Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

за другим (магістерським) рівнем вищої освіти

на тему

“Дослідження та програмна реалізація технології побудови рекомендаційної системи для просування товарів і послуг компанії в мережі Інтернет”

Виконав здобувач вищої освіти
ІІ курсу, групи КН22М-1
ОПП «Комп'ютерні науки»
спеціальності 122 «Комп'ютерні науки»
_____ Михайлов Д.С.
«____» _____ 2023р.

Керівник проекту
кандидат технічних наук, доцент
_____ Пархоменко Ю.М.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *магістр*
Галузь знань *12 "Комп'ютерні науки"*
Спеціальність *122 "Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерні науки"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
"____" _____2023_року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Михайлову Дмитру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація технології побудови рекомендаційної системи для просування товарів і послуг компанії в мережі Інтернет*

2. Керівник роботи *Босько Віктор Васильович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 32-13 від 04.08.23

3. Строк подання роботи до захисту *8.01.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є програмне дослідження та програмна реалізація рекомендаційної систем*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В., к.т.н., доцент	09.11.2023 р.	17.11.2023 р.
Охорона праці	Оришака О.В., к.т.н., доцент	03.11.2023 р.	21.11.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	12.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	18.10.2023 р.	
3.	Розробка моделі компонента	23.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	32.10.2023 р.	
6.	Програмування алгоритмів	11.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	16.11.2023 р.	
9.	Оформлення ПЗ	18.11.2023 р.	
10.	Попередній захист роботи	04.12.2023 р	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Михайлов Д.С. Дослідження та програмна реалізація технології побудови рекомендаційної системи для просування товарів і послуг компанії в мережі Інтернет. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній магістерській роботі розроблено та досліджено метод побудови рекомендаційних систем. Проаналізовано реалізацію цих підходів на прикладі відомих інтернет-сервісів з каталогом товарів.

Розроблено веб-додаток з використанням Rails та імплементацією рекомендаційної системи на базі колаборативної фільтрації для більш ефективного залучення користувачів до роботи з реалізованим веб-додатком.

Метою роботи є широкий аналіз існуючих підходів до створення рекомендаційних систем, та реалізація додатку із вбудованою системою рекомендацій та підвищення ефективності роботи рекомендаційної системи.

Об'єктом дослідження – процес формування особистих рекомендацій на основі вподобань користувачів.

Предметом дослідження є методи та моделі формування особистих рекомендацій.

Методи дослідження базуються на методах інформаційного пошуку, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація сайту компанії з вбудованою рекомендаційною системою.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача.

Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10/11.

Програму розроблено в середовищі Ruby(Ruby on Rails) та СУБД MySQL та MongoDB.

Ключові слова: рекомендаційні системи, рекомендації, прогнозування, інтернет-маркетинг, колаборативна фільтрація, контентна фільтрація, веб-додаток, Rails.

ABSTRACT

Mikhailov D.S. Research and software implementation of the technology of building a recommendation system for the promotion of goods and services on the Internet. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this master's thesis, the method of building recommendation systems is developed and researched. The implementation of these approaches was analyzed using the example of well-known Internet services with a product catalog.

A web application was developed using Rails and the implementation of a recommender system based on collaborative filtering for more effective involvement of users in working with the implemented web application.

The purpose of the work is a broad analysis of existing approaches to the creation of recommendation systems, and the implementation of an application with a built-in recommendation system and improvement of the effectiveness of the recommendation system.

The object of the study is the process of forming personal recommendations based on user preferences.

The subject of the research is the methods and models of forming personal recommendations.

Research methods are based on the methods used in this work, based on information search methods, software development methods.

The result of the work is the software implementation of the company's website with a built-in recommendation system.

In the process of working on the software model, an analysis of existing hardware and software was performed.

All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows XP/Vista/7/8/10/11.

The program was developed in the environment of Ruby (Ruby on Rails) and DBMS MySQL and MongoDB.

Keywords: recommender systems, recommendations, forecasting, internet marketing, collaborative filtering, content filtering, web application, Rails.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ ТА СПЕЦІАЛЬНИХ ТЕРМІНІВ

РС – Рекомендаційна Система

КФ – Колаборативна Фільтрація

CF – Collaborative Filtering

CBF – Content-Based Filtering

SVD – Singular Value Decomposition

TF-IDF – Term Frequency - Inverse Document Frequency

ORM – Object Relational Mapping

БД – база даних

ПЗ – програмне забезпечення

ВСТУП

Актуальність теми. Протягом тривалого періоду часу у всьому світі значно зросла кількість інформації зі швидкістю блискавки. Люди щодня зустрічають і фільтрують потік інформації, що надходить з різних джерел: робота, побутові проблеми, популярні джерела інформації і так далі. Після винайдення Інтернету обсяг інформації стрімко збільшився, і з'явилося безліч сервісів, які мають за мету забезпечити користувачів усім необхідним для комфортного життя.

Останнім десятиліттям набули великої популярності інтернет-сервіси, які пропонують товари всіх можливих видів (інтернет-магазини), інформацію на будь-який смак (інтернет-журнали, новини, книги, статті) і т. д. Однак великий обсяг інформації ускладнює вибір користувачів навіть при наявності вбудованого пошуку та фільтрації.

Рекомендаційні системи виникли як засіб полегшення пошуку та покупок на веб-сайтах, замінюючи статичні списки рекомендацій. Ці системи формують рейтинг об'єктів (товарів, фільмів, музичних композицій) на основі різних критеріїв, таких як релевантність, популярність, історія оцінок і т. д. Інтернет-компанії широко використовують такі системи в інтернет-маркетингу, сподіваючись залучити більше користувачів до своїх сервісів. Також розробники рекомендаційних систем можуть використовувати довірені рекомендації для включення промоційних товарів серед рекомендацій для користувачів

Мета й завдання дослідження. Метою роботи є широкий аналіз існуючих підходів до створення систем, що прогнозують рекомендації, та реалізація додатку із вбудованою системою рекомендацій.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– розглянути відомі базові підходи, що використовуються при розробці системи, що надає рекомендації, та методи кореляції результатів роботи таких систем;

– проаналізувати реалізацію цих підходів на прикладі відомих інтернет-сервісів з каталогом товарів;

– розробити веб-додаток з використанням Rails та імплементацією рекомендаційної системи на базі колаборативної фільтрації для більш ефективного залучення користувачів до роботи з реалізованим веб-додатком;

Об'єктом дослідження є процес формування особистих рекомендацій на основі вподобань користувачів.

Предметом дослідження є методи та моделі формування особистих рекомендацій.

Методи дослідження базуються застосовані у даній роботі, базуються на методах інформаційного пошуку.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– проаналізовано основні алгоритми та базові підходи для прогнозування рекомендацій, описано особливості, переваги та недоліки кожного з них та виконано порівняльний аналіз існуючих підходів;

– розроблено підхід, який поєднує в собі фільтрацію даних, на основі вмісту з колаборативною та демографічною фільтрацію;

– зроблено огляд допоміжних засобів для кореляції результатів, отриманих після застосування основних алгоритмів;

– розроблено вітчизняний продукт управління рекомендаційною системою, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для реалізації рекомендаційних систем. Були розглянуті шляхи до подолання типових недоліків базових алгоритмів. Для вирішення недоліків колаборативної фільтрації

використовується алгоритм SVD. Адже з ростом кількості користувачів та їх оцінок в будь-якому інтернет-сервісі буде зростати розмірність матриці оцінок. Цей алгоритм дозволяє зменшити розмірність цієї матриці з невеликою втратою точності.

Також було виконано порівняльний аналіз існуючих рішень імлементації рекомендаційних систем у відомі інтернет-компанії, які збільшили свої прибутки саме через інтеграцію рекомендаційних систем до каталогу своїх продуктів та сервісів. Було проаналізовано рекомендаційні системи та модулі широковідомих компаній Amazon та eBay.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Апробація роботи відбулася на VII Міжнародній науково-практичній конференцію до 30-ти річчя кафедри кібербезпеки та програмного забезпечення «Інформаційна безпека та комп’ютерні технології» м.Кропивницький.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація рекомендаційної, є актуальною задачею, яка потребує вирішення у даній магістерській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

У світлі сучасних технологій та розширення інформаційного простору розробка рекомендаційних систем стає ключовою складовою впровадження персоналізованих сервісів. В умовах величезних обсягів інформації, що перевищують можливості індивідуального освоєння, рекомендаційні системи набувають значущості як важливий інструмент для допомоги користувачам у знаходженні контенту, який відповідає їхнім унікальним вподобанням та потребам.

Магістерська робота націлена на дослідження та розробку ефективної рекомендаційної системи, використовуючи передові методи машинного навчання та аналізу даних. У контексті постійного росту обчислювальної потужності та доступу до обширних обсягів інформації завдання розробки рекомендаційних систем стає як складним, так і захоплюючим напрямком досліджень.

Основна мета роботи - вивчити та реалізувати сучасні підходи до рекомендаційних систем, ураховуючи їх переваги та обмеження. Особливу увагу приділено розробці моделей, які можуть адаптуватися до змін у вподобаннях користувачів і ефективно реагувати на зміни в даних. Додатково, досліджено можливості глибокого навчання та вдосконалення алгоритмів з урахуванням контексту взаємодії з користувачами.

Ця магістерська робота спрямована на внесення суттєвого внеску в розуміння та практику рекомендаційних систем, розкриваючи нові можливості для покращення персоналізації інтернет-сервісів та підвищення якості користувацького досвіду.

1.1. Призначення системи

Під час виконання роботи було проведено аналіз основних алгоритмів та базових підходів для прогнозування рекомендацій. Описано унікальні особливості, переваги та недоліки кожного із них, а також виконано порівняльний аналіз наявних підходів. Додатково був проведений огляд допоміжних інструментів для кореляції отриманих результатів після використання основних алгоритмів. Для створення платформи були використані сучасні бібліотеки та бази даних з відкритим вихідним кодом. З метою продемонструвати проведений аналіз базових алгоритмів, був розроблений веб-додаток з інтегрованою рекомендаційною системою на основі колаборативної фільтрації. В якості платформи для створення інтернет-сервісу було обрано фреймворк Ruby on Rails, який дозволяє ефективно створювати проекти високої складності, використовуючи різноманітні модулі, як внутрішні, так і зовнішні.

Метою роботи є розробка та реалізація веб-додатоку з рекомендаційною системою на базі колаборативної фільтрації.

Система повинна забезпечити такі вимоги:

- вирішує основні задачі, що висуваються до подібних систем, тобто реалізує вищевказані підсистеми, з урахуванням загальноприйнятих підходів до розробки розподілених рекомендаційних систем;

- розроблена з урахуванням кращих практик і підходів до архітектури програмного забезпечення [8-13], що дають можливість забезпечити функціональні та нефункціональні вимоги до системи;

- має специфічну функціональність, якої не мають інші системи (наприклад, надійна доставка повідомлень між пристроями й адміністраторами та засоби моніторингу пристроїв).

1.2. Область застосування

У контексті сучасних технологій та розширення інформаційного простору, розробка рекомендаційних систем відіграє ключову роль у впровадженні персоналізованих сервісів. Рекомендаційні системи є ефективним інструментом, що надає користувачам індивідуалізовані поради та рекомендації, ґрунтуючись на їхніх вподобаннях та діяльності. У даній магістерській роботі досліджується процес створення рекомендаційної системи, включаючи наступні етапи:

- збір даних - здійснення збору інформації про користувачів та об'єкти, для яких створюються рекомендації. Це може включати історію покупок, переглядів, оцінок, пошукові запити та інші відомості;
- підготовка даних - це обробка та очищення зібраних даних, включаючи вилучення дублікатів, відсутніх значень та аномалій. Створення матриці відношень між користувачами та об'єктами;
- вибір алгоритму – це вибір алгоритму або методу, який найкраще відповідає поставленій задачі. Популярні алгоритми включають колаборативну фільтрацію, факторизацію матриць, контент-базовані методи, глибокі нейронні мережі та інші;
- побудова моделі – це навчання моделі рекомендаційної системи на підготовлених даних. Це може включати обчислення схожості між користувачами та об'єктами, навчання ваг параметрів моделі тощо;
- оцінка та налуштування моделі - використання метрики, таких як середній квадратичний корінь помилки (RMSE), точність, відхилення тощо, для оцінки якості моделі. Налаштування моделі та алгоритму для досягнення найкращих результатів;
- генерація рекомендацій - використання навченої моделі для генерації рекомендацій для користувачів на основі їхніх інтересів та історії взаємодій;
- виведення рекомендацій - представити рекомендації користувачам, наприклад, на веб-сайті, в мобільному додатку або через електронну пошту.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація рекомендаційної системи є актуальною задачею, яка потребує вирішення у даній магістерській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

Рекомендаційні системи виникли на сучасному ринку ІТ як заміна статичним спискам рекомендацій під час пошуку або покупок на веб-сайтах. Ці системи створюють рейтинг об'єктів (товарів, фільмів, музичних композицій) на основі різних критеріїв: релевантність, популярність, історія оцінок тощо. Існуючі інтернет-компанії широко використовують такі системи в інтернет-маркетингу з метою залучення користувачів до конкретного сервісу. Крім того, при розробці довіреної рекомендаційної системи можна включати серед рекомендацій інші товари, що рекламуються.При розробці рекомендаційних систем зазвичай розробники стикаються з рядом проблем прогнозування:

- розрідженість даних (більшість користувачів не ставить оцінки товарам, отже дані з попередніми оцінками являють собою розріджену матрицю);
- холодний старт (робота з новими користувачами або товарами);
- синонімія (проблема розпізнавання схожих товарів з різними назвами);
- шахрайство (цілеспрямоване завищення рейтингів певних товарів їх власниками);
- розмаїття (при великій вибірці нові або маловідомі товари мають низькі позиції в рейтинговому списку);
- білі ворони (унікальні користувачі, смаки яких дуже важко обробити, оскільки вони не співпадають зі смаками відокремлених типів).

Отже, як можна помітити, проблем створення системи прогнозування для інтернет-сервісів чимало, тож дуже цікаво подивитися як саме різні підходи до створення рекомендаційних систем вирішують ці проблеми та покращують якість рекомендацій.

Метою даної магітерської роботи є аналіз існуючих рекомендаційних систем для інтернет-маркетингу, дослідження рекомендаційних алгоритмів та виявлення їх переваг та недоліків.

Також, в якості прикладу застосування системи прогнозування рекомендацій буде розроблено веб-додаток за допомогою Rails, PostgreSQL та Redis, де буде протестовано якість рекомендацій.

Отримані результати дослідження пропонується використовувати в якості методичного матеріалу розробниками ПЗ під час проектування та розробки рекоменаційних систем.

2.1. Огляд існуючих систем

Аналіз рекомендаційних алгоритмів

Рекомендаційна система - це механізм, який надає користувачам персоналізовані рекомендації серед обширного обсягу інформації відповідно до їхніх потреб. Об'єктами в такій системі можуть бути різні елементи сервісу, такі як фільми, ресторани, книги чи статті. Інтереси користувачів можуть бути представлені за допомогою оцінок, які вони присвоюють товарам, або ключових слів, які характеризують кожен об'єкт.

Щоб зберігати вподобання користувачів стосовно товарів, РС використовують профілі користувачів. У більшості РС профіль користувача містить набори оцінок та/або ключових слів (тегів). Оцінки, надані користувачами товарам, можуть належати різним проміжкам (0-1, 1-5, 1-10): чим вищий рейтинг, тим більше конкретний товар сподобався користувачу. Після кожного оцінювання все рейтинги користувача агрегуються через ряд обчислень, вимірюються схожість користувачів, а потім прогнозуються рекомендації для даного користувача. Ключові слова автоматично підвантажуються з текстів або товарів, які користувачі проглядали або оцінювали в минулому. Вони також можуть мати ваги в залежності від того, на скільки користувач оцінив конкретне слово, або більш значущі слова матимуть більшу вагу, ніж менш значущі (алгоритм TF-IDF). Після цього тексти (товари) зіставляються з профілем користувача та найбільш відповідающі йому – рекомендуються. Рейтинги можуть бути явними та неявними. Явна оцінка – це оцінка, якою користувач

показав зацікавленість даним товаром в межах своєї системи оцінювання. Неявні рейтинги вираховуються з історії покупок або поведінки користувачів. До їх переваг можна віднести зниження навантаження на користувача оцінюванням товарів.

Джерелом неявних рейтингів можуть бути час, витрачений на читання статті, посилання на товар в інших джерелах (наприклад алгоритм ранжування сторінок Google). Інші індикатори поведінки перегляду, як рух курсору, ввід клавіатури та швидкість прокрутки сторінки, також були досліджені в якості неявних показників інтересу та показали непогані результати.

Загальна архітектура рекомендаційної системи може бути представлена наступним чином:

1) Довідкові дані (background data) – системна інформація про товари сервісу, що збирається, як правило, в фоновому режимі.

2) Вхідні дані (input data) – інформація, яку користувач вносить в систему, щоб отримати рекомендації.

3) Рекомендаційний алгоритм - алгоритм, що комбінує системну інформацію та вхідні дані для отримання рекомендацій. Типові системи в якості довідкових даних використовують профілі користувачів, а в якості вхідних – дії користувача (оцінювання товарів, час, проведений на сторінці, тощо).

Базові підходи розробки рекомендаційних систем. У більшості рекомендаційних систем використовується один з двох базових підходів - колаборативна фільтрація (collaborative filtering) та контентна фільтрація (content-based filtering). Також існує клас підходів, що базуються на поєднанні двох основних – гібридна фільтрація (hybrid filtering).

Колаборативна фільтрація. Метод прогнозу в рекомендаційних системах, який використовує відомі переваги (оцінки) групи користувачів для прогнозування невідомих переваг (оцінок) іншого користувача. За допомогою цього алгоритму будується певна таблиця користувачів, які групуються за схожістю, та прогнозуються результати для інших користувачів. [2]

Колаборативна фільтрація прогнозує рекомендації, засновані на моделі попередньої поведінки користувача.

Ця модель може бути побудована виключно на основі поведінки цього користувача або - що більш ефективно - з урахуванням поведінки інших користувачів з подібними характеристиками.

У тих випадках, коли колаборативна фільтрація бере до уваги реакцію інших користувачів, вона використовує знання про групу (group knowledge) для вироблення рекомендацій на основі схожості користувачів. По суті рекомендації базуються на автоматичній взаємодії множини користувачів і на виділені (методом фільтрації) тих користувачів, які демонструють схожі уподобання або шаблони поведінки. Наприклад, при створенні веб-блогу, на якому необхідно запровадити рекомендаційну систему на основі інформації від багатьох користувачів, які підписуються на блоги і читають їх, можна згрупувати цих користувачів за їх інтересами.

Можна об'єднати в одну групу користувачів, які читають кілька однакових блогів і за цією інформацією ідентифікувати найпопулярніші блоги серед тих, які читають учасники цієї групи. Потім конкретному користувачу з цієї групи будуть рекомендуватися найпопулярніший блог з тих, на які він ще не підписаний. Осередок на перетині рядка блогу і строки користувача містить кількість статей, прочитаних цим користувачем в цьому блозі. Кластеризація користувачів на основі читацьких вподобань (наприклад, за допомогою алгоритму найближчих сусідів) дозволяє виділити два кластери, кожен з яких містить по два користувача.

Інший спосіб розгляду цих відносини заснований на їх схожості та відмінності, як ілюструє діаграма Венна.

Схожість визначають, за якими ознаками (за допомогою відповідного алгоритму) слід згрупувати користувачів. Відмінності - це можливості, які можуть бути використані для вироблення рекомендацій - наприклад, за допомогою застосування фільтра популярності.

Переваги: швидка робота алгоритмів (K-based та ін.) - мала кількість ітерацій; прості в реалізації.

Недоліки: не вирішені проблеми холодного старту, шахрайства, нема що рекомендувати новим або нетиповим користувачам; розріджені матриці оцінок (іноді неможливо зробити прогноз).

Алгоритм SVD є ефективним методом колаборативної фільтрації, призначеним для подолання ряду недоліків традиційних алгоритмів. Нижче подано основні етапи роботи цього алгоритму:

- матриця Рейтингів (R): Починаємо з матриці рейтингів, де рядки представляють користувачів, а стовпці - об'єкти (наприклад, фільми). Елемент R[i, j] містить рейтинг користувача i для об'єкту j або нуль, якщо користувач не має взаємодії з цим об'єктом;
- розклад матриці (U, Σ, V): Мета полягає в розкладі матриці R на три інші матриці: U (матриця користувачів), Σ (діагональна матриця сингулярних значень) та V (матриця об'єктів);
- матриця користувачів (U): Матриця U містить ваги користувачів для латентних факторів. Кожен рядок відповідає одному користувачеві, а стовпці латентним факторам;
- діагональна матриця сингулярних значень (Σ): Σ є діагональною матрицею, яка містить сингулярні значення, вказуючи на важливість кожного латентного фактора;
- матриця об'єктів (V): Матриця V містить ваги об'єктів для латентних факторів. Кожен рядок відповідає одному об'єкту, а стовпці - латентним факторам;
- підготовка матриці Σ: Зазвичай сингулярні значення сортуються в порядку спадання, а лише деякі залишаються, щоб скоротити розмірність матриці Σ;

- обчислення рекомендацій: Застосовують розкладані матриці U, Σ і V, обчислюючи добуток U * Σ * V. Це дає оцінки рейтингів для всіх користувачів і об'єктів;

- вибір рекомендацій. Для конкретного користувача можна вибрати та відсортувати об'єкти з найвищими оцінками, щоб знайти рекомендації для нього.

- оцінка результатів. Використовуються метрики якості (наприклад, RMSE або інші) для оцінки ефективності моделі та точності рекомендацій;

Проте алгоритм SVD також має свої проблеми:

- рекомендація новим користувачам. Для нових користувачів може бути складно знайти відповідний кластер зі схожими користувачами;

- рекомендація для нетипового користувача: Користувачі розподіляються за якимись класами, що може призвести до втрати індивідуальних особливостей;

- неможливість прогнозування, якщо в кластері ніхто не оцінив об'єкт: Якщо в кластері відсутні оцінки для певного об'єкта, зробити передбачення може бути важко.

Матриця Рейтингів: Почнемо з матриці рейтингів R, де рядки відповідають користувачам, а стовпці - об'єктам (наприклад, фільмам в системі рекомендацій для фільмів). Елемент R[i, j] містить рейтинг, який користувач і надав об'єкту j, або нуль, якщо користувач не взаємодіяв з цим об'єктом.

– **Розклад матриці**: Мета полягає в розкладі матриці R на три інші матриці: U, Σ та V. Де U - матриця користувачів, Σ - діагональна матриця сингулярних значень, а V - матриця об'єктів.

– **Матриця користувачів (U)**: Матриця U містить ваги користувачів для латентних факторів. Кожен рядок матриці U відповідає одному користувачеві, а стовпці - латентним факторам.

– **Діагональна матриця сингулярних значень (Σ)**: Σ є діагональною матрицею, яка містить сингулярні значення. Вони вказують на важливість кожного латентного фактора.

- **Матриця об'єктів (V)**: Матриця V містить ваги об'єктів для латентних факторів. Кожен рядок відповідає одному об'єкту, а стовпці - латентним факторам.

