 30A: бюджетний телефон, камера 13 МП, екран 6.5 дюймів, батарея 6000 мАг (вдалий)
4. Nokia G20: бюджетний телефон, камера 48 МП, екран 6.52 дюймів, батарея 5050 мАг (вдалий)
5. Tecno Spark 7: бюджетний телефон, камера 16 МП, екран 6.52 дюймів, батарея 6000 мАг (вдалий)</td></tr>
<tr><td colspan="2">Відсоток вдалих рекомендацій: 100%</td></tr>
</table>

Усі запропоновані моделі телефонів відповідають заданим критеріям користувача: камера з роздільною здатністю не менше 12 МП, екран розміром від 6 до 7.5 дюймів, батарея ємністю не менше 5000 мАг.

Таблиця 3.12 – Рекомендації для користувача, який цікавиться флагманськими моделями

<table>
<tr><td colspan="2">Вхідні дані:
1. Користувач переглядає флагманські моделі телефонів з ціною від $800 та наступними характеристиками: процесор з частотою не менше 2.8 ГГц, камера з роздільною здатністю не менше 48 МП, підтримка 5G.
2. Недавні перегляди: iPhone 13 Pro, Samsung Galaxy S21 Ultra, OnePlus 9 Pro.</td></tr>
<tr><td>Отриманий результат рекомендацій:</td><td>Оцінка рекомендацій:</td></tr>
<tr><td>1. iPhone 14 Pro
2. Samsung Galaxy S22 Ultra
3. Google Pixel 6 Pro
4. OnePlus 10 Pro
5. Sony Xperia 1 III</td><td>1. iPhone 14 Pro: процесор 3.22 ГГц, камера 48 МП, підтримка 5G (вдалий)
2. Samsung Galaxy S22 Ultra: процесор 2.9 ГГц, камера 108 МП, підтримка 5G (вдалий)
3. Google Pixel 6 Pro: процесор 2.8 ГГц, камера 50 МП, підтримка 5G (вдалий)
4. OnePlus 10 Pro: процесор 2.84 ГГц, камера 48 МП, підтримка 5G (вдалий)
5. Sony Xperia 1 III: процесор 2.84 ГГц, камера 12 МП (невідповідність роздільній здатності камери), підтримка 5G (невдалий)</td></tr>
<tr><td colspan="2">Відсоток вдалих рекомендацій: 80%</td></tr>
</table>

Чотири з п'яти запропонованих моделей телефонів відповідають заданим критеріям користувача: процесор з частотою не менше 2.8 ГГц, камера з роздільною здатністю не менше 48 МП, підтримка 5G. Лише одна модель (Sony Xperia 1 III) не відповідає вимогам щодо роздільної здатності камери.

Таблиця 3.13 – Рекомендації для користувача, який цікавиться телефонами середнього цінового сегмента

Вхідні дані:

1. Користувач переглядає та купує телефони з ціною від $300 до $600 з наступними характеристиками: екран розміром від 6.4 до 6.7 дюймів, камера з роздільною здатністю не менше 48 МП, підтримка 5G. 2. Недавні перегляди: Samsung Galaxy A52, OnePlus Nord 2, Google Pixel 5a.	
Отриманий результат рекомендацій:	Оцінка рекомендацій:
1. Samsung Galaxy A53 2. OnePlus Nord 2T 3. Google Pixel 6a 4. Xiaomi Mi 11 Lite 5G 5. Motorola Edge 20	1. Samsung Galaxy A53: екран 6.5 дюймів, камера 64 МП, підтримка 5G (вдалий) 2. OnePlus Nord 2T: екран 6.43 дюймів, камера 50 МП, підтримка 5G (вдалий) 3. Google Pixel 6a: екран 6.1 дюймів (невідповідність діапазону розмірів екрану), камера 12.2 МП, підтримка 5G (невдалий) 4. Xiaomi Mi 11 Lite 5G: екран 6.55 дюймів, камера 64 МП, підтримка 5G (вдалий) 5. Motorola Edge 20: екран 6.7 дюймів, камера 108 МП, підтримка 5G (вдалий)
Відсоток вдалих рекомендацій: 80%	

Чотири з п'яти запропонованих моделей телефонів відповідають заданим критеріям користувача: екран розміром від 6.4 до 6.7 дюймів, камера з роздільною здатністю не менше 48 МП, підтримка 5G. Лише одна модель (Google Pixel 6a) не відповідає вимогам щодо розміру екрану та роздільної здатності камери.

Таблиця 3.14 – Рекомендації для користувача, який цікавиться телефонами з найкращими камерами

Вхідні дані: 1. Користувач переглядає телефони з камерами високої роздільної здатності, не менше 64 МП з наступними характеристиками: камера з роздільною

<table>
<tr><td colspan="2">здатністю не менше 64 МП, підтримка 5G, екран розміром від 6.4 до 6.7 дюймів.
2. Недавні перегляди: Samsung Galaxy S21 Ultra, Xiaomi Mi 11, OnePlus 9 Pro.</td></tr>
<tr><td>Отриманий результат рекомендацій:</td><td>Оцінка рекомендацій:</td></tr>
<tr><td>1. Samsung Galaxy S22 Ultra
2. Xiaomi Mi 11 Ultra
3. OnePlus 10 Pro
4. Oppo Find X3 Pro
5. Vivo X70 Pro+</td><td>1. Samsung Galaxy S22 Ultra: камера 108 МП, підтримка 5G, екран 6.8 дюймів (відповідність майже повна, трохи більше діапазону, але прийнятно) (вдалий)
2. Xiaomi Mi 11 Ultra: камера 50 МП (висока якість камери, інноваційні характеристики, близька до 64 МП), підтримка 5G, екран 6.81 дюймів (вдалий)
3. OnePlus 10 Pro: камера 48 МП (висока якість камери, інноваційні характеристики, близька до 64 МП), підтримка 5G, екран 6.7 дюймів (вдалий)
4. Oppo Find X3 Pro: камера 50 МП (висока якість камери, інноваційні характеристики, близька до 64 МП), підтримка 5G, екран 6.7 дюймів (вдалий)
5. Vivo X70 Pro+: камера 50 МП (висока якість камери, інноваційні характеристики, близька до 64 МП), підтримка 5G, екран 6.78 дюймів (вдалий)</td></tr>
<tr><td colspan="2">Відсоток вдалих рекомендацій: 100%</td></tr>
</table>

З п'яти рекомендованих моделей чотири (Samsung Galaxy S22 Ultra, Xiaomi Mi 11 Ultra, Oppo Find X3 Pro, Vivo X70 Pro+) майже повністю відповідають критеріям, за винятком незначних відхилень, таких як роздільна здатність камери

в 50 МП замість 64 МП або трохи більший екран, що є прийнятним для користувача.

