Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Програмна реалізація алгоритму Apriori рекомендаційної системи товарів

Виконав: студент IV курсу, групи СНс-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Касіян М. І. (прізвище та ініціали)

Керівник (підпис) Матійчук Л. П. (прізвище та ініціали)

Нормоконтроль (підпис) Липак Г. І. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Тимощук Д. І. (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« 8 » _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Касіяну Максиму Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна реалізація алгоритму Apriori рекомендаційної системи товарів

Керівник роботи Матійчук Любомир Павлович, доктор екон. наук, професор кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *14* » *травня* 2026 року № *4/9-237*

2. Термін подання студентом завершеної роботи 22 червня 2026 р.

3. Вихідні дані до роботи Літературні джерела про методології добування даних, побудови рекомендацій, розробки програмних систем та інтерфейсів

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз рекомендаційних систем та постановка завдання. 1.1. Аналіз предметної області рекомендаційної системи товарів. 1.2. Визначення вимог для рекомендаційної системи товарів 1.3. Визначення можливих варіантів використання рекомендаційної системи товарів, розробка діаграми прецедентів. 1.4. Аналіз існуючих рішень. 2. Проєктування рекомендаційної системи товарів з алгоритмом Apriori. 2.1. Проєктування архітектури. 2.2. Проєктування Функціоналу. 2.3. Визначення інтерфейсів. 3. Програмна реалізація рекомендаційної системи товарів. 3.1. Програмна реалізація компонентів. 3.2. Тестування та експлуатація рекомендаційної системи товарів. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік використаних джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка 2. Поточні тенденції щодо машинного навчання 3. Добування рекомендацій, або Data Mining 4. Алгоритми добування даних 5. Apriori алгоритм 6. Комбінаційні кандидати 7. Стратегія розподілу 8. Колаборативні фільтри та персоналізація 9. Програмні технології 10. Питання адаптації. 11. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Мариненко С. Ю., к. т. н., доцент кафедри МТ	03.06.2026	08.06.2026

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	27.01.2026-29.01.2026	*Виконано*
3.	Виконання дослідження рекомендаційних систем	30.01.2026-18.05.2026	*Виконано*
	Розроблення технічного документу, програмна реалі-		
	зація рекомендаційної системи товарів		
4.	Оформлення розділу «Аналіз рекомендаційних сис-	18.05.2026-23.06.2026	*Виконано*
	тем та постановка завдання. Аналіз правил кошика»		
5.	Оформлення розділу «Проєктування рекомендаційної	24.05.2026-31.06.2026	*Виконано*
	системи товарів з алгоритмом Apriori»		
6.	Оформлення розділу «Програмна реалізація рекомен-	01.05.2026-02.06.2026	*Виконано*
	даційної системи товарів»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	03.06.2026-08.06.2026	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	03.06.2026-08.06.2026	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	08.06.2026-10.06.2026	*Виконано*
10.	Нормоконтроль	11.06.2026-14.06.2026	
11.	Перевірка на плагіат	17.06.2026	
12.	Попередній захист кваліфікаційної роботи	18.06.2026	
13.	Захист кваліфікаційної роботи	26.06.2026	

Студент _____ Касіян М. І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Програмна реалізація алгоритму Apriori рекомендаційної системи товарів // Кваліфікаційна робота освітнього рівня «Бакалавр» // Касіян Максим Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-41 // Тернопіль, 2026 // С. 65 , рис. – 16 , табл. – 4 , бібліогр. – 43 .

Ключові слова: рекомендаційна система, реалізація алгоритмів, машинне навчання, аналіз даних, програмна мова rust, сервісно орієнтована архітектура, великі дані

В кваліфікаційній роботі реалізовано дослідження рекомендаційних систем та їх алгоритмів. Перший розділ кваліфікаційної роботи розглядає сферу аналізу даних та машинного навчання, визначаючи прецеденти використання для рекомендаційної системи товарів. Після аналізу переваг і недоліків інших алгоритмі добування даних було висвітлено вимоги до рекомендаційної системи товарів, включаючи використання алгоритму Apriori.

В другому розділі проєктування розглянуто процес алгоритму Apriori, досліджуючи його недоліки, визначаючи необхідні змін в алгоритмі для їх вирішення. Додатково на основі характеристик даних було визначено сервісну програмну архітектуру рекомендаційної системи та методи взаємодії з нею.

В третьому розділі описано програмну реалізацію рекомендаційної системи товарів. Використовуючи програмну мову Rust для реалізації, в розділі проаналізовано кожен крок роботи реалізованих функцій. В додаток до тестування внутрішніх алгоритмів, також проведено тестування інтерфейсів інтеграції та демонстрація експлуатації рекомендаційної системи товарів.

ANNOTATION

Software Implementation of the Apriori Algorithm for a Product Recommendation System // Qualification work of the educational level «Bachelor» // Kasiyan Maksym // Ternopil Ivan Puluy National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-41 // Ternopil, 2026 // P. 65 , fig. – 16, tabl. – 4, references – 43 .

Keywords: recommendation systems, algorithm implementation, machine learning, data mining, rust programming language, service oriented architecture, big data

The qualification work is dedicated to researching recommendation systems and the algorithms used in it. The first chapter examines the fields of data mining (including market basket analysis) and machine learning, identifying use cases of a product recommendation system. After analyzing the weaknesses and strengths of other data mining algorithms, the functional requirements for a product recommendation system were set, including usage of the Apriori algorithm.

The goal of the work is to implement the Apriori algorithm in a product recommendation system; in second chapter examines the algorithm, analyzing its weaknesses, defining necessary changes to the algorithm to mitigate these flaws. In addition, based on characteristics of data a service software architecture was decided for the product recommendation system, as well as the means of interacting with it.

The third chapter covers the software implementation of product recommendation system and its components. Using the Rust programming language for implementation, the chapter contains thorough analysis of each process in implemented functions. In addition to software testing of internal algorithms, an external interface testing was carried out, as well as demonstration of using the product recommendation system.

ЗМІСТ

ВСТУП

Після вибуху генеративного штучного інтелекту розробки у всіх сферах причетних з інформаційними технологіями (та не дуже причетних, як холодильники) насамперед передбачають інтеграцію мовної моделі в будь якому вигляді. Галереї, планування маршрутів, медицина та навіть аналіз даних тепер мають функції штучного інтелекту у формі «розмови», який внутрішньо є зв'язаний рядами функцій, під-запитами та інтерпретаторами.

На довгій перспективі, для більшості з цих сфер використання такого виду штучного інтелекту призведе до проблем. На ближнє майбутнє вирішення всіх задач умовним «молотком» мовного штучного інтелекту привертає увагу як технологічний прогрес; але пов'язані проблеми ресурсоємності, інтеграції та покращення такої загальної моделі призводить до особливо значних затрат в часі та ресурсах. Це відчувається навіть мегакорпораціями, як Google та Microsoft, які поступово зменшують доступ до моделей, натомість переходячи до окремої оплати за час обробки [39], з більшим проміжком часу між кожною новою моделлю.

Зазвичай краще мати правильний інструмент для задачі. Інші види машинного навчання, як класифікаційні, можуть надати необхідні результати в рази швидше та більш зрозумілому форматі. Одним з таких випадків є аналіз даних для магазинів, де продавцям необхідно визначити як товари є пов'язані між собою в очах покупців; що також є корисно покупцеві, який більш імовірно купить пару товарів, побачивши їх в рекомендаціях як «часто куплено разом».

Такий вид аналізу найкраще виконується використовуючи правила асоціації, фундаментальним для якого є алгоритм Apriori. Хоча старий, алгоритм Aprіori є досі ефективним та надійним інструментом, що може розглядати та знаходити асоціації в масштабних масивах даних, як кошиках та транзакціях клієнтів. Також оскільки вона є спеціалізованою під це завдання, на відмінно від мовних моделей які фундаментально розглядають усі дані як ймовірнісні жетони тексту а не цілу інформацію, вона легко піддається інтеграції та покращенню.

При цьому зазнаючи менше ресурсів, що є особливо важливим в поточній кризі щодо постачання оперативної пам’яті та обчислювальних частин загалом, ці характеристики роблять алгоритм Apriori особливо актуальним для розробки.

Згідно з цим, для кваліфікаційної роботи освітнього рівня «Бакалавр» метою є програмна реалізація алгоритму Apriori, що передбачає проєктування та розробку алгоритму як системи рекомендації товарів. Для досягнення цієї мети в кваліфікаційній роботі передбачається виконання наступних завдань:

1) розглянути предметну область рекомендаційних систем, аналізуючи практичні застосування отриманих даних та принцип роботи алгоритмів які використовуються в рекомендаційних системах товарів;

2) дослідити механізм обробки даних за алгоритмом Apriori, його логіку, характеристики (переваги та недоліки), методи інтерпретації результатів;

3) проаналізувати та розробити технічне завдання, передбачити архітектуру, функціонал, програмний інтерфейс для рекомендаційної системи товарів;

4) реалізувати розроблене технічне завдання рекомендаційної системи товарів з алгоритмом Apriori, використовуючи оптимальні технології для покращення ефективності роботи;

5) провести тестування програмної реалізації рекомендаційної системи товарів на різних (близьких до реальності) даних, оцінюючи ефективність та якість результатів;

6) здійснити аналіз охорони прав, визначаючи вплив комп’ютерної системи та рекомендаційної системи товарів на життєдіяльність людини.

В результаті кваліфікаційної роботи, розроблена система рекомендації товарів за алгоритмом Apriori може бути впровадженою для інтеграції в інтернет магазини. З кращою якістю добутих даних, її використання дозволяє до більш ефективного продажу товарів, при цьому не порушуючи набутий досвід користувачів взаємодії з магазином.

РОЗДІЛ 1 АНАЛІЗ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ. АНАЛІЗ ПРАВИЛ КОШИКА

Рекомендаційні системи будуються на основі процесу добування даних, що насамперед є процесом знаходження та відповідно інтерпретації збіжностей в масивних наборах даних. У даному розділі процес цього добування розглядається як в загальних термінах, для зрозуміння процесу та цілей, так і методи його застосування, як в сфері продажу товарів де воно застосовується в аналізі кошику ринку.

1.1 Аналіз предметної області рекомендаційної системи товарів

Рекомендаційні системи насамперед є системою фільтрування даних, на якому з загальної інформації виводиться лише те, що може бути причетним для певного товару або користувача [24]. Одним з найбільш відомих прикладів такої системи є загальні алгоритми в соціальних мережах, які також підбирають користувачеві що йому бажано побачити.

1.1.1 Спрощення прийняття рішень для користувача застосовуючи механізм рекомендацій товарів

Основною ціллю рекомендаційної системи є спрощення та прискорення процесу прийняття рішення від користувача; замість самостійного обирання та пошуку інформації, вона рекомендується їм прямо. Сама інформація вже залежить від сфери у якій працює ця система рекомендації, з кінцевою метрикою більшого використання тої інформації. В соціальних мережах це більший час прокручування, а для магазинів – купівля більшої кількості товарів.

Для комерції це є особливо необхідним у зв’язку з можливим масштабним набором товарів, що може призвести в користувача умовний параліч вибору розглядаючи та порівнюючи різниці між кожним підвидом певного товару. В

такому середовищі, з метою збільшення прибутку рекомендаційна система товарів дозволяє отримати наступні результати [36]:

- пропонуючи єдиний дорожчий товар, користувач більш ймовірно візьме його замість дешевших альтернатив;
- знаходячи більш специфічні товари пов’язані з головним товаром, користувач може купити додаткові предмети, які в іншому випадку без рекомендаційної системи товарів не могли-б знайти;
- на довгостроковій перспективі рекомендації набудуть «довіри» в користувача, що відповідно підтримає їх лояльність до магазину.

В залежності від прибутковості алгоритму, комерційні сайти можуть відобразити рекомендовані товари як частина «комплекту», що рекомендує його зі знижкою, як зображено на рисунку 1.1.

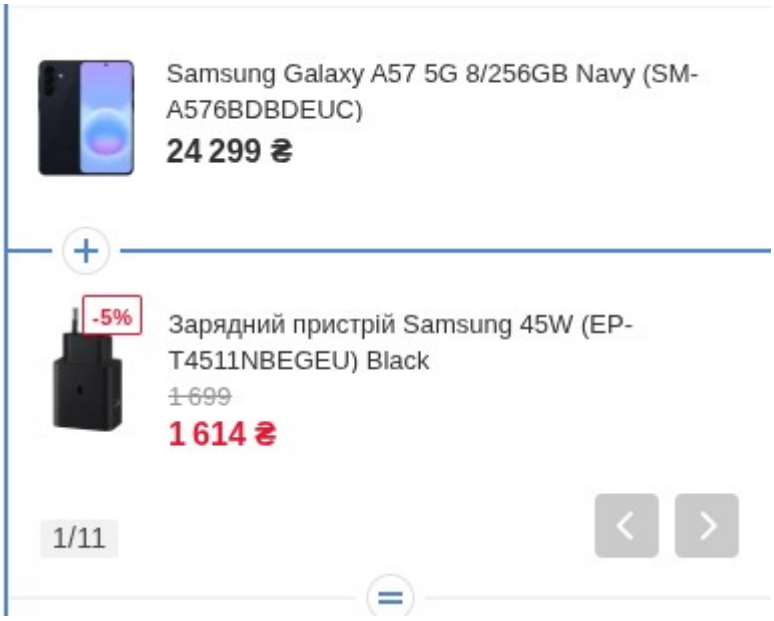


Рисунок 1.1 – Рекомендація зарядки як частина комплекту

Іншими видами відображення цих рекомендацій включає загальну стрічку, яка відображає різні товари які мають високу впевненість в асоціації, або рекомендації під час пошуку. При цьому, такі види інтеграцій потребують набагато більше персональних даних, включаючи попередні перегляди, та пошукові запити для створення асоціацій товарів до різних ключових слів. [33]

1.1.2 Добування даних для побудови персоналізованих та загальних рекомендацій товарів

Для того, щоб рекомендаційна система товарів могла знайти правильний товар для рекомендації, необхідні дані взаємодії з товарами. Процес знаходження цих даних є другою частиною в роботі рекомендаційних систем товарів, ціль якого є створення правил асоціації одного товару до іншого.

Цей принцип є більш відомим як «data mining», або добування даних. Не плутаючись з іншим видом «data mining»-у (де дані знаходяться з інших джерел в умовно сирому варіанті), процес добування використовує алгоритми для обробки набору даних, як наприклад транзакції та перегляди товарів, в статистично корисну інформацію [19]. Згідно назви, головною метою цього процесу є добування даних включаючи кількісні та якісні характеристики, правила асоціації між товарами, частотність їх покупок.

