МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗАСОБАМИ PYTHON ТА VUE.JS»**

Виконала: студентка 4 курсу, групи 6.1210-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
А.С. Кравчук
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Кравчук Аріадні Сергіївні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційної системи засобами Python та Vue.js

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Проєктування та розробка інформаційної системи за допомогою Python та Vue.js.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.01.2024	
2.	Збір вихідних даних.	05.02.2024	
3.	Обробка методичних та теоретичних	04.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	15.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2024	

Студент _____ А.С. Кравчук
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інформаційної системи засобами Python та Vue.js»: 47 с., 20 рис., 26 джерел.

API, FLASK, MONGODB, RECOMMENDATIONS, VUE.

Об’єкт дослідження – процес розробки інформаційної системи.

Мета роботи – створення інформаційної системи, яка забезпечить ефективний збір і структурування новин з різних джерел, а також надання користувачам персоналізованих рекомендацій на основі їхніх інтересів та взаємодій з системою.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

При розробці додатку було визначено функціональні та не функціональні вимоги до інформаційної системи. Обрано та спроєктовано архітектуру системи за допомогою ER-моделі та UML у вигляді наборів діаграм варіантів використання та класів. Реалізовано серверну частину з використанням Python фреймворку Flask, клієнтська частина була розроблена за допомогою фреймворку Vue.js.

В результаті роботи отримано інформаційну систему – агрегатор новин з персональними рекомендаціями.

SUMMARY

Bachelor’s qualifying paper “Development of an Information System Using Python and Vue.js”: 47 pages, 20 figures, 26 references.

API, FLASK, MONGODB, RECOMMENDATIONS, VUE.

The object of the study is the process of developing an information system.

The aim of the study is to create an information system that ensures efficient collection and structuring of news from various sources, as well as providing users with personalized recommendations based on their interests and interactions with the system.

The methods of research are methods of gathering and analyzing software requirements, modeling methods, design, construction, and software testing methods.

During the development of the application, functional and non-functional requirements for the information system were defined. The system architecture was selected and designed using an ER model and UML in the form of use case and class diagrams. The server-side was implemented using the Python framework Flask, and the client-side was developed using the Vue.js framework.

As a result of the work, an information system has been developed – a news aggregator with personalized recommendations.

ЗМІСТ

ВСТУП

У сучасному цифровому світі інформаційні потоки надзвичайно швидко зростають, надаючи користувачам велику кількість контенту для споживання. Зручний доступ до актуальної інформації стає важливою вимогою, що стимулює розробку нових інструментів та технологій.

Ця кваліфікаційна робота присвячена розробці інформаційної системи для агрегації новин з використанням сучасних технологій програмування. Основні технологічні стеки, які використовуються в проєкті, включають Python для реалізації серверної частини системи та Vue.js для розробки ефективної та зручної клієнтської частини.

Метою дослідження є створення інформаційної системи, яка забезпечить ефективний збір і структурування новин з різних джерел, а також надання користувачам персоналізованих рекомендацій на основі їхніх інтересів та взаємодій з системою.

Для досягнення цієї мети потрібно провести наступні кроки:

- оцінка функціональності та технічних рішень, що використовуються в схожих проєктах, для визначення найкращих практик і можливих підходів до розробки;
- визначення основних функціональних та нефункціональних вимог до майбутньої інформаційної системи, які забезпечать її ефективну роботу та високу якість обслуговування користувачів;
- розробка оптимальної архітектури системи та структури бази даних, що враховуватиме потреби у зберіганні, обробці та доступі до інформації;
- реалізація інтерфейсу користувача, який буде зручним у використанні та максимально відповідати потребам аудиторії;
- створення серверної логіки для обробки запитів користувачів, взаємодії з базою даних та реалізації бізнес-логіки системи;

– проведення комплексного тестування розробленої системи з метою виявлення та усунення можливих помилок і недоліків, а також оптимізації продуктивності та швидкодії роботи системи.

Об’єктом дослідження є інформаційна система агрегатора новин з використанням технологій Python та Vue.js.

Звіт складається з трьох розділів, які охоплюють весь процес розробки інформаційної системи агрегатора новин.

У першому розділі визначаються функціональні та нефункціональні вимоги до майбутньої інформаційної системи. Розділ також включає обговорення використовуваних технологій та аналіз типів рекомендаційних систем для побудови оптимального рішення.

Другий розділ присвячений проєктуванню інформаційної системи. В рамках цього розділу розробляються UML діаграми, які моделюють структуру системи, включаючи взаємодії між її складовими частинами, архітектурні рішення та логічну організацію.

Третій розділ охоплює написання і тестування інформаційної системи. В цьому розділі описується процес розробки програмного забезпечення, проведення тестувань для перевірки функціональності та відповідності вимогам, а також демонстрація роботи системи з урахуванням отриманих результатів тестувань.

1 АНАЛІЗ ВИМОГ ТА ІНСТРУМЕНТІВ

1.1 Визначення вимог до інформаційної системи

Аналіз вимог відіграє ключову роль у проєктуванні агрегатора новин з рекомендаціями користувача. Цей розділ спрямований на визначення функціональних та нефункціональних характеристик системи, необхідних для її ефективної роботи та задоволення потреб користувачів. В рамках аналізу буде розглянуто, які функції необхідно реалізувати.

1.1.1 Функціональні вимоги

Функціональні вимоги визначають набір функцій і можливостей, які мають бути реалізовані в агрегаторі новин з рекомендаціями користувача. Вони є основою для функціональності системи, спрямованої на забезпечення зручності використання інтерфейсу користувача, ефективного управління контентом і точності рекомендацій.

Функціональні вимоги до інформаційної системи:

- система має підтримувати механізми аутентифікації користувачів, включаючи реєстрацію нових користувачів та вхід до системи для зареєстрованих користувачів;
- користувачі повинні мати можливість редагувати свої персональні дані, такі як електронна пошта та пароль;
- система повинна збирати дані про переглянуті новини користувачем, ці дані необхідні для створення персоналізованих рекомендацій;
- система має надавати користувачам персоналізовані рекомендації контенту на основі аналізу їхньої активності та інтересів;
- система повинна підтримувати автоматичне збирання новин з RSS стрічок – це включає автоматизоване оновлення контенту на основі

наданих джерел RSS, а також обробку та структуризацію отриманих даних для подальшого аналізу та відображення на інтерфейсі користувача.

1.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні аспекти системи, такі як продуктивність, безпека, надійність та зручність використання, які не пов’язані безпосередньо з функціональністю, але критично важливі для успішної експлуатації та прийнятного досвіду користувача.

Нефункціональні вимоги до інформаційної системи:

– зробити інтерфейс інтуїтивно зрозумілим та легким для навігації, з функціональними кнопками та чітким розміщенням контенту;
– захистити конфіденційні дані користувачів за допомогою сучасних методів шифрування та заходів безпеки;
– забезпечити оптимальну швидкодію завантаження сторінок і обробки запитів користувачів, з метою зменшення часу очікування на мінімум;
– інтерфейс повинен бути адаптивним і зручно відображатися на різних пристроях (комп’ютерах, планшетах, мобільних телефонах) з різними розмірами екранів;
– впевнитися, що агрегатор коректно відображається і працює в усіх популярних веббраузерах (Chrome, Firefox, Safari).

1.2 Основи та застосування Python

Python – це високорівнева інтерпретована мова програмування, яка має кілька значущих переваг. Інтерпретованість без необхідності компіляції дозволяє розробникам швидко розробляти та тестувати програмне

забезпечення. Python не потребує перед запуском програми фази компіляції, що дозволяє миттєво бачити результати роботи, сприяючи швидкому виявленню та виправленню помилок. Цей підхід особливо корисний в умовах швидко змінюваних вимог ринку та коротких циклів розробки [3].

Python використовує динамічну типізацію, що означає, що типи даних змінних визначаються автоматично під час виконання програми. Це спрощує написання коду та роботу з різними типами змінних, що своєю чергою забезпечує більшу гнучкість та швидкість розробки [4].

Однією з ключових переваг Python є його багата стандартна бібліотека, яка включає модулі для роботи з операційною системою, мережевими запитами, управлінням файлами та іншими завданнями. Це забезпечує широкі можливості без необхідності встановлення додаткових пакетів, спрощуючи розробку і забезпечуючи надійність [5].

Python легко інтегрується з кодом на C і C++, що дозволяє використовувати наявні бібліотеки цих мов та оптимізувати продуктивність програм. Це особливо корисно для високопродуктивних застосунків або завдань, що потребують доступу до низькорівневих системних ресурсів [6].

Python також підтримує багатопоточність і асинхронне програмування завдяки модулям threading і asyncio. Це дозволяє розробникам ефективно працювати з паралельними задачами і забезпечувати високу реактивність програм [7, 8].