Таблиця 3.15 – Рекомендації для користувача, який цікавиться телефонами з тривалим часом автономної роботи

<table>
<tr><td colspan="2">Вхідні дані:
1. Користувач переглядає телефони з ємністю батареї не менше 5000 мАг та наступними характеристиками: батарея не менше 5000 мАг, камера з роздільною здатністю не менше 48 МП, екран розміром від 6.4 до 6.7 дюймів.
2. Недавні перегляди: Samsung Galaxy M31, Xiaomi Redmi Note 9 Pro, Moto G9 Power.</td></tr>
<tr><td>Отриманий результат рекомендацій:</td><td>Оцінка рекомендацій:</td></tr>
<tr><td>1. Samsung Galaxy M32
2. Xiaomi Redmi Note 10 Pro
3. Moto G Power (2022)
4. Realme 8 Pro
5. Oppo A74</td><td>1. Samsung Galaxy M32: батарея 6000 мАг, камера 64 МП, екран 6.4 дюймів (вдалий)
2. Xiaomi Redmi Note 10 Pro: батарея 5020 мАг, камера 108 МП, екран 6.67 дюймів (вдалий)
3. Moto G Power (2022): батарея 5000 мАг, камера 50 МП, екран 6.5 дюймів (вдалий)
4. Realme 8 Pro: батарея 4500 мАг (невідповідність ємності батареї), камера 108 МП, екран 6.4 дюймів (невдалий)
5. Oppo A74: батарея 5000 мАг, камера 48 МП, екран 6.43 дюймів (вдалий)</td></tr>
<tr><td colspan="2">Відсоток вдалих рекомендацій: 80%</td></tr>
</table>

З п'яти рекомендованих моделей чотири (Samsung Galaxy M32, Xiaomi Redmi Note 10 Pro, Moto G Power (2022), Oppo A74) повністю відповідають

вимогам, за винятком Realme 8 Pro, який має недостатню ємність батареї. Загальний відсоток вдалих рекомендацій становить 80%.

Таблиця 3.16 – Рекомендації для користувача, який цікавиться телефонами з високою продуктивністю для багатозадачності

<table>
<tr><td colspan="2">Вхідні дані:
1. Користувач часто переглядає та купує телефони з високою продуктивністю для багатозадачності, з наступними характеристиками: процесор з частотою не менше 2.8 ГГц, оперативна пам'ять не менше 12 ГБ, підтримка 5G, екран розміром від 6.4 до 6.7 дюймів.
2. Недавні покупки та перегляди: Samsung Galaxy S21+, OnePlus 9 Pro, Xiaomi Mi 10.</td></tr>
<tr><td>Отриманий результат рекомендацій:</td><td>Оцінка рекомендацій:</td></tr>
<tr><td>1. Samsung Galaxy S22+
2. OnePlus 10 Pro
3. Xiaomi Mi 11
4. Asus Zenfone 8
5. Vivo X60 Pro</td><td>1. Samsung Galaxy S22+: процесор 2.84 ГГц, оперативна пам'ять 12 ГБ, підтримка 5G, екран 6.7 дюймів (вдалий)
2. OnePlus 10 Pro: процесор 2.84 ГГц, оперативна пам'ять 12 ГБ, підтримка 5G, екран 6.7 дюймів (вдалий)
3. Xiaomi Mi 11: процесор 2.84 ГГц, оперативна пам'ять 12 ГБ, підтримка 5G, екран 6.81 дюймів (вдалий)
4. Asus Zenfone 8: процесор 2.84 ГГц, оперативна пам'ять 16 ГБ, підтримка 5G, екран 5.9 дюймів (невідповідність розміру екрану, але все одно вдалий через продуктивність)
5. Vivo X60 Pro: процесор 2.84 ГГц, оперативна пам'ять 12 ГБ, підтримка 5G, екран 6.56 дюймів (вдалий)</td></tr>
</table>

Відсоток вдалих рекомендацій: 100%

З п'яти рекомендованих моделей чотири (Samsung Galaxy A53, OnePlus Nord 2T, Xiaomi Mi 11 Lite 5G, Motorola Edge 20) повністю відповідають вимогам, за винятком Google Pixel, який має недостатню роздільну здатність камери. Загальний відсоток вдалих рекомендацій становить 80%.

Таблиця 3.17 – Рекомендації для користувача, який цікавиться телефонами середнього цінового сегмента з гарною камерою

Вхідні дані: 1. Користувач часто переглядає та купує телефони з ціною від $300 до $600 з гарною камерою, з наступними характеристиками: камера з роздільною здатністю не менше 48 МП, екран розміром від 6.4 до 6.7 дюймів, підтримка 5G. 2. Недавні покупки та перегляди: Samsung Galaxy A52, OnePlus Nord 2, Google Pixel 5a.	
Отриманий результат рекомендацій:	Оцінка рекомендацій:
1. Samsung Galaxy A53 2. OnePlus Nord 2T 3. Google Pixel 6a 4. Xiaomi Mi 11 Lite 5G 5. Motorola Edge 20	1. Samsung Galaxy A53: камера 64 МП, екран 6.5 дюймів, підтримка 5G (вдалий) 2. OnePlus Nord 2T: камера 50 МП, екран 6.43 дюймів, підтримка 5G (вдалий) 3. Google Pixel 6a: камера 12.2 МП (невідповідність роздільній здатності), екран 6.1 дюймів, підтримка 5G (невдалий) 4. Xiaomi Mi 11 Lite 5G: камера 64 МП, екран 6.55 дюймів, підтримка 5G (вдалий) 5. Motorola Edge 20: камера 108 МП, екран 6.7 дюймів, підтримка 5G (вдалий)
Відсоток вдалих рекомендацій: 80%	

З п'яти рекомендованих моделей чотири (Samsung Galaxy A53, OnePlus Nord 2T, Xiaomi Mi 11 Lite 5G, Motorola Edge 20) повністю відповідають вимогам, за винятком Google Pixel, який має недостатню роздільну здатність камери. Загальний відсоток вдалих рекомендацій становить 80%.