Сам процес аналізу даних можливо розкласти на п’ять кроків та шість станів даних, зображених на рисунку 1.2:

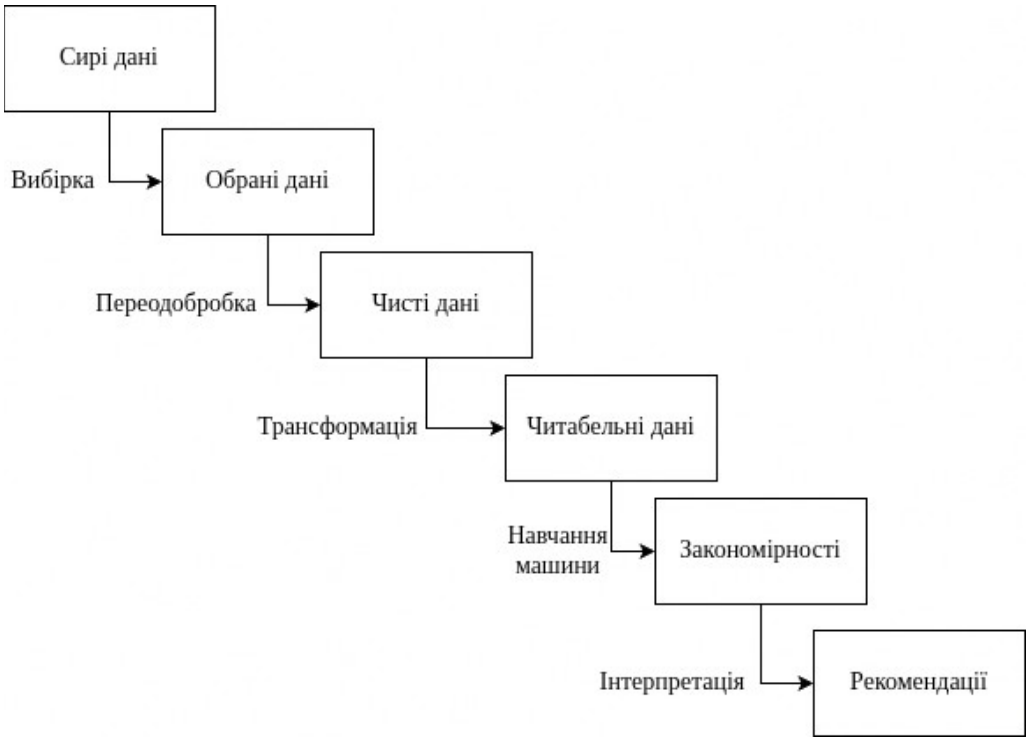


Рисунок 1.2 – Візуалізація процесу добування даних

1) вибірка, де з сирих/усіх даних обирається лише необхідні фрагменти для проведення аналізу;

2) перед-обробка, у якому обрані дані перевіряються на належність включаючи правильність типу даних, форматування тексту, та кількісні характеристики;

3) трансформація, де оброблені дані перетворюються в належний формат для навчання машини;

4) навчання машини, в кінцевому результаті якого в наданих даних виявляються певні закономірності та якісні характеристики;

5) інтерпретація цих закономірностей, включаючи застосування в рекомендаціях.

Інтерпретація та добування даних включає в себе різні задачі, включаючи виявлення аномалій (наприклад, надзвичайна популярність певних товарів), групування товарів по подібних характеристиках, класифікація нових даних, знаходження закономірностей [2] як і для створення парних товарів, так і для побудови функції що відповідає цим характеристикам (тим самим визначаючи ймовірність ефективності рекомендації певного товару), та підсумування для спрощення набору даних в певні загальні якості.

Процес добування даних має характеристики динамічності. Оскільки асортимент товарів доступних для продажу постійно змінюються, і через що інформація для більш популярних товарів можуть вже бути не актуальними на період їх знаходження. У зв’язку з цим рекомендаційні системи стикаються з декількома проблемами пов’язаними з машинним навчанням цих систем.

Перша з них є «занос концепцій» [42]. Оскільки навчання інколи може зайняти довгий час для визначення нових рекомендацій, в достатньо складних проміжках через зміни сезонів або трендів її результати стають неактуальними.

Інша проблема є пов’язана з початком використання рекомендаційної системи товарів, звана «холодним початком» [12]. Вона виникає у двох випадках: коли в системі реєструється новий користувач і база даних ще не містить історії його транзакцій для формування персоналізованої видачі; або коли в каталог

магазину додається цілком новий товар, з яким ще не відбувалося жодних взаємодій, через що системі бракує статистичних даних для здобуття певних асоціацій або правил для неї. До моменту набирання достатньої кількості інформації рекомендаційні системи мають тимчасово спиратися на більш загальні рекомендації для товару.

1.1.3 Аналіз кошику ринку та побудування правил асоціації для рекомендаційної системи товарів

Аналіз кошику ринку є підвидом добування даних який приділяє більшу увагу до задач закономірностей в наборі даних. Виділення цих закономірностей відбувається за їх одночасною появою в наборі даних, та в двох етапах. Перший етап є простим знаходженням усіх даних які часто з'являються в масиві даних, а у другому етапі з цих даних визначаються найважливіші, в залежності наскільки вони задовільняють певні правила асоціацій.

В сфері продажі товарів аналіз кошику ринку дозволяє якісно зобразити поведінки покупців, які беруть один чи інший продукт. В додаток до цього, такий аналіз автоматично також розкладає товари на категорії, які потім продавець може застосувати в процесі розставлення їх в магазині (наприклад, молочні продукти поблизу яєць) або під час виставлення знижок на товари, для збільшення продажів одного продукту застосовуючи знижку на іншому.

На рисунку 1.3 зображено графічний приклад цілого механізму. В прикладному магазині що продає товари A, B та C зводиться перелік транзакцій, зображених зліва. Під час добування даних в залежності від загальної кількості отриманих транзакцій система виводить припущення асоціацій товарів A-B та C-A,B. На етапі виведення рекомендацій, оскільки перша асоціація має більшу правдоподібність, лише для товару A відображається рекомендація товару B.

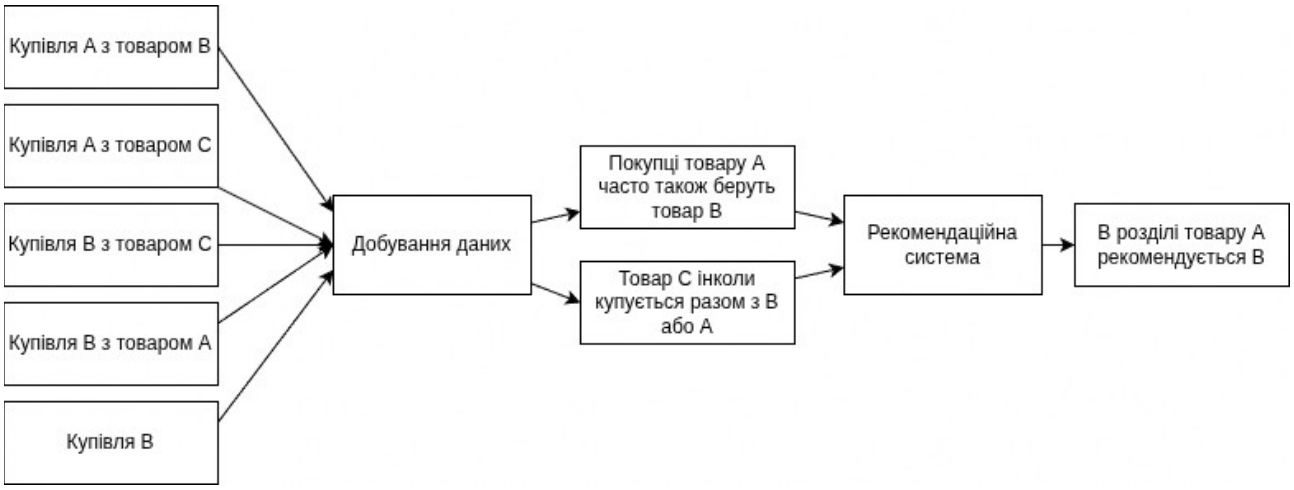


Рисунок 1.3 – Візуалізація процесу аналізу та рекомендації товарів

При цьому рекомендація товару С не здійснюється, хоча обидва товари є присутніми в кошику; це пов’язано з відсутністю їх асоціацій водночас з більшою кількістю, тому товар може бути подібним до А або В і тим самим не ефективним для рекомендації.

Інші сфери які застосовують даний метод аналізу включає медицину, де парна поява симптомів може сигналити про появу одної чи іншого захворювання.

1.2 Визначення можливих варіантів використання рекомендаційної системи товарів, розробка діаграми прецедентів

Для ефективного проектування функціоналу, в додаток до функціональних вимог необхідно визначити ким і для чого відбувається взаємодія з рекомендаційною системою товарів. Через різноманітність введеної інформації та виведених рекомендацій та статистик, до категорій користувачів як клієнтів та адміністраторів припадають маркетологи, які можуть координувати продажі певних товарів та аналізувати їх ефективність. У таблиці 1.1 описано різні функціонали, які припадають для різних ролів користувачів для рекомендаційної системи товарів.

Таблиця 1.1 – Варіанти використання рекомендаційної системи товарів

Роль користувача	**Функція**	**Опис**
Клієнт	Надання рекомендацій	Виводить рекомендовані товари при перегляді певного іншого товару
	Персоналізація пропозицій	Використовуючи інформацію про користувача, виводить різноманітні товари які особисто рекомендуються їм
	Доповнення даних	При покупці, додає їх транзакцію до системи для навчання рекомендацій
Маркетолог	Перегляд статистик	Виводить поточні на момент перегляду інформацію щодо найпопулярніших товарів
	Зміна чорного списку	Дозволяє змінювати які товари НЕ враховуватимуться для рекомендації.
	Експорт інформації	Зберігає поточну статистику у читабельний текстовий файл для подальшого аналізу
Адміністратор	Перезавантаження	Дозволяє виконати перенавчання, зчитуючи нові налаштування
	Доступ до звітів	У випадку проблем з рекомендаційною системою, проблеми виводяться у звітовий файл для діагностики

Згідно цих даних можливо побудувати діаграму прецедентів. Діаграма прецедентів дозволяє описати як методи використання розширюються у функціоналі від інших методів або передбачають його використання (наприклад, здійснення транзакцій обов’язково включає завантаження її інформації для

подальшого навчання рекомендацій) [11]. Діаграма прецедентів є зображеною на рисунку 1.4.

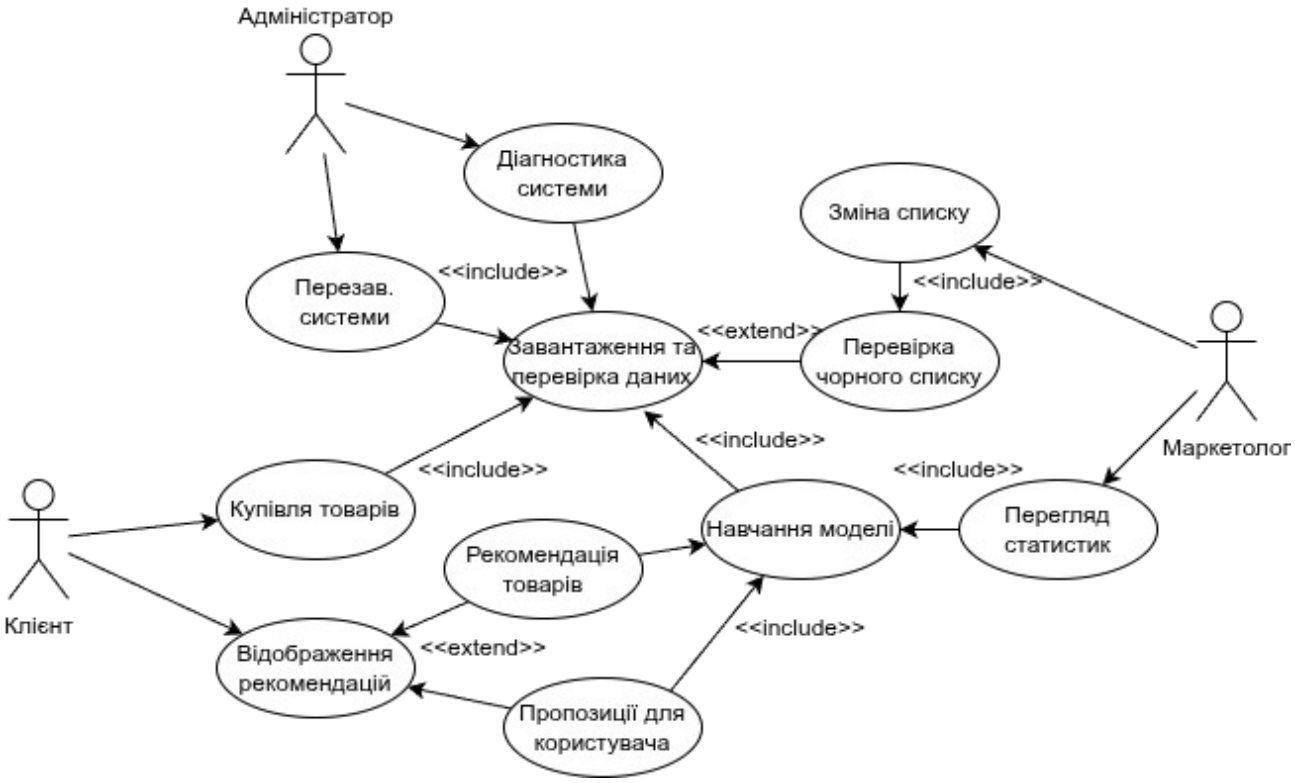


Рисунок 1.4 – Діаграма прецедентів користувачів системи

«Клієнт» є умовно головною групою користувачів, оскільки вони є єдиними хто безпосередньо заповнюють дані транзакцій для навчання рекомендаційної системи товарів. Основною процедурою для них є відображення рекомендацій, які в залежності від де саме вони знаходяться розширюються та надають рекомендації для певного товару або пропозиції на основі даних користувача, які передбачають навчання моделі для отримання рекомендацій. Під час купівлі товарів, клієнт створює транзакцію; відповідно цей процес завжди включає в себе перевірку правильності інформації та завантажується в базу даних для навчання рекомендацій товару та користувача.

«Маркетолог» має подібні можливості, але з більш глобальним доступом над цілою системою. Процедура «Перегляд статистик» також передбачає навчену модель, але на відміну від ізольованого користувача статистика відображатиме якнайбільше більше корисної інформації в загалом. Звідси

маркетолог може тоді зробити знімок даних (описаний у таблиці як «експорт») для подальшого аналізу після оновлення даних. Інший функціонал що передбачається є зміна чорного списку; оскільки цей список є чутливим для правильної працездатності цілої моделі, воно обов’язково включає в себе процес перевірки цілісності цього списку.

Група адміністраторів має спеціалізовані права та процедури, фокусуючись лише на працездатності системи а не її результатах. Перезавантаження рекомендаційної системи передбачає перенавчання моделі, що відповідно включає завантаження та перевірки цілої бази даних, а також перевіряючи чорний список для захисту від змін поза призначеними редакторами. Діагностика системи є простою, рекомендаційна система товарів буде надавати діагностичну інформацію включаючи помилки та час обробки у власних файлах-звітах про роботу.

1.3 Аналіз існуючих рішень для системи рекомендацій товарів

Оскільки максимізація уваги клієнтів є актуальним питанням в сфері комерції та соціальних мереж, рекомендаційні системи мають постійний прогрес в експериментуванні та покращенні кінцевих рекомендацій з метою персоналізації цілого процесу використання сервісу користувачем.

Алгоритм FP-Growth (від Frequent Pattern Growth) часто наводять як головного конкурента для алгоритму Apriori [18], який вирішує проблему багаторазового сканування бази даних, що є притаманним для Apriori. Замість ітеративного тестування кандидатів, даний алгоритм використовує структуру дерева зображеного на рисунку 1.5 для стиснення цілої бази даних транзакцій.

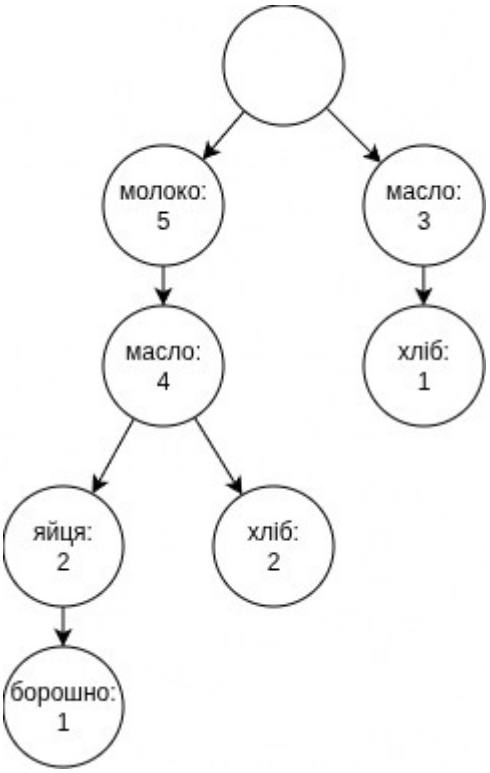


Рисунок 1.5 – Дерево алгоритму FP-Growth

Хоча FP-Growth вимагає лише два проходи по наборі транзакцій з бази даних, побудова та зберігання самого дерева в оперативній пам'яті займає набагато більше ресурсів, особливо якщо дані є розрідженими або містять велику кількість унікальних елементів. В таких умовах структура дерева стає дуже розгалуженою, відповідно споживаючи значну кількість пам'яті, що є небажаним в поточних умовах доступності пам’яті. В цей час алгоритм Apriori не приховує дані у складних структурах; а також поступовий підхід до підготовки даних дозволятиме керувати використанням пам'яті на кожному кроці, а сам процес є простішим для розуміння для підтримки стабільності.