Python знаходить широке застосування в різних сферах та проєктах завдяки своїй гнучкості, простоті та багатому екосистему інструментів. Ось кілька основних областей, де Python активно використовується:

– Python широко використовується для веброзробки завдяки популярним фреймворкам, таким як Django та Flask, що надають потужні інструменти для створення вебдодатків будь-якої складності [9, 10]; Django використовується для розробки високонавантажених вебсайтів, таких як Instagram та Pinterest, тоді як Flask надає простий підхід для швидкого створення прототипів і невеликих програм;

- Python є однією з основних мов програмування в галузі наукових обчислень та досліджень; бібліотеки, такі як NumPy, SciPy та Pandas, надають потужні інструменти для роботи з чисельними даними, виконання математичних обчислень, аналізу даних та машинного навчання – ці бібліотеки роблять Python незамінним інструментом для вчених та дослідників у багатьох галузях науки [11, 12, 13];
- Python ідеально підходить для написання скриптів та автоматизації завдань – завдяки чистому синтаксису та численним бібліотекам для роботи з файлами, мережевими протоколами та системними ресурсами, Python часто використовується для створення скриптів для адміністрування операційних систем та мережевих пристроїв [14];
- Python став основною мовою для розробки та розгортання моделей машинного навчання та штучного інтелекту – бібліотеки, такі як TensorFlow та PyTorch, надають потужні інструменти для створення та навчання нейронних мереж та інших моделей машинного навчання [15, 16].

Обираючи Python для створення агрегатора новин з рекомендаціями, варто врахувати кілька ключових переваг цієї мови програмування. По-перше, Python дозволяє швидко створювати прототипи та розгортати вебдодатки завдяки фреймворкам, таким як Flask. Ця можливість особливо важлива для швидкого тестування нових ідей та функціональності.

Динамічна типізація Python спрощує роботу з різноманітними типами даних, що є важливим для агрегатора новин, який обробляє великі обсяги інформації з різних джерел. Це забезпечує більшу гнучкість у розробці та дозволяє швидко адаптувати систему до нових вимог і даних.

Python також значно сприяє створенню користувацьких рекомендацій, оскільки надає зручні бібліотеки для роботи з даними та машинного навчання. Бібліотеки, такі як spaCy та scikit-learn, дозволяють легко нормалізувати дані та надавати персоналізовані рекомендації [17, 18]. Це робить Python з

фреймворком Flask ідеальним вибором для розробки системи рекомендацій, що забезпечує високу якість сервісу для користувачів.

1.3 Розробка інтерфейсів з Vue.js

Vue (вимовляється /vjuː/,, так само як і view) – це JavaScript-фреймворк для створення вебінтерфейсів, заснований на стандартних HTML, CSS і JavaScript і використовує декларативну модель програмування на основі компонентів. Це дозволяє ефективно розробляти інтерфейси користувача будь-якої складності [18].

Vue.js дозволяє розділяти додаток на незалежні, повторно використовувані компоненти, що значно спрощує розробку та тестування. Кожен компонент у Vue.js інкапсулює свою логіку, стиль та шаблон, що робить код більш модульним та структурованим. Такий підхід сприяє повторному використанню коду та полегшує роботу з великими проєктами, дозволяючи розробникам концентруватися на окремих частинах додатка, не втручаючись у решту кодової бази [19].

Крім того, Vuex – офіційна бібліотека для управління станом програми у Vue.js, що забезпечує централізоване сховище даних. Вона дозволяє організувати керування станом програми таким чином, щоб усі компоненти мали доступ до одного джерела істини. Vuex використовує концепції односпрямованого потоку даних та реактивності, що робить його ідеальним рішенням для управління станом у великих та складних додатках [22].

Директиви Vue.js являють собою спеціальні атрибути, такі як v-if, v-for і v-bind, які спрощують роботу з динамічним контентом. Наприклад, директива v-if дозволяє умовно відображати елементи, v-for використовується для ітерації масивами, а v-bind допомагає прив’язувати атрибути до даних. Ці директиви роблять шаблони виразнішими та полегшують управління динамічним контентом [20].

Також, Vue Router є офіційним маршрутизатором для Vue.js, що забезпечує легке налаштування маршрутів для односторінкових програм (SPA). Він дозволяє визначати маршрути, асоціювати їх з компонентами та керувати переходами між різними станами програми. Vue Router підтримує динамічні маршрути, вкладені маршрути та захист маршрутів, що робить його потужним інструментом для створення складних та інтерактивних SPA [21].

Використання Vue.js у поєднанні з Flask для агрегату новин з користувальницькими рекомендаціями обґрунтовано декількома ключовими перевагами. По-перше, Vue.js забезпечує високу чуйність інтерфейсу користувача завдяки своїй реактивній природі, що критично важливо для створення інтерактивних елементів.

По-друге, компонентна архітектура Vue.js сприяє зручності розробки та підтримки коду. Поділ інтерфейсу на незалежні компоненти дозволяє легко масштабувати проєкт, перевикористовувати код та впроваджувати зміни без значних витрат часу. Це особливо важливо для складних вебдодатків, таких як агрегати новин, де інтерфейс може включати безліч взаємопов’язаних елементів.

Таким чином, вибір Vue.js для клієнтської частини у поєднанні з Flask для серверної частини забезпечує не тільки високу продуктивність та чуйність, але й зручність розробки, гнучкість розширення функціонала та гарну інтеграцію із серверною частиною програми. Це поєднання дозволяє створювати потужні та зручні у використанні платформи, що відповідають сучасним вимогам користувачів.

1.4 Аналіз типів рекомендаційних систем

Рекомендаційні системи є ключовим елементом сучасних платформ і сервісів, спрямованих на персоналізацію досвіду користувача. Вони використовують різноманітні алгоритми та методи для передбачення переваг

користувачів та рекомендації їм найбільш відповідних контентних пропозицій.

1.4.1 Контентна фільтрація

Фільтрація на основі контенту (Content-Based Filtering) – це один із двох основних типів рекомендаційних систем. Вона рекомендує користувачеві елементи на основі їх індивідуальних характеристик. У цьому методі використовуються властивості елементів для вибору і повернення елементів, що відповідають запиту користувача. Часто враховуються характеристики інших елементів, які цікавлять користувача [23].

В рекомендаційних системах ключовими аспектами є профілі елементів та користувачів, які дозволяють персоналізувати рекомендації відповідно до індивідуальних вподобань і поведінки користувачів.

Профіль елемента (Item Profile) визначає представлення кожного елемента в системі. Він включає широкий набір характеристик, таких як жанр, дата випуску, режисер, та інші параметри, що описують властивості елемента.

Профіль користувача (User Profile) відображає вподобання та поведінку кожного користувача. Він містить інформацію про ті елементи, які вже цікавили користувача раніше, а також дані про взаємодію з системою, такі як лайки, дизлайки, оцінки та запити.

Ці два типи профілів є основою для інтелектуального аналізу і рекомендацій, що дозволяють системам підтримувати персоналізований підхід до кожного користувача.

1.4.2 Колаборативна фільтрація

Колаборативна фільтрація (Collaborative Filtering) є методом рекомендаційних систем, який використовує схожість між користувачами та

елементами одночасно для надання персоналізованих рекомендацій. Цей підхід дозволяє системі пропонувати несподівані варіанти, наприклад рекомендувати користувачеві елементи на основі інтересів схожого користувача [24, 25].

Основними перевагами колаборативної фільтрації є автоматичне вивчення вкладень (embeddings) без необхідності ручної інженерії ознак. Ці вкладення є числові уявлення користувачів та елементів, які модель використовує для прогнозування оцінок чи переваг.

Коли користувач заходить на головну сторінку системи, колаборативна фільтрація розглядає такі аспекти:

– подібність до елементів, які користувач раніше позитивно оцінив;
– уподобання схожих користувачів та їх взаємодії з аналогічними елементами.

Використовуючи вкладення користувачів та елементів, модель робить передбачення про те, які елементи користувач, швидше за все, оцінить позитивно. Наприклад, якщо користувач часто позитивно оцінює певний тип елементів і ці елементи також мають високі оцінки у подібних користувачів, їх вкладення будуть відповідним чином налаштовані.

1.4.3 Вибір типу рекомендаційної системи

Для створення рекомендацій в агрегатору новин було вибрано метод content-based рекомендацій з наступних причин:

– адаптація до обмеженого обсягу даних: в агрегаторі новин зазвичай обмежений обсяг даних, таких як історія переглядів або оцінок – метод content-based дозволяє будувати рекомендації на основі аналізу самих новинних статей та їх змісту, минаючи необхідність у великих обсягах даних користувача;
– прозорість та інтерпретація: content-based підходи часто більш

прозорі та інтерпретовані для користувача – рекомендації можуть бути пояснені через зміст статей та ключові характеристики, що сприяє більшій довірі з боку користувачів;

– найменша залежність від «холодного старту»: content-based методи починають пропонувати рекомендації з першого використання, аналізуючи зміст статей, на які реагує користувач – це дозволяє швидше адаптуватися до інтересів нових користувачів і покращує їх досвід взаємодії з системою.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 ER-модель бази даних

Діаграма зв’язків сутностей (ERD), також відома як ER-діаграма або модель ER, є структурною діаграмою, призначеною для проєктування баз даних. Вона використовує різні символи та з’єднувачі для візуалізації двох основних аспектів: основних сутностей, які належать до області системи, та взаємозв’язків між ними [1].

