КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-20-1
Шевчук П.Т

2024

ЗАКЛАД ВИЩОЇ ОСВІТИ
УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних в прикладних наук
Кафедра інформаційних технологій

на правах рукопису

Шевчук Петро Тарасович

УДК 004.4

Розробка онлайн-платформи для освітнього блогу

Спеціальність 121 – «Інженерія програмного забезпечення»
Кваліфікаційна робота на здобуття освітнього ступеню бакалавра

Нормоконтроль

_____Стисло О.В.
(підпис, дата, розшифрування підпису)

Студент

_____Шевчук П.Т
(підпис, дата, розшифрування підпису)

Допускається до захисту
Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.
(підпис, дата, розшифрування підпису)

Керівник роботи

асистент каф.ІТ
_____ Шкатуляк В.В
(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Шевчук Петро Тарасович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Розробка онлайн-платформи для освітнього блогу

керівник роботи:
Шкатуляк Василь Васильович, асистент каф. ІТ

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року
№ 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: Мова програмування Python, Django, Vue.js, HTML

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
   1. Виявлення потреб для блогу та аналіз існуючих рішень
   2. Обґрунтування вибору технологій та платформ
   3. Реалізація сайту

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз та дослідження аналогів	19.03.2024	Виконано
2.	Вибір платформи розробки	25.03.2024	Виконано
3.	Вибір мови програмування	25.03.2024	Виконано
4.	Вибір технологій програмування	13.04.02024	Виконано
5.	Створення Use Case діаграми	26.04.2024	Виконано
6.	Розробка основного функціоналу	15.05.2024	Виконано
7.	Оформлення графічного матеріалу	22.052024	Виконано
8	Підготовка до захисту	25.05.2024	Виконано

Студент _____ Шевчук П.Т

(підпис) (прізвище та ініціали)

Керівник роботи _____ Шкатуляк В.В

(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
14	Вигляд головної сторінки платформи “Prometheus”	40	Обмін повідомленнями
15	Курси платформи “EdEra”	40	Сповіщення
16	Предмети для вивчення платформи “На Урок”	43	Пошук за ім'ям користувача
17	Напрямки для навчання які надає “Освіторія”	43	Пошук за темою поста
30	Форма реєстрації	45	Профіль користувача
34	Головної сторінки	48	Сертифікат

АНОТАЦІЯ

Дана кваліфікаційна робота присвячена розробці та імплементації онлайн-платформи для освітнього блогу. Робота демонструє, як сучасні технології можуть бути застосовані для створення ефективних та інтерактивних освітніх ресурсів, сприяючи кращому залученню та взаємодії користувачів.

У першому розділі зосереджено увагу на аналізі потреб користувачів онлайн-платформ для освітніх блогів. Також проведено дослідження існуючих аналогічних платформ, їх функціональності та користувацького досвіду. Розглянуто переваги та недоліки цих систем, що дозволило визначити ключові аспекти для розробки власної платформи.

Другий розділ присвячений обґрунтуванню вибору технологій та методологій розробки. Зокрема, було обрано Python і Django для бекенду та Vue.js для фронтенду з урахуванням їх гнучкості, масштабованості та великої підтримки спільноти. Також у цьому розділі описано архітектурні вибори, включаючи структуру бази даних та системи керування контентом.

У третьому розділі детально описано процес реалізації платформи: від розробки інтерфейсу користувача до імплементації серверної логіки. Подано також методику тестування розробленої системи, включаючи юніт-тестування та інтеграційне тестування для забезпечення надійності та безпеки платформи. Наведено приклади тестових сценаріїв і результатів виконання тестів.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-ПЛАТФОРМА, ОСВІТНІЙ БЛОГ, РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА, ЛОГІКА НА СЕРВЕРНІЙ СТОРОНІ, ПРОФІЛІ КОРИСТУВАЧІВ, ФУНКЦІОНАЛЬНІСТЬ ПОШУКУ, СИСТЕМА ПОВІДОМЛЕНЬ, СИСТЕМА ПОВІДОМЛЕНЬ, ВЕБ-ТЕХНОЛОГІЇ, ЗАЛУЧЕННЯ КОРИСТУВАЧІВ.

SUMMARY

This thesis is dedicated to the development and implementation of an online platform for an educational blog. The work demonstrates how modern technologies can be applied to create effective and interactive educational resources, enhancing user engagement and interaction.

The first chapter focuses on analyzing the needs of users of online platforms for educational blogs. It also includes a study of existing similar platforms, their functionalities, and user experiences. The advantages and disadvantages of these systems are considered, which has helped to identify key aspects for the development of our own platform.

The second chapter is devoted to the justification of the choice of technologies and development methodologies. Specifically, Python and Django were chosen for the backend and Vue.js for the frontend, considering their flexibility, scalability, and strong community support. This section also describes architectural choices, including the database structure and content management systems.

The third chapter details the process of implementing the platform: from developing the user interface to implementing server logic. It also presents the testing methodology of the developed system, including unit testing and integration testing to ensure the reliability and security of the platform. Examples of test scenarios and test results are provided.

KEYWORDS: ONLINE PLATFORM, EDUCATIONAL BLOG, USER INTERFACE DEVELOPMENT, SERVER-SIDE LOGIC, USER PROFILES, SEARCH FUNCTIONALITY, MESSAGE SYSTEM, WEB TECHNOLOGIES, USER ENGAGEMENT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	–	Application Programing Interface
МООК	–	Масові відкриті онлайн-курси
БД	–	База даних
HTML	–	Hyper Text Markup Language
CSRF	–	cross-site request forgery

ВСТУП

Актуальність теми. У сучасному світі, де технології та інформація розвиваються надзвичайно швидко, освітня сфера переживає значні трансформації. Особливе місце в цьому процесі займає онлайн-навчання, яке не тільки забезпечує доступ до освіти без обмежень за географічним розташуванням або часом, але й відкриває нові можливості для індивідуального та гнучкого підходу до навчання. Завдяки цифровим технологіям, освітні блоги стали важливим інструментом в системі освіти, здатним задовольнити зростаючий попит на постійне оновлення знань та навичок.

Освітні блоги використовуються не тільки для самостійного навчання, але й як доповнення до традиційної освіти, пропонуючи студентам та фахівцям глибші знання в певних областях. Це дозволяє авторам не просто ділитися інформацією, але й створювати спільноту однодумців, що обмінюються ідеями, обговорюють нові дослідження та сприяють взаємному навчанню.

Враховуючи важливість та потенціал онлайн-навчання, розробка ефективної, зручної та інтерактивної освітньої платформи для блогів стає ключовим завданням у сфері освіти. Така платформа повинна не тільки підтримувати стандарти якості контенту та навчальних матеріалів, але й забезпечувати надійність, доступність та зручність користування, що робить її невід'ємною частиною освітнього процесу в цифрову епоху.

Однак, існуючі освітні блоги часто мають ряд недоліків:

1. Незручність використання: блоги зазвичай не розроблені спеціально для освітніх цілей, що може ускладнювати навігацію та пошук потрібної інформації на сторінках.
2. Відсутність інтерактивності: блоги зазвичай є статичними веб-сторінками, що не дає можливості для спілкування між автором та читачами, а також для інтерактивного навчання.

3. Відсутність системності: блоги не завжди мають чітку структуру та послідовність, що може ускладнювати процес навчання.

Розробка онлайн-платформи для освітнього блогу може вирішити ці проблеми та зробити онлайн-навчання більш ефективним та цікавим.

Мета роботи. Метою кваліфікаційної роботи є розробка онлайн-платформи для освітнього блогу, яка б відповідала сучасним потребам користувачів та забезпечувала ефективне та цікаве онлайн-навчання.

Завдання роботи. Для досягнення поставленої мети необхідно вирішити ряд завдань:

1. Провести аналіз існуючих онлайн-платформ для освітніх блогів.
2. Розробити концепцію онлайн-платформи для освітнього блогу.
3. Вибрати програмні та апаратні засоби для реалізації онлайн-платформи.
4. Розробити інтерфейс користувача онлайн-платформи.
5. Реалізувати онлайн-платформу.
6. Провести тестування онлайн-платформи.
7. Оцінити ефективність онлайн-платформи.

Об’єкт роботи. Об’єктом дослідження є онлайн-платформи для освітніх блогів.

Предмет роботи. Предметом дослідження є методи та засоби розробки онлайн-платформи для освітнього блогу.

Методи дослідження. Для вирішення поставлених завдань будуть використовуватися такі методи дослідження:

- аналіз літературних джерел;
- аналіз існуючих онлайн-платформ;
- анкетування;
- інтерв'ювання;
- опитування;
- експеримент;
- математичне моделювання.

Практичне значення одержаних результатів. Розроблена онлайн-платформа може бути використана для створення та ведення освітніх блогів, а також для організації онлайн-навчання.

Практичне значення роботи полягає в:

– створенні нового програмного продукту;

– підвищенні ефективності та доступності онлайн-навчання.

Одержані результати можуть бути використані для подальшого розвитку онлайн-навчання.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на XI Міжнародній науковій конференції «IT екосистема: цифровізація бізнес-процесів в умовах війни» у тезі на тему: «Використання AI-інструментів в роботі над UI/UX частиною проекту», що проходила 23-24 листопада 2023 року.

Структура. Розділи - 3. Загальний обсяг основної частини - 40 сторінки. Список використаних джерел - 20

РОЗДІЛ 1. ВИЯВЛЕННЯ ПОТРЕБ ДО ПРОЕКТУ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Виявлення потреб до проекту

Сучасний світ характеризується стрімким розвитком інформаційних технологій, які знаходять широке застосування в усіх сферах людської діяльності, включаючи освіту. Онлайн-освіта стає все більш популярною завдяки своїй доступності та зручності. У цьому контексті освітні блоги відіграють важливу роль, надаючи можливість для обміну знаннями, досвідом та ідеями. Розробка онлайн-платформи для освітнього блогу відкриває нові перспективи для авторів контенту та їх аудиторії. Проте, перед початком розробки важливим пунктом було провести детальний аналіз потреб проекту та огляд існуючих рішень у цій сфері [1].

Для того щоб створити успішну онлайн-платформу для освітнього блогу, перш за все, необхідно було зрозуміти, що саме шукає цільова аудиторія та які проблеми вона намагається вирішити за допомогою такої платформи. Основні потреби сформулював наступним чином [2]:

1. Доступність та зручність користування: платформа має бути простою у використанні. Це передбачає наявність зрозумілого інтерфейсу, спрощеного процесу реєстрації, а також зручних інструментів для навігації та читання контенту.
2. Інтерактивність та залучення аудиторії: сучасні користувачі цінують можливість не лише отримувати інформацію, а й активно взаємодіяти з контентом. Наявність функцій для коментування, обміну повідомленнями є невід'ємною частиною платформи.
3. Персоналізація: залученість користувачів зростає, коли вони можуть адаптувати платформу під свої потреби. Це може включати персоналізовані рекомендації контенту та можливість створювати індивідуальні списки читання.
4. Якість контенту: високий рівень контенту безпосередньо впливає на

привабливість платформи. Засоби для створення, редагування та форматування постів, що включають функції для інтеграції зображень, відео та інших мультимедійних елементів, є важливими для авторів.

5. Захист даних та конфіденційність: в умовах зростаючої уваги до приватності та безпеки, платформа повинна гарантувати захист особистих даних та контенту користувачів, включаючи шифрування та захист від несанкціонованого доступу, з міцними політиками конфіденційності.

Ці ключові потреби формують фундамент для розробки онлайн-платформи, яка не тільки задовольнятиме базові вимоги користувачів, але й пропонуватиме унікальні функції та інструменти, які вирізняють її на тлі інших освітніх ресурсів. Залучення користувачів до процесу розробки, збір зворотного зв'язку на ранніх етапах та гнучке впровадження змін згідно з їхніми потребами допоможуть створити продукт, який буде високо цінуватись та широко використовуватись в освітньому середовищі.

Виконуючи ці кроки, проект зможе реалізувати візію створення інноваційної платформи, яка не тільки слугуватиме майданчиком для обміну знаннями, але й стимулюватиме користувачів до постійного навчання, дослідження та розвитку.