Другий алгоритм що конкурує в сфері рекомендаційних систем є ECLAT, або Equivalence Class Transformation. На відмінно від попередніх алгоритмів, цей алгоритм обертає структуру даних зверху вниз; замість використання більш інтуїтивного формату асоціацій як ідентифікатор транзакції до списку куплених товарів в ньому, ECLAT перевертає базу даних у зворотній формат, де єдиний товар асоціюється зі списком транзакцій, у яких він зустрічається [29]. Ці дві структури даних є описаними у таблицях 1.2 та 1.3.

Таблиця 1.2 – Вертикальна структура даних

№ Тр.	Молоко	Масло	Яйця	Хліб	Борошно
1	+	+	-	-	+
2	+	+	+	+	-
3	+	-	+	+	-
4	-	+	-	-	+
5	-	+	+	+	-

Іншою назвою для таких структур даних є горизонтальними та вертикальними. Походячи з принципів будування бази даних, горизонтальна структура (як на таблиці 1.2) виділяє рядки-транзакції як сутність, в яку входять атрибути-кошик. У вертикальній відповідно здійснюється зворотній процес, де можливий вміст кошика є сутністю, яка перелічує як атрибути всі транзакції, в яких вона була.

Таблиця 1.3 – Горизонтальна структура даних

Товар	Перелік транзакцій
Молоко	1, 2, 3
Масло	1, 2, 4, 5
Яйця	2, 3, 5
Хліб	2, 3, 5
Борошно	1, 4

Як недолік алгоритм зустрічається з аналогічною проблемою пам’яті, оскільки обчислення впевненості здійснюється через перетин множин ідентифікаторів транзакцій, використовуючи пошук у глибину. У зв’язку з цим у

великих наборах даних глибинний пошук має рекурсивний характер, в якого необхідний обсяг пам’яті та ресурсів є непередбачуваним.

Останній алгоритм для аналізу є факторизація матриць. Це є більш подібним до популярних систем машинного навчання, де замість простого знаходження закономірностей алгоритм намагається інтерпретувати дані з невидимими просторами (як особистими схильностями користувача) та спроєктовує їх над частково дослідженими даними [10]. Результат є інколи точнішим, але цілком втрачає зрозумілість; замість використання метрик впевненості як попередні алгоритми, отримані матриці є складними для аналізу маркетологами.

1.4 Визначення вимог для рекомендаційної системи товарів з алгоритмом Apriori

Згідно зазначених вище характеристик та проблем в задачах рекомендаційних систем товарів можливо вивести певні вимоги щодо його функціоналу. Ці розроблені вимоги дозволять цілеспрямовано спроєктувати частини системи згідно них, та визначити певні мірки за якими на етапі супроводу визначити ефективність розробленої рекомендаційної системи товарів.

Головною вимогою до рекомендаційної системи товарів є використання алгоритму Apriori для добування даних рекомендацій. З інших функціональних вимог, тобто функцій або завдань, які система повинна виконувати, можливо виділити наступні пункти:

- створення персоналізованих рекомендацій товарів для окремих користувачів, яка базується на попередніх транзакціях користувача;
- визначення асоціацій товарів, що дозволить в розділі окремого товару відобразити подібні/супутні товари, які часто купують разом з головним;

- динамічне налаштування фільтрів для рекомендацій лише певних груп товарів, включаючи підтримку «чорного списку» товарів, які не будуть рекомендованими незалежно від асоціацій.

В структурі, розроблена програмна реалізація повинна представляти програмний інтерфейс для інтеграції в системи інтернет магазинів, надаючи доступ до функцій отримання даних від них для навчання, видача рекомендацій за певним номером товару або користувача, та зчитування поточних налаштувань чорного списку.

Варто зазначити, що останнє завдання (фільтри) має фундаментальні обмеження через природу машинного навчання. Через це, завантаження фільтрів буде виконуватись виключно на етапі запуску програмної реалізації системи рекомендації товарів, для дотримання цілісності внутрішнього набору даних під час роботи аналізу даних.

1.5 Висновки до першого розділу

В першому розділі кваліфікаційної роботи було здійснено дослідження області рекомендаційних систем. Починаючи з простого розуміння та цілі використання рекомендаційних систем, наводячи різні методи їх інтеграції для продажу товарів, було розглянуто процес добування даних, з яких будуються рекомендації товарів. Аналізуючи усіх кроків загального добування, було також засвоєно принцип їх застосування для аналізу кошику ринку, що дозволяє представити рекомендаційної системи як інструмент аналізу маркетологам.

Після визначення як клієнти та маркетологи можуть взаємодіяти з рекомендаційною системою системою та порівняння інших алгоритмів добування даних рекомендацій товарів як FP-Growth та ECLAT, було визначено функціональні вимоги для рекомендаційної системи товарів, як персональні пропозиції та фільтрування даних.

РОЗДІЛ 2 ПРОЄКТУВАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ З АЛГОРИТМОМ APRIORI

З визначеними вимогами щодо реалізації рекомендаційної системи товарів, в даному розділі проводиться її проєктування як з архітектурної сторони (що та як взаємодіє між іншими частинами), так і з визначення принципу роботи функціональних компонентів.

2.1 Проєктування архітектури рекомендаційної системи товарів

Архітектура програмного продукту дозволяє визначити та розподілити рекомендаційну систему товарів на декілька частин, що зменшує складність в проєктуванні, реалізації та підтримці системи [27].

Для початку для проєктування архітектури необхідно визначити всі сутності, з якими рекомендаційна система товарів може взаємодіяти як на вході так і на виході. З них можливо визначити шість сутностей, зображених на рисунку 2.1.

Рисунок 2.1 – Вхідні та вихідні сутності рекомендаційної системи товарів

- База даних, у якій знаходяться транзакційні дані для побудови моделі рекомендацій товарів.
- Інформація про товари для визначення подібних товарів для уникання рекомендації подібного за призначенням товару.
- Налаштування для рекомендаційної системи, включаючи толерантність для рекомендації, чорний список, обмеження у вході та інше.
- Рекомендації певного товару, який інтегрується в інші сервіси включаючи веб-сайти магазинів товарів.
- Загальні звіти для маркетологів, які містять інформацію про усі побудовані рекомендації товарів.
- Діагностичні дані доступні адміністраторам, у випадку проблем з роботою рекомендаційної системи товарів.

Додатковим важливим елементом для визначення правильної архітектури є масштаб даних для цих сутностей. Частіше назване як «Big Data» (укр. «Великі дані») [35], воно визначається в трьох основних характеристиках: обсяг даних, швидкість накопичення, та їх різноманітність.

В залежності від їх пропорцій в реалізації можливо застосовувати цілком різні архітектури для побудови системи що опрацьовує їх – для прикладу, малий обсяг зі швидким накопиченням може працювати як моноліт що прямо приймає дані та видає результати майже в реальному часі; а навпаки то можливо побудувати серверне забезпечення, яке запускається лише періодично для довгої обробки [26].

У випадку рекомендаційної системи товарів з цих характеристик можливо визначити дуже високий обсяг даних (оскільки усі поточні і минулі дані є необхідними для побудови якісних рекомендацій товарів), середній темп накопичення цих даних (до однієї тисячі транзакцій на добу) з низькою різноманітністю цих даних, оскільки інформація транзакцій має умовно передбачуваний формат даних для побудови швидкого алгоритму обробки [30].

Через це, монолітна архітектура (де єдине програмне забезпечення містить у собі усі компоненти системи) з прикладу не є доцільним для

проєктування – хоча прискорення через локальні дані є суттєвим, ця безпосередність також збільшує ресурсоємність рекомендаційної системи товарів що зберігає цілу копію даних, що в гіршому випадку також сповільнить процес навчання моделі рекомендації товарів. [38]

Натомість рекомендаційна система товарів буде побудована на основі сервісів. Сервісно-орієнтована архітектура програмного забезпечення спрямовується на ізолювання та розподілення компонентів системи в окремі функціонально незалежні сервіси як абстрактні «чорні ящики» з прихованим внутрішнім станом. З такою абстрактністю, інші сервіси (застосунки) не можуть втручатися в процес, натомість взаємодіючи лише за чітко визначеними входами та виходами (так званими програмними інтерфейсами) [25].

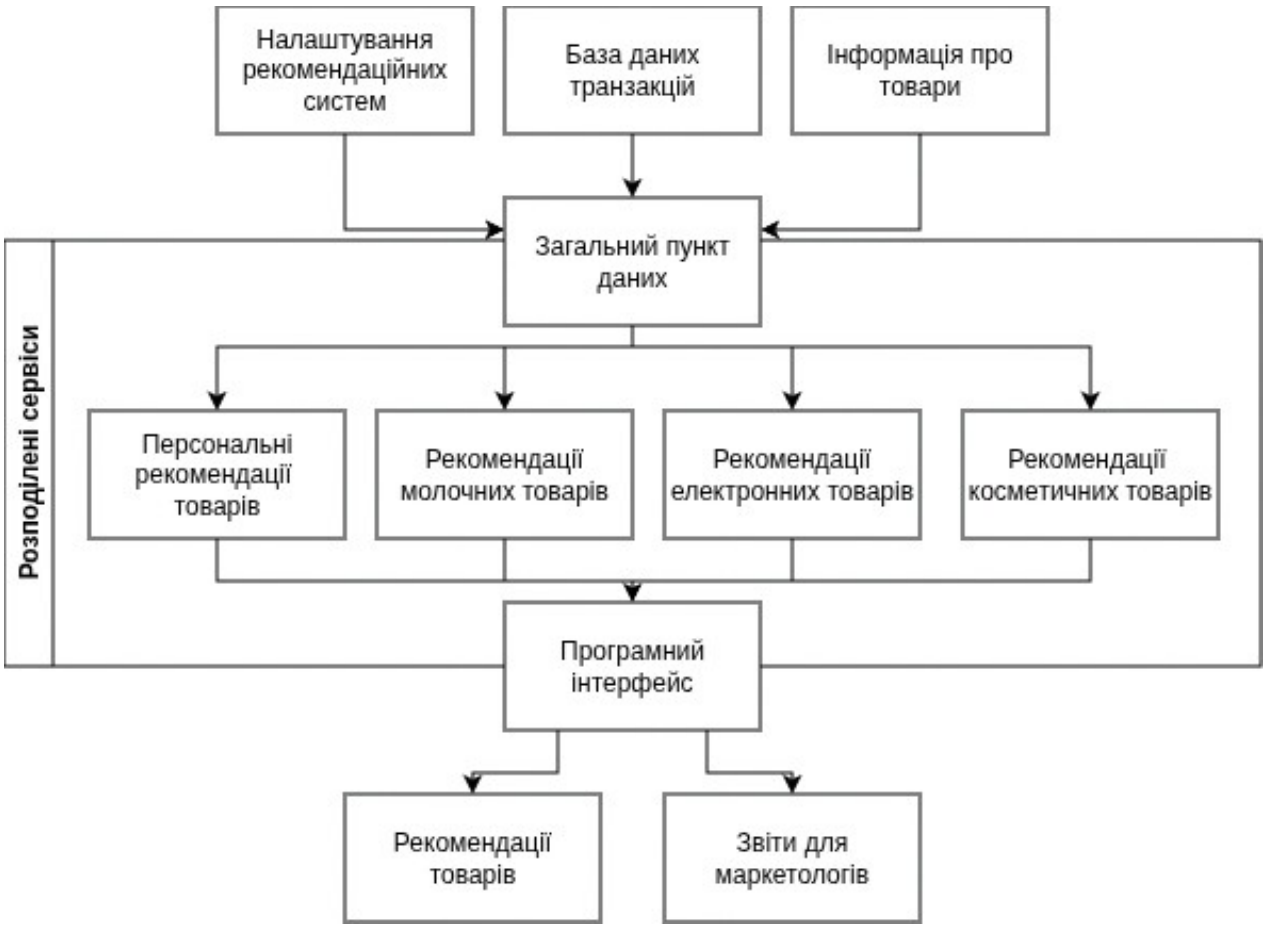


Рисунок 2.2 – Діаграма рекомендаційної системи як сервісів

Основна перевага такого розподілу проявляється в математично складних процесах. Оскільки рекомендаційні системи під час тренування можуть бути

доволі ресурсоємними особливо з більшими масивами транзакційних даних, поза загальним інтерфейсом її можливо розділити на окремі категорії та відповідно системи [31], як зображено на рисунку 2.2. Оскільки внутрішній процес не є видимим іншим, це дозволяє створювати обгортки для рекомендаційної системи, та динамічно розгортати їх по системах не змінюючи нічого ззовні, що є вкрай важливим для початкових комерційних фірм та сфери DevOps [20].

З боку різноманітності даних, їхня проблема є більш причетною до їх формату та змісту, з якого необхідно виводити аналіз чи інші обчислення; дані як відео або цілком не структуровані є найгіршими для обробки оскільки вони займають значного часу на просте форматування в належний формат, що в свою чергу несе ризик спотворення інформації з цих даних. Сама рекомендаційна система товарів взаємодіє з найпростішими даними – таблицями, які в програмній реалізації потребують мінімальної обробки. [7]

Формат табличних даних транзакцій складається з трьох полів – ідентифікаційний номер транзакції, номер клієнта та зміст кошика під час покупки, як описано в таблиці 2.1.

Таблиця 2.1 – Структура даних транзакції з прикладними даними

ID транзакції	**ID Клієнта**	**Список покупок**
05-28-4932	543	[60,33,21]
05-28-4933	642	[55,33,21,50,94,33]
05-28-4934	546	[60,94,58,92,21]
05-28-4935	322	[60]

Дотримуючись нормалізаційних вимог бази даних [37], в самих даних транзакцій застосовується мінімальна кількість текстових полів, натомість націлюючись на зовнішні ключі, які посилаються на належну інформацію в іншій таблиці. Хоча сама рекомендаційна система товарів в будь якому випадку опрацьовуватиме список покупок як рядків, дане точне визначення дозволяє

дуже просто та алгоритмічно розділити зміст та відбудувати масив покупок, зменшуючи ресурсоємність в порівнянні з додаванням цілої назви товару та клієнта.

Через це рекомендаційна система товарів буде звертатися до іншої таблиці баз даних, що містить інформацію про товари за їхніми ключами; головним з цього є лише категорія товару, за якою товари що збігаються будуть відраховуватися з рекомендацій.

2.2 Проєктування функціоналу алгоритму Apriori рекомендаційної системи товарів

Згідно раніше згаданих етапів в роботі систем добування корисних даних, продукт розкладається на щонайменше три частини – завантаження даних, трансформація та обробка даних, й отримання закономірностей та рекомендацій. З додаткових функціональних вимог закономірності також повинні містити певну залежність від інформації про користувача, а також використання чорного списку для уникання небажаних товарів у рекомендації.

