Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка мікроблогінгової соціальної мережі для підтримки освітніх ініціатив»

Виконав: студент 4 курсу
групи 4ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Григоренко Д. П.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Рейда О.М.
(прізвище та ініціали)

Рецензент: д.т.н., проф. каф. ОТ Мартинюк Т.Б.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ _____
« 10 » червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – «Інформаційні технології»
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Григоренку Дмитру Петровичу

1. Тема роботи – розробка мікроблогінгової соціальної мережі для підтримки освітніх ініціатив.

Керівник роботи: Рейда Олександр Миколайович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2023.

3. Вихідні дані до роботи: середовище розробки IntelliJ IDEA, мови розробки: Java та JavaScript, операційна система – Windows 10, система управління базою даних PostgreSQL; алгоритми авторизації користувачів та захисту їх даних.

4. Зміст текстової частини: вступ; аналіз розвитку мікроблогінгових соціальних мереж для освітніх ініціатив та постановка задач дослідження; розробка структури та алгоритмів програмного продукту; розробка програмних модулів мікроблогінгової соціальної мережі для підтримки освітніх ініціатив; тестування програми; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: порівняльний аналіз аналогів; структура інтерфейсу програмного засобу; графічний інтерфейс застосунку; модель роботи системи; структура бази даних; блок-схеми алгоритмів роботи застосунку; тестування застосунку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Рейда О.М., к.т.н., доцент кафедри ПЗ	13.03.2024	03.06.2024

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору теми та постановка задач дослідження	14.03.24 - 20.03.24	Вик.
2	Аналіз і вибір середовища розробки	21.03.24 - 31.03.24	Вик.
3	Розробка структури бази даних та проектування інтерфейсу користувача	01.04.24 - 10.04.24	Вик.
4	Розробка основних програмних модулів	11.04.24 - 20.04.24	Вик.
5	Інтеграція модулів та розробка мікроблогінгової функціональності	21.04.24 - 10.05.24	Вик.
6	Тестування розробленого вебзастосунку	11.05.24 - 20.05.24	Вик.
7	Оформлення матеріалів до захисту БДР	21.05.24 - 30.05.24	Вик.

Студент _____ Д. П. Григоренко
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ О. М. Рейда
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 76 сторінок формату А4, на яких є 31 рисунків, 8 таблиць, 4 додатки, список використаних джерел містить 23 найменування.

У бакалаврській дипломній роботі проведено аналіз сучасного стану мікроблогінгових соціальних мереж у контексті освітніх ініціатив. Виконано порівняльний аналіз аналогів, визначено їх переваги та недоліки, обґрунтовано необхідність створення нової платформи.

Розроблено мікроблогінгову соціальну мережу для підтримки освітніх ініціатив. Обрано мову програмування Java та фреймворк Spring Boot для реалізації проекту. Розроблено структуру бази даних, інтерфейс користувача та основні програмні модулі. Проведено тестування програмних модулів системи та розроблено інструкцію користувача.

Отримані результати можна використовувати для подальшого розвитку платформ підтримки освітніх ініціатив. Робота може стати основою для майбутніх досліджень у цій галузі.

Ключові слова: мікроблогінг, соціальна мережа, освітні ініціативи, вебзастосунок.

ANNOTATION

The bachelor thesis consists of 76 pages of A4 format, on which there are 31 figures, 8 tables, 4 appendices, the list of used sources contains 23 titles.

The bachelor's diploma thesis analyzed the current state of microblogging social networks in the context of educational initiatives. A comparative analysis of analogues was performed, their advantages and disadvantages were determined, and the necessity of creating a new platform was substantiated.

A microblogging social network has been developed to support educational initiatives. The Java programming language and the Spring Boot framework were chosen for the implementation of the project. The database structure, user interface and main software modules have been developed. Testing of the system's software modules was conducted and a user manual was developed.

The obtained results can be used for further development of support platforms for educational initiatives. The work may provide a basis for future research in this area.

Keywords: microblogging, social network, educational initiatives, web application.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У глибоко змінному світі, де кожен день народжується нова інформація, важливість обміну знаннями та ідеями стає не тільки питанням досягнення успіху, але й ключовим фактором для збереження конкурентоспроможності та розвитку [1]. Освітні ініціативи, що відзначаються глибоким впливом, вимагають платформ, які не лише сприяють швидкому та ефективному поширенню інформації, але й створюють сприятливі умови для активного діалогу та співпраці між учасниками.

У цьому контексті розробка мікроблогінгової соціальної мережі, спрямованої на задоволення освітніх потреб, виступає як відповідь на сучасні виклики [2]. Мікроблогінг, як форма комунікації, дозволяє користувачам публікувати короткі, але інформативні повідомлення, що можуть містити текст, зображення або відео. Це відкриває двері до широкого обміну освітнім контентом, де кожен може одночасно виступати як учитель, ділитися власними знаннями, та як учень, здобувати нові відкриття та інсайти.

У змішаному навчанні і спільній підготовці до занять та проектів мікроблогінг стає ефективним інструментом. Він сприяє обговоренню та обміну думками, стимулює активну участь учасників у навчальному процесі та розвиває навички критичного мислення.

Проте розробка такої платформи вимагає більш глибокого розуміння специфіки освітнього процесу, потреб освітян та учнів, а також технологічних можливостей, що можуть задовольнити ці потреби. Ця бакалаврська робота покликана дослідити ці аспекти та запропонувати інноваційне рішення, яке сприятиме розвитку освітніх ініціатив через мікроблогінгову соціальну мережу.

Тому розробка веборієнтованого застосунку для підтримки освітніх ініціатив є не лише актуальною, але й необхідною для створення більш ефективного та інтерактивного навчального середовища. Використання передових технологій та інноваційних підходів до дизайну та функціональності

дозволить створити платформу, яка стане незамінним інструментом у руках освітян та учнів по всьому світу.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є забезпечення покращення комунікативних можливостей у сфері освіти за допомогою мікроблогінгової соціальної мережі, яка сприятиме обміну інформацією та ресурсами, підтримуватиме ефективну взаємодію між учасниками освітнього процесу, та допоможе у досягненні навчальних та професійних цілей, стимулюючи активний обмін знаннями, сприяючи співпраці між студентами, викладачами та іншими учасниками освітнього середовища, ініціюючи створення простору, де кожен може вільно ділитися ідеями, досвідом та ресурсами, що сприятиме зростанню як особистості, так і освітнього середовища в цілому.

Основними задачами дослідження є:

- провести аналіз існуючих мікроблогінгових систем та визначити їхні можливості для освітніх ініціатив;
- покращити метод формування постів;
- розробка інтерфейсу користувача;
- розробка бази даних, яка ефективно зберігатиме користувацькі дані, освітній контент, забезпечуючи швидкий доступ та високу продуктивність;
- розробка модуля автентифікації та реєстрації користувачів для підтримки додаткових функцій та безпеки платформи;
- розробка модуля публікації та перегляду, який дозволить користувачам публікувати короткі повідомлення, сприяти обміну ідеями та знаннями, а також підтримувати освітні дискусії;
- розробка модуля пошуку публікацій за тегами, що забезпечить користувачам можливість знаходити пости за тегами;

- розробити програмні компоненти та системи: інтерфейс, модуль аутентифікації, модуль публікації та перегляду, модуль пошуку за тегами постів;
- провести експериментальні дослідження розроблених засобів.

Об’єктом дослідження є процес розробки мікроблогінгової соціальної мережі для підтримки освітніх ініціатив.

Предметом дослідження є методи і засоби розробки програмного продукту, принципи програмування мови Java та засоби роботи з інтегрованим середовищем розробки IntelliJ IDEA.

Методи дослідження. Під час розробки мікроблогінгової соціальної мережі для підтримки освітніх ініціатив використовувалися такі методи: методи лінгвістичного аналізу, методи синтаксичного та семантичного аналізу, методи машинного навчання для класифікації та кластеризації даних, методи візуалізації соціальних мереж для аналізу взаємодій, методи моделювання даних для ефективного зберігання та обробки інформації, методи прототипування для розробки та тестування прототипів користувацького інтерфейсу, збору зворотного зв’язку від користувачів та подальшого вдосконалення дизайну та функціональності.

Новизна отриманих результатів. Подальшого розвитку отримав метод формування постів, який, на відміну від існуючих, використовує лінгвістичний аналіз для вивчення структури тексту та частоти вживаних слів, а також синтаксичний, семантичний та тематичний аналіз для формування постів за вподобаннями. Це дозволяє підбирати найбільш релевантні пости для користувачів на основі їх інтересів і уподобань, що підвищує рейтинг мікроблогінгової системи, покращує якість взаємодії між користувачами та сприяє збільшенню активності та утриманню цільової аудиторії.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає в тому, що розроблені модулі системи включають інтерфейс користувача, модуль автентифікації та реєстрації, модуль публікації та перегляду, а також модуль пошуку публікацій за тегами. Такі модулі забезпечують зручне та ефективне використання мікроблогінгової платформи,

яка надає користувачам можливість легко знаходити та обмінюватися освітніми матеріалами, а також підтримувати освітні дискусії. Повністю розроблена мікроблогінгова соціальна мережа для освітніх ініціатив дозволяє покращити взаємодію користувачів та сприяє підвищенню їхньої залученості завдяки ефективному використанню алгоритмів рекомендацій та захисту даних.

Особистий внесок. Всі наукові результати, що викладені у бакалаврській дипломній роботі, були отримані безпосередньо автором роботи.

Апробація результатів роботи. Матеріали роботи доповідались на таких конференціях: LIII Науково-технічна конференція факультету інформаційних технологій та комп’ютерної інженерії (2024) [3]; Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)» [4].

Публікації. За тематикою дослідження опубліковано 2 наукових праці в матеріалах конференцій.

Технічне завдання наведено в додатку А.

1 АНАЛІЗ РОЗВИТКУ МІКРОБЛОГІНГОВИХ СОЦІАЛЬНИХ МЕРЕЖ ДЛЯ ОСВІТНІХ ІНІЦІАТИВ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану питання розробки мікроблогінгових вебсистем

У сучасному світі цифрові технології стали не просто необхідністю, але й ключовим фактором, що перетинає всі сфери життя. Вони трансформують наші підходи до комунікації, навчання, розваг та спілкування, змінюючи наше сприйняття та реалізацію цих процесів. Особливо важливим є їхній вплив на освіту, де цифрові інструменти відкривають безліч нових можливостей для навчання, спілкування та обміну знаннями. Відтак, мікроблогінгові соціальні мережі, такі як Twitter, стають не просто платформами для розважальних цілей, але і ключовими інструментами у сфері освіти.

Завдяки своїй природі, мікроблогінгові платформи забезпечують швидкий та зручний спосіб обміну короткими повідомленнями. Це робить їх ідеальними для передачі короткої, але важливої інформації, що є важливим фактором у сфері освіти. Викладачі можуть миттєво повідомляти студентів про зміни в розкладі, надавати посилання на цікаві статті або навіть проводити короткі опитування або тести, щоб перевірити рівень засвоєння матеріалу. Такий активний обмін інформацією сприяє підвищенню ефективності навчання та розвитку спільної освітньої спільноти.

Мікроблогінг у сфері освіти відіграє ключову роль у створенні динамічного та інтерактивного середовища навчання, що відповідає сучасним вимогам швидкості та доступності інформації. Його переваги виявляються у кількох аспектах.

По-перше, мікроблогінг забезпечує швидкий доступ до інформації. Викладачі та студенти можуть миттєво публікувати та отримувати оновлення, що дозволяє оперативно ділитися важливими оголошеннями, змінами у розкладі чи навіть найсвіжішими науковими відкриттями. Це створює можливість для всіх бути в курсі найсвіжіших подій у світі освіти.

Другий аспект полягає в тому, що мікроблогінг сприяє колаборативному навчанню [5]. Він відкриває можливості для спільної роботи та обговорень, де учасники можуть взаємодіяти та обмінюватися ідеями. Це може включати спільні проекти, групові дискусії або навіть міжнародні співпраці, що розширює горизонти навчання за межі класної кімнати.

Також, мікроблогінг підтримує активне навчання. Використання мікроблогінгу для обговорення та аналізу тем залучає студентів до активного навчального процесу. Вони стають не лише пасивними споживачами інформації, а активними учасниками, які критично мислять та висловлюють свої думки.

Ще одним важливим аспектом є, мікроблогінг надає гнучкість у навчанні. Студенти можуть вчитися у власному темпі та в зручний для них час, переглядати матеріали та взаємодіяти з іншими учасниками навчального процесу, коли це найбільш зручно для них. Це робить навчання більш доступним та індивідуалізованим.

І нарешті, мікроблогінг сприяє розвитку професійних зв’язків. Він може служити платформою для налагодження професійних контактів та мереж. Студенти та викладачі можуть зв’язуватися з колегами, експертами у своїй галузі та іншими освітніми закладами, що сприяє професійному розвитку та обміну знаннями.

Зважаючи на те, що не всі мікроблогінгові платформи можуть ідеально відповідати потребам освіти, є настійна потреба в розробці спеціалізованих інструментів, спрямованих на оптимізацію освітніх процесів. Основною проблемою стандартних мікроблогінгових платформ є їхні обмежені можливості в управлінні контентом, що ускладнює роботу викладачів з додаванням та оновленням навчальних матеріалів. Тому ключовим завданням для розробників програмного забезпечення є створення інструментів, які максимально відповідають освітнім потребам.

Одним із напрямків у розробці спеціалізованих інструментів для освіти є створення модулів для управління контентом. Ці модулі мають спрощувати процес додавання та оновлення навчальних матеріалів для викладачів. Вони

повинні надавати зручний інтерфейс, що дозволяє швидко та ефективно завантажувати різноманітний навчальний контент, включаючи тексти, відео, аудіо та інші формати.

Додатково, розробники повинні приділити увагу розробці аналітичних інструментів, які дозволяють відстежувати активність студентів та їхні досягнення [6]. Це можуть бути інструменти для аналізу даних про відвідуваність курсів, виконані завдання, результати тестів тощо. Збір та аналіз цих даних дозволить викладачам краще розуміти потреби та інтереси студентів, а також адаптувати навчальні програми та методики відповідно до їх потреб.

Отже, розробка програмного забезпечення для мікроблогінгових соціальних мереж, спрямованих на освітні ініціативи, є критично важливою для сучасного освітнього середовища. Вона відкриває нові можливості для навчання та спілкування, роблячи освіту більш доступною та ефективною. З огляду на постійний розвиток цифрових технологій, важливо бути в курсі останніх розробок у цій галузі, щоб забезпечити актуальність та ефективність освітніх ресурсів.

1.2 Порівняльний аналіз аналогів

У сучасному світі, де освітній процес невпинно трансформується під впливом цифровізації, мікроблогінгові соціальні мережі відіграють вирішальну роль у формуванні нових освітніх парадигм. Вони не лише сприяють зручному обміну інформацією та ідеями між учасниками освітнього процесу, але й відкривають широкі можливості для колаборативного навчання, професійного розвитку та громадської активності [7]. Розробка програмних засобів, які б підтримували ці процеси, стає ключовим завданням для IT-спеціалістів, освітян та всіх, хто зацікавлений у прогресивному розвитку освіти.

До найбільш популярних аналогів у цій сфері відносяться:

- Edmodo;
- Schoology.

Edmodo (див. рисунок 1.1) була однією з перших мікроблогінгових платформ, спеціально розроблених для освітніх потреб [8]. Ця платформа дозволяла вчителям та учням створювати віртуальні класні кімнати, де вони могли спілкуватися, обмінюватися матеріалами та виконувати завдання.

Edmodo надавала інструменти для розподілу квізів та завдань, управління комунікацією між учнями, колегами та батьками, а також дозволяла інтегрувати зовнішні освітні ресурси.

Переваги Edmodo полягали у створенні безпечного середовища для освітнього спілкування та можливості залучення батьків до навчального процесу їх дітей.

Однак, платформа мала обмеження, такі як відсутність прямого спілкування між учнями та обмежені аналітичні можливості для вчителів. Ці аспекти варто врахувати при розробці нових освітніх платформ, щоб забезпечити більш гнучке та ефективне середовище для навчання.

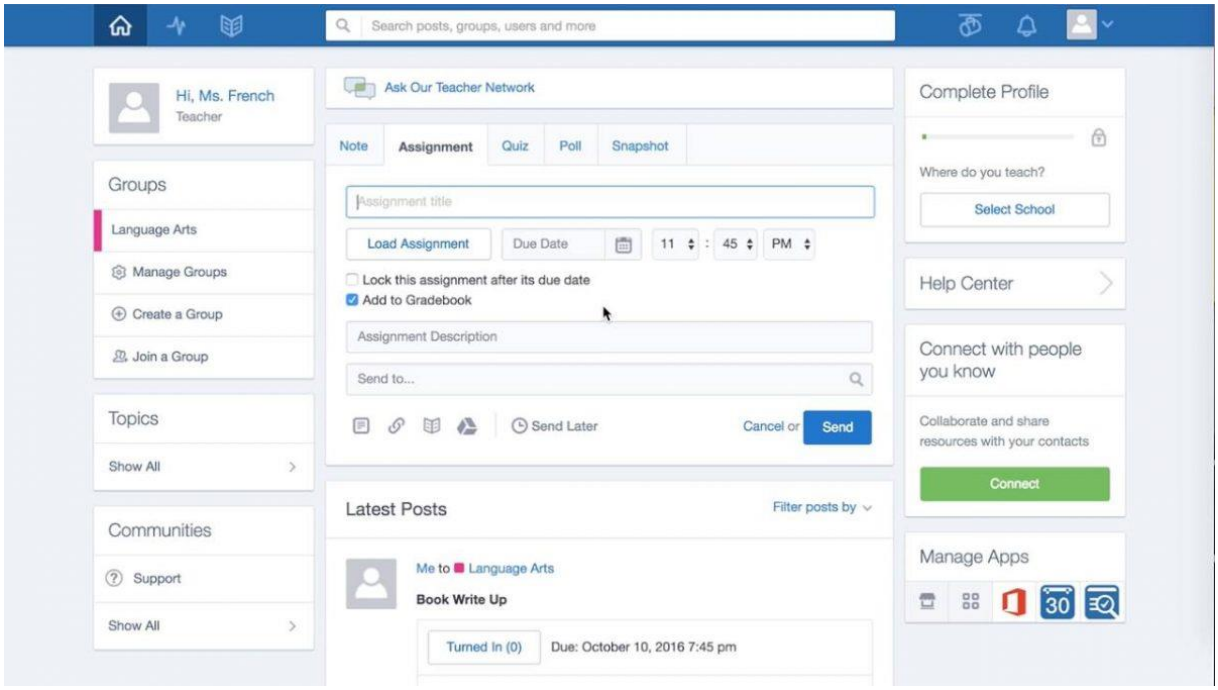

Рисунок 1.1 – Інтерфейс додатку Edmodo

Edmodo припинила свою діяльність у вересні 2022 року. Однак, її функціональність та вплив на освітній процес залишаються важливими

прикладами для розробки нових освітніх платформ. Розроблювана система може взяти на озброєння кращі практики Edmodo, адаптувавши їх під сучасні вимоги та додавши нові функції для підтримки освітніх ініціатив.

Застосунок Schoology (див. рисунок 1.2) являє собою комплексну освітню платформу, яка інтегрує функції соціальної мережі з інструментами управління навчальним процесом. Її основна перевага полягає у забезпеченні взаємодії між учасниками освітнього процесу, включаючи вчителів, учнів та батьків, та створенні сприятливого середовища для колаборативного навчання [9].

Особливістю Schoology є її гнучкість та адаптивність до різних освітніх моделей, що дозволяє використовувати платформу як у традиційних класних кімнатах, так і в онлайн-навчанні. Це досягається завдяки широкому спектру інструментів, які допомагають вчителям створювати навчальні плани, розподіляти завдання та відстежувати успішність учнів.

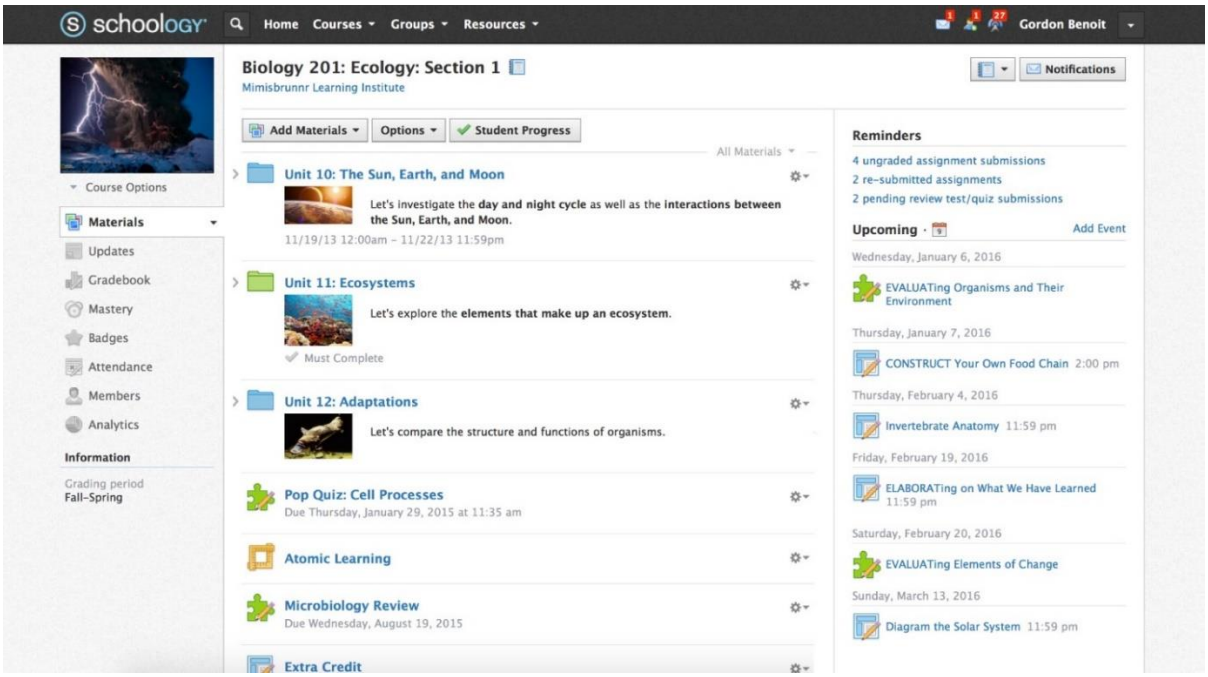

Рисунок 1.2 – Інтерфейс додатку Schoology

Проте, Schoology має і свої виклики. Наприклад, хоча базова версія платформи безкоштовна, деякі розширені функції вимагають підписки. Також, новим користувачам може знадобитися час, щоб звикнути до інтерфейсу та

повноцінно використовувати всі можливості платформи.

Розробка програмних засобів мікроблогінгової соціальної мережі для підтримки освітніх ініціатив потребує поєднання експертизи в галузі соціальних мереж, розробки програмного забезпечення та освітніх технологій. Це передбачає створення інтерактивної платформи, яка сприяє обміну ідеями, знаннями та ресурсами серед освітньої спільноти.