1.2 Аналіз існуючих українських освітніх платформ

Для забезпечення конкурентоспроможності нової платформи проведений аналіз існуючих освітніх блогів та платформ. Такий аналіз дозволив виявити сильні та слабкі сторони конкурентів, а також визначити потреби аудиторії [3]. Наразі існують численні платформи, які пропонують різноманітні функціональні можливості для авторів та їхніх читачів. Серед найпопулярніших я виділив такі [4]:

1. Prometheus (рис. 1.1) – це масовий відкритий онлайн курс (МООК) платформа, яка пропонує безліч безкоштовних курсів від провідних українських та міжнародних університетів. Платформа зосереджена на забезпеченні доступу до якісної освіти для кожного. Особливістю Prometheus є висока якість відео

курсів, інтерактивність навчання через тести та задачі, а також можливість отримання сертифікатів про проходження курсів [5].

Сильні сторони:

– широкий спектр курсів від відомих університетів;

– висока якість навчального контенту та професійні відеоматеріали;

– наявність сертифікації після завершення курсів, що додає цінності для професійного розвитку;

– можливості для інтерактивного навчання через тести та вправи.

Слабкі сторони:

– обмежена кастомізація навчального досвіду для користувачів;

– не всі курси безкоштовні, що може бути бар'єром для користувачів.

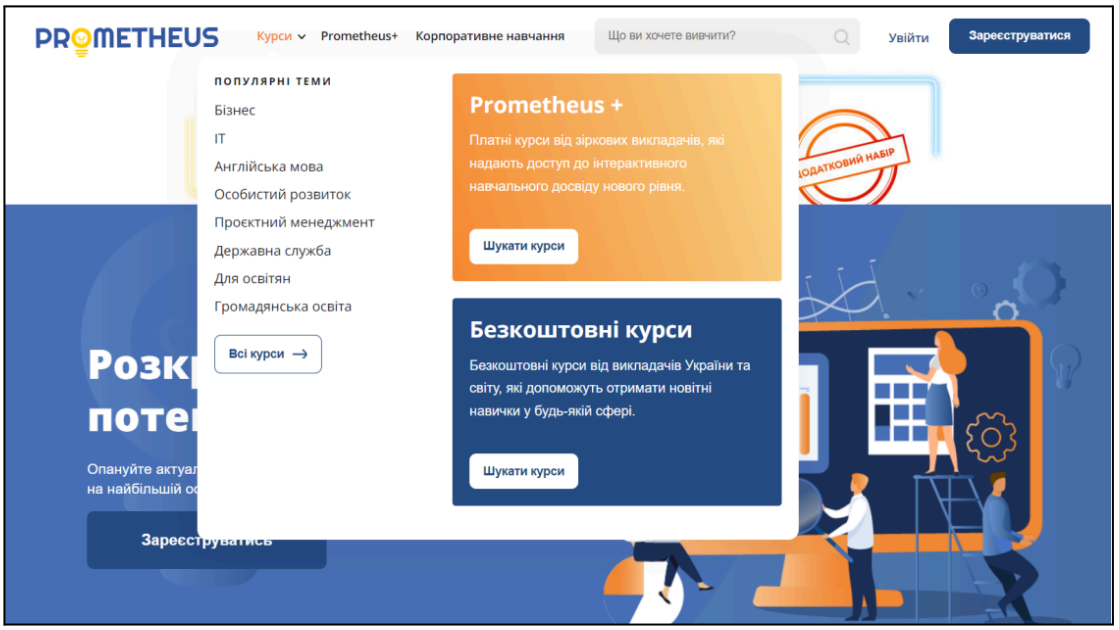


Рисунок 1.1 – Вигляд головної сторінки платформи “Prometheus”

2. EdEra (рис. 1.2) – ще одна платформа для онлайн-навчання, яка пропонує курси з різноманітних дисциплін. EdEra відома своїм зосередженням на наданні україномовного контенту високої якості. Платформа співпрацює з університетами та індивідуальними викладачами для створення курсів, що відповідають потребам сучасного ринку праці [6] .

Сильні сторони:

– велика кількість курсів українською, сприяючи локалізації освіти;

– співпраця з кваліфікованими викладачами та університетами;

– курси орієнтовані на актуальні потреби ринку праці.

Слабкі сторони:

– інтерфейс може виявитися не таким зрозумілим для користувачів;

– обмежені можливості для взаємодії та обговорення між студентами.

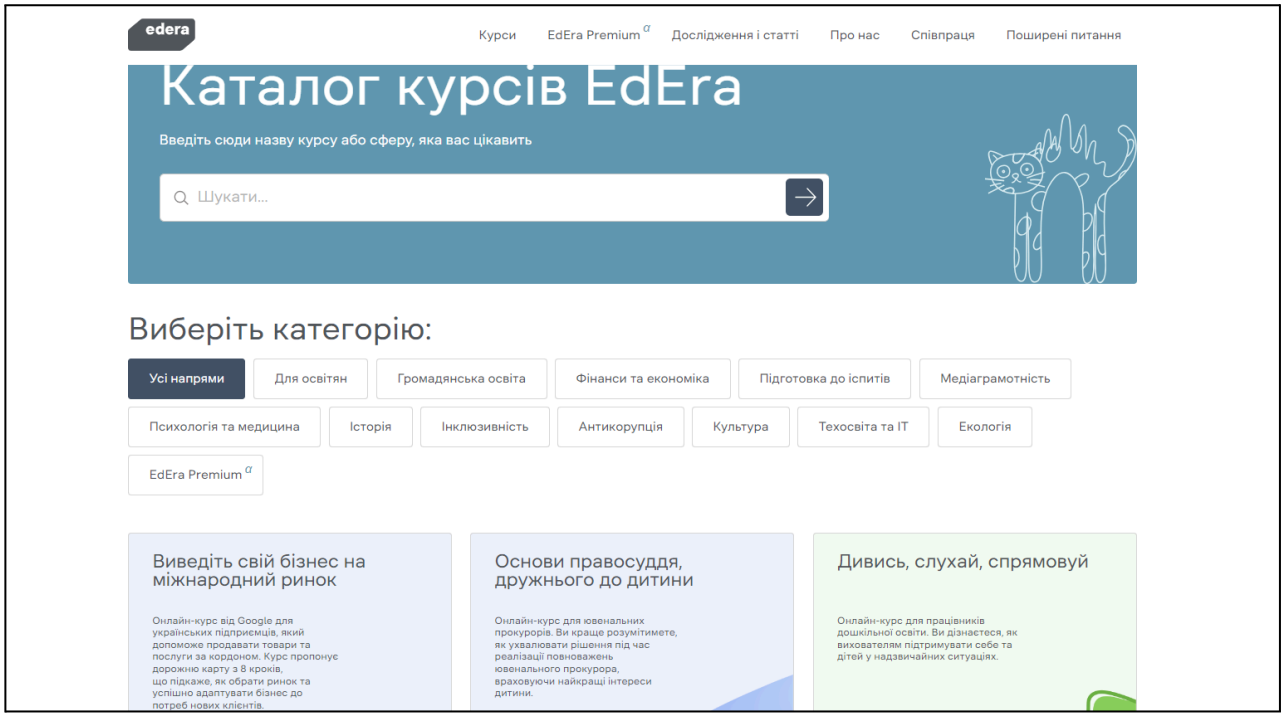


Рисунок 1.2 – Курси платформи “EdEra”

3. На Урок (рис. 1.3) – платформа, яка надає ресурси для вчителів та учнів основної та середньої школи. Вона включає розробки уроків, тести, презентації та інші матеріали. Відмінною рисою платформи є її орієнтація на шкільну програму України, що робить її незамінним ресурсом для вчителів [7].

Сильні сторони:

– платформа містить базу ресурсів, для шкільної програми України;

– матеріали доступні для різних вікових груп і шкільних предметів;

– інструменти для вчителів, які допомагають планувати уроки та використовувати інноваційні методи навчання.

Слабкі сторони:

– обмежена адаптивність сайту для мобільних пристроїв.

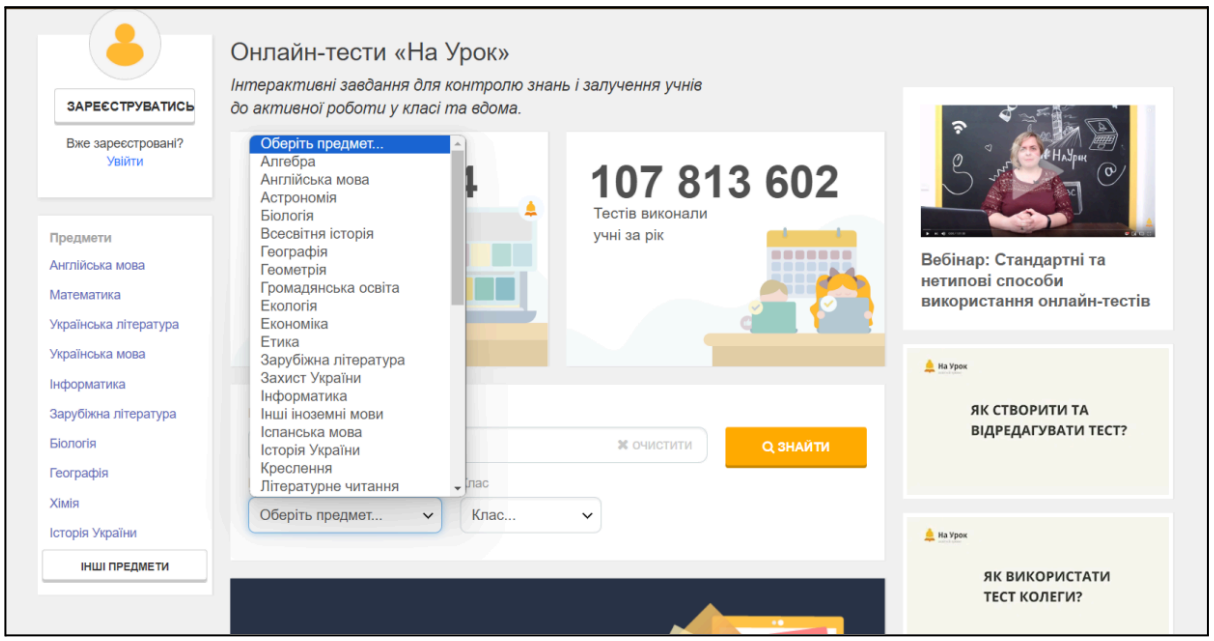


Рисунок 1.3 – Предмети для вивчення платформи “На Урок”

4. Освіторія – некомерційна організація, яка прагне інноваційно змінити українську освіту. Вони пропонують різноманітні освітні проекти, зокрема, для підвищення кваліфікації вчителів, розвитку дітей та батьківської освіти. Їхній підхід базується на інтеграції сучасних технологій у навчальний процес та розвитку критичного мислення [8].

Сильні сторони:

– широкий діапазон освітніх проектів, стимулюють інновації в освіті;

– зосередженість на розвитку критичного мислення та сучасних підходах до навчання;

– велика увага до розвитку вчителів та батьківської освіти.

Слабкі сторони:

– обмежена інтерактивність та взаємодія на платформі;

– може відчуватися недолік індивідуального підходу до кожного користувача через широкий охоплення різних проектів.

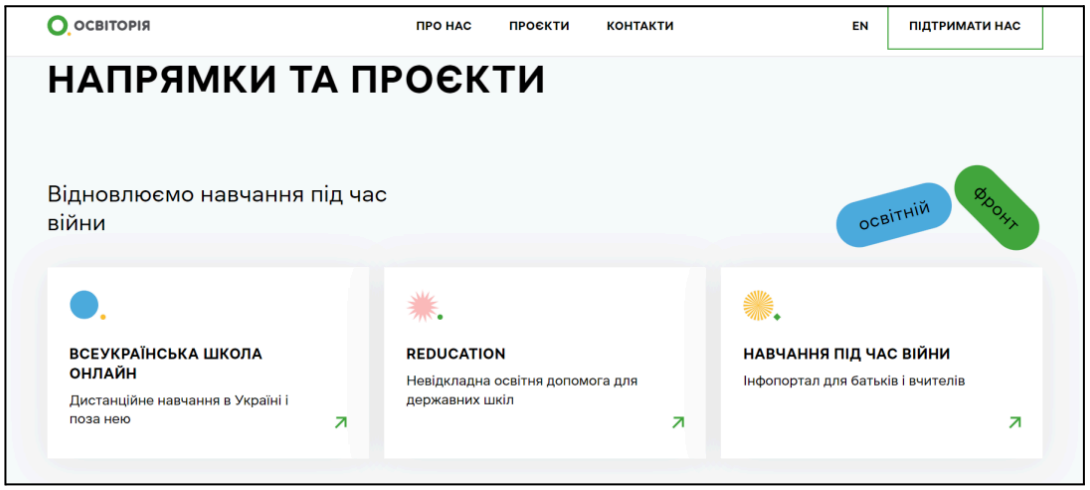


Рисунок 1.4 – Напямки для навчання які надає “Освіторія”

З урахуванням цих аспектів, розробка онлайн-платформи в Україні вимагатиме не лише технічної експертизи, але й глибокого розуміння освітніх потреб та переваг місцевої аудиторії. Це включає знання про те, як найкраще структурувати та представити освітній контент, щоб він був зручний для користувачів, а також розуміння способів залучення та утримання аудиторії.

