НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.4:001.89

«ПОГОДЖЕНО»
Декан факультету інформаційних технологій
Болбот І. М., д.п.н., професор
_____
____ _____ 2024 р.

«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»
Завідувач кафедри комп'ютерних наук
Голуб Б.Л., к.т.н., доцент
_____
____ _____ 2024 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Експертна система рекомендацій у системі розповсюдження авторських розробок

Спеціальність 121 «Інжереія програмного забезпечення»
(код і назва)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітня-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

професор, д.т.н. _____ Семко В. В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

доцент к.т.н. _____ Голуб Б. Л.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Яковлєв О. І.
(підпис) (ПІБ студента)

КИЇВ-2024

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____інформаційних технологій_____

ЗАТВЕРДЖУЮ
Завідувач кафедри___комп'ютерних наук__
_____
_____доцент к.т.н._____ _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____20_____року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

_____Яковлєва Олексія Ігоровича_____
(прізвище, ім'я, по батькові)

Спеціальність ___121 – «Програмне Забезпечення Інформаційних Систем»_____
(код і назва)

Освітня програма _____
(назва)

Орієнтація освітньої програми_____
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи___Експертна система рекомендацій у системі розповсюдження авторських розробок _____
_____

затверджена наказом ректора НУБіП України від "____"_____20___р. №_____

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи _____
_____
_____
_____
_____
_____

Перелік питань, що підлягають дослідженню:

1. __Предметна область дослідження. Системи рекомендацій_____
2. __Аналіз існуючих рішень_____
3. __Моделювання, розробка та тестування системи рекомендацій_____

Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "_____"_____20___ р.

Керівник магістерської кваліфікаційної роботи _____ ___Голуб Б.Л.___
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____Яковлєв О.І._____
(підпис) (прізвище та ініціали студента)

ЗМІСТ

Експертні системи рекомендацій отримали широке застосування у багатьох сферах діяльності, особливо в контексті цифрових платформ, що розповсюджують авторські розробки, такі як статті, фото та відео. Сьогодні існує значна кількість рекомендаційних систем, які використовуються для автоматизації підбору контенту для користувачів. Проте, не всі такі системи можуть бути адаптовані для специфіки розповсюдження авторських розробок, оскільки деякі рішення орієнтовані на інші галузі, такі як електронна комерція, потокове передавання відео або музики. Крім того, існує програмне забезпечення, яке використовується для розробки, моделювання та аналізу рекомендаційних систем. 12

Метою даної роботи є розробка програмного рішення, яке автоматизує процеси збору, обробки та аналізу даних про взаємодію користувачів з авторськими матеріалами. Це рішення має забезпечувати точні та релевантні рекомендації на основі вподобань користувачів, що ґрунтуються на їхніх попередніх взаємодіях із контентом. Система повинна бути гнучкою та масштабованою, що дозволить їй адаптуватися до змін у поведінці користувачів та розвитку платформи. Основна мета розроблюваного програмного забезпечення – забезпечити зручний інтерфейс для керування рекомендаціями, а також автоматизувати обробку даних про перегляди користувачів..... 16

В якості користувацького інтерфейсу зручно використовувати веб-інтерфейс на основі PHP та Bootstrap, що дозволить доступ до системи з будь-якого пристрою у будь-який час. Такий підхід також спрощує інтеграцію з існуючими системами управління контентом платформи та

надає можливість адміністратору легко керувати рекомендаційними правилами. Основна функціональність системи включає збір та аналіз даних про перегляди, створення правил для формування рекомендацій, автоматичний підбір контенту для користувачів, а також управління цими процесами через інтерфейс адміністратора..... 16

Сформулюємо основні функції розроблюваної системи рекомендацій для платформи розповсюдження авторських розробок. Ці функції ґрунтуються на потребах користувачів платформи та особливостях наявних рішень. Основні функції можна представити у вигляді списку:..... 16

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АСУП – автоматизована система управління підприємством;

БД – база даних;

СКБД – система керування базами даних;

СППР – система підтримки прийняття рішень;

ПЗ – програмне забезпечення;

DSS – Decision Support System;

UML – Unified Modeling Language;

ER – Entity-relationship;

LAMP – Linux, Apache, MySQL, PHP;

HTML – HyperText Markup Language;

PHP – Hypertext Preprocessor;

JS – JavaScript;

CSS – Cascading Style Sheets;

SQL – Structured Query Language.

ВСТУП

Протягом останніх років у світі спостерігається стрімкий розвиток інформаційних технологій. Разом із цим відбувається зростання конкуренції на ринку цифрових платформ, і ефективне управління контентом стає невід'ємною складовою успіху для будь-якого бізнесу, що працює з інформацією та авторськими розробками. Особливо це стосується платформ для розповсюдження авторських робіт, де необхідно забезпечити зручну взаємодію з користувачами, підтримку нових авторів, швидкий доступ до нових розробок і рекомендацій. У таких умовах важливим стає впровадження експертних систем рекомендацій, які дозволяють обробляти великі обсяги інформації, проводити аналіз даних про вподобання користувачів та надавати релевантні рекомендації. Сучасні експертні системи можуть поєднувати в собі різноманітні методи обробки даних, моделювання та алгоритми аналізу. Розробка та впровадження таких систем часто вимагає врахування специфічних потреб користувачів, особливостей платформи та вимог до обробки інформації.[1]

У даній роботі розглядається розробка програмного забезпечення експертної системи рекомендацій у системі розповсюдження авторських розробок, реалізованого за допомогою PHP, MySQL та Bootstrap. Актуальність теми дослідження обумовлена стрімким розвитком інформаційних технологій та їхньою інтеграцією в різні сфери бізнесу, зокрема в галузі цифрового контенту та креативних індустрій. Сучасні платформи потребують ефективного управління контентом та рекомендацій для підвищення залучення користувачів і підтримки нових авторів. Експертні системи рекомендацій стають необхідними, оскільки вони дозволяють автоматизувати процеси підбору контенту, персоналізувати взаємодію з користувачами та забезпечувати зростання інтересу до платформи. Важливим є впровадження автоматизованих механізмів, які дозволяють швидко реагувати на зміни в уподобаннях користувачів та популярності авторських робіт.

Метою дослідження є створення програмного рішення, що дозволить автоматизувати процеси збору, обробки та аналізу даних для надання оптимальних рекомендацій користувачам. Враховуючи можливі зміни у вподобаннях користувачів, розвитку авторського контенту та технологічні новації, розроблювана система має бути гнучкою, масштабованою та адаптивною, тобто бути придатною для використання на різних платформах розповсюдження авторських робіт.

Об’єктом дослідження є платформа розповсюдження авторських розробок, процеси якої потребують автоматизованих рекомендацій. Предметом дослідження виступає програмне забезпечення для підтримки рекомендацій, яке забезпечує автоматизацію процесів надання персоналізованих пропозицій та аналізу користувацької активності.

Завданнями дослідження є: системний аналіз предметної області, визначення основних проблем у сфері рекомендацій; формулювання вимог до експертної системи рекомендацій; моделювання архітектури та розробка системи, аналіз отриманих результатів.

Методи дослідження включають використання сучасних веб-технологій, таких як PHP для серверної логіки, MySQL для зберігання та обробки даних, та Bootstrap для створення адаптивного та зручного інтерфейсу. Для аналізу даних і формування рекомендацій використовуються алгоритми автоматизації процесів обробки інформації та методи персоналізації, що забезпечують адаптивність системи.

Наукова новизна роботи полягає в тому, що вперше буде розроблено веб-сайт як універсальну та просту у використанні експертну систему рекомендацій для платформи розповсюдження авторських розробок, що забезпечує автоматизацію процесів надання рекомендацій та підтримки нових авторів. Запропоновано удосконалення архітектури системи з інтеграцією автоматизованих інтерфейсів для управління рекомендаціями.

У рамках роботи виконується дослідження, що включає аналіз існуючих рішень у сфері рекомендаційних систем, вивчення методів обробки даних, а також визначення основних вимог до програмного забезпечення, які дозволять забезпечити максимальну ефективність та зручність використання для кінцевих користувачів. Робота має на меті проведення не лише теоретичних досліджень, але і практичну реалізацію, яка дозволить платформі розповсюдження авторських розробок підвищити ефективність взаємодії з користувачами, скоротити час на пошук цікавих матеріалів і, як наслідок, підвищити залученість користувачів до платформи. Також очікується, що результати роботи стануть корисними для подальших досліджень у галузі інформаційних технологій, відкриваючи нові можливості для впровадження інноваційних рішень у цифрових платформах.

Отже, дана робота спрямована на розробку комплексного підходу до створення програмного забезпечення експертної системи рекомендацій, що відповідає сучасним вимогам та потребам платформ для розповсюдження авторських розробок, створюючи тим самим основи для подальшого вдосконалення процесів надання рекомендацій та підтримки авторів.

Робота складається з чотирьох розділів. Перший розділ присвячено системному аналізу предметної області, в ньому визначено мету, предмет та об’єкт дослідження, проаналізовано існуючі рішення та поставлено задачі дослідження. У другому розділі виконано моделювання системи, визначено її структуру, побудовано відповідні діаграми. Третій розділ включає розробку клієнтського та серверного програмного забезпечення, відповідно до визначених у другому розділі моделей. У четвертому розділі описано системні вимоги, наведено інструкцію користувачів, проведено тестування та описано отримані результати роботи.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Предметна область дослідження

У даній роботі розглядається розробка програмного забезпечення для експертної системи рекомендацій у системі розповсюдження авторських розробок. Така система дозволяє автоматизувати процес підбору релевантного контенту для користувачів платформи, забезпечуючи персоналізовані рекомендації на основі їхніх вподобань та історії взаємодій. Це рішення орієнтоване на платформи, що працюють з великими обсягами авторських матеріалів, таких як статті, фото, відео та інші креативні розробки. Впровадження таких систем обумовлено зростанням обсягів інформації та потребою у персоналізованому підході до користувачів, що дозволяє підвищити їхню залученість та зручність взаємодії з платформою.[2]

Історія рекомендаційних систем бере свій початок з 90-х років XX століття, коли інтернет почав активно використовуватися як платформа для обміну інформацією.[3] Ранні системи були простими фільтрами, що надавали рекомендації на основі популярності контенту чи базових уподобань користувачів. Згодом розвиток алгоритмів машинного навчання та зростання потужностей обчислювальних систем дозволили створювати більш складні рекомендаційні механізми. З'явилися методи колаборативної та контент-орієнтованої фільтрації, які дали змогу враховувати як особисті вподобання користувачів, так і їх схожість з іншими користувачами.[4]

У 2000-х роках, з поширенням онлайн-платформ для обміну контентом, таких як YouTube, Amazon і Netflix, рекомендаційні системи стали невід'ємною частиною стратегії залучення користувачів. Вони дозволяли не лише збільшувати час перебування на платформі, але й сприяти відкриттю нового контенту та підвищувати лояльність користувачів. Сучасні рекомендаційні системи часто поєднують кілька підходів, створюючи гібридні системи, що комбінують колаборативну фільтрацію, фільтрацію на основі контенту та обробку великих обсягів даних для максимального задоволення потреб користувачів.

Розвиток рекомендаційних систем був обумовлений також зростанням кількості даних, які потрібно було аналізувати. На початку 2010-х років відбулося бурхливе зростання обсягів даних (Big Data), що вимагало нових підходів до обробки інформації. Водночас розвиток хмарних технологій дозволив створювати системи, здатні обробляти значні обсяги інформації в реальному часі. Це особливо важливо для платформ, що працюють з авторськими розробками, де кожен користувач має індивідуальні вподобання, і автоматизація процесу підбору рекомендацій стає критично важливою.[5]

Експертні системи рекомендацій у цифрових платформах дозволяють автоматизувати процес надання рекомендацій залежно від поведінки користувачів та наявної інформації про їхні інтереси. Вони є спеціалізованими формами інформаційних систем, які допомагають платформам аналізувати великі обсяги даних та надавати користувачам найбільш релевантний контент. Завдяки цьому, такі системи можуть значно підвищити якість роботи платформи, зменшуючи втрату інтересу користувачів та забезпечуючи релевантні рекомендації. Сучасні рекомендаційні системи враховують не лише попередні взаємодії користувача з контентом, але й його динамічні вподобання, що дозволяє адаптувати рекомендації відповідно до змін у його інтересах.

Інформація у рекомендаційних системах відіграє ключову роль, оскільки саме на її основі формуються висновки та рекомендації. Збір, обробка та аналіз даних забезпечують створення цілісної картини вподобань користувачів, що дозволяє платформі не лише надавати релевантні пропозиції, але й прогнозувати зміни у вподобаннях користувачів. Наприклад, на основі історії переглядів та взаємодій користувачів з різними матеріалами, система може визначити нових авторів або роботи, які можуть зацікавити користувача. Таким чином, інформаційна система забезпечує не лише зручність користувацької взаємодії, але й стратегічну перевагу в умовах зростаючої конкуренції між платформами.