2.2.1 Знаходження кандидатів за алгоритмом Apriori та правила побудови рекомендаційних асоціацій

Починаючи з самого ядра рекомендаційної системи, алгоритм Apriori є одним з ранніх алгоритмів для побудови правил закономірностей. Робота алгоритму починається з пошуку в базі даних та внутрішньої структури даних за деревом що дозволяє додатково зменшити ресурсоємність, оскільки цілі дублікати кошиків вказуватимуть на однакову гілку. За принципом послідовного ускладнення, суть алгоритму полягає в дедалі дрібнішому визначенні комбінацій товарів, які досі підтримують певний поріг появи – усі менші зазвичай цілком знищуються з дерева. Принцип роботи алгоритму Apriori є зображеним на рисунку 2.3.

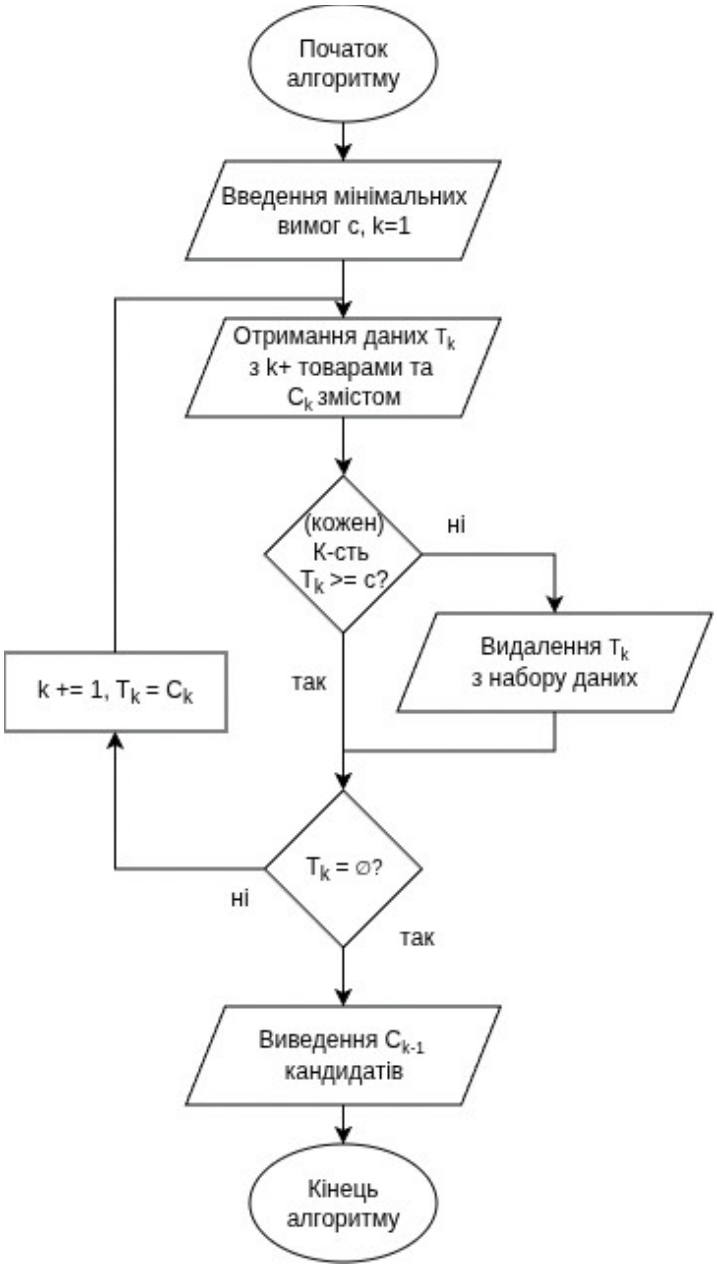


Рисунок 2.3 – Діаграма алгоритму Apriori

Детальніше описуючи цю діаграму, алгоритм здійснює наступні кроки:

1) алгоритм приймає одне значення на вході, яке характеризується як мінімальна межа впевненості, а також має внутрішнє число що починається з 1 та позначається як k;

2) до алгоритму надходять усі транзакційні дані, які містять щонайменше k кількість елементів у собі;

3) кожна комбінація товарів рахується за кількістю появ в транзакціях;

4) над кожною групою товару, їхня кількість звіряється з мінімальною вимогою у впевненості; якщо вона не задовільна, то даний товар «відраховується» з подальших циклів;

5) після зрівняння всіх елементів, перевірка на відсутність в наборі транзакційних даних

6) якщо дані залишилися, то вони додаються до кандидатів, які повинні знаходитись у нових транзакційних даних, а також збільшується значення k;

7) якщо дані відсутні, то попередні кандидати залишаються кінцевим результатом, завершуючи роботу алгоритму.

Така ітеративна послідовність дозволяє отримати рекомендації для дедалі більшої кількості товарів за специфічністю кошика. Поза алгоритмом але як його частина є добування правил асоціацій [40], яке характеризується метрикою «підйому» (англ. lift) у формулі 2.1.

$$lift(X \Rightarrow Y) = \frac{support(X \cup Y)}{support(X) \times support(Y)} \tag{2.1}$$

В залежності від отриманого значення можливо побачити пропорцію появи обох товарів X та Y водночас в порівнянні з їхньою появою окремо, де більше значення відповідно сигналізує про певну кореляцію між двома товарами. Інша згадана у попередній формулі метрика є «підтримка», що просто складається як відношення появи певних даних до загальної кількості оброблених транзакцій, а також «впевненість» у формулі 2.2, що натомість визначає наскільки здобуте правило працює відносно першого товару.

$$conf(X \Rightarrow Y) = \frac{support(X \cup Y)}{support(X)} \tag{2.2}$$

Але алгоритм Apriori містить чимало недоліків у своїй роботі, що потребує змін у його функціоналі або в інших етапах цілої рекомендаційної системи товарів.

2.2.2 Оптимізація ресурсоємності та персоналізація рекомендацій

Перші проблеми з ресурсами з’являються ще у першому єтапі цілого циклу алгоритму – добування даних. Пошук даних з транзакційних бази даних є надзвичайно ресурсоємним в часі процесом, оскільки алгоритм здійснює повторне сканування бази даних для кожного рівня глибини k. Враховуючи це, безпосередньо надавати прямий доступ до транзакційної бази даних в середовищі експлуатації не є оптимальним; натомість варто застосовувати кешовану копію транзакційних даних, призначена суто для тренування (цей процес не входить в програмну реалізацію ядра, оскільки вхідні даних та відповідно база даних є поза межами функціоналу сервісу).

Друга проблема з’являється на етапі визначення кандидатів та побудови асоціативних правил для рекомендації товарів. Даний процес здійснюється як обчислення ймовірностей (підтримки) кожної комбінації тих товарів, які досягли мінімальної кількості підтримки.

При цьому, зі збільшеною кількістю унікальних товарів для рекомендації відповідно збільшується кількість комбінацій, які необхідно розглянути, обчислити та зберегти. Для прикладу, при чотирьох товарах зображених на рисунку 2.4 здійснюється максимум 15 обчислень. Також зване як «комбінаторний вибух» [43], це спричиняє практично експоненційну складність у ресурсоємності як з обчислювання так і з зберігання інформації – особливо враховуючи складніші ланцюги транзакцій з різноманітними товарами, які сягатимуть мільйони пар товарів-кандидатів.

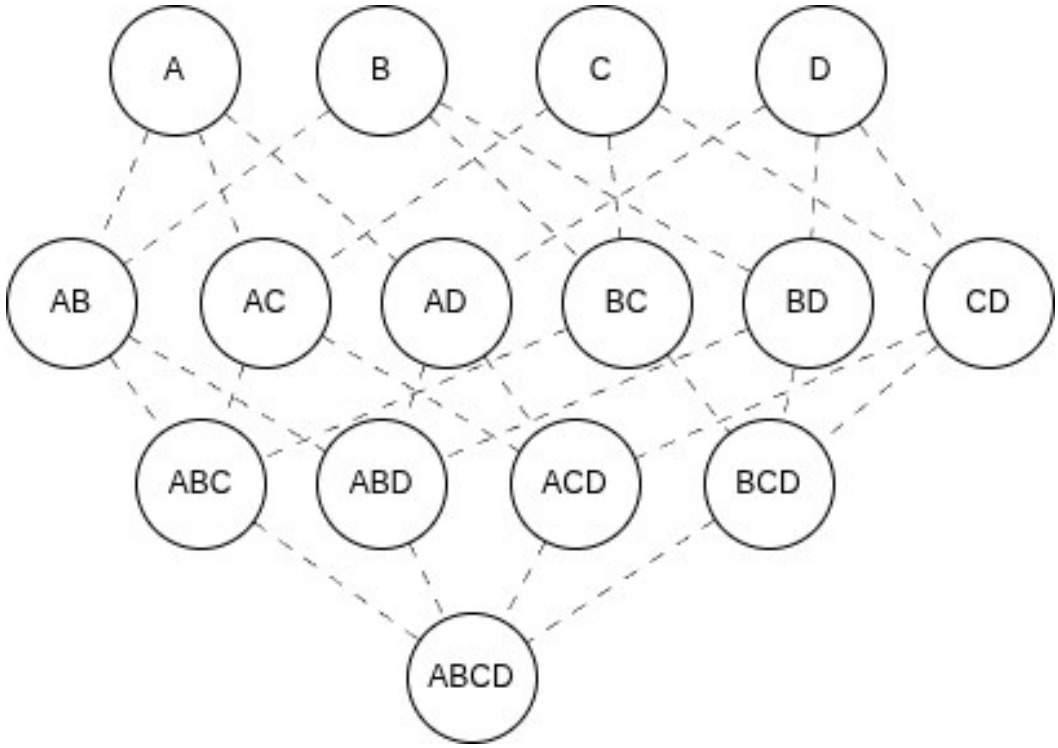


Рисунок 2.4 – Усі комбінації в транзакціях з товарами ABCD

Для уникання цієї проблеми необхідно застосовувати різні оптимізації для зменшення набору даних або максимальної довжини цих транзакцій. Хоча таке штучне обмеження довжини частково зменшує сумарну кількість кандидатів для обчислення, найбільше ускладнення обов'язково відбувається саме на першому кроці обчислення пар товарів. Як згадано у рисунку 2.2, рекомендаційну систему можливо розкласти на окремі сервіси за категоріями товарів, що в свою чергу значно зменшує простір товарів лише до тих продуктів, які дійсно можуть залежати одне від одного.

Таке рішення як побічний ефект також призводить до певної фільтрації даних. Оскільки результати рекомендаційної системи часто є цілком автоматизованими до незалежності від будь якого персоналу що відповідає за якість рекомендацій (особливо в електронних магазинах), їхні результати можуть бути вкрай провокативними та неетичними (для прикладу, алкогольні напої з дитячими товарами). Такий розподіл відповідно дозволяє ізолювати проблематичний товар від решти товарів, незалежно від їхньої справжньої залежності але дозволяючи знаходити подібності (рекомендації) лише між ними.

Останньою функцією для визначення є персоналізація пропозицій. За замовчуванням, алгоритм Apriori не має можливості розглядати дані особисто за даними користувача, тим самим не задовільняє вимогу щодо створення персоналізованих пропозицій для клієнта як зазначено у таблиці 1.1. Такий загальний підхід вважається як «колаборативним» [32], оскільки дані усіх користувачів входять без надання переваги для будь якого з них (окрім власне кількості транзакцій). Деякі методи вирішення цієї проблеми полягають в добуванні подібних товарів за характеристиками, які є важливими клієнту [41].

Спроєктованим методом для добування даних саме з інформації одного користувача нам необхідно переосмислити їхні транзакції як окремий список, над яким необхідно проводити аналіз та добування даних. Найпростішим методом для цього є застосування розбудованих правил для додаткового фільтрування, як зображено на рисунку 2.5.

Рисунок 2.5 – Персоналізований алгоритм рекомендаційної системи

За даною процедурою персоналізації, окрема рекомендаційна система товарів бере за собою інформацію про транзакції для визначення можливих продуктів саме певного користувача. Для цього система також шукає в побудованих загальних рекомендаціях товарів усіх кандидатів, які також зустрічалися у транзакціях користувача, і тоді повторно фільтрує цих кандидатів для видалення товарів, які будь-коли раніше з'являлися в транзакціях клієнта. На кінець, з залишкових кандидатів знову знаходяться характеристики впевненості та підтримки для кінцевого виведення.

Процес видалення кандидатів забезпечує унікальність та релевантність товарів, збільшуючи ймовірність успішної рекомендації та покупки товару. У випадку що дублікати є передбаченими або очікуваними (наприклад, побутові товари як папір), для даного елементу логіки передбачається можливість деактивувати його.

Додатковим питанням є адаптація до короткотривалих трендів та нових товарів. Оскільки алгоритм Apriori під час тренування відкидає усі кандидати, які не відповідають мінімальним вимог щодо підтримки, динамічне додавання нових трендів та товарів вимагає постійного перенавчання. В гіршому випадку, в залежності від складності транзакцій та кількості товарів, цей процес може зайняти більше часу ніж сам короткотривалий тренд, втрачаючи користь клієнтам як неактуальні рекомендації. На даний момент це є планом на майбутнє покращення системи, але на даний момент передбачаються два часткових рішення, які змінюють систему достатньо для підтримки певної адаптації в близькому часі.

Перший можливий метод спрямовується виключно на тренди, які в кінцевому етапі дійшли до кандидатів у рекомендаційній системі під час останнього тренування. Додаючи окремий функціональний пункт для доповнення підтримки системи, нові транзакційні дані можуть додаватися в реальному часі, тим самим дозволяючи поступово налаштовувати найбільш рекомендований товар серед його конкурентів.

Другий метод є більш пасивним – замість активного доповнення самою транзакційною системою, в часі простою рекомендаційна система товарів зможе власноруч прослуховувати систему бази даних транзакцій на зміни які відбулися після останнього сканування. У випадку, що один або декілька товарів в цей самий період перетнули поріг мінімальної підтримки як на рисунку 2.6 (в ідеалі значно більше від нього, враховуючи затрачений час на перенавчання), система може розпочати процес перенавчання для врахування нових даних.

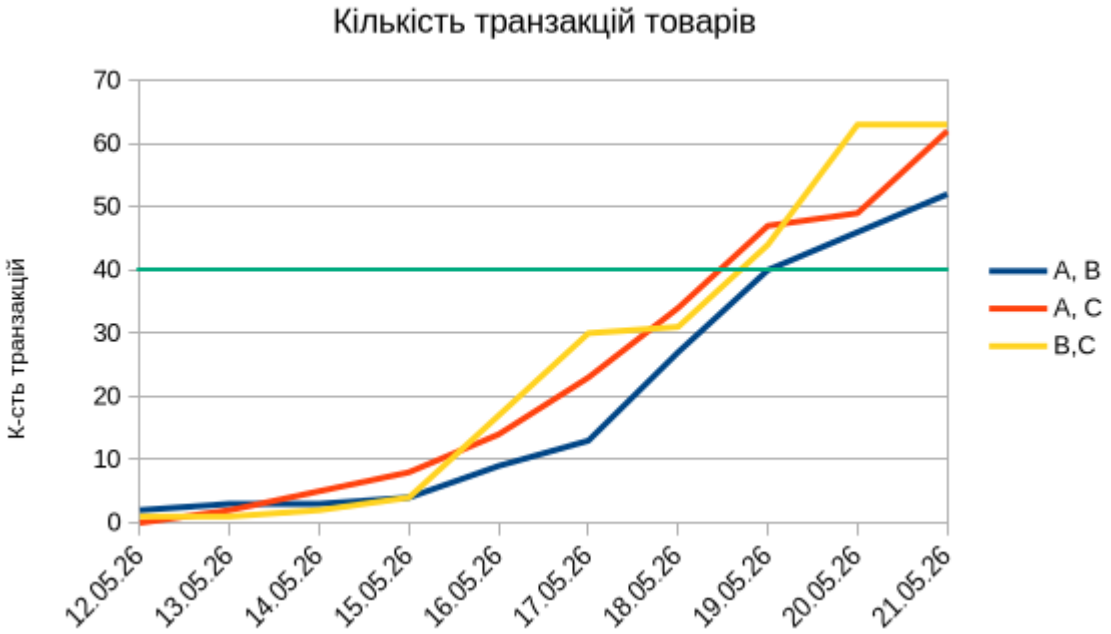


Рисунок 2.6 – Графік швидкого тренду товару (з межею перенавчання)

Недоліком такого підходу, в додаток до ризику постійного перенавчання через достатнього заповнення даних протягом процесу побудови рекомендацій для товарів, є ризик спотворення або отруєння даних. Деякі продавці мають випадки шахраїв, які повертають або недоплачують товар, тим самим маючи фальсифіковані транзакції. В таких умовах, часте або автоматичне оновлення може спричинити спотворення даних та відповідно рекомендаційної системи перед тим як адміністратор чи інша особа виявить та очистить дані, що є особливо критичним в онлайн ринках з ризиком нелегітимних продавців «товари» яких можуть стати широко рекомендованими в системі [28].