– **Підготовка матриці Σ**: Зазвичай, сингулярні значення сортуються в порядку спадання, і лише деякі з них залишаються, тим самим скорочуючи розмірність матриці Σ.

– **Обчислення рекомендацій**: Щоб знайти рекомендації для користувача, використовуються розкладані матриці U, Σ і V. Для цього обчислюють добуток U * Σ * V, який дає оцінки рейтингів для всіх користувачів і об'єктів.

– **Вибір рекомендацій**: Для конкретного користувача можна вибрати та відсортувати об'єкти з оцінками, які найвищі, щоб знайти рекомендації для нього.

– **Оцінка результатів**: Використовуйте метрики якості (наприклад, RMSE або інші) для оцінки того, наскільки добре ваша модель працює і наскільки точні її рекомендації.

Проблеми алгоритму:

- рекомендація новим користувачам. Для таких користувачів не знайдется відповідного кластера зі схожими на них користувачами;

- рекомендація для нетипового користувача. Ми ділимо усіх користувачів на якісь класи (шаблони);

- якщо в кластері ніхто не оцінив об’єкт, то зробити передбачення не вийде.

Контентна фільтрація

Тематична фільтрація формує рекомендацію на базі поведінки користувача. Наприклад, цей підхід може використовувати ретроспективну

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

Алгоритм TF-IDF

Алгоритм TF-IDF розшифровується як “частота терміна” (term frequency) – “зворотня частота зустрічання терміна в документі” (inverse document frequency). TF-IDF – статистичний показник, що використовується для оцінки важливості слів у контексті документа, що є частиною колекції документів. Вага (значимість) слова пропорційна кількості вживань цього слова у документі, і обернено пропорційна частоті вживання цього слова у інших документах. Наприклад, запит “The Civil War” буде розбито на слова: “the”, “civil” та “war”. Вага слова “the” буде наближуватися до нуля, оскільки в будь-якій статті англійською мовою воно зустрічається дуже часто. Два інших слова матимуть більшу вагу при пошуку документа у колекції. Показник TF-IDF використовується в задачах аналізу текстів та інформаційного пошуку. Його можна застосовувати як один з критеріїв релевантності документа до пошукового запиту, а також при розрахунку міри спорідненості документів при кластеризації.

Гібридні алгоритми фільтрації

Гібридні підходи об'єднують колаборативну і контентну фільтрацію для підвищення ефективності рекомендаційних систем. Об'єднання результатів цих двох підходів дозволяє підвищити точність рекомендацій. Гібридні підходи можуть бути особливо корисними в умовах сильно розріджених даних,

наприклад, при холодному старті, коли недостатньо інформації для використання чисто колаборативного підходу. Гібридні системи можуть використовувати контентні дані для зважування результатів колаборативної фільтрації та підвищення релевантності рекомендацій.

Моделі гібридних систем. Гібридні рекомендаційні системи об'єднують різні підходи, такі як колаборативна і контентна фільтрація. Їх переваги включають велику швидкодію та здатність до досягнення кращих результатів у порівнянні з базовими моделями.

Переваги. Гібридні системи можуть забезпечувати ефективну та швидку генерацію рекомендацій, особливо у порівнянні з деякими іншими складними моделями. Комбінування різних підходів дозволяє досягти більш точних та релевантних рекомендацій для користувачів.

Недоліки. Реалізація гібридних рекомендаційних систем може бути дорогим та часовитратним завданням. Розробникам потрібно поєднати різні алгоритми та моделі, що може вимагати значних зусиль і ресурсів.

Інтеграція різних компонентів гібридних систем може зробити їх вразливими до змін. Навіть невеликі зміни в роботі алгоритмів можуть призвести до змін у загальному функціонуванні системи, що може бути важко підтримувати.

Переваги базових алгоритмів. Базові алгоритми зазвичай є простими для реалізації і розуміння. Ефективність при роботі з невеликою кількістю даних: Деякі базові алгоритми можуть бути ефективними, коли даних недостатньо для складніших методів.

Недоліки базових алгоритмів. Проблеми холодного старту. Базові алгоритми можуть важко вирішувати проблеми, пов'язані з холодним стартом, коли для нових користувачів або об'єктів немає достатньої інформації. У порівнянні з більш складними методами, базові алгоритми можуть демонструвати меншу точність в рекомендаціях, особливо в умовах складних та розріджених даних.

Моделі гібридних систем

Таблиця 2.1 – Моделі рекомендаційних систем на базі гібридних алгоритмів

Тип	Характеристика
Зважена	Оцінки, отримані методами колаборативної та контентної фільтрації, агрегуються для отримання єдиної рекомендації.
Комутуюча	Система використовує критерії, щоб перемикатися між методами контентної та колаборативної фільтрації.
Каскадна	Рекомендація такої моделі з'являється за рахунок удосконалення рекомендації інших систем.
Комбінаційна	Критерії з різних джерел рекомендацій вкидаються в єдиний рекомендаційний алгоритм.
Нарощувальна	Вихідні дані, отримані з однієї системи, використовуються як вхідні для іншої.
Змішана	Рекомендації з різних рекомендаційних систем показуються одночасно
Мета-рівень	Навчена модель з однієї рекомендаційної системи використовується в якості системних даних для іншої системи.

Переваги: велика швидкодія; кращі результати.

Недоліки: дуже дорога розробка рекомендаційної системи, оскільки реалізація цього типу алгоритмів дуже складна; важко підтримувати, оскільки навіть незначні зміни в роботі призводять до змін роботи алгоритму.

Порівняльна характеристика базових рекомендаційних алгоритмів Базові рекомендаційні алгоритми мають переваги та недоліки. Їх список наведено у таблиці 2.2.

Таблиця 2.2 – Переваги та недоліки базових рекомендаційних алгоритмів

	Переваги	***Недоліки***	***Застосування***
Колаборативна фільтрація	1. Рекомендації незалежні від контенту сервісу 2. Використовується якість оцінок та смаки користувачів 3. Інтуїтивно прозорий	1. Проблема холодного старту (першого користувача) 2. Розрідженість матриці оцінок	• Блоги • Інформаційні портали
Контентна фільтрація	1. Немає проблеми холодного старту 2. Немає проблеми розрідженості даних	1. Залежна від контенту 2. Ніяк спиратися на смаки та якість оцінок користувачів 3. Вузькоспрямована	• Інтернет-магазини • Інтернет-аукціони • Блоги
Гібридна фільтрація	1. Майже відсутні базові проблеми CF та CBF 2. Висока швидкодія	1. Висока складність розробки 2. Складна підтримка 3. Вузькоспрямована	• Великі інтернет-магазини • Складні соціальні мережі

Допоміжні алгоритми, що використовуються рекомендаційними системами

Конкурс Netflix Prize, який відбувався в період з 2006 по 2009 рік, був спрямований на поліпшення рекомендаційної системи для стрімінгового сервісу Netflix. Організатори конкурсу обіцяли винагороду у вигляді мільйона доларів команді, яка зможе покращити точність їхньої рекомендаційної системи на 10%.

Цей конкурс наочно демонструє, що для досягнення високої точності у рекомендаційних механізмах можна використовувати різноманітні алгоритми. Декілька ключових аспектів, які виникли з цього конкурсу та застосовуються в рекомендаційних системах, включають:

- машинне навчання та алгоритми прогнозування: Учасники конкурсу використовували широкий спектр алгоритмів машинного навчання, таких як ансамблі, нейронні мережі, методи матричної факторизації та інші;
- розмаїття підходів: Різні учасники конкурсу використовували різні підходи, такі як колаборативна фільтрація, контентна фільтрація, гібридні методи, тематична фільтрація та інші;
- латентні методи та матрична факторизація: Виявлення латентних (прихованих) залежностей у даних, таке як в методах матричної факторизації, стало ключовим для поліпшення точності рекомендацій;
- оптимізація та експерименти: Учасники проводили численні оптимізації та експерименти, спрямовані на поліпшення різних аспектів системи.

Отримані результати демонструють, що немає універсального алгоритму, який підходить до всіх сценаріїв. Важливо враховувати специфіку задачі та характеристики даних для вибору оптимального підходу чи їх комбінації.

Кореляція Пірсона

Кореляція Пірсона використовується для вимірювання лінійної залежності між двома змінними. У контексті рекомендаційних систем, цей алгоритм може вимірювати схожість між користувачами на основі їх атрибутів (наприклад, параметрів статей в блогах). Проте, для обчислення цієї схожості, спочатку множину користувачів слід розбити на групи, які схожі за високорівневими показниками схожості.

Алгоритми кластеризації є інструментами "навчання без вчителя", які виявляють структуру в немаркованих даних. В контексті рекомендаційних систем, вони можуть використовуватися для групування користувачів зі схожими інтересами.

- k-середніх (k-means): Це один з найпростіших алгоритмів кластеризації. Він розділяє елементи на k кластерів, обчислюючи центр мас для кожного кластера і призначаючи члени кластера на підставі їхньої відстані до центру. Цей процес повторюється до стабілізації кластерів;

- є багато інших методів кластеризації, таких як теорія адаптивного резонансу, нечітка кластеризація методом С-середніх, імовірнісна кластеризація за допомогою ЕМ-алгоритму. Алгоритми кластеризації можуть допомагати в групуванні користувачів зі схожими інтересами, що може бути корисним для рекомендаційних систем.

Алгоритми кластеризації

Алгоритми кластеризації - це різновид "Навчання без вчителя" (unsupervised learning), що дозволяє виявити структуру в рядах на перший погляд випадкових (або немаркованих) даних. У загальному випадку такий алгоритм базується на виявленні подібностей між елементами (наприклад, між читачами блогу) за допомогою обчислення їх відстані від інших елементів в просторі ознак (feature space) (ознакою в просторі ознак може, наприклад, бути кількість прочитаних статей в групі блогів). Кількість незалежних ознак визначає розмірність простору ознак. Якщо елементи "близькі" один до одного, то їх можна об'єднати в один кластер.Існує безліч алгоритмів кластеризації. Найпростішим з них є алгоритм k-середніх (k-means), який розділяє елементи на k кластерів. Спочатку елементи розподіляються за цими кластерам в довільному порядку. Потім для кожного кластера обчислюється центр мас (або просто центр) як функція від його членів. Після цього перевіряється відстань кожного члена кластера від центру цього кластера.

Якщо за результатами цієї перевірки член виявляється ближче до іншого кластеру, то він переміщається в цей кластер. Після перевірки всіх відстаней для всіх членів центри кластерів обчислюються наново. При досягненні стабільного стану (в процесі чергової ітерації члени не переміщувалися) набір вважається кластеризованим належним чином, і алгоритм зупиняє роботу. Відстань між двома об'єктами може бути важкою для візуалізації. Один з поширених методів вирішення цього завдання полягає в тому, щоб розглядати кожен член кластера як багатовимірний вектор і обчислювати для нього евклідову відстань. Існує безліч інших різновидів кластеризації, в тому числі теорія адаптивного резонансу

(Adaptive Resonance Theory), нечітка кластеризація методом С-середніх (Fuzzy C-means), імовірнісна кластеризація за допомогою ЕМ-алгоритму (Expectation-Maximization).

Байєсові мережі довіри (Bayesian Belief Nets)

Байесові мережі використовуються для моделювання ймовірнісних і причинно-наслідкових відносин між змінними. У рекомендаційних системах вони можуть бути використані для прогнозування рекомендацій на основі історії покупок або інших взаємодій користувачів.

Переваги застосування байесових мереж в рекомендаційних системах:

- байесові мережі можуть взаємодіяти з емпіричними частотами, суб'єктивними оцінками та теоретичними ймовірностями, що дозволяє враховувати різноманітність даних у рекомендаційних системах;
- байесові мережі можуть адаптуватися до змін в даних та структурі системи рекомендацій;
- байесові мережі дозволяють робити прогнози на основі актуальних даних, що робить їх ефективними для сучасних інтернет-магазинів.

Використання байесових мереж у рекомендаційних системах може сприяти уточненню рекомендацій та забезпечити більш персоналізований підхід до кожного користувача.

Ланцюги Маркова (Markov chains)

Ланцюг Маркова є математичною моделлю, що визначає послідовність подій, де ймовірність кожної події залежить від попередніх подій. У контексті рекомендаційних систем, ланцюги Маркова можуть бути використані для прогнозування рекомендацій, враховуючи попередні дії або взаємодії користувачів з системою.

Станові події. Кожен можливий стан системи визначається певними характеристиками або параметрами, які можуть змінюватися внаслідок взаємодії з користувачами.

Ймовірності переходу. Для кожного стану визначається ймовірність переходу в інший стан внаслідок певних подій. Ці ймовірності можуть бути апроксимовані на основі історії взаємодій користувачів.

Прогнозування рекомендацій. Система може прогнозувати рекомендації для користувачів, виходячи з їхніх поточних станів і імовірностей переходу. Це дозволяє враховувати контекст в часі та динаміку взаємодій.

Оптимізація якості рекомендацій. Можливість прогнозування дозволяє системі проводити послідовну оптимізацію якості рекомендацій, адаптуючись до змін у поведінці користувачів та оточуючому середовищі.

Переваги використання ланцюгів Маркова:

- контекстуальні рекомендації: Ланцюги Маркова можуть допомагати враховувати контекст в часі, що особливо важливо для рекомендацій в змінних умовах;

- динамічні рекомендації: Здатність моделювати зміни стану дозволяє системі адаптуватися до нових умов та взаємодій;

- послідовна оптимізація: Ланцюги Маркова дозволяють системі вдосконалювати рекомендації відповідно до попередніх взаємодій, підвищуючи загальну якість рекомендаційного алгоритму.

Використання ланцюгів Маркова у рекомендаційних системах може бути ефективним, особливо в сценаріях, де важливий контекст та змінність у взаємодії з користувачами.

Класифікація за методом Роккіо (Rocchio classification)

Метод Роккіо використовується для класифікації документів у векторному просторі, і його застосування в рекомендаційних системах може підвищити точність рекомендацій за рахунок аналізу схожості документів.

Основні кроки методу. Векторна модель. Представлення документів і користувачів у векторному просторі, де кожен термін є виміром.

TF-IDF. Обчислення значень TF-IDF для всіх термінів у документах корпусу. Це стандартний підхід до оцінки важливості термінів у контексті документів.

Вектор предмета (Document Vector). Представлення кожного документа у вигляді вектора векторного простору, де кожен елемент вектора відповідає значенню TF-IDF для певного терміна.

Вектор користувача (User Vector). Формування вектора користувача на основі його взаємодій з системою. Наприклад, якщо користувач оцінив чи переглянув певні документи, його вектор може бути сформований як середнє значення векторів цих документів.

Класифікація. Застосування алгоритму Роккіо для класифікації нових документів. Алгоритм оцінює схожість вектора користувача і вектора документа, присвоюючи документу певний клас або рейтинг релевантності для користувача.

Переваги методу Роккіо в рекомендаційних системах. Контентно-орієнтований підхід: Метод Роккіо може бути ефективним у контентно-орієнтованих рекомендаційних системах, де важливий вміст документів.

Схожість векторів. Враховує схожість між векторами користувачів і документів, що дозволяє точно визначити релевантність.

Простота реалізації. Метод Роккіо є відносно простим у реалізації та може бути ефективним для невеликих наборів даних.

Недоліки методу Роккіо. Відсутність особистої моделі користувача: Метод Роккіо може бути менш ефективним, якщо взаємодії користувача з системою не надають достатньої інформації для побудови точного вектора користувача.

Оцінювання в рекомендаційних системах для збору вхідних даних. Проблеми з великими обсягами даних. Зі збільшенням обсягів даних можуть виникнути проблеми з ефективністю і обчислювальною складністю. Обробка великої кількості даних може вимагати значних обчислювальних ресурсів і

призвести до спаду продуктивності. Може бути необхідно застосовувати оптимізаційні стратегії, такі як паралельні обчислення або використання розподілених систем.

Вибір методу оцінок товару. Важливим етапом створення рекомендаційної системи є збір вхідних даних, і для цього використовуються явні та неявні методи. Явні методи, такі як системи оцінювання товару чи послуги, є поширеними.

Розглянемо кілька типів систем оцінювання:

- 5-бальна система. Оцінка від 1 до 5, де 5 - найвища оцінка, а 1 - найнижча. Цей підхід дозволяє користувачам виражати власні враження про товари чи послуги відповідно до п'ятибальної шкали;
- 10-бальна система. Такий метод надає більше деталей для оцінки, дозволяючи користувачам вибирати оцінки від 1 до 10.

Бінарна система (сподобалось/не сподобалось). Користувачі можуть виражати своє ставлення до товару чи послуги, вказуючи, чи їм сподобався.

Вибір найкращого методу.

Бінарне оцінювання:

- Вказується, що бінарний підхід може бути менш суб'єктивним та зручнішим для користувачів. Це може допомогти в уникненні невизначеності та швидше визначати ставлення користувача до товару.

Проблеми бінарного підходу. Зазначається, що бінарний підхід залишає деяку невизначеність при оцінюванні, оскільки користувачі можуть не ставити оцінку, якщо товар не сподобався. Це може призвести до "все або нічого" ситуації.

Інші підходи. Зазначається, що деякі соціальні мережі вирішують проблему відмови від голосування, де користувачі просто вказують своє ставлення, чи сподобався їм товар чи ні.

Не зважаючи на велику кількість методів оцінювання товарів у проекті, автор вважає, що бінарний метод є найкращим вибором. Його переваги

полягають в меншій суб'єктивності, зручності для користувачів та вирішенні проблеми "холодного старту". Алгоритм може більш ефективно обробляти бінарні дані, що сприяє покращенню швидкодії та точності рекомендаційної системи.

Оцінювання в рекомендаційних системах для збору вхідних даних РС

Важливим етапом створення рекомендаційної системи є збір вхідних даних.

Для збору рейтингів використовують явні та неявні способи: вимірювання кількісті часу, яку користувач проводить на конкретній сторінці, чи оцінювання типу товарів що користувач обирає найчастіше.

Найбільш розповсюджений та перевірений спосіб – явний спосіб, через систему оцінювання товару чи послуги.

Для оцінювання товарів використовують різні типи систем оцінювання. Найпоширеніші з них:

- 5-бальна;
- 10-бальна;
- бінарна (сподобалось/не сподобалось).

Вибір методу оцінок товару впливає на подальший розвиток проекту та прибуток від нього. Також він суттєво впливає на роботу алгоритму рекомендаційної системи.

Отже, необхідно проаналізувати усі типи систем оцінювання та обрати найкращий. Рекомендаційні системи та алгоритми у свій час започаткувала та активно розвивала компанія Netflix. На своему сайті вони надали інформацію, як саме користуватися такою системою:

-зовсім не сподобався;
- не сподобався;
- сподобався;
- дійсно сподобався;
- в захваті.

Такий підхід перейняли багато інших відомих компаній, які надають послуги: Amazon, eBay та ін. Але він дуже суб'єктивний. Що може означати "дійсно сподобався" фільм? Чому інтервали між різними операціями нерівні (немає опції "дійсно не сподобався")? Тож, можна зазначити що навіть текст, який несе на своїй меті допомогти виставити рейтинг товару, містить в собі суб'єктивність.

Незважаючи на росповсюдженість п'ятибальної шкали користувачі несвідомо зводять свій вибір до оцінювання бінарним підходом. В 2009 році Youtube (відео-платформа від Google) поширила статистику оцінок користувачів, серед яких найпопулярнішими оцінками стали "5" (~70%) та "1" (~25%).

Бінарне оцінювання використане на практиці показало, що воно є менш суб'єктивним, менш багатозначним, адже користувачам значно зручніше та швидше визначити: чи сподобався їм товар, чи ні. Бінарний підхід також містить декілька недоліків. Він залишає деяку невизначеність при оцінюванні. Якщо користувачу не сподобався товар чи послуга, то існує велика вірогідність, що він зовсім не буде ставити оцінку.

Така система - це система типу "все або нічого". Але користувачі, які все таки ставлять оцінки дуже серйозно відносяться до своїх рекомендацій та дуже зацікавлені в тому, щоб система пропонувала їм релевантні результати. Деякі проекти (в основному - соціальні мережі) долають проблему відмови від голосування (такі як VK, Facebook).

Якщо опустити деякі деталі, то якийсь товар буде оцінено в +1, якщо ви проголосували позитивно, та -1, якщо ви не звернули увагу на нього. Це є різновид бінарного голосування, який теж гарно працює в сучасних умовах (в основному його використовують СМІ).

Отож, незважаючи на велику кількість способів оцінити товар чи послугу в проекті, краще це робити за допомогою бінарного методу, оскільки він не містить недоліків та суттєво знижує суб'єктивність оцінки. Також завдяки бінарному підходу відпадає проблема "холодного старту" (рекомендації новим

користувачам) та значно покращується робота алгоритму (менше даних для обробки - більша швидкість).

2.2 Обґрунтування вибору методів розробки

Аналіз рекомендаційних технологій відомих сервсів

Програмна платформа eBay та її вплив на рекомендаційну систему eBay має досить розвинену програмну платформу, яка включає різні програмні інтерфейси (API) та сервіси для поліпшення взаємодії користувачів з каталогом товарів. Це сприяє покращенню рекомендаційної системи та зростанню релевантності пропозицій товарів чи послуг. Основні особливості програмної платформи eBay, які впливають на рекомендації, включають:

- семантичний пошук. eBay використовує семантичний пошук, що дозволяє краще враховувати семантику пошукових запитів користувачів. Кожен рівень таксономічного графу пов'язаний з ключовими словами, що сприяє точному відображенню термінів в пошукових запитах.
- відгуки користувачів. Використання Feedback API дозволяє eBay формувати рейтинг продавців на основі відгуків покупців. Товари від продавців з високим рейтингом можуть мати більший пріоритет у рекомендаціях.
- групування товарів. Використання Related Items Management API дозволяє формувати групи взаємопов'язаних товарів, які можуть бути пропоновані як пакети. Це дозволяє продавцям явно вказувати, які інші товари можуть бути корисні покупцям.
- взаємозв'язки комплектуючих та аксесуарів. Використання Product Services API дозволяє створювати зв'язки, які описують взаємозв'язки між комплектуючими та аксесуарами з продуктами. Наприклад, покупець шукаючи ноутбук може одразу отримати список сумок, які підходять до його моделі.

Використання цих функціональностей допомагає не лише поліпшити роботу рекомендаційної системи, а й надає користувачам більше контексту та

допомагає забезпечити більшість релевантних рекомендацій, що підвищує загальне задоволення користувачів платформи.

Метрики "ефективності" просування товарів в каталозі eBay

Окремої уваги в контексті задачі збільшення релевантності пропозицій товарів та послуг заслуговує сервіс eBay Listing Analytics, який дозволяє виконати аналіз "ефективності" пропозицій конкретних продавців. Для оцінки "ефективності" (в даному контексті, успішності поточної групи товарів або послуг для цільової аудиторії) використовуються наступні ключові метрики:

- **Rank.** Дана метрика визначає положення конкретних товарів та послуг в загальному рейтингу покупців. Так, наприклад, товар з рейтингом "5" буде пропонуватися покупцю раніше, ніж товар з рейтингом "15".