Внутрішня система розробки повинна включати в себе інструменти для створення, редагування та візуалізації коротких повідомлень, що спрямовані на підтримку навчання та розвитку. Крім того, важливо розглянути можливості інтеграції з існуючими освітніми платформами та сервісами, щоб забезпечити максимальний вплив і користь для користувачів.

Інтерфейсна система повинна бути спроектована з урахуванням потреб освітньої громадськості, з використанням простих та інтуїтивно зрозумілих інструментів для навігації та взаємодії з платформою. Технологія мікроблогінгу може бути поєднана з функціоналом відстеження прогресу, обговорення та спільного розв’язання завдань, що дозволить створити стимулююче та підтримуюче середовище для навчання та співпраці.

Результати порівняння аналогів приведено у таблиці 1.1.

Таблиця 1.1 – Порівняльні характеристика

Характеристика	Edmodo	Schoology	LearnNook
Можливість управління повідомленнями	1	1	1
Інтелектуальні рекомендації на основі алгоритмів уподобань	0	0	1
Можливість оцінювання постів	1	0	1
Аналіз активності користувачів та візуалізація даних	0	1	1
Можливість розширеного пошуку публікацій за тегами	0	0	1
Сумарний коефіцієнт	2	2	5

Після ретельного аналізу таблиці порівняльних характеристик стає очевидним, що LearnNook виявився найбільш ефективним і комплексним рішенням серед аналогів, таких як Edmodo та Schoology. Розроблюваний продукт перевершує конкурентів у всіх ключових аспектах, що були оцінені.

Найперше, LearnNook має вищу оцінку у всіх важливих категоріях, включаючи можливість управління повідомленнями та персоналізовані рекомендації на основі машинного навчання, а також можливість розширеного пошуку за тегами постів. У порівнянні з Edmodo та Schoology, які мають деякі обмеження у функціоналі, LearnNook відзначається більшою універсальністю та гнучкістю.

Друге, LearnNook показує високий рівень функціоналу відстеження активності користувачів та візуалізація даних, що робить його хорошим інструментом для освітніх ініціатив. Ця можливість дозволяє користувачам не лише спілкуватися та обмінюватися інформацією, але й ефективно відстежувати активності серед інших користувачів та популяризацію використання тем постів.

Отже, на підставі аналізу порівняльних характеристик стає очевидним, що LearnNook виявився найбільш повним і функціональним серед своїх аналогів, випереджаючи Edmodo та Schoology. Висока оцінка у всіх аспектах свідчить про те, що він може стати цінним інструментом для підвищення якості освіти та підтримки освітніх ініціатив. Впровадження LearnNook може виявити значний вплив на розвиток освітніх ініціатив, забезпечуючи користувачам інноваційні і ефективні засоби для навчання.

1.3 Аналіз методів розробки мікроблогінгової соціальної мережі

У процесі розробки програмних засобів мікроблогінгової соціальної мережі для підтримки освітніх ініціатив, важливо ретельно проаналізувати різноманітні підходи до створення такого продукту. Кожен з методів має свої переваги та недоліки, які необхідно врахувати для ефективного вирішення поставленої задачі.

Створення власної платформи для мікроблогінгової соціальної мережі є одним із можливих та досить перспективних методів розв’язання завдання. Цей підхід відкриває широкі можливості для команди розробників, надаючи повний контроль над функціоналом та архітектурою системи. Власна платформа дозволяє гнучко налаштовувати сервіс, адаптуючи його під конкретні потреби освітніх ініціатив.

Основною перевагою цього підходу є максимальна гнучкість у розробці та підтримці продукту. Розробник може вільно визначати функціонал, архітектуру та дизайн платформи, забезпечуючи індивідуальний підхід до кожного аспекту розробки. Це дозволяє ефективно впроваджувати нові функції та вдосконалення, відповідаючи на зміни в потребах користувачів.

Однак, слід зазначити, що розробка власної платформи може бути часо- та ресурсомісткою. Вона вимагає значних зусиль у початковому етапі, починаючи від проектування та розробки і закінчуючи тестуванням та впровадженням. Крім того, такий процес може забирати значну кількість людських ресурсів та фінансових витрат, що потребує уважного планування та керування. В цілому, створення власної платформи для мікроблогінгової соціальної мережі є варіантом з великим потенціалом для успіху, особливо якщо освітні ініціативи мають унікальні потреби та вимоги. Цей підхід надає можливість реалізувати продукт, що ідеально відповідає специфіці цільової аудиторії, та забезпечує гнучкий та індивідуальний підхід до розробки.

Використання відкритих рішень та готових платформ для створення соціальних мереж є ще одним можливим методом, який варто розглянути. Цей підхід передбачає використання вже існуючих API (інтерфейсів програмування застосунків) та фреймворків, які розроблені та підтримуються спільнотою.

Переваги такого підходу досить очевидні. По-перше, він може значно зменшити час розробки та витрати, оскільки команда може скористатися готовими рішеннями та компонентами, замість того щоб створювати їх з нуля. Це дозволяє прискорити випуск продукту на ринок та забезпечити конкурентні переваги. Крім того, використання відкритих рішень може надати доступ до

широкого спектру готових функцій та можливостей, які вже випробувані та відшліфовані спільнотою розробників. Це може включати такі компоненти, як системи аутентифікації, системи керування контентом, інструменти аналітики тощо.

Однак, варто враховувати, що цей підхід може обмежити гнучкість та індивідуальні можливості розробленої платформи. Використання готових рішень може призвести до того, що деякі аспекти продукту можуть бути обмежені функціональними можливостями цих рішень. Крім того, існує ризик залежності від сторонніх постачальників технологій, який може вплинути на стабільність та безпеку продукту.

Третій можливий метод – це інтеграція розробленого засобу з існуючими соціальними мережами, такими як Facebook, Twitter, або LinkedIn. Це дозволить забезпечити швидку і просту взаємодію з користувачами, оскільки багато людей вже мають облікові записи у цих мережах. Однак, цей підхід може обмежити функціональні можливості та контроль над даними, які збираються та обробляються.

Ураховуючи переваги та недоліки кожного з методів, для розробки програмних засобів мікроблогінгової соціальної мережі для підтримки освітніх ініціатив рекомендується поєднання різних підходів. Наприклад, використання відкритих рішень та фреймворків може забезпечити швидкий старт проєкту, а подальша розробка власної платформи може забезпечити необхідну гнучкість та індивідуальність продукту. Такий комплексний підхід дозволить створити ефективний та конкурентоздатний програмний продукт.

Після ретельного аналізу різних методів розробки програмних засобів мікроблогінгової соціальної мережі для підтримки освітніх ініціатив, виявлено, що найбільш відповідним та перспективним підходом є розробка власної платформи. Цей висновок заснований на кількох ключових факторах.

По-перше, розробка власної платформи дозволить забезпечити повний контроль над продуктом. Це означає, що буде можливість налаштувати функціонал та архітектуру системи відповідно до конкретних потреб освітнього

сегменту, забезпечуючи максимальну гнучкість та індивідуальність продукту.

По-друге, розробка власної платформи дозволить забезпечити його адаптацію до конкретних потреб освітнього сегменту. Освітні ініціативи можуть виявити різні потреби та вимоги до функціоналу соціальної мережі, і розробка власної платформи дозволить врахувати ці потреби та створити інструмент, який оптимально підтримує їх.

Хоча розробка власної платформи може вимагати значних зусиль, проте вона надасть більшу свободу та можливості для подальшого розвитку та вдосконалення продукту. Отже, на основі цих розглядів, розробка власної платформи є оптимальним вибором для реалізації програмних засобів мікроблогінгової соціальної мережі для підтримки освітніх ініціатив.

1.4 Постановка задач дослідження

Проаналізувавши переваги та недоліки існуючих мікроблогінгових соціальних мереж, спрямованих на підтримку освітніх ініціатив, було визначено наступні завдання, які необхідно виконати для розробки програмних засобів:

- провести аналіз існуючих мікроблогінгових систем та визначити їхні можливості для освітніх ініціатив;
- покращити метод формування постів;
- розробка бази даних, яка ефективно зберігатиме користувацькі дані, освітній контент, забезпечуючи швидкий доступ та високу продуктивність;
- розробка модуля автентифікації та реєстрації користувачів для підтримки додаткових функцій та безпеки платформи, який забезпечить надійний контроль доступу та персоналізацію функцій платформи для кожного користувача;
- розробка модуля публікації та перегляду, який дозволить користувачам публікувати короткі повідомлення, сприяти обміну ідеями та знаннями, а також підтримувати освітні дискусії. Модуль буде основним інструментом для створення та обміну контентом на платформі;

- розробка модуля пошуку публікацій за тегами, що забезпечить користувачам можливість знаходити пости за різними параметрами та ключовими словами. Це допоможе користувачам швидко знаходити необхідну інформацію та контент, що відповідає їхнім інтересам;
- розробити програмні компоненти та системи: інтерфейс, модуль аутентифікації, модуль публікації та перегляду, модуль пошуку за тегами постів. Всі ці компоненти будуть інтегровані в єдину систему, забезпечуючи її цілісність та функціональність;
- провести експериментальні дослідження розроблених засобів. Необхідно для оцінки ефективності розроблених рішень, виявлення можливих проблем та їх усунення перед впровадженням платформи в експлуатацію.

1.5 Висновки

У першому розділі було проведено аналіз сучасного стану мікроблогінгових соціальних мереж з орієнтацією на освітні ініціативи. Результати підкреслили важливість розвитку таких мереж для підтримки освітніх процесів, що може значно полегшити доступ до освіти та зробити її більш ефективною. Порівняльний аналіз платформ Edmodo, Schoology та LearnNook підтвердив переваги LearnNook у відношенні повноти та функціональності, що свідчить про його потенційну цінність як інструменту для підвищення якості освіти.

На основі аналізу методів розв'язання задачі було виявлено, що найбільш перспективним підходом є розробка власної платформи для мікроблогінгової соціальної мережі. Цей висновок підкріплюється потребою в індивідуалізованих рішеннях та швидким розвитком цифрових технологій. Отже, дослідження підтвердило актуальність та необхідність розробки програмних засобів мікроблогінгової соціальної мережі для підтримки освітніх ініціатив, з фокусом на платформі LearnNook.

2 РОЗРОБКА СТРУКТУРИ, МОДЕЛІ, МЕТОДУ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМИ

2.1 Розробка структури інтерфейсу програмного засобу

Перед початком детального аналізу окремих елементів структури інтерфейсу мікроблогінгової соціальної мережі для підтримки освітніх ініціатив, варто розглянути процес розробки цієї структури. Розробка структури інтерфейсу є надзвичайно важливим етапом у створенні додатку, оскільки саме на цьому етапі визначається, як користувачі будуть взаємодіяти з платформою. На цьому етапі виникає потреба в грамотному та лаконічному розташуванні різноманітних елементів вебсторінки, щоб забезпечити їхню максимальну зручність для користувачів.

Основна мета цього етапу полягає в створенні структури, яка забезпечить зручну та ефективну навігацію для користувачів. Це включає організацію інформації на вебсторінці та побудову її структури так, щоб користувачі могли легко та швидко знаходити потрібну інформацію. Важливо, щоб всі елементи інтерфейсу були розташовані таким чином, щоб вони були інтуїтивно зрозумілими та доступними.

Під час розробки структури інтерфейсу ключовим завданням є забезпечення зручності та доступності для користувача. Особливу увагу слід приділяти характеристикам цільової аудиторії, яка складається з учасників освітніх ініціатив. Це вимагає глибокого розуміння потреб та очікувань користувачів, що дозволяє створити інтерфейс, який максимально відповідає їхнім вимогам.

При створенні структури інтерфейсу мікроблогінгової соціальної мережі для підтримки освітніх ініціатив, необхідно враховувати основні принципи проектування інтерфейсу користувача. Це включає принципи зручності, доступності та зрозумілості. Важливо, щоб користувачі могли легко орієнтуватися у додатку, знаходити потрібну інформацію та виконувати

необхідні дії з мінімальними зусиллями.

Один з основних принципів при розробці інтерфейсу – це простота. Кожен елемент інтерфейсу повинен мати чітке призначення і бути розташованим таким чином, щоб користувачі могли інтуїтивно зрозуміти його функціонал. Не менш важливим є принцип послідовності, що означає використання однакових елементів та стилів у всьому додатку, щоб користувачі не губилися під час навігації.

Врахування специфіки цільової аудиторії є критично важливим. Оскільки платформа орієнтована на освітні ініціативи, необхідно врахувати, що користувачами будуть як учні та студенти, так і викладачі та освітні адміністратори. Це означає, що інтерфейс повинен бути інтуїтивно зрозумілим як для молодшої аудиторії, так і для дорослих користувачів, можливо, з меншою технічною підготовкою.

Процес розробки інтерфейсу включає кілька етапів: від аналізу вимог користувачів і створення прототипів до тестування та впровадження. Кожен з цих етапів є важливим для досягнення кінцевої мети – створення зручного, функціонального та привабливого інтерфейсу, який сприятиме активному використанню платформи та задоволенню потреб користувачів.

Головна сторінка інтерфейсу мікроблогінгової мережі створена для забезпечення інтуїтивно зрозумілої та ефективної навігації. Вона включає навігаційну панель, яка дозволяє користувачам швидко переходити до ключових розділів, таких як домашня сторінка, створення та перегляд постів, перегляд власного профілю створення нового поста та доступ до налаштувань. Панель інструментів, розташована у верхній частині сторінки, надає швидкий доступ до пошуку та інших важливих функцій, а також можливість приховувати або розкривати навігаційну панель за потреби.

Секція рекомендацій допомагає користувачам відкривати нові групи та контент, що може бути цікавим, сприяючи залученню та взаємодії в освітній спільноті. Функціонал мікроблогінгу, інтегрований у сторінку, дозволяє користувачам ділитися короткими повідомленнями та ідеями, створюючи

динамічне середовище для обміну інформацією та спілкування. Всі ці елементи разом формують зручний та функціональний інтерфейс, який спрощує взаємодію користувачів з платформою та підвищує їхню залученість.

Детальний вигляд інтерфейсу головної сторінки, який включає всі вище перераховані елементи представлено на рисунку 2.1, що демонструє взаємодію користувача з веб-сайтом.

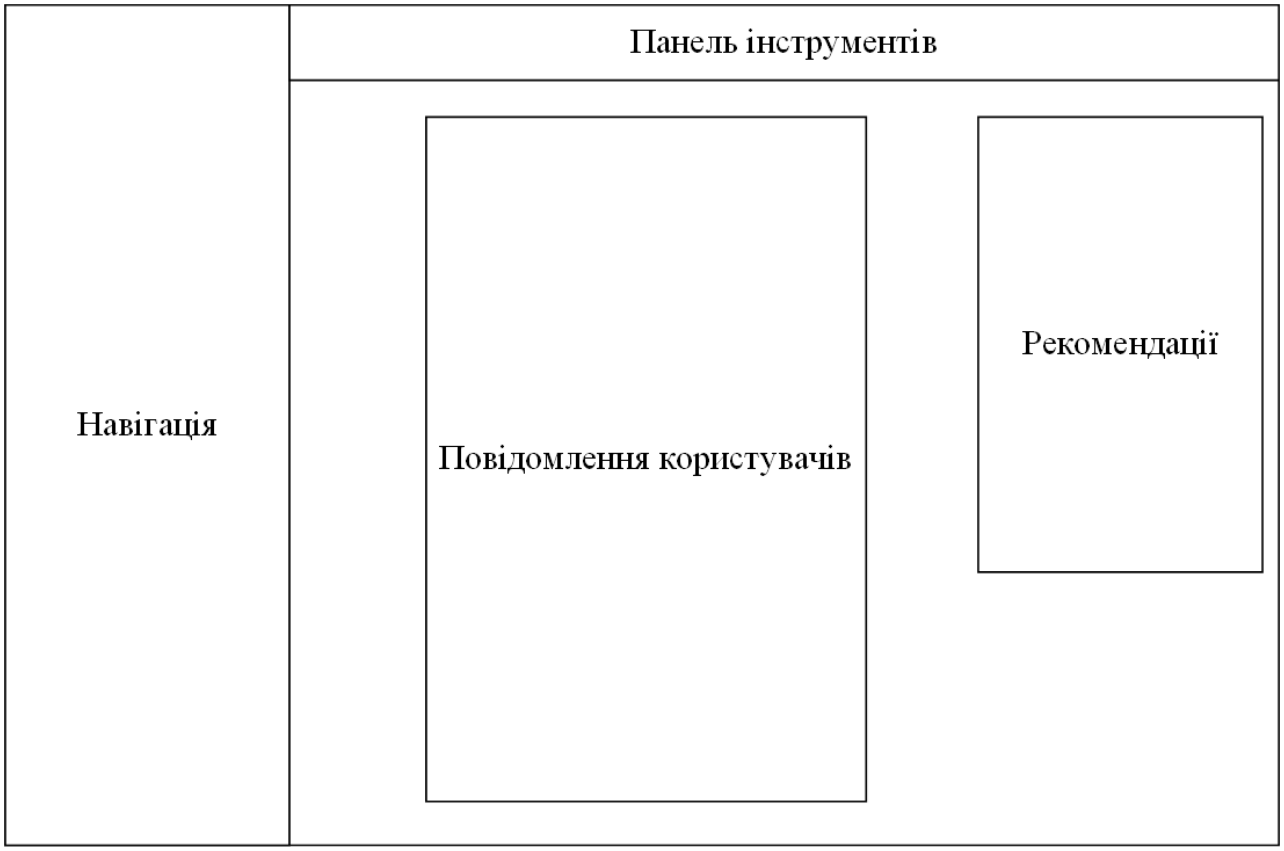

Рисунок 2.1 – Структура інтерфейсу головної сторінки

На сторінці профілю користувача відвідувачі знаходять зручну структуру інтерфейсу, яка забезпечує легкий доступ до всіх важливих аспектів соціальної мережі. Верхня частина сторінки включає панель навігації, яка дозволяє користувачам швидко переходити до основних розділів сайту, таких як домашня сторінка, перегляд та створення постів, перегляд власного профілю та доступ до налаштувань акаунта.

Вище на сторінці розташована панель інструментів, яка надає користувачам можливість управління своїм профілем та взаємодії з іншими

елементами мережі. Ця панель містить інструменти для редагування профілю, відправлення повідомлень та перегляду сповіщень, що робить взаємодію з платформою більш ефективною.

На сторінці також присутня секція з рекомендаціями, яка адаптується до інтересів користувача, пропонуючи контент або інших користувачів, які можуть бути цікавими. Це допомагає користувачам відкривати нові теми та знайомитися з іншими учасниками спільноти.

Структура інтерфейсу сторінки профілю користувача зображена на рисунку 2.2.

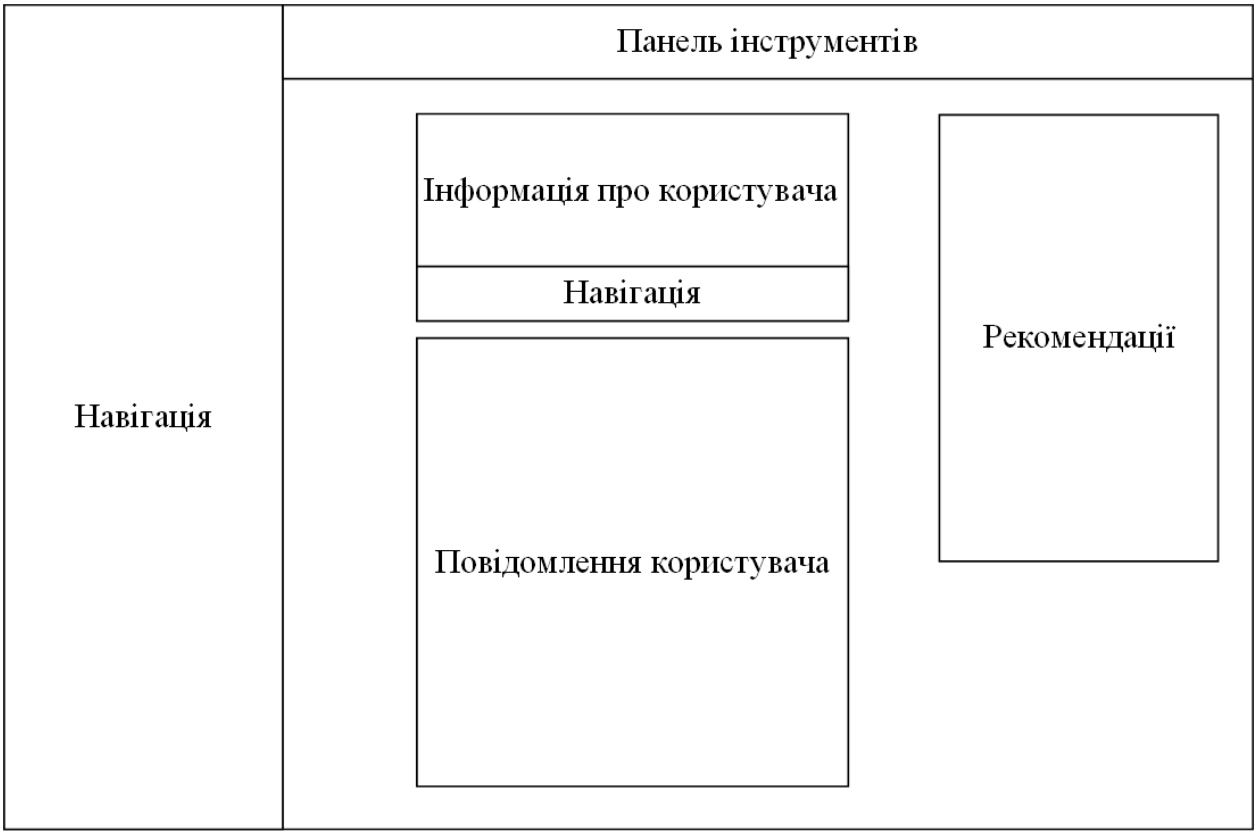

Рисунок 2.2 – Структура інтерфейсу сторінки профілю користувача

Основна інформація профілю, розташована під панеллю інструментів, надає детальний огляд особистих даних користувача, його активності на сайті, включаючи публікації та взаємодію з друзями. Додаткові пункти меню навігації можуть включати доступ до особистих постів користувача, підписників та підписок, які він підписався слідкувати, та інших розділів, які забезпечують

глибший занурення в контент, що його цікавить.

Таким чином, сторінка профілю є центральним місцем для користувача, де він може керувати своєю присутністю в мережі, взаємодіяти з контентом та іншими користувачами, а також отримувати персоналізовані рекомендації, що робить досвід користування платформою більш особистим та залученим.

Також слід розглянути структуру сторінки аналітики, яка є ключовим інструментом для відстеження та аналізу активності користувача, що забезпечує адміністраторам та іншим користувачам доступ до важливих статистичних даних і показників у зручній і зрозумілій формі.

Загальну структуру інтерфейсу сторінка аналітики відображено на рисунку 2.3.