Історично, першими експертними системами рекомендацій були ті, що використовували фіксовані правила та бази знань для формування

рекомендацій. Вони не могли адаптуватися до змін у вподобаннях користувачів без ручного втручання розробників. Проте з часом такі системи еволюціонували, і сьогодні вони здатні автоматично навчатися на основі даних про взаємодії користувачів. Це дозволяє створювати динамічні рекомендації, що враховують постійно змінювані інтереси користувачів.

Таким чином, система, що розглядається в роботі, є інформаційною системою, яка обробляє внутрішні дані з метою автоматизованого формування рекомендацій для користувачів. Основною метою такої системи є забезпечення персоналізованого підходу до кожного користувача платформи, що дозволяє підвищити його залученість та задоволеність від взаємодії з платформою.

Робота експертних систем рекомендацій пов’язана з процесами прийняття рішень, що включають збір, аналіз та оцінку даних. Рішеннями в контексті рекомендацій називають автоматизовану послідовність дій, спрямовану на надання користувачу рекомендацій на основі його взаємодій із платформою. Управлінське рішення в цьому випадку є результатом аналізу, обробки даних та вибору найбільш релевантних матеріалів для конкретного користувача.

Отже, підсумовуючи проведений аналіз предметної області дослідження, можна зробити висновок, що розробка експертної системи рекомендацій для платформи розповсюдження авторських розробок дозволяє вирішити низку важливих завдань. Це включає забезпечення персоналізованого підходу до надання рекомендацій, інтеграцію з базою даних платформи та адаптивний підхід до змін у вподобаннях користувачів, що є важливим для збереження конкурентних переваг у сучасному цифровому середовищі.

Рис. 1.1 Структура системи рекомендацій

1.2 Аналіз наявних рішень

Експертні системи рекомендацій отримали широке застосування у багатьох сферах діяльності, особливо в контексті цифрових платформ, що розповсюджують авторські розробки, такі як статті, фото та відео. Сьогодні існує значна кількість рекомендаційних систем, які використовуються для автоматизації підбору контенту для користувачів. Проте, не всі такі системи можуть бути адаптовані для специфіки розповсюдження авторських розробок, оскільки деякі рішення орієнтовані на інші галузі, такі як електронна комерція, потокове передавання відео або музики. Крім того, існує програмне забезпечення, яке використовується для розробки, моделювання та аналізу рекомендаційних систем.

У рамках даної роботи всі наявні системи можна умовно розділити на спеціалізовані системи, призначені для розповсюдження цифрового контенту, та універсальні рішення, що можуть бути адаптовані для різних галузей. Це поділення важливе, оскільки універсальні рішення часто не враховують специфіку роботи з авторськими матеріалами, а також можуть мати надлишковий функціонал, що ускладнює їх використання для конкретних завдань.

Також доцільно провести аналіз наукових публікацій, присвячених розробці рекомендаційних систем та їх застосуванню в різних галузях. Це

дозволить оцінити поточний стан досліджень, виявити актуальні напрямки розвитку та проаналізувати результати, отримані іншими дослідниками. Особливу увагу варто приділити системам, що орієнтовані на цифрові платформи для розповсюдження контенту, а також загальним підходам до розробки рекомендаційних систем, які можуть бути адаптовані для цієї сфери.

Аналіз наявних рішень доцільно розпочати з наукових статей та доповідей. Після цього варто розглянути існуюче програмне забезпечення, орієнтоване на рекомендації контенту, а також універсальні рекомендаційні системи, що можуть бути адаптовані для роботи з різними типами даних. Такі системи, як правило, є найбільш універсальними, проте для задач конкретної платформи можуть вимагати адаптації.

Детальний аналіз літературних джерел показав, що в науковій літературі питання розробки рекомендаційних систем для розповсюдження авторських розробок не завжди досліджується глибоко. Більшість публікацій зосереджена на загальних методах побудови рекомендаційних систем, таких як колаборативна фільтрація, контент-орієнтовані підходи та гібридні методи. Ці підходи використовуються в різних сферах, включаючи потокове відео, електронну комерцію та соціальні мережі.

Огляд літературних джерел пропонується розпочати з фундаментальних посібників, що описують основи побудови рекомендаційних систем, таких як "Introduction to Recommender Systems" та "Recommender Systems Handbook".

Ці посібники надають детальну інформацію про методи колаборативної фільтрації, алгоритми на основі матричної факторизації та гібридні методи. Ці теоретичні знання є основою для розробки ефективних систем рекомендацій, включаючи ті, що застосовуються на платформах для розповсюдження авторських робіт.

Одними з найбільш значущих прикладів застосування рекомендаційних систем є платформи, такі як Netflix та YouTube, які використовують складні алгоритми для надання персоналізованих рекомендацій на основі історії переглядів, оцінок та поведінки користувачів.

Такі системи можуть аналізувати великі обсяги даних, застосовуючи алгоритми машинного навчання та моделі глибокого навчання для визначення вподобань користувачів. Наприклад, Netflix використовує методи матричної факторизації для розуміння взаємозв'язків між користувачами та контентом, тоді як YouTube застосовує рекомендаційні системи на основі глибоких нейронних мереж для підбору відео, що можуть бути цікаві користувачам.

Для розповсюдження авторських розробок, таких як статті або графічні матеріали, варто розглянути такі платформи, як Medium або Behance, які застосовують рекомендаційні системи для надання користувачам персоналізованих рекомендацій. Medium використовує комбінацію алгоритмів на основі контенту та колаборативної фільтрації, враховуючи тематику та інтереси читачів. Behance пропонує рекомендації на основі переглядів робіт користувачів та схожості між художніми проектами.

Розглянуті літературні джерела доповнюють одне одного та надають достатній обсяг теоретичних відомостей для розробки рекомендаційної системи. Вони демонструють, як можна використовувати алгоритми та моделі для створення рекомендацій на базі даних про користувачів. Наприклад, у статті "Hybrid Recommender Systems for Digital Content" досліджуються способи поєднання колаборативної фільтрації та контент-орієнтованих підходів для покращення точності рекомендацій.

Огляд програмного забезпечення для рекомендаційних систем варто розпочати з розгляду таких фреймворків, як TensorFlow Recommenders та Surprise. TensorFlow Recommenders є бібліотекою для розробки моделей рекомендаційного типу на основі машинного навчання, що дозволяє створювати складні нейронні мережі для обробки великих обсягів даних. Surprise є бібліотекою Python для роботи з колаборативною фільтрацією, що дозволяє розробляти алгоритми для створення рекомендацій на основі взаємодії користувачів з контентом.

Також варто розглянути комерційні рішення, такі як SAP Commerce Cloud та Adobe Experience Platform, що надають інструменти для управління

цифровим контентом і рекомендацій. Ці рішення часто використовуються великими компаніями для надання персоналізованого досвіду користувачам. Проте, у випадку платформ для розповсюдження авторських розробок, такі системи можуть виявитися надмірно складними та дорогими для впровадження, що робить їх менш придатними для використання.

Серед безкоштовних та відкритих рішень варто виділити Apache Mahout та LensKit, які дозволяють будувати рекомендаційні системи на основі аналізу взаємодій користувачів з контентом. Apache Mahout орієнтований на обробку великих обсягів даних і може бути використаний для створення систем на основі колаборативної фільтрації. LensKit є більш спеціалізованим інструментом для академічних досліджень у сфері рекомендацій.

Завершуючи аналітичний огляд наявних рішень, можна зробити висновок про важливість адаптації існуючих алгоритмів та моделей під специфіку платформи розповсюдження авторських розробок. Розробка власної системи з урахуванням специфіки потреб користувачів та контенту дозволить досягти високої якості рекомендацій. Наприклад, система може використовувати базу даних переглядів для зберігання інформації про те, які роботи користувач вже бачив, і застосовувати колаборативну фільтрацію для пошуку подібних матеріалів, що ще не були переглянуті

1.3 Постановка завдання

Для подальшого виконання роботи необхідно, спираючись на проведений аналіз предметної області дослідження, визначити завдання, які мають бути вирішені при розробці експертної системи рекомендацій для платформи розповсюдження авторських розробок. Виходячи з проведеного аналізу, можна сформувати висновки про потреби цифрових платформ для розповсюдження контенту, недоліки існуючих рішень та особливості побудови рекомендаційних систем. Це дозволяє чітко сформулювати основні завдання для подальшої роботи.

Метою даної роботи є розробка програмного рішення, яке автоматизує процеси збору, обробки та аналізу даних про взаємодію користувачів з авторськими матеріалами. Це рішення має забезпечувати точні та релевантні рекомендації на основі вподобань користувачів, що ґрунтуються на їхніх попередніх взаємодіях із контентом. Система повинна бути гнучкою та масштабованою, що дозволить їй адаптуватися до змін у поведінці користувачів та розвитку платформи. Основна мета розроблюваного програмного забезпечення – забезпечити зручний інтерфейс для керування рекомендаціями, а також автоматизувати обробку даних про перегляди користувачів.

В якості користувацького інтерфейсу зручно використовувати веб-інтерфейс на основі PHP та Bootstrap, що дозволить доступ до системи з будь-якого пристрою у будь-який час. Такий підхід також спрощує інтеграцію з існуючими системами управління контентом платформи та надає можливість адміністратору легко керувати рекомендаційними правилами. Основна функціональність системи включає збір та аналіз даних про перегляди, створення правил для формування рекомендацій, автоматичний підбір контенту для користувачів, а також управління цими процесами через інтерфейс адміністратора.

Сформулюємо основні функції розроблюваної системи рекомендацій для платформи розповсюдження авторських розробок. Ці функції ґрунтуються на потребах користувачів платформи та особливостях наявних рішень. Основні функції можна представити у вигляді списку:

1. Рекомендаційний модуль:
- Збір даних про взаємодії: автоматичний запис переглядів користувачів у таблицю user_views, зберігання інформації про те, які матеріали переглядав кожен користувач.
- Аналіз історії переглядів: визначення авторських розробок, з якими користувач ще не взаємодіяв, та формування списку потенційних рекомендацій.

- Генерація рекомендацій: створення списку рекомендацій на основі аналізу історії переглядів та правил, визначених адміністратором.

2. Управління рекомендаційними правилами:

- Створення правил рекомендацій: адміністратор може створювати нові правила для формування рекомендацій через інтерфейс.
- Редагування правил: адміністратор має змогу редагувати існуючі правила для адаптації рекомендаційної логіки.
- Видалення правил: видалення застарілих або нерелевантних правил для оптимізації роботи системи.

3. Інтерфейс користувача та адміністратора:

- Виведення рекомендацій: інтерфейс для відображення користувачам рекомендованих авторських робіт у зручному форматі.
- Керування користувачами та їх взаємодіями: адміністратор може переглядати інформацію про користувачів та їхні взаємодії з контентом, що допомагає аналізувати ефективність рекомендацій.
- Адаптивний дизайн: забезпечення коректного відображення інтерфейсу на різних пристроях, включаючи персональні комп'ютери та мобільні пристрої.

Таким чином, на основі зазначеного вище можна виділити основні завдання, які необхідно вирішити під час розробки нового програмного продукту для рекомендаційної системи:

- Реалізувати функціонал збору та аналізу даних про взаємодії користувачів із контентом.
- Забезпечити можливість створення та редагування правил для автоматизації процесу рекомендацій.
- Реалізувати динамічне оновлення рекомендацій без перезавантаження сторінок.
- Забезпечити інтеграцію з базою даних для збереження інформації про перегляди та налаштування рекомендацій.
- Підтримувати адаптивний інтерфейс, що коректно відображається на різних пристроях.
- Забезпечити валідацію введених даних на клієнтській та серверній стороні для захисту системи від помилок та некоректних даних.

Виходячи з визначеного функціоналу, можна визначити приблизну структуру інтерфейсів користувачів та адміністратора для системи рекомендацій.

Реалізація зазначених завдань передбачає розробку відповідних інтерфейсів для створення, редагування та перегляду рекомендаційних правил, аналізу взаємодії користувачів та управління контентом. Усі дані, що вводяться через

графічний інтерфейс, повинні проходити валідацію для забезпечення надійної роботи системи.

Рішення системи носять рекомендаційний характер, а їх прийняття здійснюється користувачами системи, які мають можливість перегляду запропонованих матеріалів. При цьому користувачі системи повинні мати різні ролі, зокрема роль Адміністратора, що має доступ до всіх функцій управління правилами, та роль Користувача, що взаємодіє з контентом та отримує рекомендації.

Головні вимоги до програмного забезпечення: функціональність, адаптивність, безпека та продуктивність. Для досягнення цих характеристик програмне забезпечення має використовувати спеціальні засоби розробки на базі PHP, MySQL та JavaScript, що забезпечать стабільну роботу системи та її масштабованість.