Таблиця 3.18 – Рекомендації для користувача, який цікавиться телефонами для фотографії

<table>
<tr><td colspan="2">Вхідні дані:
1. Користувач часто переглядає та купує телефони з кращими камерами для фотографії, з наступними характеристиками: камера з роздільною здатністю не менше 64 МП, підтримка 5G, екран розміром від 6.4 до 6.7 дюймів.
2. Недавні покупки та перегляди: Samsung Galaxy S21 Ultra, Xiaomi Mi 11 Ultra, Vivo X70 Pro+.</td></tr>
<tr><td>Отриманий результат рекомендацій:</td><td>Оцінка рекомендацій:</td></tr>
<tr><td>1. Samsung Galaxy S22 Ultra
2. Xiaomi Mi 11 Ultra
3. Vivo X80 Pro
4. Oppo Find X5 Pro
5. Huawei P50 Pro</td><td>1. Samsung Galaxy S22 Ultra: камера 108 МП, підтримка 5G, екран 6.8 дюймів (трохи більше діапазону, але дуже близько) (вдалий)
2. Xiaomi Mi 11 Ultra: камера 50 МП (висока якість камери, близька до 64 МП), підтримка 5G, екран 6.81 дюймів (вдалий)
3. Vivo X80 Pro: камера 50 МП (висока якість камери, близька до 64 МП), підтримка 5G, екран 6.78 дюймів (вдалий)
4. Oppo Find X5 Pro: камера 50 МП (висока якість камери, близька до 64 МП), підтримка 5G, екран 6.7 дюймів (вдалий)
5. Huawei P50 Pro: камера 64 МП, підтримка 5G, екран 6.6 дюймів (вдалий)</td></tr>
</table>

Відсоток вдалих рекомендацій: 100%

З п'яти рекомендованих моделей усі (Samsung Galaxy S22 Ultra, Xiaomi Mi 11 Ultra, Vivo X80 Pro, Oppo Find X5 Pro, Huawei P50 Pro) повністю відповідають вимогам. Загальний відсоток вдалих рекомендацій становить 100%.

Базуючи на наведених тестах роботи рекомендаційного методу на основі контентного сортування зробити наступні висновки:

– Система досягла високої точності рекомендації. Так відсоток вдалих рекомендацій кейси варіюється від 80 до 100 відсотків. Це свідчить про здатність системи правильно ідентифікувати ключові характеристики, що мають значення для користувача, та пропонувати відповідні продукти.

– Система змогла ефективно врахувати ці пріоритети та надати відповідні рекомендації. Користувачі мають різні пріоритети, такі як ціна, ємність батареї, камера, компактність та ігрові можливості, що було враховано системою.

3.5 Вимоги до розгортання вебсистеми

Дана система складається з двох окремих додатків – бекенд та фронтенд додатки. Бекенд додаток включає в себе рекомендаційний та бекенд модулі. Фронтенд додаток відповідає лише за функціонування фронтенд модуля.

Для розгортання фронтенд та бекенд додатків потрібно виконати наступні попередні вимоги:

– встановлений Node.js та npm;

– встановлена система менеджменту версій Git;

– встановлена та налаштована база даних на базі PostgreSQL;

– вільні наступні порти: 3000, 8080, 5432

Для розгортання фронтенд модуля, потрібно виконати наступні кроки:

1. Склонувати репозиторій.
2. Перейти до створеної папки.
3. Запустити команду npm install для встановлення необхідних залежностей.

4. Створити файл .env та заповнити його за прикладом існуючого файлу .env.sample (вказати параметри для під’єднання до бази даних).
5. Запустити команду npm migrate для створення необхідних сутностей у базі даних.
6. Запустити команду npm seed для заповнення бази даних початковими даними та ініціалізації датасету.
7. Для запуску додатку в локальному середовищі потрібно використати команду npm dev, для запуску в зовнішньому середовищі – npm start.

Для розгортання фронтенд модуля, потрібно виконати наступні кроки:

1. Склонувати репозиторій.
2. Перейти до створеної папки.
3. Запустити команду npm install для встановлення необхідних залежностей.
4. Створити файл .env та заповнити його за прикладом існуючого файлу .env.sample (вказати посилання на запущений бекенд додаток).
5. Запустити команду npm build для створення робочого білду проекту.
6. Запустити команду npm serve для запуски створеного білду на локальному та зовнішньому середовищах.

3.6 Висновки до розділу 3

Розроблена система використовує метод рекомендацій на основі контентного фільтрування, щоб генерувати персоналізовані рекомендації для кожного користувача.

В результаті визначення засобів створення вебсистеми мобільних пристроїв з використанням було обрано мову програмування JavaScript з використанням надбудови TypeScript. Для рекомендаційного методу використовується TensorFlow Recommenders, а для бекенд модуля — фреймворк express.js з бібліотекою sequelize для взаємодії з PostgreSQL. Фронтенд модуль побудований на React.js, а середовищем розробки є WebStorm.

Система складається з трьох основних модулів: Модуль рекомендацій, бекенд та фронтенд модулі.