1.3 Постановка задачі

Аналізуючи потреби проекту та огляд існуючих українських освітніх платформ виявив важливі аспекти, які необхідно врахувати при розробці онлайн-платформи для освітнього блогу. Цей процес підкреслив значення цільової аудиторії, її потреб та очікувань, а також необхідність створення змістовного, інтерактивного та доступного навчального контенту.

Для успішної реалізації проекту, я зосередився на наступних ключових елементах:

1. Забезпечення високої якості та релевантності контенту, який відповідає актуальним освітнім тенденціям та потребам української аудиторії.
2. Створення зручного та інтуїтивно зрозумілого інтерфейсу, що спрощує навігацію по сайту та доступ до навчальних матеріалів.
3. Інтеграція інтерактивних елементів для залучення користувачів та підвищення ефективності навчання.

4. Впровадження функцій персоналізації для адаптації контенту під індивідуальні потреби та інтереси користувачів.

5. Забезпечення безпеки даних та приватності користувачів, що є критично важливим у сучасному цифровому світі.

Здійснення цих рекомендацій дозволить створити платформу, яка не тільки приваблюватиме користувачів високоякісним та актуальним контентом, але й забезпечуватиме їм зручний і безпечний досвід користування. Така платформа стане цінним ресурсом для розвитку освітньої спільноти в Україні, сприятиме поширенню знань і формуванню навичок, необхідних у XXI столітті.

Висновок до розділу 1

У цьому розділі було проведено ретельний аналіз потреб проекту онлайн-платформи для освітнього блогу, а також огляд існуючих українських освітніх платформ. Спочатку ми визначили ключові потреби, які має задовольняти наша платформа, включаючи доступність, інтерактивність, персоналізацію, високу якість контенту та безпеку даних. Ці потреби були виокремлені на основі аналізу цільової аудиторії та основних трендів у сфері цифрової освіти.

Далі, ми розглянули чотири відомі українські освітні платформи — Прометеус, ЕдЕра, На Урок та Освіторія і проаналізували їхні сильні та слабкі сторони. Цей аналіз дозволив нам краще зрозуміти ринкове середовище та визначити, які функціональні можливості та елементи дизайну можуть зробити нашу платформу більш конкурентоспроможною та привабливою для користувачів.

Таким чином, у першому розділі ми заклали фундамент для подальшої розробки проекту, чітко визначивши, які основні потреби та вимоги має враховувати нова платформа, а також які уроки можна винести з досвіду існуючих платформ. Ця робота допоможе нам зосередитися на створенні вдосконаленого та інноваційного продукту, що задовольнятиме специфічні потреби та вимоги освітянської спільноти.

РОЗДІЛ 2. ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА ПЛАТФОРМИ РОЗРОБКИ

Розробка онлайн-платформи для освітнього блогу вимагає вибору найбільш ефективних та адаптованих технологій, які дозволять створити надійний, масштабований та зручний у використанні продукт. Наступні підпункти детально описують обґрунтування вибору основних технологічних рішень для проекту.

2.1 Вибір платформи розробки

У світі веб-розробки, вибір платформи для створення онлайн-платформи для освітнього блогу можна порівняти з вибором фундаменту для будівлі. Так само, як міцний фундамент забезпечує стабільність та довговічність будівлі, так і правильно обрана платформа визначає успіх, масштабованість та безпеку веб-додатку. Ось детальний огляд платформ, які я розглядав, та міркування, що стояли за вибором найбільш підходящої.

Перше рішення, яке я прийняв, стосувалося загальної архітектури, будувати монолітну систему або вибрати архітектуру мікросервісів.

Монолітна архітектура дозволяє швидко розпочати проект завдяки своїй простоті та зручності управління. Однак вона може ускладнити масштабування та оновлення в майбутньому.

Архітектура мікросервісів надає більшу гнучкість та можливість незалежного масштабування окремих частин системи. Але водночас вона вимагає більш складного управління та координації між сервісами.

Основу моєї платформи складає веб-фреймворк, що допомагає в розробці структури веб-додатку, управлінні даними, забезпеченні безпеки та багатьох інших аспектах. Розглянуто кілька популярних фреймворків:

1. Django (Python): оснащений філософією "все включено", Django

забезпечує широким набором вбудованих інструментів, що спрощують процес розробки. Завдяки своїм розгалуженим функціональним можливостям та солідному рівню безпеки, він став ідеальним вибором для нашого проекту [9].

2. Flask (Python): легкий веб-фреймворк для Python, який надає гнучкість та простоту в розробці, дозволяючи розробникам з більшою контролюванням будувати веб-додатки з нуля [10].

3. Ruby on Rails (Ruby): повнофункціональний фреймворк для веб-додатків на Ruby, що використовує принципи "Convention over Configuration" та "Don't Repeat Yourself" для прискорення розробки комплексних веб-сайтів [11].

4. React (JavaScript)для фронтенду: бібліотека JavaScript, яка використовується для створення інтерактивних користувацьких інтерфейсів і динамічних веб-сторінок, дозволяючи розробникам ефективно управляти станом та даними на клієнтській стороні [12] .

Після ретельного аналізу та порівняння я обрав Django як основний фреймворк для бекенду нашої платформи з кількох причин:

1. Швидкість розробки: Django славиться своєю здатністю прискорювати процес розробки веб-додатків. Він надає велику кількість готових до використання модулів та інструментів, що зменшують необхідність писати багато бойлерплейту та дозволяють розробникам концентруватися на більш важливих аспектах розробки.

2. Безпека: безпека є однією з вирішальних переваг Django. Фреймворк автоматично забезпечує захист від багатьох загроз, таких як SQL ін'єкції, кросс-сайтовий скриптинг та фальсифікація запитів між сайтами (CSRF). Це зменшує ризик вразливостей, що важливо для платформ, де безпека даних є критичною.

3. Масштабованість: Django добре підходить для проектів будь-якого розміру, від маленьких до великих веб-додатків. Він має потужні інструменти для управління трафіком і може ефективно обробляти великі обсяги користувачів та запитів.

4. Гнучкість та розширюваність: фреймворк забезпечує високу гнучкість, дозволяючи легко розширювати та налаштовувати функціональність відповідно до конкретних потреб проекту. Розробники можуть використовувати численні сторонні пакети або створювати власні плагіни для інтеграції нових функцій.

5. Велика спільнота та підтримка: Django підтримується великою і активною спільнотою розробників. Це забезпечує регулярні оновлення, широку документацію та швидке вирішення будь-яких виникаючих проблем. Наявність великої кількості ресурсів та готових рішень робить вивчення та роботу з Django ефективнішою.

Загалом, вибір Django як платформи для розробки моєї онлайн-платформи для освітнього блогу забезпечує міцний фундамент, на якому можна побудувати надійний, безпечний та легко масштабований веб-додаток, здатний задовольнити потреби наших користувачів сьогодні та в майбутньому.

2.2 Вибір мови програмування Python

У процесі вибору інструментарію для розробки моєї онлайн-платформи для освітнього блогу, серед різноманіття мов програмування, вибір припав на Python. Цей вибір не був зроблений випадково. Python відомий своєю універсальністю, читабельністю коду та широкими можливостями у різних областях розробки, від веб-додатків до наукових досліджень та штучного інтелекту. Нижче наведено ключові фактори, які вплинули на мій вибір.

Читабельність та простота. Python славиться своїм чистим та читабельним синтаксисом, що робить розробку швидшою та ефективнішою. Це дозволяє розробникам з різним рівнем досвіду легко приєднуватися до проекту та вносити свій вклад без довгого періоду адаптації. Код на Python легко читати та розуміти, що сприяє кращій співпраці в команді та підвищенню якості коду.

Широка підтримка фреймворків. Python підтримує численні веб-фреймворки, такі як Django та Flask, які спрощують процес розробки, надаючи потужні інструменти для роботи з БД, шаблонами, формами та іншими стандартними елементами веб-додатків. Для моєї платформи освітнього блогу

важливою є можливість швидкого розгортання стабільного та безпечного веб-додатку, що і забезпечується цими фреймворками.

Велика спільнота та бібліотеки. Python має одну з найбільших та найактивніших розробницьких спільнот, завдяки чому доступна велика кількість відкритих бібліотек та інструментів для різноманітних задач. Це означає, що для більшості проблем, з якими можу зіткнутися під час розробки, вже існують готові рішення. Це значно прискорює процес розробки та дозволяє зосередитися на унікальних особливостях нашого проекту.

Міжплатформеність. Python міжплатформена мова, що дозволяє запускати розроблені на ній додатки на будь-якій операційній системі без необхідності зміни коду. Це особливо цінно для моєї платформи, оскільки я прагну досягти широкої аудиторії користувачів з різними пристроями та операційними системами. Незалежно від того, чи використовують користувачі Windows, macOS, Linux або будь-яку систему на основі UNIX, Python забезпечує їм безперешкодний доступ до всіх функцій нашого веб-додатку.

Відкритий код. Python є мовою з відкритим вихідним кодом, що дозволяє користуватися потужним інструментарієм без додаткових витрат на ліцензії. Це також означає, що я можу вносити зміни до вихідного коду мови або фреймворків, якщо це необхідно для досягнення певних цілей проекту. Така гнучкість і можливість адаптації є важливими для створення інноваційних та надійних веб-додатків.

Багатофункціональність. Python підтримує різноманітні парадигми програмування, включаючи об'єктно-орієнтоване, функціональне та процедурне програмування. Ця гнучкість робить Python ідеальним вибором для розробки складних веб-додатків, які вимагають інтеграції з іншими системами або реалізації складної бізнес-логіки. Його здатність працювати з різними типами БД та використовувати розширені алгоритми обробки даних є великою перевагою для розробки освітніх платформ.

Враховуючи вищезазначені переваги, Python виявляється найкращим вибором для розробки моєї онлайн-платформи для освітнього блогу. Його

універсальність, велика спільнота підтримки, багатий набір інструментів та бібліотек, а також легкість у вивченні та використанні роблять Python ідеальною мовою для мого проекту. Я впевнений, що ця мова забезпечить моєму додатку стабільність, масштабованість та гнучкість, необхідні для успішного впровадження та подальшого розвитку.

2.3 Вибір технологій програмування

При створенні онлайн-платформи для освітнього блогу, вибір правильних технологій є критично важливим для забезпечення високої продуктивності, безпеки, і відмінного користувацького досвіду. У цьому контексті, я зосередив свій вибір на фронтенд фреймворку Vue.js і бекенд фреймворку Django, які разом формують потужне поєднання для розробки сучасних веб-додатків. Ось детальніший огляд наших виборів [13].

Vue.js прогресивний JavaScript-фреймворк, призначений для створення інтерфейсів користувача. Він забезпечує реактивне та компонентне програмування, що робить його ідеальним для розробки динамічних веб-додатків, таких як онлайн-платформа для освітнього блогу. Ось кілька ключових переваг вибору Vue.js:

1. Легкість інтеграції: Vue.js вирізняється своєю здатністю легко інтегруватися як в нові, так і в існуючі веб-проекти, дозволяючи розробникам впроваджувати Vue поступово без необхідності переписувати існуючий код.
2. Детальна документація та велика спільнота: Vue.js має дуже чітку і зрозумілу документацію, яка робить його доступним для новачків, а також достатньо глибокою для досвідчених розробників. Велика і активна спільнота надає додаткові ресурси, підтримку та розширення, що сприяє постійному вдосконаленню ваших навичок розробки.
3. Гнучка реактивність системи: система реактивності у Vue дозволяє автоматично оновлювати UI відповідно до змін у стані додатка, що значно спрощує управління динамічним контентом і даними без помилок.
4. Компонентна архітектура: Vue сприяє створенню

перевикористовуваних компонентів, що можуть бути легко управляні та масштабовані. Це дозволяє розбивати інтерфейс користувача на незалежні, легко підтримувані частини, що спрощує процес розробки та тестування.