ER-діаграми широко використовуються на етапах аналізу та проєктування баз даних. Вони дозволяють зрозуміти структуру даних, забезпечують чітке уявлення про взаємозв’язки між різними частинами системи та допомагають виявити можливі недоліки або неузгодженості на ранніх етапах розробки. На рисунку 2.1 наведено ER-модель для інформаційної системи, ще розробляється.

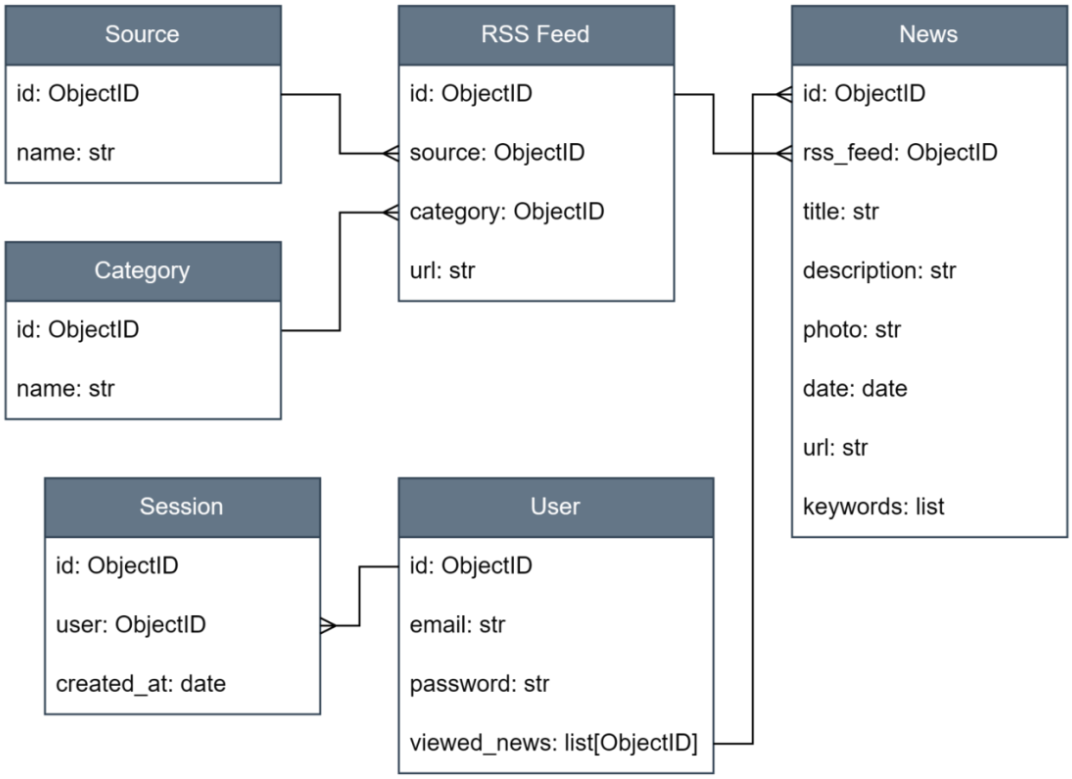

Рисунок 2.1 – ER-модель для агрегатора новин

2.1.1 Сутності моделі

Сутність Category.

Призначення: сутність Category визначає категорію або тематичний розділ для джерел новин. Її головною задачею є організація і класифікація різноманітних джерел інформації за спільними ознаками. Категорії дозволяють користувачам структурувати свій контент і швидко знаходити новини за відповідною темою чи інтересами.

Атрибути:

– name: зберігає назву категорії джерел новин – це обов’язкове поле, яке є унікальним для кожної категорії і дозволяє однозначно ідентифікувати тематичний розділ.

Сутність Source.

Призначення: сутність Source являє собою конкретне джерело новин, таке як видання, портал або медіакомпанія, яке публікує інформаційний контент. Вона використовується для ідентифікації джерел, з яких отримані новини.

Атрибути:

– name: назва джерела новин – це обов’язкове унікальне поле, яке ідентифікує конкретне джерело, і використовується для посилання на нього в інших частинах системи.

Сутність RSS Feed.

Призначення: сутність RssFeed представляє RSS-стрічку, яка містить новини відповідного джерела. Вона використовується для автоматизованого отримання та обробки інформаційних матеріалів з медіаресурсів.

Атрибути:

– source: посилання на джерело новин через об’єкт Source – це поле визначає конкретне джерело, з якого отримані дані;

– category: посилання на категорію джерел новин через об’єкт Category – це поле допомагає класифікувати новини за тематичними ознаками;

– url: URL RSS-стрічки – це унікальне поле, яке вказує на місцеперебування самої стрічки в мережі Інтернет.

Сутність User.

Призначення: сутність User представляє користувача системи, який має можливість взаємодіяти з новинами та іншими функціями платформи.

Атрибути:

– email: адреса електронної пошти користувача – це обов'язкове унікальне поле, яке використовується для ідентифікації користувача в системі;
– password: пароль користувача – це обов'язкове поле, що забезпечує конфіденційність даних користувача;
– viewed_news: список новин, які переглянув користувач – це поле містить посилання на об'єкти новин, які були переглянуті користувачем і використовується для персоналізації вмісту і взаємодії з платформою.

Сутність News.

Призначення: сутність News представляє інформацію про конкретну новину, яка була отримана з RSS-стрічки відповідного джерела.

Атрибути:

– rss_feed: посилання на об'єкт RssFeed – це поле визначає, з якої RSS-стрічки була отримана ця новина;
– title: заголовок новини – це обов'язкове поле, яке містить заголовок або назву новини;
– descriptions: опис новини – це обов'язкове поле, яке містить короткий текстовий опис події чи новини;
– photo: посилання на фотографію, пов'язану з новиною – це поле може містити URL або шлях до зображення;
– date: дата публікації новини – це обов'язкове поле, яке вказує на дату і час, коли новина була опублікована;
– url: URL новини – це унікальне поле, яке вказує на посилання на саму

новину в мережі Інтернет;

– keywords: список ключових слів, отриманих новиною – це поле містить ключові слова або терміни, які характеризують основні аспекти новини.

2.1.2 Зв’язки між сутностями

Зв’язок між RssFeed і Category.

Тип зв’язку: кожен об’єкт RssFeed асоціюється з однією конкретною категорією через поле category.

Значення зв’язку: це забезпечує можливість класифікації кожної RSS-стрічки за певною темою або категорією, що сприяє кращій організації та структуризації інформаційного контенту.

Зв’язок між RssFeed і Source.

Тип зв’язку: кожна RSS-стрічка пов’язана з конкретним джерелом новин через поле source.

Значення зв’язку: цей зв’язок вказує на те, з якого саме джерела або видання була отримана інформація, яка потім попадає в RSS-стрічку.

Зв’язок між News і RssFeed.

Тип зв’язку: кожна новина посилається на конкретну RSS-стрічку через поле rss_feed.

Значення зв’язку: це поле визначає, з якої саме RSS-стрічки була отримана інформація про цю новину.

Зв’язок між Session і User.

Тип зв’язку: кожен об’єкт Session (сесія) посилається на конкретного користувача (об’єкт User) через поле user_id.

Значення зв’язку: цей зв’язок визначає, який саме користувач виконав певну дію в системі (наприклад, переглядав новини або виконував інші дії).

Зв’язок між User і News через viewed_news.

Тип зв’язку: в полі viewed_news об’єкта User зберігаються посилання на новини (об’єкти News), які користувач переглянув.

Значення зв’язку: цей зв’язок використовується для відстеження історії перегляду користувача та підтримки функцій персоналізації та рекомендацій.

2.2 Діаграма класів

Діаграма класів UML є графічною нотацією, яка використовується для побудови та візуалізації об’єктноорієнтованих систем. Це тип статичної діаграми структури, яка описує структуру системи, показуючи такі складові:

– класи;
– їх атрибути;
– операції (або методи);
– взаємозв’язки між об’єктами [2].

Клас є схемою для об’єкта. Об’єкти та класи нерозривно пов’язані між собою. Не можна говорити про одне без згадки про інше. І суть об’єктноорієнтованого проєктування не в об’єктах, а в класах, оскільки саме класи використовуються для створення об’єктів. Таким чином, клас описує, яким буде об’єкт, але не є самим об’єктом.

Фактично, класи описують типи об’єктів, тоді як об’єкти є конкретними екземплярами класів. Кожен об’єкт побудований за однією і тією ж схемою (або через одні та ті самі схеми), тому містять однакові компоненти (властивості та методи). Основне значення полягає в тому, що об’єкт є екземпляром класу, і об’єкти мають стани та поведінки.

Клас представляє концепцію, яка інкапсулює стан (атрибути) та поведінку (операції). Кожен атрибут має тип, кожна операція має сигнатуру.

На рисунку 2.2 показана діаграма класів для агрегатора новин, яка включає класи, їх атрибути та операції.