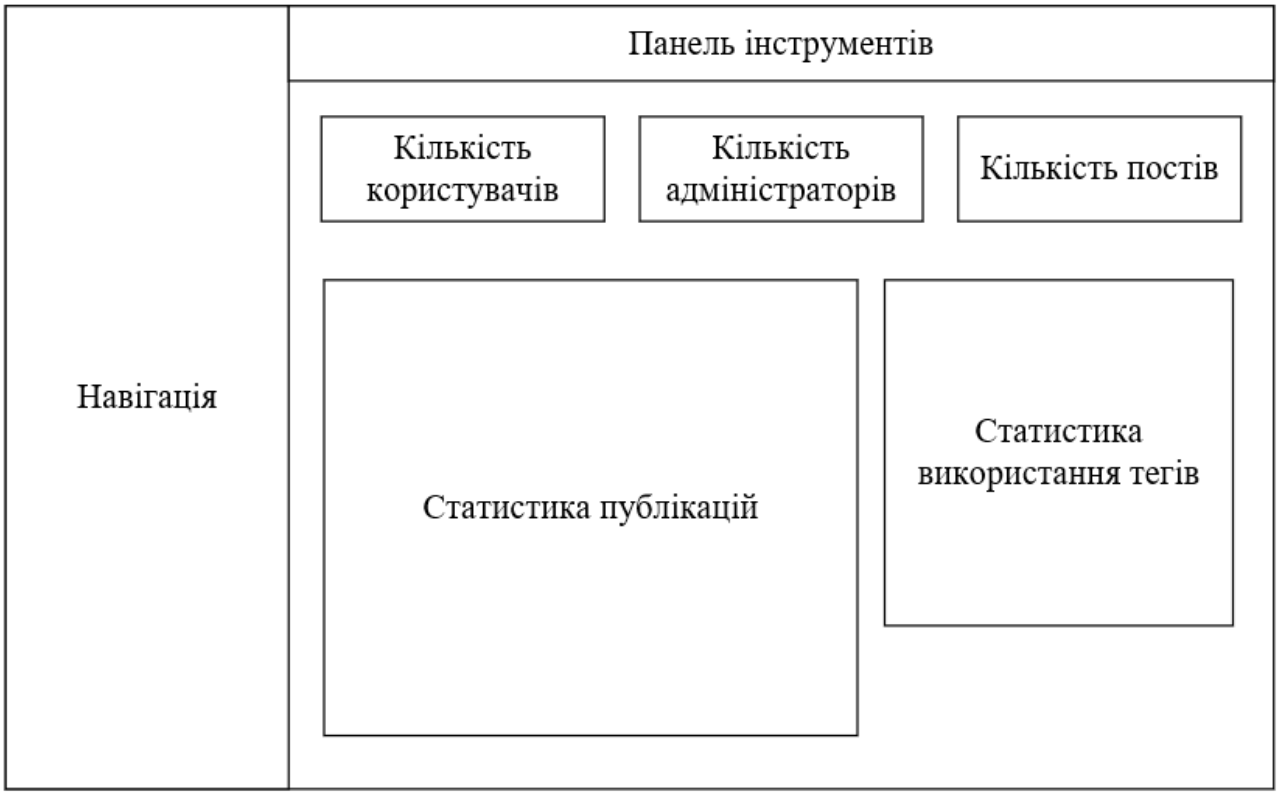

Рисунок 2.3 – Структура інтерфейсу сторінки аналітики

Структура сторінки аналітики веб-сайту розроблена таким чином, щоб забезпечити зручний інструментарій для моніторингу та аналізу діяльності користувачів. Вона включає панель, яка відображає ключові показники, такі як

загальна кількість користувачів, адміністраторів та кількість створених постів. Ця інформація допомагає відслідковувати зростання або зміни в користувацькій базі та активності на сайті.

Статистика публікацій надає детальний огляд контенту, який користувачі створюють та діляться, дозволяючи зрозуміти, які теми є найбільш популярними та залучають увагу аудиторії.

Додатково, сторінка аналітики включає аналіз використання тегів, що дає можливість виявити найбільш часто використовувані ключові слова та теги в постах. Це важливо для оптимізації пошукової системи та підвищення видимості контенту.

Використання програмного забезпечення Visio дало змогу створити структуру інтерфейсу мікроблогінгової соціальної мережі, що оптимізує взаємодію учасників з освітніми ініціативами [10]. Завдяки цьому, користувачі можуть ефективно використовувати функціонал платформи, швидко орієнтуватися завдяки підтримці навігації та легко доступним основним функціям та рекомендаціям. Це сприяє активній участі в освітньому процесі та залученню нових учасників у віртуальному просторі.

2.2 Розробка методу реєстрації та авторизації користувача

Пропонується метод реєстрації та аутентифікації користувачів, який підвищує рівень безпеки через застосування хешування паролів та використання механізмів сеансової автентифікації.

Одним із ключових аспектів захисту даних користувачів є хешування паролів. BCrypt є одним з найпопулярніших і найнадійніших алгоритмів хешування, що використовується для збереження паролів. Цей алгоритм забезпечує надійний захист завдяки кільком важливим властивостям.

По-перше, BCrypt використовує сіль (salt) – випадковий набір символів, що додається до паролю перед його хешуванням. Це робить кожен хеш унікальним,

навіть якщо два користувачі використовують однакові паролі. Використання солі захищає від атак заздалегідь обчислених хешів, таких як атаки «rainbow tables».

Також перевагою BCrypt є те, що він є адаптивним алгоритмом, що означає можливість регулювання складності хешування шляхом зміни кількості ітерацій. Це дозволяє зберігати високу стійкість до атак із підбором паролів у майбутньому, коли обчислювальні потужності зростатимуть.

BCrypt також генерує хеші фіксованої довжини, що спрощує зберігання та порівняння хешів. Навіть невеликі зміни у паролі призводять до значних змін у хеші, що унеможливлює його прогнозування.

Аутентифікація на основі сеансу є ще одним важливим елементом, що забезпечує безпеку користувачів у системі. Цей підхід використовує інфраструктуру сеансів, яка доступна у багатьох контейнерах, таких як Tomcat. Сеанси служать для зберігання даних з ідентифікатором сеансу, а контейнер webapp керує їх зберіганням і зв’язує їх з ідентифікатором сеансу. Принцип роботи аутентифікації на основі сеансу у вигляді діаграми варіантів використання зображено на рисунку 2.4.

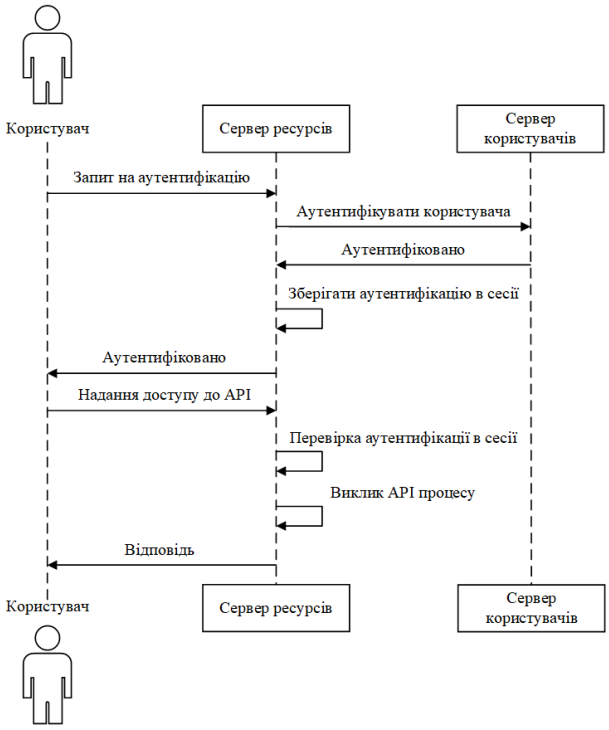

Рисунок 2.4 – Діаграма варіантів використання аутентифікації користувача на основі сеансу

Суть сеансової аутентифікації полягає у збереженні інформації про автентифікованого користувача в сеансі. Якщо користувач присутній у сеансі, це означає, що він автентифікований. Якщо його немає, користувач не є автентифікованим. У сеансі також можна зберігати додаткові дані, такі як набір дозволів користувача або інша інформація, корисна для роботи системи.

Ідентифікатори сеансів зазвичай передаються за допомогою файлів cookie або шляхом їх введення в URL-адресу. Використання файлів cookie є зручнішим, оскільки контейнер виконує більшу частину роботи за нас. Контейнер також обробляє весь життєвий цикл сеансу, включаючи його завершення після періоду бездіяльності користувача. Це означає, що користувач залишається автентифікованим лише під час активного використання системи. Однак використання сеансів вимагає доступу до сховища, щоб усі виклики від одного клієнта досягали одного сервера або забезпечували реплікацію сеансів між серверами. Також необхідний механізм обробки виходу з системи, який зазвичай реалізується шляхом завершення всього сеансу або видалення даних користувача.

Метод реєстрації та автентифікації користувачів є фундаментальним аспектом безпеки будь-якої інформаційної системи. Він вимагає ретельного підходу до зберігання та обробки особистих даних, зокрема паролів. Сучасні методи передбачають використання складних алгоритмів хешування та сеансових механізмів для забезпечення надійності та зручності користувачів.

Алгоритму починається з моменту, коли користувач заповнює форму реєстрації, вводячи свої персональні дані та пароль. Пароль піддається процесу хешування з використанням алгоритму BCrypt, який включає додавання солі та виконання кількох ітерацій для збільшення безпеки. Після цього хешований пароль разом з іншими даними користувача зберігається у базі даних. Успішна реєстрація ініціює створення сеансу, ідентифікатор якого зберігається у файлі cookie, що надсилається користувачеві. При кожному подальшому зверненні до системи відбувається перевірка ідентифікатора сеансу для підтвердження автентифікації. Сеанс завершується автоматично після певного періоду

неактивності або коли користувач виходить із системи, що вимагає повторної автентифікації для доступу до ресурсів.

В загальному метод забезпечує високий рівень безпеки та зручності для користувачів, поєднуючи надійне хешування паролів та ефективну сеансову аутентифікацію.

2.3 Розробка моделі роботи мікроблогінгової вебсистеми

Модель роботи мікроблогінгової вебсистеми передбачає можливість створення інформативних постів з функціями редагування, оцінювання та рекомендацій на основі алгоритмів уподобань. Це дозволяє системі підбирати найбільш релевантні пости для користувачів, що покращує їхній досвід та сприяє більшій залученості аудиторії.

Нижче зображено на рисунку 2.5 діаграму діяльності системи.

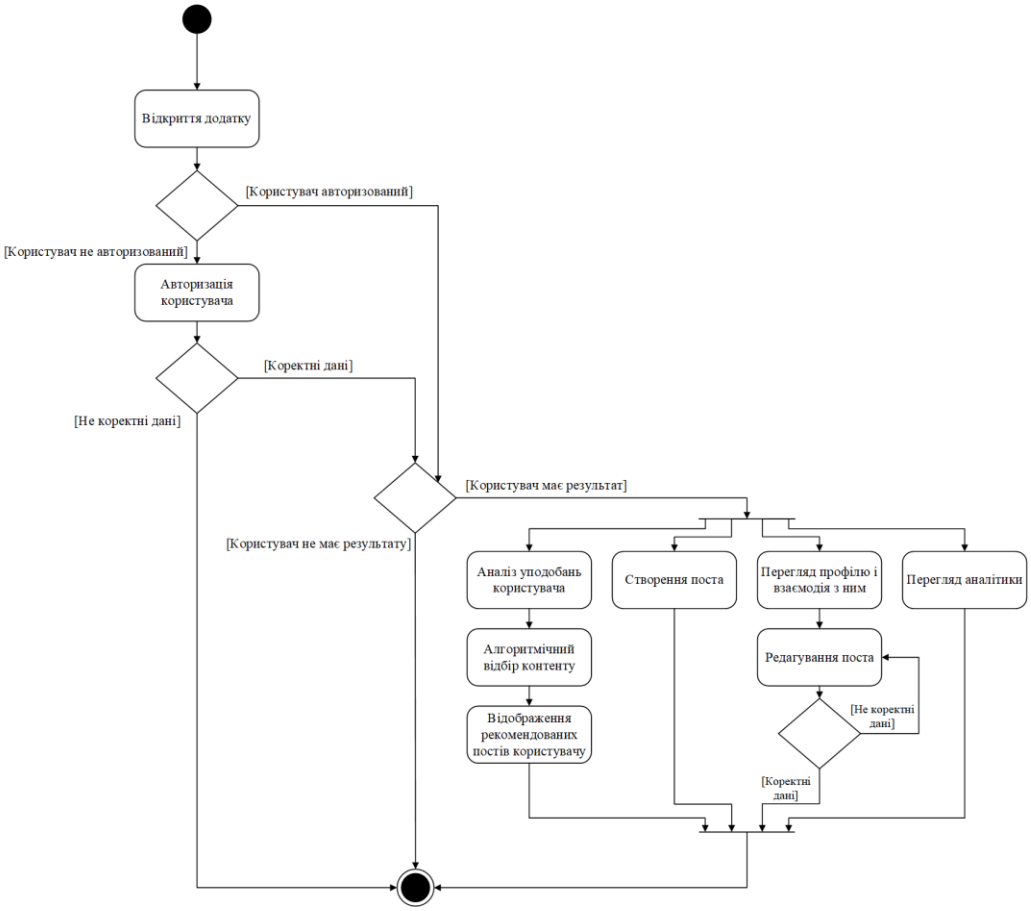

Рисунок 2.5 – Діаграма діяльності системи

Діаграма діяльності відображає всі основні процеси, включаючи реєстрацію, автентифікацію, створення постів, перегляд контенту та взаємодію з ним. Зокрема, діаграма показує, як користувачі проходять через різні етапи взаємодії з системою.

Модель функціонування вебсистеми починається з авторизації користувачів. Система перевіряє статус автентифікації і надає доступ до основних функцій тільки після успішної автентифікації. Користувачі можуть створювати нові пости, редагувати існуючі, оцінювати контент через лайки, а також отримувати персоналізовані рекомендації.

Алгоритми уподобань, вбудовані в систему, аналізують поведінку користувачів, їхню історію переглядів, взаємодії з постами та іншими користувачами. На основі цього аналізу система формує стрічку рекомендацій, що відповідає інтересам кожного користувача. Це забезпечує високу релевантність контенту та підвищує задоволеність користувачів від роботи з платформою.

Діаграма діяльності ілюструє основні процеси, які відбуваються в системі. Користувачі проходять етапи реєстрації та авторизації, після чого можуть переглядати, створювати та редагувати пости. Діаграма також відображає, як система обробляє запити, перевіряє права доступу і надає рекомендації. Зокрема, коли користувач створює новий пост, система аналізує його зміст і надає відповідні рекомендації іншим користувачам, базуючись на їх інтересах та вподобаннях.

Завдяки впровадженню таких алгоритмів і методів аналізу, мікроблогінгова система покращує якість взаємодії між користувачами. Це сприяє збільшенню активності користувачів та утриманню цільової аудиторії, підвищуючи рейтинг платформи. В кінцевому результаті, така система забезпечує більш персоналізований досвід користувачів, що є ключовим фактором у створенні успішної мікроблогінгової платформи для освітніх ініціатив.

2.4 Розробка структури бази даних

Для успішного функціонування будь-якого вебзастосунку критично важливо мати добре спроектовану структуру бази даних. Вона повинна забезпечувати ефективне зберігання, обробку та доступ до даних. Структура бази даних розроблена таким чином, щоб задовольнити всі вимоги додатку, уникнути дублювання даних та забезпечити легкість управління ними.

Однією з основних методологій, використаних при розробці структури бази даних, є нормалізація. Цей процес передбачає поділ даних на логічно окремі таблиці з метою усунення надлишкових даних і зменшення залежності між ними. Нормалізація дозволяє мінімізувати повторення даних, що значно спрощує процес оновлення інформації та забезпечує її цілісність. У структурі бази даних вебзастосунку дані користувачів, ролей та повідомлень розділені по окремих таблицях. Такий підхід дозволяє не лише підвищити ефективність системи, але й зменшити ризик виникнення аномалій при модифікації даних, таких як аномалії вставки, оновлення чи видалення.

Також одним з важливих аспектів оптимізації бази даних є індексація. Індексація дозволяє значно прискорити пошук і доступ до даних. Кожна таблиця має унікальні ідентифікатори, що використовуються як основні ключі. Це не тільки покращує швидкість запитів, але і забезпечує цілісність даних, гарантуючи унікальність кожного запису в таблиці.

Зв’язки між таблицями є ще одним важливим елементом структури бази даних. Вони дозволяють організувати дані у вигляді взаємопов’язаних сутностей, що полегшує виконання складних запитів. Наприклад, таблиці з даними про користувачів і повідомлення пов’язані таким чином, що дозволяє легко визначити, які повідомлення належать конкретному користувачу. Це здійснюється шляхом встановлення відповідних зв’язків між таблицями, що оптимізує процеси вибірки та обробки даних. Завдяки цьому, база даних може обробляти запити з високою швидкістю, що підвищує загальну продуктивність системи та забезпечує кращий досвід для користувачів.

Таблиця 2.1 ілюструє особливості зв’язків між елементами структури бази даних, що використовується в системі управління даними мікроблогінгової соціальної мережі для підтримки освітніх ініціатив.

Таблиця 2.1 – Характеристика зв’язків бази даних

Ім’я суті 1	Ім’я суті 2	Тип зв’язку	Клас належності
usr	message	1:N	Обов., необов.
usr	user_role	1:N	Обов., обов.
usr	user_subscriptions	N:N	Обов., необов.
usr	message_likes	1:N	Обов., необов.
message	message_likes	1:N	Обов., необов.

Перший зв’язком між таблицями «usr» і «message», є типу 1:N, який означає, що один користувач може створювати багато повідомлень, але кожне повідомлення має належати до одного користувача. Для таблиці «usr» цей зв’язок є обов’язковим, тоді як для таблиці «message» він необов’язковий. Такий зв’язок забезпечує, що всі повідомлення мають відповідного автора, що критично важливо для відстеження контенту на платформі.

Наступний зв’язок між таблицями «usr» і «user_role» також є типу 1:N. В такому випадку кожен користувач може мати кілька ролей, причому цей зв’язок є обов’язковим для обох сутностей. Кожен користувач має певну роль, яка визначає його права доступу та функції в системі. Це допомагає забезпечити відповідний рівень доступу до різних функцій та даних на платформі.

Зв’язок між таблицями «usr» і «user_subscriptions» є типу N:N, що означає, що кожен користувач може підписуватись на багато інших користувачів і навпаки. Для таблиці «usr» цей зв’язок є обов’язковим, тоді як для таблиці user_subscriptions він необов’язковий. Цей зв’язок дозволяє користувачам підписуватись на оновлення від інших користувачів, сприяючи взаємодії та обміну інформацією,що є важливою частиною функціоналу соціальної мережі, спрямованого на підвищення залученості користувачів.

Зв’язок між таблицями «usr» і «message_likes» є типу 1:N. Це означає, що кожен користувач може ставити лайки на багато повідомлень. Для таблиці «usr» цей зв’язок є обов’язковим, тоді як для таблиці «message_likes» він необов’язковий. Цей зв’язок дозволяє користувачам взаємодіяти з контентом, висловлюючи свою підтримку або зацікавленість через лайки, що допомагає визначати популярність контенту та стимулює активність на платформі.

Зв’язок між таблицями «message» і «message_likes» також є типу 1:N, де кожне повідомлення може мати багато лайків. Цей зв’язок є обов’язковим для першої сутності. Кожне повідомлення може отримувати лайки від різних користувачів, що сприяє оцінці популярності та важливості кожного повідомлення. Це допомагає користувачам швидко знаходити цікаві та корисні пости.

Також одним з важливих аспектів оптимізації бази даних є індексація. Індексація дозволяє значно прискорити пошук і доступ до даних. Кожна таблиця має унікальні ідентифікатори, що використовуються як основні ключі. Це не тільки покращує швидкість запитів, але і забезпечує цілісність даних, гарантуючи унікальність кожного запису в таблиці.

Зв’язки між таблицями є ще одним важливим елементом структури бази даних. Вони дозволяють організувати дані у вигляді взаємопов’язаних сутностей, що полегшує виконання складних запитів. Наприклад, такі таблиці, як дані про користувачів і повідомлень пов’язані між собою таким чином, що дозволяє легко визначити, які повідомлення належать конкретному користувачу. Це здійснюється шляхом встановлення відповідних зв’язків між таблицями, що оптимізує процеси вибірки та обробки даних.

Ще один метод, який був використаний для оптимізації бази даних, – це каскадне видалення. Цей метод дозволяє автоматизувати процес видалення пов’язаних даних. Наприклад, при видаленні користувача автоматично видаляються всі його повідомлення. Це спрощує управління залежностями в базі даних і забезпечує її цілісність.

Для подальшої оптимізації структури бази даних планується використовувати міграційні скрипти. Це дозволить поступово вносити зміни до структури бази даних, контролюючи процес розвитку та забезпечуючи її консистентність. Використання міграційних скриптів також дозволить легко адаптувати базу даних до нових вимог, що виникають в процесі розвитку вебзастосунку.

Розроблена структуру бази даних зображена на рисунку 2.6.

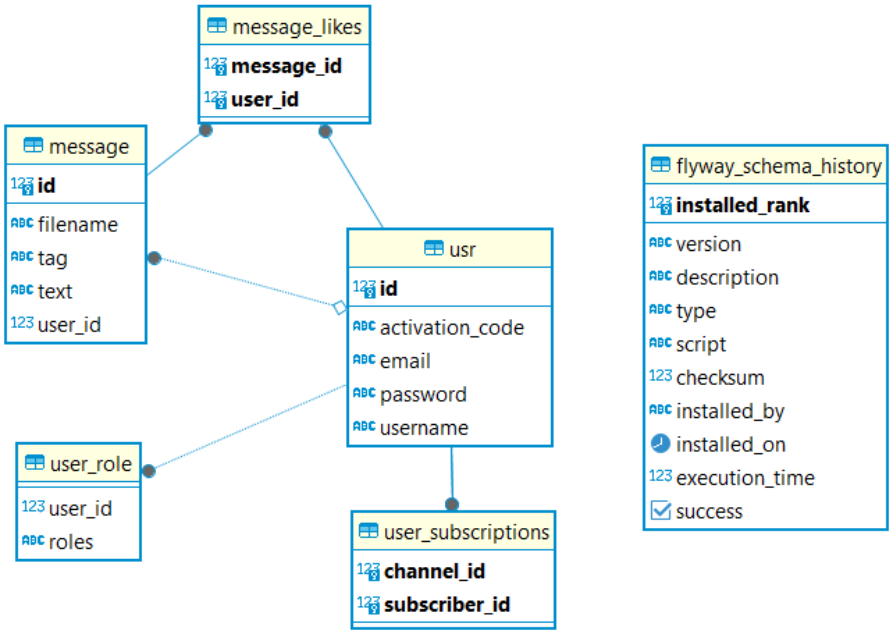

Рисунок 2.6 – Структура бази даних

На рисунку представлена структура бази даних, яка включає декілька таблиць, що взаємодіють між собою. Кожна таблиця має свої поля, які містять конкретні дані.

Таблиця «usr» є центральною для зберігання основної інформації про користувачів системи мікроблогінгу, включаючи їх ідентифікаційні дані та контактну інформацію. Вона складається з таких полів:

- «id» – унікальний ідентифікатор користувача;
- «activation_code» – код активації для верифікації аккаунту;
- «email» – електронна адреса користувача;
- «password» – зашифрований пароль користувача;

- «username» – ім’я користувача, яке використовується для входу в систему.