При роботі з сервісом eBay Listing Analytics пошук товарів виконується із використанням заданих ключових слів з метою оцінки їх рейтингу. Таким чином, конкретне місце товару чи послуги в списку пропозицій залежить від рангу та ключових слів, які використовуються при формуванні пошукового запиту;

- **Format.** Ця метрика описує формат, якому відповідають товар чи послуга, що рекомендуються користувачу (auction-style listing, fixed price listing);

- Impressions. Метрика "Враження" характеризує кількість разів, що поточна група товарів та послуг рекомендувалася потенційним покупцям в процесі пошуку;

- **Clicks.** Ця метрика оцінює кількість запитів користувачів до детальних характеристик товарів чи послуг після перегляду списку рекомендацій;

- Click through. Дана метрика – це відношення метрики Clicks до метрики Impressions (Clicks count / Impressions count). Чим більше значення цієї метрики, тим краще, бо це означає, що покупці частіше обирають (тобто заходять на сторінку детального опису) рекомендовані товари та послуги, на відміну від аналогічних;

- Sold items. Ця метрика представляє собою кількість разів, коли покупці придбали рекомендований товар чи послугу;

- **Sell through.** Ця метрика представляє собою відношення кількості проданих товаров чи послуг до кількості їх переглядів при пошуку. Чим більше значення цієї метрики, тим краще, адже це означає, що покупці частіше після перегляду описів цих пропозицій приймають рішення про купівлю;

- **Watchers**. Загальна кількість переглядів конкретних товарів чи послуг;

- **Sales**. Загальна кількість проданих товарів чи послуг в грошовому еквіваленті (тобто загальна сума покупок кожного товару зі списку). Використання аналітичних засобів, таких як eBay Listing Analytics, дозволяє проаналізувати успішність різних груп товарів та послуг у покупців, визначити найефективнішу стратегію маркетингу товарів та скорегувати асортимент, а також збільшити якість обслуговування покупців за рахунок пропозицій тих товарів та послуг, які користувач очікує побачити. Але варто нагадати, що метрики зазначені вище якраз і забезпечують зріз з величезних об'ємів даних, що збирає каталог.

Рекомендаційна система інтернет-магазину Amazon

Інший досить крупний інтернет-магазин Amazon (amazon.com) розробив власне високопродуктивне сховище пар "ключ-значення" (Highly Available Key-Value Store) Dynamo, яке використовується рекомендаційною системою Amazon. Dynamo використовує синтез добре відомих технік для досягнення масштабування та високої доступності: дані кластеризуються (partioning) та реплікуються, використовуючи узгоджене хешування (consistent hashing), а корректність даних забезпечується за допомогою версій об'єктів.

Для 99% запитів СУБД забезпечує час відгуку на запит не більше, ніж 300 мс. Для витягання інформації зі сховища достатньо знати значення ключа. В період 40 пікових навантажень система забезпечує обробку декількох мільйонів запитів на день. Рекомендаційна система інтернет-магазину Amazon реалізує різні підходи до формування рекомендацій, основною метою яких є максимальний облік інтересів користувачів за допомогою їх залучення до процесу "оцінювання товарів", а також неявного аналізу їх поведінки на сайті:

Customers who Bought

Даний підхід до формуванню списку рекомендацій використовує інформацію про популярність товарів у покупців зі схожими інтересами. Так при виборі конкретної книги, користувачу буде запропонований список книг, які користуються попитом у людей, що вже придбали цю книгу, а також список авторів книг, чиї роботи купують покупці книг, автором яких є саме автор обраної книги. Для реалізації даного механізму сервіс використовує спеціальний алгоритм “Item to item Correlation”, що був запатентований компанією Amazon. Основна ідея алгоритму – це зіставлення товарів, що придбав користувач, з аналогічними товарами (з використанням ключових параметрів, характеристик) та формування рекомендацій з урахуванням рейтингу цих товарів.

- Eyes. Цей сервіс дозволяє користувачам отримувати листи на електрону пошту про додавання нових товарів до каталогу Amazon. Користувачі можуть повністю контролювати параметри товарів, на базі яких буде готуватися нова вибірка для сповіщення. Запити можна формувати неявно, за допомогою використання вже існуючої вибірки в якості шаблона для пошуку. • Amazon.com Delivers. Даний серів дуже близький за функціоналом до сервісу Eyes. Користувач має можливість задати категорії каталогу (наприклад, кулінарні книги або фантастика) та оформити підписку на отримання сповіщень про рекомендовані товари з обраних категорій.

- **Book Matcher.** Цей сервіс дає можливість користувачам залишати відгуки безпосередньо про куплені чи прочитані книги. Для прочитаних книг покупець формує рейтинг за п’ятибальною шкалою (від “hated it” до “loved it”). На базі інтересів користувачів сервіс формує рекомендації для нього. При цьому рекомендовані книги в свою чергу можуть бути оцінені (за допомогою надання рейтингу) покупцем (функція “rate these books”), що дозволить наступного разу більш точно передбачити його побажань при формуванні рекомендацій.

- **Customer Comments.** Даний сервіс дає можливість отримувати рекомендації, базуючись на думках інших користувачів. Так, наприклад, для

кожної книги в каталозі закріплено читацький рейтинг у вигляді списку від однієї до п’яти зірочок, який супроводжується текстовими коментарями покупців. Це дає можливість більш точно отримувати досвід про книгу перед її купівлею. Коли користувач обирає для купівлі який-небудь товар, Amazon на базі цього вихідного товару рекомендує користувачу інші товари, які було куплено іншими користувачами (за допомогою матриці купівлі наступного товара на базі його схожості з попередньою покупкою).

Пошуковий сервіс компанії A9

Паралельно з Dynamo, Amazon використовує сервіс пошуку товарів A9, який також заслуговує окремої уваги. Він запустився в 2003 році та мав на меті допомагати людям шукати речі, які вони справді жадають знайти. Він складається з декількох складових, які, працюючи разом, допомагають користувачам знайти товари.

Порівняльна характеристика сервісів Amazon та eBay

Нижче наведено перелік критично важливих рекомендаційних сервісів двох інтернет-гігантів та їх короткі характеристики.

Було детально проаналізовано базові підходи до створення рекомендаційних систем.

Таблиця 2.3 – Зведена таблиця рекомендаційних сервісів

Інтернет-сервіс	Вхідні дані	Рекомендаційна технологія	Вихідні рекомендації
Amazon			
Customers who Bought	Схожий товар (база покупок)	Контентна фільтрація	Блок рекомендованих товарів
Eyes	Електронна пошта	Контентна фільтрація	Лист з рекомендаціями
Amazon.com Delivers	Електронна пошта	Контентна фільтраця	Лист з рекомендаціями, де наведені форми для корегування інтересів
Book Matcher	Найкращі товари	Колаборативна фільтрація	Список товарів
Customer Comments	Середній рейтинг товару. Коментарі користувачів	Гібридна фільтрація на базі рейтингів та ключових слів в коментарях	Кореляція рекомендацій інших сервісів Amazon
Product Search	Пошуковий запит (текст / картинка)	Гібридна фільтрація	Список товарів

Продовження таблиці 2.3 – Зведена таблиця рекомендаційних сервісів

Інтернет-сервіс	Вхідні дані	Рекомендаційна технологія	Вихідні рекомендації
eBay			
Feedback API	Середній рейтинг товару. Коментарі користувачів	Гібридна фільтрація на базі рейтингів та ключових слів в коментарях	Кореляція рекомендацій інших сервісів eBay
Related Items Management API	Зібрані дані про інтереси споживача	Контентна фільтрація	Блок рекомендацій з комплектами товарів
Product Services	Дані інших користувачів про інтерес до аксесуарів для базового товару	Колаборативна фільтрація	Блок рекомендацій з аксесуарами
Product Search	Пошуковий запит	Гібридна фільтрація	Список рекомендацій
eBay Listing Analytics	Дані про поведінку всіх користувачів	Гібридна фільтрація	Кореляція рекомендацій інших сервісів

В результаті дослідження було зроблено висновок, що не існує універсального алгоритму, який би підійшов до будь-якого інтернет сервісу. Кожен тип алгоритмів слід використовувати лише за призначенням.

Алгоритми колаборативної фільтрації прогнозують рекомендації на основі схожості користувачів. В загальному випадку, алгоритми цієї групи розбивають користувачів на кластери, базуючись на їх оцінках, та прогнозують нові товари в рамках конкретної групи (кластеру). Реалізація таких алгоритмів порівняно проста, але на великому обсязі даних їх швидкодія різко знижується. Такі алгоритми добре підходять для блогів з помірними навантаженнями,

соціальних мереж з невеликим обсягом інформації, що прогнозується тощо. Підходи контентної фільтрації підходять до вирішення проблеми з іншого боку. Вони надають рекомендації на базі інтересів користувача та атрибутів продуктів.

Реалізація таких алгоритмів є доволі складною, оскільки необхідно будувати складні математичні моделі, а їх важко створювати та підтримувати. Але швидкодія таких алгоритмів знаходиться на високому рівні.

Цей блок алгоритмів широко застосовується в інтернет-магазинах. Гібридні алгоритми показують найкращі результати, оскільки вони мають на меті вирішити основні проблеми колаборативної та контентної фільтрації. Вони базуються на змішуванні результатів в різному вигляді алгоритмів цих двох груп. Але проблема в тому, що такі системи створювати та підтримувати в декілька разів складніше, ніж системи з контентною фільтрацією, так як вони мають обраховувати велику кількість даних одночасно. Тому їх використання буде виправданим лише у випадку використання у складі крупного та високонавантаженого інтернет-сервісу з мільйонами користувачів та десятками мільйонів одиниць всілякого контенту. Зазвичай використовуються у великих інтернет-магазинах, соціальних мережах тощо.

Для покращення точності результатів колаборативної та контентної фільтрації необов'язково використовувати гібридні системи. Існують деякі алгоритми, які покращують результати цих алгоритмів. Наприклад, SVD алгоритм значно збільшує швидкодію алгоритмів колаборативної фільтрації, 36 оскільки знижує кількість обчислень, а алгоритм TF-IDF дозволяє створювати більш тонкі моделі для контентної фільтрації. Також під час дослідження було виявлено, що майже будь-який алгоритм має отримувати на вхід оцінки користувачів, щоб вимірювати їх зацікавленість в даному товарі. Тому необхідно ретельно вибирати систему оцінювання.

Найкращою виявилася бінарна система оцінювання.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на магістерську роботу, реалізації підлягає програмне забезпечення, яке призначено для реалізації рекомендаційної системи.

В процесі розробки магістерської роботи необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Виходячи з теми магістерської роботи потрібно розробити програмне забезпечення рекомендаційної системи.

Розглянемо основні компоненти системи, потоки даних між ними, а також її входи та виходи.

Вибір мови програмування

Особливості фреймворку Ruby on Rails

Ruby - це динамічна мова програмування, спрямована на забезпечення простоти та продуктивності в розробці коду. Вона відзначається зручним синтаксисом, що сприяє зрозумілості та легкості написання коду. Фреймворк Ruby on Rails, написаний на Ruby, є програмним забезпеченням, спрямованим на полегшення розробки та інтеграції різних компонентів проекту. Мова визначає типи під час виконання, що надає гнучкість у роботі з даними. Читабельний та лаконічний код сприяє ефективній розробці. Відмінна для початківців та швидкі розробники.

Ruby on Rails

- фреймворк, забезпечує готові рішення для розробки веб-проектів;

- конвенція перед конфігурацією, заснований на стандартах та конвенціях, що полегшує роботу над проектами;

- швидкість розробки, прискорює процес розробки завдяки використанню готових компонентів та вбудованих інструментів;

Різниця між фреймворком та CMS:

- фреймворк Ruby on Rails, вимагає висококваліфікованих розробників для проектування та розробки, дозволяє створювати проекти з високим рівнем специфікації та функціональністю.

CMS, зазвичай призначена для користувачів без технічного досвіду, легко

розгортати та налаштовувати, але має обмежений функціонал порівняно зі складнішими проектами.

Фреймворк Ruby on Rails ідеально підходить для веб-студій та агентств, які працюють над складними та індивідуальними проектами, оскільки надає швидкість розробки та гнучкість у роботі зі змінами в проекті.

Переваги платформи Ruby on Rails

Переваги щодо швидкості розробки за допомогою Ruby та фреймворку Ruby on Rails (RoR) підкріплено рядом переваг та позитивних характеристик:

- швидкість розробки, наявність готових інструментів, бібліотек та зручний синтаксис мови Ruby допомагають розробникам прискорити процес розробки проектів. Навіть зазначено, що швидкість розробки може бути вищою на 30-40% порівняно з іншими мовами та фреймворками;

- готові рішення та бібліотеки, використання готових рішень "з коробки" та можливість використання бібліотек інших розробників полегшують завдання розробки;

- автоматизоване тестування, вбудовані засоби для автоматизованого тестування дозволяють швидше переходити від етапу написання коду до етапу, коли програма працює без помилок;

- безпека: RoR забезпечує вбудовані засоби безпеки, такі як захист від SQL-ін’єкцій та XSS-атак. Це може сприяти запобіганню помилок безпеки в розробці;

- масштабованість, наведені проекти, такі як Kickstarter, Groupon, та Basecamp, свідчать про успішне використання RoR у великих та успішних інтернет-проектах.

Зазначено, що можливі проблеми продуктивності виникають не через помилковий вибір платформи чи мови програмування, а, скоріше, через помилки у проектуванні архітектури, управлінні кешуванням даних або неоптимальному виборі СУБД. Це підкреслює важливість правильного планування та розробки для успішного використання Ruby та Ruby on Rails у проектах будь-якого масштабу.

Обмеження фреймворку

Розробників на Ruby on Rails менше, ніж розробників на PHP та його фреймворках, оскільки тут вищий поріг входження та, зазвичай, програміст приходить до Ruby вже після декількох років PHP. Але при цьому слід пам'ятати, що досвідчених розробників дуже мало в цих двох сферах. Важливим обмеженням RoR вважається вбудована ORM ActiveRecord та модуль Active Support, які містять велику кількість допоміжних методів, які майже не використовуються в проектах, але ці модулі можна замінити чимось менш ресурсозатратним (Sequel).

Побудова memory-based алгоритму колаборативної фільтрації з бінорною системою оцінювання Як було описано вище, існує величезна кількість алгоритмів для різних типів рекомендацій. Найшвидшими в реалізації та імплементації виявилися рекомендаційні алгоритми, що використовують колаборативну фільтрацію. В цьому розділі буде приділено увагу memory-based рішенню для колаборативної фільтрації. Memory-based підхід досить розповсюджений під час реалізації рекомендаційної системи, що використовує колаборативну фільтрацію, на будь-якій серверній мові програмування, оскільки в якості "пам'яті" рекомендаційної системи він використовую ту саму ж базу даних, що і основний додаток.

Декілька з цих алгоритмів використовують широковідомі підході як Евклідова відстань, кореляція Пірсона, коефіцієнт схожості векторів Оцукі-Отіаі та алгоритм k-найближчих сусідів. Вони мають повну документацію, багато прикладів реалізації та базуються на 5-бальній шкалі оцінювання товарів. Але, як було показано вище, зрозуміліше та краще спрацьовує підхід, що використовує лише бінарну систему рекомендацій ("сподобалося – не сподобалося").

Прогнозування рекомендацій

Коефіцієнт подібності між користувачами використовується для прогнозування списку рекомендацій для користувачів. Для цього необхідно обрахувати для кожного товару hive-mind сумму (коефіцієнт колективної думки).

Для користувачів із заданої групи, які якимось чином оцінили товар (він їм сподобався або ні), необхідно виконати наступні обчислення: якщо користувачу сподобався товар, то до загальної суми додається коефіцієнт подібності поточного користувача з тим, що оцінював даний товар, в протилежному випадку – від суми віднімається коефіцієнт подібності між користувачами. Після цього отриманий результат ділиться на загальну кількість людей, що якимось чином оцінили даний товар. Для кращого розуміння нижче наведений приклад коду, що обраховує прогнозовану оцінку товару для користувача:

```
class User
def similarity_with(user)
# Array#& is the set intersection operator.
agreements = (self.likes & user.likes).size
agreements += (self.dislikes & user.dislikes).size
disagreements = (self.likes & user.dislikes).size
disagreements += (self.dislikes & user.likes).size
# Array#| is the set union operator
total = (self.likes + self.dislikes) | (user.likes + user.dislikes)
return (agreements - disagreements) / total.size.to_f
end
def prediction_for(item)
hive_mind_sum = 0.0
rated_by = item.liked_by.size + item.disliked_by.size
item.liked_by.each { |u| hive_mind_sum += self.similarity_with(u) }
item.disliked_by.each { |u| hive_mind_sum -= self.similarity_with(u) }
return hive_mind_sum / rated_by.to_f
end
end
```

Таким чином такий спосіб обрахунку прогнозів дозволить користувачам, що ще невстигли оцінити багато товарів, отримати прогнози від користувачів, що оцінили велику кількість товарів

3.2 Розробка структурної схеми

Структурна схема системи – це сукупність об’єктів та частин та взаємозв’язки між ними.

Призначенням структурної схеми є наглядне відображення складових частин розробляємої системи, її основних блоків, вузлів та взаємозв’язок між ними.

Структурна схема рекомендаційної системи зображена на рисунку 3.2 .

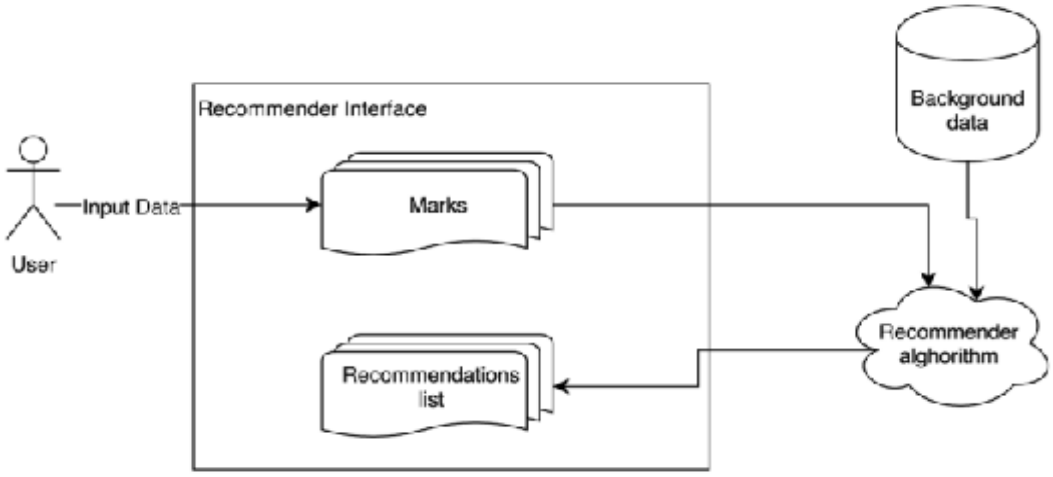

Рисунок 3.2 – Структурна схема РС

Структурна схема рекомендаційної системи має наступні блоки:

Збір даних:

- вхідні дані: Інформація про користувачів, об'єкти (товари, статті, фільми тощо) та їх характеристики;

- збір історії: Історія взаємодії користувачів з системою, така як покупки, перегляди, оцінки.

Обробка даних:

- профілі користувачів: Створення профілю для кожного користувача на основі його дій та вподобань;

- опис об'єктів: Створення опису для об'єктів на основі їхніх характеристик.

Моделювання:

- фільтрація контенту: Застосування алгоритмів фільтрації, таких як колаборативна фільтрація або контент-базована фільтрація, для визначення релевантних об'єктів;

- ранжування: Визначення порядку рекомендацій на основі ймовірності зацікавленості користувача.

Вивід рекомендацій:

- персоналізовані рекомендації: Подання користувачеві списку рекомендацій на основі попередньої обробки та моделювання.

Зворотній зв'язок:

- оцінка рекомендацій: Збір інформації про те, наскільки задоволений користувач запропонованими рекомендаціями.

- оновлення моделі: Використання отриманих відгуків для покращення алгоритмів та збору нових даних.

Background

Background - це контекст, який оточує якусь подію або ситуацію. В контексті рекомендаційних систем, "background" може вказувати на інформацію про користувачів або об'єкти, яка використовується для покращення якості рекомендацій.

Наприклад, у контексті користувачів, background може включати в себе їхні попередні взаємодії з системою, історію покупок, відгуки, або будь-яку іншу інформацію, яка вказує на їхні вподобання і потреби.

В разі об'єктів, background може включати характеристики об'єктів, їхню популярність серед користувачів, категорії, до яких вони відносяться, тощо.

Врахування background допомагає створити більш точні та персоналізовані рекомендації, оскільки система враховує попередні дії та уподобання користувачів або характеристики об'єктів при формуванні рекомендацій.

Marks

"Marks" в даному контексті може вказувати на оцінки або позначення, які користувачі надають об'єктам або ресурсам у рекомендаційній системі. Це може бути числова оцінка, наприклад, рейтинг від 1 до 5, або якась інша форма відгуку, така як "подобається" чи "не подобається".

Оцінки користувачів важливі для рекомендаційних систем, оскільки вони надають інформацію про те, наскільки конкретний об'єкт був корисний чи

цікавий для користувача. На основі цих оцінок система може адаптуватися та надавати більш персоналізовані рекомендації в майбутньому.

Marks також може вказувати на мітки або теги, які призначаються об'єктам для покращення алгоритмів рекомендацій. Наприклад, тег "комедія" може допомагати системі визначати, що користувачам подобаються комедійні фільми.

Input Data

"Input Data" в рекомендаційних системах - це інформація, яка подається в систему для генерації рекомендацій. Ці дані можуть бути різноманітні та включати інформацію про користувачів, об'єкти та їх взаємодію. Ось кілька прикладів:

Інформація про користувачів:

- демографічні дані: Вік, стать, місце проживання;
- споживацькі звички: Інтереси, попередні покупки, перегляди, оцінки;
- соціальний контекст: Зв'язки з іншими користувачами, соціальні мережі.

Інформація про об'єкти:

- характеристики об'єктів: Деталі товарів, опис фільмів, ключові слова для статей;
- категорії та теги: Класифікація об'єктів за категоріями або тегами;
- популярність: Рейтинги, кількість переглядів, коментарі.

Історія взаємодії:

- історія покупок: які товари користувач купував раніше;
- історія переглядів: які об'єкти користувач переглядав;
- оцінки та відгуки: як користувач оцінює та коментує об'єкти.

Ці дані служать основою для алгоритмів рекомендацій, які аналізують і їх використовують для передбачення того, які об'єкти можуть зацікавити конкретного користувача.

3.3 Розробка функціональної схеми

Як бачимо із структурної схеми є декілька пристроїв на різноманітних платформах, які певним чином надсилають дані до серверу.

Функціональна схема розробленої системи зображена на рисунку 3.3.

Ця схема вказує на основні функціональні кроки, які використовуються рекомендаційною системою для забезпечення персоналізованих рекомендацій користувачам.

Схема складається з наступних блоків.

Збір та обробка даних:

Збір інформації: Отримання даних про користувачів (демографічні дані, історія взаємодії) та об'єкти (характеристики, категорії).

Обробка даних: Створення профілів користувачів та описів об'єктів на основі зібраних даних.

Моделювання та прогнозування:

- колаборативна фільтрація: Визначення схожості між користувачами або об'єктами для передбачення інтересів;

- контент-базована фільтрація: врахування характеристик об'єктів та інтересів користувачів для рекомендацій;

- клибинне навчання: використання нейронних мереж для аналізу складних взаємозв'язків у даних.