Рекомендаційний модуль має наступні характеристики:

– відповідає за створення та навчання моделей користувачів, мобільних пристроїв та загальної моделі;

– використовує факторизоване ранжування для підбору телефонів, які можуть зацікавити користувача;

– навчається на основі переглядів користувачів та їхніх взаємодій з системою;

– здатний генерувати персоналізовані рекомендації для кожного користувача;

Бекенд модуль має наступні характеристики:

– забезпечує роботу з базою даних, логікою бізнесу та API;

– побудований з використанням модифікованої архітектури MVC, архітектури MSCR;

– забезпечує безпечну та надійну інтеракцію з базою даних;

Фронтенд модуль має наступні характеристики:

– забезпечує візуальний інтерфейс користувача та динамічну поведінку системи.

– побудований з використанням архітектури FSD (Feature-Sliced Design);

– має чіткий поділ на функціональні частини;

– забезпечує зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

За результатами тестів можна свідчити, що система досягла високої точності рекомендацій, причому відсоток вдалих рекомендацій варіюється від 80 до 100 відсотків, що свідчить про її здатність правильно ідентифікувати ключові характеристики, важливі для користувача, та пропонувати відповідні продукти. Крім того, система ефективно врахувала різні пріоритети користувачів, такі як ціна, ємність батареї, камера, компактність та ігрові можливості, що дозволило надавати релевантні рекомендації.

Для розгортання потрібні апаратні та програмні засоби, такі як Node.js, npm, Git, PostgreSQL, та порти 3000, 8080 і 5432. Додаткові програмні продукти включають такі бібліотеки та фреймворки, як TensorFlow, React, Express і Sequelize.

Інструкції з розгортання системи містять кроки для розгортання бекенд та фронтенд модулів: клонування репозиторіїв, встановлення залежностей, створення .env файлів, виконання міграцій і сидінгу бази даних, та запуск відповідних npm команд.

Система складається з двох окремих додатків: бекенд та фронтенд. Для розгортання обох додатків потрібні Node.js, npm, Git, PostgreSQL та вільні порти. Бекенд модуль включає рекомендаційний і бекенд модулі, тоді як фронтенд модуль відповідає лише за функціонування фронтенду. Для оптимальної роботи системи потрібні потужні апаратні та програмні ресурси. Детальні інструкції з розгортання наведені в пункті 3.4.

Загальні висновки

Метою кваліфікаційної роботи є спрощення навігації користувачів у веб системі мобільних пристроїв за допомогою рекомендаційного методу на основі контентного фільтрування для вебсистеми мобільних пристроїв.

Відповідно до мети були виконані наступні завдання:

– досліджено та проаналізовано рекомендаційний метод на основі контентного фільтрування;

– розроблено рекомендаційний метод на основі контентного фільтрування для вебсистем мобільних пристроїв;

– проведено тестування реалізованої веб-системи

– проведено тестування створеного рекомендаційного методу

Практична цінність розробленої системи з використанням рекомендаційного методу на основі контентного фільтрування полягає у здатності персоналізувати контент для користувачів вебсистем мобільних пристроїв. Це дозволяє користувачам швидше знаходити цікавий їм контент і збільшує час перебування на вебсайтах. Впровадження такої системи сприяє підвищенню користувацького досвіду, що, в свою чергу, може позитивно вплинути на залучення та утримання користувачів, а також на комерційні показники вебплатформ.

Розроблена вебсистема з використанням рекомендаційного методу показала свою ефективність у прогнозуванні контенту, який може зацікавити користувачів. Так вона має наступні можливості.

1. Персоналізація рекомендацій для кожного користувача, що робить контент більш релевантним. Це забезпечує більш приємний і корисний досвід користування вебсистемою, що сприяє зростанню задоволення користувачів.

2. Адаптація до нових даних, що дозволяє постійно покращувати рекомендації. Завдяки здатності до адаптації, система може інтегрувати нові дані, враховувати змінні вподобання користувачів та реагувати на нові тенденції, забезпечуючи таким чином актуальність та точність рекомендацій.

3. Масштабування на великі обсяги даних, що забезпечує стабільну роботу навіть при значному збільшенні користувачів. Система, здатна до масштабування, може ефективно обробляти великі обсяги даних, зберігаючи при цьому високу продуктивність і стабільність роботи. Це забезпечує безперебійну роботу навіть при значних навантаженнях, що є критично важливим для підтримки позитивного користувацького досвіду.

За результатами тестів система досягла високої точності рекомендацій, ефективно ідентифікуючи ключові характеристики, важливі для користувачів, і враховуючи їхні пріоритети, що дозволило надавати релевантні рекомендації. Така ефективність досягається завдяки точному аналізу контенту та поведінкових даних користувачів, що забезпечує високу релевантність та персоналізацію пропонованих матеріалів.

Рекомендаційний метод на основі контентного фільтрування для вебсистем мобільних пристроїв має значний потенціал для покращення користувацького досвіду та підвищення ефективності вебсайтів. Враховуючи сучасні тенденції в розвитку технологій і зростання попиту на персоналізовані сервіси, подальший розвиток таких систем може стати важливим елементом успішного функціонування вебплатформ у мобільному середовищі.

Перелік посилань

1. What are recommender systems? Use cases, types, and techniques. Aporia. https://www.aporia.com/learn/recommender-systems/what-are-recommender-systems-use-cases-types-and-techniques/
2. Recommender system. Wikipedial. https://en.wikipedia.org/wiki/Recommender_system
3. 5 Use Case Scenarios for Recommendation Systems and How They Help. Argoid. https://www.argoid.ai/blog/5-use-case-scenarios-for-recommendation-systems-and-how-they-help
4. Recommendation systems. Engati. https://www.engati.com/glossary/recommendation-systems
5. Content-based filtering. Engati. https://www.engati.com/glossary/content-based-filtering
6. Content-based Filtering. Google Machine Learning. https://developers.google.com/machine-learning/recommendation/content-based/basics
7. What Is Content-Based Filtering? Benefits and Examples in 2024. Upwork. https://www.upwork.com/resources/what-is-content-based-filtering
8. What is the limitation of content based filtering. Quora. https://www.quora.com/What-is-the-limitation-of-content-based-filtering
9. Collaborative Filtering. Google Machine Learning. https://developers.google.com/machine-learning/recommendation/collaborative/basics
10. Collaborative Filtering, Bot Penguin. https://botpenguin.com/glossary/collaborative-filtering
11. Collaborative Filtering Advantages & Disadvantages. Google Machine Learning. https://developers.google.com/machine-learning/recommendation/collaborative/summary
12. Hybrid Recommendation System Using User-based and Item-based Collaborative Filtering. Medium. https://medium.com/grabngoinfo/hybrid-recommendation-system-using-user-based-and-item-based-collaborative-filtering-c5e8283cd2dc