Таблиця «user_role» визначає ролі користувачів у системі та їх пов’язані права доступу, що дозволяє налаштовувати рівні доступу до різних функцій мікроблогінгової платформи. Вона включає такі поля:

- «user_id» – ідентифікатор користувача, який пов’язаний з роллю;
- «roles» – роль користувача в системі, яка визначає його доступ до функціоналу.

Таблиця «message» призначена для зберігання повідомлень, які користувачі публікують у мікроблогінговій мережі, включаючи текстові дані та посилання на медіа-вкладення. Вона складається із таких полів:

- «id» – унікальний ідентифікатор повідомлення;
- «filename» – ім’я файлу, якщо повідомлення містить медіа-вкладення;
- «text» – текст повідомлення, яке користувач публікує;
- «user_id» – ідентифікатор користувача, який створив повідомлення.

Таблиця «flyway_schema_history» використовується для відстеження історії міграцій бази даних, що дозволяє контролювати зміни схеми та забезпечувати її консистентність. Вона включає такі поля:

- «installed_rank» – порядковий номер міграції;
- «version» – версія схеми бази даних;
- «description» – опис міграції;
- «type» – тип міграції (наприклад, SQL);
- «script» – назва скрипта міграції;
- «checksum» – контрольна сума для верифікації;
- «installed_by» – користувач, який виконав міграцію;
- «installed_on» – дата та час встановлення міграції;
- «success_time» – час успішного виконання міграції;
- «execution_time» – загальний час виконання міграції.

Таблиця «user_subscription» використовується для відстеження підписок користувачів на інший контент чи користувачів у системі. Вона містить наступні поля:

- «channel_id» – унікальний ідентифікатор підписки;
- «subscriber_id» – ідентифікатор користувача, який здійснює підписку.

Таблиця «message_likes» призначена для зберігання інформації про лайки, які користувачі залишають на постах у системі. Вона містить такі поля:

- «message_id» – ідентифікатор повідомлення, до якого належить лайк, що також являється зовнішнім ключем, який пов’язаний з таблицею «messages», де зберігаються всі повідомлення системи;
- «user_id» – ідентифікатор користувача, який поставив лайк, що теж є зовнішнім ключем, що пов’язаний з таблицею «users», яка зберігає інформацію про всіх користувачів платформи.

Така структура бази даних забезпечує необхідну гнучкість для розширення функціоналу соціальної мережі та підтримки різноманітних освітніх ініціатив. Використання реляційної моделі дозволяє легко зв’язувати дані між таблицями, що є критично важливим для забезпечення цілісності даних у мікроблогінгових платформах. Таблиці пов’язані між собою через унікальні ідентифікатори, що забезпечує цілісність і узгодженість даних. Структура бази даних також включає індекси, які прискорюють виконання запитів до бази даних.

У результаті, розробка структури бази даних з урахуванням нормалізації, індексації, зв’язків між таблицями та каскадного видалення дозволяє забезпечити ефективне управління даними, їхню цілісність та швидкий доступ до них. Такі підходи до проектування бази даних є важливими для забезпечення стабільної роботи вебзастосунку і його подальшого розвитку.

2.5 Висновки

У розділі було виконано систематичний аналіз основних аспектів створення програмного забезпечення. В першу чергу, була розроблена структура

інтерфейсу програмного засобу, яка відповідає сучасним стандартам та забезпечує зручність взаємодії користувача з системою.

Було розглянуто методи забезпечення безпеки, такі як хешування паролів та сеансова автентифікація, що сприяють захисту конфіденційної інформації користувачів.

Також описано модель роботи системи, яка враховує потреби користувачів та забезпечує їм можливість створювати, редагувати та оцінювати пости, а також отримувати рекомендації на основі їх уподобань. Важливим аспектом розділу є розробка структури бази даних, яка забезпечує ефективне зберігання та обробку інформації.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ВЕБСИСТЕМИ

3.1 Архітектурний вибір та інструментарій для розробки застосунку

При розробці програмного забезпечення для мікроблогінгової соціальної мережі, спрямованої на підтримку освітніх ініціатив, ключовим є вибір технологій, які забезпечать надійність, швидкість та гнучкість системи.

Використання Java у поєднанні з фреймворком Spring Boot дозволяє створити масштабовану та безпечну платформу, яка може ефективно обробляти великі обсяги даних та запитів користувачів. Програмування на Java при розробці серверної частини вебзастосунків надає ряд значних переваг [11]:

1. Принцип «write once, run anywhere» гарантує, що написаний код може виконуватися на будь-якій платформі з підтримкою Java Virtual Machine, що забезпечує гнучкість у виборі середовища виконання.

2. Компіляція в байт-код для виконання на JVM сприяє високій продуктивності та ефективному використанню апаратних ресурсів, що критично для серверних додатків з великим обсягом запитів і даних.

3. Розмаїття доступних бібліотек та фреймворків, таких як Spring Boot і Hibernate, полегшує розробку та інтеграцію нових можливостей, підвищуючи ефективність роботи розробників.

4. Високий рівень безпеки, який забезпечується завдяки вбудованим механізмам управління пам'яттю та контролю доступу, робить Java надійним вибором для вебзастосунків, де захист даних є пріоритетом.

5. Можливості багатопоточності та підтримка розподілених обчислень дозволяють створювати масштабовані системи, здатні ефективно обробляти великі навантаження, та забезпечують високий рівень паралельної обробки запитів, що є важливим для сучасних вебзастосунків.

Python є мовою з великою гнучкістю, яка відома своєю читабельністю та лаконічністю синтаксису [12]. Це робить її популярним вибором для швидкої розробки та прототипування, а також для використання в наукових обчисленнях

та аналітиці даних. Python також має сильну підтримку у спільноті, що забезпечує широкий спектр бібліотек та фреймворків для різноманітних завдань.

PHP, з іншого боку, є мовою, яка була спеціально створена для веб-розробки. Вона пропонує простоту вбудовування у HTML та широкий набір інструментів для створення динамічних веб-сайтів [13]. PHP має велику базу користувачів та хостинг-сервісів, що підтримують її «з коробки», що робить її доступною для широкого кола розробників.

Результати порівняння мов програмування для розробки серверної частини приведено в таблиці 3.1.

Таблиця 3.1 – Порівняльня характеристика мов програмування

Характеристика	Java	Python	PHP
Платформна незалежність	1	1	0
Продуктивність	1	0	0
Наявність бібліотек та фреймворків для розробки серверних додатків	1	1	1
Вбудовані механізми безпеки	1	0	0
Підтримка багатопоточності	1	1	0
Сумарний коефіцієнт	5	3	1

Згідно з цією таблицею, Java має найвищий сумарний коефіцієнт, що підтверджує її переваги для розробки серверної частини вебзастосунків. Python також має хороші показники, але вона поступається Java у продуктивності, яка компілюється в байт-код або машинальний код та вбудованих механізмах безпеки. PHP має найменший загальний бал, що відображає її обмеження у платформній незалежності, продуктивності та багатопоточності.

В загальному Java є оптимальним вибором для розробки серверної частини вебзастосунку через високу продуктивність, багату екосистему бібліотек і фреймворків, безпеку та масштабованість. Платформна незалежність і підтримка

багатопоточності дозволяють ефективно обробляти великі обсяги запитів. Розвинена екосистема Java дозволяє швидко інтегрувати нові функції, зосереджуючись на бізнес-завданнях.

Далі слід розглянути фреймворк для розробки вебзастосунку, який підходить для мови програмування Java. Першим краще взяти до уваги фреймворк, який вирізняється своєю автоматизацією та ефективністю, тобто Spring Boot [14]. Його особливість полягає у вбудованих інструментах, які забезпечують швидке налаштування та розгортання великих додатків та мікросервісів. Його механізми не лише полегшують інтеграцію з іншими частинами Spring, але й також надають готові рішення для реалізації складних завдань.

Інший фреймворк Micronaut пропонує альтернативний спосіб, зосереджуючись на мінімізації споживання ресурсів та швидкості запуску [15]. Це робить його ідеальним для розробки легковагових мікросервісів, особливо в випадках, коли ресурси обмежені, наприклад, у хмарних обчисленнях.

Quarkus також слід взяти до уваги. Він розроблений для використання з GraalVM та HotSpot, що дозволяє досягти низького споживання пам’яті та швидкого часу відгуку, що є критично важливим для сучасних мікросервісних архітектур та серверлес додатків [16].

Для порівняння фреймворків сформовано порівняльну таблицю 3.2.

Таблиця 3.2 – Порівняльна характеристика фреймворків

Характеристика	Spring Boot	Micronaut	Quarkus
Споживання пам’яті	1	0	0
Гнучке налаштування	1	1	0
Автоматизація конфігурації та коду	1	0	0
Підтримка мікросервісів	1	1	1
Сумарний коефіцієнт	4	2	2

За результатами порівняння, Spring Boot краще завдяки своїм перевагам у

категоріях, таких як споживання пам’яті та автоматизації конфігурації та коду, що значно спрощує процес розробки за рахунок використання анотацій. Він також забезпечує відмінну підтримку мікросервісів, що є ключом для сучасних вебзастосунків. Таким чином, Spring Boot явно переважає свої аналоги і є оптимальним рішенням для розробки.

Не менш важливим є вибір шаблонізатору для створення гнучкого та ефективного вебзастосунку. Шаблонізатор впливає на швидкість розробки та легкість підтримки програмного коду, також на можливості оптимізації продуктивності. Вибір буде здійснюватися між Thymeleaf та FreeMarker. Thymeleaf може бути кращим вибором, якщо вже використовується Spring, завдяки його глибокій інтеграції з цим фреймворком. Він дозволяє використовувати природне шаблонування та підтримка HTML5/XML, як основу для шаблонів, що робить код його ідеальним для розробки вебзастосунку [17].

З іншої сторони FreeMarker, може бути кращим якщо потрібна висока продуктивність та швидкість обробки шаблонів, особливо, якщо використовуються великі обсяги даних або складні шаблони [18].

Нижче в таблиці 3.3 наведено порівняння розглянутих шаблонізаторів.

Таблиця 3.3 – Порівняльна характеристика фреймворків

Характеристика	Thymeleaf	FreeMarker
Сумісність з фреймворком Spring для спрощення розробки	1	0
Можливість використання статичного HTML для динамічного контенту	1	0
Підтримка сучасних стандартів веб-розмітки	1	0
Підтримка умовних виразів, циклів, змінних	1	1
Легкість освоєння та використання синтаксису	1	1
Сумарний коефіцієнт	5	2

В результаті обидва фреймворки мають сильні сторони в легкості освоєння та підтримці умовних виразів, циклів та різних змінних, проте Thymeleaf переважає в таких аспектах, як інтеграція з Spring, природнє шаблонування, та підтримка HTML5/XML. Тому краще обрати Thymeleaf, тому що він надає якісніші можливості для розробки, що робить його привабливим вибором.

Для розробки мікроблогінгової соціальної мережі для освітніх ініціатив, важливо вибрати СУБД, яка не тільки забезпечує високу продуктивність та надійність, але й підтримує складні запити та аналітичні функції, які можуть бути необхідними для обробки та аналізу великих обсягів освітніх даних.

Першою для розгляду є система управління базами даних PostgreSQL, яка відома своєю потужною функціональністю та високою надійністю, забезпечуючи ефективну роботу зі складними запитами та транзакціями [19]. MySQL, інша популярна система управління базами даних, широко застосовується для створення вебзастосунків завдяки швидкому доступу до великих обсягів даних, оптимізаціям для операцій читання і запису, а також підтримці реплікації [20].

Нижче таблиця 3.4 представляє порівняльний аналіз PostgreSQL та MySQL, враховуючи їх застосування у Java-розробці вебзастосунків.

Таблиця 3.4 – Порівняльна характеристика СУБД

Характеристика	PostgreSQL	MySQL
Інтеграція з Java та підтримка JDBC	1	1
Підтримка складних запитів, CTE, віконних функцій	1	0
Здатність адаптуватися до зростання обсягів даних	1	0
Ліцензування та можливість розширення	1	0
Робота з JSON-даними та запитами	1	1
Сумарний коефіцієнт	5	2

У контексті розробки вебзастосунку для мікроблогінгу з освітньою спрямованістю, PostgreSQL має значні переваги, які відповідають специфічним вимогам. Його здатність до обробки складних запитів та гнучкості у масштабуванні робить його оптимальним вибором для платформи, що дозволить розширення та адаптацію до змін в обсягах даних. З огляду на потребу в інтеграції з Java та використанні розширених аналітичних інструментів для освітнього контенту, PostgreSQL є кращим вибором.

Вибір цих технологій та інструментів є вирішальним для створення ефективного, безпечного та легко масштабованого програмного забезпечення, яке може підтримувати активну взаємодію користувачів та обмін освітніми ресурсами. Поєднання Java, Spring Boot, Thymeleaf, PostgreSQL та Spring Security надає міцну основу для розробки мікроблогінгової соціальної мережі, яка може ефективно виконувати свої освітні та комунікативні функції.

3.2 Розробка модуля реєстрації та автентифікації користувача

Модуль реєстрації та автентифікації користувача є важливою складовою будь-якого вебзастосунка, зокрема мікроблогінгової соціальної мережі. Цей модуль відповідає за створення нових облікових записів користувачів та перевірку їх доступу до системи.

Для реєстрації нового користувача в системі використовується HTTP-запит методом POST до ендпоінту /registration. При надходженні запиту виконується обробка у контролері RegistrationController.

Контролер RegistrationController відповідає за обробку запитів, пов’язаних із реєстрацією нових користувачів. Він виконує такі завдання:

Метод registration(), який відповідає на GET-запити до ендпоінту /registration. В цьому методі відбувається повернення сторінки реєстрації.

Метод addUser(User user) у контролері RegistrationController приймає POST-запити до ендпоінту /registration. Перш ніж реєструвати нового користувача, метод перевіряє чи існує користувач з такими ж обліковими даними

у базі даних. Якщо користувач вже існує, повертається відповідне повідомлення про те, що користувач вже зареєстрований. Алгоритм методу addUser, що здійснює реєстрацію користувача подано на рисунку 3.1 та на рисунку 3.2.

Далі, метод перевіряє, чи вказаний email не використовується іншим користувачем. Якщо такий email вже прив'язаний до іншого облікового запису, також повертається відповідне повідомлення про те, що email вже зайнятий.

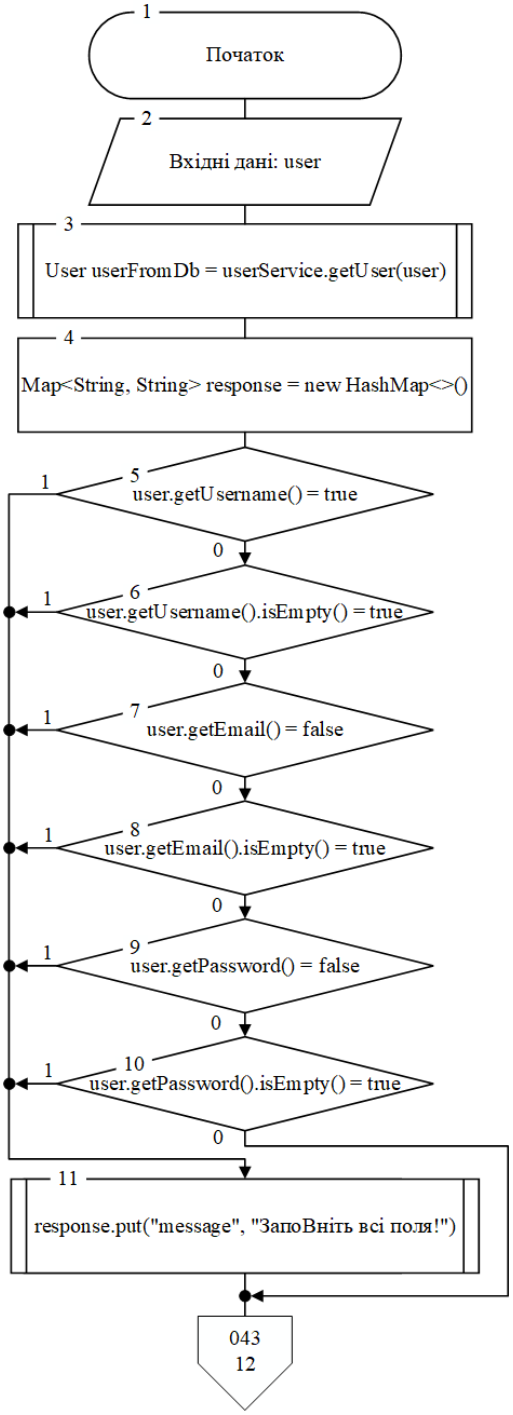

Рисунок 3.1 – Алгоритм методу реєстрація користувача addUser (частина 1)

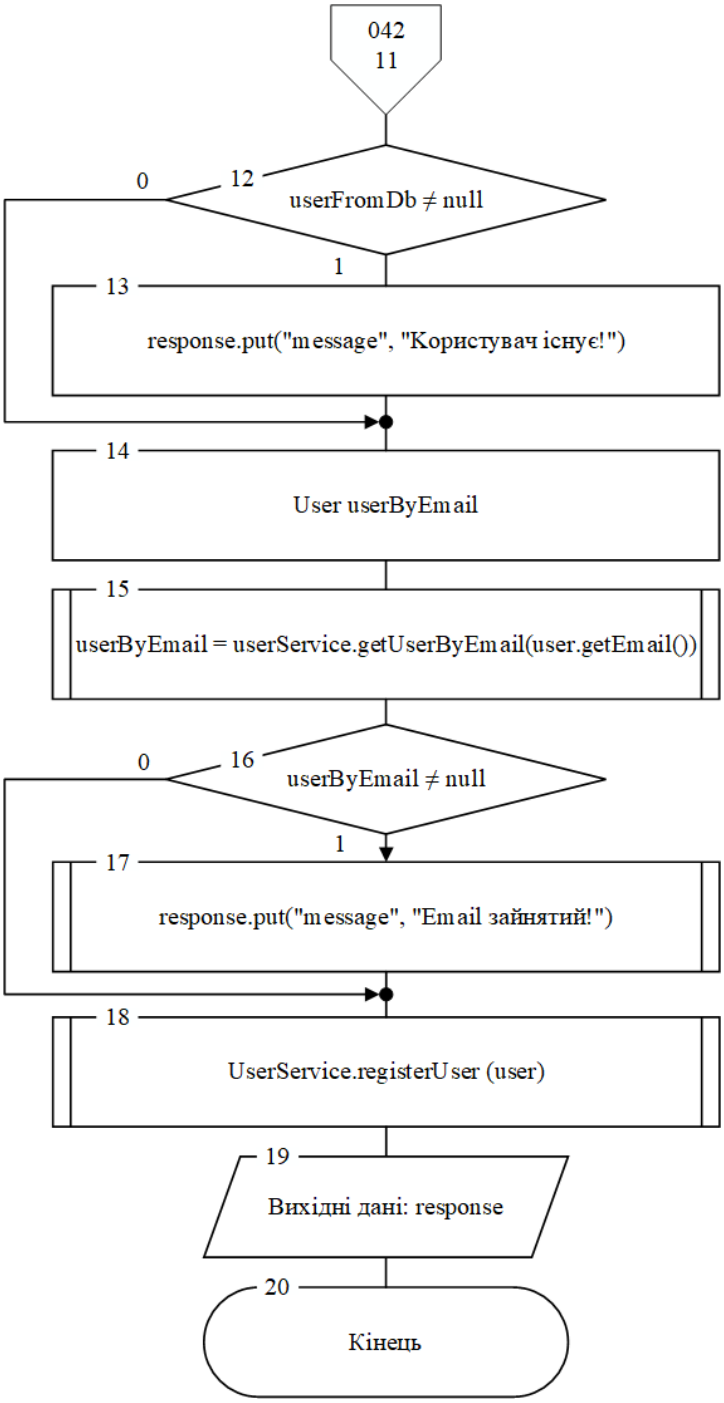

Рисунок 3.2 – Алгоритм методу реєстрація користувача addUser (частина 2)

У випадку, якщо ні користувач, ні email не існують у базі даних, метод реєструє нового користувача, використовуючи сервіс userService. Після успішної реєстрації, встановлюється редірект на сторінку автентифікації /login.

Клас MailConfig відповідає за налаштування відправлення електронних листів у системі. Для використання цього класу необхідно забезпечити належні налаштування поштового сервера та облікового запису для відправлення листів.

У класі MailConfig виконується налаштування параметрів поштового сервера, таких як хост, порт, протокол та інші. Ці параметри визначаються за допомогою значень, які зберігаються у файлі налаштувань application.properties. Конфігурація включає такі значення, як хост (наприклад, smtp.gmail.com), ім'я користувача, пароль, порт (наприклад, 465), протокол (наприклад, smtps) та параметр налагодження (true). Значення для імені користувача та пароля беруться з файлу налаштувань, де вони зберігаються як змінні «spring.mail.username» та «GMAIL_APP_PASSWORD» відповідно. Ці змінні використовуються для підключення до поштового сервера та надсилання електронних листів від імені користувача.

Для створення та налаштування об’єкта, який буде відповідати за відправлення електронних листів, використовується спеціальний метод. Цей метод позначений анотацією «@Bean», що означає, що він повертає екземпляр об'єкта, який буде керованим контейнером Spring. У даному випадку метод створює об’єкт JavaMailSender, який відповідає за відправлення електронних листів. Параметри, що налаштовуються в цьому методі, включають хост, порт, ім'я користувача та пароль, які вже були задані раніше у класі MailConfig.

Крім основних параметрів підключення, в об’єкті JavaMailSender налаштовуються додаткові властивості через об'єкт типу Properties. Наприклад, задається протокол передачі (наприклад, smtps) та режим налагодження (true), що дозволяє виводити детальну інформацію про процес відправлення листів у логах. Це корисно для діагностики та вирішення можливих проблем з відправленням електронних листів.

Отже, контейнер Spring автоматично керує життєвим циклом створеного об'єкта JavaMailSender, забезпечуючи його правильне створення, ініціалізацію та використання в інших компонентах Spring. Це дозволяє легко інтегрувати функціонал відправки електронних листів у застосунок без необхідності вручну керувати всіма аспектами підключення та налаштування поштового сервера.

У файлі application.properties визначаються основні параметри поштового сервера, такі як хост, адреса електронної пошти, пароль, порт, протокол тощо.

Нижче на рисунку 3.3 подані приклади параметрів для підключення до поштового сервера Gmail.

```
spring.datasource.url=jdbc:postgresql://localhost/learn_nook
spring.datasource.username=postgres
spring.datasource.password=${DB_POSTGRES_PASSWORD}
spring.jpa.generate-ddl=false
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=validate
spring.flyway.baseline-on-migrate = true
```

Рисунок 3.3 – Параметри поштового сервера Gmail

Значення ${GMAIL_APP_PASSWORD} визначається окремо та не зберігається у відкритому вигляді, щоб забезпечити безпеку облікових даних.