2.3 Визначення програмного інтерфейсу для взаємодії з рекомендаційною системою товарів

Оскільки ядро рекомендаційної системи товарів не передбачає будь якого графічного інтерфейсу у своєму складі, натомість орієнтуючись на інтеграцію в (існуючу) інфраструктуру, для роботи передбачається розробка програмного інтерфейсу.

Прикладний програмний інтерфейс (або API) визначається як набір стандартизованих та публічно доступних методів або протоколів, за допомогою яких зовнішнім елементам можливо взаємодіяти з системою. Хоча графічний інтерфейс також вважається як програмним інтерфейсом за даним визначенням, різниця між ними полягає в структурі та методі доступу цих методів. На відміну від графічних кнопок та списків де зовнішнім елементом передбачається лишу людина, програмні інтерфейси передбачають певну структуру та інструменти, які практично виконуються лише іншими програмними системами, для прикладу застосування веб-запитів.

Як було визначено раніше, рекомендаційна система товарів проєктується у виді сервісно-орієнтованої архітектури, у яких система насамперед розглядається «чорним ящиком». В такому випадку прямий доступ до ядра є цілком недопустимим, тому програмний інтерфейс є єдиним доступним методом для комунікації, з рядом переваг:

- оскільки рекомендаційна система товарів самостійно визначає та представляє свої кінцеві точки для взаємодії, вона відповідно має цілковитий контроль над їх функціоналом, що дозволяє ввести авторизацію та контроль даних, значно обмежуючи поверхню атаки та можливість вразливостей;
- програмний інтерфейс діє як шар абстракції, що дає змогу у подальшій розробці значно змінювати внутрішню логіку рекомендаційної системи, при цьому за умови що вихідні результати дотримуються визначених інтерфейсів зовнішні інтеграції не потребують змін;

- незалежність від програмних мов, дозволяючи інтегрувати рекомендаційне ядро з мови Python до різних зовнішніх компонентів з мовами PHP або C++

В залежності від потреб рекомендаційної системи, для розробки програмного інтерфейсу можливо застосувати різні архітектури та методології, з певними вимогами щодо структур даних. Хоча дані цілком можливо передати у бінарному вигляді, як це відбувається у розподілених обчисленнях наприклад OpenMP, для більш загальних програмних систем часто передбачається структурований формат даних. В структурованих форматах, таких як JSON [13] (який походить від JavaScript, але давно переріс його) та XML, дані представляються у вигляді дерева та гілок у форматі ключ-значення, як описано у лістингу 2.1.

Лістинг 2.1 – Інформація транзакції в JSON структурі

```
{
    "transaction_id": 3,
    "user_id": 6,
    "contents": [
        34,
        22,
        15,
        75
    ],
    "timestamp": "2026-05-23"
}
```

В умовах рекомендаційної системи, оскільки внутрішня структура даних також відображається як дерево кандидатів товарів, цей формат підходить для передачі рекомендаційних даних – виведені дані будуть описуватись як ланцюг товарів, в кінцевих гілках якого знаходяться характеристики добутих правил цих рекомендацій.

У випадках де дерево-подібність не потрібна для даних (наприклад налаштування рекомендаційної системи товарів) можливо застосувати простіші формати як TOML та INI.

Сам програмний інтерфейс буде розроблений за принципом REST, яка передбачає доступ до сервісу за HTTP запитами [34]. Перелік функцій, форматів даних та адрес описується у таблиці 2.2.

Таблиця 2.2 – Пункти функцій рекомендаційної системи товарів

Функція	**Адреса**	**Вхідні дані**	**Вихідні дані**
Надання рекомендацій	`GET /api/apriori/item`	Ідентифікатор товару	Структурований список кандидатів для введеного товару
Персоналізація пропозицій	`GET /api/apriori/user`	Ідентифікатор користувача	Структурований список кандидатів за даними користувача
Перегляд статистик	`POST /api/apriori/model`	Ключ маркетолога	Структурований список усіх рекомендацій, HTTP 401 (відсутній ключ)
Зміна чорного списку	`POST/DELETE /GET /api/apriori/blacklist`	Ключ, ідентифікатор товару (POST/DELETE)	Перелік чорного списку Код 201 (успішний POST) Код 200 (усп. DELETE) Код 400 (неуспішно)
Перезапуск	`POST /api/apriori/reset`	Ключ адміністратора	Код 200 (успішно) Код 401 (неуспішно)
Виведення звіту	`GET /api/ apriori/logs`	Ключ адміністратора	Текстовий файл з переліком подій за часом

Така парадигма є умовно простою для реалізації – застосовуючи прості операції доступні в протоколі HTTP (як GET, POST, UPDATE, REMOVE), реалізовані інтерфейси лише повинні не мати будь якого внутрішнього стану, або іншими словами не залежати від попередніх взаємодій з клієнтом. Це

дозволяє відразу звільнювати ресурси замість затримки сесії в пам’яті, але потребує що клієнт постійно вводить усі необхідні дані при взаємодії з інтерфейсом.

Інші характеристики архітектури є кешуванням частих відповідей по шляху, що дана рекомендаційна система товарів задовольняє тримаючи усі можливі дані (окрім клієнтських рекомендацій) в пам’яті, а також уніформно визначений інтерфейс. Завдяки тому, що ці запити є побудовами поверх протоколу HTTP, інтеграція результатів рекомендаційної системи товарів є дуже легкою у інші веб сервіси, наприклад інтернет магазини.

Для введення даних в рекомендаційну систему з бази даних можливо застовувати лише запитуючи її крізь SQL запити. Дана мова є широко застосованою серед більшості систем керування реляційної бази даних, що передбачає фільтрацію даних по категоріям та користувачам ще до моменту їх передачі по мережі. Тому для підтримки різних конфігурацій передбачається лише зміна адреси системи бази даних та назви таблиць.

2.4 Висновки до другого розділу

В другому розділі кваліфікаційної роботи для рекомендаційної системи товарів було спроєктовано як майбутній програмний продукт, визначаючи її архітектуру, функціонал, та інтерфейс.

Розклавши сутності на окремі підрозділи та визначивши їхнє навантаження на систему в цілому, було вирішено розкласти програмний продукт за сервісною архітектурою, що дозволяє розкласти та розподілити частини рекомендацій за фільтрами, що в свою чергу прискорює знаходження необхідних рекомендаційних товарів.

Для розробки персоналізації за користувачем було розроблено допоміжний алгоритм, який застосовує Apriori разом з фільтром на некуплені товари клієнта. Для взаємодії з системою було передбачено архітектуру REST, яка використовує методи веб-протоколу для легкої інтеграції з веб-магазинами.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ

Після визначення кожної функції у рекомендаційній системі товарів, їхнього механізму роботи, та причини за певними унікальними змінами як системою категорії, можливо застосувати ці знання на практиці. У даному розділі проводиться практичний етап розробки рекомендаційної системи товарів алгоритмом Apriori, а також тестування готового продукту, тим самим демонструючи результати роботи.

3.1 Програмна реалізація компонентів рекомендаційної системи товарів

Як фундаментальною мовою для програмної реалізації алгоритму Apriori рекомендаційної системи товарів було обрано Rust. З примусовою вимогою до безпечних реалізацій функцій (тим самим уникаючи проблеми щодо витоку, зловживання даних та більшість вразливостей) на рівні компілятора, дана мова не нехтує швидкодією, будучи більш подібною до C як по синтаксису так і по можливостях та швидкості роботи [21]. Враховуючи що алгоритм Aprirori є ресурсоємним як і по швидкості, крізь процес обчислення пар кандидатів, так і утримання їх в пам’яті (що може призвести до вищезгаданих проблем витоку), дана мова є оптимальною для реалізації даної системи.

Рекомендаційну систему товарів можливо розділити на три ключових частини (не включаючи головну яка проводить ініціалізацію):

- обчислювальний компонент, у якому проводиться процес рекомендації товарів за Apriori, та побудови рекомендації для користувачів;
- компонент введення/виведення даних, який зчитує налаштування, редагує чорний список, та здійснює з’єднання з базою даних для отримання транзакцій товарів для побудови рекомендацій;

- програмний інтерфейс, який слухатиме за запитами від зовнішніх систем та здійснювати операції згідно них та повертати відформатовані дані.

Дотримання такого розподілу функцій як окремих частин цілого продукту дозволить підтримувати якість та читабельність коду в майбутньому за принципом чітко визначених меж у їх функціоналі.

3.1.1 Ядро рекомендаційної системи товарів, алгоритм Apriori

Як ключовий компонент, функціональне ядро складається лише з функцій які пов’язані з рекомендаційною моделлю – побудови рекомендаційної моделі товарів, отримання даних за певним ідентифікатором товару, виведення цілої моделі, та знаходження персоналізованих рекомендацій для певного користувача. У лістингу 3.1 описується процес побудови можливих рекомендованих комбінацій товарів, що включає в себе логіку запобіганню дублікатів категорій.

Лістинг 3.1 – Процес побудови кандидатів

```
while !current_frequent.is_empty() && k <= max_depth {
    let mut candidates: BTreeSet<BTreeSet<String>> =
BTreeSet::new();
    let prev_frequent_sets: Vec<_> =
current_frequent.keys().cloned().collect();
    for i in 0..prev_frequent_sets.len() {
        for j in (i + 1)..prev_frequent_sets.len() {
            let union_set: BTreeSet<String>=prev_frequent_sets[i]
.union(&prev_frequent_sets[j]).cloned().collect();
            if union_set.len() == k {
               if k == 2 {
               let items: Vec<_> = union_set.iter().collect();
                   let sub1 = item_infos.get(items[0])
                   .map(|info| &info.subcategory);
                   let sub2 = item_infos.get(items[1])
                   .map(|info| &info.subcategory);
               if sub1.is_some() && sub1 == sub2 {continue;}
               }
            candidates.insert(union_set);
          }
        }
    }
```

Побудова кандидатів є прямолінійною, складаючись з двох циклів, які поступово додають до дерева BTreeSet свою комбінацію крізь метод union(). В мові Rust також є концепція «привласнення» змінної, що унеможливлює прямої передачі значення до дерева (принаймні без видалення його з цілком з циклу); для цього після об'єднання створюється клон, а з нього «колекція». При парних комбінаціях алгоритм розглядає інформацію про товари (що складається просто з ідентифікатора та підкатегорії) та у випадку що їхні категорії збігаються пропускає додавання до кандидатів. Сам процес обчислення підтримки складніших кандидатів є насправді простим – у типі даних дерева (або точніше гілки, оскільки воно викликається з неї) є функція is_subset(), до якої приймається транзакція – у випадку що в транзакції є даний кандидат, лічильник підтримки збільшується.

У лістингу 3.2 описаний кінцевий крок алгоритму, де над здобутими кандидатами здійснюється перевірка підтримки та будуються правила асоціації та їх статистики.

Лістинг 3.2 – Фільтр підтримки, визначення впевненості

```
current_frequent = candidate_counts.into_iter()
.filter(|(_, count)| *count >= min_support_count).collect();
for (set, count) in &current_frequent {
    all_supports.insert(set.clone(), *count);
    let mut confidence = 1.0;
    if set.len() > 1 {
        let mut prefix = set.clone();
        if let Some(last) = set.iter().next_back().cloned() {
            prefix.remove(&last);
            if let Some(prefix_support) =
all_supports.get(&prefix) {
            confidence = *count as f64 / *prefix_support as f64;
            }
         }
    }
frequent_itemsets.push(FrequentItemset {
      items: set.clone(),
      support: *count,
      confidence,
});}
```

Цей процес також знаходиться в загальному циклі алгоритму Apriori, оскільки інформація про підтримку k-1 елементів в all_supports (що відповідно необхідне для обчислення впевненості) не виходить за її межі; через це впевненість обчислюється та знаходиться в структурі даних FrequentItemset. Знову як перевага структури даних за деревом, знаходження відповідного k-1 елемента здійснюється крізь .remove() останнього елемента (який знаходиться як останній елемент в «ітераторі» гілки). Якщо воно існує (крізь перевірку Some()) та є інформація в загальному списку, впевненість є відношенням кількості даної комбінації відносно усіх з попереднім товаром.

В кінцевому результаті виводиться простіший масив з структурою FrequentItemsets (що складається з масиву ідентифікаторів товарів, її підтримка та впевненість), з яким легше працювати у процесі виведення корисної інформації. У лістингу 3.3 описано процес надання рекомендації за певним товаром.

Лістинг 3.3 – Знаходження рекомендованого товару

```
pub fn items(item_id: &str, model: &[FrequentItemset]) ->
Vec<FrequentItemset> {
    logger(&format!("Fetching items for item_id: {}", item_id));
    let mut candidates: Vec<FrequentItemset> = model
        .iter()
        .filter(|set| set.items.contains(item_id))
        .cloned()
        .collect();
    candidates.sort_by(|a, b|
b.confidence.partial_cmp(&a.confidence).unwrap());
    candidates.into_iter().take(2).collect()
}
```

Для знаходження кандидатів з моделі за фільтром (міститься товар) клонуються можливі кандидати, які тоді сортуються за їхніми характеристиками впевненості. Стандартні функції в Rust повертають дані у вигляді Result<T,E> або Option<T> – ці типи даних свідчать про невизначеність (у випадку що операція неправильно виконалась), яку необхідно передбачити або безпечно розпакувати для отримання необхідних даних. В крайньому випадку як у даному лістингу, unwrap() розгортає отриману невизначеність. В результаті, з

сортованого списку кандидатів обирається лише 2 верхніх кандидати та автоматично повертається як вектор.

Останньою функцією в ядрі рекомендаційної системи товарів є виведення даних за користувачем, який є описаним у лістингу 3.4.

Лістинг 3.4 – Функція рекомендацій для користувача

```
pub fn userprefs(
    model: &[FrequentItemset],
    transactions: &[String],
    duplicate_toggle: bool,
) -> Vec<FrequentItemset> {
    logger("Fetching user preferences");

    let mut results: Vec<FrequentItemset> = model
        .iter()
        .filter(|set| {
           let mut items = set.items.clone();
           if let Some(last) = items.iter().next_back().cloned() {
                items.remove(&last);
                let contains_prefix = items.iter()
                .any(|i| transactions.contains(i));
                let last_in_transactions =
transactions.contains(&last);
contains_prefix && (duplicate_toggle || !last_in_transactions)
            } else {
                false
            }
        })
        .cloned()
        .collect();

    results.sort_by(|a, b|
b.confidence.partial_cmp(&a.confidence).unwrap());
    results.into_iter().take(20).collect()
}
```

Згідно раніше визначених вимог, в процесі визначення рекомендованих товарів саме для користувача також передбачається відфільтрування будь-яких товарів які були раніше купленими – через це під час виклику функції також передається перемикач для вимкнення даного фільтру (що є одним з недоліків в даній мові, оскільки значення аргументів за замовчуванням немає, що відповідно вимагає визначення цього перемикача при кожному виклику функції).