Таким чином, у результаті постановки завдання були визначені основні цілі та функціональні вимоги до розроблюваної системи рекомендацій для платформи розповсюдження авторських розробок. Це дозволить створити ефективне та зручне програмне рішення, яке відповідатиме сучасним вимогам та потребам користувачів платформи.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Архітектура системи експертних рекомендацій для платформи розповсюдження авторських розробок.

Перед початком розробки програмного забезпечення важливим етапом є моделювання. На цьому етапі визначається архітектура системи, проектуються діаграми, що відображають взаємозв’язки між елементами системи та їхню взаємодію з користувачами[6]. Це досягається шляхом застосування методів моделювання, які ґрунтуються на результатах аналізу предметної області та постановці завдань.[7].

Процес моделювання системи рекомендацій пропонується виконати у три етапи. На першому етапі буде визначено архітектуру системи, на другому — побудовано діаграми, що детальніше відображатимуть структуру системи, її роботу та взаємодію з користувачами, а третій етап присвячений моделюванню бази даних, яка використовуватиметься системою для зберігання даних про користувачів, авторські матеріали та правила рекомендацій. Кожен з цих етапів може включати додаткові кроки, наприклад, на етапі складання моделей може бути доцільно визначити інформаційні потоки системи. Аналогічно, на етапі моделювання бази даних необхідно визначити всі можливі елементи, які потребуватимуть зберігання.

Моделювання системи підтримки прийняття рішень для рекомендаційної системи можна виконати як за функціональним, так і за об’єктно-орієнтованим підходом. В рамках даної роботи пропонується використовувати поєднання цих підходів, що дозволяє оптимально розподілити час на розробку кожного компонента системи. Тому доцільно розглянути визначення обох підходів, щоб обрати відповідний метод для кожної частини системи.

Архітектура системи рекомендацій визначається тим, що система буде реалізована як веб-додаток. Отже, архітектура повинна відповідати типовій клієнт-серверній моделі для вебсайтів.

Клієнт-серверна архітектура — це основний шаблон архітектури для програмного забезпечення, особливо під час розробки розподілених додатків, таких як вебсайти. Ця архітектура передбачає взаємодію між компонентами системи через обмін даними. Основними складовими клієнт-серверної архітектури є сервер, що обробляє дані і надає послуги, клієнт, що ініціює запити, та мережа, яка забезпечує їхню комунікацію[11].

У нашій системі клієнт-серверна архітектура передбачає, що клієнт є інтерфейсом користувача, через який адміністратори створюють та налаштовують правила рекомендацій, а користувачі взаємодіють із запропонованими рекомендаціями. Сервер обробляє запити, звертається до бази даних для отримання необхідних даних про авторські матеріали, взаємодії користувачів та рекомендації, формує відповіді та повертає їх клієнту для відображення.

На стороні клієнта виконуються наступні функції:

- відображення графічного інтерфейсу користувача;
- формування запитів та їх відправлення на сервер;
- отримання результатів виконання запитів від сервера і відображення їх у зручному для користувача форматі.

На стороні сервера виконуються наступні функції:

- зберігання, резервне копіювання та захист даних;
- обробка запитів від клієнта та формування відповідей;
- забезпечення функціоналу для роботи з базою даних, зокрема, для додавання нових правил рекомендацій та аналізу даних про взаємодію користувачів.

Система рекомендацій буде працювати на основі дворівневої клієнт-серверної архітектури, що включає клієнтську та серверну частини. Такий підхід є оптимальним для підтримки достатньої кількості одночасних запитів, спрощуючи серверне програмне забезпечення та підвищуючи ефективність обробки запитів від користувачів.

Основні операції системи:

1. Авторизація користувача. Першим кроком є авторизація, яка дозволяє розмежувати доступ до функціоналу для адміністраторів (налаштування правил рекомендацій) та користувачів (взаємодія з авторським контентом).

2. Відображення рекомендаційних списків. Відображення користувачу списку рекомендованих авторських матеріалів, створеного на основі його попередніх взаємодій із контентом.

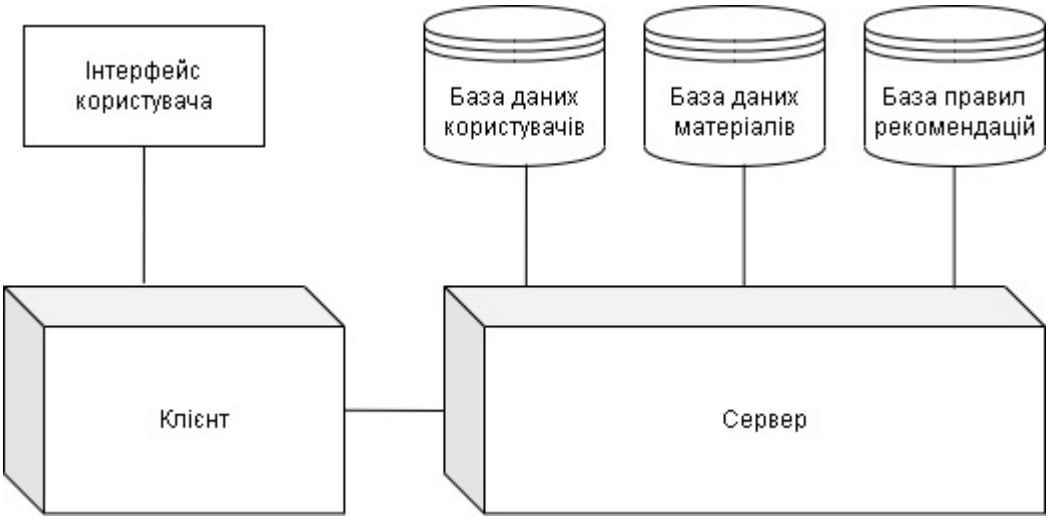

Рис. 2.1 Схема клієнт-серверної архітектури системи

3. Управління правилами рекомендацій (для адміністратора):

- Створення правил: можливість визначення нових правил рекомендацій на основі атрибутів матеріалів і поведінки користувачів.

- Редагування правил: зміна існуючих правил рекомендацій.

- Видалення правил: видалення нерелевантних або застарілих правил.

4. Аналіз даних про взаємодію: Система автоматично записує взаємодії користувачів із контентом у таблицю user_views, що дозволяє відстежувати перегляди та будувати рекомендації на їхній основі.

5. Пошук та фільтрація матеріалів: можливість для користувача знаходити авторські матеріали за певними критеріями, що покращує зручність навігації по платформі.

Функціональна структура системи:

Система підтримки рекомендацій для авторських розробок складається з наступних компонентів:

- Інтерфейс користувача (UI): забезпечує взаємодію між користувачем та системою, включаючи відображення рекомендованих матеріалів та форму для створення й редагування правил рекомендацій.
- Серверний компонент: обробляє запити від клієнтів, виконує запити до бази даних, формує рекомендації на основі правил та даних про взаємодії, зберігає і обробляє інформацію в базі даних.
- База даних: зберігає інформацію про користувачів, авторські матеріали, правила рекомендацій, історію переглядів, а також адміністративні налаштування. База даних виконує роль єдиного джерела даних для формування рекомендацій.

Основні операції системи:

- Авторизація користувачів: розмежовує доступ до функціоналу між адміністраторами та користувачами.
- Відображення рекомендацій користувачам: формує список рекомендованих матеріалів на основі переглядів і взаємодій.
- Управління правилами рекомендацій: адміністратор може створювати, редагувати і видаляти правила для покращення якості рекомендацій.
- Аналіз історії переглядів: зберігає дані про перегляди для створення індивідуальних рекомендацій.
- Фільтрація матеріалів: забезпечує зручність навігації та можливість знаходити потрібні матеріали на платформі.

Після визначення архітектури та функціональної структури системи, можна перейти до наступних етапів моделювання, що передбачають створення діаграм та моделювання бази даних.

Редагування матеріалу здійснюється шляхом зміни перелічених даних, таких як назва, опис, категорія, рейтинг, і збереження оновлень. Видалення матеріалу, як і у випадку з користувачем, здійснюється за ідентифікатором матеріалу і не потребує введення додаткових даних.

Додавання нового правила рекомендацій здійснюється адміністратором. Для цього необхідно ввести умову (логічний вираз у форматі системи керування базами даних), опис, а також саму рекомендацію, яка відображатиметься для користувачів. Рекомендації та умови мають бути внесені до бази даних заздалегідь. Видалення правила рекомендацій також здійснюється за ідентифікатором і не потребує введення додаткових даних.

Додавання нового користувацького запиту (наприклад, бажання отримати рекомендацію або залишити відгук) може здійснюватися як адміністратором, так і користувачем. Для цього необхідно вибрати категорію матеріалу, вказати критерії пошуку, ввести ім'я користувача та електронну пошту, а також додати опис або коментар до запиту. Статуси запитів (наприклад, "Новий", "Оброблено") мають бути попередньо внесені до бази даних, щоб забезпечити відповідне відображення запитів та їхній стан у системі. Редагування запиту передбачає зміну перелічених вище відомостей і збереження змін. Видалення запиту потребує лише ідентифікатора запиту, який система автоматично призначає, введення додаткових даних для цього не потрібно.

Для виконання пошуку матеріалів здійснюється запит до таблиці матеріалів з параметром пошуку. Пошук матеріалів може здійснюватися за назвою матеріалу, категорією або популярністю. Для користувацьких запитів пошук може здійснюватися за ім'ям користувача, електронною поштою або категорією матеріалу. Система знаходить всі схожі записи, в яких міститься пошукова послідовність, що полегшує роботу користувача. Відображення запитів за критеріями (наприклад, тільки оброблених) здійснюється аналогічно до пошуку: замість пошукового запиту на сервер передається ідентифікатор критерію. Ця операція не потребує введення додаткових запитів або створення додаткових таблиць у базі даних.

Отже, враховуючи необхідність автоматизації процесу підтримки рекомендацій, система повинна мати організоване зберігання даних (про матеріали, правила рекомендацій та користувацькі запити) та надавати простий,

інтуїтивно зрозумілий графічний інтерфейс, який дозволяє ефективно управляти рекомендаціями та запитами в системі.

2.2 Розробка UML-діаграм клієнтської та серверної частини системи

Важливим етапом моделювання системи підтримки рекомендацій є розробка діаграм, які відображають структуру системи, взаємодію її елементів між собою та з користувачем, а також її архітектуру. Для цього використовуються графічні засоби моделювання, що дозволяють створювати відповідні схеми. Завдяки такому моделюванню можна отримати діаграми, на основі яких буде виконуватися розробка програмного забезпечення.

На цьому етапі пропонується розглянути діаграму прецедентів використання, діаграму послідовності та діаграму впровадження, що відображає структуру системи. Також необхідно визначити структуру бази даних та представити її графічно у вигляді моделі. У цій системі база даних виконує одну з найважливіших функцій — зберігає всі дані про авторські розробки, правила рекомендацій, а також записи переглядів користувачів.

Завершення аналізу інформаційних процесів у системі підтримки рекомендацій дозволяє деталізувати функціонал клієнтської та серверної частини. Для цього зручно використовувати UML-діаграми, які дозволяють наочно відобразити взаємодію системи і користувача, а також елементів системи між собою.

UML-діаграми (Unified Modeling Language Diagrams) представляють собою засіб візуального моделювання, що сприяє розробникам програмного забезпечення у концептуалізації, проектуванні та втіленні складних систем. Вони включають кілька типів, які широко використовуються: діаграма класів, діаграма відношень, діаграма варіантів використання, діаграма діяльності, діаграма станів, діаграма послідовності, діаграма компонентів та діаграма впровадження[9].

В рамках цієї роботи доцільно виконати діаграму прецедентів використання, діаграму впровадження та діаграму послідовності. Це дозволить

наочно та у зручній формі представити відношення між користувачем і елементами системи, а також процес взаємодії елементів.

Діаграма прецедентів використання (Use-Case Diagram) відображає функціональність системи з точки зору взаємодії користувачів з нею. Основна мета цієї діаграми — показати, як користувачі або інші системи взаємодіють з системою, які функції вона виконує для них і як вони пов'язані між собою[9].

Діаграма впровадження (Deployment Diagram) використовується для моделювання фізичної архітектури системи. Основна її мета — показати, як компоненти програмного забезпечення розміщені на різних фізичних середовищах, таких як сервери, вузли мережі та пристрої.

Діаграма послідовності (Sequence Diagram) ілюструє взаємодії між об’єктами у хронологічному порядку, відображаючи участь об'єктів і порядок обміну повідомленнями[10].

За допомогою цих трьох діаграм можна створити уявлення про структуру системи, її функціонал і взаємодію елементів між собою.

Також необхідно побудувати діаграми, що відображають зв'язки між елементами системи та їхню взаємодію. Враховуючи рішення про об'єднання бази даних і бази правил, варто також розглянути структуру системи підтримки рекомендацій для авторських розробок.

Така структурна схема наведена на рис. 2.2.