Автентифікація користувача здійснюється за допомогою HTTP-запиту методом POST до ендпоінту /login. При цьому виконується перевірка активності облікового запису користувача. Для цього використовується фільтр «AccountActivationLoginFilter», який перехоплює запити до /login та проводить перевірку статусу облікового запису. Все це знаходиться в методі doFilter().

У випадку, якщо обліковий запис користувача не активний, відбувається перенаправлення на сторінку автентифікації з повідомленням про не активований обліковий запис.

Глобальне перехоплення винятків дозволяє централізовано обробляти виняткові ситуації, що виникають у всьому вебзастосунку, і відповідно реагувати на них. У вебзастосунках на основі Spring це реалізовано за допомогою анотації @ControllerAdvice і класу «GlobalExceptionHandler».

Клас «GlobalExceptionHandler» клас використовується для обробки виняткових ситуацій у всьому вебзастосунку. У ньому визначені методи, які обробляють конкретні типи винятків.

Метод «handleRedirectWithModel Exception» в класі «GlobalException Handler» приймає виняток «Redirect-WithModelException», модель «Model» для

передачі атрибутів у представлення та об’єкт «HttpServletRequest». У цьому методі відбувається обробка винятка і визначається, як саме на нього реагувати.

Клас «RedirectWithModelException» клас є спеціалізованим типом винятку, який використовується для перенаправлення користувача на іншу сторінку з передачею додаткових даних у моделі. Клас містить два поля для зберігання назви та значення атрибута, який буде переданий у модель. Також містить конструктор для ініціалізації цих полів та методи для отримання їх значень.

При виникненні винятка «RedirectWithModelException», метод «handleRedirectWithModelException» в класі «GlobalExceptionHandler» відповідає за обробку цього винятка. Він додає в модель атрибут з переданими значеннями та відправляє користувача на іншу сторінку, яка визначена у методі, наприклад, сторінку автентифікації /login.

Таким чином, модуль реєстрації та автентифікації користувача використовує низку класів та методів для забезпечення безпеки та коректності процесу входу та реєстрації в мікроблогінговій соціальній мережі.

3.3 Розробка модуля публікації та перегляду постів

Головною задачею модуля, як і всього вебзастосунку, є забезпечення зручного та ефективного механізму для публікації та перегляду постів користувачами. Цей модуль включає в себе ряд ключових компонентів, які разом формують функціональність, необхідну для взаємодії користувачів з мікроблогінговою платформою.

Контролер «MainController» відіграє центральну роль у модулі, керуючи запитами на створення нових постів та їх відображення. Метод обробляє HTTP GET запити до кореневого шляху («/») і повертає назву вебсторінки «learnNook», яка є головною точкою взаємодії користувача з додатком.

Коли користувач бажає створити новий пост, він заповнює форму на вебсторінці, вводячи текст посту та вибираючи тег. Якщо користувач хоче додати зображення до свого посту, він може завантажити файл через цю ж

форму. Після заповнення форми користувач натискає кнопку «Додати», щоб надіслати інформацію. На сервері ця інформація обробляється спеціальним методом, який називається add. В загальному алгоритм роботи методу add представлено на рисунках 3.4 та 3.5.

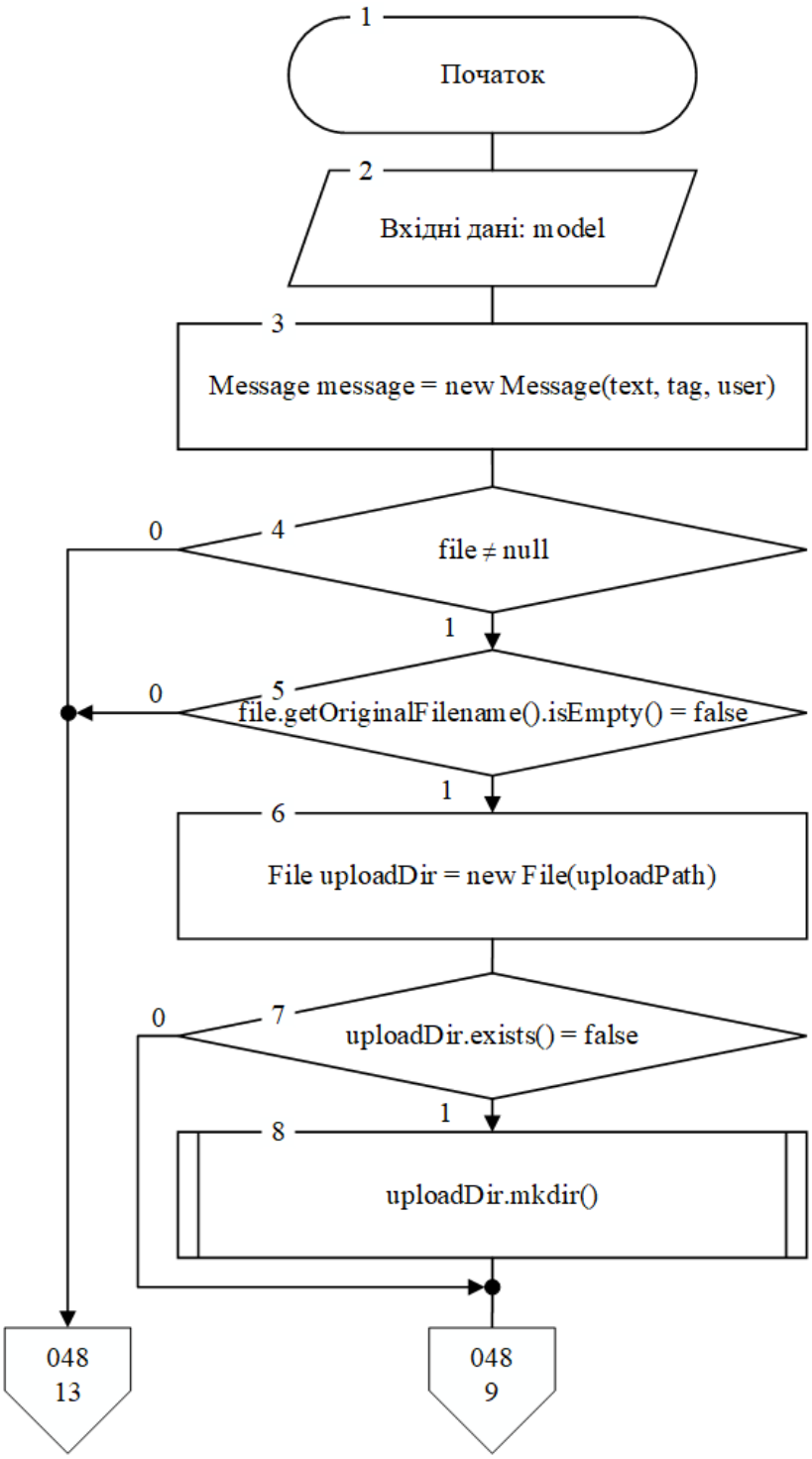

Рисунок 3.4 – Алгоритм роботи методу додавання нового посту та перегляд інших (частина 1)

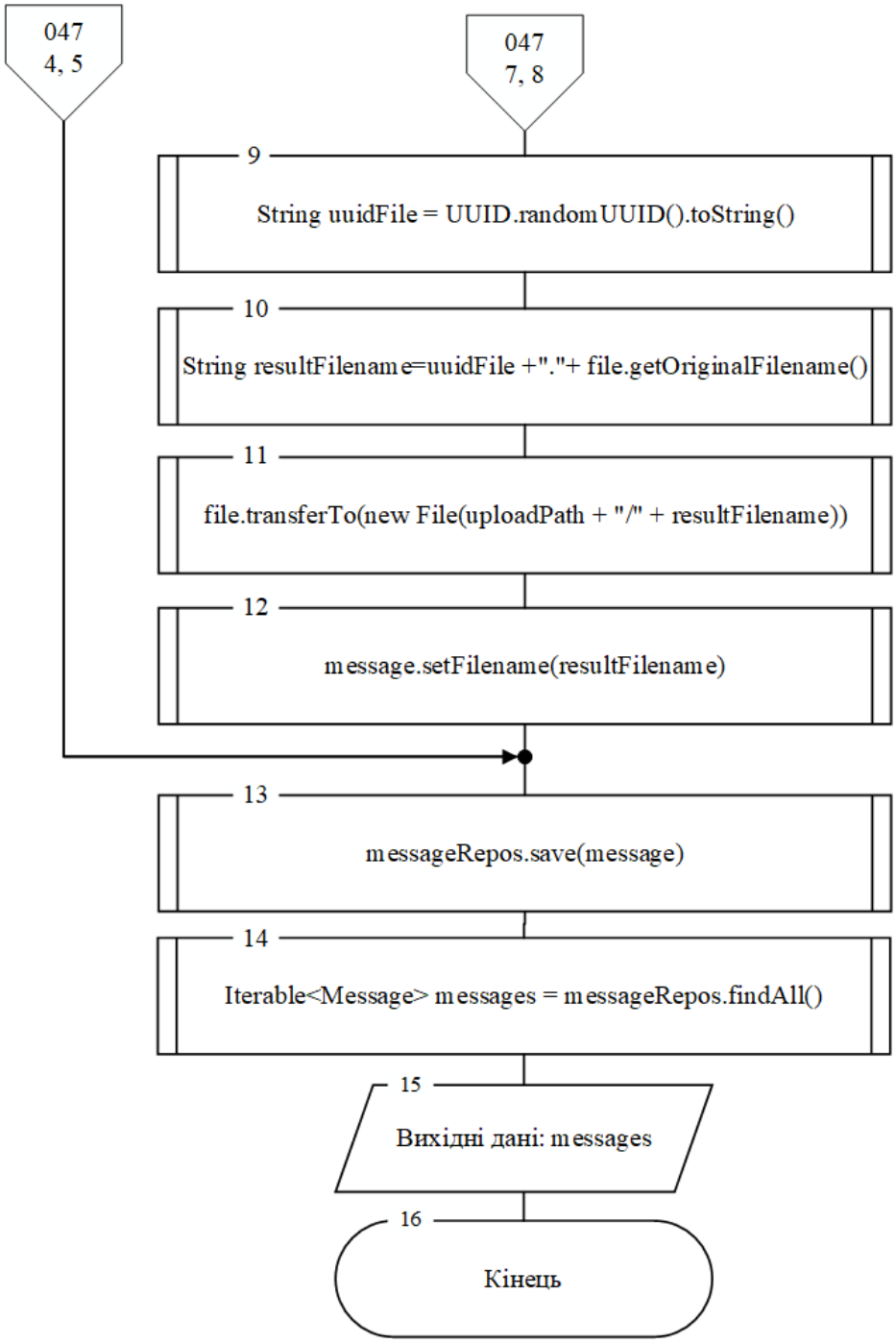

Рисунок 3.5 – Алгоритм роботи методу додавання нового посту та перегляд інших (частина 2)

Метод «add» приймає дані, введені користувачем, і створює новий об’єкт, який представляє пост. Об’єкт містить текст посту, тег, а також інформацію про користувача, який створив пост. Якщо користувач завантажив зображення, метод перевіряє, чи файл дійсно існує та чи не є він порожнім. Якщо умови виконуються, метод створює унікальне ім’я для файлу, щоб уникнути конфліктів

із іншими файлами, і зберігає файл у визначеній папці на сервері. Після цього ім’я файлу зберігається разом з іншою інформацією про пост. Нарешті, новий пост зберігається в базі даних, що дозволяє іншим користувачам переглядати його на вебсайті. Всі пости, які зберігаються в базі даних, можуть бути отримані та відображені на головній сторінці соціальної мережі.

Нижче на рисунках 3.6 і 3.7 представлено алгоритм роботи методу формування відображення постів.

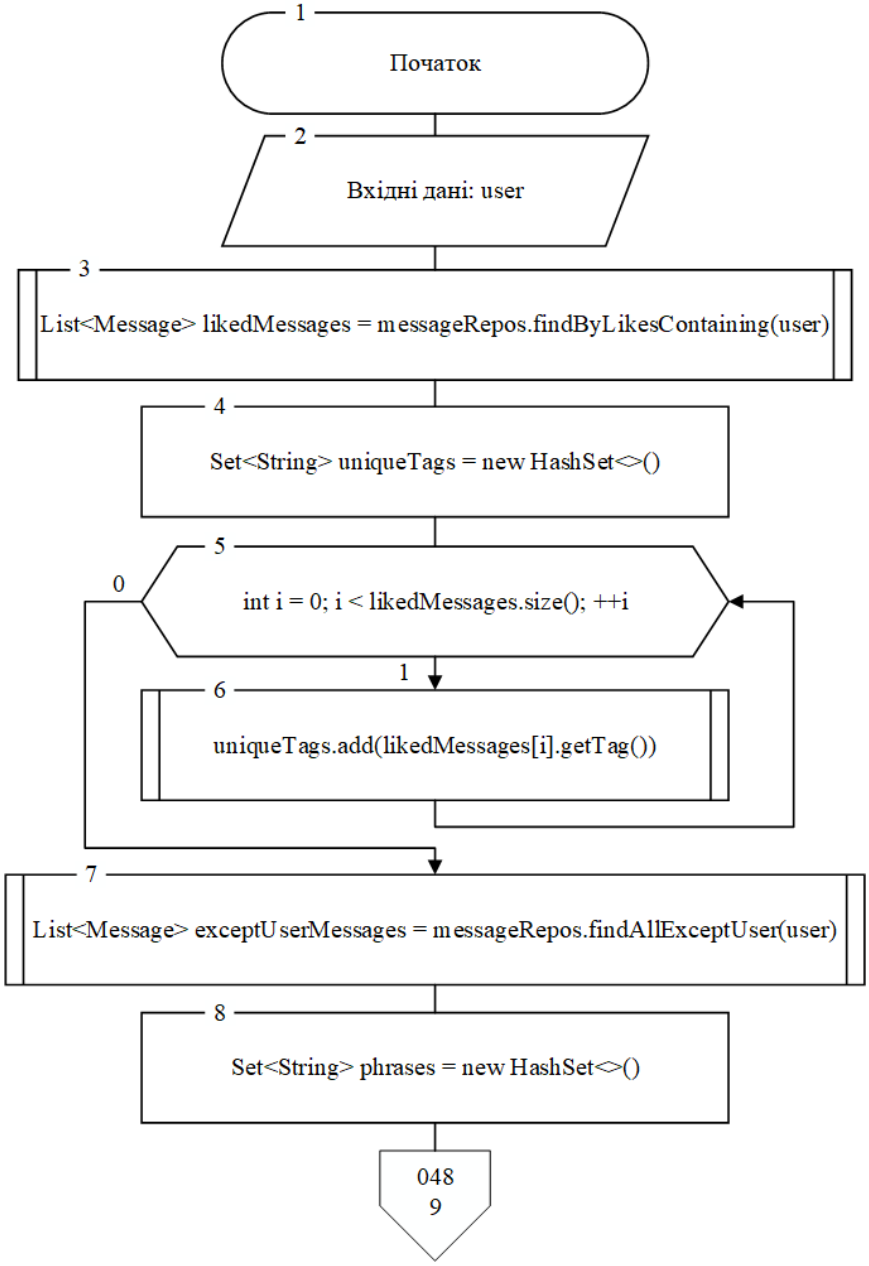

Рисунок 3.6 – Алгоритму роботи методу формування відображення постів (частина 1)

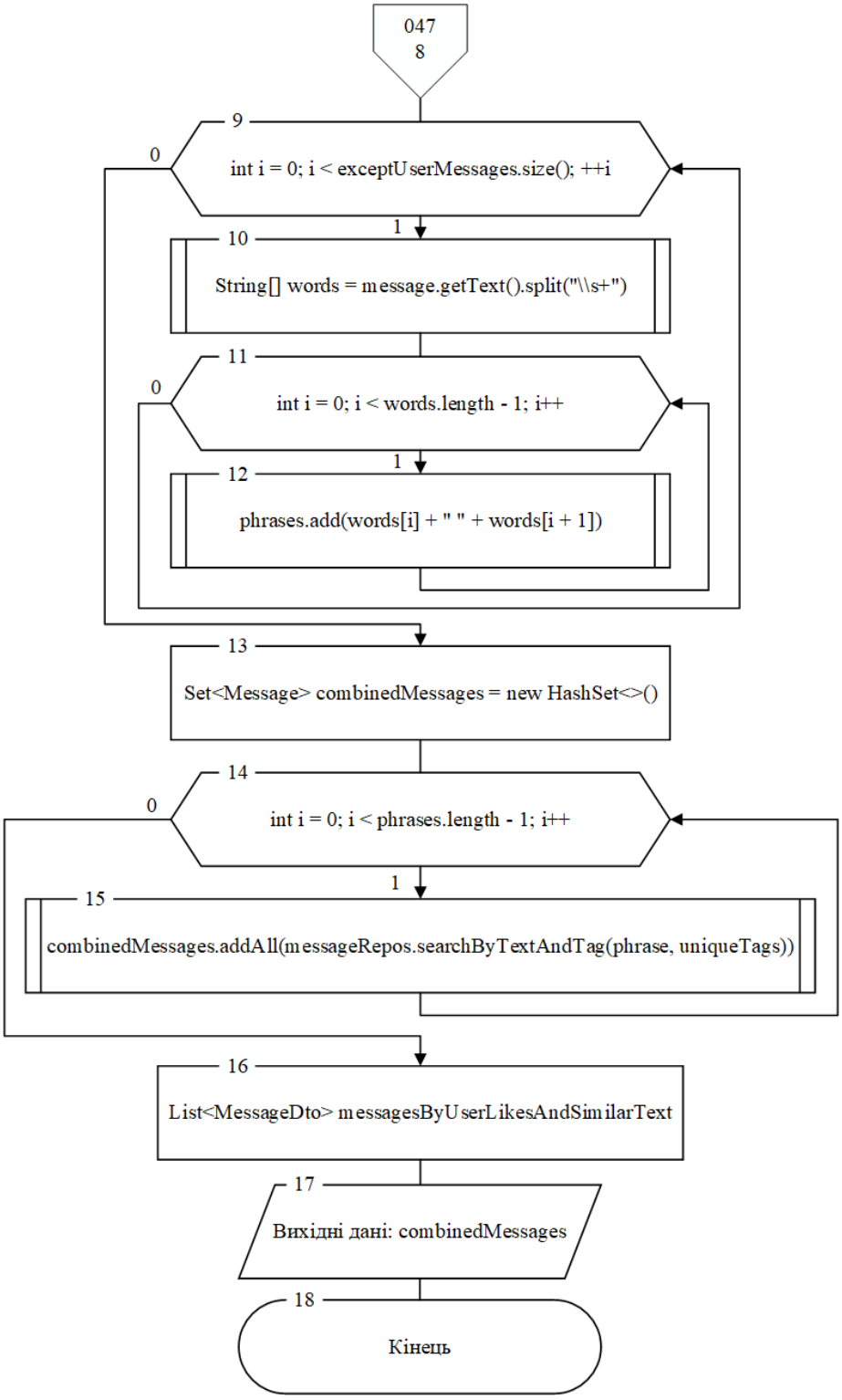

Рисунок 3.7 – Блок-схема алгоритму роботи методу формування відображення постів (частина 2)

Алгоритм роботи методу формування відображення постів складається з кількох важливих етапів, які забезпечують коректне та ефективне відображення контенту для користувачів. Після того, як новий пост збережено в базі даних,

починається процес формування списку постів для відображення на головній сторінці вебзастосунку.

На першому етапі алгоритм отримує всі доступні пости з бази даних. Це включає вибірку постів, що були створені раніше, а також нових постів, щойно доданих до системи. Метод, який відповідає за цей процес, виконує запит до бази даних, щоб отримати всю необхідну інформацію про пости: текст, тег, інформацію про автора та посилання на зображення, якщо вони є. Ці дані збираються та структуруються для подальшого використання.

Далі всі отримані пости додаються до моделі під ключем «messages». Модель є структурою даних, яка використовується для зберігання та передачі інформації між сервером та клієнтом (веббраузером користувача). Вона забезпечує ефективний спосіб організації даних, що дозволяє системі швидко та коректно відображати їх на вебсторінці.

Коли користувач відвідує головну сторінку вебзастосунку, система автоматично використовує дані з моделі для формування та відображення списку постів. Кожен пост включає кілька ключових елементів: текст, тег, інформацію про автора та, якщо наявне, зображення. Ці елементи структуровані та відображаються у визначеному порядку, забезпечуючи зрозумілий та зручний інтерфейс для користувачів.

Окрім базових елементів постів, система також аналізує вподобання користувача для більш персоналізованого відображення контенту. Зокрема, аналізуються ті пости, які користувач вподобав або з якими він взаємодіяв раніше. На основі цього аналізу визначаються теги, що є найбільш релевантними для конкретного користувача. Цей процес дозволяє системі першочергово показувати пости, що містять відповідні теги, підвищуючи тим самим релевантність контенту для кожного окремого користувача.

Такий підхід забезпечує користувачам більш релевантний контент, що сприяє підвищенню їх залученості та задоволеності від використання платформи. Релевантність контенту означає, що користувачі бачать ті пости, які найбільше відповідають їхнім інтересам та потребам, що робить взаємодію з

платформою більш продуктивною та приємною. Крім того, аналіз вподобаних постів та теги, які показуються першими, допомагають користувачам швидко знаходити цікавий для них контент, що також сприяє зростанню активності та взаємодії на платформі.

3.4 Розробка модуля пошуку публікацій за тегами

Модуль пошуку за тегами є невід'ємною частиною функціональності мікроблогінгової платформи, яка спрямована на поліпшення користувацького досвіду шляхом надання можливості вибіркового перегляду контенту. Реалізація цього модуля вимагає глибокого розуміння як користувацьких потреб, так і технічних аспектів взаємодії з базою даних.

Алгоритм роботи методу пошуку публікацій за тегами зображено на рисунку 3.8.

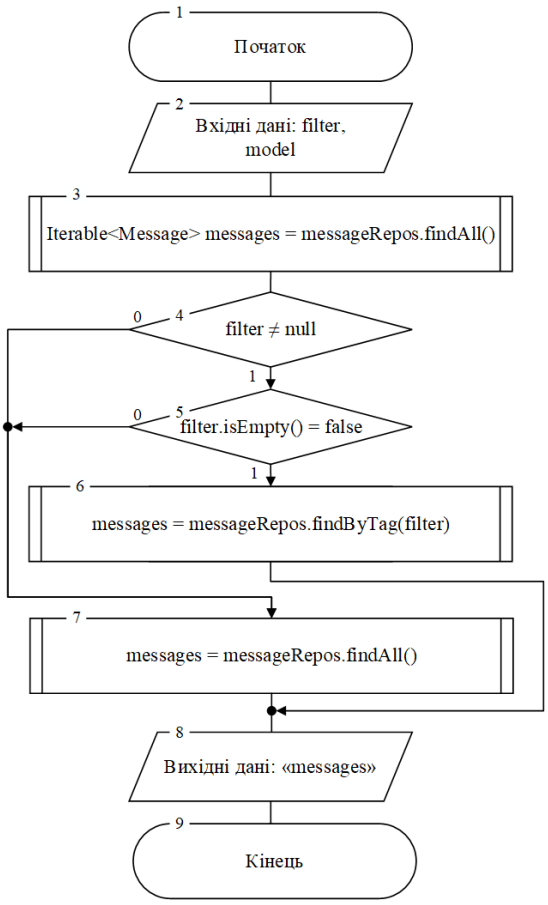

Рисунок 3.8 – Алгоритм роботи методу пошуку публікацій за тегами

Контролер «MainController» відіграє ключову роль у взаємодії користувача з платформою. Його метод «main» є точкою входу для запитів на перегляд головної сторінки, де відображаються всі пости. Метод приймає параметр «filter», який дозволяє користувачам вказувати теги для фільтрації постів. Це забезпечує гнучкість та персоналізацію відображення контенту, дозволяючи користувачам зосередитися на темах, які їх найбільше цікавлять.