Ранжування та вивід рекомендацій:

- формування списку рекомендацій: Визначення ранжування об'єктів відповідно до ймовірності зацікавленості користувача;

- вивід рекомендацій: Подання рекомендацій користувачеві через інтерфейс або інші канали зв'язку.

Зворотній зв'язок та оновлення:

- збір зворотнього зв'язку: Отримання відгуків та оцінок від користувачів щодо рекомендацій;

- оновлення моделі: Використання зворотного зв'язку для покращення алгоритмів та оновлення профілів користувачів та об'єктів.

Інтерфейс користувача:

- відображення рекомендацій: Спілкування з користувачем через інтерфейс для представлення рекомендацій та отримання відгуків.

Рисунок 3.3 - Функціональна схема системи

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.4.

Діаграма взаємодії процесів рекомендаційної системи представлена у вигляді блок-схеми або схеми взаємозв'язків між різними складовими системи. Загальний опис можливих етапів і взаємодій у рекомендаційній системі:

серверна частина рекомендаційної системи включає в себе ряд компонентів та процесів, які відповідають за обробку даних, аналіз, моделювання та генерацію рекомендацій.

Нижче наведено загальний опис ключових етапів серверної частини:

Збір та зберігання даних

Система управління базою даних: Спеціалізована база даних для зберігання інформації про користувачів, предмети, їхні взаємодії та інші необхідні дані.

API для збору даних

Відкритий інтерфейс програмування додатків для збору даних із зовнішніх джерел або інших компонентів системи.

Обробка та аналіз даних

Компонент, який відповідає за обробку та аналіз отриманих даних.

Модуль профілювання користувачів та предметів

Аналіз даних для створення та оновлення профілю користувача та характеристик предметів.

Моделювання рекомендацій

Моделі колаборативного та контентного аналізу: Системи для визначення схожості між користувачами або предметами та генерації рекомендацій на основі цих моделей.

Алгоритми машинного навчання

Використання різних алгоритмів для покращення точності та персоналізації рекомендацій.

Управління сесіями та запитаннями

Керування станом сесій: Відстеження взаємодії з користувачем та управління станом сесії для покращення рекомендацій.

Обробка запитань та фільтрація: Аналіз запитань користувачів та фільтрація результатів для оптимізації рекомендацій.

Модуль колаборативної фільтрації є ключовою складовою рекомендаційних систем і використовується для генерації рекомендацій, базуючись на знаннях про взаємодію між користувачами або предметами. Цей метод може бути поділений на дві основні категорії: колаборативний фільтр за спільною взаємодією (user-based collaborative filtering) та колаборативний фільтр

за спільними характеристиками (item-based collaborative filtering).

Модуль відбору користувачів (user selection module) у рекомендаційній системі відповідає за вибір та фільтрацію користувачів, які будуть використовуватися при створенні рекомендацій. Цей модуль може мати різні завдання, такі як визначення групи схожих користувачів, виділення активних користувачів чи врахування контекстуальних факторів при виборі аудиторії для рекомендацій.

Модуль асоціативних правил у рекомендаційній системі використовується для виявлення асоціативних зв'язків між різними предметами чи елементами, що дозволяє рекомендувати користувачеві товари або послуги, що часто співвідносяться з тим, що він чи вона вже вибрав. Цей модуль може застосовувати алгоритми асоціативного аналізу, такі як алгоритм Apriori або FP-Growth, для виявлення зв'язків в масиві даних.

Модуль формування правил в рекомендаційній системі відповідає за визначення та створення правил, які вказують на взаємозв'язки між різними об'єктами чи характеристиками, що можуть бути використані для генерації рекомендацій. Цей модуль може використовувати різні методи та алгоритми для аналізу даних та визначення правил.

Модуль аналізу транзакцій у рекомендаційній системі відповідає за обробку і аналіз історії транзакцій користувачів або клієнтів для виявлення корисних зв'язків, патернів або тенденцій. Цей модуль може використовувати методи асоціативного аналізу, класичного машинного навчання чи інші техніки для витягування інсайтів і формування рекомендацій.

Модуль обчислення оцінок в рекомендаційній системі відповідає за розрахунок прогнозованих оцінок або інших метрик, які вказують на ймовірність того, що користувач оцінить чи взаємодіє з певними предметами. Ці оцінки використовуються для ранжування та визначення найбільш релевантних рекомендацій для користувача.

Основні етапи та функції модуля обчислення оцінок включають:

- вибір алгоритму обчислення;
- визначення методу: Вибір алгоритму чи методу для обчислення оцінок, такого як колаборативний аналіз, контентний аналіз, глибинне навчання чи інші.
- обробка даних
- підготовка фідбеку користувача: Врахування даних про оцінки, які користувачі вже надали певним предметам;
- нормалізація даних: Застосування нормалізації, яка може включати в себе центрування даних чи інші методи для вирівнювання значень;
- розрахунок оцінок.

Використання алгоритму

Застосування обраного алгоритму для розрахунку прогнозованих оцінок для предметів, з якими користувач не мав взаємодії.

- визначення рекомендацій;
- ранжування рекомендацій: Визначення та ранжування предметів згідно з обчисленими оцінками;
- вивід результатів;

Подання рекомендацій: Відправлення результатів розрахунків модулю, що відповідає за вивід рекомендацій користувачеві чи іншому модулю для подальшої обробки.

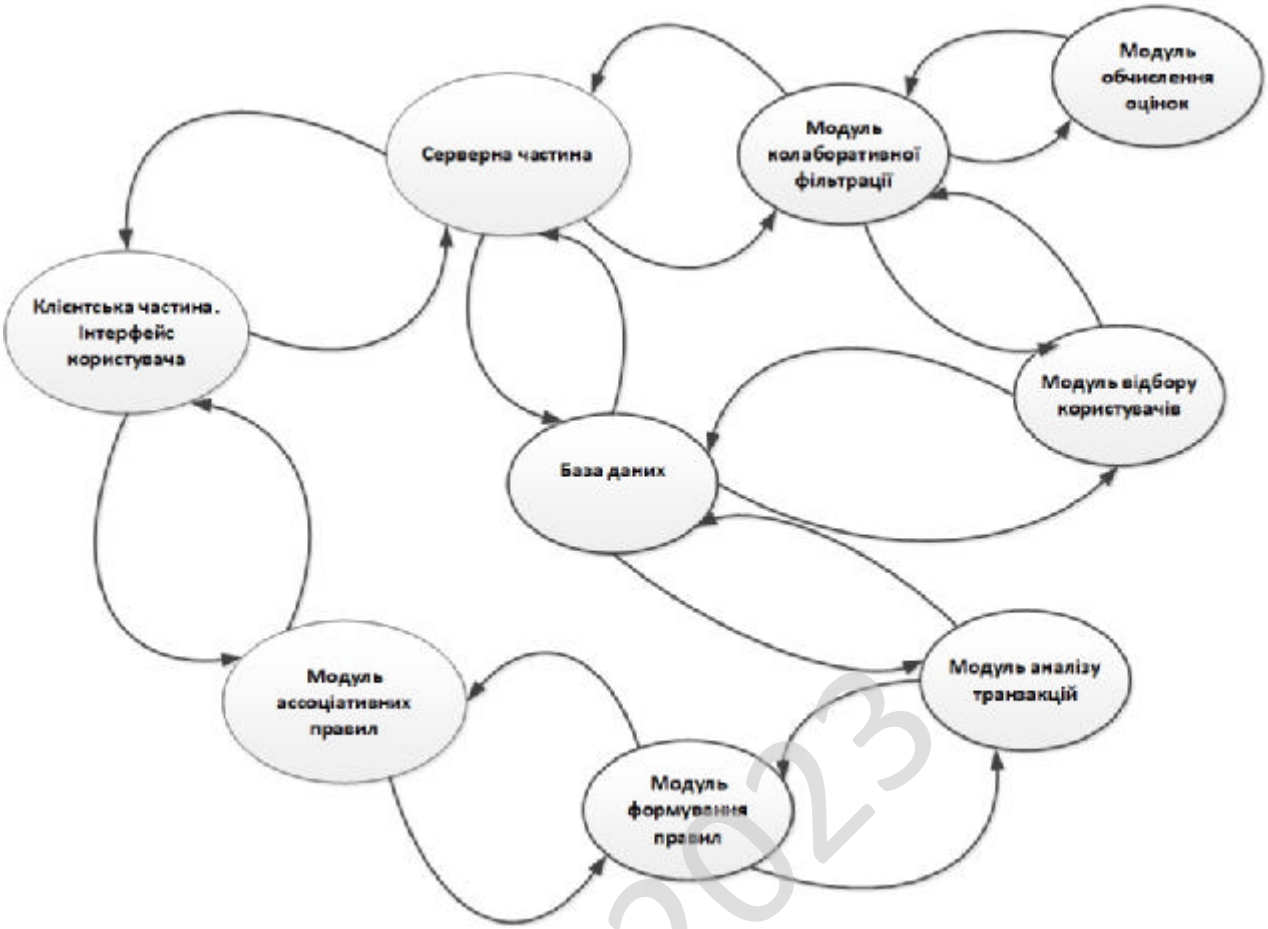

Рисунок 3.3 – Діаграма процесів системи

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

Особливості фреймворку Ruby on Rails

Ruby – динамічна мова програмування, яка має на меті збільшити простоту та продуктивність вихідного коду. Він має зручний синтаксис, який приємно читати та легко писати.

Ruby on Rails – фреймворк, написаний на мові програмування Ruby, тобто програмне забезпечення, що полегшує розробку та об'єднання декількох окремих складових проекту (наприклад, аутентифікація та авторизація користувачів або каталог статей в блозі).

Фреймворк, на відміну від CMS (система керування вмістом), яку може розгорнути та налаштувати навіть не-програміст, потребує проектування та розробки кваліфікованими спеціалістами. Але на ньому зручніше та швидше створювати проекти, які зовсім відрізняються функціоналом від типового сайту. А до веб-студій та агентств нечасто приходять за повністю типовими сайтами, оскільки замовники часто змінюють поведінку "на льоту".

Переваги платформи Ruby on Rails

Основною перевагою мови програмування Ruby та фреймворку Ruby on Rails є швидкість розробки. На практиці швідкість розробки проектів на RoR вище на 30-40% по відношенню до інших мов програмування або фреймворків. Такий приріст швидкості розробки пояснюється широким набором готових до роботи «з коробки» інструментів RoR, можливістю використовувати готові бібліотеки інших розробників та, звичайно, зручністю програмування на Ruby.

Крім цього, на відміну від інших фреймворків, до складу RoR входять ефективні засоби для автоматизованого тестування, що прискорює перехід проекту від стадії "програму написано" до стадії "програма працює без

помилок”. Цей перехід майже завжди займає найбільшу кількість часу під час реалізації майже будь-яких проектів.

Також варто відмітити, що Ruby on Rails забезпечує кращу безпеку для додатків. Під час використання інструментів RoR виключені SQL-ін’єкції та XSS-атаки, всі вхідні параметри екрануються за замовчанням, вихідні змінні в шаблонах також екрануються. У розробника майже немає шансів допустити помилку безпеки.

Деякі розробники, що недостатньо добре знайомі з цією технологією, чомусь вважають, що інтернет-проекти на RoR погано масштабуються. Як приклад майже всі розробники наводять Twitter, який в свій час відмовився від Rails через якісь внутрішні причини. Але треба звернути увагу на більш відомі проекти, як Kickstarter, Groupon або Basecamp – всі ці проекти написані з використанням Rails без проблем з масштабуванням. В будь-якому випадку, проблеми продуктивності будь-якого проекту, - це не проблеми помилкового вибору платформи чи мови програмування. Найчастіше ці проблеми були викликані помилками під час проектування архітектури проекту, кешуванням даних або неоптимальним вибором СУБД.

Обмеження фреймворку

Розробників на Ruby on Rails менше, ніж розробників на PHP та його фреймворках, оскільки тут вищий поріг входження та, зазвичай, програміст приходить до Ruby вже після декількох років PHP. Але при цьому слід пам’ятати, що досвідчених розробників дуже мало в цих двох сферах.

Важливим обмеженням RoR вважається вбудована ORM ActiveRecord та модуль Active Support, які містять велику кількість допоміжних методів, які майже не використовуються в проектах, але ці модулі можна замінити чимось менш ресурсозатратним (Sequel).

Типові проекти на Ruby on Rails

Фреймворк RoR найкраще підходить для наступних типів проектів:

- інтернет-магазини з складною системою фільтрації вмісту, модулями підбору та інтеграціями із зовнішніми сервісами;

- купонні сервіси, веб-сервіси для колективних закупівель;

- інформаційні портали, електронні видання;

- біржи та торгівельні майданчики;

- сайти повідомлень та знайомств;

- соціальні мережі;

- незвичайні/нестандартні, технічно важкі проекти;

- сервіси та SaaS-проекти.

Побудова memory-based алгоритму колаборативної фільтрації з бінорною системою оцінювання

Як було описано вище, існує величезна кількість алгоритмів для різних типів рекомендацій. Найшвидшими в реалізації та імплементації виявилися рекомендаційні алгоритми, що використовують колаборативну фільтрацію. В цьому розділі буде приділено увагу memory-based рішенню для колаборативної фільтрації.

Memory-based підхід досить розповсюджений під час реалізації рекомендаційної системи, що використовує колаборативну фільтрацію, на будь-якій серверній мові програмування, оскільки в якості “пам’яті” рекомендаційної системи він використовую ту саму ж базу даних, що і основний додаток.

Декілька з цих алгоритмів використовують широковідомі підході як Евклідова відстань, кореляція Пірсона, коефіцієнт схожості векторів Оцукі- Отіаі та алгоритм k-найближчих сусідів. Вони мають повну документацію, багато прикладів реалізації та базуються на 5-бальній шкалі оцінювання товарів. Але, як було показано вище, зрозуміліше та краще спрацьовує підхід, що використовує лише бінарну систему рекомендацій (“сподобалося – не сподобалося”).

Прогнозування рекомендацій

Коефіцієнт подібності між користувачами використовується для прогнозування списку рекомендацій для користувачів. Для цього необхідно обрахувати для кожного товару hive-mind сумму (коефіцієнт колективної думки).

Для користувачів із заданої групи, які якимось чином оцінили товар (він їм сподобався або ні), необхідно виконати наступні обчислення: якщо користувачу сподобався товар, то до загальної суми додається коефіцієнт подібності поточного користувача з тим, що оцінював даний товар, в протилежному випадку – від суми віднімається коефіцієнт подібності між користувачами. Після цього отриманий результат ділиться на загальну кількість людей, що якимось чином оцінили даний товар.

Для кращого розуміння нижче наведений приклад коду, що обраховує прогнозовану оцінку товару для користувача:

```
class User
def similarity_with(user)
# Array#& is the set intersection operator. agreements = (self.likes & user.likes).size agreements += (self.dislikes & user.dislikes).size

disagreements = (self.likes & user.dislikes).size disagreements += (self.dislikes & user.likes).size

# Array#| is the set union operator
total = (self.likes + self.dislikes) | (user.likes + user.dislikes)
return (agreements - disagreements) / total.size.to_f end
def prediction_for(item) hive_mind_sum = 0.0
rated_by = item.liked_by.size + item.disliked_by.size
item.liked_by.each { |u| hive_mind_sum += self.similarity_with(u) } item.disliked_by.each { |u| hive_mind_sum -= self.similarity_with(u) }
return hive_mind_sum / rated_by.to_f end
end
```

Таким чином такий спосіб обрахунку прогнозів дозволить користувачам, що ще невстигли оцінити багато товарів, отримати прогнози від користувачів, що оцінили велику кількість товарів.

Формулювання вимог до веб-додатку

В результаті аналізу існуючих рішень рекомендаційних систем, прикладів реалізації різних інтернет-сервісів було обрано платформу Ruby on Rails. Тому, зважаючи на вибір фреймворку, сформулюємо ряд характерних вимог до реалізації веб-додатку:

1) Створення базового функціоналу типового інтернет-блогу

a. Авторизація та аутентифікація для користувачів.

b. Базові методи роботи зі статтями: створення, редагування, видалення.

c. Можливість динамічного створення секцій для впорядкування статей.

d. Додавання тегів для статей.

2) Імлементація рекомендаційної системи на базі методів API бібліотеки Recommendable:

a. Інтеграція з поточними моделями користувачів та статей.

b. Створення методів для оцінок статей користувачами.

c. Налаштування фонових процесів для прогнозування рекомендацій.

3. Аналіз результатів тестування.

Особливості середовища розробки

В даному підрозділі буде наведено покроковий процес установки та налаштування всіх необхідних додатків. Необхідно мати встановлену Unix-подібну операційну систему

Встановлення RVM та Rails

Ruby Version Manager (RVM) – це консольний додаток для керуванням версіями Ruby. Він дає можливість мати скільки завгодно версій Ruby, встановлених в одній системі, та кервути наборами бібліотек (gemsets lists).

Щоб встановити rvm в систему, необхідно ввести в командну строку (термінал) команду:

```
\curl -sSL https://get.rvm.io | bash -s stable
```

Після цього rvm стане доступним за допомогою команди rvm. Далі необхідно встановити ruby останньої версії за допомогою rvm:

```
rvm install 2.3.0
```

Необхідно завантажити бібліотеки bundler (менеджер бібліотек) та власне rails:

```
sudo gem install bundler
sudo gem install rails --no-ri --no-rdoc
```

Створення та налаштування проекту

Майже всі додатки Rails починаються однаково – з команди rails new. Ця команда створює скелет додатку в будь-якому каталозі (папці). Для початку треба обрати каталог для проектів, після цього запустити команду:

```
$ rails new blog_application create
create README.rdoc create Rakefile create config.ru create .gitignore create Gemfile create app
create app/assets/javascripts/application.js create app/assets/stylesheets/application.css create app/controllers/application_controller.
…
create test/test_helper.rb create tmp/cache
create tmp/cache/assets
create vendor/assets/javascripts create vendor/assets/javascripts/.keep create vendor/assets/stylesheets create vendor/assets/stylesheets/.keep run bundle install
.
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled
gem is installed.
```

Як можна побачити з цього лістингу ця команда створює структуру проекту, генерує деякі основні файли та виконує команду bundle install, що встановлює всі залежності, що перераховані в Gemfile. Після всіх команд буде побудовано додаток із наступною структурою папок:

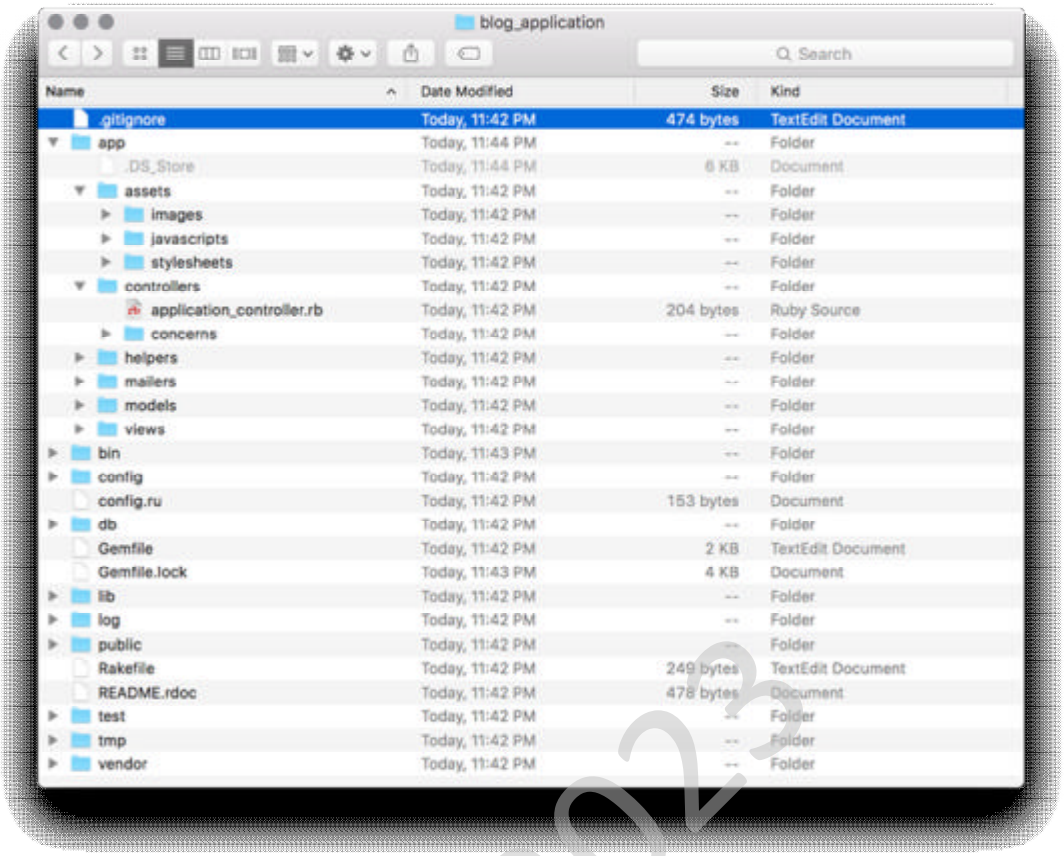

Рисунок 4.1 – Базова структура Rails-додатку

Після того, як проект було створено та згенеровано базову структуру необхідно додати усі бібліотеки для коректної роботи додатку. [14] Для цього необхідно додати усі залежності до Gemfile:

```
source 'https://rubygems.org'
gem 'rails', '4.2.6'
gem 'pg', '~> 0.18.4' gem 'sidekiq'
gem 'redis'
# Layout
gem 'slim-rails' gem 'sass-rails'
gem 'bootstrap-sass'
gem 'autoprefixer-rails'
gem 'therubyracer'

# Scripts
gem 'jquery-rails' gem 'turbolinks' gem 'uglifier'
gem 'coffee-rails'
# Utilities
```

```
gem 'puma', '~>3.1.0'
gem 'web-console', group: :development gem 'simple_form'
gem 'devise' gem 'nokogiri'
gem 'select2-rails'
gem 'acts-as-taggable-on', '~> 3.4' gem 'ransack'
gem 'dotenv'
# Recommender system gem 'recommendable'
group :development, :test do gem 'byebug'
gem 'rubocop' end
group :test do gem 'rspec'
end
```

Якщо не вказати для кожного гему корректної версії, то bundler автоматично підтягне найсвіжішу версію бібліотеки. На жаль, оновлення бібліотек інколи викликають незначні помилки. На вирішення цих проблем інколи необхідно витратити велику кількість часу, оскільки stack-trace цих помилок інколи незрозумілий для програміста.

Після кожного редагування Gemfile необхідно завжди виконувати команду bunlde install, щоб встановити всі залежності для додатку.

Далі необхідно коректно налаштувати підключення до бази даних, адже під час цього пункту часто виникає велика кількість помилок. В якості СУБД було обрано PostgreSQL версії 9.5, адже серед open-source СУБД PostgreSQL є лідером через велику швидкодію, зручний інтерфейс доступу та купу нових функцій та модулів, які інколи значно спрощують роботу.

Для підключення бази даних необхідно мати бібліотеку PG та адаптер postgresql встановленими в системі.