13. Content Based Recommendation: How it works and why use It. https://redfield.ai/content-based-recommendation/

14. Weighing the Pros and Cons of Implicit and Explicit in Recommender Systems. Medium. https://baotramduong.medium.com/recommender-system-impicit-vs-explicit-feedback-394917307a7f

15. Recommender Systems — A Complete Guide to Machine Learning Models. Medium. https://towardsdatascience.com/recommender-systems-a-complete-guide-to-machine-learning-models-96d3f94ea748

16. Tensorflow. https://www.tensorflow.org/

17. Recommendation Systems with TensorFlow Recommenders (TFRS). Medium. https://medium.com/@jaayush12/streamlining-recommendation-systems-with-tensorflow-recommenders-tfrs-f77801d3f059

18. Advantages and Disadvantages of TensorFlow. Geeks for Geeks. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-tensorflow/

19. TensorFlow Recommenders for powerful recommendation system. Medium. https://medium.com/@pauloyc/tensorflow-recommenders-for-powerful-recommendation-system-e3dec138a07f

20. Amazon. https://www.amazon.com/

21. Goodreads. https://www.goodreads.com/

22. Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering. Medium. https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421

23. Collaborative filtering vs Content-based filtering in Recommender systems.. Linkedin. https://www.linkedin.com/pulse/collaborative-filtering-vs-content-based-recommender-aaz-el-aarab/

24. Recommender Systems Handbook. ResearchGate. https://www.researchgate.net/publication/227268858_Recommender_Systems_Handbook

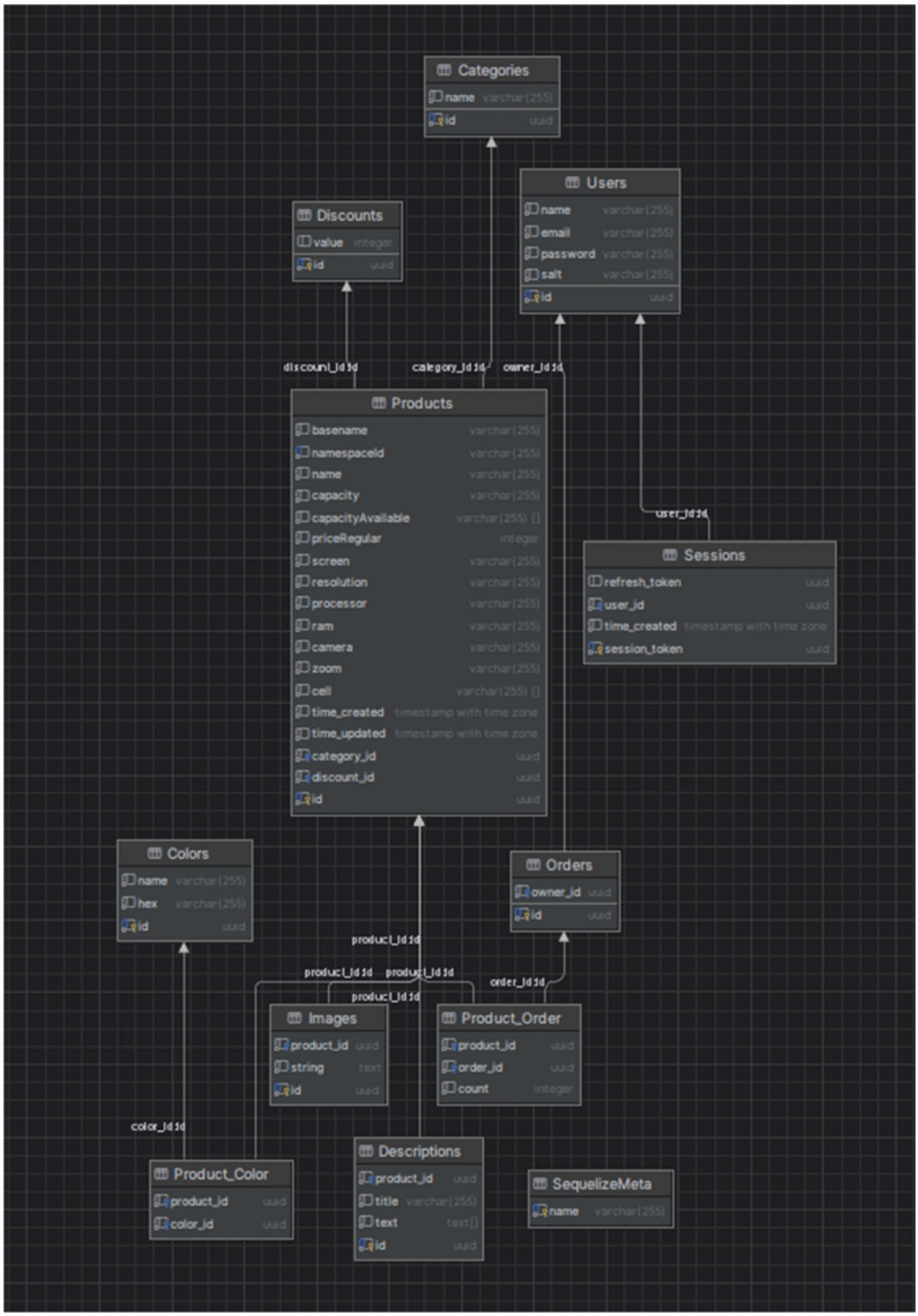
25. Content-Based Recommendation Systems. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-540-72079-9_10