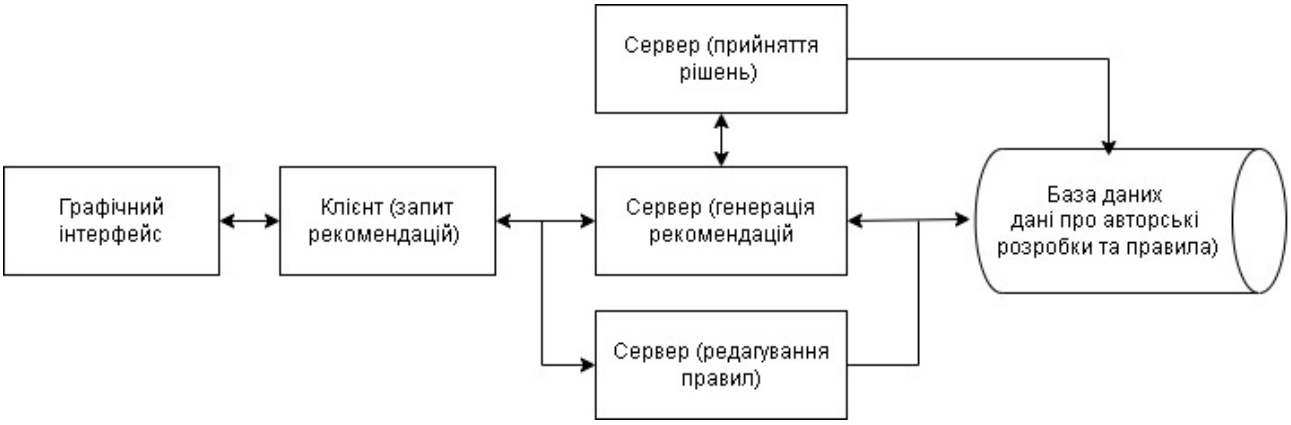

Рис. 2.2 Структурна схема системи

Згідно з наведеною схемою в системі відбувається взаємодія елементів зі сторони клієнта та сервера (включаючи базу даних).

Результати аналізу інформаційних процесів дозволяють зробити висновок про потреби користувача та створити діаграму прецедентів використання (рис. 2.3). Вона відображає основні функціональні блоки програмного забезпечення, що повинні бути виконаними.

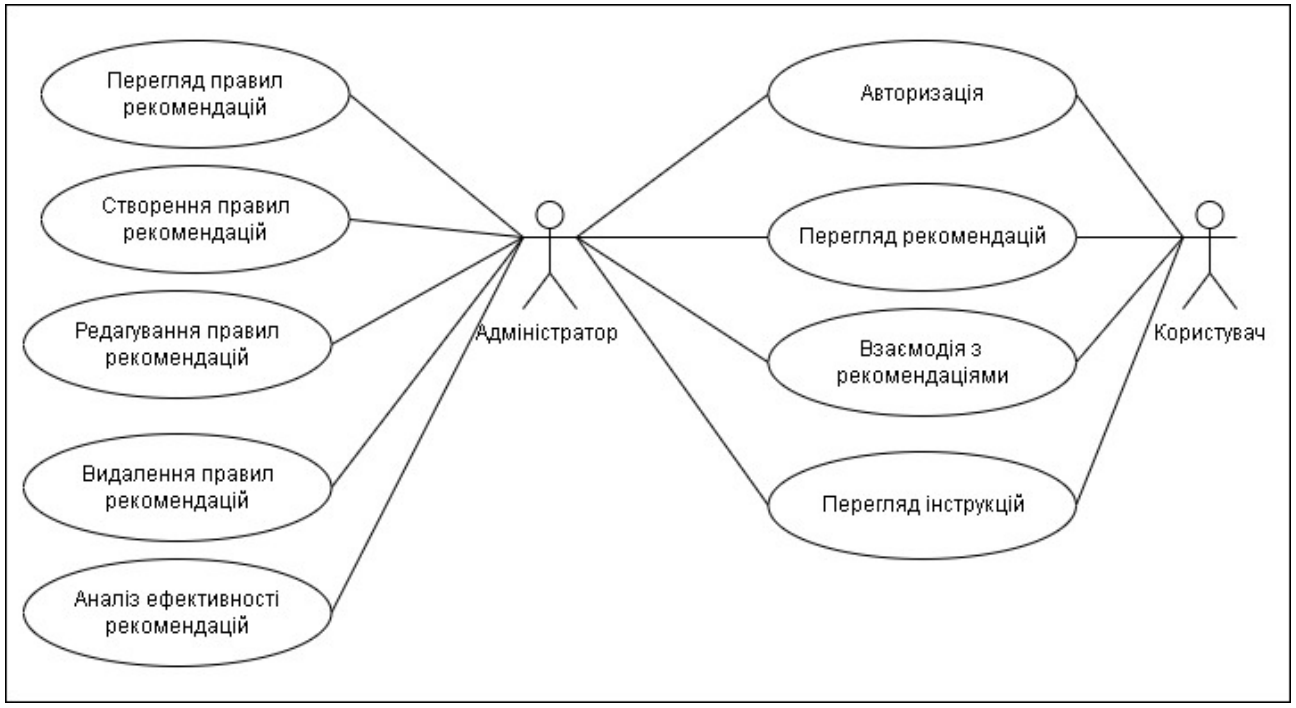

Рис. 2.3 Діаграма прецедентів використання

Згідно з побудованою діаграмою, можна зробити висновок про функціонал системи для різних користувачів та наочно побачити відмінності між ролями. Усі користувачі системи можуть переглядати інструкцію, проходити авторизацію та переглядати надані їм рекомендації. Водночас увесь інший функціонал системи доступний лише Адміністратору, який має можливість керувати правилами рекомендацій, створювати та редагувати їх, а також аналізувати ефективність наданих рекомендацій. Це розмежування забезпечує зручність використання для користувачів і дозволяє Адміністратору контролювати якість та релевантність рекомендацій.

На діаграмі впровадження (рис. 2.4) зображуються програмно-апаратні компоненти системи, що розробляється. В клієнт-серверній архітектурі виділяється частина клієнта, частина сервера та частина сервера бази даних, що відповідним чином відображається на структурі програмного забезпечення.

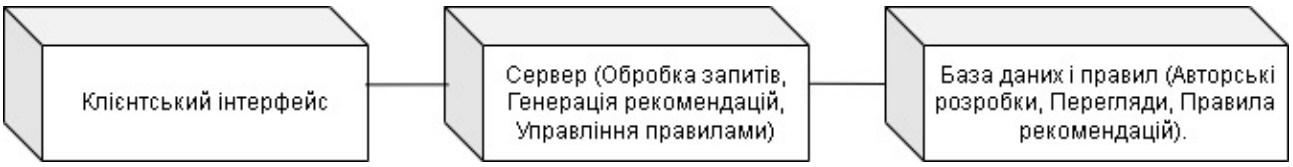

Рис. 2.4 – Діаграма впровадження

Як видно з діаграми впровадження, обробка даних здійснюється клієнтським інтерфейсом, який взаємодіє з сервером, що виконує операції з базою даних. Така структура системи дозволяє формувати рекомендації на основі аналізу даних про перегляди та інтереси користувачів, не використовуючи надлишкові зовнішні ресурси.

Крім того, в рамках цієї роботи доцільно розробити діаграму послідовності. Розглянемо діаграму послідовності для процесу відображення персоналізованих рекомендацій в інтерфейсі користувача. Розроблена діаграма послідовності наведена на рис. 2.5.

На цій діаграмі буде показано, як користувач надсилає запит на рекомендації, сервер обробляє запит, застосовуючи правила рекомендацій, і передає результат у вигляді списку розробок для відображення в інтерфейсі користувача.

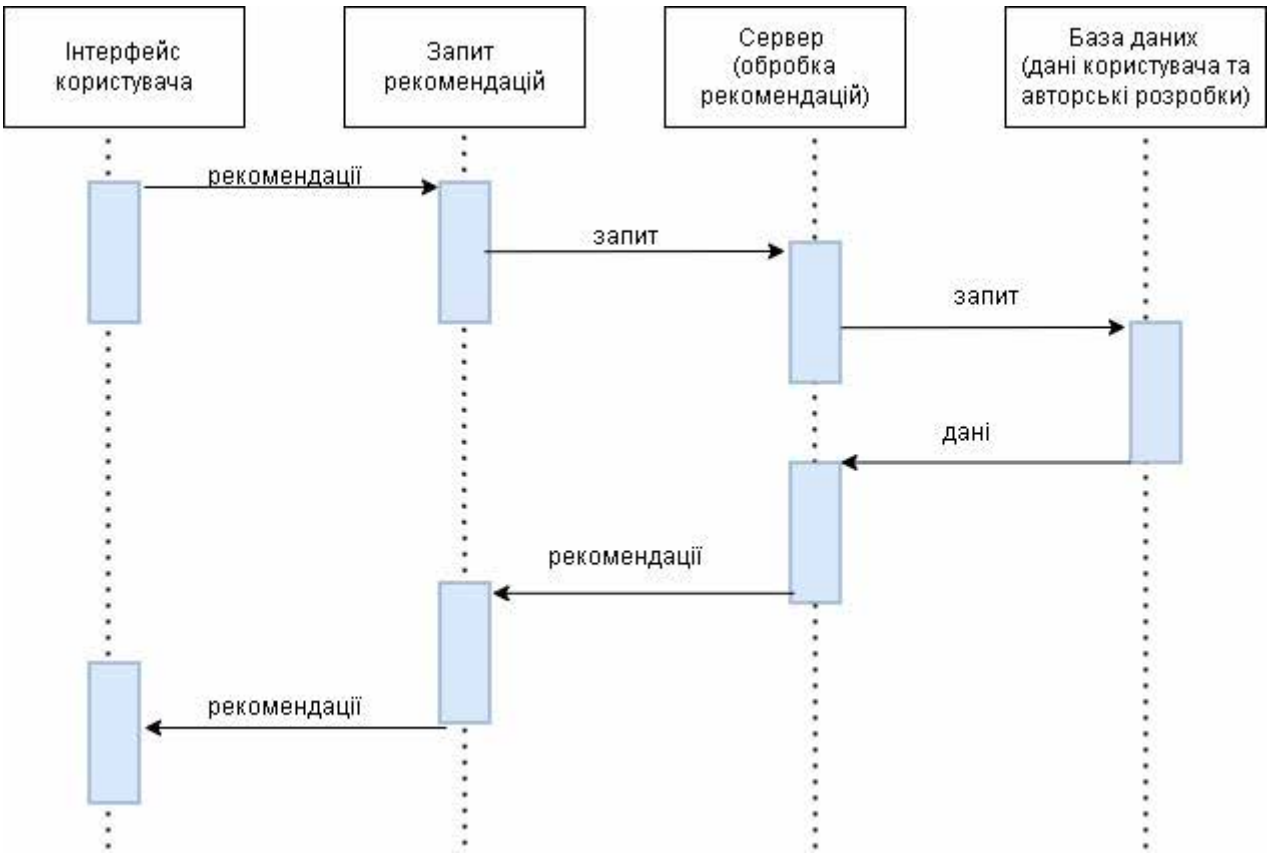

Рис. 2.5 Діаграма послідовності

З діаграми видно, що процес рекомендацій ініціюють елементи графічного інтерфейсу користувача. Після цього програмне забезпечення клієнта формує та відправляє запит на сервер для отримання персоналізованих рекомендацій. Сервер, в свою чергу, звертається до бази даних, щоб отримати інформацію про користувача, його попередні перегляди та авторські розробки, які можуть його зацікавити. База даних повертає серверу відповідні дані, на основі яких сервер формує рекомендації. Потім сервер відправляє сформовані рекомендації клієнту, а клієнт відображає їх у графічному інтерфейсі для користувача.

Такий підхід забезпечує динамічне формування рекомендацій, що покращує взаємодію користувача із системою.

2.3 Моделювання бази даних

Моделювання бази даних системи рекомендацій у системі розповсюдження авторських розробок доцільно розпочати з формулювання концептуальних вимог користувачів до бази даних. Це можливо зробити за допомогою концептуальних запитів. Вивчення потреб користувачів та аналіз доступної інформації дозволяє сформувати такі концептуальні запити:

1. Вивести інформацію про користувачів (логін, E-mail, ПІБ, роль).

2. Вивести інформацію про авторські розробки (назва, опис, категорія, зображення, кількість переглядів, дата додавання, дата останнього перегляду, рекомендація).

3. Вивести інформацію про рекомендації (ID, ID розробки, назва розробки, категорія, кількість переглядів, дата рекомендації, тип рекомендації, статус).

4. Вивести інформацію про правила формування рекомендацій (умова, опис, рекомендація).

5. Вивести інформацію про авторські розробки за заданою назвою або категорією.

6. Вивести інформацію про рекомендації для певного користувача або категорії розробок.

7. Вивести активні рекомендації, тобто рекомендації з поточним статусом «активна».

8. Вивести завершені або неактуальні рекомендації.