Далі налаштовується файл config/database.yml наступним чином:

```
default: &default adapter: postgresql encoding: unicode port: 5432
pool: 5
development:
<<: *default
username: alekseymazurik
database: mazurik_blog_development
```

```
test:
<<: *default
username: alekseymazurik database: mazurik_blog_test
production:
<<: *default username: postgres
password: password
database: mazurik_blog_production
```

Зрозуміло, що цей лістинг має містити власні username та password для доступу до системного postgres (окрім production – більш детально про розгортання у хмарі в наступних підрозділах). Після підготовки всіх файлів необхідно створити базу даних у системі для коректної роботи додатку. Для цього необхідно запустити команду $ rake db:create:all, що створить усі необхідні бази даних для середовищ розробки та тестування. [15]

На цьому етапі закінчена початкова конфігурація додатку і можливе його запущення командою bundle exec rails server. Після виконання цієї команди на локальному сервері буде розгорнуто додаток, який стане доступним за адресою localhost:3000 (стандартний для rails порт).

Особливості реалізації додатку на базі Rails

Додаток з базовою конфігурацією готовий, тепер необхідно спроектувати та створити додаток: веб-блог з вбудованою рекомендаційною системою.

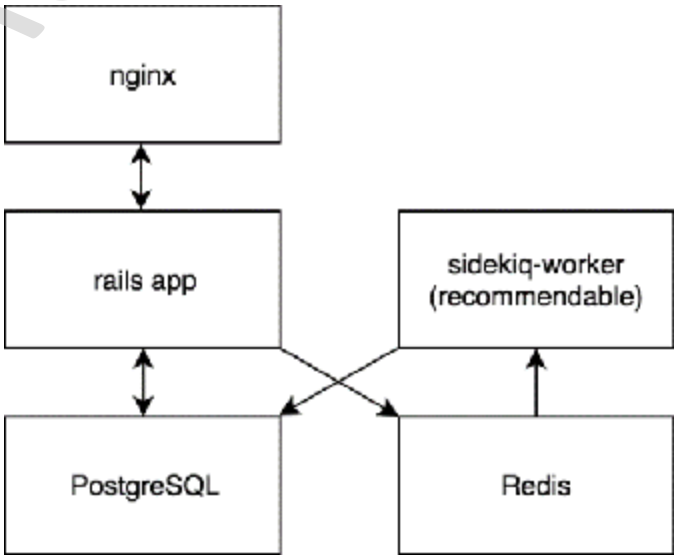

Рисунок 4.2 - Загальна архітектура веб-додатку

Задачею є створення додатоку, який зручно буде розмістити у хмарі із максимально простою архітектурою. В якості HTTP-серверу обрано nginx через його популярність та простоту налаштування. Сервер буде спілкуватися із rails додатком через HTTP-запити. В якості СУБД було обрано PostgreSQL.

Для коректної роботи та збільшення швидкодії додатку було вирішено виконувати обрахунки, що стосуються рейтингів в фоні, адже це зменшить навантаження на загальний проект. Для цього використовується технологія sidekiq, що дозволяє розгортати в фоні урізану версію додатку та виконувати обрахунки, використовуючи потоки, а не процеси (на відміну від rescue з його малою швидкодією).

Для того, щоб основний додаток міг спілкуватися з фоновим, використовується сховище пар ключ-значення. Найчастіше для таких дій використовують Redis через його простоту та високу швидкодію. Основний додаток буде додавати в чергу Redis запити на обрахунок потрібних рейтингів, а sidekiq після обробки поточних запитів буде витягувати та обробляти нові запити звідти.

Структура проекту Моделі

Концепція моделі в Rails майже не відрізняється від базового означення цього компоненту в MVC. Модель – це складова патерну проектування MVC, що відповідає роботу з базою даних та надає зручний інтерфейс доступу до них. В базовому додатку rails кожна модель наслідується від ActiveRecord::Base. Після цього при роботі з моделями можна використовувати допоміжні методи, що значно спрощують подальшу розробку проекту (додаються методи для валідації, роботи з помилками, витягання з бази необхідної інформації тощо). Детальний код створення моделей представлений в Додатку А.

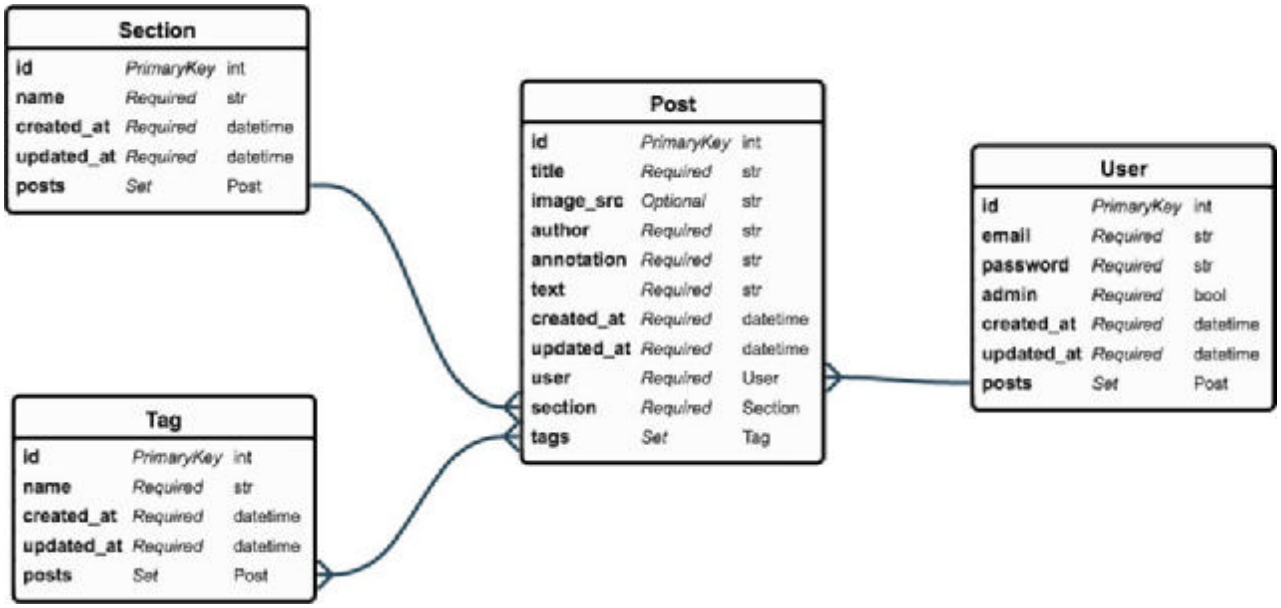

Рисунок 4.3 - Структура взаємозв’язків моделей блогу

Контролери

Контролери в Rails також беруть свій початок від базового означення. Контролер – це компонент патерну MVC, що забезпечує зв’язок між користувачами та системою: контролює введення даних користувачами та використовує моделі та представлення для реалізації необхідної реакції. Всі контролери наслідуються від базового контролера ApplicationController, що може містити методи, які мають бути доступними для всіх контролерів додатку.

Між контролерами та маршрутами існує прямий зв’язок. При розробці маршрутів в додатку на кожний маршрут має бути представлений метод контролера, який виконує ту чи іншу дію. Деякі методи можуть бути пустими: в цьому випадку як дія контролеру буде виконане лише надсилання представлення для цього маршруту. Увесь код контролерів представлений в Додатку А.

Представлення

Представлення в Rails – компонент, що відповідає за графічне відображення (візуалізацію) даних нашого додатку. В даному проекті для спрощеної та ефективнішої роботи з відображеннями (представленнями) було вирішено обрати шаблонізатор Slim, що дозволяє писати більш простий та

зрозумілий код, ніж базовий шаблонізатор Erb. Шаблонізатори необхідні в першу чергу для того, щоб вставляти на веб-сторінку частини коду на будь-якій мові програмування (в даному випадку – Ruby).

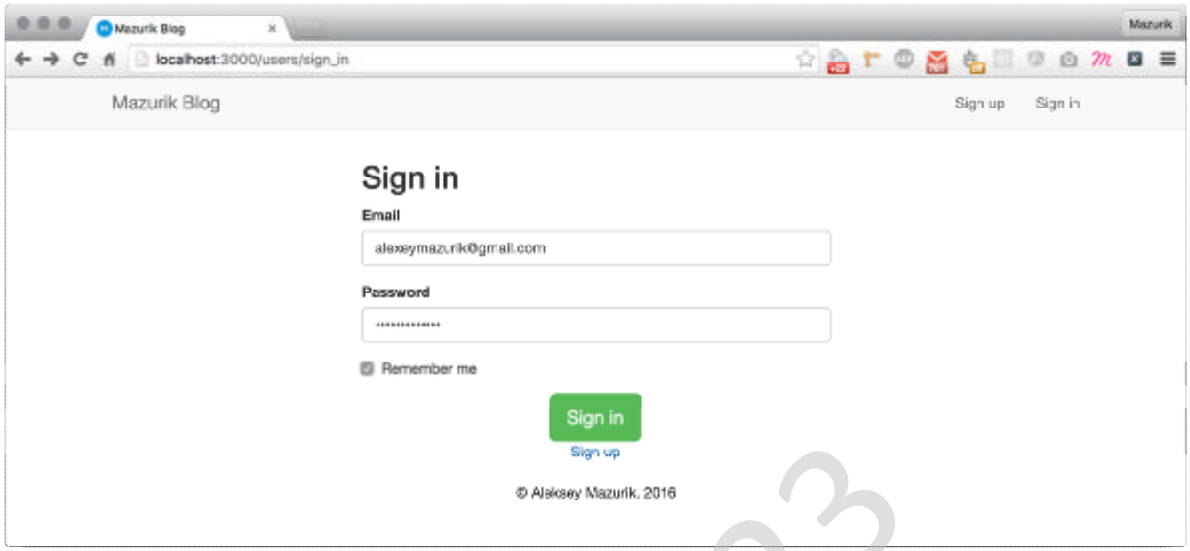

Рисунок 4.4 - Представлення сторінки входу

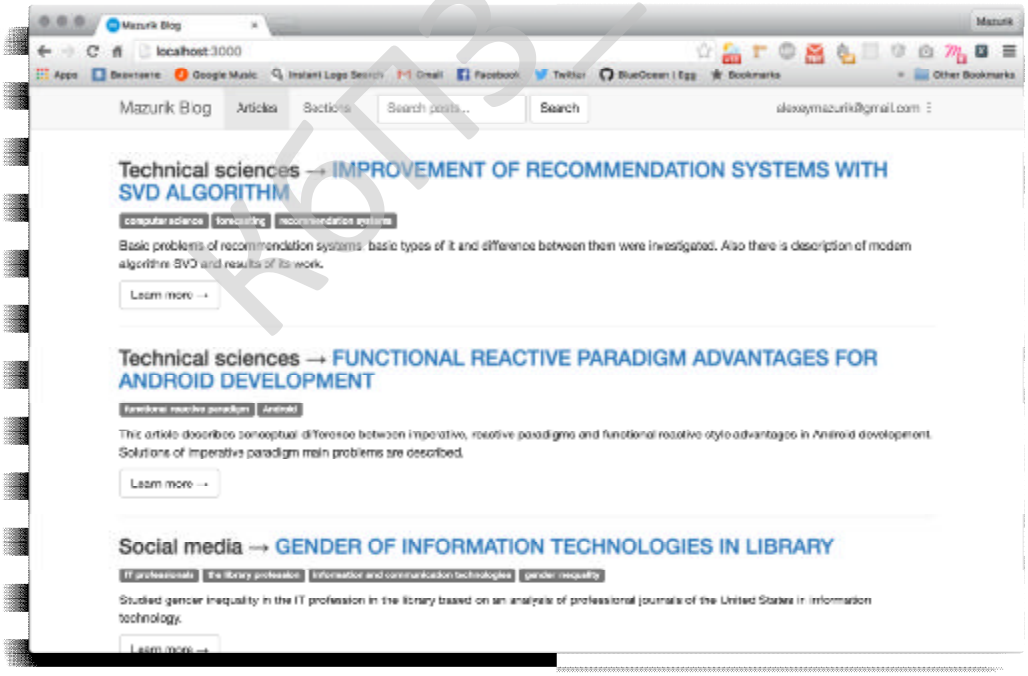

Рисунок 4.5 - Представлення сторінки зі всіма статтями

Маршрутизація

Метою маршрутизації (роутинга) в будь-якому Rails проекті є зв'язування вхідних url (адрес сторінок) з конкретною дією контролера. Роутер в Rails розпізнає url, з'єднує запит користувача з необхідним контролером та віддає результат обробки запиту. Також маршрутизатор використовується для генерації шляхів (маршрутів) в додатку без необхідності явно прописувати кожен маршрут. Для цього використовуються спеціальні структури, які зрозуміло та ефективно описують блоки маршрутів. Нижче представлений приклад роутингу в нашому додатку з коментарями:

```
Rails.application.routes.draw do
root to: 'posts#index' # переадресація головної сторінки
get 'manage', to: 'posts#manage' # маршрут для керування постами
resources :posts do # базові маршрути для статей
member do
# методи для оцінки постів (рекомендації) post :like
delete :like, action: :unlike, as: :unlike post :dislike
delete :dislike, action: :undislike, as: :undislike end
end
resources :sections, only: %w(index create update destroy) # маршрути для списку та керування секціями
resources :tags, only: %w(index) # список тегів
devise_for :users # роутинг для сторінок реєстрації та входу
end
```

Міграції та автоматичне заповнення бази даних

Міграції – це одна з функцій Active Record, що довзволяє змінювати схему бази даних. Замість того, щоб описувати зміни схеми на чистому SQL, міграції в Rails дозволяються використовувати простий Ruby DSL для опису змін у ваших таблицях.

Кожну міграцію можна розглядати як нову версію бази даних. З самого початку схема нічого не містить, а кожна наступна міграція змінює її, додаючи чи видаляючи таблиці, стовпці або записи. Active Record за допомогою вбудованих кодових структур знає, як оновлювати вашу схему час від часу. Active Record також оновлює файл db/schema.rb, щоб він був синхронізований з поточною

структурою вашої бази даних. Нижче наведений код міграції бази даних для створення таблиці статей:

```
class CreatePosts < ActiveRecord::Migration def change
create_table :posts do |t| t.string :title
t.text :text t.string :image_src t.string :author
t.text  :annotation t.references :user, index: true
t.timestamps null: false end
end end
```

Між міграціями можна зручно переключатися як вперед, так і назад.

Файли міграцій необхідно завжди називати декларативно. Перша частина має містити випадково згенероване число (зазвичай – поточна дата і час, записані без розділових знаків), а друга – дію, що виконує ця міграція.

Після створення міграції необхідно промігрувати базу даних до найновішої версії за допомогою команди rake db:migrate.

Для початкового заповнення бази даних також інколи використовують міграції. Але більш коректний спосіб – сидування бази даних. Для цього необхідно написати код на Rails, який буде записувати в базу даних потрібну інформацію, використовуючи API Active Record.

Щоб наповнити базу текстами статей, було використано ресурс http://www.inter-nauka.com/, де розміщені свіжі наукові публікації для різних секцій, тому можна доволі повно заповнити ресурс статтями різних розділів, щоб прослідкувати крок за кроком роботу рекомендаційного алгоритму, вбудованого в систему.

Для парсингу даних з обраного ресурсу вирішено було використовувати бібліотеку Nokogiri, оскільки вона має об’ємний функціонал для парсингу різних типів документів, оформлених на мові розмітки (HTML, XML), та є на сьогодні швейцарським ножем для парсингу веб-сайтів будь-якої складності. Код для парсингу представлений нижче:

```
require 'open-uri'
SECTIONS = [ 'Technical sciences', ... ]
```

```
ARTICLE_URLS = [ 'http://www.inter-nauka.com/issues/2015/9/592/', ... ]
user = User.create!( email: 'admin@mazurik.me', password: 'password',
password_confirmation: 'password', admin: true
)
SECTIONS.each do |section| Section.create!(name: section)
end
ARTICLE_URLS.each do |url|
en_url = url.split('/').insert(3,'en').join('/')
page = Nokogiri::HTML(open(en_url))
title = page.css('div[record] .page-header a').text author = ''
authors = page.css('div[record] .page-header + .row a') if authors.size ==
1
author = authors.text elsif authors.size > 1
authors.each_with_index { |a, i| i == (authors.size - 1) ? author +=
a.text : author += a.text.concat(', ') }
end
basic_info = page.css('div[record] br+div p') basic_info.map do |p|
p.css('strong').remove end
summary, keywords = basic_info.map(&:text)
keywords = keywords.chop.split(', ')
section = page.css('b:contains("Branch of science:") + a').text
# Parsing text of an article
page = Nokogiri::HTML(open(url)).css('hr + div')
page.search('div[style="text-align: center;"]').each { |div| div.remove }
page.search('p').each do |p|
p.remove
break if p.text.downcase.include?('key words:') end
article_text = page.children.to_html.strip
Post.create!( title: title, text: article_text, annotation: summary,
user_id: user.id, author: author, section_id: Section.where(name:
section).first.id, tag_list: keywords
)
End
```

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Робота алгоритму програми рекомендаційної системи полягає у використанні аналізу даних та алгоритмів для надання користувачам персоналізованих рекомендацій. Програма рекомендаційної системи взаємодіє зі

збереженими даними користувачів і предметів, а також використовує визначений алгоритм для генерації рекомендацій.

В процесі роботи програма аналізує дані про взаємодію користувачів і предметів, щоб визначити патерни та уподобання. Це включає в себе аналіз історії переглядів, покупок, чи будь-яких інших взаємодій користувача з системою.

На основі цього аналізу програма використовує алгоритми рекомендації, які можуть бути основаними на фільтрації на основі вмісту, колаборативній фільтрації, або гібридних методах. Ці алгоритми враховують інформацію про користувачів та предмети, щоб здійснити персоналізовану рекомендацію.

У деяких випадках, особливо в системах, що використовують машинне навчання, програма може бути здатна адаптуватися до змін у взаємодії користувачів та оновлювати свої рекомендації з часом.

Загалом, робота програми рекомендаційної системи базується на поєднанні збору та аналізу даних, вибору та використання алгоритмів рекомендацій, і постійному вдосконаленні через адаптацію до нових даних та умов.

Блок-схема надає можливість за допомогою графічних блоків та елементів, представити роботу додатку у зручній для сприйняття формі.

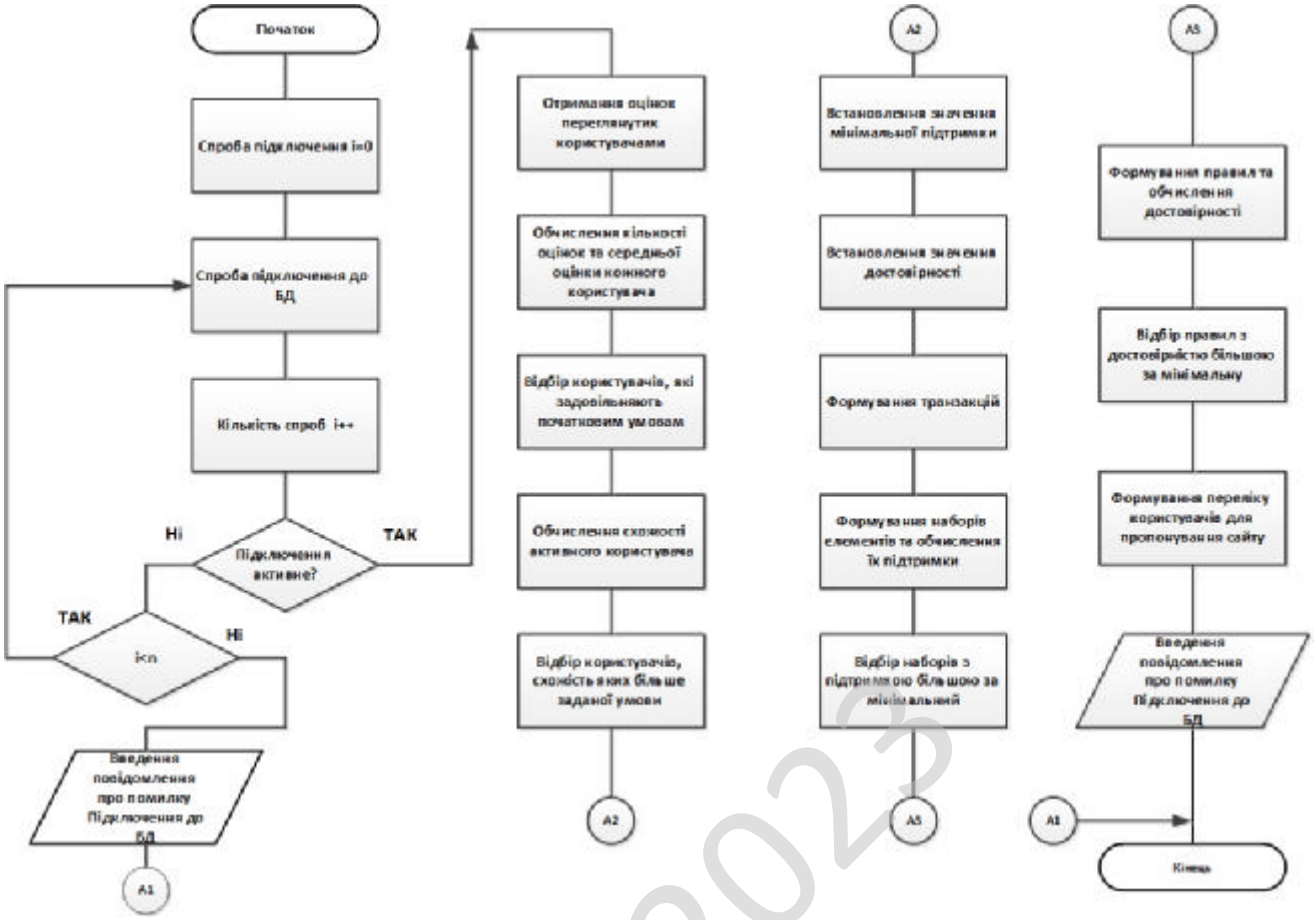

Рисунок 4.6 - Блок-схема алгоритму роботи рекомендаційної системи

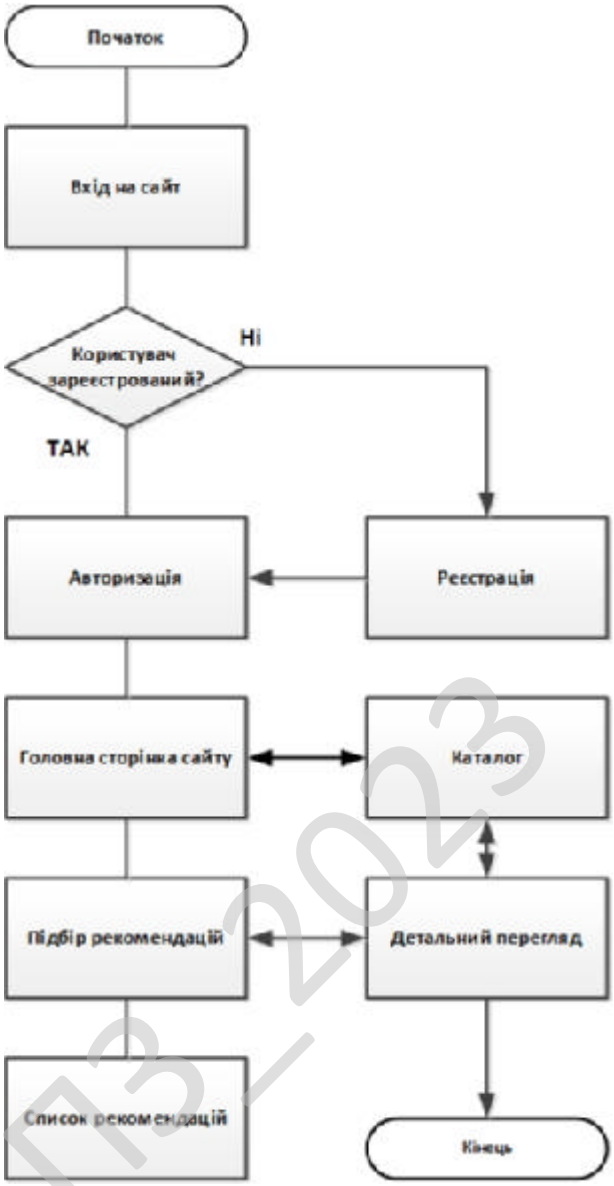

Рисунок 4.7 - Блок-схема роботи підпрограми алгоритму рекомендаційної системи

Висновок

Розробка веб-додатків за допомою Rails – ефективне рішення, якщо необхідно швидко розробити доволі складний, але типовий інтернет-сервіс, використовуючи патерн MVC. Це рішення широко застосовується для створення комерційних інтернет-сервісів (магазини, журнали тощо).