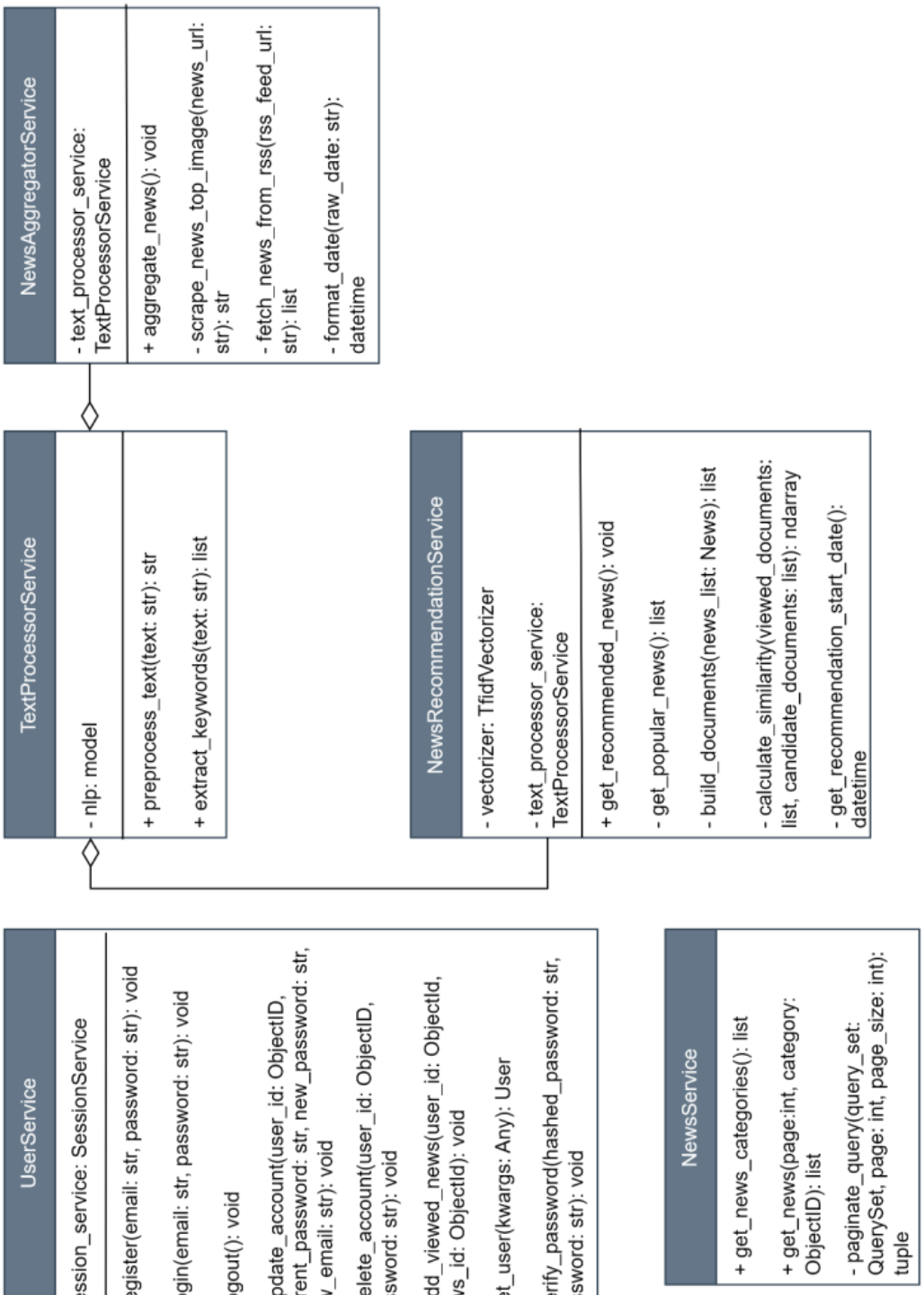

Рисунок 2.2 – Діаграма класів для агрегатора новин

Клас UserService.

Опис: клас UserService управляє аутентифікацією користувачів та обробкою їхніх даних у додатку. Він відповідає за реєстрацію нових користувачів, вхід та вихід з облікового запису, оновлення даних акаунта, видалення акаунта, а також за ведення журналу новин, які користувач переглядав.

Поля:

– session_service: об’єкт класу SessionService, який керує сесіями користувачів.

Методи:

– register(self, email: str, password: str): створює новий обліковий запис користувача;
– login(self, email: str, password: str): аутентифікує користувача за допомогою email та пароля;
– logout(self): завершує поточну сесію користувача;
– update_account(self, user_id: ObjectId, current_password: str, new_password: str = None, new_email: str = None): оновлює пароль та/або email користувача;
– delete_account(self, user_id: ObjectId, password: str): видаляє обліковий запис користувача;
– add_viewed_news(self, user_id: ObjectId, news_id: ObjectId): додає новину до списку переглянутих користувачем та збільшує лічильник переглядів новини;
– _get_user(self, **kwargs): отримує користувача за вказаними ключовими аргументами;
– _verify_password(self, user_id: ObjectId, hashed_password: str, password: str): перевіряє відповідність введеного пароля та збереженого хешованого пароля.

Клас SessionService.

Опис: клас SessionService представляє сервіс для управління сесіями користувачів за допомогою Flask-сесій та MongoDB. Він відповідає за

створення, отримання, видалення поточних сесій користувачів, а також видалення прострочених сесій.

Методи:

- get_current_session(self) -> Optional[Session]: отримує поточну сесію користувача;
- create_session(self, user_id: ObjectId): створює сесію для користувача;
- delete_current_session(self): видаляє поточну сесію користувача;
- delete_expired_sessions(self): видаляє з бази даних сесії, які перевищили максимальний термін життя.

Клас NewsService.

Опис: клас NewsService є сервісом для обробки операцій, пов’язаних з новинами. Він забезпечує методи для отримання категорій новин, пошуку новин з можливістю фільтрації за категоріями, а також пагінації результатів пошуку.

Методи:

- get_news_categories(self) -> list: отримує всі категорії новин з бази даних;
- get_news(self, page: int, category: ObjectId = None) -> list: отримує новини з можливістю фільтрації за категоріями;
- _paginate_query(self, query_set: QuerySet, page: int, page_size: int) -> tuple: виконує пагінацію заданого набору результатів запиту.

Клас NewsAggregatorService.

Опис: клас NewsAggregatorService є сервісом для агрегування новинних статей з різних RSS-стрічок, їх обробки для отримання релевантної інформації та збереження в базі даних. Він забезпечує отримання новин, обробку тексту та збереження оброблених новин.

Поля:

- text_processor_service: об’єкт класу TextProcessorService, який використовується для обробки тексту новин і витягування ключових слів.

Методи:

- aggregate_news(self): агрегує новини з усіх джерел і зберігає їх у базі даних;
- _scrape_news_top_image(self, news_url: str) -> str: отримує URL основного зображення новинної статті шляхом скрапінгу;
- _fetch_news_from_rss(self, rss_feed_url: str) -> list: отримує і повертає записи новин з RSS-стрічки за вказаним URL;
- _format_date(self, raw_date: str) -> datetime: конвертує строку з датою в об’єкт datetime у часовому поясі UTC.

Клас NewsRecommendationService.

Опис: клас NewsRecommendationService надає персоналізовані рекомендації новин користувачам на основі їхньої історії переглядів. Він використовує методи обробки тексту для аналізу новин і обчислення схожості між переглянутими новинами та потенційними новинами для рекомендацій.

Поля:

- vectorizer: об’єкт TfidfVectorizer, який використовується для перетворення текстів у TF-IDF матрицю;
- text_processor_service: об’єкт класу TextProcessorService, який використовується для обробки текстів новин.

Методи:

- get_recommended_news(self, user_id: ObjectId) -> dict: отримує рекомендовані новини для користувача на основі його історії переглядів;
- _build_documents(self, news_list: News) -> list: створює документи для TF-IDF векторизації;
- _get_recommendation_start_date(self) -> datetime: обчислює початкову дату для рекомендацій новин на основі конфігурації;
- _calculate_similarity(self, viewed_documents: list, candidate_documents: list) -> ndarray: обчислює схожість між переглянутими документами й документами кандидатів;

– _get_popular_news(self) -> dict: отримує популярні новини на основі кількості переглядів.

Клас TextProcessorService.

Опис: клас TextProcessorService містить методи для обробки тексту та вилучення ключових слів. Використовуючи бібліотеку SpaCy для української мови, цей клас забезпечує функціональність для попередньої обробки тексту (лемматизації, видалення стоп-слів і пунктуації) та вилучення ключових слів з тексту.

Поля:

– nlp: об’єкт SpaCy, завантажений з моделлю для української мови uk_core_news_md, який використовується для обробки тексту.

Методи:

– preprocess_text(self, text: str) -> str: попередньо обробляє вхідний текст;
– extract_keywords(self, text: str) -> list: вилучає ключові слова з вхідного тексту.

2.3 Діаграма прецедентів

Діаграма використання UML є важливим інструментом для моделювання вимог до програмного забезпечення, оскільки вона дозволяє чітко визначити очікувану поведінку системи з погляду користувача. Використання в цьому контексті описують, що система повинна робити (що), а не як саме це повинно бути реалізовано (як) [26]. Це спрощує розуміння функціональних вимог та сприяє ефективній комунікації між усіма учасниками проєкту.