Наведений перелік не є вичерпним і може розширюватись, проте описаного достатньо для демонстрації основного функціоналу системи, оскільки це мінімально необхідний набір функцій. Головна задача – відображення даних, отриманих з бази та перевірених системою рекомендацій.

Етап інфологічного моделювання включає формування категорій предметної області. Категорії предметної області будуються на основі запитів користувача, але при цьому враховують зв’язки між сутностями, тому в описі однієї сутності можуть бути присутні інші, що її характеризують. Основні категорії системи рекомендацій у системі розповсюдження авторських розробок можна подати в такому вигляді:

1. Роль (ID, роль).

2. Рекомендація (ID, рекомендація).

3. Статус рекомендації (ID, статус).

4. Користувач (ID, логін, пароль, E-mail, ПІБ, роль).

5. Розробка (назва, опис, категорія, зображення, кількість переглядів, дата додавання, дата останнього перегляду, рекомендація).

6. Рекомендаційний запис (ID, розробка, категорія, кількість переглядів, дата рекомендації, тип рекомендації, статус).

7. Правила формування рекомендацій (умова, опис, рекомендація).

Отже, для опису бази даних системи рекомендацій у системі розповсюдження авторських розробок необхідні такі сутності: роль, рекомендація, статус рекомендації, користувач, розробка, рекомендаційний запис і правила формування рекомендацій. Співвідношення між цими сутностями зручно представити у вигляді ER-моделі.

ER-модель (Entity-Relationship Model) є методологією моделювання баз даних, яка служить для опису структури та взаємозв’язків між даними в системі. Використання ER-моделі дозволяє абстрагувати ключові аспекти даних та їх відносини, щоб створити зрозумілу та легко керовану базу даних. В рамках ER-моделі бази даних виокремлюються кілька основних понять, зокрема: сутності, відношення та атрибути. Цю модель можна представити у вигляді діаграми, що ілюструє структуру та взаємозв’язки між сутностями бази даних.

ER-модель є важливим інструментом для розробки баз даних, оскільки вона дозволяє розуміти структуру даних і відносини між ними ще на етапі проектування, що сприяє ефективній розробці та підтримці бази даних у майбутньому.

Представлення ER-моделі для бази даних системи підтримки прийняття рішень керівництвом мереж магазинів зображено на рисунку 2.6.

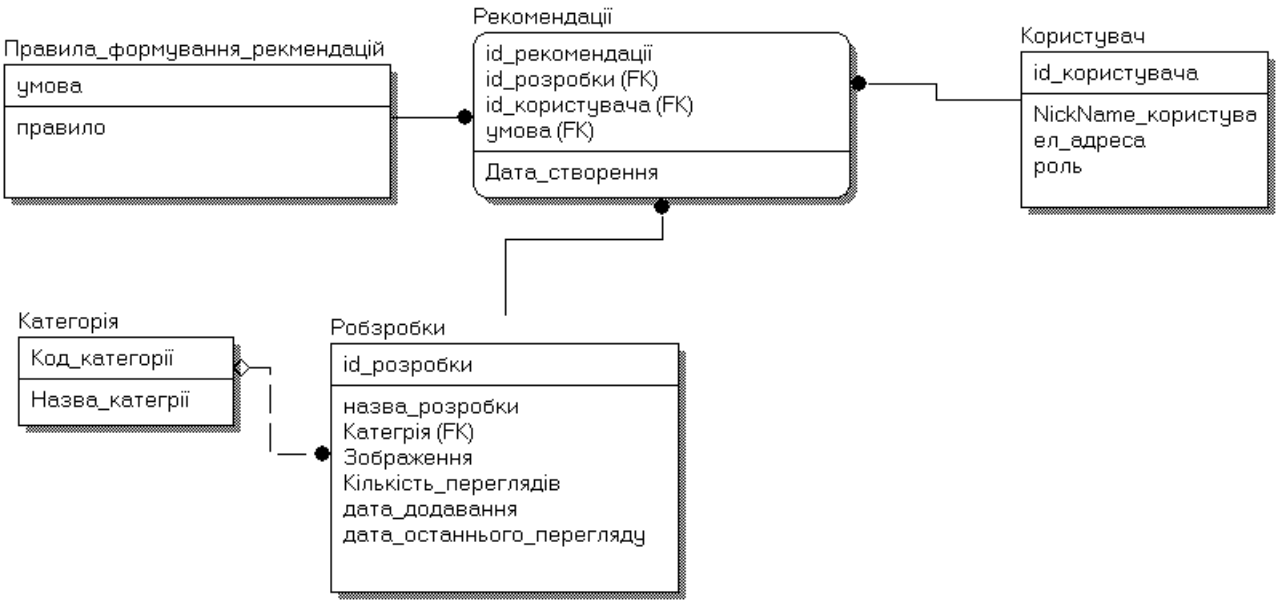

Рисунок 2.6 – ER-модель бази даних

Як видно з діаграми, користувачу належить роль, користувач може створювати, редагувати (в тому числі, приймати рішення) та видаляти товари, замовлення та правила прийняття рішень. Замовлення мають статус, товари мають прийняті рішення. Правила прийняття рішення стосуються товарів. Товар може належати до замовлення. Таким чином взаємодіють елементи бази даних.

Проаналізувавши діаграму ER-моделі бази даних системи ідтримки прийняття рішень, можна зробити висновок про характеристики зв’язків бази даних. Для цього зручно використовувати таблицю, до якої вносяться сутності і опис їхніх зв’язків. Характеристики зв’язків для бази даних СППР представлено в таблиці 2.1.

Таблиця 2.1 – Характеристики зв’язків бази даних

Ім’я сутності 1	Ім’я сутності 2	Тип зв’язку	Ім’я зв’язку	Клас належності
Користувач	Рекомендації	1:Б	Має рекомендації	Обов’язковий
Рекомендації	Автрські розробри	1:1	З’вязані з робтою	Обов’язковий
Автрські розробри	Категрія	1:Б	Належить до категрії	Обов’язковий

Правила формування рекомендацій	Рекомендації	1:Б	Використовує правило	Обов’язковий
Користувач	Автрські розробри	1:Б	Додає/переглядає/ видаляє	Обов’язковий
Користувач	Категрія	1:Б	Додає/переглядає/ категрії	Обов’язковий

Моделювання бази даних для системи рекомендацій авторських розробок

Процес моделювання бази даних системи рекомендацій авторських розробок включає визначення концептуальних вимог користувачів, аналіз функціональних залежностей та проектування логічної структури бази даних. Основними завданнями є забезпечення ефективного зберігання, обробки даних та швидкого доступу до рекомендацій, розробок і користувачів.

Концептуальні запити

Концептуальні запити дозволяють формалізувати потреби користувачів у вигляді операцій, які база даних повинна підтримувати:

- Вивести інформацію про користувачів (нікнейм, електронна адреса, роль).
- Вивести рекомендації для конкретного користувача.
- Вивести список авторських розробок (назва, категорія, кількість переглядів, дата додавання).
- Вивести категорії розробок.
- Вивести список правил формування рекомендацій (умова, опис, пріоритет).
- Оновити кількість переглядів розробки після взаємодії користувача.
- Згенерувати рекомендації на основі правил та взаємодій користувача.
- ER-модель бази даних

Основні сутності системи рекомендацій:

- Користувачі (Users): зберігає дані про користувачів, їх ролі, нікнейми та електронні адреси.
- Рекомендації (Recommendations): зв’язує користувачів із запропонованими розробками на основі умов.
- Авторські розробки (Works): містить інформацію про роботи, їх категорії, перегляди та дати оновлення.
- Категорії (Categories): класифікує авторські розробки за напрямками.
- Правила формування рекомендацій (RecommendationRules): встановлює умови для створення рекомендацій.

Зв’язки між сутностями

- База даних розроблена з урахуванням зв’язків типу «один до багатьох»:
  - Один користувач може отримувати багато рекомендацій.
  - Одна розробка може належати до однієї категорії.
  - Одне правило може використовуватись для багатьох рекомендацій.

Нормалізація

Для оптимізації структури бази даних використано принципи нормалізації до третьої нормальної форми:

- Усі атрибути залежать тільки від первинного ключа.
- Уникнено транзитивних залежностей.
- Забезпечено мінімізацію надмірностей даних.

Структура бази даних

Таблиця Users

id (INT, PRIMARY KEY, AUTO_INCREMENT): унікальний ідентифікатор користувача.

nickname (VARCHAR, 50): нікнейм користувача.

email (VARCHAR, 100): електронна адреса.

role (ENUM): роль користувача (адміністратор, користувач).

Таблиця Works

id (INT, PRIMARY KEY, AUTO_INCREMENT): ідентифікатор розробки.

name (VARCHAR, 100): назва розробки.

category_id (INT, FOREIGN KEY): категорія розробки.

views (INT): кількість переглядів.

created_at (TIMESTAMP): дата створення.

Таблиця Categories

id (INT, PRIMARY KEY, AUTO_INCREMENT): ідентифікатор категорії.

name (VARCHAR, 50): назва категорії.

Таблиця Recommendations

id (INT, PRIMARY KEY, AUTO_INCREMENT): ідентифікатор рекомендації.

user_id (INT, FOREIGN KEY): ідентифікатор користувача.

work_id (INT, FOREIGN KEY): ідентифікатор розробки.

created_at (TIMESTAMP): дата створення рекомендації.

Таблиця RecommendationRules

id (INT, PRIMARY KEY, AUTO_INCREMENT): ідентифікатор правила.

condition (TEXT): умова формування рекомендації.

description (TEXT): опис правила.

priority (TINYINT): пріоритет правила.

Зв’язки таблиці users з іншими таблицями представлені на рисунку 2.7.

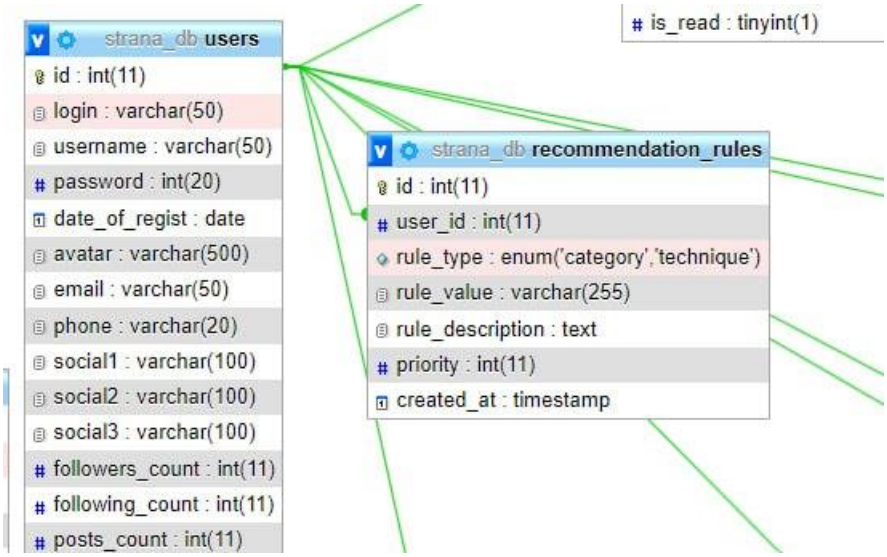

Рис. 2.7 Співвідношення таблиці users з іншими таблицями

Аналогічними за своєю структурою є таблиці recommendation_rules та user_views, що зберігають правила формування рекомендацій та записи про перегляди контенту користувачами. Такі дані зручно зберігати в окремих таблицях, оскільки це дозволяє автоматизувати процес формування рекомендацій, використовуючи стандартні правила для всіх користувачів системи.

Таблиця recommendation_rules містить інформацію про типи правил, їх значення, опис та пріоритети, що дозволяє легко змінювати або додавати нові правила без потреби ручного втручання в алгоритми системи.

Таблиця user_views зберігає записи про перегляди контенту користувачами, включаючи дату перегляду, контент, який було переглянуто, та позначку, якщо контент не відповідає рекомендаціям. Це дозволяє системі на основі зібраних даних про перегляди автоматично аналізувати поведінку користувачів і адаптувати рекомендації. Отже, таблиця recommendation_rules наведена на рис. 2.8.