Використання рекомендаційних систем в додатках значно збільшує прибутковість сервісів, оскільки збільшується залученість користувачів до вашого сервісу. База Rails дозволяє створювати гнучкі рішення стосовно

імлементації готових систем до вашого сервісу або написання їх з нуля. В будь-якому випадку це не буде великою проблемою, оскільки Rails використовує доволі зрозумілі методи для інтеграції зовнішніх або внутрішніх модулів до існуючої системи.

В даному розділі було детально описано процес реалізації веб-додатку з використанням Rails, проаналізовано обумовленість вибору фреймворку для розробки інтернет-блогу, особливості реалізації рекомендаційної системи. Звичайно, було протестовано та проаналізовано модуль веб-додатку, що відповідає за надання користувачам сервісу рекомендації.

В результаті дослідження роботи реалізованої рекомендаційної системи було зроблено висновок, що система поводить себе коректно, рекомендації – релевантні. Тому є сенс використовувати поточне рішення в контексті інтернет-маркетингу, щоб збільшувати процент конверсії, тобто залучати користувачів до необхідних дій, а саме – рекомендувати якомога більше релевантного контенту, щоб заслужити довіру користувачів.

4.2 Захист розробленого програмного забезпечення

Захист сайту від взлому - це критичний аспект веб-розробки. Нижче подано загальні рекомендації, які можна використовувати для забезпечення безпеки веб-сайту:

Оновлення програмного забезпечення

Регулярно оновлюйте веб-сервер, операційну систему та всі використовувані бібліотеки та фреймворки.

Захист від SQL-ін'єкцій

Використовуйте параметризовані запити та підготовлені вирази для уникнення SQL-ін'єкцій.

Використання HTTPS

Захищайте передачу даних між користувачем і сервером шляхом використання протоколу HTTPS.

Сильні паролі

Вимагайте від користувачів використовувати складні паролі, і використовуйте хеш-функції для збереження паролів в базі даних.

Обмеження доступу

Обмежте права доступу до файлів та каталогів на сервері, використовуючи налагодження прав доступу.

Захист від кросс-сайтового сценарію (XSS)

Валідуйте та екрануйте вхідні дані, і використовуйте безпечні API для вставки даних на сторінки.

Моніторинг та журналювання

Встановіть системи моніторингу та журналювання для виявлення можливих атак та аномальних дій.

Захист від кросс-сайтового запиту (CSRF)

Використовуйте механізми, такі як токени безпеки, для захисту від атак CSRF.

Файрвол

Налаштуйте файрвол для фільтрації неправильних запитів і блокування небезпечних IP-адрес.

Регулярні аудити безпеки

Проводьте регулярні аудити безпеки, включаючи тестування на проникнення, для виявлення та виправлення потенційних уразливостей.

Захист від отримання неправомірного доступу

Використовуйте двофакторну аутентифікацію для ускладнення процесу отримання доступу до системи.

Регулярні резервні копії

Регулярно створюйте резервні копії ваших даних і перевіряйте їх відновлення.

Розробка захисту розробленого ПЗ

Для захисту даних розробленого програмного забезпечення веб-системи для командного керування проектами було використано стандартні функції мови Ruby: password_hash() і password_verify().

Password_hash() використовується для хешування паролю користувача, а password_verify() – для перевірки вказаного паролю на відповідність до хешу.

Першим параметром функції password_hash() являється пароль, який необхідно захистити, а другий параметр визначає алгоритм за яким буде обчислюватись хеш. На даний момент підтримуються два алгоритми хешування:

– PASSWORD_DEFAULT – використовує BCrypt алгоритм, що виводить хеш у розмірі 60 або більше символів;

– PASSWORD_BCRYPT – використовує алгоритм CRYPT_BLOWFISH для генерації хешу. В даному випадку результатом буде завжди рядок з 60 символів.

Головною перевагою даної функції є те, що програмісту не потрібно піклуватися про значення “солі” і вартості обчислення хешу, тому що дана функція все робить за програміста. Якщо ж програміст бажає використовувати власне значення “солі”, то він має передати його у третьому параметрі функції.

Приклад використання даної функції:

<?php echo password_hash("rasmuslerdorf", PASSWORD_DEFAULT); ?>

Результатом виконання функції буде наступне значення:

$2y$10$.vGA1O9wmRjrwAVXD98HNOgsNpDczlqm3Jq7KnEd1rVAGv3Fykk1a

Приклад з передачею свого значення “солі” параметром функції:

```
<?php
	$options = [ 'cost' => 12 ];
	echo password_hash("rasmuslerdorf", PASSWORD_BCRYPT, $options);
	?>
```

Результатом виконання функції буде наступне значення:

$2y$12$QjSH496pcT5CEbzjD/vtVeH03tfHKFy36d4J0Ltp3lRtee9HDxY3K

Як бачимо два значення відрізняються один він іншого.

Також сайт було захищено від таких атак як: XSS-атаки та SQL-ін’єкції.

Від XSS-скриптів сайт було захищено наступним чином: достатньо замінювати спеціальні символи "<" и ">" на "<" и ">", що виконує php-функція htmlspecialchars та strip_tags. Також можна написати у файлі .htaccess такий код:

```
Options +FollowSymLinks
RewriteEngine On
RewriteCond %{QUERY_STRING} (\<|%3C).*script.*(\>|%3E) [NC,OR]
RewriteCond %{QUERY_STRING} GLOBALS(=|\[|\%[0-9A-Z]{0,2}) [OR]
RewriteCond %{QUERY_STRING} _REQUEST(=|\[|\%[0-9A-Z]{0,2})
RewriteRule ^(.*)$ index.php [F,L]
```

Для захисту сайту від SQL-ін’єкцій було використано php-функцію mysql_real_escape_string. Ця функція екранує спеціальні символи в рядку, що використовуються в SQL-запиті, беручи до уваги кодування з'єднання.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Реєстрація домена

Реєстрація доменів - процес внесення в реєстр зони першого рівня запису про нове доменне ім’я. Процедура реєстрації домена проста, для цього достатньо зареєструвати аккаунт у реєстратора доменних імен, поповнити рахунок, перевірити доменне ім'я на зайнятість і створити заявку, якщо доменне ім'я виявилося вільним. Після реєстрації домену (внесення до реєстру запису, що містить дані адміністратора, реєстратора, дати реєстрації та її закінчення, стан делегування), доменне ім'я готове для використання через, як правило від 5 до 10 хвилин.

Для використання домену, необхідно вказати для нього в інтерфейсі реєстратора (делегувати) dns сервера (хостинг).

Реєстратор доменних імен - організація, уповноважена створювати (реєструвати) нові доменні імена і подовжувати термін дії вже існуючих доменних імен в домені, для якого встановлено обов'язкову реєстрацію. Такими доменами є:

- домен нульового рівня (кореневий домен);
- всі домени верхнього рівня (першого рівня);
- деякі домени другого рівня (наприклад, com.ru або co.uk).

У всіх інших доменах для створення субдоменів спеціальних повноважень не потрібно.

Хостинг

Хо́стинг (англ. hosting) — послуга, що включає надання дискового простору, підключення до мережі та інших ресурсів для розміщення фізичної інформації на сервері, що постійно перебуває в мережі (наприклад Internet).

Поняття хостингу включає в себе широкий спектр послуг із використанням різного апаратного та програмного забезпечення. Зазвичай під поняттям послуги хостингу мають на увазі, як мінімум, послугу розміщення файлів сайту на сервері, на якому запущене ПЗ, необхідне для обробки запитів до цих файлів (веб-сервер). Як правило, до послуг хостингу вже входить

надання місця для поштової кореспонденції, баз даних, DNS файлового сховища тощо, а також підтримка функціонування відповідних сервісів, однак вони можуть надаватися і окремо. А саме, послуга може бути обмежена розміщенням поштової кореспонденції та відповідного ПЗ (поштовий хостинг), клієнтських файлів (файловий хостинг), виключно відео файлів (відео хостинг) або інших файлів певного типу та за певними умовами.

Послуги хостингу можуть надаватися у пакеті з іншими інформаційними послугами, такими як реєстрація доменного імені, створення сайту, надання додаткового ПЗ тощо.

Провайдерами хостингу можуть виступати як компанії, що спеціалізуються на цих послугах («хостери»), так і великі провайдери інформаційних послуг, що спеціалізуються на інших послугах (такі як Google, Microsoft, Yahoo та інші).

Розрізняють безкоштовний та платний хостинг. Безкоштовні «хостери» заробляють на тому, що розміщують рекламу на своїх сайтах або на наданні інших платних послуг (у пакеті з безкоштовними або опціонально).

Завантаження вихідного коду

Після придбання доменного імені та послуг по хостингу необхідно завантажити на сервер вихідний код розробленого ПЗ за допомогою програм на кшталт FileZilla та перевірити правильність роботи сайту.

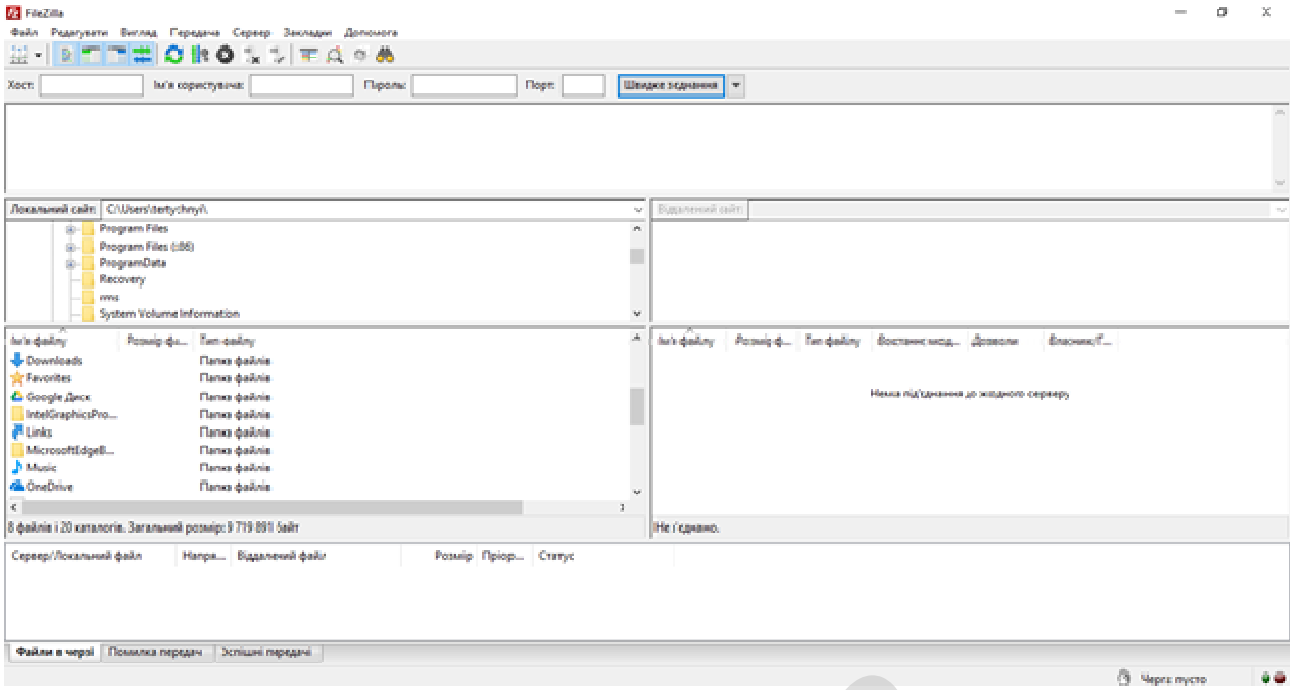

Рисунок 5.3 – Головне вікно програми для завантаження файлів на сервер

Після завантаження на сервер БД сайту можна керувати через спеціальні додатки.

Веб-додаток phpMyAdmin з відкритим кодом на мові PHP із графічним веб-інтерфейсом для адміністрування бази даних.

phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Ця програма користується великою популярністю у веб-розробників, оскільки дозволяє керувати базу даних MySQL без вводу SQL команд через дружній інтерфейс і з будь-якого комп'ютера під'єднаного до інтернету без необхідності встановлення додаткового програмного забезпечення.

На сьогоднішній день phpMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт, з огляду на всі нововведення СКБД MySQL. Переважна більшість українських провайдерів використовують цей застосунок як панель керування для того, щоб надати своїм клієнтам можливість адміністрування виділених їм баз даних.

Реалізація запуску системи управління

Користувацький інтерфейс системи має зручний та інтуїтивно зрозумілий інтерфейс, що забезпечує низький поріг входу для користування системою.

Система тестування має простий графічний інтерфейс, щоб максимально спростити тестування системи. Надано опис основних компонентів програмного забезпечення та наведено детальний опис роботи додатку. Створено інструкцію користувача програмного забезпечення, що показує послідовність дій при роботі з додатком.

Після запуску програми, відображається вікно завантаження програми.

Після завантаження програми відображається вікно авторизації у системі.

Для авторизації необхідно ввести логін користувача та пароль.

Якщо дані користувача невірні, отримуємо повідомлення про некоректно введені дані користувача.

Щоб ввести дані повторно, потрібно натиснути на кнопку “TRY AGAIN”.

Користувачу відобразиться вікно авторизації для повторного введення даних авторизації.

Якщо користувач коректно ввів свої дані, йому відобразиться вікно з вибором функціоналу системи:

Користувач може вибрати один із доступних варіантів використання системи зі списку.

При виборі варіанту «Моніторинг системи» відкривається вікно функціоналу «Моніторинг системи».

При виборі одного з варіантів для моніторингу користувачу відобразиться діалогове вікно зі станом вибраного пристрою:

При виборі варіанту «Операції з системою» відкриється вікно функціоналу «Операції з системою».

При виборі одного з варіантів у списку відобразиться вікно з поточним станом вибраного пристрою та дві кнопки: з можливістю встановити новий стан для пристрою або поверненням до попереднього меню.

Далі необхідно за допомогою spinner`а вибрати нову температуру та натиснути ACCEPT.

В результаті натиснення на одну із кнопок користувач повернеться в меню вибору «Операції з системою». Для повернення до головного меню потрібно зробити свайп вліво.

При виборі планувальника завдань з головного меню програми відкриється вікно функціоналу «Планувальник».

При виборі одного з варіантів у списку відобразиться вікно з годинником для встановлення часу для виконання завдання.

Користувачу необхідно встановити час, коли буде виконано вибрану операцію, та натиснути кнопку «ОК» або «Cancel» для відміни.

Для перегляду списку запланованих завдань потрібно зробити свайп вправо.

Видалити завдання зі списку можна свайпом вправо на вибраному завданні, редагувати завдання можна за допомогою затримування пальця на вибраному елементі списку та натисненні кнопки редагувати.

Отже у проекті даної системи задоволені усі визначені раніше вимоги.

6 НАУКОВА НОВИЗНА

У магістерській роботі розроблено програмне забезпечення, яке призначено для побудови рекомендаційної системи для просування сайту компанії.

Метою роботи є широкий аналіз існуючих підходів до створення рекомндаційних систем, та реалізація додатку із вбудованою системою рекомендацій.

Об'єктом дослідження є процес формування особистих рекомендацій на основі вподобань користувачів.

Предметом дослідження є методи та моделі формування особистих рекомендацій.

Методи дослідження базуються на методах інформаційного пошуку, методах розробки програмного забезпечення та методах побудови рекомендаційних систем.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– проаналізовано основні алгоритми та базові підходи для реалізації рекомендацій, описано особливості, переваги та недоліки кожного з них та виконано порівняльний аналіз існуючих підходів;

– розроблено підхід, який поєднує в собі фільтрацію даних, на основі вмісту з колаборативною та демографічною фільтрацію;

– зроблено огляд допоміжних засобів для кореляції результатів, отриманих після застосування основних алгоритмів;

– розроблено вітчизняний продукт управління рекомендаційною системою, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми дипломного проекту

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведено дослідження та виконана програмна реалізація технології побудови рекомендаційної системи для просування товарів і послуг компанії в мережі Інтернет.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликі системні потреби;

б) незалежність від встановлених на комп'ютері баз даних;

в) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	19
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	19000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i\,, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45\cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78\cdot(0{,}88\cdot 0{,}93\cdot 0{,}88\cdot 0{,}91\cdot 0{,}95\cdot 1\cdot 1\cdot 0{,}87\cdot 1{,}22\cdot 1{,}16\cdot 1{,}1\cdot 1{,}1\cdot$$
$$\cdot 1{,}12\cdot 1{,}1\cdot 1\cdot 1{,}1\cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S\,, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4). Для вибраної мови програмування він складає 2,66;

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо *S* =100 %

$$T_{РП} = 0{,}3\cdot 2{,}66\cdot 9{,}37^{0{,}33+0{,}2(1{,}026\text{-}1{,}01)}\cdot 100 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{pq} - H_{ев}}, \quad (7.5)$$

де: F_{pq} – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3{,}8 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	11	990	16,5
Монітор	60	11	660	11
Клавіатура	30	11	330	5,5
Маніпулятор «мишка»	30	11	330	5,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	3	360	6
Принтер струминний	60	2	120	2
Сканер	20	1	20	0,33
Концентратор–маршрутизатор	30	5	150	2,5
Кабельні господарства ЛВС на 1 м. п.	2,5	400	1000	16,67
Копіювальний апарат	140	2	280	4,67
Усього за рік:			$З_ч$	70,67

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф^{с}_{др} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф^{с}_{др} = \frac{71 \cdot 3}{1,2} = 177,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел}$ = 177,5/(60·8) = 0,4 ставки.

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2022, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12525	37575
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	0,4	11500	13800
Інженер-системотехнік	0,25	11500	8625
Адміністратор мережі	0,5	11500	17250
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 7,45	-	$Ф_{роб}$ = 268200

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{268200}{7,45 \cdot 60} = 600 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторалъ» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно $8 м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 24.10.23 – джерело http://computorg.ua/ru/price.html

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	AMD Ryzen 5 2400G (YD2400C5M4MFB) AM4, 4 ядра, 8 потоків, 3.6 GHz, 3.9 GHz, TDP - 65 Вт, 14nm, L1: 384KB, L2: 2MB, L3: 4MB, Radeon Vega 11, Zen, Tray	-
Системна плата	GIGABYTE A520M H сокет - AM4, DDR4, 64 ГБ, 5100 MHz, LAN - 1 Гбіт/с, DVI, HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	Radeon Vega 11	-
Жорсткий диск	SSD 2.5" 240GB SA100 Acer, 3D TLC NAND, 2.5", SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 16GB 2666 MHz Essentials Mushkin	-
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slim Black Bulk (DVR-TD10RS)	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA-489, PSU 550W(FSP Brand: ATX-350PNR, 12cm), black, (front bezel – black+light silver; body material – 0.6mm), 80mm fan (rear), 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відраху-вання, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	28000	25	7000
Всього по групі	130690	-	27797,5
7. Нематеріальні активи	19000	10	1900
Разом	$К_р$ = 1672575		$А_р$ = 157540

Примітка: вартість автомобіля взята по даним з автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 97500 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 600 \cdot 209/19 = 6600 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 6600 \cdot 10 \cdot 0{,}01 = 660 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(6600+660) = 1597 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картриджів, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих норм приймаємо 1/6 пачки паперу на три місяці розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 210$ грн., визначаємо вартість паперу за період розробки $N_м = 3$ міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 210 \cdot 1/6 \cdot 3 = 105 \text{ грн.}$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 4 CD +1 DVD:

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 20 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 27 грн./шт.

$$З_{М2} = 20 \cdot 4 + 27 = 107 \text{ грн.}$$

Згідно виданих норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_М = (105 + 107 + 1702)/19 = 101 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п = 15\%$) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$O_n = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	6600
2. Додаткова зарплата виконавців	$З_д$	660
3. Відрахування на соціальні потреби	$С_{оц}$	1597
4. Загальногосподарські витрати	$Г_{осп}$	990
5. Витрати на матеріали	$З_М$	101
6. Освоєння нових операційних систем, мов програмування	$О_п$	990
7. Амортизація основних фондів	$А_м$	2073
8. Повна собівартість програмного забезпечення	$С_п$	13011
9. Плановий прибуток	$П_р$	7156,05
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	20167,05
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4033,41
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	24200,46

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм (N_e = 19 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_e \cdot 12}, \quad (7.20)$$

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 157540 \cdot 3/(19 \cdot 12) = 2073 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 6600+660+1597+990+101+990+2073 = 13011 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 55 \cdot 13011 = 7156 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	24200
Всього капітальних витрат	–	24200

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на налаштування та технічне обслуговування	$З_р$	32208	18788
2. Витрати на електроенергію	$З_{ел}$	5320	4560
3. Витрати на амортизацію	$З_{ам}$	0	6050
Всього витрат за рік	$І$	37528	29398

Витрати на налаштування та технічне обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де: $Т_р$ – кількість годин налаштування та обслуговування за рік, *год.*;

$З_г$ – заробітна плата персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин на проведення робіт по налаштуванню і обслуговуванню системи зменшилася з 240 до 140 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 32208 \; грн,$$

$$З_{р\,нов} = 140 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 18788 \; грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 0{,}35 \cdot 4000 \cdot 3{,}8 = 5320 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}3 \cdot 4000 \cdot 3{,}8 = 4560 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	24200	–	6050
Всього відрахувань	-	–	24200	–	6050

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (20167 - 13011) \cdot 19 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 114885 +$$
$$+0{,}25 \cdot 33190 + 0{,}2 \cdot 97500 + 0{,}1 \cdot 19000) \cdot 3/12 = 96579 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{1672575}{(20167 - 13011) \cdot 19 \cdot 12 / 3} = 3 \text{ роки}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	19
2. Повна собівартість розробленої програми	Грн.	13011
3. Ціна розробленої програми	Грн.	20167
4. Плановий прибуток від реалізації розробленої програми	Грн.	7156
5. Рентабельність програмної продукції	%	55
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1672575
7. Загальний прибуток від реалізації програмної продукції	Грн.	135964
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	96579
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	3
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	25380
11.Величина економічного ефекту у користувача програмної продукції	Грн.	2080
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	2,9

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (37528\text{-}29398)\text{-}0{,}25 \cdot 24200 = 2080 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \quad (7.28)$$

$$T_{сп} = \frac{24200}{37528 - 29398} = 2{,}9 \text{ року}.$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Важливий фактор ергономіки - відсутність шуму на робочому місці. Якщо ви довго працюєте з комп'ютером, який шумить, то це стане фактором підвищеної стомлюваності.