5. Ефективність і продуктивність: Vue оптимізований для швидкої відповіді та мінімального впливу на продуктивність, навіть при роботі з великими додатками. Це робить його ідеальним вибором для розробки як малих, так і великих масштабних проектів.

Django високорівневий Python-фреймворк для швидкої розробки безпечних та обслуговуваних веб-додатків. Він націлений на автоматизацію можливо якнайбільшої кількості конфігурацій для веб-додатку, дозволяючи розробникам зосередитися на написанні додатку, а не на повторному винаході колеса. Деякі з переваг Django:

1. Швидкість розробки: Django дозволяє швидко створювати додатки завдяки великій кількості готових рішень, що скорочує час розробки.

2. Безпека: вбудовані засоби безпеки захищають додаток від поширених загроз, таких як SQL-ін'єкції та фальсифікація запитів.

3. Масштабованість: Django підходить для проектів будь-якого розміру, від маленьких особистих блогів до великих корпоративних сайтів.

4. Гнучкість: завдяки модульній архітектурі, Django дозволяє легко налаштовувати та розширювати функціональність додатку.

5. Підтримка спільноти: велика та активна спільнота надає багато ресурсів, включаючи документацію, бібліотеки і підтримку, що полегшує роботу з Django.

Інтеграція Vue.js і Django. Поєднання Vue.js для фронтенду та Django для бекенду створює потужну платформу для розробки сучасних веб-додатків. Vue.js пропонує гнучку та ефективну систему для створення користувацьких інтерфейсів, тоді як Django забезпечує надійний бекенд із сильними інструментами для роботи з даними та безпекою [14].

Інтеграція цих технологій дозволяє розробникам:

– ефективно розділяти клієнтську та серверну частини додатку,

спрощуючи розробку та тестування;

– використовувати API-підхід для зв'язку між фронтендом і бекендом, що підвищує гнучкість розробки та дозволяє легше масштабувати проект;

– створювати швидкі веб-додатки, які ефективно обробляти великі обсяги даних та забезпечувати відмінний користувацький досвід.

Використання Vue.js і Django разом у проекті нашої онлайн-платформи для освітнього блогу дозволяє нам не тільки швидко розвивати продукт з потужними функціональними можливостями, але й забезпечити його надійність, безпеку та високу продуктивність. Таке поєднання технологій робить нашу платформу гнучко будовану захист від багатьох видів кібератак, таких як SQL-ін'єкції, cross-site scripting (XSS), cross-site request forgery (CSRF) та clickjacking. Фреймворк автоматично керує безпечними користувацькими сесіями та паролями.

1. Масштабованість: однією з ключових переваг Django є його здатність до масштабування. Він спроектований таким чином, що добре справляється як з маленькими, так і з великими навантаженнями, що робить його ідеальним вибором для проектів будь-якого розміру.

2. Підтримка RESTful API: Django може легко інтегруватися з Django REST framework для створення могутніх API, що є важливим для сучасних веб-додатків та мобільних застосунків, забезпечуючи швидке та ефективне спілкування між фронтендом та бекендом.

Комбінація Vue.js та Django для розробки онлайн-платформи для освітнього блогу створює потужний стек технологій, що забезпечує високу продуктивність, швидку розробку та гнучкість у впровадженні функціональних вимог проекту. Vue.js пропонує інтуїтивно зрозумілий інтерфейс, що ідеально підходить для створення динамічного та взаємодіючого користувацького досвіду, тоді як Django забезпечує міцний фундамент для розробки бекенду з високим рівнем безпеки та масштабованості.

Ця комбінація дозволяє розділити відповідальність між клієнтською та серверною частинами додатку, що сприяє кращій організації коду та спрощує

процес розробки та тестування. Також це відкриває широкі можливості для майбутньої інтеграції з іншими сервісами та платформами, що є ключовим фактором для адаптації та розвитку онлайн-платформи в динамічному цифровому світі.

Обрання Vue.js та Django як основи для розробки нашої онлайн-платформи для освітнього блогу є стратегічним рішенням, що забезпечує баланс між продуктивністю розробки, гнучкістю в реалізації вимог проекту, високим рівнем безпеки та зручність користування для кінцевих користувачів. Ця комбінація технологій не тільки дозволяє створити інтуїтивно зрозумілий і привабливий інтерфейс за допомогою Vue.js, але й забезпечити потужну, надійну бекенд-структуру на основі Django. Таким чином, ми отримуємо здатність швидко реагувати на зміни в проектних вимогах та розвивати платформу, адаптуючи її до майбутніх потреб користувачів та технологічних трендів.

Окрім того, вибір цих технологій відкриває доступ до величезної кількості плагінів і бібліотек від спільнот, що постійно зростають і розвиваються. Це не тільки спрощує розробку, дозволяючи інтегрувати перевірені часом компоненти, але й знижує загальні витрати на розробку та підтримку платформи.

Використання Django як бекенд-фреймворку відкриває можливості для розширеного управління даними та автоматизації багатьох процесів, таких як аутентифікація користувачів, управління сесіями, а також обробка і зберігання великих обсягів інформації з високою продуктивністю та надійністю. Така функціональність є критично важливою для освітніх платформ, де безпека та надійність зберігання даних є пріоритетом.

Vue.js, з іншого боку, дозволяє нам створювати динамічний і залучаючий користувацький інтерфейс з легкістю і ефективністю, використовуючи сучасні підходи до розробки фронтенду. Це допомагає забезпечити відмінний досвід користувача, підвищуючи залученість та задоволення користувачів від використання платформи.

2.4 Use Case діаграма

Use Case (Випадок Використання) діаграма є важливим інструментом у процесі збору вимог до програмного продукту. Вона демонструє різні способи, якими користувачі (актори) можуть взаємодіяти з системою. У контексті розробки онлайн-платформи для освітнього блогу, така діаграма допомагає розробникам, менеджерам проекту, та зацікавленим сторонам отримати чітке уявлення про функціональність системи (А). Це особливо корисно для ідентифікації та з'ясування потреб користувачів, структурування функцій додатку, а також для уникнення непорозумінь між стейкхолдерами проекту. ДіаграмаUse Case також служить базою для створення детальних сценаріїв тестування, що забезпечують покриття всіх вимог до функціональності продукту. Відповідно, вона є ключовою частиною документації, що використовується впродовж усього життєвого циклу розробки проекту, сприяючи якісній підготовці та реалізації програмного рішення [15].

Актори та їх ролі

1. Автор блогу: основний контент-генератор, який створює статті.
2. Читач: кінцевий користувач, що інтерактивно переглядає статті.
3. Адміністратор: відповідає за модерацію та управління системою включаючи користувацькі акаунти та контент.

Випадки використання:

– створення аккаунту: описує процес реєстрації користувача;

– автентифікація: вхід користувачів у систему для доступу до персоналізованого досвіду;

– публікація статті: процес, який дозволяє авторам публікувати оригінальний контент на платформі;

– перегляд статей: можливість для читачів читати та взаємодіяти з опублікованим контентом;

– коментування: взаємодія читачів з контентом через коментарі;

– модерація контенту: адміністративні дії з нагляду за публікаціями;

– управління профілем: функціональність, що дозволяє користувачам

керувати особистими даними та налаштуваннями.

Деталізація випадків використання:

– додавання сертифікату: дає можливість авторам блогу додати відомості про свої кваліфікації та досягнення;

– перевірка даних: важливий процес верифікації інформації при створенні акаунту;

– управління блогом: включає здійснення важливих адміністративних операцій та забезпечення належної роботи платформи [16] .

Процеси та Діяльності:

– перегляд сторінок інших користувачів: забезпечує додатковий рівень взаємодії та спільноти в межах платформи;

– пошук по темі статті/імені автора: важлива функція, яка полегшує навігацію користувачів і пошук необхідного контенту;

– видалення статті/аккаунту: надає користувачам та адміністраторам контроль над контентом та особистою інформацією.

Використання Use Case діаграми дає змогу структуровано підійти до аналізу функціональних вимог платформи та ясно визначити границі системи. Ця діаграма відіграє ключову роль у процесі проектування, сприяючи кращому розумінню потреб користувачів та шляхів їх реалізації. Завдяки ній команда розробників отримує високорівневе бачення системи, що допомагає у плануванні та пріоритизації робочих процесів, забезпечуючи тим самим злагоджене впровадження проекту [17].

Висновок до розділу 2

В обранні технологій для моєї онлайн-платформи освітнього блогу я віддав перевагу Python та фреймворкам Vue.js і Django за їхню гнучкість, швидкість розробки, безпеку та масштабованість. Завдяки своїй читабельності та широкій підтримці спільноти, Python ідеально підходить для бекенду, а Django забезпечує "з коробки" набір інструментів для ефективного управління даними та безпеки. Vue.js забезпечує легкість і інтуїтивність при розробці

динамічних фронтендів. Ця комбінація створює міцну основу для платформи, яка здатна розвиватися та адаптуватися, задовольняючи потреби користувачів сьогодні та в майбутньому.

Use Case діаграма надає високорівневий огляд взаємодій між користувачами та системою, що дозволяє нам краще розуміти та задовольняти їх потреби. Цей інструмент є невід'ємною частиною планування функціональності, допомагаючи гарантувати, що кожна вимога користувача буде врахована та реалізована в продукті. Завдяки цьому я впевнений у створенні продукту, що не тільки відповідає поточним вимогам ринку, але й має потенціал для інновацій та розширення в майбутньому.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ САЙТУ

3.1 Реалізація основного функціоналу процесу реєстрації

Процес реєстрації користувачів на онлайн-платформі для освітнього блогу реалізований за допомогою веб-інтерфейсу (рис. 3.1), розділеного на дві головні частини: інформаційну панель та форму реєстрації. Використовується фреймворк Vue.js для створення реактивного користувацького інтерфейсу.

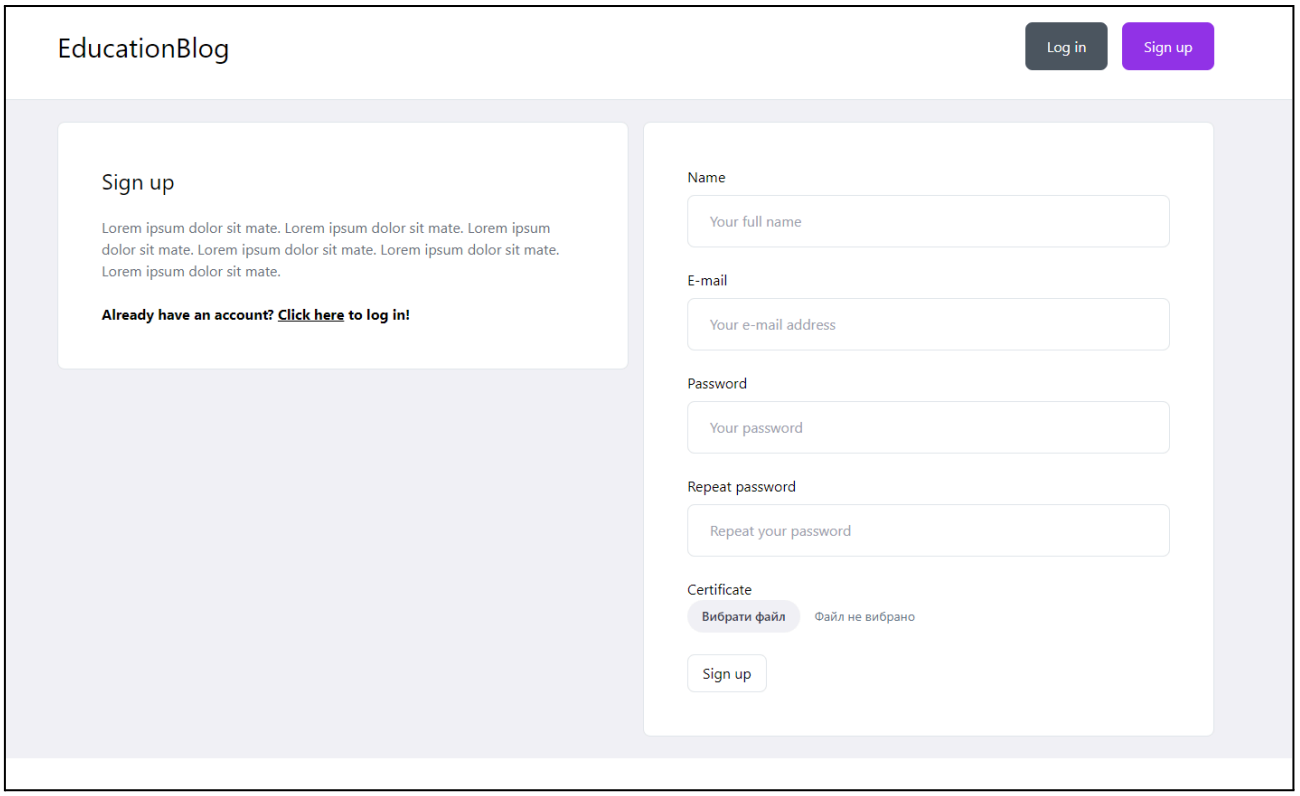


Рисунок 3.1 – Форма реєстрації

HTML Template (Шаблон):