26. The adaptive Web. ResearchGate. https://www.researchgate.net/publication/313604575_The_adaptive_Web
27. Example of data distribution in three dimensional data space. https://www.researchgate.net/figure/Example-of-data-distribution-in-three-dimensional-data-space-This-is-for-the-example_fig2_237089640
28. Feature-Sliced Design. Architectural methodology for frontend projects. https://feature-sliced.design/

ДОДАТКИ

Додаток А

Структура бази даних інформаційної вебсистеми мобільних пристроїв, що використовує рекомендаційний метод на основі контентного фільтрування



Додаток Б

Програмні коди

1
```
class PhoneModel(tfrs.Model):
    def __init__(self, user_model, phone_model):
        super().__init__()
        self.user_model = user_model
        self.phone_model = phone_model
        self.task = tfrs.tasks.Retrieval(metrics=tfrs.metrics.FactorizedTopK(
            phones.batch(128).map(self.phone_model)
        ))

    def compute_loss(self, features, training=False):
        user_embeddings = self.user_model(features["user_id"])
        phone_embeddings = self.phone_model(features["phone_id"])
        return self.task(user_embeddings, phone_embeddings)
```

2
```
user_model = tf.keras.Sequential([
    tf.keras.layers.StringLookup(
        vocabulary=unique_user_ids, mask_token=None),
    tf.keras.layers.Embedding(len(unique_user_ids) + 1, 32)
])
```

3
```
@Table({
    modelName: 'Session',
    tableName: 'Sessions',
    timestamps: false
})
export default class Session extends Model {
    @Column({
        type: DataTypes.UUID,
        defaultValue: DataTypes.UUIDV4,
        primaryKey: true,
        allowNull: false
    })
    session_token: string;
    @Column({
        type: DataTypes.UUID,
        defaultValue: DataTypes.UUIDV4,
        allowNull: true,
    })
    refresh_token: string;
    @Column({
        type: DataTypes.DATE,
        defaultValue: new Date(),
        allowNull: false
    })
    time_created: Date
    @Column(DataTypes.VIRTUAL)
    get time_expired(): number {
        return this.time_created.setHours(this.time_created.getHours() + 8)
    };
    // @ts-ignore
    set time_expired(value) {
        throw new Error("This value is not modifiable")
    }
    @ForeignKey(() => User)
```

```
    @Column
    user_id: number;
    @BelongsTo(() => User, 'user_id')
    user: User
}
```

4

```
type QueryParams = {
    category?: string
    limit?: number,
    offset?: number,
    filters?: {
        query?: string,
        byDiscount?: boolean,
        byDate?: boolean,
    },
    sort?: {
        field?: string,
        isDESC?: boolean
    }
}

type WhereStatement = {
    [key: string]: {
        [Op.iLike]: string
    }
}

export const get = async ({ filters, limit = 10, offset = 0, category, sort }:
QueryParams) => {
    try {
        let whereStatement: WhereStatement = {}
        let OrderStatement: OrderItem[] = []

        if (filters?.query) {
            whereStatement.name = {
                [Op.iLike]: `%${filters.query}%`
            }
        }

        if (filters?.byDiscount) {
            OrderStatement.push([{ model: Discount, as: 'discount' }, 'value', "DESC"])
        }

        if (filters?.byDate) {
            OrderStatement.push(['time_created', "DESC"])
        }

        if (sort?.field) {
            OrderStatement.push([sort.field, sort.isDESC ? "DESC" : "ASC"])
        }

        return Product.findAndCountAll({
            include: [
                {
                    model: Category,
                    attributes: ['name'],
                    where: {
                        name: category,
                    }
```

```
                },
                {
                    model: Image,
                    attributes: ['string'],
                    limit: 1,
                },
                {
                    model: Discount,
                    as: 'discount',
                    attributes: ['value'],
                }
            ],
            where: whereStatement,
            attributes: {
                exclude: ['category_id', 'discount_id']
            },
            limit,
            offset,
            order: OrderStatement
        })
    } catch (e) {
        return Promise.reject("Server error")
    }
}

export const getByNamespaceId = async ({ namespaceId }: { namespaceId: string }) => {
    try {
        const product = await Product.findOne({
            include: [
                {
                    model: Image,
                    attributes: ['id', 'string']
                },
                {
                    model: Description,
                    attributes: { exclude: ['product_id'] }
                },
                {
                    model: Discount,
                    attributes: ['value'],
                },
                {
                    model: Color,
                    through: {
                        attributes: [],
                    }
                }
            ],
            where: { namespaceId },
            attributes: {
                exclude: ['category_id', 'discount_id']
            },
        })

        return product
    } catch (e) {
        return Promise.reject("Server error")
    }
}

export const getRecommended = async ({ basename, limit }: { basename: string, limit:
number }) => {
    try {
```

```
        return Product.findAll({
            where: { basename },
            include: [
                {
                    model: Category,
                    attributes: ['name'],
                },
                {
                    model: Image,
                    attributes: ['string'],
                    limit: 1,
                },
                {
                    model: Discount,
                    as: 'discount',
                    attributes: ['value'],
                }
            ],
            attributes: {
                exclude: ['category_id', 'discount_id']
            },
            limit,
        })
    } catch (e) {
        return Promise.reject("Server error")
    }
}
```