Окрім цього, процес добування рекомендацій є подібним до простих товарів – модель фільтрується, знаходяться кандидати за транзакціями без останнього товару, якщо цей кандидат товар вже був в списку транзакцій то він не зараховується. Отримані кандидати також сортуються але на цей раз їх виводиться 20.

3.1.2 Модуль вводу/виводу транзакційних даних та налаштувань

Оскільки підтримка операцій з системою керування бази даних виходить далеко поза межі рекомендаційної системи товарів, задля уникання занадто великого масштабу роботи в реалізації рекомендаційної системи товарів було вирішено застосувати окрему бібліотеку для серіалізації даних та зв'язку з базами даних. Для зв'язку з базою даних використовується sqlx, що підтримує повну безпеку пам'яті та асинхронність маючи чисті драйвери для різних систем керування бази даних [23]. Як недолік, певні основні функції (як з'єднання) мають різні імплементації в залежності від цільової системи – але з боку розробника воно не вимагає змін окрім власне варіанту функції, оскільки розмітка є стандартизованою.

У лістингу 3.5 описується основна функція завантаження транзакцій.

Лістинг 3.5 – Завантаження даних за категорією

```
pub async fn load(db_ip: &str, db_u: &str, db_p: &str,
                  tables: &[String], cat: &str,
) -> Result<Vec<Vec<String>>, Box<dyn Error>> {
    let pool = PgPoolOptions::new()
        .max_connections(5)
        .connect(&format!("postgres://{}:{}@{}/{}", db_u, db_p,
db_ip,db_name))
        .await?;

    let table_name = &tables[0];
    let query = format!("SELECT contents FROM {}", table_name);
    //Отримання категорій
    let allowed_items: Option<std::collections::HashSet<String>> =
if cat != "all" {
        let items_table = tables.get(1).map(|s|
s.as_str()).unwrap_or("items");
```

```
        let item_query = format!("SELECT item_id FROM {} WHERE
category = '{}'", items_table, cat);
        let item_rows =
sqlx::query(&item_query).fetch_all(&pool).await?;
        let set: std::collections::HashSet<String> =
item_rows.into_iter().map(|r| r.get::<String, _>(0)).collect();
        Some(set)
    } else {
        None
    };
    let rows = sqlx::query(&query).fetch_all(&pool).await?;
    let mut transactions = Vec::new();
    //Фільтр
    for row in rows {
        let items_str: String = row.get("contents");
        let mut items: Vec<String> = items_str.split(',').map(|s|
s.trim().to_string()).collect();
        if let Some(ref set) = allowed_items {
            items.retain(|item| set.contains(item));
        }
        transactions.push(items);
    }
    Ok(transactions)
}
```

Умовно складніша операція, з’єднання передбачається саме на систему керування бази даних PostgreSQL [22] (що має окрему адресу для з’єднання). У випадку що змінна категорії не є «all», в додаток до отримання даних про транзакції окремо здійснюється вибірка ідентифікаторів товарів що мають певну категорію – після цього з кожного рядку транзакцій крізь retain() залишаються лише дозволені товари.

Завантаження налаштувань виконується крізь комбінацію бібліотек serde та toml. Serde є широко поширеною бібліотекою розпакування та запакування будь яких даних в різні формати, а TOML є стандартом для зберігання інформації в Rust проєктах (включаючи про сам проєкт та застосовані бібліотеки) [16]; в даному випадку воно містить адресу до системи керування бази даних, назва та пароль користувача, назви таблиць для транзакцій та предметів (на даний момент назва самої бази даних є строго закодовано), категорія, чорний список, ключі та налаштування алгоритму.

У лістингу 3.6 описується процес завантаження даних про користувача.

Лістинг 3.6 – Завантаження інформації про користувача

```
pub async fn load_user_transactions(
    db_ip: &str,
    db_u: &str,
    db_p: &str,
    table_name: &str,
    user_id: i32,
) -> Result<Vec<String>, Box<dyn Error>> {
    let pool = PgPoolOptions::new()
        .max_connections(5)
        .connect(&format!("postgres://{}:{}@{}/{}", db_u, db_p,
db_ip,db_name))
        .await?;
    let query = format!("SELECT contents FROM {} WHERE user_id =
$1", table_name);
    let rows =
sqlx::query(&query).bind(user_id).fetch_all(&pool).await?;
    let mut user_items = Vec::new();
    for row in rows {
        let items_str: String = row.get("contents");
        for item in items_str.split(',') {
            let item_trimmed = item.trim();
            if !item_trimmed.is_empty() {
                user_items.push(item_trimmed.to_string());
            }
        }
    }

    Ok(user_items)
}
```

Причиною поза створенням нових з’єднань при повторних викликах є кількість зв’язків – як сама система керування бази даними так і програмний продукт мають обмежену кількість сеансів які вони можуть підтримати; в зв’язку з цим операції зчитування даних натомість відбуваються з окремим з’єднанням що займає лише необхідну кількість часу перед виходом.

Останньою функцією є звітування. В реалізації алгоритмів рекомендаційних систем товарів воно з’являється як logger(), який приймає рядок. Сама функція є також дуже простою – беручи штамп часу за за форматом та комбінуючи крізь форматний макрос (рівний до функції printf), якщо файл log.txt відкривається то в нього прописується повідомлення.

3.1.3 Програмний інтерфейс рекомендаційної системи товарів

Реалізація програмного інтерфейсу для будь якої системи насамперед проявляє проблему потоків. По замовчуванням (тобто без додаткових зусиль), системи та їх функціонал виконуються таким чином, що лише одна дія може виконуватись в один час; в рекомендаційних системах товарів це недопустимо, оскільки навіть у випадку що операція, яка прослуховується в момент виконується дуже швидко, усерівно існує момент «провалу», який може створити збої в роботі при сильних навантаженнях.

Для вирішення цього в рекомендаційній системі товарів застосовується фреймворки Tokio для реалізації асинхронності [15] в функціоналі та Actix для кінцевих точок. У лістингу 3.7 описується реалізація пункту item, який виводить рекомендації за певним товаром.

Лістинг 3.7 – Інтерфейсний пункт рекомендації товарів

```
#[get("/api/apriori/item/{item_id}")]
async fn get_item(path: web::Path<String>, data:
web::Data<Arc<AppState>>) -> impl Responder {
    let item_id = path.into_inner();
    if item_id.is_empty() {
        return HttpResponse::BadRequest().body("Item ID cannot be
empty");
    }
    let model = match data.model.lock() {
        Ok(m) => m,
        Err(_) => {
            io::logger("Error: Model lock poisoned");
            return HttpResponse::InternalServerError()
            .body("Internal server error: model lock poisoned");
        }
    };
    let item_infos = data.item_infos.lock().unwrap();
    if !item_infos.contains_key(&item_id) {
        return HttpResponse::NotFound().body(format!("Item ID '{}'
not found", item_id));
    }
    let results = algo::items(&item_id, &model);
    HttpResponse::Ok().json(results)
}
```

Реалізація маршрутів (адрес, в яких здійснюються функції) виконується за допомогою макросу в першому рядку – просто визначивши метод взаємодії та шлях до неї [14] та визначивши як реалізацію інтерфейса Responder, наступна функція буде виконуватись при звертанні до цієї точки. Сама функція також має приймати ряд іншої інформації, включаючи дані цілої системи (для доступу до моделі без побудови при виклику), параметри як в шляху так і в запиті, корисні частини сформатовані як JSON так і простою формою, тощо.

Для даної операції необхідно лише взяти модель та ідентифікатор товару. При цьому, модель «блокується» на період роботи – це для запобігання переписання інформації в неї поки добуваються рекомендації за певним товаром. В кінцевому результаті повертається код 200 зі змістом двох кандидатів у відформатованому вигляді використовуючи HttpResponse. Складніша операція додавання до чорного списку є описаною у лістингу 3.8.

Лістинг 3.8 – Функція додавання чорного списку

```
#[derive(Deserialize, Serialize)]
pub struct BlacklistBody {
    pub key: String,
    pub item_id: String,
}
#[post("/api/apriori/blacklist")]
async fn post_blacklist(
    body: web::Json<BlacklistBody>,
    data: web::Data<Arc<AppState>>, )
    -> impl Responder {
    let mut settings = data.settings.lock().unwrap();
    if body.key != settings.api_key_1 {
        io::logger("Unauthorized access to
/api/apriori/blacklist");
        return HttpResponse::Unauthorized().finish();
    }
    if !settings.blacklist.contains(&body.item_id) {
        settings.blacklist.push(body.item_id.clone());
    }

    HttpResponse::Ok().finish()
}
```

Додатковою перевагою використання структурованих даних є швидкість розкладання їх за чітко визначеною структурою – після декларування очікуваної

структури під час взаємодії з чорним списком, в самому процесі отримання не описується процес добування ключа та ідентифікатора товару. Окрім цього, в даному коді також здійснюється перевірка ключа який знаходиться в запиті до внутрішнього ключа – якщо ні, це повідомляється та повертається код 401. Після цього до чорного списку додається товар (або в зворотньому випадку залишаються всі товари окрім зазначених) та завершується робота.

Завершуючи розділ програмної реалізації, у лістингу 3.9 описується початкова функція, яка ініціалізовує модель та програмний інтерфейс.

Лістинг 3.9 – Ініціалізація системи

```
#[tokio::main]
async fn main() -> Result<(), Box<dyn std::error::Error>> {
    println!("Initializing APRIORI recommendation system...");
    let settings = io::init()
    println!("Loading item information...");
    let item_infos = io::load_item_infos(&settings.db_ip,
&settings.db_u, &settings.db_p, &settings.tables[1],
    )
    .await.unwrap_or_default();
    println!("Loading transactions and generating model...");
    let transactions = io::load(&settings.db_ip,&settings.db_u,
&settings.db_p,&settings.tables,&settings.category,)
    .await.unwrap_or_default();
    let model = algo::apriori(settings.min_support,
settings.max_depth,&transactions,&item_infos,&settings.blacklist,
);
    let state = Arc::new(api::AppState {
        settings: Mutex::new(settings),
        model: Mutex::new(model),
        item_infos: Mutex::new(item_infos),
    });
    println!("Starting API server on 127.0.0.1:6782");
    api::run_server(state).await?;
    Ok(())
}
```

Стан системи зберігається у вигляді arc, що є структурою яка підтримує клонування при цьому наводячи на той самий комірець пам’яті, що власне робить асинхронну роботу системи можливою. Компоненти системи також знаходяться поза замком Mutex для запобігання зміни стану під час певних обробок. Хоча тут знову використовується процес розгортання, в програмних

пунктах виведення замість простого розгортання цього замка необхідно зробити виняткову процедуру на випадок що замок зайнятий в іншому потоці. Як згадано раніше, розгортання станів простим чином рекомендовано лише в умовах де воно гарантовано успішно розгорнутись – як на початку роботи програми; в інших випадках необхідно розробити повідомлення (як описано у лістингу 3.7) на випадок що об’єкт є зайнятим іншим потоком для уникання «паніки» [17], що примусово зупинить роботу сиситеми.

3.2 Тестування та експлуатація рекомендаційної системи товарів алгоритмом Apriori та її компонентів

Для використання рекомендаційної системи товарів з алгоритмом Apriori передбачається система керування бази даних PostgreSQL [1]. В ній повинні знаходитися дві таблиці:

- перелік транзакцій, з полями transaction_id, user_id та contents (у форматі цифр розділених комою);
- інформація про товари, з полями item_id, category (може бути пустим) та subcategory.

Для початку роботи з рекомендаційною системою надається пробний скрипт, який заповнює дані з 100 транзакціями, декількома користувачами та сорок товарів різних видів та категорій.

У кореневій папці рекомендаційної системи товарів знаходиться файл налаштувань setting.toml. У даному файлі необхідно змінити параметри DB_IP на адресу системи керування бази даних, DB_U та DB_P на інформацію користувача що має авторизацію для взаємодії з базою даних, назви вищезгаданих таблиць у форматі масиву.

Поля min_support, max_depth та duplicate керують функціоналом рекомендаційної системи товарів та налаштовують пороги та обмеження на добування даних алгоритмом Apriori. min_support визначається в межах 0-1 та становить відсоток транзакцій, у яких повинен бути товар перед визначенням

товарів як кандидатів для рекомендації; max_depth визначає максимальну глибину транзакцій для аналізу.

Після введення актуальних даних в налаштування, під час запуску рекомендаційної системи товарів у консолі виводяться діагностичні повідомлення про побудову моделі та відкриття програмного інтерфейсу на порту, як зображено на рисунку 3.1.

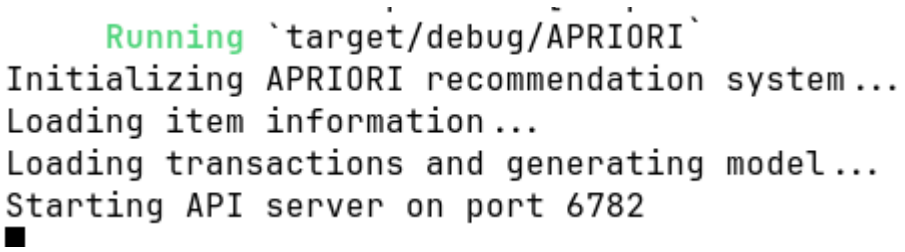


Рисунок 3.1 – Успішний запуск рекомендаційної системи

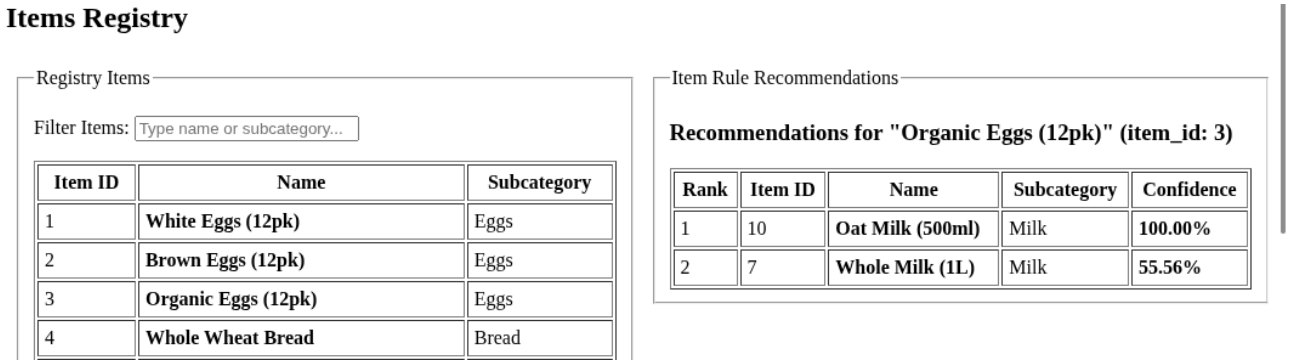
Взаємодія з рекомендаційної системою товарів здійснюється крізь запити на пункти програмного інтерфейсу /api/apriori/item, user або blacklist. Для налагодження та тестування цих пунктів можливо використати утиліти тестування пунктів. [9]

Для тестування надається окремий діагностичний веб-інтерфейс на основі Node.js [8]. Після налаштування файлу .env з параметрами входу в систему керування бази даних а також адресу рекомендаційної системи товарів, при запуску веб-сервера з файлом server.js за адресою localhost:5001 у веб-браузері можливо увійти в панель тестування програмного інтерфейсу рекомендаційної системи товарів.

Інтерфейс тестування складається з шести розділів – перелік товарів та їх рекомендації, рекомендації користувачів, керування чорним списком, інформація про модель, адмін панель та серверний журнал, який є присутнім на усіх розділах веб-інтерфейсу.

Розділ товарів складається з двох панель – перелік товарів в реєстрі та перелік рекомендацій. При натиску на будь який товар з переліку здійснюється

програмний запит за пунктом /api/apriori/item, результати якого виводяться на другу панель, як зображено на рисунку 3.2.