«MessageRepos» є інтерфейсом, який визначає методи для взаємодії з базою даних. Використання методу «findByTag» дозволяє ефективно відфільтровувати пости за заданим тегом. Це оптимізує процес пошуку, оскільки система витягує лише ті записи, які відповідають критеріям фільтрації, замість того, щоб переглядати всю базу даних.

На стороні клієнта, шаблонізатор Thymeleaf використовується для динамічного відображення постів. Цикл «th:each» дозволяє ітерувати через колекцію «messages», відображаючи кожен пост із його унікальним ідентифікатором, текстом, тегом та ім'ям автора. Якщо до посту додано зображення, воно відображається за допомогою тегу `img`, що забезпечує візуальне представлення контенту. Це створює більш залучений та інтерактивний досвід для користувачів, які можуть візуально взаємодіяти з постами, що їх цікавлять.

Завдяки цим елементам, модуль пошуку за тегами не лише спрощує навігацію по контенту, але й стимулює користувачів до активнішої участі в освітньому процесі, ділення знаннями та взаємодопомоги. Він є прикладом того, як технології можуть бути використані для створення більш ефективного та змістовного освітнього середовища.

3.5 Розробка модуля аналітичних інструментів для моніторингу активності

У сучасному світі ефективне управління даними є критичним для підтримки успішних освітніх ініціатив. З цією метою було розроблено модуль аналітичних інструментів, який дозволяє моніторити активність користувачів,

аналізувати їх взаємодії та виявляти ключові тренди. Цей модуль є невід'ємною частиною мікроблогінгової соціальної мережі, створеної для підтримки освітніх потреб. У цьому розділі детально описано структуру, функціональність та технічні аспекти розробки цього модуля.

На рисунку 3.9 та 3.10 зображено загальний алгоритм роботи модуля аналітики, який обробляє дані про теги повідомлень, кількість користувачів, адміністраторів та повідомлень, а також перетворює ці дані у формат JSON для подальшого відображення на аналітичній сторінці.

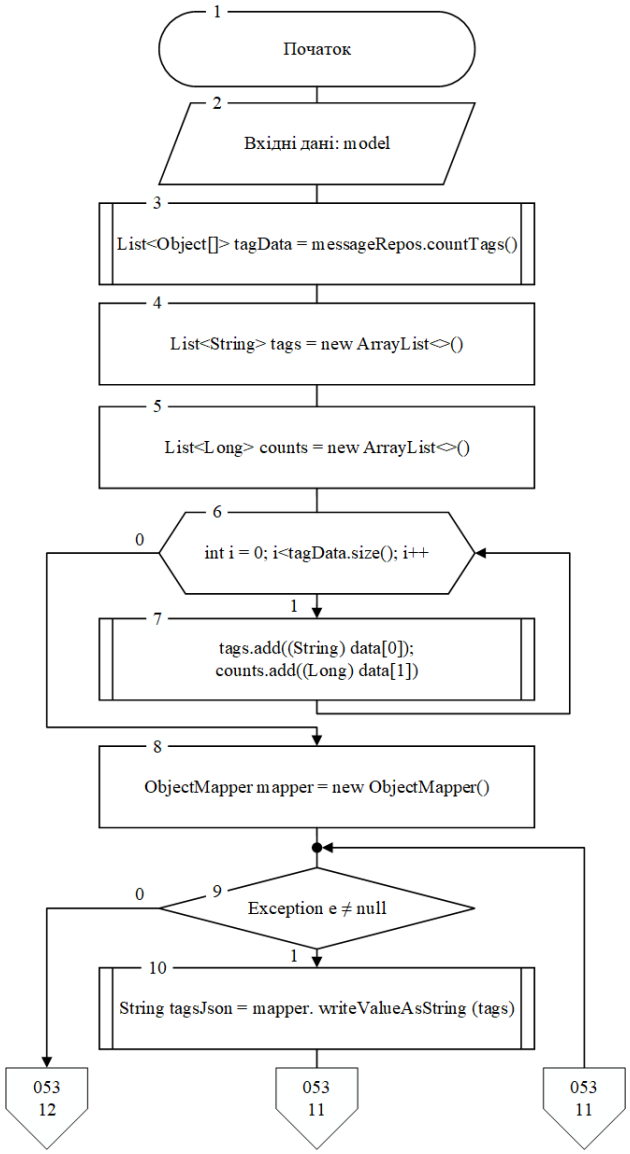

Рисунок 3.9 – Загальний алгоритм роботи модуля аналітики для моніторингу активності (частина 1)

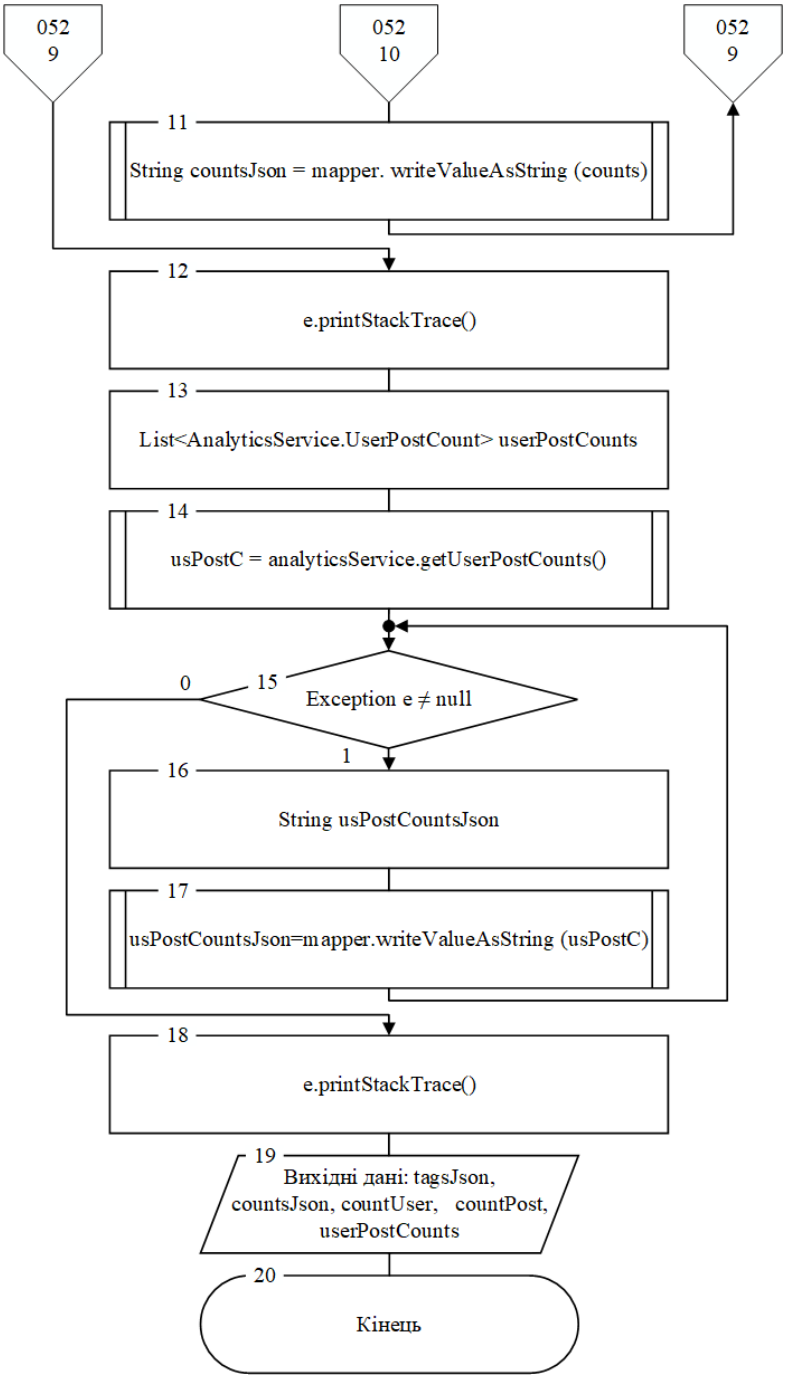

Рисунок 3.10 – Загальний алгоритм роботи модуля аналітики для моніторингу активності (частина 2)

Алгоритм починається з отримання вхідних даних моделі для відображення. Спершу викликається метод «messageRepos.countTags()» для отримання даних про теги. Далі створюються нові списки для тегів та кількостей повідомлень. Потім відбувається заповнення цих списків на основі отриманих даних.

Після заповнення списків, теги та кількості додаються до відповідних списків. Далі відбувається ініціалізація об’єкта «ObjectMapper» для перетворення даних у формат JSON. Наступним кроком є перевірка на виключення під час перетворення тегів у JSON. Якщо виключень немає, теги перетворюються у формат JSON. Аналогічно відбувається і з перетворенням кількостей у JSON.

У разі виникнення виключення, алгоритм виводить інформацію про помилку. Після цього викликається метод «analyticsService.getUserPostCounts()», який отримує кількість повідомлень користувачів. Ці дані також перетворюються у формат JSON.

Наступним кроком є перевірка на виключення під час перетворення даних у JSON. Якщо виключень немає, перетворені у JSON дані додаються до моделі. У разі виникнення виключення алгоритм знову виводить інформацію про помилку. Нарешті, алгоритм підготовлює вихідні дані для відображення на аналітичній сторінці і завершує свою роботу.

Контролер «AnalyticsController» відповідає за обробку запитів до аналітичної сторінки та взаємодію з моделлю для підготовки даних, які будуть відображені користувачеві. Контролер використовує три основні компоненти: репозиторій повідомлень «MessageRepos», репозиторій користувачів «UserRepos» та сервіс аналітики «AnalyticsService».

Коли користувач надсилає запит до аналітичної сторінки, контролер звертається до методу countTags() з «MessageRepos», який повертає список тегів і кількість повідомлень з кожним тегом. Отримані дані обробляються для створення окремих списків тегів та їх кількостей. Ці списки потім перетворюються у формат JSON за допомогою «ObjectMapper» і додаються до моделі, щоб їх можна було відобразити на вебсторінці.

Далі контролер отримує загальну статистику з «UserRepos»: загальну кількість користувачів, кількість адміністраторів та кількість повідомлень. Ці значення також додаються до моделі для відображення на сторінці.

Крім того, контролер викликає метод getUserPostCounts() з «AnalyticsService», який повертає список користувачів разом із кількістю їх повідомлень. Ці дані також перетворюються у JSON і додаються до моделі.

Сервіс «AnalyticsService» відповідає за бізнес-логіку отримання даних для аналітики. Основна функція цього сервісу полягає у виклику методу countUserPosts() з «MessageRepos» для отримання кількості повідомлень, створених кожним користувачем. Отримані дані перетворюються у список об’єктів UserPostCount, кожен з яких містить ім'я користувача та кількість його повідомлень. Цей список потім повертається контролеру для подальшої обробки та відображення.

Репозиторій «MessageRepos» забезпечує доступ до даних повідомлень у базі даних. Він містить методи для пошуку повідомлень за тегом, пошуку повідомлень за ідентифікатором, а також для підрахунку повідомлень за тегами та підрахунку повідомлень для кожного користувача. Метод countTags() групує повідомлення за тегами і повертає кількість повідомлень для кожного тегу. Метод countUserPosts() об'єднує повідомлення з їхніми авторами і підраховує кількість повідомлень для кожного користувача.

Репозиторій «UserRepos» відповідає за доступ до даних користувачів у базі даних. Він містить методи для пошуку користувачів за ім'ям користувача, за активаційним кодом та за електронною поштою. Крім того, репозиторій містить метод countByRole(), який підраховує кількість користувачів, що мають певну роль (наприклад, роль адміністратора).

Модуль аналітики розроблений з використанням сучасних технологій та підходів, що забезпечують надійність і продуктивність. Контролер, сервіси та репозиторії взаємодіють для отримання та обробки необхідних даних, що дозволяє користувачам отримувати детальну інформацію про активність на платформі. Така архітектура дозволяє легко масштабувати систему та додавати нові функції у майбутньому.

3.6 Висновки

У цьому розділі було розглянуто розробку програмних модулів для мікроблогінгової соціальної мережі з орієнтацією на підтримку освітніх ініціатив. Вибір технологій, таких як Java, Spring Boot, Spring Security, Thymeleaf та PostgreSQL, був обґрунтований їх перевагами у швидкості розробки, безпеці, динамічному вебінтерфейсі та надійності бази даних. Використання цих технологій дозволило створити стабільну та безпечну платформу, яка здатна витримувати високе навантаження та забезпечувати надійне зберігання даних.

Розроблені модулі включають функціонал для реєстрації та автентифікації користувачів, публікації та перегляду постів, пошуку публікацій за тегами, а також модуль для аналітичних інструментів, які дозволяють моніторити активність користувачів. Ці модулі забезпечують безпеку, ефективну взаємодію з платформою та стимулюють активну участь користувачів у процесі освіти. Крім того, функціонал аналітики сприяє більш точному розумінню потреб користувачів та оптимізації роботи платформи. Таким чином, розроблені програмні засоби сприяють створенню інтерактивного та безпечного середовища для освітніх ініціатив.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування системи є невід’ємною частиною процесу розробки програмного забезпечення, яке забезпечує високу якість кінцевого продукту та задоволення потреб користувачів. Воно включає в себе різноманітні методи та підходи, які дозволяють ідентифікувати та усунути помилки, забезпечити стабільність роботи системи та її відповідність вимогам користувачів. Тестування допомагає виявити недоліки у дизайні інтерфейсу, логіці роботи системи, а також перевірити безпеку зберігання та обробки даних [21].

Функціональне тестування вебзастосунків зосереджується на перевірці кожної функції системи згідно з визначеними специфікаціями та вимогами [22]. Це включає тестування інтерфейсів користувача, взаємодії з базою даних, інтеграції з іншими системами, а також валідації введення даних.

Тестування відіграє ключову роль у забезпеченні інтуїтивно зрозумілого та ефективного користувацького досвіду, а також у підтримці безперервної та надійної роботи вебзастосунку. Завдяки тестуванню можна забезпечити, що система буде правильно реагувати на дії користувачів, надавати коректні повідомлення про помилки та вказівки для їх виправлення.

Таким чином, тестування сприяє підвищенню якості продукту та забезпеченню позитивного досвіду користувачів.

Тестування вебзастосунку мікроблогінгової соціальної мережі є ключовим етапом розробки, який забезпечує виявлення та усунення помилок перед запуском платформи.

Для мікроблогінгової соціальні мережі, тестування є важливим, оскільки ця платформи обробляє великі обсяги даних та взаємодій користувачів у реальному часі. Тестування допомагає переконатися, що алгоритми працюють коректно.

На прикладі сторінки реєстрації, тестування інтерфейсу дозволяє виявити, як система реагує на помилки введення даних. Якщо користувач пропускає обов’язкові поля, система повинна не тільки сповістити про це, але й надати чіткі вказівки щодо виправлення помилок. На фотографії модального вікна видно, що у випадку не заповнення одного або декількох обов’язкових полів, система відображає повідомлення «Не всі поля заповнені» (див. рисунок 4.1). Це дозволяє користувачу миттєво зрозуміти, що необхідно повернутися до форми та заповнити пропущені поля.

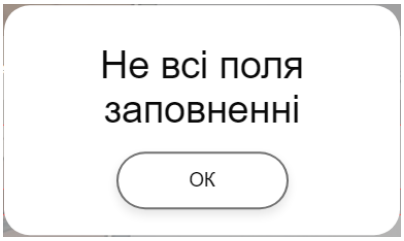

Рисунок 4.1 – Повідомлення про помилку «Не всі поля заповненні»

Додатково, на рисунку 4.2 після закриття модального вікна можна побачити поля, які не були заповнені, підсвічуються червоним кольором. Це візуальний засіб, який привертає увагу користувача до полів, які потребують редагування, що підвищує швидкість та ефективність процесу реєстрації.

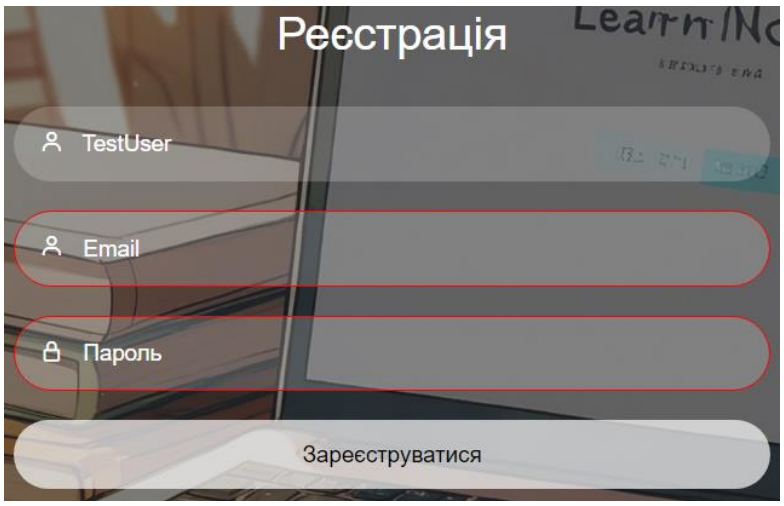

Рисунок 4.2 – Візуалізація помилки введених даних

Такий підхід до дизайну інтерфейсу та системи повідомлень є прикладом забезпечення зручності використання вебзастосунку. Він також відіграє важливу роль у запобіганні помилок, що можуть призвести до неправильного введення даних, та забезпечує високий рівень користувацького досвіду.

Тестування відповідей системи на спробу реєстрації з уже існуючим нікнеймом зображено на рисунку 4.3.

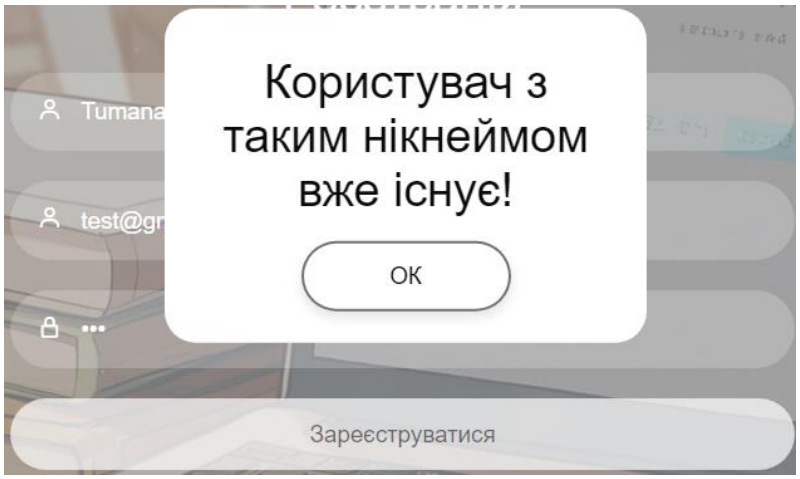

Рисунок 4.3 – Повідомлення про помилку «Користувач з таким нікнеймом вже існує!»

На фотографії модального вікна видно, що коли користувач намагається зареєструватися з нікнеймом, який вже існує в системі, він отримує повідомлення «Користувач з таким нікнеймом вже існує». Система ефективно перевіряє унікальність логінів та надає користувачу зрозумілу відповідь, яка спонукає до вибору іншого варіанту нікнейму.

Тестування системних відповідей на спроби реєстрації є критично важливим для забезпечення безперебійної роботи вебзастосунку та задоволення потреб користувачів. Цей процес включає перевірку реакції системи на різноманітні сценарії, включаючи спроби створення облікового запису з використанням електронної адреси, яка вже зареєстрована в системі. Така перевірка дозволяє виявити потенційні проблеми інтерфейсу та логіки перевірки

даних, а також забезпечити, що користувачі отримують зрозумілі та корисні повідомлення про помилки.

На рисунку 4.4 представлено результати тестування, де система інформує користувача про те, що введена електронна адреса вже використовується іншим користувачем. Повідомлення «Емейл зайнятий!» не лише сповіщає про конкретну помилку, але й надає користувачу чітке розуміння того, що для продовження реєстрації необхідно ввести альтернативну електронну адресу. Це важливо для запобігання зайвому зверненню до служби підтримки та для підвищення загальної задоволеності користувачів.

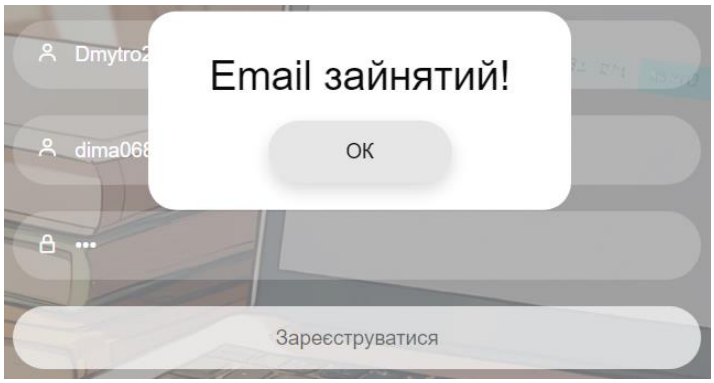

Рисунок 4.4 – Повідомлення про помилку «Email вже зайнятий!»

Ці тестові сценарії підтверджують, що система реєстрації користувачів ретельно перевіряє введені дані та надає користувачам зрозумілі інструкції для виправлення помилок при реєстрації.

У рамках тестування системи авторизації користувачів було також проведено перевірку реакції вебзастосунку на введення неіснуючих даних. На рисунку 4.5 модального вікна, яке з’являється після спроби авторизації з неправильними даними, видно повідомлення «Такого користувача не існує!». Це свідчить про те, що система ефективно ідентифікує випадки, коли користувач намагається увійти з даними, які не відповідають жодному обліковому запису в базі даних.

Це повідомлення про помилку є важливим для інформування користувача про невдалий вхід та необхідність перевірки введених даних. Такий механізм

сприяє забезпеченню безпеки системи, запобігаючи несанкціонованому доступу до користувацьких облікових записів.

Тестування показало, що вебзастосунок має належні засоби для валідації даних користувачів під час авторизації, що є ключовим для забезпечення надійності та безпеки користувацького досвіду. Завдяки цьому користувачі можуть бути впевнені, що їхні дані захищені, а система реагує адекватно на спроби не авторизованого доступу.