Одним із ключових аспектів діаграм використання є їхнє спрямування на кінцевого користувача. Вони дозволяють інженерам та аналітикам проєктувати систему з погляду того, як її буде використовувати користувач.

Це надає можливість уникати технічних деталей і фокусуватися на зовнішньо важливих аспектах функціональності.

Наприклад, діаграма використання може показати, як користувач взаємодіє з системою у вигляді послідовності дій, не вдаючись до деталей внутрішньої реалізації. Вона ідентифікує акторів (таких як користувачі чи інші системи), використання (функціональні можливості системи) та зв'язки між ними. Такий підхід дозволяє чітко визначити, які функціональність мають бути включені в систему, і як ці елементи взаємодіють між собою.

На рисунку 2.3 показана діаграма прецедентів для агрегатора новин, яка включає взаємодію користувачів з системою та функціональні можливості, які система повинна надавати.

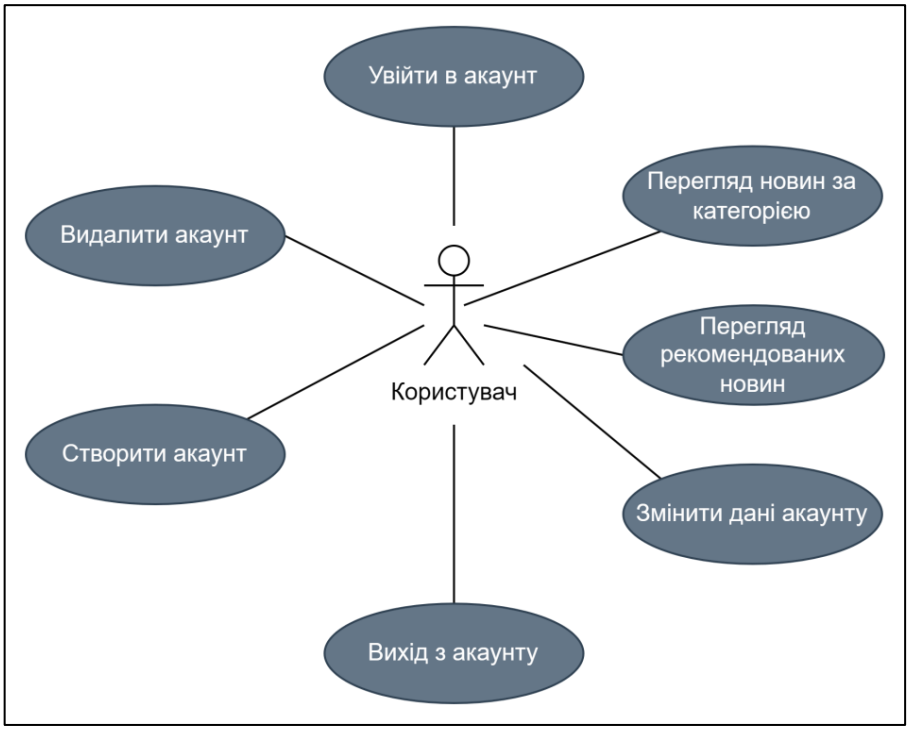

Рисунок 2.3 – Діаграма прецедентів для агрегатора новин

Прецедент «Створення акаунту».

Призначення: дозволяє користувачам створити особистий акаунт в агрегаторі новин.

Основний потік подій: користувач відкриває сторінку реєстрації. Вводить обов'язкову інформацію: електронну адресу та пароль. Натискає

кнопку «Зареєструватися». Система перевіряє введені дані та створює новий акаунт. Система перевіряє введені дані та створює новий акаунт.

Виняткова ситуація 1: якщо введені дані недійсні, система показує повідомлення про помилку та дозволяє користувачеві ввести дані ще раз.

Виняткова ситуація 2: якщо користувач вже зареєстрований з введеною електронною адресою, система відображає повідомлення про наявність користувача з такою email і дозволяє користувачеві змінити свої дані.

Прецедент «Видалення акаунту».

Призначення: дозволяє користувачам видалити свій акаунт з системи.

Основний потік подій: користувач увійшов у свій акаунт. Відкриває сторінку налаштувань акаунта. Введе старий пароль Обирає опцію «Видалити акаунт». Система видаляє всі дані користувача та деактивує акаунт.

Виняткова ситуація: якщо введені дані недійсні, система показує повідомлення про помилку та дозволяє користувачеві ввести дані ще раз.

Прецедент «Вхід в акаунту».

Призначення: дозволяє зареєстрованим користувачам увійти у свій особистий акаунт.

Основний потік подій: користувач відкриває сторінку входу. Вводить свою електронну адресу та пароль.

Виняткова ситуація 1: якщо введені дані некоректні, система відображає повідомлення про помилку і дає відвідувачу можливість повторно ввести дані.

Виняткова ситуація 2: якщо користувача з введеною електронною адресою не знайдено в системі, відображається повідомлення про відсутність користувача з таким email, і відвідувач може внести зміни у введених даних.

Прецедент «Вихід з акаунту».

Призначення: дозволяє користувачам вийти зі свого особистого акаунту.

Основний потік подій: користувач увійшов у свій акаунт. Обирає опцію «Вийти» або «Вийти з акаунту». Система завершує сеанс і перекидає користувача на головну сторінку.

Прецедент «Зміна даних акаунту».

Призначення: дозволяє користувачам змінювати особисті дані, пов’язані з їхнім акаунтом. Такі як пароль та електрона пошта.

Основний потік подій 1: користувач увійшов у свій акаунт та відкрив сторінку налаштувань акаунта. Введе старий пароль і новий пароль. Натисне кнопку «Зберегти зміни». Система підтвердить успішну зміну пароля.

Основний потік подій 2: користувач увійшов у свій акаунт та відкрив сторінку налаштувань акаунта. Введе нову електронну адресу та старий пароль. Натисне кнопку «Зберегти зміни». Система підтвердить успішну зміну електронної пошти.

Основний потік подій 3: користувач увійшов у свій акаунт та відкрив сторінку налаштувань акаунта. Введе старий пароль, новий пароль і нову електронну адресу. Натисне кнопку «Зберегти зміни». Система підтвердить успішну зміну і пароля, і електронної пошти.

Виняткова ситуація: якщо введені дані некоректні, система відображає повідомлення про помилку і дає відвідувачу можливість повторно ввести дані.

Прецедент «Перегляд новин за категорією».

Призначення: дозволяє користувачам переглядати новини, які відповідають певній категорії.

Основний потік подій: користувач увійшов у свій акаунт або відкрив сторінку новин. Вибирає категорію новин (наприклад, спорт, політика, наука). Система відображає список новин, що відповідають обраній категорії.

Виняткова ситуація: якщо для обраної категорії відсутні новини або сталася помилка завантаження, система повідомляє користувача про це і пропонує спробувати знову пізніше.

Прецедент «Перегляд рекомендованих новин».

Призначення: дозволяє користувачам переглядати персоналізовані рекомендації новин на основі їхніх інтересів або попередніх переглядів.

Основний потік подій: користувач увійшов у свій акаунт або відкрив сторінку рекомендацій. Система аналізує інформацію про користувача (історія

перегляду). Відображає список рекомендованих новин на основі зібраної інформації.

Виняткова ситуація: якщо система не може згенерувати рекомендації через технічні проблеми або відсутність достатньої інформації про користувача, вона повідомляє про це та рекомендує переглянути загальний список новин.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Розробка програми

3.1.1 Реалізація клієнтської частини

Для реалізації запланованої функціональності агрегатора новин було вирішено створити такі сторінки: Вхід, Реєстрація, Головна, Рекомендації та Профіль.

На сторінках Входу та Реєстрації розташовані форми для введення пошти та пароля (див. рис. 3.1). На головній сторінці всі користувачі, незалежно від авторизації, можуть переглядати новини за категоріями (див. рис. 3.2). Сторінка Рекомендацій доступна лише зареєстрованим користувачам; вони можуть бачити рекомендовані новини, які оновлюються при кожному відкритті сторінки, щоб зберігати інтерес користувачів.

Сторінка профілю призначена для керування обліковим записом користувача (див. рис. 3.3). Тут користувач може видалити обліковий запис або змінити дані, такі як пароль та електронна пошта.