Items Registry

Registry Items

Filter Items: Type name or subcategory...

Item ID	Name	Subcategory
1	**White Eggs (12pk)**	Eggs
2	**Brown Eggs (12pk)**	Eggs
3	**Organic Eggs (12pk)**	Eggs
4	**Whole Wheat Bread**	Bread

Item Rule Recommendations

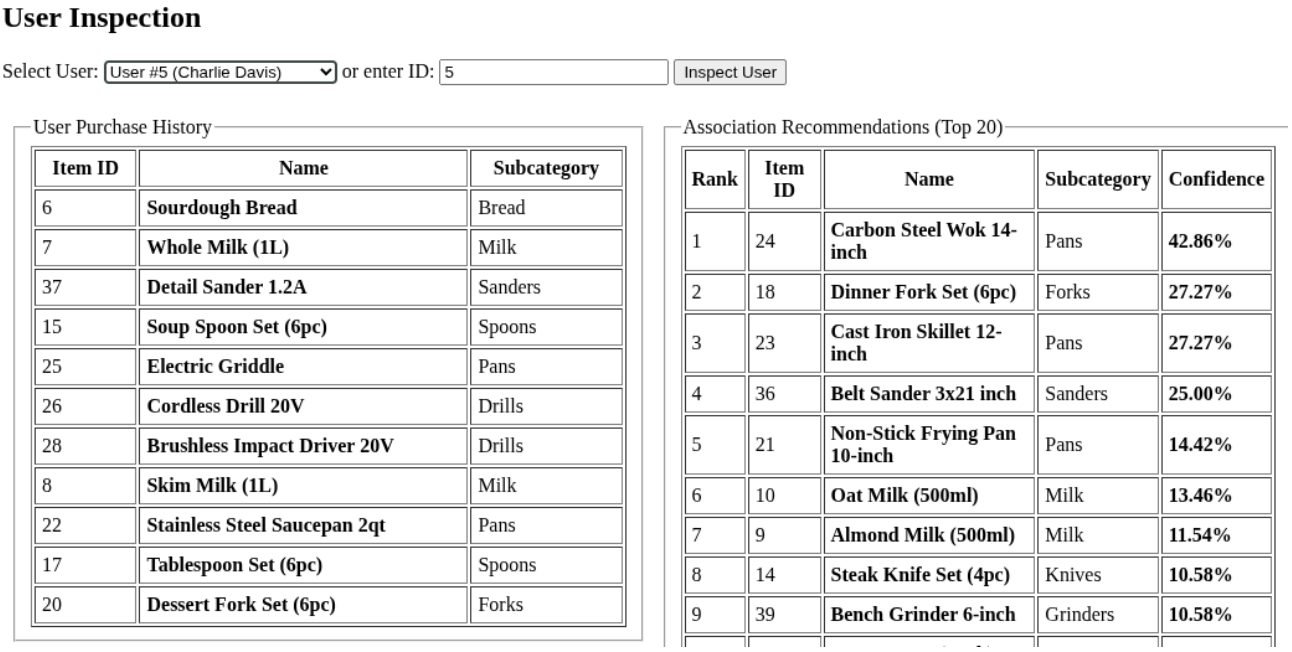
Recommendations for "Organic Eggs (12pk)" (item_id: 3)

Rank	Item ID	Name	Subcategory	Confidence
1	10	**Oat Milk (500ml)**	Milk	**100.00%**
2	7	**Whole Milk (1L)**	Milk	**55.56%**

Рисунок 3.2 – Виведення рекомендацій для певного товару

Рекомендації відображаються з додатковою інформацією підкатегорій та міри впевненості для діагностики механізму запобіганню дублікатів за видом товару.

Розділ виведення рекомендацій для користувачів також застосовує дві панелі для перевірки правила уникання раніше куплених товарів. Вибір користувача здійснюється крізь випадаюче меню або текстове поле для вказування певного ідентифікатора користувача, як зображено на рисунку 3.3.



User Inspection

Select User: User #5 (Charlie Davis) or enter ID: 5 Inspect User

User Purchase History

Item ID	Name	Subcategory
6	**Sourdough Bread**	Bread
7	**Whole Milk (1L)**	Milk
37	**Detail Sander 1.2A**	Sanders
15	**Soup Spoon Set (6pc)**	Spoons
25	**Electric Griddle**	Pans
26	**Cordless Drill 20V**	Drills
28	**Brushless Impact Driver 20V**	Drills
8	**Skim Milk (1L)**	Milk
22	**Stainless Steel Saucepan 2qt**	Pans
17	**Tablespoon Set (6pc)**	Spoons
20	**Dessert Fork Set (6pc)**	Forks

Association Recommendations (Top 20)

Rank	Item ID	Name	Subcategory	Confidence
1	24	**Carbon Steel Wok 14-inch**	Pans	**42.86%**
2	18	**Dinner Fork Set (6pc)**	Forks	**27.27%**
3	23	**Cast Iron Skillet 12-inch**	Pans	**27.27%**
4	36	**Belt Sander 3x21 inch**	Sanders	**25.00%**
5	21	**Non-Stick Frying Pan 10-inch**	Pans	**14.42%**
6	10	**Oat Milk (500ml)**	Milk	**13.46%**
7	9	**Almond Milk (500ml)**	Milk	**11.54%**
8	14	**Steak Knife Set (4pc)**	Knives	**10.58%**
9	39	**Bench Grinder 6-inch**	Grinders	**10.58%**

Рисунок 3.3 – Виведення пропозицій в рекомендаційній системі товарів

Перелік рекомендацій користувача виводить до 20 рекомендацій водночас, призначений для інтеграції на головних сторінках або окремих розділів персональних пропозицій.

Операції з розділом чорного списку вимагає коректного ключа №1, який автоматично завантажується з налаштувань панелі при запуску. Аналогічно до попередніх розділів, на рисунку 3.4 відображаються товари в чорному списку рекомендаційної системи товарів, а також приховані ключі та перелік доступних товарів для швидкого додавання в список.

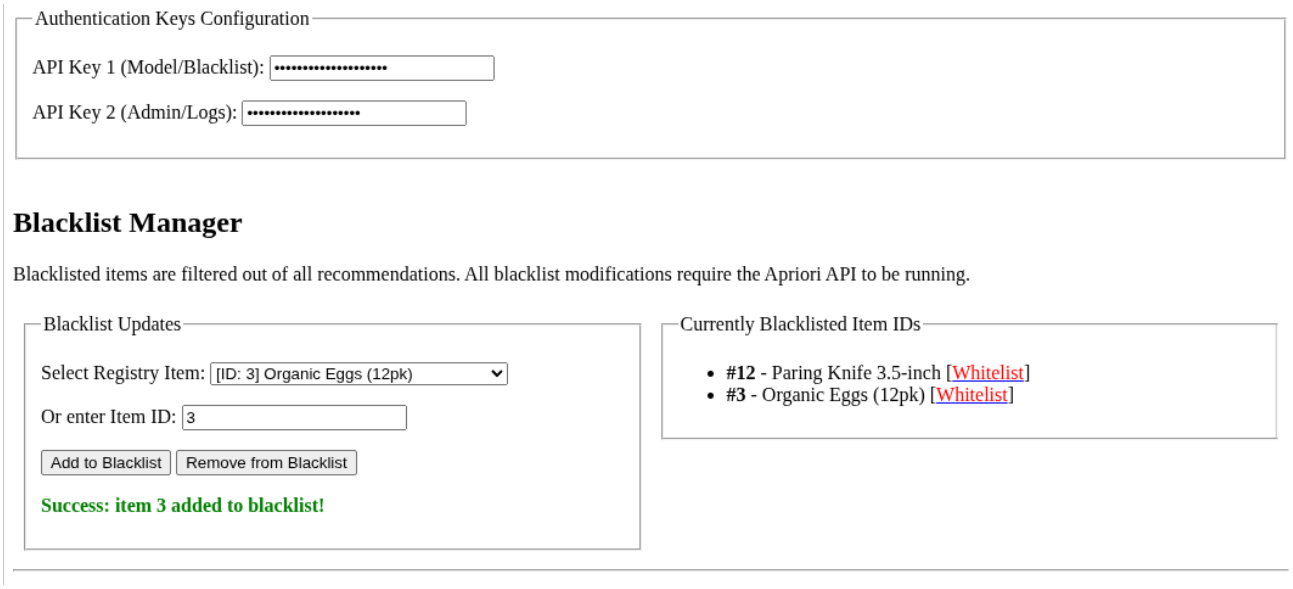


Рисунок 3.4 – Перегляд чорного списку в рекомендаційній системі товарів

У випадку, що ключ не співпадає, ідентифікатор товару неправильний або інші проблеми з рекомендаційною системою товарів, у панелі та консолі веб-інтерфейсу виводиться повідомлення з кодом помилки та поясненням для передбачених помилок, як зображено на рисунку 3.5.

Розділ інформації про панелі також застосовує ключ №1 та виводить усі добуті правила під час побудови рекомендацій товарів алгоритмом Apriori. Правила складаються з пари товарів, їх підтримки (кількість її появи в списку транзакцій) та впевненості у правилі (відношення кількість транзакцій з задовільненим правилом до транзакцій з першим товаром).

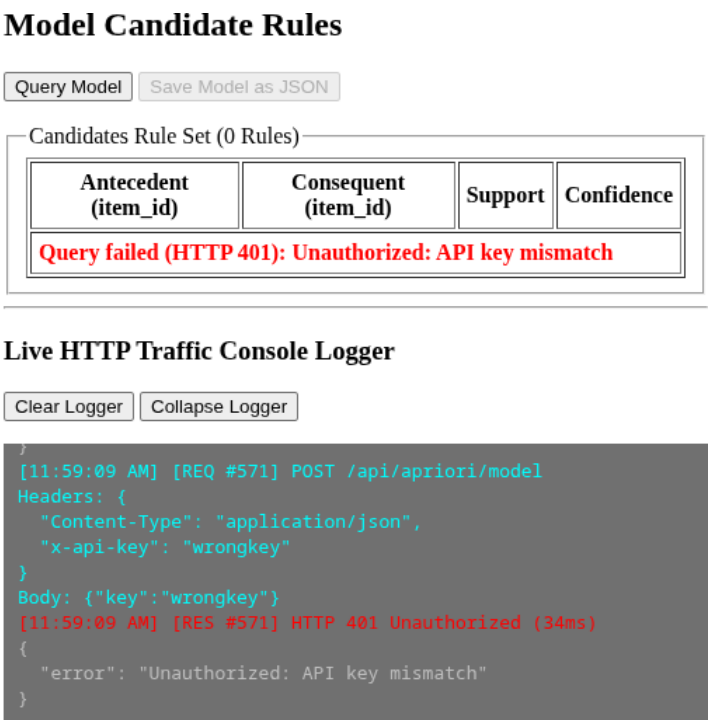


Рисунок 3.5 – Спроба аутентифікації з неправильним ключем

Адміністративний розділ передбачає використання другого ключа авторизації, та містить функції перенавчання моделі та виведення звіту рекомендаційної системи товарів. Функція перенавчання вимагає подвійної перевірки маючи потенційно критичні наслідки.

3.3 Висновки до третього розділу

В третьому розділі кваліфікаційної роботи проводився процес програмної реалізації алгоритму Apriori та рекомендаційної системи товарів. Після визначення необхідних технологій та компонентів для розробки, було описано кожен елемент логіки в компонентах алгоритму, зв’язку з базою даних та програмного інтерфейсу.

Під час тестування в додаток до інтегрованих тестів які виконуються в процесі компіляції також здійснено функціональне тестування у прикладному веб-інтерфейсі діагностики рекомендаційної системи товарів. Після перевірки кожної спроєктованої функції та її варіантів на працездатність, даний проєкт є готовим для застосування та інтеграції в сервіси.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Для кінцевих користувачів, як маркетологів які здійснюють аналіз добутих правил рекомендації або клієнтів які здійснюють рішення з допомогою виведених рекомендацій, процес прийняття рішень може бути довгим та напруженим. В залежності від середовища в якому проводиться робота, якість здійсненої роботи може значно погіршуватись або й нести ризик здоров'ю людини у випадку порушення норм безпеки та гігієни. Оскільки комп'ютерна система має власний вплив до обох проблем стресу від розумової праці та на робоче місце в якому вона знаходиться, у даному розділі розглядаються питання безпеки життєдіяльності людини, включаючи вплив стресу на людину та необхідні умови робочого місця для оптимальної працездатності.

4.1 Значення адаптації в трудовому процесі. Стрес і втома організму від тривалої розумової роботи

В роботі, центральна нервова система є відповідальною за функції координації, інтеграції (адаптації) та рефлексів. При цьому, функція адаптації відповідає за привлаштування до змін як в зовнішньому середовищі (адаптація зору), так і до більш психічних навантажень як робота з комп'ютерними системами, де адаптація відповідає за швидкість сприйняття інформації. Враховуючи це, адаптація також несе за собою попередження травмування, передбачаючи нещасні випадки до їх появи [3].

В трудовому процесі адаптація розрізняється на фізіологічну, соціальну, психічну та професійні види. Під фізіологічною адаптацією мається на увазі ряд реакцій, за допомогою яких організм пристосовується до змін зовнішніх умов при цьому притримуючи стабільний внутрішній стан крізь ціле тіло. У більш екстремальних адаптаціях функції організма порушуються через несумісність адаптованих частин, з часом знаходячи оптимальні стани для відновлення та стійкого пристосування до нових умов. В більшості випадках такий вид адаптації

за оптимальних умов веде до покращеної працездатності організму та стійкості до захворювань. В інших випадках, де умови постійно змінюються, натомість з’являється загроза захворювання організму.

Соціальна адаптація є пов’язаною з психічною адаптацією. В середовищі колективу, дана адаптація націлюється на вивчення та пристосування до правил та норм про колектив для пристосування в нього. У випадках коли адаптація не проходить сприятливо, натомість набирається стрес під час роботи.

Психічна адаптація відповідає за відповідність психології людини до середовища в робочому середовищі. Через це, адаптація залежить від різних психічних характеристик, включаючи загальний психічний стан, реакції на стреси, умови праці, особистий досвід людини тощо. В таких випадках стрес може призвести до ірраціональної поведінки людини та призвести до конфліктів або випадків.

Професійна адаптація розглядає процес пристосування до трудового процесу, влючаючи робоче місце, технологічний процес, час роботи, тощо. Адаптація виражається формування навичками для якісного виконання роботи, та володінням необхідними специфічними уміннями.

Належачи до розумової праці, робота з комп’ютерною системою передбачає професійну та психічну адаптацію. Це, в додаток до інших характеристик притаманних для тривалої праці як вищі вимоги до якості роботи, з малою рухливістю та погіршенням адаптації ока, призводить до високого ризику перевтоми, який залежно від ступеня може цілком порушити роботу організму тримаючи її в стресі без навантаження.

Проста тимчасова втома від роботи розрізняється за місцем впливу, як зорове втомлення очей, розумова втома, м’язова, загальна та інші. За психікою, втома характеризується поганою увагою до подразників, погіршенною концентрацією, погіршена довготривала увага до робочого процесу, проблеми запам’ятовування (тим самим погіршуючи професійну адаптацію), а також характеристики стресу як збільшена дратівливість та порушення координації.

Хоча втома цілком проходить з часом, у випадку для цього не виділений час, або умови праці є несприятливими вона може накопичуватись та перейти в перевтому. В важких ступених перевтоми нервова система людини тримається в стресі попри погану працездатність, натомість маючи безсоння з значно порушеною увагою та пам’яттю; в сукупності маючи збільшену загрозу до нещасних випадків як через погіршену адаптацію до середовища, так і крізь погіршену увагу роблячи помилкові дії а також несучи ризик захворювань.