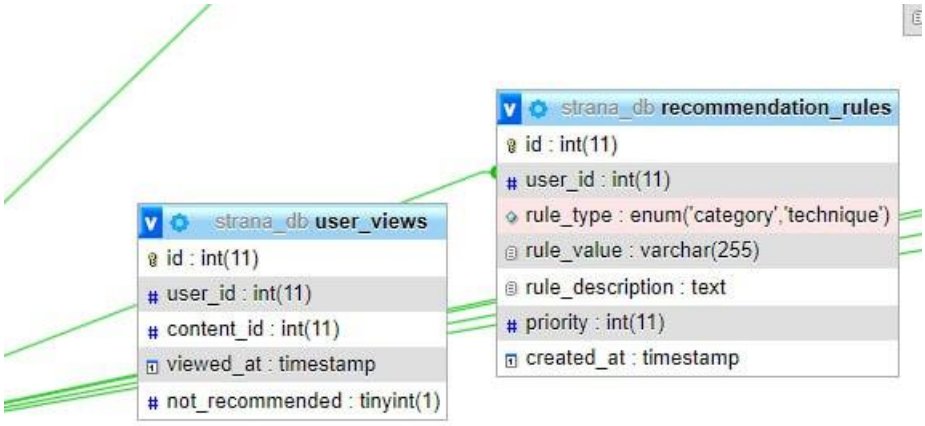

Рис. 2.8 Співвідношення таблиці acts з іншими таблицями

3 РОЗРОБКА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

3.1 Обґрунтування вибору засобів розробки

Під час розробки важливо правильно визначити оптимальні засоби для реалізації отриманих моделей, відповідно до поставлених задач. Необхідно враховувати архітектуру системи, її структуру, взаємодію елементів, вимоги до кінцевого програмного продукту тощо. Вибір засобів розробки необхідний для того, щоб підібрати оптимальні технології, які можна застосувати для досягнення поставленої мети[12].

В першу чергу варто відзначити, що розроблювана система є веб-додатком, а значить має клієнт-серверну архітектуру, що складається з клієнта, сервера та бази даних. Отже, і вибір засобів розробки має відповідати структурі системи. Технології, що використовуються для розробки програмного забезпечення на стороні клієнта та сервера можуть мати суттєві відмінності. Незважаючи на існування технологій, що дозволяють розробляти програмне забезпечення як на стороні клієнта, так і на стороні сервера, в даному випадку доцільно розглянути окремо мови розмітки та програмування, як для клієнта, так і для сервера[

Також під час вибору програмних засобів розробки важливо враховувати вимоги до кінцевого програмного продукту. Наприклад, якщо є умова збереження адаптивності на різних пристроях, необхідно підбирати такі засоби розробки, які дозволяють це забезпечити.

Отже, вибір засобів розробки програмного забезпечення можна умовно розділити на три основні частини: вибір засобів розробки клієнтської частини, вибір засобів розробки серверної частини та вибір системи управління базами даних. Послідовне виконання цих трьох етапів дозволить підібрати оптимальні засоби розробки та створити якісний програмний продукт. Порядок виконання цих етапів наразі не має значення, тому в роботі пропонується спочатку розглянути бази даних, потім клієнт і в останню чергу сервер. Враховуючи вимоги до програмного забезпечення, сформульовані в попередніх розділах, підберемо також технології для оптимальної клієнт-серверної взаємодії.

Оскільки в даній системі найважливішу роль відіграють дані, очевидно, що для її роботи необхідне застосування системи управління базами даних. База даних (БД) – це впорядкований набір логічно пов’язаних даних, що може використовуватись спільно, відповідно до інформаційних потреб користувачів. БД призначені для задоволення користувацьких інформаційних потреб .

Застосування БД пов’язане з численними перевагами, серед яких одна з найважливіших – оптимізація та полегшення зберігання та організації великих обсягів інформації. БД дозволяють зберігати дані в таблицях та структурах, що значно спрощує управління ними. Права доступу, що застосовуються в базах даних, дозволяють забезпечити належну конфіденційність та захист інформації. Крім цього, застосування баз даних дозволяє формувати і виконувати різноманітні запити до даних для швидкого та ефективного отримання доступу до необхідної інформації. Використання БД забезпечує цілісність даних та уникнення дублювання інформації.

Бази даних можуть засновуватись на різних моделях. Модель даних являє собою деяку абстракцію, що відображає найважливіші аспекти функціонування предметної області, ігноруючи другорядні. Іншими словами, модель даних є деякою цільовою моделлю предметної області. У моделях даних можна розрізнити три основні складові: структурна, управляюча частина та класи обмежень цілісності даних. Процес створення логічного представлення структури БД називається моделюванням даних. Таке моделювання було виконано в попередньому розділі, але уточнень моделі та без конкретизації вибору системи управління базами даних.

Однією з найбільш розповсюджених моделей даних є реляційна модель, що заснована на математичному понятті відношення і представлення відношень у формі таблиць. Саме така модель буде використана в системі підтримки прийняття рішень. Такий вибір обумовлений логічністю, простотою та інтуїтивністю в структуруванні даних, що полегшить розробку та експлуатацію програмного забезпечення. Отже, необхідно обґрунтувати вибір реляційної системи управління базами даних.

Мови реляційного числення засновані на класичному численні предикатів і надають користувачу правила для написанні запитів до БД. Серед загальних характеристик для різних видів серверів баз даних можна відзначити використання реляційної мови SQL для реалізації запитів до даних.

SQL (Structured Query Language) – це стандартна мова для зберігання, обробки та отримання даних в базах даних. За допомогою цієї мови можна описати набори даних, що допомагають відповісти на запити. Використовуючи SQL, важливо дотримуватися правильного синтаксису – набору правил, що забезпечують правильне поєднання елементів мови.

Існує велика кількість систем управління базами даних, серед яких найбільш відомими є MS SQL, MySQL та PostgreSQL. Ці системи мають багато спільного і використовують мову SQL, проте можуть мати незначні відмінності. Вибір оптимальної реляційної СУБД для системи підтримки прийняття рішень[13].

Прийнято рішення про доцільність використання СУБД MySQL, що є вільною системою управління реляційними базами даних і спроектована для оптимізації обробки великих обсягів даних. Вона створена як альтернатива комерційним системам і відзначається своєю швидкодією та ефективністю. MySQL широко використовується у веб-розробці, має велику спільноту розробників та багату документацію, що полегшує її використання та підтримку. Крім того, MySQL є популярним серед багатьох хостинг-провайдерів, включаючи безкоштовні, і має високий рівень розповсюдження. Отже, така система управління базами даних ідеально підходить для реалізації експертної системи рекомендацій у системі розповсюдження авторських розробок.

Отже, прийнято рішення про використання реляційної системи управління базами даних MySQL. Розробка серверної частини програмного забезпечення буде виконуватись з використанням цієї технології для забезпечення доступу до бази даних. Таким чином, обґрунтування вибору технологій для роботи з базою даних завершено.

На стороні клієнта програмне забезпечення має надавати користувачу зручний та інтуїтивно зрозумілий графічний інтерфейс. Його розробляють з використанням мови розмітки HTML, доповнюючи її каскадними таблицями стилів та скриптами, що дозволяють робити сторінки інтерактивними.

На цьому етапі важливим фактором є те, що веб-додаток має бути адаптивним і працювати однаково на всіх пристроях від настільних комп'ютерів до мобільних пристроїв. Такий результат може бути забезпечений завдяки спеціальним технологіям, таким як шаблони стилів, каскадні таблиці стилів та скрипти тощо. Проте спочатку варто розглянути загальні відомості про мову розмітки HTML.

HTML (Hypertext Markup Language) – це стандартна мова розмітки, що застосовується для розробки та відображення веб-сторінок у веб-браузерах. Вона складається з різних тегів, які визначають структуру та зовнішній вигляд контенту на сторінці (текстові дані, зображення, гіперпосилання тощо). Сьогодні HTML використовується як основна мова для створення веб-сторінок та їхньої взаємодії з користувачами через гіпертекстові посилання та форми. Це одна з основних технологій веб-розробки, яка разом з CSS та JavaScript надає розробникам можливості створення структурованих та доступних сторінок, які можна легко інтерпретувати та відображати у веб-браузерах різноманітних пристроїв.

CSS (Cascading Style Sheets) – це мова розробки стилів, що використовується для оформлення та контролю зовнішнього вигляду веб-сторінок. CSS описує, як елементи відображатимуться у браузері. Застосування CSS дозволяє відокремити оформлення від структури сторінки, спрощуючи підтримку та модифікацію коду. Стилі можна застосовувати безпосередньо до елементів сторінки або використовувати зовнішні стилі, які потім включаються в код сторінки. Крім того, CSS підтримує каскадне спадкування стилів, завдяки чому можна визначати загальні стилі для батьківських елементів і змінювати їх для дочірніх, спрощуючи управління виглядом веб-сайтів[16].

JavaScript – це скриптова мова програмування, призначена для розробки динамічних веб-сторінок та інтерактивних додатків. Вона застосовується для реалізації різноманітних функціональностей на веб-сторінках, таких як обробка подій користувача, динамічне оновлення вмісту сторінок, анімація, валідація форм, інтерактивність з користувачем та взаємодія з сервером без перезавантаження сторінки. JavaScript виконується у веб-браузері клієнта, що дозволяє створювати багатофункціональні та захоплюючі веб-додатки .

На JavaScript засновані також інші технології, які можуть бути застосовані під час розробки системи підтримки прийняття рішень. Однією з таких технологій є AJAX (Asynchronous JavaScript and XML) – це технологія, яка дозволяє веб-додаткам обмінюватися даними з сервером асинхронно, без необхідності перезавантаження всієї веб-сторінки. Завдяки AJAX, користувачі можуть взаємодіяти з веб-сайтом більш інтерактивно, оскільки лише частини контенту оновлюються в реальному часі. Це здійснюється за допомогою JavaScript, який надсилає запити до сервера і отримує відповіді, які можуть бути у форматах XML, JSON або HTML. AJAX робить веб-додатки швидшими.

ootstrap – це фреймворк, призначений для розробки стильних і адаптивних веб-сайтів. Заснований на HTML, CSS і JavaScript, він пропонує розмаїття готових компонентів та стилів, що спрощує швидке створення інтерфейсів. Bootstrap має широкий вибір гнучких компонентів, таких як кнопки, навігаційні панелі, форми, таблиці, каруселі, модальні вікна тощо. Особливістю є його сіткова система, що дозволяє створювати респонсивні макети, які легко адаптуються до різних розмірів екранів та пристроїв. Bootstrap широко використовується веб-розробниками завдяки простоті використання та багатому функціоналу, що допомагає зберегти час та зусилля під час створення сучасних веб-додатків[17].

Отже, клієнтська частина системи буде розроблена з використанням мови розмітки HTML, каскадних таблиць стилів CSS та JavaScript, зокрема технології AJAX, що дозволить динамічно оновлювати дані на сторінках без необхідності повного перезавантаження. Для забезпечення адаптивності сайту вирішено

використовувати технологію Bootstrap. На цьому обґрунтування вибору засобів розробки програмного забезпечення на стороні клієнта завершено.

Серверна сторона повинна обробляти великі обсяги інформації. На сервер надходитимуть запити від клієнтів, формуватимуться та виконуватимуться запити до бази даних, виконуватиметься обробка даних, формування та відправлення відповідей. Для цього необхідне спеціалізоване серверне програмне забезпечення, для роботи з яким існують спеціалізовані серверні мови розробки. Для розробки серверної частини можуть бути використані різні мови програмування. Серед них PHP, Node.js, ASP.NET та інші. Кожна з цих мов має свої особливості, які важливо враховувати під час вибору технологій для розробки сервера. Прийнято рішення використовувати мову програмування PHP для розробки серверної частини системи підтримки прийняття рішень.

PHP (Hypertext Preprocessor) – мова сценаріїв загального призначення з відкритим вихідним кодом, що отримала широке застосування у веб-розробці. Однією зі значних особливостей PHP є те, що такий код може бути вбудований безпосередньо в HTML, спрощуючи розробку динамічних веб-сторінок. Основна відмінність PHP від клієнтських мов програмування полягає в тому, що PHP-код виконується на стороні сервера, що дозволяє генерувати HTML, який потім надсилається клієнту. Тобто клієнт отримує результати виконання сценарію, але не має доступу до вихідного коду. Завдяки можливості налаштування веб-сервера для обробки всіх HTML-файлів через PHP, можна забезпечити, що користувачі не отримають доступ до вихідного коду. Одна з найбільш значних переваг PHP – його легкість в опануванні для початківців, а також наявність багатьох додаткових функцій для професійних розробників.

Однією з особливостей мови програмування PHP є її підтримка взаємодії з базами даних MySQL. Це дозволяє легко реалізувати сервер, здатний обробляти значні обсяги даних, без використання стороннього програмного забезпечення.

Отже, для розробки серверної частини експертної системи рекомендацій у системі розповсюдження авторських розробок буде застосовано мову програмування PHP та реляційну СУБД MySQL.

3.2 Розробка клієнтської частини системи

Для розробки програмного забезпечення на стороні клієнта, як було визначено в попередньому підрозділі, буде використовуватись мова розмітки HTML. Проте у клієнт-серверних системах, що використовують мову програмування PHP, є можливість поєднання клієнтського та серверного коду в одному файлі. Зазвичай такі файли мають розширення .php і містять код обома мовами одночасно. Сервер приховує від користувачів частину, розроблену з використанням PHP, і відправляє виключно HTML-код. Таке рішення обумовлено більшою простотою розробки та більш широкими можливостями щодо об’єднання клієнтських та серверних функцій в одній сторінці[14].