Рисунок 3.1 – Сторінка входу в акаунт

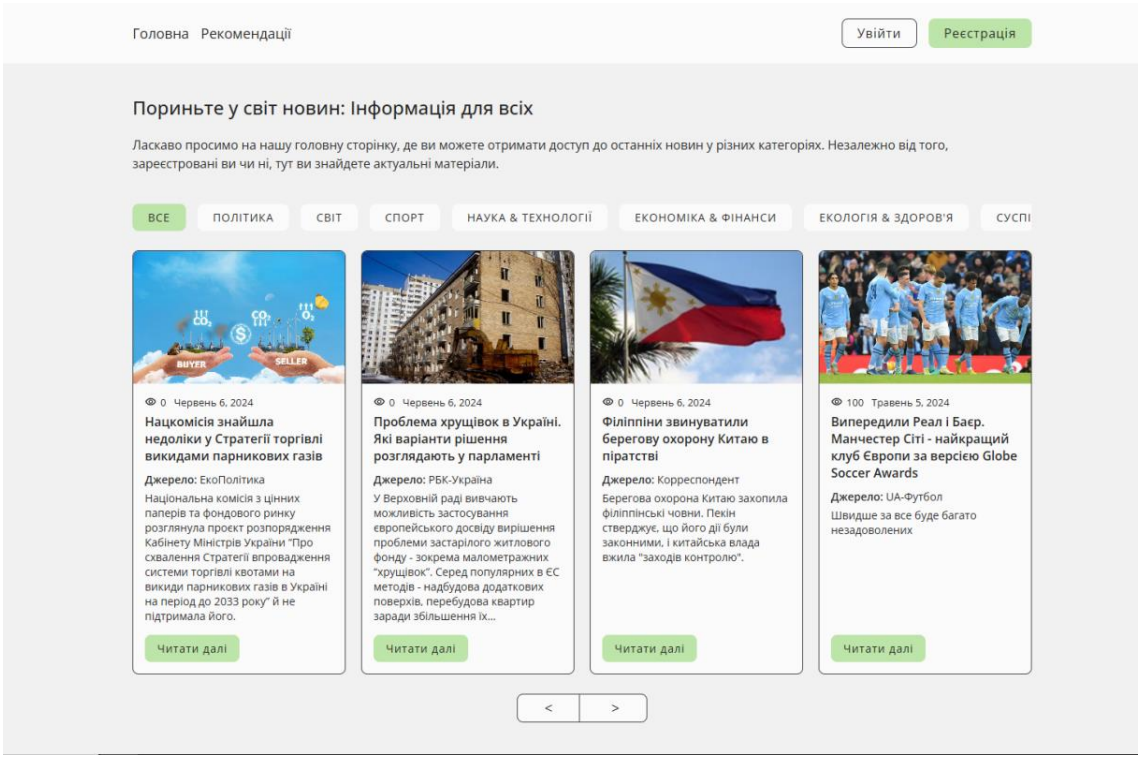

Рисунок 3.2 – Головна сторінка

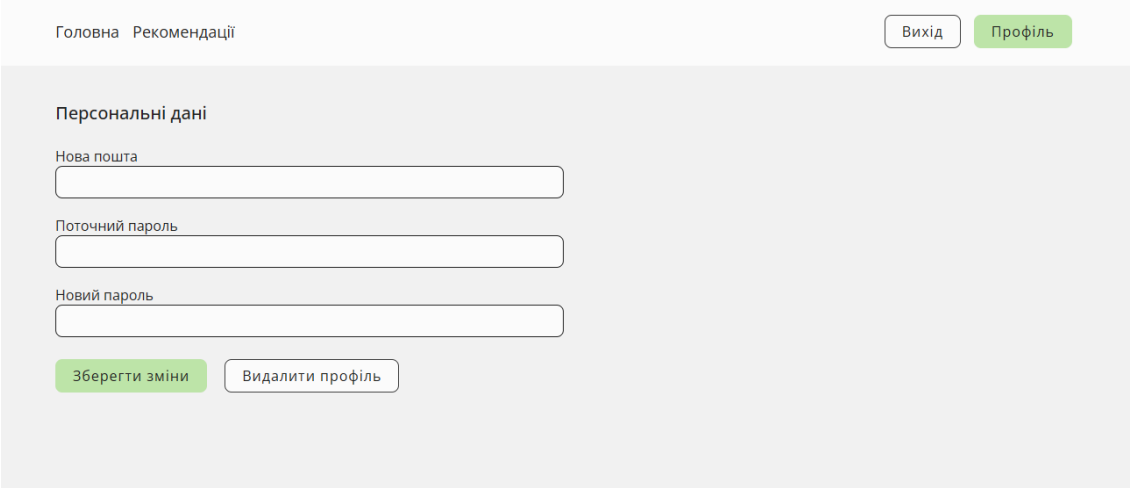

Рисунок 3.3 – Сторінка профілю

3.1.2 Реалізація серверної частини

На рисунку 3.4 показана рекомендована структура проєкту API на Flask. Вона включає деякі теки, розглянемо їх та їх призначення.

Тека **models**: цей каталог призначений для зберігання класів моделей бази даних. Він служить основою для опису структури даних, яка

використовується в програмі. Кожна модель відповідає певній таблиці в базі даних або сутності, що використовується в додатку.

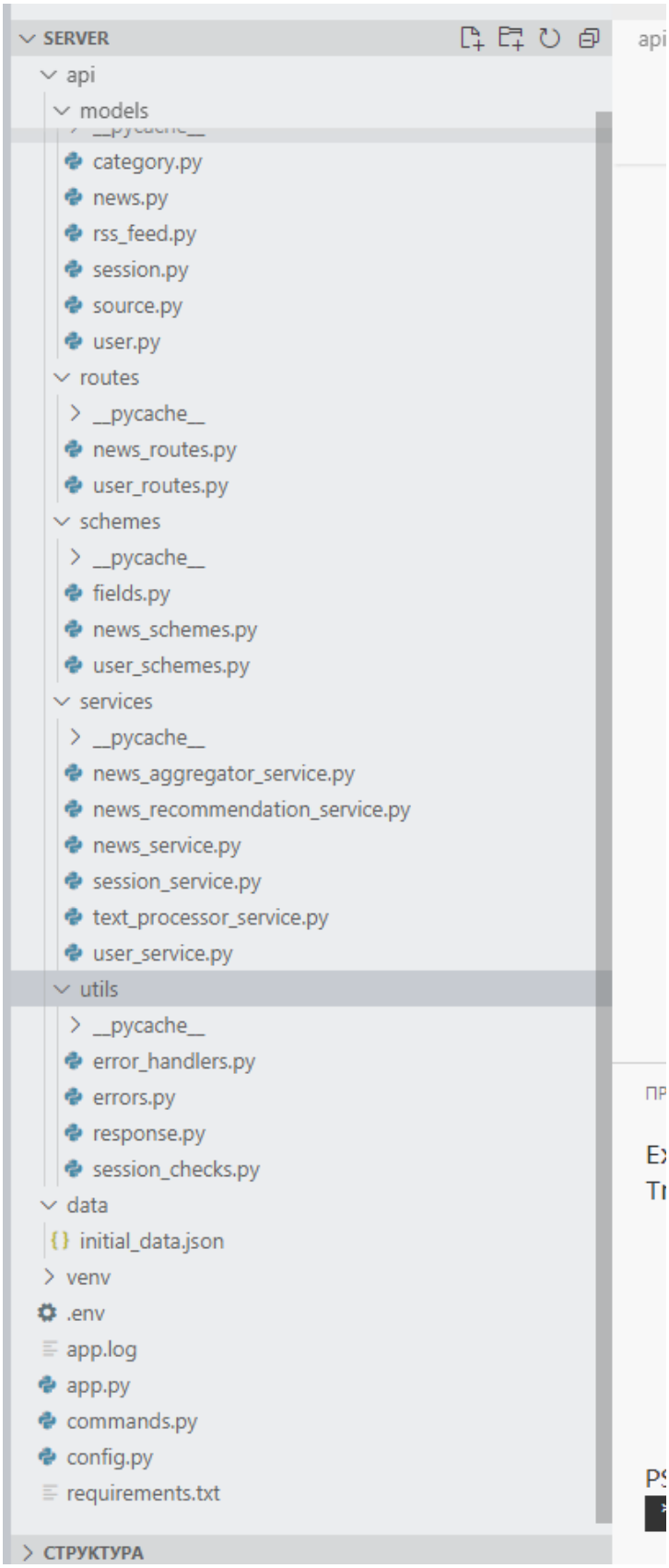

Рисунок 3.4 – Структура проєкту

Тека **schemes**: у цій директорії розміщуються валідаційні класи Pydantic. Вони відповідають за опис структури та валідацію вхідних даних до API. Це важливий етап для забезпечення коректної обробки вхідної інформації та уникнення помилок.

Тека **routes**: тут зберігаються файли, які визначають маршрути API. Вони відповідають за обробку HTTP-запитів і визначають, які дії потрібно виконати при отриманні певних запитів до сервера. Розділення цієї логіки дозволяє зберігати код організовано і легко змінювати.

Тека **services**: цей каталог призначений для зберігання класів сервісів. Вони відокремлюють бізнес-логіку від маршрутів і моделей бази даних, що робить систему більш модульною і підготовленою до подальшого розширення. Сервіси виконують проміжну роль між рівнями додатка, що сприяє чіткій структурі та полегшує тестування.

Тека **utils**: у цій директорії містяться допоміжні функції та класи. В реалізованому проєкті тут зберігаються декоратори, обробники помилок Flask і інші допоміжні інструменти, такі як методи для стандартизації відповідей API. Наприклад, метод generate_response може використовуватися для забезпечення однакової структури відповіді для всіх запитів, що спрощує роботу клієнтського коду.

Така організація структури дозволяє підтримувати чистоту коду, легко знаходити необхідний фрагмент програми та швидко впроваджувати зміни без необхідності переписування великої частини проєкту.