4.2 Вимоги безпеки до робочих місць для виконання робіт

Робоче місце є просторовою зоною у підприємстві, яка є оснащеною необхідними засобами з якими відвуваються трудові роботи одного працівника або цілої групи. [4]

Безпека роботи з виробничими засобами забезпечується насамперед крізь безпечні принципи дії з ними та схеми елементів конструкції засобу, що здійснюється на етапі проєктування робочого місця. При проєктуванні принципу дій обладнання необхідно визначити та врахувати всі можливі чинники, які можуть бути небезпечними або шкідливими здоров’ю. Для прикладу, вентилятори комп’ютерної системи мають загрозу задання травми крізь контакт з швидко обертаючими лопатками; згідно цього їх необхідно розташовувати виключно всередині корпуса. [6]

Комп’ютерні системи складаються з джерел живлення, які містять трансформатор для переведення напруги та відповідно несе загрозу ураження струмом, що в людині тіла проводить перегрів м’яз (термічна дія), потенційно розриваючи їх, а також порушує органічні рідини (як кров) та відповідно порушує біологічні життєві функції.

Для забезпечення захисту робочого місця від електротравм передбачається максимальна ізоляція частин які можуть вести струм та відповідно попасти під напругу а також використання електрозахисних засобів як діелектричні рукавиці

або вимірювальні кліщі, які переноситимуть струм замість людини. Ізоляція струмопровідних частин включає наступні методи:

- розташувати їх в недосяжному місці;
- ввести робочу ізоляцію діелектриком (елементом високого опору), який забезпечить протікання струму лише по певному шляху;
- використати блоківки, які запобігають помилковим діям з живленними компонентами (наприклад замикання корпусу системи при живленні);
- заземлення корпусу та розеток.

Робота комп'ютерної системи також передбачає гігієнічність робочого місця, маючи вплив на повітря (крізь нагрівання та вентиляції під час роботи), шум компонентів, включаючи жорсткі диски, та освітлення при роботі з монітором.

Повітря (або мікроклімат) робочого місця характеризується вологістю, температурою, швидкістю руху повітря, тиском та тепловим випромінюванням. В разі порушення одного або декількох з цих характеристик може призвести до зниження працездатності людини, викликаючи професійні захворювання в серйозних та тривалих порушеннях. Оскільки система може сильно нагріватись під час навантаження, в особливо поганих умовах це може призвести до теплового удару. В залежності від вологості температура впливає на людину з більшим ефектом, а для системи з'являється ризик конденсації та короткого замикання компонентів. [5]

Для контролю якості повітря на робочому місці комп'ютерної системи виділяється удосконалення технологічного устаткування, дистанційне управління обладнанням, використання індивідуального захисту або введення вентиляції. Метод вентиляції може розрізнятися за способом переміщенням повітря та за зоною і часом дії (активно чи лише аварійно).

Від освітлення залежить працездатність не тільки психіки людини, а й її цілого організму, впливаючи на певні фізіологічні процеси в організмі людини. Правильне освітлення робочого місця сприяє підвищенню продуктивності роботи людини, та має позитивний вплив в залежності від складу світла. При

порушенні освітлення людина втомлюється та несе більший ризик виконання помилкових дій та отримання професійних захворювань пов’язаними з зором.

Декілька визначених вимог щодо освітлення робочих місць включає певний рівень за гігієнічними нормами, який має бути рівномірним в цілому приміщені (для зменшення адаптацій), бути спектрально подібним до сонячного світла у випадку штучного освітлення, та не бути блискучим в зоні праці.

4.3 Висновок до четвертого розділу

В четвертому розділі кваліфікаційної роботи висвітлені потенційні небезпеки, пов’язані з комп’ютерними системами. Враховуючи характер даних рекомендаційної системи товарів, процес прийняття рішень або аналіз даних передбачає адаптацію та несе за собою ризик перевтоми та стресу. Комп’ютерні системи також несуть за собою різні ризики в процесі роботи, будучи електричними приладами з окремою системою вентиляції, блоку живлення та освітленням від екранів.

ВИСНОВКИ

Під час розробки алгоритму Apriori рекомендаційної системи товарів, в розділах кваліфікаційної роботи освітнього рівня «Бакалавр» були здійснені три етапи розробки

В першому розділі роботи подано методи використання рекомендаційної системи товарів як допоміжного елементу в прийнятті рішень та аналітичним інструментом маркетологів, а також висвітлено загальний принцип роботи процесу добування даних як правил асоціації. Після аналізу подібних алгоритмів та їх характеристик було визначено загальні вимоги до рекомендаційної системи товарів.

В другому розділі були визначені питання щодо реалізації як архітектуру цілої рекомендаційної системи товарів за допомогою характеризування масштабу та різноманітності даних які входять для функцій побудови рекомендацій. Також визначені принципи роботи додаткових механізмів, які вирішують недоліки алгоритму Apriori, як ресурсоємність, та сформовані пункти програмного інтерфейсу до системи для інтеграції в зовнішні сутності.

Третій розділ розглядає реалізацію функціонального програмного продукту, включаючи розробку програмного коду рекомендаційної системи товарів у мові Rust, розділяючи її на окремі незалежні компоненти для полегшення майбутньої розробки. Через вимоги мови до безпеки було спроєктовано декілька заходів безпеки у програмному коді, вимагаючи стабільність програмного продукту та його виводів в процесі подальшої розробки. На кінці рекомендаційну систему товарів протестовано як з внутрішніми тестовими випадками, так і в розробницькому середовищі яке симулює повну інтеграцію з окремою системою бази даних та взаємодія з веб-інтерфейсом.

В розділі безпеки життєдіяльності висвітлені питання безпеки щодо розумово інтенсивної праці з рекомендаційною системою товарів, а також

загальні вимоги робочого місця через можливий вплив комп’ютерної системи на середовище роботи.

Результатом виконання кваліфікаційної роботи ступеня «Бакалавр» є реалізована система рекомендацій, яка ефективно застосовує алгоритм Apriori для побудови правил асоціацій. Дотримуючись сервісної архітектури, її робота передбачає відокремлення компонентів інтеграції та зберігання даних, дозволяючи розподілено масштабувати її для покращеної паралелізації. Оскільки метою даної роботи було доведення ефективності застосування даного алгоритму для аналізу даних, мета роботи вважається досягнутою.

В додаток до звичайної побудови рекомендацій алгоритмом Apriori також розроблено функція виведення рекомендацій користувачеві, таким чином створюючи компонент персоналізації до колаборативного алгоритму; інші функції включають фільтрування побудови рекомендацій товарів за їхніми категоріями та за чорним списком, виведення цілої моделі даних у структурованому виді, та інтеграція засобів безпеки в чутливих функціях діагностики роботи системи.

Для подальшої роботи розглядається введення адаптивності до системи, можливі методи якого були виведеними під час проєктування алгоритму, для зменшення часу відклику між появами трендових комбінацій товарів та їх внесення в рекомендаційну систему для аналізу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Висоцький Н. І. Легковагові керовані бази даних в хмарній інфраструктурі : робота на здобуття кваліфікаційного ступеня бакалавра : спец. 121 - інженерія програмного забезпечення / наук. кер. Ю. Б. Гладьо. Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2025. 51 с.

2. Головко А. Дослідження та розробка рекомендаційної системи на основі штучного інтелекту для вибору стійких рослин для озеленення міста в умовах війни / А. Головко, Л. Матійчук // Матеріали XIII науково-технічної конференції „Інформаційні моделі, системи та технології“, 17-18 грудня 2025 року. — Т. : ТНТУ, 2025. — С. 38–39.

3. Грибан В.Г. Безпека життєдіяльності та охорона праці : підручник / В.Г. Грибан, А.Є. Фоменко, Д.Г. Казначеєв. – Дніпро : ДДУВС, 2022. – 388 с.

4. Сокуренко В. В., Бандурка О. М. Безпека життєдіяльності та охорона праці : підручник. Харків : ХНУВС, 2021. 308 с.

5. Стиценко Т. Є., Пронюк Г. В., Сердюк Н. М. Безпека життєдіяльності : навч. посібник. Харків : ХНРУЕ, 2018. 336 с.

6. Карбівська У. М., Майстер М. Д. Основи охорони праці : навчально-методичний посібник. Івано-Франківськ : НАІР, 2020. 182 с.

7. Кварамба Р. Р. Методи аналізу та процесу великих даних у створенні системи рекомендацій для інтернет-магазину : магістерська кваліфікаційна робота «123 — Комп’ютерна інженерія» / Кварамба Рувімбо Рона. – Тернопіль : ТНТУ, 2021. – 56 с.

8. Коваль В. В. Розробка веб-інтерфейсу для інтерактивної візуалізації API даних на основі Java-Script : робота на здобуття кваліфікаційного ступеня магістра : спец. 121 - інженерія програмного забезпечення / наук. кер. Г. Б. Цуприк. Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024. 76 с.

9. Яцишин В. А. Аналіз методів тестування програмних інтерфейсів для RESTFul API : робота на здобуття кваліфікаційного ступеня магістр : спец. 122 - комп’ютерні науки / наук. кер. Л. П. Дмитроца. Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2025. 48 с.

10. Recommendation Systems: Marketing Applications, Benefits and Risks. *Multidisciplinary Approaches to Contemporary Marketing* / F. N. Ozkan. Palgrave Macmillan, Cham, 2025. ISBN 978-3-031-78026-4. URL: https://doi.org/10.1007/978-3-031-78026-4_3.

11. A Comprehensive Guide to Use Case Modeling. *Visual Paradigm*. 12.09.2023. URL: https://guides.visual-paradigm.com/a-comprehensive-guide-to-use-case-modeling/.

12. Matrix Factorization. *Google for Developers*. *Machine Learning*. 25.08.2025. URL: https://developers.google.com/machine-learning/recommendation/collaborative/matrix.

13. RFC 8259. The JavaScript Object Notation (JSON) Data Interchange Format. Чинний від 13.12.2017. Internet Engineering Task Force. 16 с. URL: https://www.rfc-editor.org/info/rfc8259.

14. Building RESTful APIs in Rust: A Comprehensive Guide. *Software Patterns Lexicon*. 25.11.2024. URL: https://softwarepatternslexicon.com/rust/ enterprise-integration-patterns/implementing-restful-apis/.

15. Actor-Based Concurrency in Rust. *Sling Academy*. *Rust*. 06.01.2025. URL: https://www.slingacademy.com/article/actor-based-concurrency-in-rust-introducing-the-actix-ecosystem/.

16. The Manifest Format. *The Cargo Book*. *Cargo Reference*. URL: https://doc.rust-lang.org/cargo/reference/manifest.html.

17. Error Handling. *The Rust Programming Language* / S. Klabnik, C. Nichols, C. Krycho. 3-тє вид. No Starch Press, 2026. С. 167–186. ISBN 9781718504448. URL: https://doc.rust-lang.org/book/ch09-00-error-handling.html.

18. @Chonyy. FP Growth: Frequent Pattern Generation in Data Mining with Python Implementation. *towards data science*. 30.10.2020. URL:

https://towardsdatascience.com/fp-growth-frequent-pattern-generation-in-data-mining-with-python-implementation-244e561ab1c3/.

19. Baig A. 15 Essential Data Mining Techniques. *Dataversity*. 04.12.2023. URL: https://www.dataversity.net/articles/15-essential-data-mining-techniques/.

20. Balalaie A., Heydarnoori A., Jamshidi P. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software*. 2016. Т. 33, вип. 4. С. 42–52. ISSN 1937-4194.

21. Blandy J., Onderdoff J., Tindall L. Programming Rust: Fast, Safe Systems Development. 2-ге вид. O'Reilly Media, 2021. ISBN 9781492052593.

22. Booz R., Fritchey G. Introduction to PostgreSQL for the data professional. Redgate Books, 2025. 348 с. ISBN 978-1036902377.

23. Eastham J. Introduction to SQLx. *Rust for .NET Developers*. *Persisting Data*. URL: https://rustfor.net/docs/module8/sqlx/.

24. Elahi M., Ricci F., Rubens N. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*. 2016. Т. 20. URL: https://doi.org/10.1016/j.cosrev.2016.05.002.

25. Erl T. Service-Oriented Architecture: Analysis & Design for Services and Microservices. 2-ге вид. Prentice Hall, 2018. 393 с. ISBN 0133858588.

26. Fox C. Data Science for Transport. Springer Cham, 2018. 185 с. ISBN 978-3-319-72952-7. URL: https://doi.org/10.1007/978-3-319-72953-4.

27. Gandhi R., Richards M., Ford N. Head First Software Architecture. 2-ге вид. O'Reilly Media, 2024. 483 с. ISBN 9781098134358.

28. Hartle III F., Mancini S., Kerry E. Data poisoning 2018–2025: A systematic review of risks, impacts, and mitigation challenges. *Issues in Information Systems*. 2025. Т. 25, вип. 4. С. 433–422. URL: https://doi.org/10.48009/4_iis_2025_135.

29. Heaton J. Comparing Dataset Characteristics that Favor the Apriori, Eclat or FP-Growth Frequent Itemset Mining Algorithms. *arXiv*. *Computer Science - Databases*. 30.01.2017. URL: https://doi.org/10.48550/arXiv.1701.09042.

30. Applications of Mathematics in Science and Technology / B. T. Hung та ін. CRC Press, 2025. 1058 с. URL: https://doi.org/10.1201/9781003606659.

31. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. 2-ге вид. O'Reilly Media, 2017. 614 с. ISBN 978-1449373320.

32. Leung C. K.-S., Jiang F., Dela Cruz E. M. Association Rule Mining in Collaborative Filtering. *Collaborative Filtering Using Data Mining and Analysis*. IGI Global, 2017. С. 159–179. ISBN 9781522504894. URL: https://doi.org/10.4018/978-1-5225-0489-4.ch009.

33. Mao C., Huang S., Sui M. Analysis and Design of a Personalized Recommendation System Based on a Dynamic User Interest Model. *arXiv*. *Computer Science - Information Retrieval*. 13.10.2024. URL: https://doi.org/10.48550/arXiv.2410.09923.

34. Pedro B. Building an API Product : design, implement, release, and maintain API products that meet user needs. Packt Publishing, 2024. 278 с. ISBN 9781837638536.

35. Prabu V. Big Data Architecture. *KaaShiv InfoTech Blog*. 27.11.2023. URL: https://www.kaashivinfotech.com/blog/big-data-architecture/.

36. Raza S., Rahman M., Kamawal S. A comprehensive review of recommender systems: Transitioning from theory to practice. *Computer Science Review*. 2026. Т. 59. ISSN 1574-0137. URL: https://doi.org/10.1016/j.cosrev.2025.100849.

37. Reis J., Housley M. Fundamentals of Data Engineering: Plan and Build Robust Data Systems. 2-ге вид. O'Reilly Media, 2022. 447 с. ISBN 978-1098108304.

38. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020. 422 с. ISBN 9781492043454.

39. Rodriguez M. GitHub Copilot is moving to usage-based billing. *Github Blog* 27.04.2026. URL: https://github.blog/news-insights/company-news/github-copilot-is-moving-to-usage-based-billing/.

40. Roy D., Dutta M. A systematic review and research perspective on recommender systems. *Journal of Big Data*. 2022. Т. 9. ISSN 2196-1115. URL: https://doi.org/10.1186/s40537-022-00592-5.

41. Singh P. K., Othman E. Optimized recommendations by user profiling using apriori algorithm. *Applied Soft Computing*. 2021. T. 106. ISSN 1568-4946. URL: https://doi.org/10.1016/j.asoc.2021.107272.

42. Souza V. M. A., Reis D., Maletzke A. G. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*. 2020. T. 34. C. 1805–1858. URL: https://doi.org/10.1007/s10618-020-00698-5.

43. Tan P.-N., Steinbach M., Kumar V. Association Analysis: Advanced Concepts. *Introduction to Data Mining*. Pearson, 2018. C. 328–390. ISBN 978-0133128901.