В першу чергу визначимо основні форми, які мають бути доступні користувачу на стороні клієнта. Користувач повинен мати доступ до головної сторінки, на якій знаходиться навігаційна панель та головне меню, що відображає основний функціонал системи. Навігаційна панель розміщується у верхній частині сторінки і має такі основні пункти: назва системи, головна сторінка, інструкція користувача, кнопка входу/виходу. Навігаційне меню має надавати можливість швидкого переходу до наступних розділів: галерея, рекомендації, профіль.

Отже, клієнтська частина системи передбачає реалізацію таких сторінок:

- Головна сторінка з навігаційним меню. Сторінка доступна всім користувачам без авторизації.

  Форма входу, де користувач може виконати авторизацію. Форма доступна всім користувачам, які ще не авторизувались на сайті.
- Розділ "Галерея", що дозволяє переглядати, додавати, редагувати та видаляти авторські розробки (роботи). Доступний також пошук робіт за назвою або автором. Розділ доступний авторизованим користувачам.
- Розділ "Рекомендації". Доступний для адміністраторів. У цьому розділі адміністратори можуть керувати правилами рекомендацій, додавати нові правила, редагувати та видаляти існуючі. Для цього

використовуються форми add_rule.php, update_rule.php, delete_rule.php.

Профіль користувача, де відображається інформація про користувача, його роботи та статистика переглядів (user_views).

Інструкція користувача. Відображення детальної інструкції з користування системою. Сторінка доступна всім користувачам, не потребує авторизації.

Всі сторінки повинні мати дизайн в одному стилі. У верхній частині екрану завжди знаходиться верхня навігаційна панель, що дозволяє в будь-який момент переключитись на головне меню або інструкцію користувача. Вхід, додавання та редагування окремих елементів виконується через форми, що відкриваються на новій сторінці.

Головна сторінка знаходитиметься у файлі index.php, форма входу – login.php, для роботи з галереєю – gallery.php, для роботи з рекомендаціями – recommendations.php, для профілю користувача – profile.php. При цьому для додавання та редагування нових робіт, правил рекомендацій використовуватимуться форми з назвами add_work.php, edit_work.php, add_rule.php, update_rule.php, delete_rule.php.

Отже, визначивши структуру клієнтської частини системи, можна перейти до розробки сторінок з використанням мови розмітки HTML.

Для всіх даних, що вводяться користувачем, у системі має бути передбачена можливість валідації. Це необхідно для того, щоб на сервер відправлялись тільки коректні дані. Валідація здійснюється завдяки JavaScript-коду на сторінці форми. Цей код відстежує стан форми та перевіряє дані перед відправленням на сервер.

Наприклад, для форми додавання нового правила рекомендацій можна реалізувати валідацію введених умов та дій, щоб вони відповідали необхідному формату та не містили заборонених символів.

Усі елементи сторінок створюються з використанням стандартних стилів бібліотеки Bootstrap. Також система підтримує динамічне оновлення всіх елементів, що відображаються на екрані. Таблиці оновлюються одразу після

натискання на кнопки додавання, редагування або видалення правил. Перезавантаження всієї сторінки не потрібне. Для цього використовується технологія AJAX.

Наприклад, для завантаження списку правил рекомендацій використовується наступний JavaScript-код:

```
unction loadRecommendationRules() {
    $.ajax({
        url: 'recommendation_rules.php',
        method: 'GET',
        dataType: 'json',
        success: function(data) {
            // Оновлення таблиці правил на сторінці
        }
    });
}
```

3.3 Розробка серверної частини

Серверна частина системи, як було обґрунтовано раніше, розроблена з використанням мови програмування PHP. Ця мова підтримує взаємодію з базами даних MySQL, що також використовується в проєкті. Розробку серверної частини пропонується розпочати зі створення бази даних за допомогою SQL-запитів. Це необхідно для коректної роботи коду, розробленого з використанням PHP. Створення бази даних є достатньо швидким процесом і потребує написання запитів для всіх необхідних таблиць.

Розглянемо код запитів для створення деяких елементів бази даних:

```
CREATE DATABASE AuthorWorksDB;
USE AuthorWorksDB;
CREATE TABLE recommendation_rules (
    id INT AUTO_INCREMENT PRIMARY KEY,
    conditions TEXT NOT NULL,
    action VARCHAR(255) NOT NULL,
```

```
    priority INT NOT NULL DEFAULT 0
);
CREATE TABLE user_views (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    content_id INT NOT NULL,
    viewed_at DATETIME NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (content_id) REFERENCES gallery(id)
);
CREATE TABLE gallery (
    id INT AUTO_INCREMENT PRIMARY KEY,
    author_id INT NOT NULL,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    image VARCHAR(255),
    date DATETIME NOT NULL,
    recommendation VARCHAR(255),
    FOREIGN KEY (author_id) REFERENCES users(id)
);
```

Основна частина серверного програмного забезпечення розроблена з використанням мови програмування PHP. Запити, що надходять зі сторони клієнта, обробляються контролерами, назва яких відповідає назві форми: add_rule.php, update_rule.php, delete_rule.php, user_views.php.

Важливою частиною роботи сервера є забезпечення регулювання прав користувачів та інші заходи захисту. Всі дані, що обробляються сервером, мають бути перевірені, а всі форми захищені від SQL-ін’єкцій та CSRF-атак. Для забезпечення обмежень доступу та захисту від CSRF-атак використовуються сесії.

Наприклад:

```
session_start();
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'Admin') {
    header('Location: access_denied.php');
    exit();
}
```

Для реалізації експертної системи рекомендацій використовується функція, яка оновлює рекомендації на основі правил з таблиці recommendation_rules. Ця функція може бути викликана після додавання або оновлення роботи в галереї, або періодично за розкладом.

Приклад реалізації:

```
function updateRecommendations($conn) {
    // Очищуємо попередні рекомендації
    $conn->query("UPDATE gallery SET recommendation = NULL");

    // Отримуємо всі правила
    $result = $conn->query("SELECT conditions, action FROM recommendation_rules ORDER BY priority DESC");
    if (!$result) {
        error_log("Error fetching rules: " . $conn->error);
        return;
    }

    while ($row = $result->fetch_assoc()) {
        $conditions = $row['conditions'];
        $action = $row['action'];

        // Формуємо запит з умовами
```

```
        $sql = "UPDATE gallery SET recommendation = '$action' WHERE $conditions";

        if (!$conn->query($sql)) {
            error_log("Error updating recommendations: " . $conn->error);
        }
    }
}
```

Таким чином, серверне програмне забезпечення забезпечує динамічне оновлення рекомендацій на основі заданих правил, що дозволяє реалізувати експертну систему рекомендацій в рамках системи розповсюдження авторських розробок.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1 Апаратні і програмні вимоги

Розгортання сучасних веб-додатків вимагає врахування системних вимог як до апаратного, так і до програмного забезпечення. Оскільки розроблена експертна система рекомендацій у системі розповсюдження авторських розробок використовує технології HTML, CSS, JavaScript, Bootstrap, jQuery, AJAX, PHP та MySQL, важливо правильно налаштувати середовище для її функціонування[15].

У даному проєкті використовується стек MAMP, який включає macOS, Apache, MySQL та PHP. Цей набір технологій забезпечує надійний фундамент для розробки динамічних веб-додатків на платформі macOS і відомий своєю простотою та доступністю.

Для ефективного розгортання MAMP-системи слід забезпечити базове апаратне забезпечення. Мінімальні вимоги включають:

- Процесор з тактовою частотою не нижче 1 ГГц, що дозволить обробляти запити з адекватною швидкістю.
- 1 ГБ оперативної пам’яті для підтримки роботи веб-сервера та бази даних. Хоча це може бути обмежуючим фактором для обробки великих обсягів даних, така кількість пам’яті є достатньою для невеликих проєктів.
- Жорсткий диск обсягом не менше 20 ГБ, що дозволить зберігати не тільки систему, але й базу даних, а також додаткові файли проєкту.
- Ці вимоги є мінімальними, і для кращої продуктивності рекомендується використовувати більш потужні конфігурації.

Для розгортання MAMP-системи потрібно попередньо встановити певне програмне забезпечення. MAMP — це готовий пакет для macOS, який містить Apache, MySQL та PHP, що значно спрощує процес налаштування.

Після встановлення MAMP необхідно налаштувати веб-сервер Apache, перевірити коректність роботи PHP та підключення до бази даних MySQL.

Також варто налаштувати віртуальні хости для зручного доступу до розроблюваного додатку.

Для полегшення роботи з базою даних можна використовувати phpMyAdmin, який входить до складу MAMP. Це забезпечить зручний графічний інтерфейс для роботи з базами даних.

Забезпечення належних системних вимог є ключовим етапом у розгортанні веб-додатків на базі MAMP. Виконання всіх вищезгаданих кроків дозволить створити стабільне та продуктивне середовище для роботи вашої системи, що використовує HTML, CSS, JavaScript, Bootstrap, jQuery, AJAX, PHP та MySQL. Кожен з цих компонентів відіграє важливу роль у формуванні якісного продукту, здатного задовольнити потреби користувачів.

Після встановлення MAMP необхідно розгорнути саму систему, створити базу даних і заповнити таблиці, недоступні для редагування через графічний інтерфейс. Пропонується виконати відповідні SQL-запити для першого запуску додатку.

Крім розгортання апаратного та програмного забезпечення, необхідно також приділити достатню увагу інструкціям для користувачів системи. У системі передбачено можливість ознайомлення з інструкцією щодо використання за посиланням у верхній навігаційній панелі.

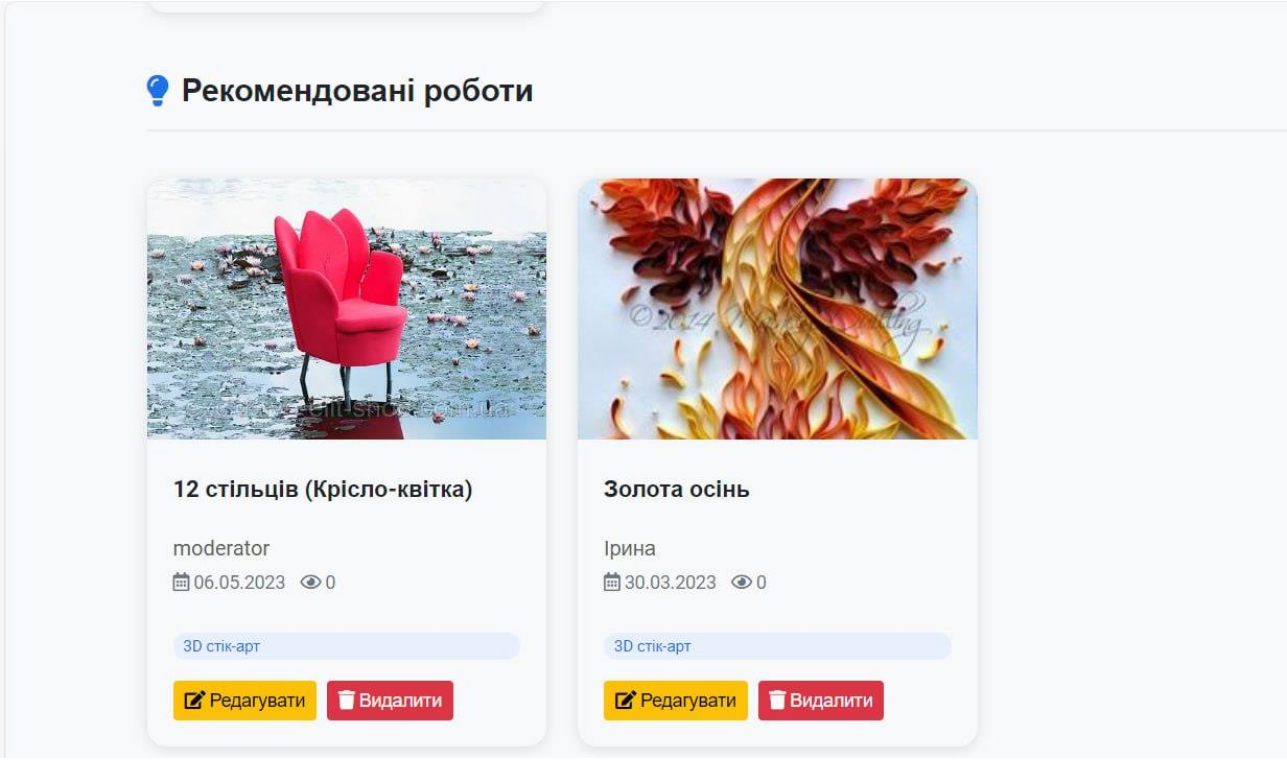

Рис. 4.1 Відображення рекомендацій у розділі "Галерея"

4.2 Тестування розробленої системи

Для тестування програмного забезпечення відомі різні підходи. Одним з найбільш поширених варіантів є тестування методами чорного та білого ящиків. В першому випадку передбачається відсутність знань про те, як влаштований об’єкт, а в другому – про об’єкт відома максимальна кількість інформації. В рамках виконання роботи доцільно застосувати обидва підходи. Також, в процесі розробки було використане модульне тестування, коли перевіряється працездатність кожної складової програми окремо[18].