1. Основні розділи: main-left та main-right: ці класи використовуються для стилізації та розміщення двох основних частин інтерфейсу – інформаційної панелі та форми реєстрації відповідно.
2. Форма реєстрації: в полях форми (input) користувач вводить ім'я,

електронну адресу та пароль. Використання директиви v-model забезпечує двостороннє зв'язування даних між формою вводу та даними компоненту.

3. Кнопка відправлення форми: При натисканні на кнопку Sign up виконується метод submitForm, який запускає процес валідації та відправлення даних на сервер.

4. Відображення помилок: блок з v-if="errors.length > 0" відображається тільки тоді, коли в масиві errors є помилки. Це дозволяє динамічно інформувати користувача про помилки введення даних.

JavaScript (логіка компоненту)

1. Дані компоненту (data):

– form: об'єкт, який зберігає дані, введені користувачем. Включає поля для імені, електронної пошти, пароля та повторення пароля;

– errors: масив, що використовується для зберігання повідомлень про помилки, які можуть виникнути під час валідації форми.

2. Методи:

– submitForm: метод, що активується при спробі відправити форму. Цей метод відповідає за перевірку введення у полях і, у разі відсутності помилок, відправляє дані на сервер.

3. Валідація форми:

– При активації методу submitForm, спочатку відбувається очищення масиву errors.

– Далі система перевіряє, чи заповнені всі обов'язкові поля (чи введений email, чи введене ім'я, чи введений пароль та чи паролі співпадають)

– Якщо будь-яка з умов не виконується, відповідне повідомлення про помилку додається до масиву errors.

4. Відправлення даних:

– Якщо в масиві errors немає записів, дані готуються до відправлення: Створюється об'єкт FormData. Додаються всі дані з форми, включно з файлом сертифікату, якщо він є.

– Виконується відправлення даних на сервер за допомогою axios

з налаштуванням відповідного заголовка для коректної обробки файлів [18].

Обробка відповіді від сервера:

Після відправлення запиту, в залежності від відповіді сервера (response.data.message), виводиться відповідне повідомлення за допомогою системи сповіщень (toastStore.showToast).

Методи компонента:

1. Метод починається з очищення масиву errors, щоб переконатися, що попередні помилки не зберігаються.

```
methods: {
    submitForm() {
        this.errors = []
```

2. Перевірка полів форми на наявність даних. Якщо дані відсутні або не відповідають умовам, до масиву errors додається відповідне повідомлення.

```
          if (this.form.email === '') this.errors.push('Your e-mail is
missing')
            if (this.form.name === '') this.errors.push('Your name is
missing')
         if (this.form.password1 === '') this.errors.push('Your password
is missing')
                   if (this.form.password1 !== this.form.password2)
this.errors.push('The password does not match')
           if (this.$refs.file.files[0] === '') this.errors.push('Your
certificate is missing')
```

3. Якщо масив errors порожній (немає помилок), тоді відбувається відправка форми.

```
        if (this.errors.length === 0) {
            let formData = new FormData()
            formData.append('certificate', this.$refs.file.files[0])
            formData.append('name', this.form.name)
            formData.append('email', this.form.email)
            formData.append('password1', this.form.password1)
            formData.append('password2', this.form.password2)
```

4. Використовується метод POST для відправки даних форми на сервер за вказаною адресою.

```
            axios.post('/api/signup/', formData, {
                headers: { "Content-Type": "multipart/form-data" }
            })
            .then(response => {
                if (response.data.message === 'success') {
                          this.toastStore.showToast(5000, 'The user is
registered. Please log in', 'bg-emerald-500')
                    this.form.email = ''
                    this.form.name = ''
                    this.form.password1 = ''
                    this.form.password2 = ''
                } else {
                         this.toastStore.showToast(5000, 'Something went
wrong. Please try again', 'bg-red-300')
                }
            })
            .catch(error => {
                console.log('error', error)
            })
```

Цей код забезпечує комплексне рішення для процесу реєстрації користувачів, включаючи валідацію даних на стороні клієнта, відправлення даних на сервер та обробку відповідей від сервера.

3.2 Процес створення головної сторінки сайту

Головна сторінка сайту (рис. 3.2) реалізована як в'ю, яке відображає індивідуальні пости та коментарі до них. Сторінка побудована на принципах реактивності та взаємодії з сервером для динамічного вмісту. Компонент включає дві основні зони: центральну для відображення постів і праву колонку для додаткових модулів, що можуть зацікавити користувача. Центральна зона динамічно оновлюється залежно від вибору користувачів чи змін, що вносяться

адміністраторами, завантажуючи вміст без перезавантаження сторінки. Права колонка містить елементи, такі як рекомендації постів, оповіщення, забезпечуючи користувачам швидкий доступ до нових функцій та інформації.

Інтерактивність та висока швидкість відгуку забезпечуються використанням асинхронних запитів та сучасних технологій фронтенду, що створює зручний і привабливий користувацький досвід.

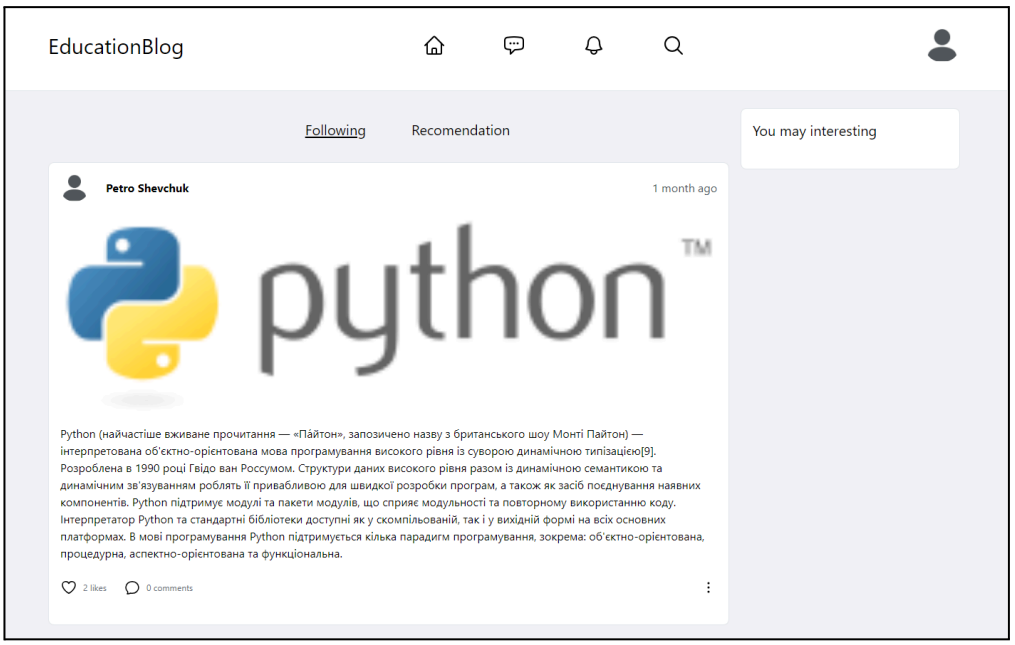


Рисунок 3.2 – Головна сторінка

Код реалізує компонент Vue.js під назвою PostView, який створює головну сторінку для блогу або платформи в стилі соціальної мережі. Ця сторінка динамічно відображає один пост і його пов'язані коментарі. Крім того, вона дозволяє користувачам надсилати нові коментарі безпосередньо на цій сторінці.

Основна частина сторінки (main-center) складається з трьох частин:

– Відображення посту: цей розділ використовує компонент FeedItem для відображення головного посту, якщо він існує (v-if="post.id"). Об'єкт post включає деталі, такі як ідентифікатор посту, що використовується для умовного рендерингу цього компоненту.

– Секція коментарів: нижче посту розташований цикл (v-for), який

рендерить кожен коментар, пов'язаний з постом, використовуючи компонент CommentItem. Кожен коментар передається в компонент за допомогою директиви v-bind:comment="comment", забезпечуючи правильне відображення кожного коментаря з його вмістом та макетом.

– Форма для надсилання коментарів: ця частина включає форму, де користувачі можуть вводити свої думки про пост у текстове поле. Під час подання форми викликається метод submitForm, який надсилає новий коментар на сервер. Метод надсилає дані коментаря на сервер, і при успішному поданні оновлює список коментарів, додаючи новий коментар на початок (unshift). Також оновлюється кількість коментарів і очищається поле введення для нових коментарів.

На правій стороні сторінки (main-right) розташований компонент під назвою MayInteresting, який, ймовірно, пропонує інші пости або контент, який може бути цікавим користувачеві. Цей компонент відокремлений від основного потоку контенту та служить додатковим інтерактивним елементом для користувача.

Частина скрипта компоненту відповідає за отримання даних поста та управління взаємодією з користувачем:

– отримання посту: після монтування компонента викликається метод getPost. Він робить HTTP GET запит для отримання посту, використовуючи його ідентифікатор з параметрів URL. Отриманий пост зберігається у властивості даних post компонента;

– обробка подання форми: метод submitForm керує поданням нових коментарів. Він відправляє вміст текстового поля на сервер як частину нового запиту на коментар. У разі успіху новий коментар додається до масиву comments і оновлюється лічильник коментарів.

Загалом, цей компонент інтегрує отримання даних, динамічний рендеринг і обробку введення користувача, створюючи повністю інтерактивну сторінку перегляду поста, де користувачі можуть читати контент, переглядати коментарі та взаємодіяти, залишаючи свої коментарі.