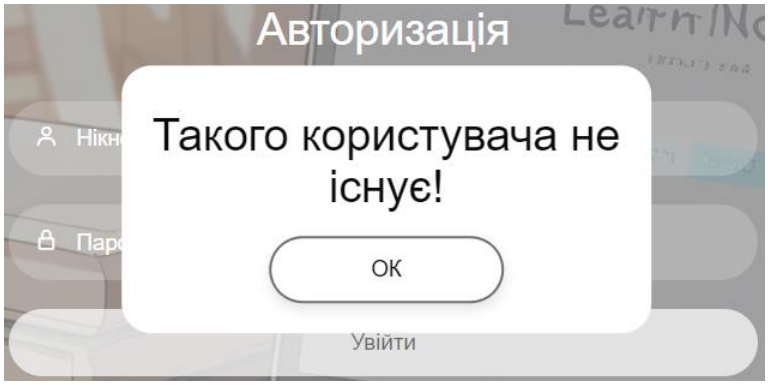

Рисунок 4.5 – Повідомлення про помилку «Такого користувача не існує!»

Тут має бути текст про форму автентифікації і що на рисунку 4.6 зображено форму з валідними даними. Після рисунку теж слід щось сказати про цю форму

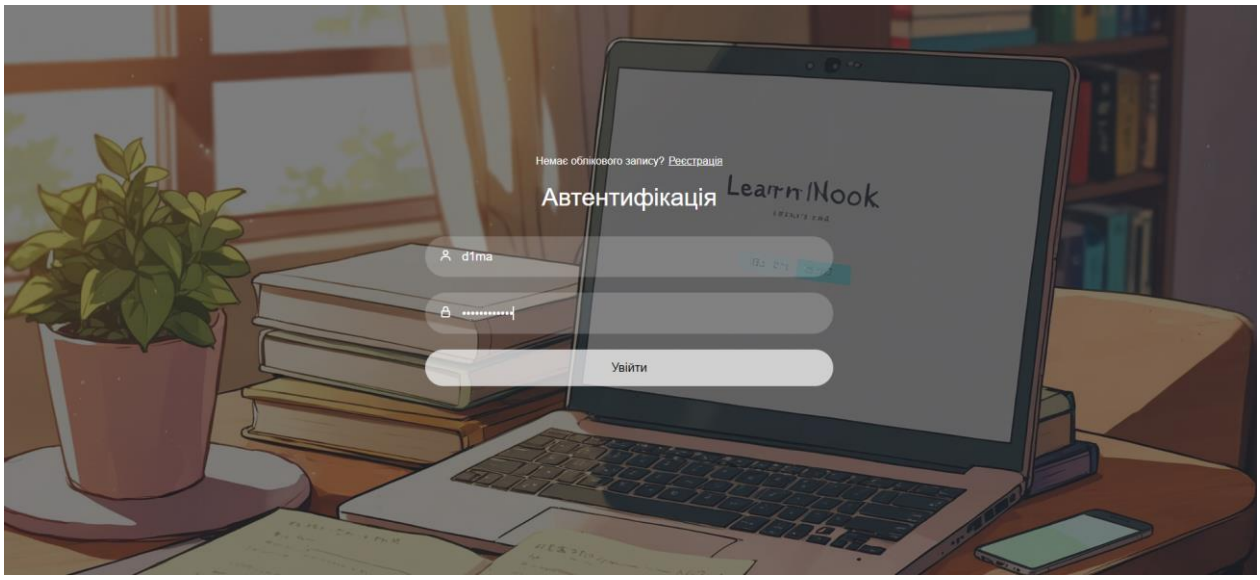

Рисунок 4.6 – Форма автентифікації з валідними даними

Під час тестування веб-застосунків, особливо тих, що вимагають авторизації, важливо перевірити, як сервер повертає та керує cookies. Cookies є критичним компонентом, який дозволяє веб-сайтам запам’ятовувати стан сесії користувача, забезпечуючи безперервність взаємодії та персоналізацію досвіду користувача.

Використання інструментів розробника для перегляду cookies дозволяє розробникам переконатися, що cookies встановлюються правильно та містять відповідні атрибути безпеки, такі як Secure, HttpOnly, та SameSite. Нижче на рисунку 4.7 зображено перевірку cookies після аутентифікації.

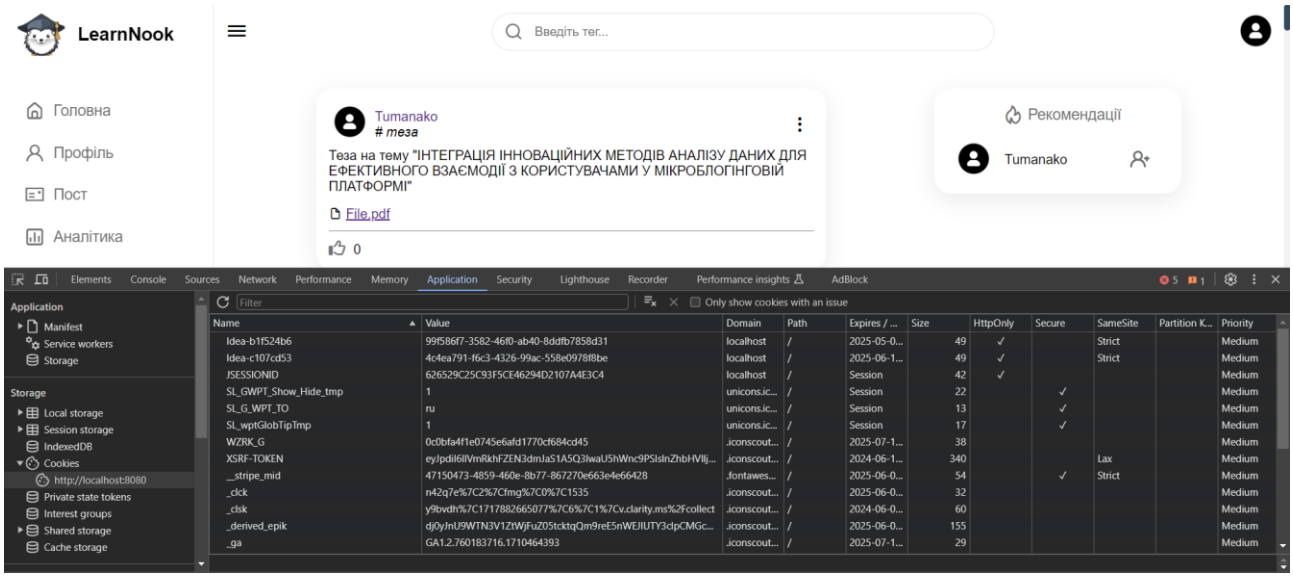

Рисунок 4.7 – Перегляд Cookies через Developer Tools

Перевірка cookies після аутентифікації показала, що веб-застосунок коректно встановлює необхідні cookies, що є важливим для забезпечення безпеки сесії та зручності користувачів. Після успішної аутентифікації система створює cookies, які містять інформацію про сеанс користувача, такі як унікальний ідентифікатор сесії. Ці cookies дозволяють підтримувати безперервність сесії, забезпечуючи користувачам доступ до обмежених ресурсів без повторного введення облікових даних.

Належне управління cookies є критичним для захисту даних користувачів і забезпечення конфіденційності їхньої інформації. Встановлені cookies

зберігаються у браузері користувача і передаються до сервера з кожним запитом, що дозволяє серверу ідентифікувати користувача та підтвердити його права доступу. Це також запобігає можливим атакам, таким як викрадення сесій або несанкціонований доступ, оскільки cookies налаштовані так, щоб зберігати важливі параметри безпеки, такі як час життя сесії та використання захищених з’єднань.

Після проходження автентифікації користувачів зустрічає сторінка завантаження, яка відіграє ключову роль у формуванні першого враження про платформу (див. рисунок 4.8).

Рисунок 4.8 – Зображення інтерфейсу сторінки завантаження

Сторінка завантаження також служить як момент очікування, під час якого користувачі можуть налаштуватися на продуктивну взаємодію з вебзастосунком.

Далі користувач потрапляє на головному сторінку, де присутня навігаційна панель, розташована зліва. Ця панель містить значки або тексти, що відповідають основним функціям платформи. При натисканні на три риски, відбувається показ або приховування бокового меню.

Одним з пунктів навігаційної панелі є створення нового поста, що користувачам швидкий доступ до основних функцій. Ця опція є важливою, адже вона дозволяє залишатися активними у спільноті, ділитися своїми думками та ідеями без зайвих затримок.

Під час тестування цієї функції було встановлено, що інтерфейс реагує на дії користувача з високою чутливістю. Якщо користувач намагається опублікувати пост без вказання тегу, система негайно реагує, підсвічуючи поле для введення тегу червоним кольором, що є зрозумілим сигналом про необхідність додаткової дії з боку користувача. Такий візуальний зворотний зв'язок є важливим елементом користувацького досвіду, оскільки він допомагає запобігти помилкам та спрощує процес публікації контенту. Деталі цього процесу та відповіді інтерфейсу на дії користувача можна побачити на рисунку 4.9.

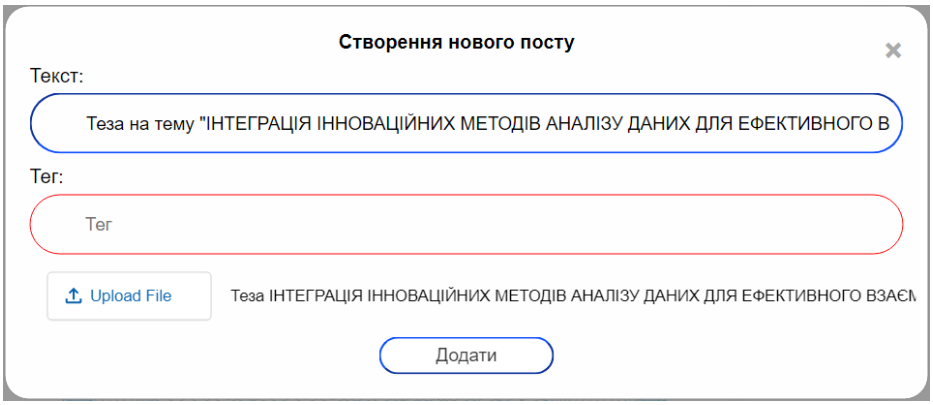

Рисунок 4.9 – Візуаліація валідації даних форми створення нового посту

Ця візуальна підказка є частиною системи валідації введення, яка забезпечує, що користувачі не пропускають важливі поля при створенні посту. Використання кольорових підказок допомагає користувачам швидко ідентифікувати та виправити помилки, тим самим підвищуючи загальну якість взаємодії з вебзастосунком. Такий підхід до дизайну інтерфейсу сприяє зручності та ефективності користувацького досвіду.

Після відправки форми створення нового посту, відправляється HTTP POST запит, який включає всі введені користувачем дані, такі як текст посту, вибраний тег, та, за необхідності, прикріплене зображення. Цей запит надходить на сервер, де він обробляється відповідним контролером. Контролер перевіряє коректність отриманих даних, здійснює валідацію та, у разі наявності зображення, перевіряє його на відповідність необхідним вимогам.

Після успішної обробки даних сервер створює новий об’єкт посту, який включає текст, тег, інформацію про автора та шлях до збереженого зображення (див. рисунок 4.10). Потім цей об’єкт зберігається у базі даних. Процес збереження включає запис усіх необхідних атрибутів посту у відповідні таблиці бази даних, що забезпечує їх подальше швидке отримання та відображення.

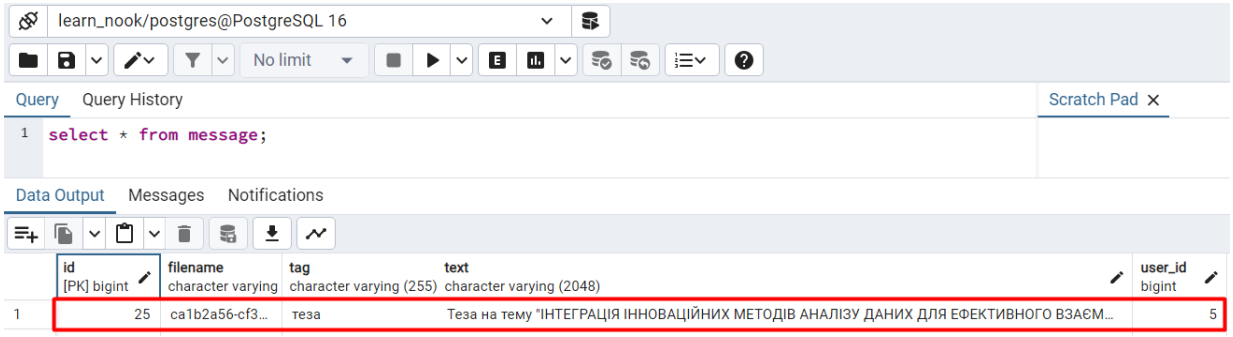

Рисунок 4.10 – Відображення створенного поста в базі даних

Після чого збережний пост відображається власне на головній сторінці (див. рисунок 4.11).

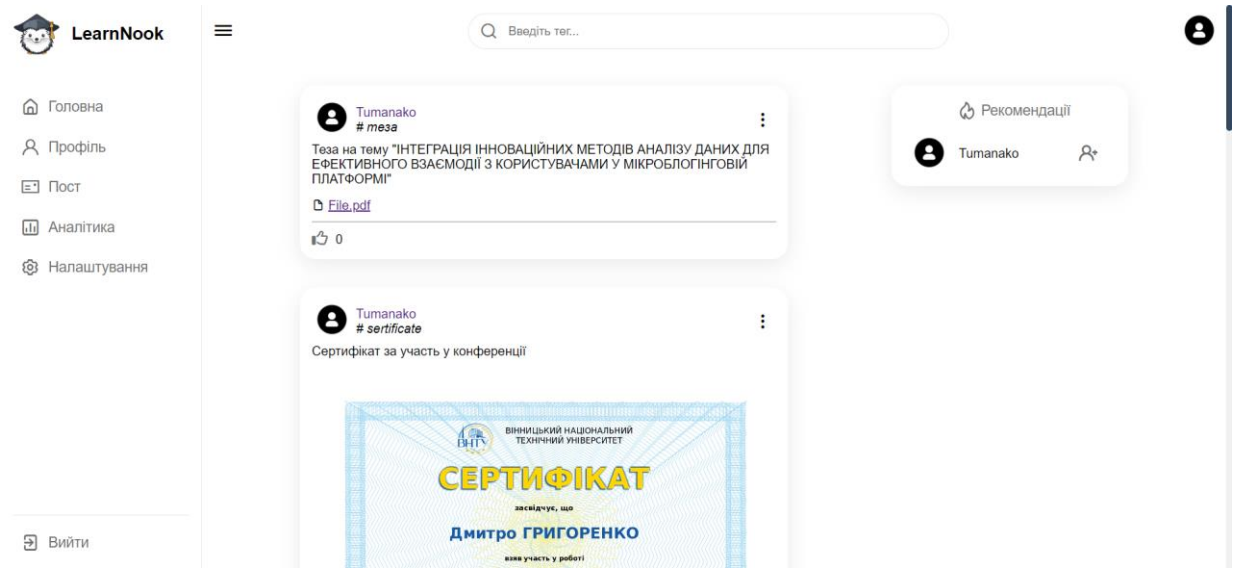

Рисунок 4.11 – Відображення створеного посту на головному сторінці

У верхній частині центральної області головної сторінки розташована стрічка пошуку по тегам мікроблогів, що дозволяє користувачам швидко знаходити контент за певними вибраними тегами.

Тестування сторінки профілю користувача, представленої на рисунку 4.12, зосереджене на оцінці її зручності та ефективності для користувача. Під час тестування перевіряється коректність відображення інформації.

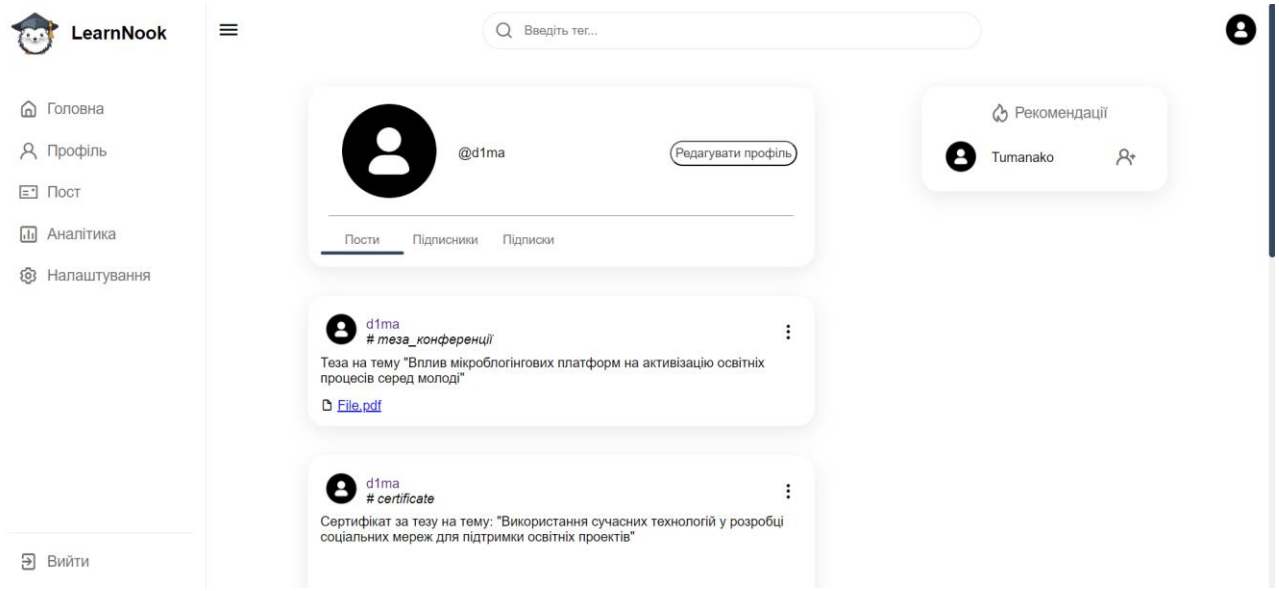

Рисунок 4.12 – Інтерфейс сторінки профілю користувача

Основна інформація про профіль користувача включає його пости, список підписників та іншу важливу інформацію. Тестування цього розділу включає перевірку коректності відображення всіх елементів, а також роботу підменю навігації, яке дозволяє швидко знаходити потрібну інформацію. Підменю забезпечує доступ до таких розділів як «Пости», «Підписки», «Підписники» та інші, забезпечуючи користувачу швидкий доступ до всіх його активностей.

Сторінка аналітики, представлена, була ретельно протестована для перевірки точності та надійності відображення ключових показників активності користувачів. Під час тестування було звернуто особливу увагу на точність графіків активності, які демонструють динаміку публікацій протягом різних періодів часу. Ці графіки дозволяють користувачам легко аналізувати свою активність, порівнюючи її з активністю інших користувачів на платформі.

Тестування підтвердило, що графіки коректно відображають усі необхідні дані, зокрема кількість публікацій, лайків, коментарів та інших взаємодій, що відбувалися протягом обраного часу. Інтерфейс аналітики забезпечує інтуїтивно

зрозумілу навігацію, що дозволяє користувачам без зайвих зусиль налаштовувати відображення даних відповідно до своїх потреб.

Власне інтерфейс сторінки аналітики зображено на рисунку 4.13.

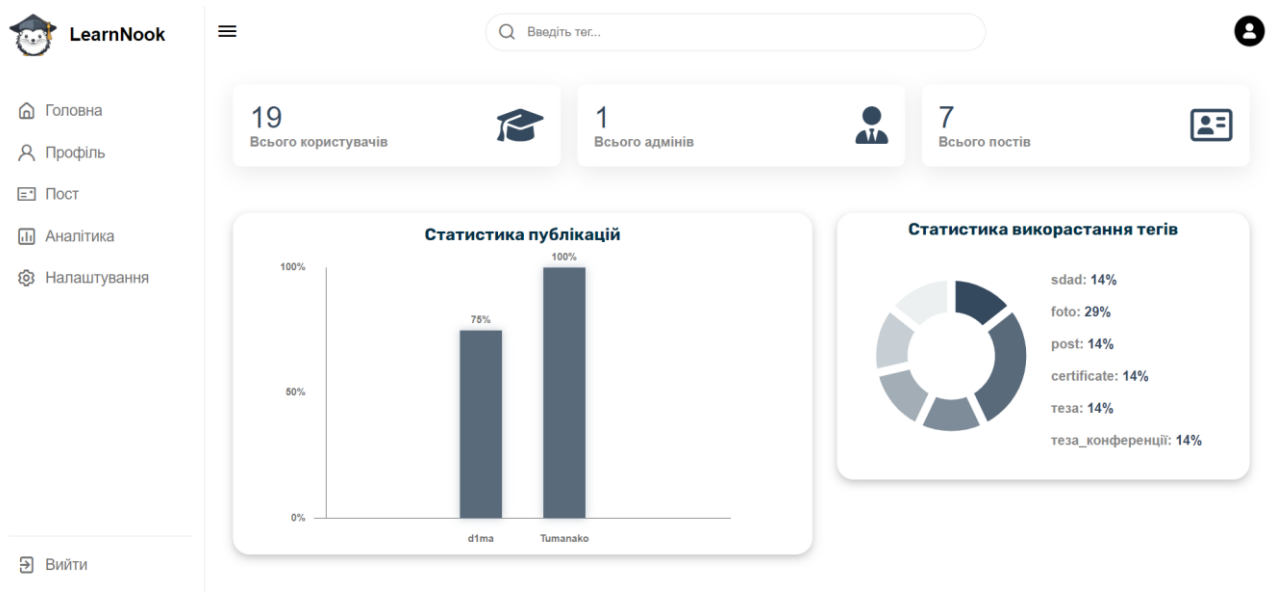

Рисунок 4.13 – Інтерфейс сторінки аналітики

Загальне тестування інтерфейсу мікроблогінгової соціальної мережі показало, що вибір кольорової палітри забезпечує максимальну читабельність контенту та зручність навігації. Кольори ефективно використовуються для виділення важливих елементів і створення контрасту.

Аналіз реакції користувачів на різні елементи інтерфейсу дозволив виявити деякі недоліки та покращити загальний користувацький досвід. В результаті, інтерфейс виявився не лише зручним у використанні, а й сприяє зосередженню користувачів на основних функціях та контенті мікроблогінгової мережі.

4.2 Розробка інструкції користувача

Інструкція користувача містить інформацію про технічні вимоги для запуску та коректної роботи програмного продукту «Мікроблогінгова соціальна

мережа для освітніх ініціатив». Деталі щодо мінімальної та рекомендованої конфігурації користувацького пристрою представлені в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація

Компонент	Вимоги
Операційна система	Не вимагається специфічна ОС, лише підтримка сучасного веббраузера
Браузер	Chrome 80, Firefox 75, Safari 12, Edge 80
Швидкість інтернет-з’єднання	512 Кбіт/с
Оперативна пам’ять	512 МБ
Роздільна здатність екрану	1024x768 пікселів для настільних комп’ютерів, 360x640 пікселів для мобільних пристроїв

Таблиця 4.2 – Рекомендована конфігурація

Компонент	Вимоги
Операційна система	Не вимагається специфічна ОС, оптимізовано для останніх версій веббраузерів
Браузер	Останні версії Chrome, Firefox, Safari, Edge
Швидкість інтернет-з’єднання	100 Мбіт/с або вище
Оперативна пам’ять	2048 МБ або більше
Роздільна здатність екрану	1920x1080 пікселів для настільних комп’ютерів, 720x1280 пікселів для мобільних пристроїв

Після запуску програми, користувач зустрічається з екраном завантаження, який відображає назву додатку та його логотип, створюючи перше враження про стиль та цілі платформи. На цьому екрані також розміщена кнопка «Перейти на головну», яка є основним засобом навігації для користувача.