Для взаємодії з базою даних MongoDB найзручніше використовувати бібліотеку mongoengine. Ця бібліотека дозволяє визначати моделі баз даних як звичайні класи Python, що значно спрощує процес роботи з даними. Замість того, щоб писати запити мовою запитів бази даних, ми можемо оперувати цими моделями як об’єктами, що абстрагує нас від деталей реалізації запитів та дозволяє працювати на чистому Python.

На рисунку 3.5 зображено приклад використання Object-Document Mapper (ODM) для моделі новин.

```python
class News(Document):
    """A Document schema representing a news article."""
    rss_feed = ReferenceField(RssFeed, required=True, reverse_delete_rule=DENY)
    title = StringField(required=True, min_length=10, max_length=2000)
    description = StringField(required=True, min_length=20, max_length=2000)
    photo = StringField(required=True, min_length=1)
    date = DateTimeField(required=True)
    url = StringField(required=True, unique=True, min_length=1)
    keywords = ListField(
        StringField(min_length=3, max_length=50), required=True
    )
    view_count = IntField(default=0)
```

Рисунок 3.5 – Приклад використання ODM для моделі новин

Крім того, моделі mongoengine можуть виступати в ролі валідаторів даних, що є ще одним плюсом використання цієї бібліотеки. Проте слід зазначити, що бібліотека mongoengine може мати певні обмеження, особливо у роботі з вкладеними структурами даних. Насправді було помічено, що вона іноді відчуває труднощі з валідацією та обробкою таких структур.

Для реалізації валідації вхідних даних користувачів у проєкті було вирішено використати бібліотеку pydantic. Рисунок 3.6 ілюструє приклад використання класу з бібліотеки pydantic для валідації вхідних даних у проєкті. Цей підхід дозволяє забезпечити надійність і коректність даних, що надходять в систему.

```python
class UserUpdate(BaseModel):
    """
    Model for updating account details. Requires at least one field to be
    updated.
    """

    current_password: PasswordField
    new_password: Optional[PasswordField] = None
    new_email: Optional[EmailField] = None

    @model_validator(mode="before")
    def check_at_least_one(cls, values):
        """Validator to ensure updates include new information"""
        if not values.get("new_password") and not values.get("new_email"):
            raise ValueError(
                "Either new_password or new_email must be provided."
            )
        return values
```

Рисунок 3.6 – Приклад класу pydantic для валідації вхідних даних

Ця бібліотека заслужила широке визнання у спільноті Python завдяки своїй простоті та потужним можливостям. На відміну від інших аналогів, таких як cerberus та marshmallow, pydantic пропонує більш інтуїтивно зрозумілий інтерфейс для взаємодії, що спрощує процес створення та використання користувацьких типів даних для валідації.

На рисунку 3.7 показано приклад створення користувацьких типів даних для валідації за допомогою бібліотеки pydantic. Цей підхід дозволяє гнучко налаштовувати правила валідації та використовувати їх для різних частин системи з мінімальними зусиллями.

```python
# This pattern ensures the password contains a combination of digits,
# uppercase and lowercase letters.
PASSWORD_PATTERN = (
    r"([0-9]+.*[A-Z]+.*[a-z]+)"
    r"|([A-Z]+.*[a-z]+.*[0-9]+)"
    r"|([a-z]+.*[0-9]+.*[A-Z]+)"
    r"|([0-9]+.*[a-z]+.*[A-Z]+)"
    r"|([A-Z]+.*[0-9]+.*[a-z]+)"
    r"|([a-z]+.*[A-Z]+.*[0-9]+)"
)

# This pattern restricts the email to certain domains, ensuring it's a valid
# email format.
EMAIL_PATTERN = (
    r"^[\w\.-]+@(gmail\.com|yahoo\.com|outlook\.com|hotmail\.com|ukr\.net"
    r"|i\.ua|meta\.ua)$"
)

# Defines a password field type with specific length and pattern criteria for
# security purposes.
PasswordField = Annotated[
    str, Field(min_length=8, max_length=20, pattern=PASSWORD_PATTERN)
]

# Defines an email field type with specific length and pattern criteria
# to validate email addresses.
EmailField = Annotated[
    str, Field(min_length=1, max_length=64, pattern=EMAIL_PATTERN)
]
```

Рисунок 3.7 – Приклад створення своїх типів даних для валідації

Основна перевага pydantic полягає не тільки в його здатності перевіряти дані на відповідність заданим типам, а й у можливості генерації докладних

повідомлень про помилки. Ці повідомлення можна легко надсилати клієнтам для інформування про невірні дані, що сприяє покращенню досвіду користувача та спрощує процес налагодження додатків.

Створення маршрутів у Flask є важливим етапом у розробці вебдодатків на Python. У процесі визначення маршрутів розробники вказують URL-адреси та пов'язані з ними функції, які обробляють HTTP-запити від клієнтів. Flask надає гнучку структуру для опису маршрутів із використанням декораторів, що дозволяє легко визначати обробники для GET, POST та інших типів запитів. Кожен маршрут може бути налаштований для виконання різних завдань, включаючи доступ до даних, їх обробку та взаємодію з шаблонами для створення динамічного вмісту сторінок.

Рисунок 3.8 ілюструє процес створення маршруту у фреймворку Flask.

```python
@news_routes.route("/news", methods=["GET"])
def get_news(page, category):
    """Retrieves news based on request parameters."""
    news = current_app.news_service.get_news(page, category)
    return generate_response("News received successfully.", data=news)
```

Рисунок 3.8 – Приклад створення маршруту

Фреймворк Flask включає бібліотеку flask_pydantic, яка безпосередньо інтегрує схеми даних, визначені за допомогою Pydantic, в маршрути (див. рис. 3.9).

```python
@news_routes.route("/news", methods=["GET"])
@validate()
def get_news(query: GetNews):
    """Retrieves news based on request parameters."""
    news = current_app.news_service.get_news(query.page, query.category)
    return generate_response("News received successfully.", data=news)
```

Рисунок 3.9 – Приклад додавання валідації з flask_pydantic вхідних даних у маршруті

Для реалізації сесії користувача був створений клас SessionService, який включає методи для управління сесіями. Сесії зберігаються на сервері, що забезпечує суворіший контроль над ними. На рисунку 3.10 показано приклад реалізації методу SessionService для створення сесії в програмному забезпеченні.

```
def create_session(self, user_id: ObjectId):
    """Creates a session for the user."""
    new_session = Session(user_id=user_id).save()
    cookie_session[current_app.config["SESSION_ID_KEY"]] = str(
        new_session.pk
    )
    current_app.logger.info(
        f"Session {new_session.pk} created for user {user_id}."
    )
```

Рисунок 3.10 – Метод SessionService для створення сесії

Створено декоратор для верифікації користувача на основі класу SessionService (див. рис. 3.11). У декораторі викликається метод get_current_session, який повертає користувача. Якщо сесія відсутня, виникає помилка UnauthorizedError. Якщо користувача знайдено, його ідентифікатор передається в маршрут, якщо він потрібен.

```
def session_required(f):
    """
    Decorator that ensures a user session is present. Raises UnauthorizedError
    if no session is found.
    """

    @wraps(f)
    def wrapper(*args, **kwargs):
        current_user = current_app.session_service.get_current_session()
        if not current_user:
            raise UnauthorizedError

        # Return the user ID if it is expected in the route
        params = inspect.signature(f).parameters
        if 'user_id' in params:
            kwargs['user_id'] = current_user.user_id

        return f(*args, **kwargs)

    return wrapper
```

Рисунок 3.11 – Декоратор для перевірки наявності сесії

Для організації процесу збору новин у проєкті використовувалися декілька бібліотек, кожна з яких виконує певні функції для отримання та обробки даних.

Основний інструмент, що використовується для збору інформації з RSS-стрічок, – бібліотека feedparser. Вона надає зручний та ефективний спосіб для вилучення даних з різних джерел новин, працюючи з форматом RSS (див. рис 3.12).

```python
def _fetch_news_from_rss(self, rss_feed_url: str) -> list:
    """Fetches and returns the news entries from an RSS feed URL."""
    feed_response = feedparser.parse(rss_feed_url)
    feed_error = feed_response.get("bozo_exception", None)
    response_status = feed_response.get("status", None)

    if feed_error or response_status != 200:
        error_details = {
            "url": rss_feed_url,
            "feed_error": str(feed_error),
            "http_status": response_status,
        }

        current_app.logger.error(
            f"Error fetching RSS feed: {error_details}."
        )
        return None

    return feed_response.entries
```

Рисунок 3.12 – Агрегації новин з RSS-стрічок за допомогою feedparser

Однак, в процесі збору даних виникає необхідність додатково аналізувати сторінки джерел новин для вилучення додаткової інформації, наприклад, зображень, які можуть бути відсутні в RSS-стрічках. Для цього використовується бібліотека newspaper3k, яка здатна аналізувати структуру вебсторінок та отримувати потрібні дані про новини. Використання даних бібліотек допомагає комплексно підходити до завдання збору інформації, забезпечуючи повноту та актуальність даних.