Код головної сторінки:

1. Компонент називається PostView і експортується для використання в інших частинах застосунку.

```
export default (await import('vue')).defineComponent({
    name: 'PostView',
```

2. Підключаються дочірні компоненти MayInteresting, FeedItem та CommentItem, які будуть використовуватися всередині цього компонента.

```
    components: {
        MayInteresting,
        FeedItem,
        CommentItem,
    },
```

3. Оголошуються початкові дані компонента. Post: об'єкт, який містить інформацію про пост, включаючи коментарі. Body: рядок, який буде використовуватися для введення тексту нового коментаря.

```
    data() {
        return {
            post: {
                id: null,
                comments: [],
            },
            body: '',
        }
    },
```

4. Метод mounted викликається після того, як компонент змонтований. Викликається метод getPost, щоб отримати дані про пост з сервера.

```
    mounted() {
        this.getPost()
    },
```

5. Метод getPost. Виконує HTTP GET-запит для отримання даних про

пост за ідентифікатором, що зберігається в маршруті ($route.params.id). У випадку успішного запиту, дані поста зберігаються в this.post. Якщо виникає помилка, вона виводиться в консоль.

```
    methods: {
        getPost() {
            axios
                .get(`/post/${this.$route.params.id}`)
                .then(response => {
                    console.log('data from post', response.data)
                    this.post = response.data.post
                })
                .catch(error => {
                    console.log('error', error)
                })
        },
```

6. Метод submitForm. Виконує HTTP POST-запит для додавання нового коментаря до поста. Дані коментаря передаються в тілі запиту (body). У випадку успішного запиту, новий коментар додається на початок масиву коментарів поста, оновлюється кількість коментарів (comments_count), а поле вводу очищується. Якщо виникає помилка, вона виводиться в консоль.

```
        submitForm() {
            console.log('submitForm', this.body)
            axios
                .post(`/post/${this.$route.params.id}/comment/`, {
                    'body': this.body
                })
                .then(response => {
                    console.log('data', response.data)
                    this.post.comments.unshift(response.data)
                                        this.post.comments_count  =
parseInt(this.post.comments_count + 1)
                    this.body = ''
                })
                .catch(error => {
                    console.log('error', error)
```

```
        })
```

Цей підрозділ підкреслює практичну реалізацію головної сторінки веб-сайту з динамічним вмістом і можливістю користувацької взаємодії, підкріпленої ефективними методами завантаження даних та обробки подій, що забезпечує гнучке та відгукове середовище для кінцевих користувачів.

3.3 Обмін повідомленнями

Компонент системи обміну повідомленнями створений для забезпечення комунікації між користувачами на платформі. Він включає інтерфейс для відображення і вибору бесід, а також форму для надсилання нових повідомлень. Компонент побудований з використанням Vue.js та взаємодіє з сервером через Axios [19] для отримання та відправлення даних.

1. Отримання і відображення бесід:

Функція getConversations() запитує список доступних бесід користувача з сервера. При успішному отриманні даних, перша бесіда в списку автоматично встановлюється як активна:

```
  getConversations() {
    axios.get('/chat/')
        .then(response => {
            this.conversations = response.data;
            if (this.conversations.length) {
                this.activeConversation = this.conversations[0].id;
                this.getMessages();
            }
        })
        .catch(error => {
            console.log('error', error);
        })
}
```

2. Отримання повідомлень для активної бесіди:

Функція getMessages() виконує запит на сервер для отримання повідомлень вибраної бесіди. Дані зберігаються у activeConversation.messages:

```
getMessages() {
    axios.get(`/chat/${this.activeConversation}/`)
        .then(response => {
            this.activeConversation.messages = response.data.messages;
        })
        .catch(error => {
            console.log('error', error);
        })
}
```

3. Відправлення нового повідомлення:

Метод submitForm() обробляє відправлення нового повідомлення. Після відправлення, повідомлення додається до локального списку повідомлень у бесіді, щоб користувач міг бачити його без перезавантаження сторінки:

```
submitForm() {
       axios.post(`/chat/${this.activeConversation.id}/send/`, { body: this.body })
        .then(response => {
            this.activeConversation.messages.unshift(response.data);
                this.body = '';   // Очищення поля введення після відправлення
        })
        .catch(error => {
            console.log('error', error);
        })
}
```

Ці методи разом забезпечують основний функціонал системи обміну повідомленнями, дозволяючи користувачам вибирати бесіди, переглядати повідомлення та надсилати нові повідомлення в реальному часі (рис 3.3).

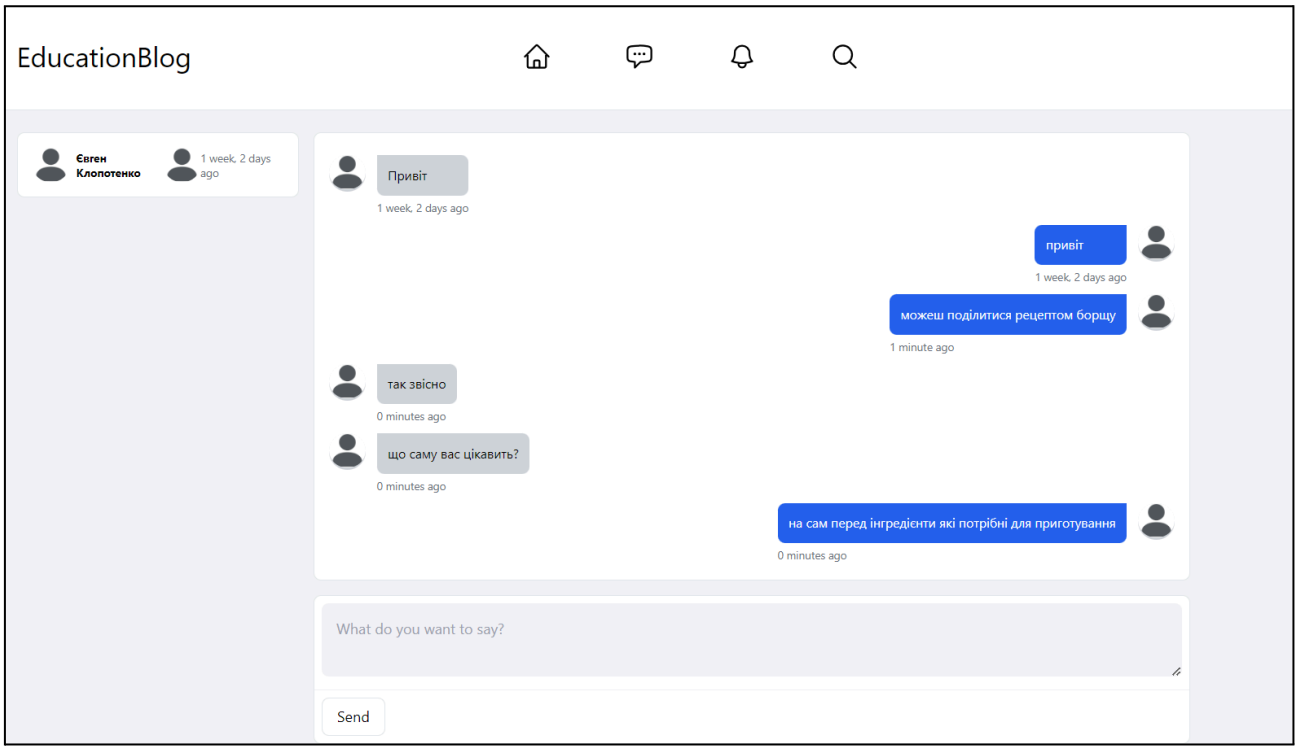


Рисунок 3.3 – Обмін повідомленнями

3.4 Реалізація Сповіщення

Компонент сповіщень на платформі реалізований для інформування користувачів про важливі події, такі як коментарі до постів або оновлення від друзів. Розробка виконана за допомогою Vue.js, який динамічно відображає сповіщення і взаємодіє з сервером через Axios [20].

Сповіщення відображаються у центральній колонці інтерфейсу. Кожне сповіщення містить текст та кнопку "Read more", що дозволяє користувачу отримати додаткову інформацію або перейти до відповідного контенту. Якщо сповіщення відсутні, виводиться повідомлення про їх відсутність, що покращує користувацький досвід (рис 3.4).

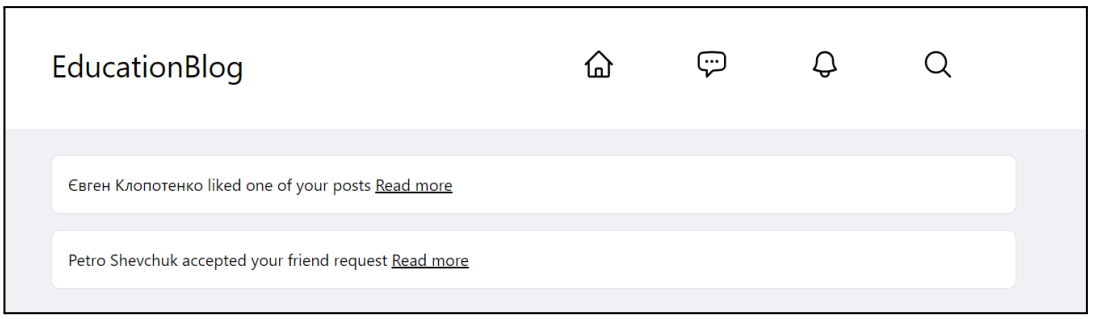


Рисунок 3.4 – Сповіщення

Метод `getNotifications()` виконується при завантаженні компонента, запитуючи список актуальних сповіщень з сервера. Отримані дані зберігаються у масиві `notifications`. При кліку на сповіщення, метод `readNotification(notification)` викликається для позначення сповіщення як прочитаного та відправляє запит на сервер. Відповідь сервера визначає подальші дії: залежно від типу сповіщення, може відбутися перенаправлення на сторінку поста або профіль користувача.

Цей підхід не тільки підтримує актуальність інформації для користувачів але й сприяє більшій взаємодії на платформі, оскільки користувачі можуть швидко реагувати на події, що їх зацікавили.

Методи компонента :

1. Даний метод виконує HTTP GET-запит до /notifications/ для отримання списку повідомлень. У випадку успішного запиту, отримані дані зберігаються в змінну this.notifications. Якщо виникає помилка, вона виводиться в консоль.

```
methods: {
    getNotifications() {
        axios
            .get('/notifications/')
            .then(response => {
                console.log(response.data)
                this.notifications = response.data
            })
            .catch(error => {
                console.log('error', error)
            })
    },
```

2. Наведений метод нище виконує HTTP POST-запит для позначення повідомлення як прочитаного. Якщо повідомлення стосується лайка або коментаря до поста, перенаправляє користувача на сторінку перегляду поста. В інших випадках перенаправляє користувача на сторінку друзів. Якщо виникає помилка, вона виводиться в консоль.

```
    async readNotification(notification) {
        console.log('readNotification', notification.id)
        await axios
            .post(`/notifications/read/${notification.id}/`)
            .then(response => {
                console.log(response.data)
                 if (notification.type_of_notification == 'post_like' ||
notification.type_of_notification == 'post_comment') {
                      this.$router.push({name: 'postview', params: {id:
notification.post_id}})
                } else {
                    console.log(notification)
                       this.$router.push({name: 'friends', params: {id:
notification.created_for_id}})
                }
            })
            .catch(error => {
                console.log('error', error)
```

3.5 Сторінка Пошук

Компонент сторінки пошуку дозволяє користувачам вводити запити у текстове поле та отримувати результати пошуку у вигляді списків користувачів та постів. Після введення запиту та натискання кнопки "Search", форма активує метод `submitForm`, який використовує Axios для відправлення запиту на сервер. Сервер обробляє запит і повертає дані, які включають користувачів та пости, відповідні запиту.

Кожен користувач відображається зі своїм аватаром та ім'ям, яке є посиланням на їх профіль, а також інформацією про кількість друзів і постів. Пости представлені через компонент `FeedItem`, який показує деталі кожного поста. Якщо пошук не дає результатів, користувачам відображається повідомлення про відсутність результатів пошуку. Цей компонент спрощує навігацію по сайту та допомагає користувачам ефективно знаходити потрібну інформацію (рис 3.5, 3.6).