На завершальному етапі виконання роботи пропонується перевірити працездатність розробленого програмного забезпечення шляхом його запуску та виконання типових робочих операцій, перелічених у попередніх розділах. При цьому необхідно моделювати різні ситуації, що можуть виникати під час роботи системи, та аналізувати отримані результати.

Розпочати тестування пропонується із запуску головної сторінки сайту без проходження авторизації. На сторінці відображається головне меню системи з можливістю навігації. Доступний вибір з розділів: Галерея, Новини, Профіль.

Сторінка відображається коректно, всі посилання працюють, помилки в консолі браузера відсутні. Дизайн сайту адаптивний, під час масштабування зміщення відсутні.

Після авторизації користувача в системі, переходимо до розділу "Галерея", де відображаються авторські розробки. Система рекомендацій аналізує перегляди, лайки та інші параметри робіт, застосовуючи правила з таблиці recommendation_rules, і на основі цього надає рекомендації.

На рисунку 4.1 відображено скріншот розділу "Галерея" з рекомендаціями. Рекомендовані роботи відмічені спеціальними позначками або мають візуальне виділення (наприклад, рамку або бейдж "Рекомендовано"). Це допомагає користувачам швидко знайти найцікавіші та найпопулярніші роботи.

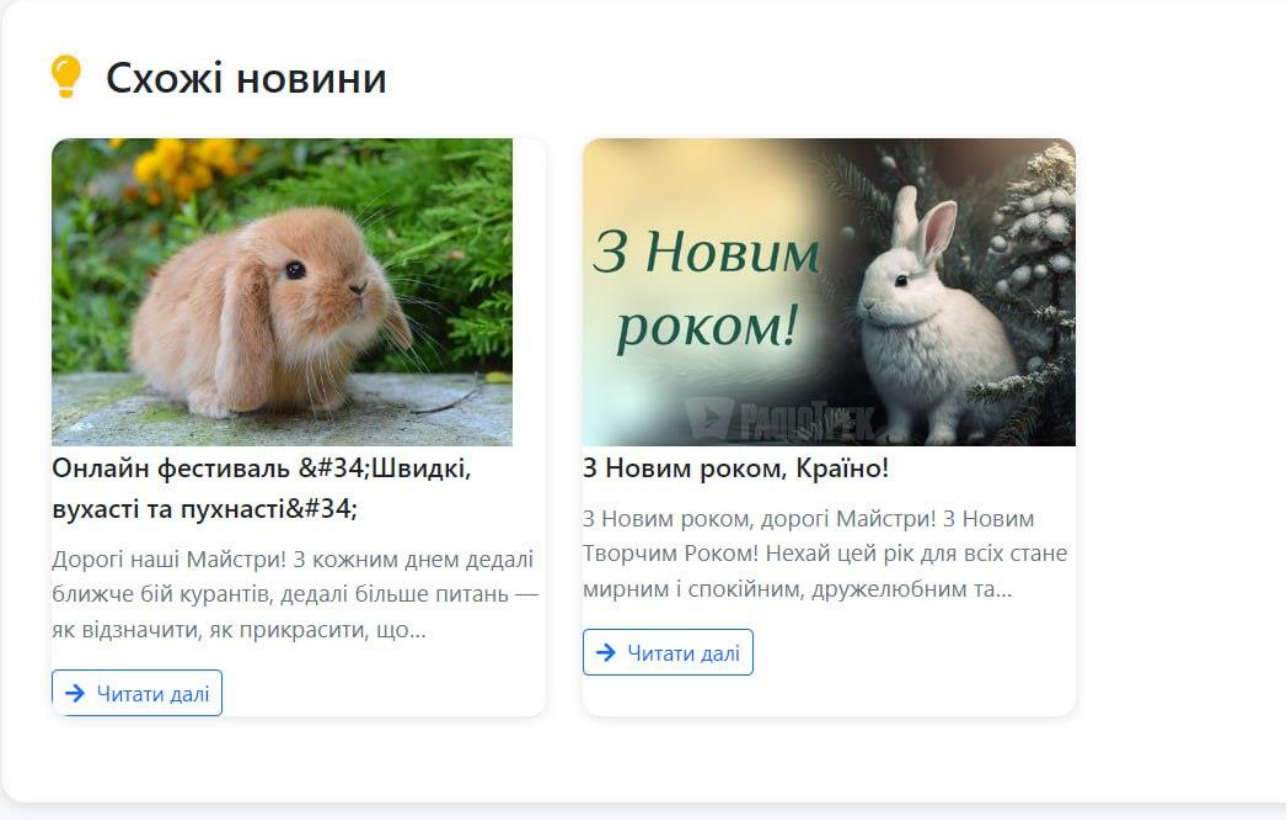

Рисунок 4.2 –Відображення рекомендацій у розділі "Новини"

Далі переходимо до розділу "Новини", де відображаються останні оновлення та події системи. Система рекомендацій також інтегрована в цей розділ, підбираючи новини, що можуть бути цікавими конкретному користувачу, на основі його взаємодії з контентом та правил рекомендацій.

На рисунку 4.2 показано, як відображаються рекомендації у розділі "Новини". Рекомендовані новини можуть бути розміщені у верхній частині сторінки або виділені візуально, щоб привернути увагу користувача.

Перевірка роботи системи рекомендацій

Для перевірки коректності роботи системи рекомендацій необхідно змоделювати дії користувача, наприклад, переглядати певні роботи, ставити лайки, додавати коментарі. Після цього система повинна оновити рекомендації відповідно до налаштованих правил.

Під час тестування було виявлено, що після взаємодії з певними категоріями робіт, у розділі "Галерея" почали відображатися рекомендації, пов'язані з цими категоріями. Це свідчить про коректну роботу системи рекомендацій.

Також у розділі "Новини" відображаються персоналізовані рекомендації, що базуються на інтересах користувача. Це підвищує залученість користувачів та покращує їх досвід взаємодії з системою.

4.3 Аналіз отриманих результатів

Аналізуючи отримані результати роботи, можна зробити висновок про доцільність проведення досліджень експертних систем рекомендацій у різних галузях. На прикладі системи розповсюдження авторських розробок було отримано результати, що можуть підвищити ефективність взаємодії користувачів з контентом за умови впровадження розробленого програмного рішення.

В ході виконання роботи було розроблено клієнт-серверне програмне забезпечення. На сьогодні подібне програмне забезпечення широко застосовується, проте розробка власної системи рекомендацій дозволяє врахувати специфіку конкретного проєкту та потреби його користувачів. Це означає, що дане дослідження має практичну користь і може стати основою для подальшого вивчення застосування експертних систем рекомендацій у різних сферах.

Система складається з клієнта, сервера та системи управління базою даних. Користувачам доступний зручний та інтуїтивно зрозумілий графічний інтерфейс з детальною інструкцією щодо використання. Сайт є адаптивним і працює на будь-яких пристроях.

В результаті тестування програмного забезпечення були досягнуті всі поставлені цілі, ПЗ працює коректно, відповідно до технічного завдання, сформульованого на основі вимог сучасних систем розповсюдження авторських розробок.

Отримані внаслідок роботи системи рекомендації є релевантними та корисними для користувачів. Система виконує поставлені задачі і може бути застосована в якості допоміжного програмного забезпечення для покращення взаємодії з користувачами та збільшення залученості аудиторії.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було проведено комплексне дослідження експертної системи рекомендацій у системі розповсюдження авторських розробок. Розглянуто як теоретичні питання теорії прийняття рішень, так і виконано практичну реалізацію спроектованої системи.

Робота складається з чотирьох структурних розділів. Перший розділ присвячено системному аналізу предметної області досліджень. Було розглянуто поняття рішень, їх прийняття та систем підтримки прийняття рішень, наведено загальні теоретичні відомості. Розглянуто існуючі проблеми в управлінні системами розповсюдження авторських розробок і доведено відсутність ефективних рішень, спрямованих саме на цю галузь. Проведено аналіз літературних джерел, в тому числі наукових публікацій за темою дослідження. В результаті було з’ясовано, що тема систем підтримки прийняття рішень добре досліджена, проте питання впровадження таких систем у галузі розповсюдження авторських розробок недостатньо досліджено і не отримало відображення у науковій літературі. Також було розглянуто існуючі програмні рішення і зроблено висновок про відсутність рішень, які можна було б легко запровадити для управління такими системами. Після проведення аналізу предметної області, було поставлено задачі до розроблюваного програмного забезпечення.

У другому розділі роботи було виконано моделювання системи. Розглянуто питання клієнт-серверної архітектури, описано інформаційні потреби та функції системи. Розроблено UML-діаграми, що відображають структуру системи та взаємодію між її елементами. Найбільшою за обсягом складовою моделювання було моделювання бази даних. Було сформульовано оптимальну структуру бази даних, складено діаграми, що відображають взаємозв’язки між таблицями.

Третій розділ роботи присвячено розробці програмного забезпечення. В першу чергу було запропоновано обґрунтування вибору засобів реалізації ПЗ,

після чого було розглянуто безпосередньо розробку. В якості засобів розробки клієнта було обрано мову розмітки HTML, каскадні таблиці стилів CSS, скриптову мову програмування JavaScript, включаючи технології Bootstrap, jQuery та AJAX. Для сервера було обрано мову програмування PHP, а базу даних створено на базі MySQL. Це оптимальне поєднання для сучасного веб-проєкту[19].

Розробка програмного забезпечення складалася з трьох основних, тісно пов’язаних, частин: розробка клієнтського ПЗ, розробка серверного ПЗ та розробка бази даних. Взаємодія з базою даних здійснюється в межах одного сервера. База даних змодельована у третій нормальній формі[20].

Четвертий розділ роботи включає огляд системних вимог системи до апаратного та програмного забезпечення, надання рекомендацій щодо експлуатації ПЗ, проведення тестування та аналіз отриманих даних.

Розроблена експертна система рекомендацій може бути застосована у системах розповсюдження авторських розробок з метою оптимізації процесів взаємодії з користувачами. Виконана робота може стати основою для подальших наукових досліджень систем підтримки прийняття рішень та рекомендаційних систем.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Рашид А. Принципы проектирования систем рекомендаций / А. Рашид. – Москва: Наука, 2020. – 354 с.
2. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook / F. Ricci, L. Rokach, B. Shapira. – Springer, 2020. – 1034 p.
3. Котлер Ф. Управление маркетингом. Анализ, планирование, внедрение, контроль / Ф. Котлер. – Москва: Прогресс, 2021. – 799 с.
4. Hybrid Recommender Systems for Digital Content [Електронний ресурс] – Режим доступу: https://hybrid-recsys.com
5. Medium Content Recommendation System [Електронний ресурс] – Режим доступу: https://medium.com/recsys
6. Netflix Recommender Systems [Електронний ресурс] – Режим доступу: https://netflix-recsys.com
7. YouTube Recommendation Algorithms [Електронний ресурс] – Режим доступу: https://youtube-recsys.com
8. Вайнштейн А. Введение в обработку данных и машинное обучение / А. Вайнштейн. – Санкт-Петербург: Питер, 2021. – 512 с.
9. Apache Mahout Project [Електронний ресурс] – Режим доступу: https://mahout.apache.org
10. LensKit Open-Source Recommender Framework [Електронний ресурс] – Режим доступу: https://lenskit.org
11. Ricci F. Introduction to Recommender Systems / F. Ricci. – Springer, 2019. – 420 p.
12. Успенский А. Методы анализа данных для систем рекомендаций / А. Успенский. – Москва: Бином, 2019. – 404 с.
13. Петров И. Создание эффективных алгоритмов машинного обучения / И. Петров. – Москва: ДМК Пресс, 2022. – 458 с.

14. TensorFlow Recommenders Library [Електронний ресурс] – Режим доступу: https://www.tensorflow.org/recommenders
15. SAP Commerce Cloud Recommendations [Електронний ресурс] – Режим доступу: https://www.sap.com/products/commerce-cloud.html
16. Adobe Experience Platform [Електронний ресурс] – Режим доступу: https://experience.adobe.com
17. Наумов С. Модели и алгоритмы для систем поддержки принятия решений / С. Наумов. – Киев: Логос, 2020. – 328 с.
18. Шило С. Алгоритмы фильтрации и их реализация / С. Шило. – Харьков: Фолио, 2019. – 392 с.
19. An Introduction to Model-Based Systems Engineering [Електронний ресурс] – Режим доступу: https://insights.sei.cmu.edu
20. Recommender Systems: Advanced Topics / Eds. F. Ricci, L. Rokach. – Springer, 2022. – 650 p.