На рисунку 3.13 показано процес отримання зображення новини за допомогою бібліотеки newspaper3k.

```python
def _scrape_news_top_image(self, news_url: str) -> str:
    """Scrapes and returns the top image URL from a news article."""
    try:
        news_article = Article(news_url, language="uk")
        news_article.download()
        news_article.parse()
    except ArticleException as error:
        current_app.logger.error(
            f"Failed to scrape top image from article: {news_url}. "
            f"Error: {error}"
        )
        return None

    return news_article.top_image
```

Рисунок 3.13 – Отримання зображення новини з newspaper3k

Процес створення рекомендацій новин із використанням класу NewsRecommendationService включає кілька ключових кроків. Спочатку сервіс аналізує історію переглядів користувача, щоб визначити найбільш відповідні новини. Якщо користувач переглянув менше статей, ніж необхідно для персоналізованих рекомендацій (вказано у конфігурації), пропонуються популярні новини на основі загальної популярності (див. рис. 3.14).

```python
def _get_popular_news(self) -> dict:
    """Retrieves popular news articles based on view count."""
    start_date = self._get_recommendation_start_date()
    recent_news = News.objects(date__gte=start_date)

    if not recent_news:
        return {}

    popular_news = recent_news.order_by("-view_count")[
        : current_app.config["PAGE_SIZE"]
    ]

    return [item.to_dict() for item in popular_news]
```

Рисунок 3.14 – Отримання N популярних новин за період N

Для початку процесу рекомендацій визначається тимчасове вікно, з якого розглядатимуться новини. Потім з бази даних вибираються кандидати на рекомендації, які відповідають цьому тимчасовому інтервалу та не були переглянуті користувачем.

Для кожної з цих новин будуються документи, які готуються векторизацією методом TF-IDF (Term Frequency-Inverse Document Frequency). Цей процес включає обробку тексту, включаючи категорію та джерело новини (див. рис. 3.15).

```
def _build_documents(self, news_list: News) -> list:
    """Build documents for TF-IDF vectorization."""
    return [
        f"{self.text_processor_service.preprocess_text(news.rss_feed.category.name)} "
        f"{' '.join(news.keywords)} "
        f"{self.text_processor_service.preprocess_text(news.rss_feed.source.name)}"
        for news in news_list
    ]
```

Рисунок 3.15 – Створення документів

Потім обчислюється подібність між документами, переглянутими користувачем, та кандидатами на основі косинусної подібності (див. рис. 3.16).

```
def _calculate_similarity(
    self, viewed_documents: list, candidate_documents: list
) -> ndarray:
    """
    Calculate similarity between viewed documents and candidate documents.
    """
    combined_documents = viewed_documents + candidate_documents
    tfidf_matrix = self.vectorizer.fit_transform(combined_documents)
    similarity_scores = cosine_similarity(
        tfidf_matrix[: len(viewed_documents)],
        tfidf_matrix[len(viewed_documents) :],
    )

    return similarity_scores
```

Рисунок 3.16 – Обчислення подібності між документами

Новини сортуються за зменшенням їхньої релевантності для користувача, що визначається шляхом підсумовування оцінок подібності. Підсумковим кроком є формування списку новин, що рекомендуються, який повертається користувачеві у вигляді словника, що містить інформацію про кожну рекомендовану новину (див. рис. 3.17).

```
similarity_scores = self._calculate_similarity(
    viewed_documents, candidate_documents
)

ranked_indices = similarity_scores.sum(axis=0).argsort()[::-1]
recommended_news = [
    candidate_news[i]
    for i in ranked_indices[: current_app.config["PAGE_SIZE"]]
]

current_app.logger.info(
    f"Returning recommended news for user: {user_id}"
)
return [item.to_dict() for item in recommended_news]
```

Рисунок 3.17 – Формування списку новин, що рекомендуються

Цей підхід дозволяє створювати персоналізовані рекомендації на основі переваг та історії переглядів користувача, покращуючи його досвід взаємодії з контентом новин.

3.2 Тестування програми

Створений агрегатор новин пройшов ручне тестування, що включає функціональне тестування, перевірку правильності обробки помилок та роботу очікуваної функціональності. У процесі тестування було виявлено та виправлено помилки, не виявлені на етапі розробки, зокрема помилка в інтервальному завданні, яка відповідала за збір новин із RSS-стрічок кожні 5 хвилин. Помилка була пов’язана з відсутністю очікуваного атрибута summary, що призводило до зупинення завдання та припинення парсингу інших стрічок.

Також було проведено тестування верстки та зручності використання. Проблем із верстkою не виявлено. Що стосується зручності використання, надалі можна внести поліпшення в проєкту, замінити пагінацію зі стрілками на головній сторінці на пагінацію, що дозволяє користувачеві вибирати сторінку. Важливо також додати функціонал пошуку новин та зручну фільтрацію результатів пошуку.

ВИСНОВКИ

У рамках кваліфікаційної роботи була розроблена інформаційна система, яка використовує інструменти Python і Flask для серверної частини та Vue.js для клієнтської частини. Розроблена система функціонує як агрегатор новин з додатковою функціональністю персоналізованих рекомендацій, що ґрунтуються на взаємодії користувача з платформою.

Основним методом, обраним для реалізації рекомендацій, є контентна рекомендація, орієнтована на аналіз схожості контенту. В контексті даного проєкту цей підхід полягає в порівнянні новин, які користувач вже переглядав, з тими, які він ще не бачив. Це дозволяє створювати персоналізовані рекомендації для кожного користувача на основі його індивідуальних інтересів і вподобань.

Використання контентної рекомендації підвищує рівень персоналізації взаємодії користувача з інформаційною системою, сприяючи поліпшенню якості інформаційних сервісів, наданих системою. Цей підхід дозволяє ефективніше задовольняти індивідуальні потреби користувачів і покращує загальний досвід взаємодії з платформою. Таким чином, розроблена інформаційна система являє собою значний крок у напрямку створення інтерактивних і персоналізованих інформаційних середовищ для користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. What is Entity Relationship Diagram (ERD)? *Visual Paradigm*. URL: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/ (дата звернення: 05.03.2024).
2. UML Class Diagram Tutorial. *Visual Paradigm*. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ (дата звернення: 05.03.2024).
3. What is Python? Executive Summary. *Python*. URL: https://www.python.org/doc/essays/blurb/ (дата звернення: 14.02.2024).
4. Data model. Python. URL: https://docs.python.org/3/reference/datamodel.html (дата звернення: 07.03.2024).
5. The Python Standard Library. *Python*. URL: https://docs.python.org/3/library/index.html (дата звернення: 07.03.2024).
6. Extending Python with C or C++. *Python*. URL: https://docs.python.org/3/extending/extending.html (дата звернення: 07.03.2024).
7. threading – Thread-based parallelism. *Python*. URL: https://docs.python.org/3/library/threading.html (дата звернення: 08.03.2024).
8. Asyncio – Asynchronous I/O. *Python*. URL: https://docs.python.org/3/library/asyncio.html (дата звернення: 08.03.2024).
9. Welcome to Flask. *Flask*. URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення: 11.03.2024).
10. Meet Django. *Django*. URL: https://www.djangoproject.com/ (дата звернення: 11.03.2024).
11. SciPy – Fundamental algorithms for scientific computing in Python. *SciPy*. URL: https://scipy.org/ (дата звернення: 12.03.2024).
12. NumPy – The fundamental package for scientific computing with Python. *NumPy*. URL: https://numpy.org/ (дата звернення: 12.03.2024).

13. Pandas – Python Data Analysis Library. *Pandas*. URL: https://pandas.pydata.org/ (дата звернення: 12.03.2024).
14. Sweigart A. Automate the Boring Stuff with Python. No Starch Press, 2015. 592 p.
15. An end-to-end platform for machine learning. *TensorFlow*. URL: https://www.tensorflow.org/ (дата звернення: 14.03.2024).
16. Industrial-Strength Natural Language Processing. *Spacy*. URL: https://spacy.io/ (дата звернення: 15.03.2024).
17. Machine Learning in Python. *Scikit-Learn*. URL: https://scikit-learn.org/stable/index.html (дата звернення: 14.03.2024).
18. Introduction. *Vue.js*. URL: https://vuejs.org/guide/introduction.html (дата звернення: 18.03.2024).
19. Основи компонентів. *Vue.js*. URL: https://ua.vuejs.org/guide/essentials/component-basics.html (дата звернення: 18.03.2024).
20. Conditional Rendering. *Vue.js*. URL: https://vuejs.org/guide/essentials/conditional.html (дата звернення: 19.03.2024).
21. The official Router for Vue.js. *Vue.js*. URL: https://router.vuejs.org/ (дата звернення: 20.03.2024).
22. What is Vuex? *Vue.js*. URL: https://vuex.vuejs.org/ (дата звернення: 20.03.2024).
23. What is content-based filtering? *IBM*. URL: https://www.ibm.com/topics/content-based-filtering (дата звернення: 22.03.2024).
24. What is collaborative filtering? *IBM*. URL: https://www.ibm.com/topics/content-based-filtering (дата звернення: 22.03.2024).
25. Collaborative Filtering. *Developers Google*. URL: https://developers.google.com/machine-

learning/recommendation/collaborative/basics (дата звернення: 25.03.2024).

26. What is Use Case Diagram? *Visual Paradigm*. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 25.03.2024).