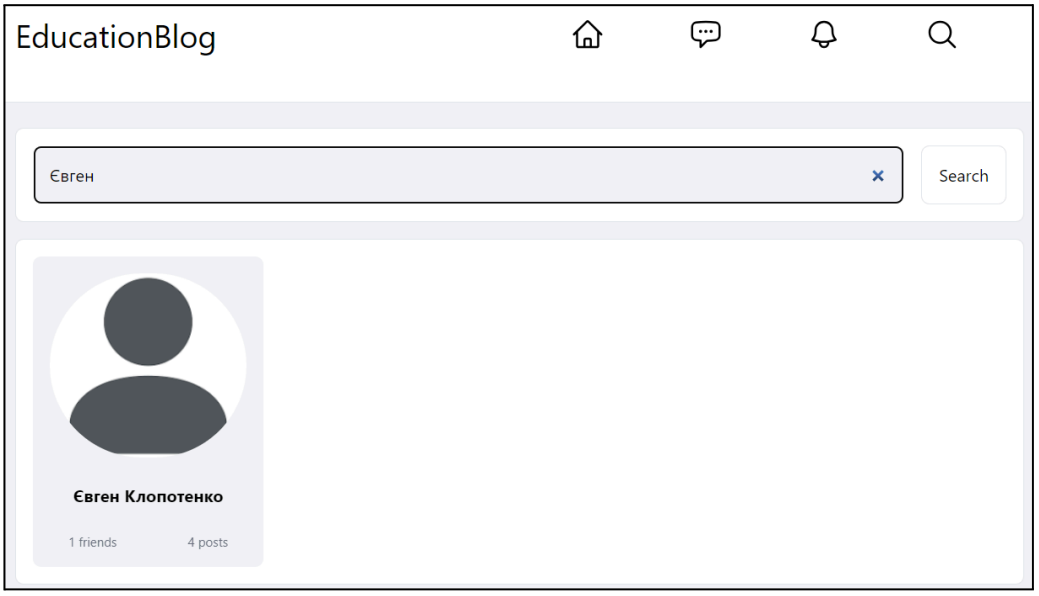


Рисунок 3.5 – Пошук за ім'ям користувача

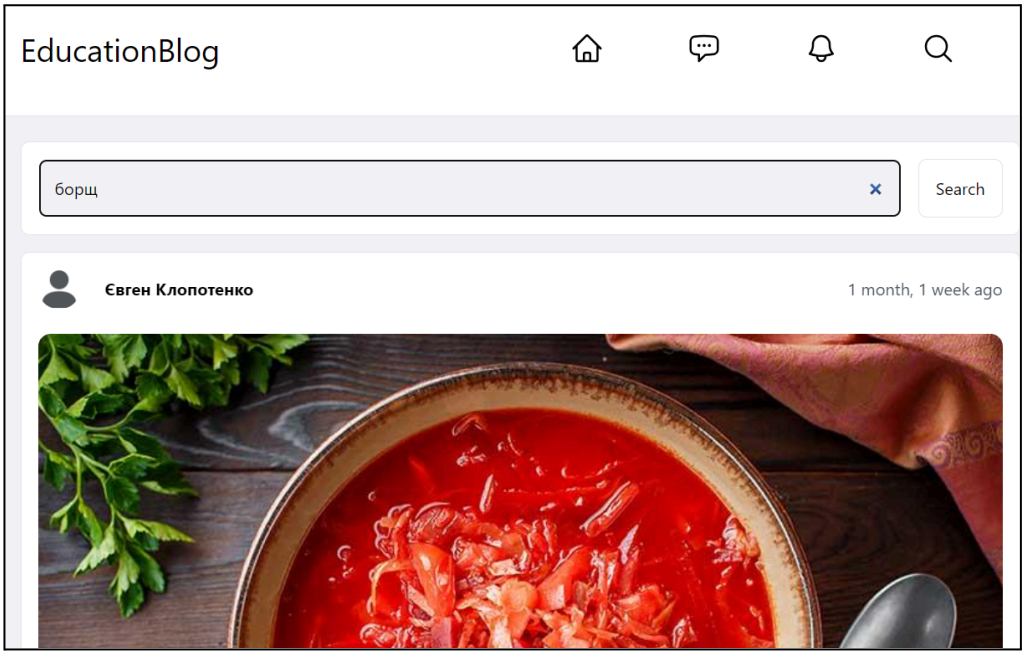


Рисунок 3.6 – Пошук за темою поста

Метод реалізації пошуку:

1. Імпортуються компоненти MayInteresting та FeedItem, а також бібліотека для HTTP запитів axios.

```
import MayInteresting from '../components/MayInteresting.vue'
import axios from 'axios'
import FeedItem from '../components/FeedItem.vue'
```

2. Оголошується компонент SearchView та підключаються імпортовані дочірні компоненти.

```
export default {
    name: 'SearchView',
    components: {
        MayInteresting,
        FeedItem,
    },
    data() {
        return {
            query: '',
            users: [],
            posts: [],
        }
    },
```

3. Метод submitForm Виконує HTTP POST-запит для виконання пошуку за вказаним запитом. У випадку успішного запиту, отримані дані користувачів і постів зберігаються відповідно в змінні this.users та this.posts. Якщо виникає помилка, вона виводиться в консоль.

```
    methods: {
        submitForm() {
            console.log('submitForm', this.query)
            axios
                .post('/search/', {
                    'query': this.query
                })
                .then(response => {
                    console.log('response:', response.data)
                    this.users = response.data.users
                    this.posts = response.data.posts
                })
                .catch(error => {
                    console.log('error:', error)
                })
```

3.6 Профіль користувача

Компонент профілю користувача на платформі включає зручний інтерфейс для перегляду особистої інформації, такої як аватар, ім'я, кількість друзів і постів. Користувачі можуть переглядати пости та взаємодіяти з іншими користувачами через надсилання запитів на дружбу або прямі повідомлення. Функціональність дозволяє також публікувати нові пости, включаючи зображення та текст, через форму, яка відправляє дані на сервер.

Користувачі, які мають доступ до свого профілю, можуть редагувати інформацію про себе або вийти з системи, використовуючи відповідні кнопки для входу в режим редагування або для виходу. Ці дії забезпечуються через API-запити, які виконують зміни на сервері і оновлюють локальний стан користувача на клієнті, що забезпечує зручну взаємодію на платформі (рис 3.7).

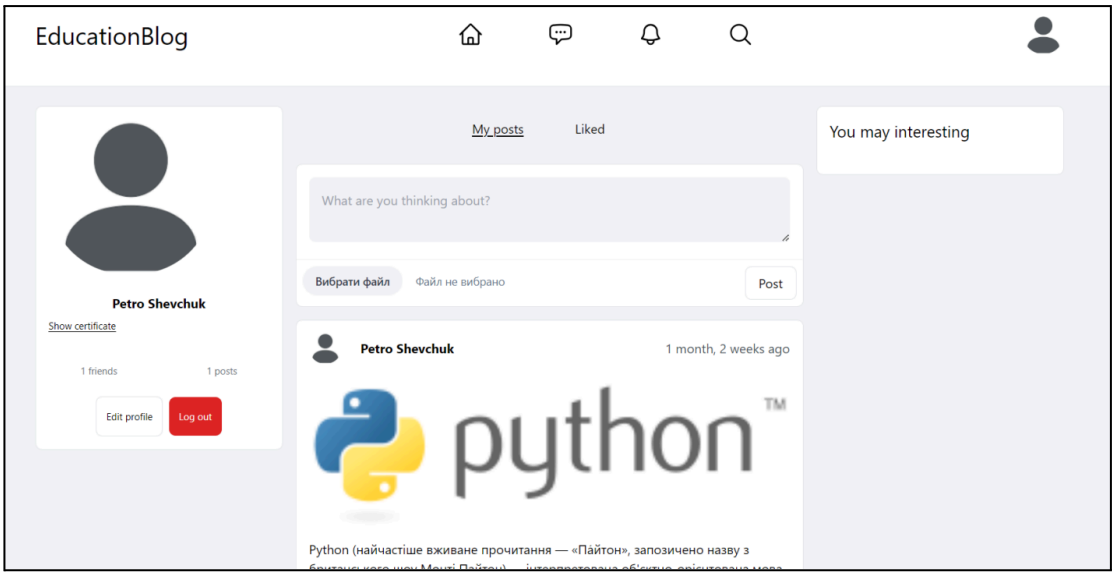


Рисунок 3.7 – Профіль користувача

Метод реалізації:

1. Налаштування компоненту. Ця частина коду відповідає за базове налаштування компоненту FeedView. Вона включає ініціалізацію необхідних реактивних станів через setup():

```
export default {
    name: 'FeedView',
```

```
setup() {
    const userStore = useUserStore()
    const toastStore = useToastStore()

    return {
        userStore,
        toastStore
    }
},
components: {
    MayInteresting,
    FeedItem
},
```

2. Дані компоненту. Цей блок data() визначає реактивні дані, які використовуються в компоненті для збереження інформації про дописи, користувача, та інші взаємодії. Всі ці дані можуть бути змінені і спостережувані Vue, що дозволяє динамічно оновлювати інтерфейс користувача, коли стан цих даних змінюється:

```
data() {
    return {
        posts: [],
        user: {
            id: null
        },
        body: '',
        liked_posts: [],
        can_send_friendship_request: null,
    }
},
```

3. Життєвий цикл та спостереження. Цей блок коду керує реакціями компоненту на важливі події життєвого циклу та зміни у маршрутизації (routing). Використання хука mounted() гарантує, що дані для стрічки будуть запитані відразу після того, як компонент буде візуально доданий до DOM.

```
mounted() {
    this.getFeed()
},

watch: {
    '$route.params.id': {
        handler: function() {
            this.getFeed()
        },
        deep: true,
        immediate: true
    }
},
```

4. Методи компоненту. Цей блок методів визначає поведінку компоненту у відповідь на дії користувача, такі як видалення постів, відправлення повідомлень, запитів на дружбу, та інші. Методи дозволяють виконувати бізнес-логіку та взаємодіяти з сервером через API для здійснення змін у даних або стані сесії.

```
methods: {
    deletePost(id) {
        this.posts = this.posts.filter(post => post.id !== id)
        this.user.posts_count -= 1
    },
            sendDirectMessage() {
    },
    sendFriendshipRequest()
    },

    getFeed() {
    },
    submitForm() {
    },
    logout() {
        this.userStore.removeToken()
        this.$router.push('/login')
```

3.7 Сертифікат

Компонент CertificateView у веб-додатку забезпечує зручний перегляд сертифікатів користувачів. При відкритті цієї сторінки, компонент автоматично виконує запит до сервера для отримання зображення сертифіката відповідно до ідентифікатора користувача, який передається через параметри маршруту. Отримані дані відображаються у вигляді зображення на сторінки.

Крім того, для підвищення зручності користування, на сторінці розміщена кнопка "Back", яка дозволяє користувачам легко повернутися до попередньої сторінки без перезавантаження, використовуючи вбудовані можливості маршрутизатора Vue.js. Це забезпечує плавний та інтуїтивно зрозумілий досвід перегляду, що є важливим для підтримки професіоналізму та довіри на платформі (рис 3.8).

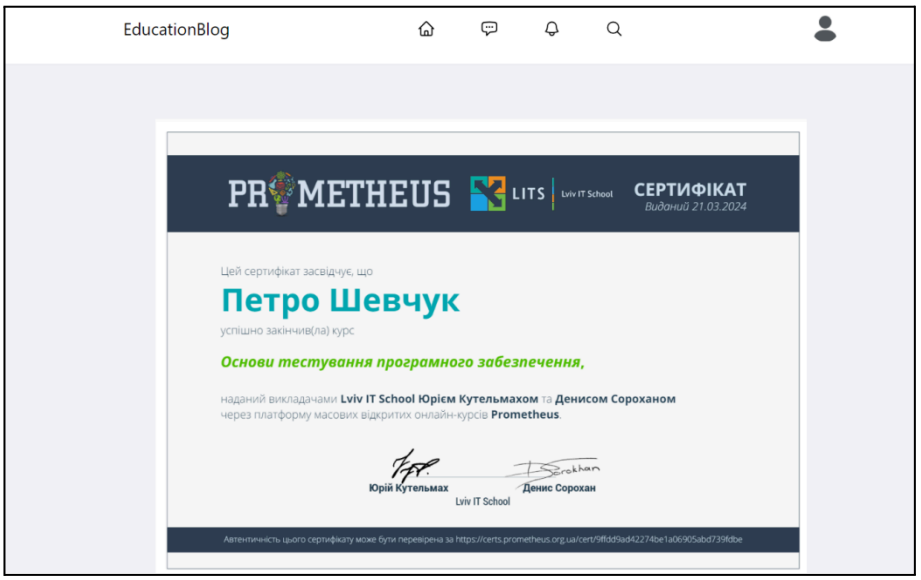


Рисунок 3.8 – Сертифікат

Метод реалізації:

1. Імпорт бібліотеки axios та оголошується компонент CertificateView.

```
import axios from 'axios'
export default {
    name: 'CertificateView',
```

2. Оголошуються початкові дані компонента: user: об'єкт, який містить

інформацію про користувача, зокрема його id.

```
data() {
    return {
        user: {
            id: null,
        }
    }
},
```

3. Метод mounted викликається після того, як компонент змонтований. Викликається метод getFeed, щоб отримати дані сертифіката з сервера.

```
mounted() {
    this.getFeed()
},
```

4. Метод getFeed. Виконує HTTP GET-запит для отримання даних сертифіката користувача за ідентифікатором, що зберігається в маршруті ($route.params.id). У випадку успішного запиту, отримані дані зберігаються в змінну this.user. Якщо виникає помилка, вона виводиться в консоль.

```
methods: {
    getFeed() {
            axios
                .get(`/api/certificate/${this.$route.params.id}/`)
                .then(response => {
                    console.log('data', response.data)
                    this.user = response.data
                })
                .catch(error => {
                    console.log('error', error)
                })
```

Висновок до розділу 3

У цьому розділі кваліфікаційної роботи була розглянута практична реалізація онлайн-платформи для освітнього блогу. Завдяки детальному аналізу

кожного аспекту функціональності платформи, від розробки архітектури системи до конкретних імплементацій модулів, було досягнуто значного прогресу у створенні високоефективної та користувацьки орієнтованої веб-платформи.