Відомо, що шум несприятливо діє на слуховий аналізатор та інші органи та системи організму людини. Визначальне значення щодо такої дії має інтенсивність шуму, його частотний склад, тривалість щоденного впливу, індивідуальні особливості людини, а також специфіка виробничої діяльності. Ті види діяльності, у яких поєднується напружена розумова робота та інтенсивне використання комп’ютера (редагування тексту, верстка оригіналу, "запуск" та відлагодження програм тощо) характеризується відчутним впливом навіть незначних рівнів шуму. Цей вплив виражається у зниженні розумової працездатності, швидкій втомлюваності, послабленні уваги, появі головного болю

Природно, що чим вище частота процесора, тим вище буде його енергоспоживання, що доходить в даний час до півтора сотень Вт. Відповідно виникає потреба у потужному охолодженні. І відповідно зростає рівень шуму, що створюється системою вентиляції. Тема магістерської роботи є дослідження та програмна реалізація програмного продукту , то робота потребує відповідного психофізіологічного навантаження і постійної роботи за комп’ютером , то розгляд впливу шуму на людський організм є актуальною темою.

8.2 Аналіз впливу шуму на користувача ПК

Від шуму в першу чергу страждає слуховий апарат. Під його впливом виникають явища стомлення органу слуху і ослаблення слуховий здібності. Ці явища не носять стійкого характеру і з припиненням шуму швидко проходять. Однак якщо перевтома органу слуху повторюється систематично протягом тривалого терміну, то розвивається приглухуватість. При професійної приглухуватості, як правило, відбувається порушення сприйняття частот в діапазоні від 4000 до 8000 Гц.

Постійний вплив сильного шуму може не тільки негативно вплинути на слух, але й викликати інші шкідливі наслідки - дзвін у вухах, запаморочення, головний біль, підвищення втоми. Шум має акумулятивним ефектів, тобто акустичні подразнення, накопичуючись в організмі, все сильніше пригнічують його. Людина, постійно піддається впливу шуму, швидко перевтомлюється, відрізняється підвищеною дратівливістю, частіше страждає від слабкості і запаморочення.

Підступність шуму пояснюється не тільки його прямою дією на барабанну перетинку, а в подальшому і на стовбурові та коркові структури мозку, а й тим, що в процесі передачі нервових імпульсів відбувається їх взаємодія з іншими областями мозку, де розташовані центри серцево-судинної, дихальної та інших систем життєдіяльності. При цьому нервові імпульси викликають підвищення тонусу судин і, як наслідок, артеріального тиску, приводячи до появи дисфункцій, а, в кінцевому рахунку - до розвитку гіпертонічної хвороби.

Негативний вплив шуму позначається не тільки на серцево-судинній системі. Від шуму через нервові шляхи страждають органи травлення. У першу чергу шлунок і печінку, порушується моторика кишечника, як наслідок розвивається виразкова хвороба.

Найбільшої шкоди завдає нехай і менш інтенсивний, але постійний шум. При цьому свій згубний вплив робить навіть шум, не відчувається вухом людини: інфразвуки, що викликають почуття тривоги і болю в хребті, а при тривалому впливі позначаються на порушенні периферичного кровообігу. На думку вчених, саме інфразвуками, нечутно проникають крізь самі товсті стіни, викликаються багато нервові хвороби жителів великих міст.

Ультразвуки, що займають помітне місце в гамі виробничих шумів, також небезпечні. Механізми їх дії на живі організми вкрай різноманітні і до кінця не вивчені. Проведені ще в радянський період дослідження показали, що у людей, що працюють в галасливій обстановці, зокрема, падає продуктивність праці (на 10%) і збільшується захворюваність (на 37%). Зростання хвороб спостерігається після роботи протягом 8-10 років при впливі шуму з інтенсивністю вище 70-80 дБ - нормативу для робочих місць. У більшості великих міст цей поріг значно перевищено. Саме цим пояснюється боротьба з галасливими засобами пересування (в основному автомобілями та літаками) в Європі.

Виявлені негативні наслідки впливу шуму комп'ютера на організм людини змушує серйозно підійти до проблеми боротьби з ним. Джерел шуму в комп'ютері декілька: це і вентилятори, охолоджуючі блок живлення, процесор і графічну плату, а також приводи оптичних і жорстких дисків. В результаті генерується досить широкий спектр звуків, причому кожен комп'ютер відрізняється в цьому сенсі своєї "індивідуальністю".

Слід враховувати, що корпус комп'ютера грає до того ж роль резонатора: він привносить у спільну картину шуму низькочастотні складові. В одному відомому конструкторському бюро співробітники намагалися захистити себе від шуму та вібрації, що генеруються робочою станцією, обклавши її важкими томами енциклопедії. Сумісні приміщення, в яких рівні шуму і вібрації не перевищують допустимих значень, не повинні межувати з виробничими приміщеннями для роботи з ВДТ.

Звукоізоляція огороджувальних конструкцій і звукопоглинання приміщень з ВДТ має забезпечити параметри шуму, які наведені в таблиці 8.1 Допустимі рівні звуку, еквівалентні рівні звуку, рівні звукового тиску в октавних смугах частот.

Згідно СанПіН 3.3.2.007 -98 «Державні санітарні правила і норми з візуальними дисплейними терміналами електронно – обчислювальних машин» рівень шуму на робочих місцях користувача ПК не має перевищувати 50 дБА, що досягається застосуванням малошумного обладнання, використанням спеціальних матеріалів для обшивки приміщень, а також різноманітними звукопоглинальними пристроями.

Таблиця 8.1 - Допустимі рівні звуку, еквівалентні рівні звуку, рівні звукового тиску в октавних смугах частот.

Вид трудової діяльності, робочі місця	Рівні звукового тиску, дБ в октавних смугах із середньо геометричними частотами, Гц									
	1,5	3	25	50	00	000	000	000	000	Рівні звуку, еквівалентні рівні звуку дБА/дБАеке
Програмісти ЕОМ	6	1	1	4	9	5	2	0	8	0/5
Оператори в залах оброб-лення інформації на ЕОМ та оператори комп’ютер-ного набору	6	3	4	8	3	0	7	5	4	5/6
У приміщен-нях для розта-шування шумних агрегатів ЕОМ	0,3	1	3	7	3	0	8	6	4	5/7

8.3 Методи боротьби з виробничим шумом та вібрацією

Для зменшення шуму доцільно буде замінити систему охолодження. Уже давно існують системи водяного охолодження з рівнем шуму менше 20 дБ. Однак вони досить дорогі і громіздкі. їх бажано застосовувати в серверних рішеннях, а ось розумність їх використання в персоналках здається сумнівною. В персоналках рекомендується замінити корпус. Чим він просторіше і зручніше, тим краще, але і дорожче. Достатній обсяг всередині корпусу і вільне розміщення комплектуючих допоможе створити сприятливу аеродинаміку всередині. Відповідно, тепло від процесора буде відходити вільніше, і вентилятору не треба буде так "напружуватися".

Основними методами боротьби з виробничим шумом і вібрацією є:

1. зменшення шуму в джерелі;
2. звукопоглинання і вібропоглинання;
3. звукоізоляція і віброізоляція;
4. акустична обробка приміщень;
5. зменшення шуму на шляху його поширення;
6. раціональне планування підприємства і цехів;
7. установка глушників шуму;
8. вживання засобів індивідуального захисту.

Зниження рівня шуму в приміщеннях з комп’ютерною технікою можна здійснити таким чином:

9. використання блоків живлення ПК з вентиляторами на гумових підвісках;

10. використанням ПК, в яких термодавачі вмонтовані в блоці живлення та в критичних точках материнської плати (процесор, мікросхеми чіпсету), які дозволяють програмним шляхом регулювати як моменти ввімкнення вентиляторів, так і їх швидкість обертання;

11. переведення жорсткого диска в режим сплячки (Standby), якщо

комп’ютер не працює на протязі визначеного часу. Цей час встановлюється в опціях керування напругою в операційних системах Windows 9x та Windows 2000. Якщо в режимі Standby немає необхідності, його можна вимкнути в BIOS материнської плати;

12. використанням ПК, в яких вентилятор на процесорі встановлено виробником (BOX-процесор);

13. застосуванням материнських плат формату ATX та ATX-корпусів, що дозволяє регулювати автономну швидкість та моменти часу відмикання вентилятора блока живлення від електромережі;

14. використанням 24-32-х швидкісних CD-ROM для застосувань, які створюють менше щуму, аніж швидкісні 48-50-х CD-ROM, або ж застосувати привід з одночасним зчитуванням декількох доріжок CD;

15. заміною матричних голчатих принтерів - струменевими і лазерними принтерами, які забезпечують при роботі значно менший рівень звукового тиску;

16. застосування принтерів колективного користування, розташованих на значній відстані від більшості робочих місць користувачів ПК;

17. зміною напряму випромінювання шуму в протилежну сторону від робочого місця;

18. зменшення шуму на шляху його розповсюдження установкою звукоізолюючого відгородження у вигляді стін, перетинок, кабін;

19. акустичною обробкою приміщень - зменшення енергії відбитих звукових хвиль шляхом збільшення площі звукопоглинання (розміщення на внутрішніх поверхнях приміщення облицювань, що поглинають звук, розміщенням в приміщеннях штучних поглиначів звуку). Облицювання внутрішніх поверхонь приміщення звукопоглинаючими матеріалами забезпечує значне зниження шуму. Крім облицювання приміщень використовують об'ємні звукопоглинаючі тіла різноманітної форми, які вільно розвішують в об'ємі приміщення. Звукопоглинання дає найбільший акустичний ефект в зоні відбитого звуку. В точках приміщення, де поширюється здебільшого прямий

звук, ефективність звукопоглинання суттєво знижується. Звукопоглинаюче облицювання розміщується на стелі та верхніх частинах стін. Максимальне звукопоглинання досягається при облицюванні не менше 60% загальної площі огороджуючих поверхонь приміщення.

8.4 Розрахункова частина

Розрахунок звукопоглинаючого облицювання поверхонь приміщення з комп'ютерною технікою. Для облицювання поверхонь використано звукопоглинаючий матеріал - плити ПА/С.

В приміщенні працюють програмісти. Шум в приміщенні створюють 3 джерела: 2 ПК та принтер.

Об'єм приміщення (V) становить - 72 $м^3$;

Для розрахунку звукопоглинаючого облицювання необхідно знати акустичні характеристики приміщення:

В - постійна приміщення, $м^2$;

А - еквівалентна площа звукопоглинання, $м^2$;

α- середній коефіцієнт звукопоглинання.

Постійна приміщення (В) акустично необробленого приміщення визначається:

$$B = B_n \cdot \mu, \qquad (8.1)$$

де B_n - постійна приміщення на середньогеометричній частоті 1000 Гц, $м^2$;

μ - частотний множник.

Таблиця 8.2 - Значення постійної приміщення

Приміщення	$B_{п,\ м^2}$
З невеликою кількістю людей (цехи заводів)	V/20
З жорсткими меблями і великою кількістю людей або з невеликою кількістю людей та м'якими меблями (лабораторії, кабінети)	V/10
З великою кількістю людей і м'якими меблями (кімнати управління, зали конструкторський бюро, учбові аудиторії)	V/6
Приміщення зі звукопоглинаючим облицюванням стелі і частини стін	V/1,5

Для даного приміщення:

$$B_n = V / 6, \qquad (8.2)$$

де V - об'єм приміщення.

Отже B_n = 72/6 = 12.

Таблиця 8.3 - Значення частотного множника μ для приміщень різних об'ємів

Об'єм приміщення, $м^3$	Частотний множник μ на середньогеометричнихчастотах октавних смуг, Гц							
	63	125	250	500	1000	2000	4000	S000
<200	0,8	0,75	**o,7**	0,8	1	1,4	1,8	2.5
200...1000	0,65	0,62	0,64	0,75	1	1,5	2,4	4,2
>1000	0,5	0,5	0,55	0,7	1	1,6	**3**	6

Значення частотного множника μ залежать від об'єму приміщення. У нашому випадку об'єм приміщення V = 72 $м^3$, отже 120 $м^3$ < 200 $м^3$, тому значення частотного множника взяті для об'єму приміщення < 200 $м^3$.

За знайденими значеннями постійних приміщення для кожної октавної смуги обчислюється еквівалентна площа звукопоглинання:

$$A = B/(B/S + 1), \qquad (8.3)$$

де S - загальна площа огороджуючих поверхонь приміщення, $м^2$.

Отже, $S = 2 \cdot (a \cdot h + b \cdot h + a \cdot b) = 2 \cdot (6 \cdot 3 + 4 \cdot 3 + 6 \cdot 4) = 108 м^2$.

Середній коефіцієнт звукопоглинання α в приміщенні до його акустичної обробки обчислюється за формулою:

$$\alpha = B/(B + S), \qquad (8.4)$$

Зона відбитого звуку визначається величиною граничного радіусу $r_{гр}$, тобто, відстанню від розповсюджувача шуму, на якій рівень звукового тиску відбитого звуку дорівнює рівню звукового тиску прямого звуку, що розповсюджується.

$$r_{гр} = 0{,}2 \cdot \sqrt{B_{п2} / n}\,, \qquad (8.5)$$

де $B_{п2}$ - постійна приміщення на середньогеометричній частоті 8000 Гц.

n - кількість однакових розповсюджувачів шуму;

$$B_{п2} = B_п \cdot \mu_{8000} = 30 м^2 \qquad (8.6)$$

$$r_{гр} = 0{,}2 \cdot \sqrt{30/2} = 0{,}77 м\,.$$

Максимальне зниження рівня звукового тиску ΔL (дБ), в кожній октавній смузі при застосуванні звукопоглинаючого облицювання в розрахунковій точці, що розміщена в зоні відбитого звуку обчислюється за формулою:

$$\Delta L = 10 \lg(B'/B)\,, \qquad (8.7)$$

де B' - постійна приміщення після облаштування в ньому звукопоглинаючих конструкцій, $м^2$.

$$B' = (A_1 + \Delta A)/(1 - \alpha_1)\,. \qquad (8.8)$$

$A_1 = \alpha \cdot (S - S_{обл})$ - еквівалентна площа звукопоглинання поверхнями не зайнятими звукопоглинаючим облицюванням, $м^2$;

α_1 - середній коефіцієнт звукопоглинання акустично обробленого приміщення;

$$\alpha_1 = (A_1 + \Delta A)/S\,, \qquad (8.9)$$

ΔA - величина сумарного додаткового поглинання, що вноситься конструкцією звукопоглинаючого облицювання або штучними поглиначами.

$$\Delta A = \alpha_{обл} \cdot S_{обл} + A_{шт} \cdot n_{шт}\,, \qquad (8.10)$$

$\alpha_{обл}$ - реверберацiйний коефiцiєнт звукопоглинаючої конструкції облицювання;

$S_{обл}$ - площа облицьованих поверхонь, $м^2$ (стіни та стеля);

Таблиця 8.4 - Зниження рівня шуму в приміщеннях по частотах

Середньогеометрична частота, Гц	3	25	50	00	000	000	000	000
Рівень звукового тиску в приміщенні, L, дБ	2	6	2	5	0	8	8	2
Допустимий рівень звукового піску в приміщенні, де працюють програмісти, дБ.	1	1	4	9	5	2	0	8
Потрібне зниження рівня шуму, дБ				6	5	6	8	4

Висновки до розділу

Так як оператори ЕОМ, оператори з підготовки даних, програмісти піддаються впливу фізично небезпечних і шкідливих виробничих факторів, то в даному розділі були розглянуті питання пов’язані чинники та основні проблеми та загрози користувачів ПЕОМ, які піддаються небезпечному чиннику, а саме шуму. Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація. Все це знижує працездатність людини і її продуктивність, якість і безпеку праці.

Були наведені приклади значно зменшити вплив від виробничого шуму працівників, які працюють напружено з ПЕОМ в офісах і на виробничому підприємстві та проведено розрахунок звукопоглинаючого облицювання поверхонь приміщення з комп’ютерною технікою.

9 ОСНОВНІ ВИСНОВКИ

В даній магістерській роботі розроблено та дослідженню метод побудови рекомендаційних систем. Проаналізовано реалізацію цих підходів на прикладі відомих інтернет-сервісів з каталогом товарів.

В результаті виконання роботи було проаналізовано основні алгоритми та базові підходи для прогнозування рекомендацій. Результати аналізу представлені на таблиці. Було описано особливості, переваги та недоліки кожного з них та виконано порівняльний аналіз існуючих підходів. Також було зроблено огляд допоміжних засобів для кореляції результатів, отриманих після застосування основних алгоритмів.

Алгоритми колаборативної фільтрації дозволяють швидко створювати та інтегрувати рекомендаційні системи, адже вони засновані лише на пошуку схожих користувачів. Тому колаборативна фільтрація найпоширеніша в інтернет-сервісах, що використовують рекомендаційні системи. Такі системи зазвичай підходять для невеликих блогів, інформаційних порталів тощо, бо в цих випадках характерні проблеми для рекомендаційних систем не матимуть широкого впливу.

Алгоритми контентної фільтрації прогнозують рекомендації завдяки створенню моделей користувачів та товарів. Рекомендаційні системи засновані на цьому підході потребують значно більше ресурсів для створення, ніж ті, що використовують колаборивну фільтрацію. Але вони не мають проблем холодного старту, унікальних користувачів, шахрайства тощо. Такі підходи зазвичай використовуються для створення інтернет-магазинів, де рекомендації товарів важливо підбирати під конкретного користувача.

Гібридні алгоритми мають на меті знизити вплив звичних для колаборативної та контентної фільтрації проблем. Але такі системи мають використовуватися виправдано, оскільки вони громіздкі, на їх побудову

необхідно витратити велику кількість часу, коштів та людских ресурсів. Вони зазвичай використовуються в крупних інтернет-магазинах та соціальних мережах.

Були розглянуті шляхи до подолання типових недоліків базових алгоритмів. Для вирішення недоліків колаборативної фільтрації використовується алгоритм SVD. Адже з ростом кількості користувачів та їх оцінок в будь-якому інтернет-сервісі буде зростати розмірність матриці оцінок. Цей алгоритм дозволяє зменшити розмірність цієї матриці з невеликою втратою точності. Щоб пришвидшити роботу алогоритмів контентної фільтрації зазвичай використовують алгоритм TF-IDF, який дозволяє будувати рейтингові списки документів великого корпусу документів.

Також було виконано порівняльний аналіз існуючих рішень імлементації рекомендаційних систем у відомі інтернет-компанії, які збільшили свої прибутки саме через інтеграцію рекомендаційних систем до каталогу своїх продуктів та сервісів. Було проаналізовано рекомендаційні системи та модулі широковідомих компаній Amazon та eBay. Amazon використовує більшу кількість рекомендаційних сервісів на базі гібридної фільтрації, що дає їм більше можливостей для надання релевантніших рекомендацій. Як можна побачити з таблиці 2.1, Amazon під кожну задачу використовує конкретний алгоритм, тому важливо проводити дослідження з вибору алгоритму для рекомендаційної системи на її налаштування під конкретну задачу, оскільки в майбутньому це зможе значно покращити результати прогнозуванн рекомендацій.

В якості демонстрації проведеного аналізу базових алгоритмів було реалізовано веб-додаток із інтегрованою рекомендаційною системою на базі колаборативної фільтрації. Платформою для створення інтернет-сервісу було обрано фреймворк Ruby on Rails, який дозволяє ефективно створювати проекти високої складності з використанням різних модулів (внутрішніх та зовнішніх).

Rails – сучасний стандарт майже усіх веб-додатків для бізнесу, який дозволяє швидко створювати прототипи веб-додатків. Вбудована рекомендаційна система на основі колаборативної фільтрації має на меті збільшити залученість користувачів до сервісу та збільшити його конверсію. Отримані результати від створеної рекомендаційної системи виявилися задовільними: прогнозовані рекомендації виявилися релевантними, а алгоритм пошуку схожих користувачів досить точно виявляв групу схожих користувачів.

Так як в даному проекті було реалізовано веб-додаток з рекомендаційною системою на базі колаборативної фільтрації, в подальшому можна зробити більш глибоке та детальне дослідження більш ефективних способів реалізації, а саме – гібридних алгоритмів. Розвиток інтернет-сервісів рухається в сторону логічного розділення модулів продукту, тому можна розробити незалежний сервіс для рекомендацій будь-якого контенту із використанням відкритого API.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 2080 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 2,9 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Meleshko E. Дослідження методів побудови рекомендаційних систем в мережі Інтернет / E. Meleshko, S. Semenov, V. Khokh // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2018. – Т. 1 (47). – С. 131-1362.

2. Лобур М. Побудова асоціативних правил для прогнозування рекомендацій в колаборативних рекомендаційних системах / М. Лобур, Ю. Стех, М. Шварц // Збірник наукових праць УАД. – Львів, 2017. − № 2 (32). − С. 82−83.

3. Nathan Marz. Big Data: Principles and best practices of scalable realtime data systems / Nathan Marz, James Warren. – Manning Publications, 2015. – 328 p.

4. Schedl M, Gómez E, Urbano J. Music information retrieval: recent developments and applications. Found Trends Inform Ret. (2014) 8:127–261. doi:

5. Downie JS. Music information retrieval. Ann Rev Inform Sci Techn. (2003) 37:295–340. doi: 10.1002/aris.1440370108

6. Dorfer M, Henkel F, Widmer G. Learning to listen, read, and follow: score following as a reinforcement learning game. In: Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018), Paris (2018). p. 784–91.

7. Chou PW, Lin FN, Chang KN, Chen HY. A simple score following system for music ensembles using chroma and dynamic time warping. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (ICMR 2018), Yokohama: ACM (2018). p. 529–32. doi: 10.1145/3206025.3206090

8. Goto M, Dannenberg RB. Music interfaces based on automatic music signal analysis: new ways to create and listen to music. IEEE Signal Proc Mag. (2019) 36:74–81. doi: 10.1109/MSP.2018.2874360

9. Schedl M. Intelligent user interfaces for social music discovery and exploration of large-scale music repositories. In: Proceedings of the 22nd ACM

International Conference on Intelligent User Interfaces (IUI 2017): Workshop on Theory-nformed User Modeling for Tailoring and Personalizing Interfaces (HUMANIZE 2017). Limassol: ACM (2017). p. 7–11. doi:

10. Oramas S, Nieto O, Barbieri F, Serra X. Multi-label music genre classification from audio, text and images using deep features. In: Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR), Suzhou (2017). p. 23–30.

11. Mayer R, Rauber A. Music genre classification by ensembles of audio and lyrics features. In: Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011). Miami, FL (2011). p. 675–80.