5

```
const SortFields: { [key: string]: string } = {
    "time": "time_created",
    "price": 'priceRegular',
    'name': "name",
}

export const get = async (req: Request, res: Response) => {
    try {
        const {
            category = req.baseUrl.slice(1),
            query = '',
            page = 0,
            limit = 10,
            sortBy,
            desc = false
        } = req.query
        const offset = Math.ceil(+page * +limit)
        const sort: { field?: string, isDESC?: boolean } = {}

        if (!category) {
            return res.sendStatus(400)
        }

        if (sortBy) {
            sort.field = SortFields[sortBy.toString()]
            sort.isDESC = (desc === 'true')
        }

        const products = await ProductService.get({
            category: category.toString(),
            limit: +limit,
```

```
            offset,
            sort,
            filters: {
                query: query.toString()
            }
        })

        if (products) {
            const paginationNav = {
                nextPage: Math.floor(offset/+limit) + 1,
                prevPage: Math.floor(offset/+limit) - 1 >= 0 ? Math.floor(offset/+limit)
- 1 : null
            }
            const resBody: { [key: string]: any } = {
                total: products.count,
                onPage: products.rows.length,
                nextPage: null,
                prevPage: null,
                data: products.rows,
            }

            if (products.rows.length >= +limit) {
                const nextSearchParams = new URLSearchParams()

                nextSearchParams.append('limit', limit.toString())
                nextSearchParams.append('page', paginationNav.nextPage.toString())
                if (query) {
                    nextSearchParams.append('query', query.toString())
                }

                resBody.nextPage = `/${category}?${nextSearchParams.toString()}`
            }

            if (paginationNav.prevPage !== null) {
                const prevSearchParams = new URLSearchParams()

                prevSearchParams.append('limit', limit.toString())
                if (paginationNav.prevPage > 0) {
                    prevSearchParams.append('page', paginationNav.prevPage.toString())
                }
                if (query) {
                    prevSearchParams.append('query', query.toString())
                }

                resBody.prevPage  = `/${category}?${prevSearchParams.toString()}`
            }

            return res.send(resBody)
        }
    }
    catch (e) {
        return res.sendStatus(500)
    }
}

export const getByNamespaceId = async (req: Request, res: Response) => {
    try {
        const { namespaceId } = req.params
        const product = await ProductService.getByNamespaceId({ namespaceId })

        if (product) {
            return res.send(product)
        }
```

```
        return res.sendStatus(404)
    } catch (e) {
        return res.sendStatus(500)
    }
}

export const getSpecials = (specialsType: 'latest' | 'discount') => {
    return async (req: Request, res: Response) => {
        try {
            const filter: { byDiscount?: boolean, byDate?: boolean } = {}
            const {
                category,
                limit = 10
            } = req.query

            if (!category) {
                return res.sendStatus(400)
            }

            switch (specialsType) {
                case "discount":
                    filter.byDiscount = true
                    break
                case "latest":
                    filter.byDate = true
            }

            const products = await ProductService.get({
                category: category.toString(),
                limit: +limit,
                filters: filter
            })

            return res.send(products)
        } catch (e) {
            return res.sendStatus(500)
        }
    }
}

export const getRecommended = async (req: Request, res: Response) => {
    try {
        const { namespaceId } = req.params
        const { limit = 10 } = req.query
        const product = await ProductService.getByNamespaceId({ namespaceId })

        if (product) {
            const recommended = await ProductService.getRecommended({
                basename: product.basename,
                limit: +limit
            })

            return res.send(recommended)
        }

        return res.sendStatus(404)
    } catch (e) {
        return res.sendStatus(500)
    }
}
```

6

```
const router = express.Router();

router.use(express.json());

router.get('/', get);

router.get('/:namespaceId', getByNamespaceId);

router.get('/:namespaceId/recommended/', getRecommended);

router.get('/specials/latest', getSpecials('latest'));

router.get('/specials/hot-price', getSpecials('discount'));

export default router
```

7

```
export const createOrder = createAsyncThunk(
  "cart/createOrder",
  async (cart: CartState) => {
    const products = cart.items.map(({ item, count }) => {
      return {
        id: item.id,
        count,
      };
    });

    try {
      const order = await OrderClient.createOrder({
        products,
        user_id: "924d1d37-f338-4865-988d-266dff0f3c1d"
      }) as { id: string, owner_id: string };
      return order.id;
    } catch (e) {
      console.log(e);
      return Promise.reject("");
    }
  }
);

export const cart = createSlice({
  name: "cart",
  initialState: getInitialState<CartState>(
    "cart",
    { items: [], sum: 0, order: { id: null, status: ActionState.Idle } }
  ),
  reducers: {
    add: (state, action: PayloadAction<{ item?: Product | ProductShorted, id?: string
}>) => {
      const { item, id } = action.payload;

      if (item) {
        state.items.push({ item, count: 1 });
        state.sum += item.priceDiscount
```

```
          ? item.priceDiscount
          : item.priceRegular;

        saveToStorage(cart.name, state);
        return;
      }

      if (id) {
        const position = state.items.find(({ item }) => item.id === action.payload.id);

        if (position) {
          position.count += 1;
          state.sum += position.item.priceDiscount
            ? position.item.priceDiscount
            : position.item.priceRegular;
        }

        saveToStorage(cart.name, state);
        return;
      }
    },
    remove: (state, action: PayloadAction<{ id: string }>) => {
      const position = state.items.find(({ item }) => item.id === action.payload.id);

      if (position) {
        position.count -= 1;
        state.sum -= position.item.priceDiscount
          ? position.item.priceDiscount
          : position.item.priceRegular;
      }

      saveToStorage(cart.name, state);
    },
  },
  extraReducers: (builder) => {
    builder.addCase(createOrder.pending, (state) => {
      state.order.status = ActionState.Loading;
    });
    builder.addCase(createOrder.rejected, (state) => {
      state.order.status = ActionState.Failed;
    });
    builder.addCase(createOrder.fulfilled, (state, action: PayloadAction<string>) => {
      state.order.status = ActionState.Idle;
      state.order.id = action.payload;
      state.sum = 0;
      state.items = [];
    });
  }
});

export const cartSelector = (state: RootState) => state.cart;
export const { add, remove, removeAll } = cart.actions;

export default cart.reducer;
```

Sun Jun 16 21:38:04 EEST 2024, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 130862 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв Додано в БД: 2024-06-16 Автора: Кирило МУКОМЕЛА Керівники: Руслан БАГРІЙ Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	73417	1082	3587 (5%)	45 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КН

ID перевірки:
1016366171

Дата перевірки:
16.06.2024 23:05:19 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
16.06.2024 23:20:10 EEST

ID користувача:
100005671

Назва документа: **КН-20-2 Мукомела_ЗАПИСКА**

Кількість сторінок: **73** **Кількість слів:** **12834** **Кількість символів:** **100526** **Розмір файлу:** **1.54 MB** **ID файлу:** **1016172377**

5.55%
Схожість

Найбільша схожість: **2.59% з джерелом з Бібліотеки** (ID файлу: **1016162677**)

3.47% Джерела з Інтернету	464	Сторінка 75
3.76% Джерела з Бібліотеки	102	Сторінка 77

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості.