Ключові аспекти, які були реалізовані, включають:

– Розробку архітектури системи, яка забезпечує надійність, масштабованість та легкість управління;

– інтерфейс користувача, що включає розробку реактивних компонентів для забезпечення плавної взаємодії;

– бекенд-реалізація, яка підтримує всю необхідну логіку обробки даних і взаємодії з БД;

– модулі пошуку, профілів користувачів, та системи обміну повідомленнями, які відіграють важливу роль у соціальній взаємодії;

– система сповіщень, яка інформує користувачів про важливі події;

– функціонал перегляду та валідації сертифікатів, що додає додатковий рівень довіри до професійних якостей учасників платформи.

Кожен з цих елементів був ретельно спланований та реалізований з метою створення ефективного, безпечного та користувацьки привабливого досвіду для кінцевих користувачів. Результатом є платформа, яка не тільки відповідає поточним технологічним стандартам, але й має потенціал для розширення та адаптації до майбутніх потреб і вимог.

ВИСНОВОК

У результаті виконання кваліфікаційної роботи, яка зосереджена на розробці онлайн-платформи для освітнього блогу, підкреслює успішну інтеграцію теоретичних знань та практичних навичок у сфері веб-розробки. Вибір технологій, таких як Python, Django для бекенду та Vue.js для фронтенду, був вмотивований їх гнучкістю, широкими можливостями для масштабування та зручністю у використанні. Ці технології дозволили ефективно реалізувати всі необхідні функції платформи, включаючи реєстрацію та аутентифікацію користувачів, публікацію та коментування статей.

Протягом роботи було ретельно сплановано та впроваджено стратегію тестування, що включала розробку тест-плану та тест-кейсів для забезпечення надійності та ефективності платформи. Результати тестування підтвердили високу якість системи та її готовність до впровадження та використання кінцевими користувачами.

Висновок підкреслює, що завдяки глибокому розумінню вимог користувачів та ринкових умов, платформа була не тільки розроблена відповідно до сучасних тенденцій веб-розробки, але й має потенціал для подальшого розвитку та оновлення відповідно до змінюваних вимог користувачів та технологічних нововведень.

Загалом, реалізація цього проекту демонструє успішне поєднання теоретичних знань та практичного застосування навичок у розробці надійних, масштабованих та зручних для користувача веб-рішень, що відкриває широкі перспективи для подальшої кар'єри в галузі інформаційних технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Team, A.,& Team, A. (2023, February 11). The importance of blogging for online learning and education. AIContentfy. веб-сайт. URL:https://aicontentfy.com/en/blog/importance-of-blogging-for-online-learning-and-education (Дата звернення: 20.02.2024)

2. Teachmint. (2023, March 16). Importance of educational blogs. Teachmint Blogs.веб-сайт. URL: https://blog.teachmint.com/importance-of-educational-blogs/ (Дата звернення: 22.02.2024)

3. Kitsoft. (n.d.). Main Site - ТОП-12 освітніх майданчиків з безкоштовними онлайн-курсами. веб-сайт. URL: https://eukraine.org.ua/ua/news/top-12-osvitnih-majdanchikiv-z-bezkoshtovnimi-onlajn-kursami (Дата звернення: 28.02.2024)

4. Laba, Ж. (2023, May 10). Вибрали для вас 23 найкращі освітні платформи з онлайн-навчанням. UA-LABA. веб-сайт. URL:https://laba.ua/blog/3542-23-naykrashchi-osvitni-platformi (Дата звернення: 10.03.2024)

5. Prometheus. (2024, May 1). Prometheus – Найбільша платформа онлайн-курсів в Україні. веб-сайт. URL:https://prometheus.org.ua/ (Дата звернення: 15.03.2024)

6. EdEra. (2024, April 29). EdEra — студія онлайн-освіти. веб-сайт. URL: https://ed-era.com/ (Дата звернення: 20.03.2024)

7. «На Урок». (n.d.). Освітній проект «На Урок» для вчителів. На Урок! веб-сайт. URL: https://naurok.com.ua/

8. Освіторія. (2023, October 10). «Освіторія Медіа» - онлайн-медіа про освіту та виховання дітей в Україні. Освіторія Медіа. веб-сайт. URL: https://osvitoria.media/ (Дата звернення: 25.03.2024)

9. Smyrnov, I. (2022, January 26). Розробка веб додатків з використанням

Python і Django. Webcase. веб-сайт. URL: https://webcase.com.ua/uk/blog/razrabotka-veb-prilozhenij-s-ispolzovaniem-python-i-django/ (Дата звернення: 02.04.2024)

10. Welcome to Python.org. (2024, April 9). Python.org. веб-сайт. URL: https://www.python.org/ (Дата звернення: 09.04.2024)

11. Ruby on rails. (n.d.). Ruby on Rails. веб-сайт. URL: https://rubyonrails.org/ (Дата звернення: 12.04.2024)

12. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. (n.d.). React. веб-сайт. URL: https://uk.legacy.reactjs.org/ (Дата звернення: 20.02.2024)

13. Вступ | Vue.js. (n.d.). веб-сайт. URL: https://ua.vuejs.org/guide/introduction.html (Дата звернення: 17.04.2024)

14. Django with Vue.js - Using Django (2023,March 4). Django Forum. веб-сайт. URL:https://forum.djangoproject.com /t/django-with-vue-js/1926 /2 (Дата звернення: 23.04.2024)

15. Каграманова, Ю. (2022, October 27). Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. DOU. веб-сайт. URL: https://dou.ua/forums/topic/40575/ (Дата звернення: 25.04.2024)

16. Maxym, Z. (2023, March 5). Варіанти використання та сценарії (Use Cases and Scenarios). Maxym Zosym. веб-сайт. URL: https://www.maxzosim.com/use-cases-and-scenarios/ (Дата звернення: 28.04.2024)

17. draw.io - free flowchart maker and diagrams online. (n.d.). diagrams.net. веб-сайт. URL: https://app.diagrams.net/ (Дата звернення: 01.05.2024)

18. Починаючи | Axios Docs. (n.d.). веб-сайт. URL: https://axios-http.com/uk/docs/intro (Дата звернення: 05.05.2024)

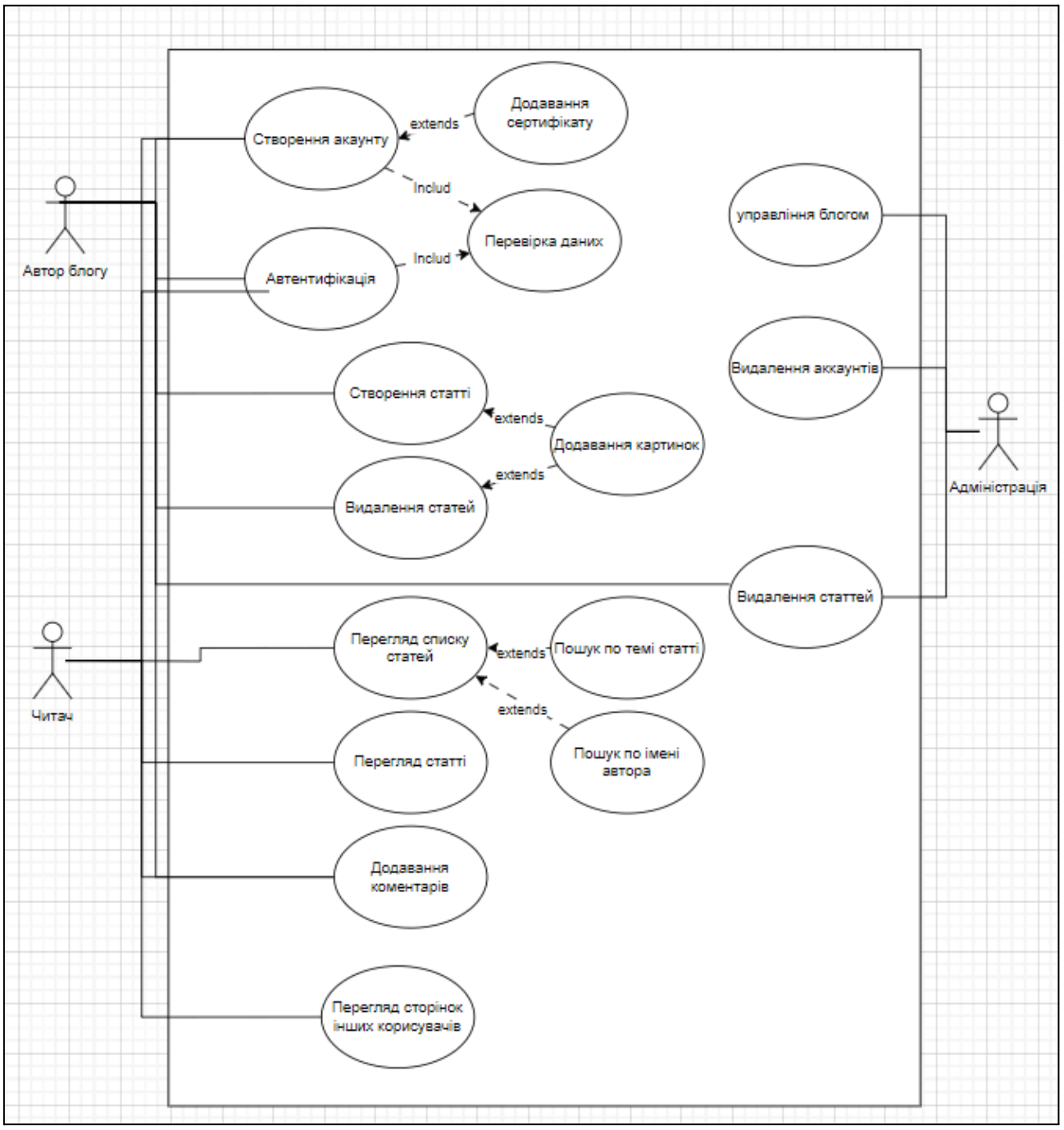
19. npm: vue-axios. (n.d.). Npm. веб-сайт. URL: https://www.npmjs.com/package/vue-axios (Дата звернення: 11.05.2024)

20. Au-Yeung, J. (2023, July 26). How to Use Axios with Vue.js. Telerik Blogs. веб-сайт. URL: https://www.telerik.com/blogs/how-to-use-axios-vue (Дата звернення: 17.05.2024)

ДОДАТКИ

Додаток А

Use Case Діаграма



метадані

Заголовок

Розробка онлайн-платформи для освітнього Блогу

Автор

Шевчук П. Т. Науковий керівник / Експерт

підрозділ

King Danylo University

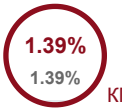
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	10

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

КП 2

25
Довжина фрази для коефіцієнта подібності 2

7287
Кількість слів

59460
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/394/%D0%A0%D1%83%D0%B4%D0%B8%D0%B9%20%D0%90%D0%BD%D0%B4%D1%80%D1%96%D0%B9%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	13	0.18 %
2	https://habr.com/ru/post/350750/	9	0.12 %
3	https://habr.com/ru/post/350750/	8	0.11 %
4	https://iamsapankumar.blogspot.com/2018/12/react-native-text-view-button.html	7	0.10 %
5	https://iamsapankumar.blogspot.com/2018/12/react-native-text-view-button.html	7	0.10 %
6	https://www.positronx.io/react-mern-stack-crud-app-tutorial/	6	0.08 %