12. Sturm BL. Classification accuracy is not enough: on the evaluation of music genre recognition systems. J Intell Inform Syst. (2013) 41:371–406. doi: 10.1007/s10844-013-0250-y

13. Yang YH, Chen HH. Machine recognition of music emotion: a review. Trans Intell Syst Techn. (2013) 3:40. doi: 10.1145/2168752.2168754

14. Huq A, Bello JP, Rowe R. Automated music emotion recognition: a systematic evaluation. J New Music Res. (2010–11) 39:227–44. doi: 10.1080/09298215.2010.513733

15. Knees P, Schedl M. Music similarity and retrieval — an introduction to audio-and web-based strategies. Berlin; Heidelberg: Springer (2016).

16. Karydis I, Lida Kermanidis K, Sioutas S, Iliadis L. Comparing content and context based similarity for musical data. Neurocomputing. (2013) 107:69–76. doi: 10.1016/j.neucom.2012.05.033

17.Schedl M, Knees P, McFee B, Bogdanov D, Kaminskas M. Music recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB, editors. Recommender Systems Handbook, 2nd ed. Boston, MA: Springer (2015). p. 453–92.

18.van den Oord A, Dieleman S, Schrauwen B. Deep content-based music recommendation. In: Burges C, Bottou L, Welling M, Ghahramani Z, Weinberger K, editors. Advances in Neural Information Processing Systems 26 (NIPS). Lake Tahoe,

NV: Curran Associates, Inc. (2013). p. 2643–51.

19. Bertin-Mahieux T, Ellis DPW, Whitman B, Lamere P. The million song dataset. In: Proceedings of the 12th International Society for Music Information Retrieval Conference. Miami, FL (2011). p. 591–6.

20. Xiaoyuan Su, Taghi M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in Artificial Intelligence archive, 2009.

21. Recommender Systems - The Textbook | Charu C. Aggarwal | Springer Springer. 2016. ISBN 9783319296579.

22."virtual (C# Reference)". docs.microsoft.com.

23. "Installation guidance for SQL Server on Linux". December 21, 2017. Retrieved February 1, 2018.

24. "SQL Server 2008 R2 Express Database Size Limit Increased to 10GB"

25. "What's up with SQL Server 2008 Express editions"

26. Kalen Delaney (2007). Inside Microsoft SQL Server 2005: The Storage Engine

27. "Pages and Extents" (http://msdn.microsoft.com/en-us/library/ms190969.aspx)

28."Table and Index Organization" (https://docs.microsoft.com/en-us/previous versions/sql/sql-server

29."Buffer Management" (https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-

30."Transact-SQL Reference" (https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2005/ms189826(v=sql.90)?redirectedfrom=MSDN

31. J. Bobadilla, F. Ortega, A. Hernando, and A. Guti´errez, “Recommender systems survey,” Knowledge-Based Systems, vol. 46, 2013 – P. 109-132.

32. Daniar Asanov (n.d.) Algorithms and Methods in Recommender Systems, Berlin, Germany: Berlin Institute of Technology – 2011 – P. 1-7

33. Jones, M. T. (2013, December 12). Recommender systems.Introduction to approaches and algorithms. Retrieved November 25, 2017, from

https://www.ibm.com/developerworks/library/os-recommender1/

34. M. Pazzani A Framework for Collaborative, Content-Based, and Demographic Filtering // Artificial Intelligence Rev. – 1999. – P. 393-408.

35. M. Balabanovic Fab: Content-Based, Collaborative Recommendation / M. Balabanovic, Y. Shoham // Comm. ACM. – 1997. – Vol. 40, No3. – P.66-72.

36. Ruby [Електронний ресурс] // Режим доступу: https://www.ruby-lang.org/en/

37. Ruby On Rails [Електронний ресурс] // Режим доступу: http://rubyonrails.org

38. Когулько О.С. Використання методів колаборативної фільтрації для роботи рекомендаційної системи / Міжнародна науково-практична конференція «Математичне та імітаційне моделювання систем» (МОДС-2018) – м. Київ., 25-29 червня 2018 р. – С. 83-86.

39. Daniar Asanov (n.d.) Algorithms and Methods in Recommender Systems, Berlin, Germany: Berlin Institute of Technology – 2011 – P. 1-7

40. Chalyi S. Доповнення вхідних даних рекомендаційної системи в ситуації циклічного холодного старту з використанням темпоральних обмежень типу «next» / S. Chalyi, V. Leshchynskyi, I. Leshchynska // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – Т. 4 (56). – С. 105-109. – doi:https://doi.org/10.26906/SUNZ.2019.4.105.

41. Чалий С.Ф., Лещинський В.О., Лещинська І.О. Моделювання контексту в рекомендаційних системах. Науковий журнал «Проблеми інформаційних технологій», 2018, №. 1(023). С. 21-26.

42. Савчук Т.О., Застосування кластерного аналізу для колаборативної фільтрації / Т.О. Савчук, А.В.Сакалюк // Вісник Хмельницького національного університету. –2011 – №1– С. 186-192

43. Matrix factorization and neighbor based algorithms for the netflix prize problem / G. Tak´acs, I. Pil´aszy, B. N´emeth, D. Tikk // Proceedings of the 2008 ACM conference on Recommender systems / ACM. — 2008. — P. 267–274

44. Лобур М.В., Шварц М.Є., Стех Ю.В. Моделі і методи прогнозування рекомендацій для колаборативних рекомендаційних систем // Вісник Національного університету «Львівська політехніка»: Інформаційні системи та мережі. Львів, 2018. № 901. С. 68-75.

45. Stekh Y., Lobur M., Shvarts M. Some methods for improving the accuracy of prediction recommendations // Вісник Національного університету «Львівська політехніка»: Комп'ютерні системи проектування. Теорія і практика. Львів, 2017. № 882. С. 46-49.

46. Смірнов О.А. Програмування комп'ютерних мереж. Основи HTML, CSS, JAVA-SCRIPT. Методичні вказівки. Кіровоград 2007–107 с.

47. https://dou.ua/ - Співтовариство програмістів.

48. Пасічник О. Г., Пасічник О. В., Стеценко І. В. Основи веб-дизайну: Навч. посіб. -К.: Вид. група BHV. 2011р. -336 с.

49. Люк Веллинг, Лора Томсон. Розробка Web -додатків за допомогою PHP і MySQL : Вільямс, 2012р., - 880 с.

50. https://idg.net.ua/blog/uchebnik-css - Підручник CSS.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку програмного забезпечення системи захисту в інформаційній системі.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 32-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою роботи є широкий аналіз існуючих підходів до створення рекомендаційних систем, та реалізація додатку із вбудованою системою рекомендацій та підвищення ефективності роботи рекомендаційної системи

4 Джерела розробки

Джерелом цієї магістерської дипломної роботи є розробки, які ведуться спільнотою розробників фреймворків і стосовна до теми технічна література.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– створення і налаштування серверних додатків;

– веб-сайт створений на мові програмування Ruby;

– простий, інтуїтивно зрозумілий інтерфейс з користувачем.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення повинно дозволити достатньо легко зберігати, редагувати, переглядати дані у браузері.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10/11 та Linux.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Ruby з використанням фреймворку ruby on rails

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці та техніки безпеки в магістерській роботі повинен бути розглянутий аналіз умов праці програміста та методи боротьби від шуму.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 118 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі бакалаврської дипломної роботи. Постановка задачі на виконання бакалаврської дипломної роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень бакалаврської дипломної роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

11.1 Подання магістерської дипломної роботи на попередній захист 10.12.2023 р.

1.2 Подання магістерської роботи на захист 08.01.2024 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Пархоменко Ю.М.

Дослідження та програмна реалізація технології побудови рекомендаційної системи для просування товарів і послуг компанії в мережі Інтернет

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 16

Літера: РП

Кропивницький – 2023 року

Лістинг файлу app/controllers/application_controller.rb

```
class ApplicationController < ActionController::Base # Prevent CSRF
attacks by raising an exception.
# For APIs, you may want to use :null_session instead.
protect_from_forgery with: :exception
before_action :authenticate_user! end
```

Лістинг файлу app/controllers/posts_controller.rb

```
class PostsController < ApplicationController RECOMMENDABLE_METHODS
= %w(like unlike dislike undislike).freeze

before_action :set_post, only: %w(show edit update
destroy).concat(RECOMMENDABLE_METHODS)
before_action :fetch_sections, only: %w(index manage show edit update new
create)
before_action :post_owner, only: %w(edit update destroy) respond_to :json,
only: RECOMMENDABLE_METHODS

# GET /posts def index
@q = Post.ransack(params[:q])
@q.sorts = 'section_id asc' if @q.sorts.empty? @posts =
@q.result(distinct: true)
end


# GEt /posts/manage def manage
if current_user.admin?


@posts = Post.all else
@posts = current_user.posts end
end


# GET /posts/1 def show
end


# GET /posts/new def new
@post = Post.new end
```

```
# GET /posts/1/edit def edit
end


# POST /posts def create
@post = Post.new(post_params)


if @post.save
redirect_to @post, notice: 'Post was successfully created.' else
render :new end
end


# PATCH/PUT /posts/1 def update

if @post.update(post_params)
redirect_to posts_path, notice: 'Post was successfully updated.' else


render :edit end
end


# DELETE /posts/1 def destroy @post.destroy
redirect_to posts_url, notice: 'Post was successfully destroyed.' end

RECOMMENDABLE_METHODS.each do |method| define_method(method) do
@post = Post.find(params[:id])


if current_user.send(method, @post) head :ok
else
head :unprocessable_entity end
end end

private


# Use callbacks to share common setup or constraints between actions. def
set_post
@post = Post.find(params[:id]) end
```

```
def fetch_sections @sections = Section.all end

# Only allow a trusted parameter 'white list' through. def post_params
params.require(:post).permit(:title, :text, :annotation, :image_src,
:section_id, :user_id, :author, tag_list: [])


end


def post_owner
unless @post.user_id == current_user.id || current_user.admin?
flash[:notice] = 'Access denied as you are not owner of this job'
redirect_to posts_path
end end
end
```

Лістинг файлу app/controllers/sections_controller.rb

```
class SectionsController < ApplicationController
before_action :set_section, only: %w(update destroy)

def index
@section = Section.new
@sections = Section.all.order(:id) end

def create
@section = Section.new(section_params) if @section.save
redirect_to sections_path, notice: 'Section was successfully created.'
else
redirect_to sections_path, notice: 'Some errors occurred.' end
end


def update
if @section.update(section_params)
redirect_to sections_path, notice: 'Section was successfully updated.'
else
redirect_to sections_path, notice: 'Some errors occurred.' end
end


# DELETE /sections/1
```

```
def destroy @section.destroy
redirect_to sections_path, notice: 'Section was successfully destroyed.'
end

private


def set_section
@section = Section.find(params[:id]) end

def section_params params.require(:section).permit(:name) end
end
```

Лістинг файлу app/controllers/tags_controller.rb

```
class TagsController < ApplicationController def index
@tags = ActsAsTaggableOn::Tag.all.order(:id) end
end
```

Моделі

Лістинг файлу app/models/post.rb

```
class Post < ActiveRecord::Base acts_as_taggable_on :tags belongs_to :user
belongs_to :section


validates :title, presence: true, length: 1..255 end
```

Лістинг файлу app/models/section.rb

```
class Section < ActiveRecord::Base has_many :posts, dependent: :destroy
```

Лістинг файлу app/controllers/application_controller.rb

```
class ApplicationController < ActionController::Base # Prevent CSRF
attacks by raising an exception.
# For APIs, you may want to use :null_session instead.
protect_from_forgery with: :exception
before_action :authenticate_user! end
```

Лістинг файлу app/controllers/posts_controller.rb

```
class PostsController < ApplicationController RECOMMENDABLE_METHODS
= %w(like unlike dislike undislike).freeze
```

```
before_action :set_post, only: %w(show edit update
destroy).concat(RECOMMENDABLE_METHODS)
before_action :fetch_sections, only: %w(index manage show edit update new
create)
before_action :post_owner, only: %w(edit update destroy) respond_to :json,
only: RECOMMENDABLE_METHODS

# GET /posts def index
@q = Post.ransack(params[:q])
@q.sorts = 'section_id asc' if @q.sorts.empty? @posts =
@q.result(distinct: true)
end


# GEt /posts/manage def manage
if current_user.admin?


@posts = Post.all else
@posts = current_user.posts end
end


# GET /posts/1 def show
end


# GET /posts/new def new
@post = Post.new end

# GET /posts/1/edit def edit
end


# POST /posts def create
@post = Post.new(post_params)


if @post.save
redirect_to @post, notice: 'Post was successfully created.' else
render :new end
end
```

```
# PATCH/PUT /posts/1 def update

if @post.update(post_params)
redirect_to posts_path, notice: 'Post was successfully updated.' else


render :edit end
end


# DELETE /posts/1 def destroy @post.destroy
redirect_to posts_url, notice: 'Post was successfully destroyed.' end

RECOMMENDABLE_METHODS.each do |method| define_method(method) do
@post = Post.find(params[:id])


if current_user.send(method, @post) head :ok
else
head :unprocessable_entity end
end end

private


# Use callbacks to share common setup or constraints between actions. def
set_post
@post = Post.find(params[:id]) end

def fetch_sections @sections = Section.all end

# Only allow a trusted parameter 'white list' through. def post_params
params.require(:post).permit(:title, :text, :annotation, :image_src,
:section_id, :user_id, :author, tag_list: [])


end


def post_owner
```

```
unless @post.user_id == current_user.id || current_user.admin?
flash[:notice] = 'Access denied as you are not owner of this job'
redirect_to posts_path
end end
end
```

Лістинг файлу app/controllers/sections_controller.rb

```
class SectionsController < ApplicationController
before_action :set_section, only: %w(update destroy)

def index
@section = Section.new
@sections = Section.all.order(:id) end

def create
@section = Section.new(section_params) if @section.save
redirect_to sections_path, notice: 'Section was successfully created.'
else
redirect_to sections_path, notice: 'Some errors occurred.' end
end


def update
if @section.update(section_params)
redirect_to sections_path, notice: 'Section was successfully updated.'
else
redirect_to sections_path, notice: 'Some errors occurred.' end
end


# DELETE /sections/1


def destroy @section.destroy
redirect_to sections_path, notice: 'Section was successfully destroyed.'
end

private


def set_section
@section = Section.find(params[:id]) end
```

```
def section_params params.require(:section).permit(:name) end
end
```

Лістинг файлу app/controllers/tags_controller.rb

```
class TagsController < ApplicationController def index
@tags = ActsAsTaggableOn::Tag.all.order(:id) end
end
```

```
validates :name, presence: true, length: 1..255 end
```

Лістинг файлу app/models/user.rb

```
class User < ActiveRecord::Base
# Include default devise modules. Others available are:
# :confirmable, :lockable, :timeoutable and :omniauthable
devise :database_authenticatable, :registerable,
:rememberable, :trackable, :validatable


recommends :posts


has_many :posts, dependent: :destroy end
```

Представлення

Лістинг файлу app/views/layouts/application.html.slim

```
doctype html html
head title
| Mazurik Blog
= stylesheet_link_tag 'application', media: 'all'
= javascript_include_tag 'application'
= csrf_meta_tags body
nav.navbar.navbar-default
.container
.navbar-header
a.navbar-brand[href="/"]
| Mazurik Blog
-     if current_user ul.navbar-nav.nav
= active_link_to 'Articles', posts_path, 'posts'
= active_link_to 'Sections', sections_path, 'sections'
```

```
-     if current_user.present? && current_page?(controller: 'posts',
action: 'index')
= search_form_for @q, html: { class: 'navbar-form navbar-left' } do |f|
.form-group
= f.search_field :title_cont, placeholder: 'Search posts...', class:
'form- control'
= f.submit 'Search', class: 'btn btn-default' ul.navbar-nav.nav.navbar-
right
-     unless current_user li
= link_to 'Sign up', new_user_registration_path li
= link_to 'Sign in', new_user_session_path
-     else li.dropdown
a.dropdown-toggle[href="#" data-toggle="dropdown" role="button" aria-
haspopup="true" aria-expanded="false"]
= current_user.email
|  
span.glyphicon.glyphicon-option-vertical ul.dropdown-menu
li
= link_to 'My posts', manage_path li.divider
li
= link_to 'Sign Out', destroy_user_session_path, method: :delete


.container
-     if notice p.alert.alert-warning
= notice
= yield footer
.text-center
```

Лістинг файлу app/views/posts/_form.html.slim

```
= simple_form_for(@post) do |f|


.form-group label.control-label
| Author:  
= current_user.email
= f.input :title, placeholder: 'Title...', hint: false
= f.input :annotation, placeholder: 'Annotation...'
= f.input :section_id, collection: @sections, selected: @post.section_id
= f.input :user_id, as: :hidden, input_html: { value: current_user.id }
.form-group
label.control-label for='post[tag_list][]'
```

```
| Tags:
select.form-control.js-select2-select multiple="multiple"
name="post[tag_list][]"
-     ActsAsTaggableOn::Tag.all.pluck(:name).each do |tag|
-     if tag.in?(@post.tag_list)
option value="#{tag}" selected='selected' #{tag}
-     else
option value="#{tag}" #{tag}
= f.input :image_src, placeholder: 'Image URL...'
= f.input :text, placeholder: 'Blog text type here...', input_html: {rows:
'30'}
= f.button :submit
```

Лістинг файлу app/views/posts/index.html.slim

```
- @posts.each do |post| h3
= post.section.name
|  
| &#8594;
|  
= link_to post do
= post.title
|   p
-     post.tags.each do |tag| span.label.label-default
= tag.name



-     if post.image_src.present?
img src="#{post.image_src}" alt="Post image" width="100%" p
= post.annotation
= link_to 'Learn more &#8594;'.html_safe, post, class: 'btn btn-default'
hr
```

Лістинг файлу app/views/posts/edit.html.slim

```
h1
| Edit Post
= render 'form'
```

Лістинг файлу app/views/posts/manage.html.slim

```
h1
| Posts  
= link_to 'New Post', new_post_path, class: 'btn btn-sm btn-default'
- unless @posts.empty? table.table
thead tr
```

```
th Title
th Annotation th Section
th tbody
- @posts.each do |post| tr
td.min-200
= link_to post.title, post td
= post.annotation td.min-100
= post.section.name td.min-130
.text-left
- if (current_user.id == post.user_id || current_user.admin?)
= link_to 'Edit', edit_post_path(post), class: 'btn btn-sm btn-warning'


|  
= link_to 'Destroy', post, method: :delete, data: { confirm: 'Are you
sure?' }, class: 'btn btn-sm btn-danger'
-     else p
| No posts yet.
```

Лістинг файлу app/views/posts/new.html.slim

```
h1
| New Post
= render 'form'
```

Лістинг файлу app/views/posts/show.html.slim

```
h2
= @post.section.name
|  
| &#8594;
|  
= @post.title
|  
- if (current_user.id == @post.user_id)
= link_to 'Edit', edit_post_path(@post), class: 'btn btn-sm btn-default'


-     if @post.image_src.present?
= image_tag @post.image_src, alt: 'Post image', width: '100%'


p
| By  strong
= @post.author
```

```
p
strong
| Created at:  
= str_date @post.created_at


h4


- @post.tags.each do |tag| span.label.label-default
= tag.name


p
= @post.text.html_safe


.like-form-section
- unless current_user.likes?(@post)
= form_for @post, url: like_post_path, remote: true, method: :post, html:
{ class: 'form-inline like-form' } do |f|
= f.button class: 'btn btn-default' do span.glyphicon.glyphicon-thumbs-up
|  Like
|  
- else
= form_for @post, url: unlike_post_path, remote: true, method: :delete,
html:
{class: 'form-inline like-form'} do |f| button.btn.btn-success
type='submit' span.glyphicon.glyphicon-thumbs-up
|  Like
|  
- unless current_user.dislikes?(@post)
= form_for @post, url: dislike_post_path, remote: true, method: :post,
html:
{class: 'form-inline like-form'} do |f| button.btn.btn-default
type='submit' span.glyphicon.glyphicon-thumbs-down
|  Dislike
|  
- else
= form_for @post, url: undislike_post_path, remote: true, method: :delete,
html: {class: 'form-inline like-form'} do |f|
```

```
button.btn.btn-danger type='submit' span.glyphicon.glyphicon-thumbs-down
|  Dislike


- if current_user.recommended_posts.size > 0 &&
!(current_user.recommended_posts.include?(@post) &&
current_user.recommended_posts.size == 1)
.also-like
.panel.panel-default
.panel-body
h3 You may also like:
-     current_user.recommended_posts(4,0).each do |post|
-     if @post != post
.media
-     if @post.image_src
.media-left
= link_to post do
img.media-object width="100" src="#{post.image_src}" alt='Post image'
.media-body
= link_to post do
h4.media-heading #{post.title} p #{post.annotation}
```

Лістинг файлу app/views/sections/index.html.slim

```
.sections
.main-section h3.title
| Technical sciences
= image_tag 'computer_sciences.png'
.sub-sections.row
.section.col-md-4
= image_tag 'social_media.jpg'
.section.col-md-4
= image_tag 'economics.jpg'
.section.col-md-4
= image_tag 'legal_sciences.jpg'


table.table.table-hover thead
tr th #


th Name
-     if current_user.admin? th Options
tbody
```

```
-     if current_user.admin? tr.info
td td
= simple_form_for @section, method: :post, html: { class: 'form-inline' }
do
|f|
.form-group
= f.input :name, required: true, placeholder: 'Section name...', label:
false
= f.button :submit, 'Add Section', class: 'btn-sm'
-     @sections.each do |section| tr
td #{section.id}
td class='hidden js-edit-form' data-id="#{section.id}"
= simple_form_for section, method: :put, html: { class: 'form-inline' } do
|f|
.form-group
= f.input :name, placeholder: 'Section name...', required: true, label:
false
= f.button :submit, 'Update', class: 'btn btn-sm btn-primary'
.form-group
button class='btn btn-default js-cancel btn-sm' data-id="#{section.id}"
Cancel td class='js-info-name' data-id="#{section.id}" #{section.name}
-     if current_user.admin? td.text-left
button class='js-edit btn btn-sm btn-warning' data-id="#{section.id}"
| Edit
|  
= link_to 'Destroy', section, method: :delete, data: { confirm: 'Are you
sure?'
}, class: 'btn btn-sm btn-danger'
```

Лістинг файлу app/views/devise/sessions/new.html.slim

```
.row
.col-md-6.col-md-offset-3 h2
| Sign in


= simple_form_for(resource, as: resource_name, url: new_user_session_path)
do
|f|
= f.input :email, required: false, autofocus: true
= f.input :password, required: false
= f.input :remember_me, as: :boolean if devise_mapping.rememberable?
.text-center
= f.button :submit, 'Sign in', class: 'btn-success btn-lg'
```

```
.text-center
= render 'devise/shared/links'
```

Лістинг файлу app/views/devise/registrations/new.html.slim

```
.row
.col-md-6.col-md-offset-3 h2
| Sign up
= simple_form_for(resource, as: resource_name, url:
registration_path(resource_name)) do |f|
= f.input :email, required: true, autofocus: true
= f.input :password, required: true
= f.input :password_confirmation, required: true
.text-center
= f.button :submit, 'Sign up', class: 'btn-success btn-lg'
.text-center
= render 'devise/shared/links';
```