Назва: *Рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв*

Автор: *студент гр. КН-20-2 Кирило Мукомела*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *к.т.н., доц. Руслан Багрій*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

1) за системою Anti-Plagiarism виявлені 3% є фрагментарними – містять поширені конструкції, загальновідомі терміни, скорочення та визначення.
2) За системою UNICHECK виявлені 5,55%, що є запозиченнями, які розміщені в розділах аналізу існуючих технологій та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 3% і 5.55% відповідно, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Руслан БАГРІЙ

Гарант ОП _____ Олександр МАЗУРЕЦЬ

Завідувач кафедри КН _____ Олександр БАРМАК

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-2 Мукомели Кирила Олеговича*

за темою *Рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв*

1. Актуальність теми

Актуальність теми достатньо обґрунтована, оскільки зростаюча кількість мобільних пристроїв та користувачів вимагає персоналізованого підходу до контенту, що сприяє покращенню користувацького досвіду та задоволеності. Системи рекомендаційного характеру на основі контентного фільтрування дозволяють пропонувати користувачам саме ті продукти, послуги або інформацію, які відповідають їхнім індивідуальним вподобанням та потребам, що, в свою чергу, підвищує ефективність взаємодії з мобільними вебсистемами.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

Тема кваліфікаційної роботи "Рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв" відповідає предметній області спеціальності 122 Комп'ютерні науки та вимогам до кваліфікаційної роботи бакалавра, оскільки результатом роботи є рекомендаційний метод на основі контентного фільтрування, що аналізує доступні мобільні пристрої та їх характеристики та надає персоналізовані рекомендації на основі вподобань користувачів. Метод використовує інформацію про властивості мобільних пристроїв та порівнює її з уподобаннями користувачів, створюючи профілі, які відображають їхні інтереси та попередні вибори. При вирішенні поставленої задачі використано методи збору та аналізу інформації, методи контентного фільтрування, технології та методи проектування вебсистеми мобільних пристроїв.

3. Професійні та особистісні якості бакалавра

Мукомела К.О. під час роботи над кваліфікаційною роботою бакалавра продемонстрував розуміння теорії та практичних аспектів використання рекомендаційних методів на основі контентного фільтрування для формування персоналізованих рекомендацій. Він успішно реалізував вебсистему мобільних пристроїв,

яка інтегрує рекомендаційний метод, що значно спрощує навігацію для користувачів. Таким чином, Мукомела К.О. продемонстрував відмінні навички роботи з сучасними технологіями програмування та здатність застосовувати їх на практиці.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Робота виконана самостійно, академічного плагіату не виявлено, стосовно всіх запозичень наведено відповідні посилання на джерела.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показала високий рівень компетентності та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи повністю розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, виконані усі поставлені задачі та розроблено програмну реалізація для підтвердження запропонованого методу.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Викладення матеріалу логічне, послідовне та аргументоване. Мова і стиль викладення кваліфікаційної роботи відповідають стандартам, що забезпечує доступність сприймання матеріалу і відповідає вимогам до сучасних наукових робіт.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблена у роботі вебсистема з рекомендаціями мобільних пристроїв має значний потенціал для практичного застосування як цілісного продукту, так і окремих її компонентів. Впровадження такої системи сприяє підвищенню користувацького досвіду, що, в свою чергу, може позитивно вплинути на залучення та утримання користувачів, а також на комерційні показники вебплатформ, зокрема онлайн-магазинів.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка ***«відмінно»*** *.*

Керівник _____ *к.т.н., доц. Руслан БАЙРІЙ*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ
на кваліфікаційну роботу бакалавра

студента *гр. КН-20-2 Мукомели Кирила Олеговича*
за темою: *Рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв*

1. Актуальність обраної теми

В умовах зростаючої популярності вебсистем на мобільних пристроях, важливість контентного фільтрування значно зростає. Користувачі мобільних пристроїв мають обмежений час і ресурси, тому для них особливо важливо отримувати персоналізований контент, який відповідає їхнім інтересам. Тому дослідження в цій сфері є надзвичайно актуальним, оскільки воно сприяє покращенню користувацького досвіду та ефективності взаємодії з технологіями.

2. Повнота розкриття мети та завдань роботи

Під час виконання кваліфікаційної роботи бакалавра було реалізовано рекомендаційний метод на основі контентного фільтрування для вебсистеми мобільних пристроїв, що відповідає меті та завданням кваліфікаційної роботи і розкриває їх повною мірою.

3. Зміст кожного розділу роботи

Записка кваліфікаційної роботи складається з трьох розділів. У першому розділі проводиться аналіз предметної області, включаючи аналіз інформаційних моделей, огляд систем та визначення мети і завдань. Другий розділ присвячений дослідженню рекомендаційного методу на основі контентного фільтрування, з описом загальної системи що імплементує досліджений метод та оглядом датасету. Третій розділ акцентує увагу на виборі засобів розробки, описі програмної реалізації рекомендаційної системи підбору телесеріалів та проведенні тестування.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблена система, що імплементує рекомендаційний метод на основі контентного фільтрування мобільних пристроїв надає інструменти, які допомагають користувачам ефективно знаходити контент, що відповідає їхнім вподобанням у середовищі надлишкової інформації.

5. Якість оформлення кваліфікаційної роботи бакалавра

Записка якісно оформлена, відповідно до встановлених вимог. Вірно та зрозуміло написана, з виразною структурою і логічною послідовністю представлення матеріалу.

6. Недоліки кваліфікаційної роботи бакалавра

Рекомендовано розглянути можливість поєднання контентного фільтрування з колаборативним фільтруванням, включення більшої кількості джерел даних та розгортання розробленої вебсистеми на окремому хостингу.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка відмінно .

Рецензент к.т.н., доцент каф. АКІТ та Р Кореєвська І.О.