Якщо користувач ще не авторизований, натискання на кнопку ініціює перехід до сторінки авторизації, де можна ввести логін та пароль. У випадку,

якщо користувач новий або бажає створити новий аккаунт, присутня опція переходу на сторінку реєстрації. Процес реєстрації простий та інтуїтивно зрозумілий, забезпечуючи зручність та швидкість створення аккаунта.

Після успішної авторизації користувач потрапляє на головну сторінку, яка є центром взаємодії в мережі, де можна створювати нові пости, ділитися різними доступними видами ресурсів, а також переглядати контент інших користувачів.

Фільтрація постів за тегами є потужним інструментом для пошуку контенту за інтересами. Користувач може швидко знайти пости на певну тему, що робить навігацію по платформі ефективною та зручною.

Для адміністраторів розроблено спеціальну адміністративну панель, доступ до якої обмежений правами користувача. Адміністратори можуть керувати ролями користувачів, надаючи або відбираючи права адміністратора, що забезпечує гнучке управління спільнотою.

Розроблені інструкції користувача та вимоги конфігурації користувацького пристрою забезпечують зрозумілість та доступність мікроблогінгової соціальної мережі для освітніх ініціатив. Вони враховують необхідні технічні параметри для ефективної роботи платформи на різних пристроях, забезпечуючи користувачам комфортне та продуктивне середовище для обміну інформацією та взаємодії. Завдяки детальному опису процесів авторизації та навігації, користувачі можуть легко адаптуватися до інтерфейсу мережі та її функціоналу. Панель навігації надає необхідні інструменти для управління контентом, що підвищує ефективність управління даними мережі.

4.3 Висновки

У четвертому розділі було проведено детальне тестування всіх програмних модулів системи, включаючи функціональне тестування користувацьких інтерфейсів, перевірку взаємодії з базою даних та інтеграцію з іншими системами. Деяка увага була приділена валідації введених даних, що забезпечує коректну роботу системи і її стійкість до помилок користувачів. Результати

тестування підтвердили ефективність повідомлень про помилки та вказівок для їх виправлення, що значно покращує користувацький досвід і зменшує ймовірність виникнення помилок при введенні даних.

Також було розроблено детальну інструкцію користувача, яка містить технічні вимоги для запуску системи. Інструкція забезпечує користувачам легкий доступ до всіх необхідних функцій системи та сприяє швидкому освоєнню інтерфейсу, що робить платформу більш зручною та доступною для широкого кола користувачів. Завдяки цьому, система готова до впровадження та забезпечує високий рівень задоволення потреб користувачів.

ВИСНОВКИ

У процесі виконання бакалаврської дипломної роботи було розроблено програмні засоби для мікроблогінгової соціальної мережі, спрямованої на підтримку освітніх ініціатив. Спершу було здійснено всебічний аналіз сучасного стану мікроблогінгових соціальних мереж у контексті освітніх ініціатив, що дозволило виділити основні проблеми та недоліки існуючих платформ. Цей аналіз став основою для постановки завдань проєкту. Бакалаврську дипломну роботу було виконано відповідно до рекомендацій, наведених у документі [23].

У ході роботи було проаналізовано та порівняно існуючі платформи, такі як Edmodo, Schoology, та LearnNook, що дозволило виявити їх переваги і недоліки та обґрунтувати актуальність створення нової платформи. Для розробки було обрано мову програмування Java та фреймворк Spring Boot як основні інструменти, а також базу даних PostgreSQL та шаблонізатор Thymeleaf для побудови вебінтерфейсу.

Під час розробки програмного продукту було створено зручний графічний інтерфейс програми. Реалізовано функціонал реєстрації та авторизації користувачів, забезпечено можливість створення користувацьких записів, публікацій та їх перегляду. Також було розроблено модуль для пошуку публікацій за тегами та модуль аналітичних інструментів для моніторингу активності користувачів.

Проведено всебічне тестування програмних модулів системи, включаючи валідацію полів введення інформації, результати якого підтвердили повну функціональність програми та відповідність технічному завданню. Крім того, було розроблено інструкцію користувача для встановлення та початкової експлуатації програмного продукту, а також підготовлено UML-діаграми для візуалізації моделі системи та алгоритму функціонування.

Таким чином, виконана бакалаврська дипломна робота підтвердила актуальність та необхідність створення спеціалізованої мікроблогінгової соціальної мережі для підтримки освітніх ініціатив.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The importance of sharing knowledge in today’s world [Електронний ресурс] – Режим доступу до ресурсу: https://trainual.com/manual/sharing-knowledge (дата звернення: 20.04.2024).

2. Educational needs in the present [Електронний ресурс] – Режим доступу до ресурсу: https:// www.teachmint.com/glossary/e/educational-needs/.

3. Григоренко Д.П. Інтеграція інноваційних методів аналізу даних для ефективної взаємодії з користувачами у мікроблогінговій платформі/ О.М. Рейда, Д.П. Григоренко // Матеріали конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20729/17209.

4. Григоренко Д.П. Вплив мікроблогінгових платформ на активізацію освітніх процесів серед молоді/ О.М. Рейда, Д.П. Григоренко // Матеріали міжнародної науково-практичної інтернет конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences. vntu.edu.ua/index.php/mn/mn2024/paper/view/21400.

5. Advantages of microblogging in education [Електронний ресурс] – Режим доступу до ресурсу: https:// journals.sagepub.com/doi/10.1177/2042753016672130.

6. Integration of educational materials [Електронний ресурс] – Режим доступу до ресурсу: https://www.edugab. com/world/technology-integration-in-education.

7. Microblogging social networks in education [Електронний ресурс] – Режим доступу до ресурсу: https://www.jotse.org/index.php/jotse/article/view/65/87.

8. Edmodo [Електронний ресурс] – Режим доступу до ресурсу: https://www.linkedin.com/pulse/edmodo-crystal-moncada.

9. Schoology [Електронний ресурс] – Режим доступу до ресурсу: https://mi01000971.schoolwires.net/Page/20417.

10. Visio [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/microsoft-365/visio/flowchart-software.

11. Java advantages [Електронний ресурс] – Режим доступу до ресурсу: https://yojji.io/blog/what-are-the-main-pros-and-cons-of-java.

12. Python advantages [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/python-language-advantages-applications/.

13. PHP advantages [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@brandingsolutionllc573/advantages-of-php-in-enterprise-level-web-applications-e62200f35961

14. Spring Boot [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/java-spring-boot.

15. Micronaut advantages [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchapparchitecture/definition/Micronaut-framework.

16. Quarkus advantages [Електронний ресурс] – Режим доступу до ресурсу: https://www.baeldung.com/spring-boot-vs-quarkus

17. Thymeleaf [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/spring-boot-thymeleaf-view.

18. Freemarker [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@dudkamv/comparing-thymeleaf-mustache-and-apache-freemarker-for-java-based-web-development-068b20caee8b.

19. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/rds/postgresql/what-is-postgresql/.

20. MySql [Електронний ресурс] – Режим доступу до ресурсу: https://www.developer.com/design/working-with-javamail-and-the-spring-mail-apis/.

21. Interface testing [Електронний ресурс] – Режим доступу до ресурсу: https://testsigma.com/blog/interface-testing/.

22. Functional testing of web applications [Електронний ресурс] – Режим доступу до ресурсу: https://www.leapwork.com/blog/website-functionality-testing.

23. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад.: О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 51 с.

Додаток А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою ПЗ

д.т.н., проф.

_____ О. Н. Романюк

"12" березня 2024 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка мікроблогінгової соціальної мережі для підтримки освітніх ініціатив» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

_____ к.т.н., доцент О.М. Рейда

« 12 » _____березня_____ 2024 р.

Виконав:

_____ студент гр. 4ПІ-20б Д.П. Григоренко

« 12 » _____березня_____ 2024 р.

Вінниця ВНТУ - 2024 рік

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка мікроблогінгової соціальної мережі для підтримки освітніх ініціатив».

Галузь застосування – освітні технології.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою роботи є забезпечення покращення комунікативних можливостей у сфері освіти за допомогою мікроблогінгової соціальної мережі, яка сприятиме обміну інформацією та ресурсами, підтримуватиме ефективну взаємодію між учасниками освітнього процесу та допоможе у досягненні навчальних та професійних цілей, стимулюючи активний обмін знаннями та співпрацю між студентами, викладачами та іншими учасниками освітнього середовища.

Основними задачами дослідження є:

- провести аналіз існуючих мікроблогінгових систем та визначити їхні можливості для освітніх ініціатив;
- покращити метод формування постів;
- розробка інтерфейсу користувача;
- розробка бази даних, яка ефективно зберігатиме користувацькі дані, освітній контент, забезпечуючи швидкий доступ та високу продуктивність;
- розробка модуля автентифікації та реєстрації користувачів для підтримки додаткових функцій та безпеки платформи;
- розробка модуля публікації та перегляду, який дозволить користувачам публікувати короткі повідомлення, сприяти обміну ідеями та знаннями, а також підтримувати освітні дискусії;

- розробка модуля пошуку публікацій за тегами, що забезпечить користувачам можливість знаходити пости за тегами;
- розробити програмні компоненти та системи: інтерфейс, модуль аутентифікації, модуль публікації та перегляду, модуль пошуку за тегами постів;
- провести експериментальні дослідження розроблених засобів.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР:

1. The importance of sharing knowledge in today’s world [Електронний ресурс] – Режим доступу до ресурсу: https://trainual.com/manual/sharing-knowledge (дата звернення: 20.04.2024).

2. Educational needs in the present [Електронний ресурс] – Режим доступу до ресурсу: https:// www.teachmint.com/glossary/e/educational-needs/ (дата звернення: 20.04.2024).

3. Advantages of microblogging in education [Електронний ресурс] – Режим доступу до ресурсу: https:// journals.sagepub.com/doi/10.1177/2042753016672130.

4. Integration of educational materials [Електронний ресурс] – Режим доступу до ресурсу: https://www.edugab. com/world/technology-integration-in-education.

5. Технічні вимоги

– середовище розробки – IntelliJ IDEA;

– мова програмування – Java;

– вхідні дані: інформація про освітні ресурси, контент для створення постів, обліковий запис користувача;

– вихідні дані: вебзастосунок із графічним користувацьким інтерфейсом для перегляду та взаємодії з освітніми ресурсами.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Обґрунтування вибору теми та постановка задач дослідження	14.03.24 - 20.03.24
2	Аналіз і вибір середовища розробки	21.03.24 - 31.03.24
3	Розробка структури бази даних та проектування інтерфейсу користувача	01.04.24 - 10.04.24
4	Розробка основних програмних модулів	11.04.24 - 20.04.24
5	Інтеграція модулів та розробка мікроблогінгової функціональності	21.04.24 - 10.05.24
6	Тестування розробленого вебзастосунку	11.05.24 - 20.05.24
7	Оформлення матеріалів до захисту БДР	21.05.24 - 30.05.24

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

Додаток Б

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка мікроблогінгової соціальної мережі для підтримки освітніх ініціатив

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доцент кафедри ПЗ, Рейда Олександр Миколайович

Оригінальність	92%
Схожість	8%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Григоренко Д.П.

Керівник роботи _____ Рейда О.М.

Додаток В

Лістинг програмного коду мобільного застосунку

Клас User.java:

```
package com.example.learnNook.domain;

import javax.persistence.*;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

@Entity
@Table(name="usr")
public class User implements UserDetails {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private Long id;
   private String username;
   private String password;

   private String email;
   private String activationCode;

   @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
   @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))
   @Enumerated(EnumType.STRING)
   private Set<Role> roles;

   @OneToMany(mappedBy = "author", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
   private Set<Message> messages;

   @ManyToMany
   @JoinTable(
         name = "user_subscriptions",
         joinColumns = {@JoinColumn(name = "channel_id")},
         inverseJoinColumns = {@JoinColumn(name = "subscriber_id")}
   )
   private Set<User> subscribers = new HashSet<>();

   @ManyToMany
   @JoinTable(
         name = "user_subscriptions",
         joinColumns = {@JoinColumn(name = "subscriber_id")},
         inverseJoinColumns = {@JoinColumn(name = "channel_id")}
```

```
)
private Set<User> subscriptions = new HashSet<>();

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    User user = (User) o;
    return Objects.equals(id, user.id);
}

@Override
public int hashCode() {
    return Objects.hash(id);
}

public void setId(Long id) {
    this.id = id;
}
public void setUsername(String username) {
    this.username = username;
}
public void setPassword(String password) {
    this.password = password;
}

public void setRoles(Set<Role> roles) {
    this.roles = roles;
}
public void setEmail(String email) {
    this.email = email;
}
public void setActivationCode(String activationCode) {
    this.activationCode = activationCode;
}
public void setMessages(Set<Message> messages) {
    this.messages = messages;
}

public void setSubscribers(Set<User> subscribers) {
    this.subscribers = subscribers;
}
public void setSubscriptions(Set<User> subscriptions) {
    this.subscriptions = subscriptions;
}

public Long getId() {
    return id;
}
public String getUsername() {
    return username;
}
```

```
public Set<Message> getMessages() {
   return messages;
}

@Override
public boolean isAccountNonExpired() {
   return true;
}

@Override
public boolean isAccountNonLocked() {
   return true;
}

@Override
public boolean isCredentialsNonExpired() {
   return true;
}

@Override
public boolean isEnabled() {
   return isActive();
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
   return getRoles();
}

public String getPassword() {
   return password;
}
public boolean isActive() {
   return activationCode == null || activationCode.isEmpty();
}
public Set<Role> getRoles() {
   return roles;
}
public String getEmail() {
   return email;
}
public String getActivationCode() {
   return activationCode;
}
public boolean isAdmin() {
   return roles.contains(Role.ADMIN);
}
public Set<User> getSubscribers() {
   return subscribers;
}
public Set<User> getSubscriptions() {
   return subscriptions;
```

```
    }
}
```

Клас Message.java:

```
package com.example.learnNook.domain;
import com.example.learnNook.domain.util.MessageHelper;
import javax.persistence.*;
import java.util.Date;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Message {
    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    private Long id;
    private String text;
    private String tag;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_id")
    private User author;

    private String filename;

    @ManyToMany
    @JoinTable(
            name = "message_likes",
            joinColumns = { @JoinColumn(name = "message_id")},
            inverseJoinColumns = { @JoinColumn(name = "user_id")}
    )
    private Set<User> likes = new HashSet<>();

    public Message() {
    }

    public Message(String text, String tag, User user) {
        this.author = user;
        this.text = text;
        this.tag = tag;
    }

    public String getAuthorName() {
        return MessageHelper.getAuthorName(author);
    }

    public void setId(Long id) {
        this.id = id;
    }
    public void setText(String text) {
        this.text = text;
    }
    public void setTag(String tag) {
        this.tag = tag;
```

```
    }
    public void setAuthor(User author) {
        this.author = author;
    }
    public void setFilename(String filename) {
        this.filename = filename;
    }

    public Long getId() {
        return id;
    }
    public String getText() {
        return text;
    }
    public String getTag() {
        return tag;
    }
    public User getAuthor() {
        return author;
    }
    public String getFilename() {
        return filename;
    }
    public Set<User> getLikes() {
        return likes;
    }
    public void setLikes(Set<User> likes) {
        this.likes = likes;
    }
}
```

Клас MessageRepos.java:

```
package com.example.learnNook.repos;
import com.example.learnNook.domain.Message;
import com.example.learnNook.domain.User;
import com.example.learnNook.domain.dto.MessageDto;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.lang.NonNull;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

import java.util.List;
import java.util.Optional;
import java.util.Set;

public interface MessageRepos extends CrudRepository<Message, Long> {

    @Query("select new com.example.learnNook.domain.dto.MessageDto(" +
            "   m, " +
```

```
            "   count(ml), " +
            "   (sum(case when ml = :user then 1 else 0 end) > 0)" +
            ") " +
            "from Message m left join m.likes ml " +
            "group by m")
    Page<MessageDto> findAll(Pageable pageable, @Param("user") User user);

    List<Message> findByTag(String tag);

    @Query("select new com.example.learnNook.domain.dto.MessageDto(" +
            "   m, " +
            "   count(ml), " +
            "   (sum(case when ml = :user then 1 else 0 end) > 0)" +
            ") " +
            "from Message m left join m.likes ml " +
            "where m.tag = :tag " +
            "group by m")
    Page<MessageDto> findByTag(@Param("tag") String tag, Pageable pageable, @Param("user")
User user);

    @Query("select new com.example.learnNook.domain.dto.MessageDto(" +
            "   m, " +
            "   count(ml), " +
            "   (sum(case when ml = :user then 1 else 0 end) > 0)" +
            ") " +
            "from Message m left join m.likes ml " +
            "where m.author = :author " +
            "group by m")
    Page<MessageDto> findByUser(Pageable pageable, @Param("author") User author,
@Param("user") User user);

    Optional<Message> findById(Long messageId);

    @Query("SELECT m.tag, COUNT(m) as count FROM Message m GROUP BY m.tag")
    List<Object[]> countTags();

    @Query("SELECT u.username, COUNT(m) as postCount FROM Message m JOIN m.author u
GROUP BY u.username")
    List<Object[]> countUserPosts();

    @Query("SELECT DISTINCT m.tag FROM Message m")
    Set<String> findAllTags();

    List<Message> findByTagIn(Set<String> tags);

    List<Message> findByLikesContaining(User user);

    @Query("SELECT m FROM Message m WHERE m.author <> :user")
    List<Message> findAllExceptUser(@Param("user") User user);

    @Query("select m from Message m where m.text like %:keyword%")
    List<Message> findByTextContainingKeyword(@Param("keyword") String keyword);
```

```
    @Query("SELECT m FROM Message m WHERE m.text LIKE %:phrase%")
    List<Message> findByTextContainingPhrase(@Param("phrase") String phrase);

    @Query("SELECT m FROM Message m WHERE m.text LIKE %:phrase% AND m.tag IN
:tags")
    List<Message> findByTextContainingPhraseAndTagIn(@Param("phrase") String phrase,
@Param("tags") Set<String> tags);
}
```

Клас UserRepos.java:

```
package com.example.learnNook.repos;

import com.example.learnNook.domain.Role;
import com.example.learnNook.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface UserRepos extends JpaRepository<User, Long> {

    User findByUsername(String username);

    User findByActivationCode(String code);

    User findByEmail(String email);

    @Query("SELECT COUNT(u) FROM User u JOIN u.roles r WHERE r = :role")
    long countByRole(@Param("role") Role role);

}
```

Клас UserServiceImpl.java:

```
package com.example.learnNook.service;

import com.example.learnNook.domain.Role;
import com.example.learnNook.domain.User;
import com.example.learnNook.repos.UserRepos;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.thymeleaf.util.StringUtils;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.TypedQuery;
import java.awt.print.Pageable;
import java.util.*;
```

```
import java.util.stream.Collectors;

@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserRepos userRepos;
    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private MailSenderService mailSenderService;

    @PersistenceContext
    private EntityManager entityManager;

    public void registerUser(User user) {
        user.setRoles(Collections.singleton(Role.USER));
        user.setActivationCode(UUID.randomUUID().toString());
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userRepos.save(user);

        sendMessage(user);
    }

    private void sendMessage(User user) {
        if (!StringUtils.isEmpty(user.getEmail())) {
            String message = String.format(
                    "Hello, %s \n" +
                            "Welcome to LearnNook. Please, visit next link: http://localhost:8080/activate/%s",
                    user.getUsername(),
                    user.getActivationCode()
            );
            mailSenderService.send(user.getEmail(), "Activstion code", message);
        }
    }

    public User getUser(User user) {
        User userFromDb = userRepos.findByUsername(user.getUsername());
        return userFromDb;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        return userRepos.findByUsername(username);
    }

    public User findById(Long id) {
        return userRepos.findById(id).orElse(null);
    }

    public List<User> findAll() {
        return userRepos.findAll();
    }

    public void saveUser(User user, String username, Map<String, String> form) {
        user.setUsername(username);
```

```
    Set<String> roles = Arrays.stream(Role.values())
            .map(Role::name).collect(Collectors.toSet());

    user.getRoles().clear();

    for (String key : form.keySet()) {
        if (roles.contains(key)) {
            user.getRoles().add(Role.valueOf(key));
        }

    }
    userRepos.save(user);
}

public void updateProfile(User user, String username, String password, String email) {

    String userEmail = user.getEmail();

    boolean isEmailChanged = (email != null && !email.equals(userEmail)) ||
            (userEmail != null && !userEmail.equals(email));

    if (isEmailChanged) {
        user.setEmail(email);

        if (!StringUtils.isEmpty(password)) {
            user.setActivationCode(UUID.randomUUID().toString());
        }
    }

    if (!StringUtils.isEmpty(password)) {
        user.setPassword(passwordEncoder.encode(password));
    }

    if (!StringUtils.isEmpty(username)) {
        user.setUsername(username);
    }

    userRepos.save(user);

    if (isEmailChanged) {
        sendMessage(user);
    }
}

public boolean activateUser(String code) {
    User user = userRepos.findByActivationCode(code);

    if (user == null) {
        return false;
    }

    user.setActivationCode(null);

    userRepos.save(user);

    return true;
```

```
    }

    @Override
    public User findByUsername(String name) {
        return userRepos.findByUsername(name);
    }

    @Override
    public User getUserByEmail(String email) {
        return userRepos.findByEmail(email);
    }

    public void subscribe(User currentUser, User user) {
        user.getSubscribers().add(currentUser);
        userRepos.save(user);
    }

    public void unsubscribe(User currentUser, User user) {
        user.getSubscribers().remove(currentUser);
        userRepos.save(user);
    }

    public void unsubscribeId(Long subscriberId, Long userId) {
        User subscriber = userRepos.findById(subscriberId).orElseThrow(() -> new
IllegalArgumentException("Invalid user Id:" + subscriberId));
        User user = userRepos.findById(userId).orElseThrow(() -> new IllegalArgumentException("Invalid
user Id:" + userId));
        user.getSubscriptions().remove(subscriber);
        userRepos.save(user);
    }
}
```

Клас MessageService.java:

```
package com.example.learnNook.service;

import com.example.learnNook.domain.Message;
import com.example.learnNook.domain.User;
import com.example.learnNook.domain.dto.MessageDto;
import com.example.learnNook.repos.MessageRepos;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

@Service
public class MessageService {
    @Autowired
    private MessageRepos messageRepos;
```

```
    public Page<MessageDto> messageList(Pageable pageable, String filter, User user) {
        if (filter != null && !filter.isEmpty()) {
            return messageRepos.findByTag(filter, pageable, user);
        } else {
            return messageRepos.findAll(pageable, user);
        }
    }

    public Page<MessageDto> messageListForUser(Pageable pageable, User currentUser, User
author) {
        return messageRepos.findByUser(pageable, author, currentUser);
    }

    public List<Message> getMessagesByUserLikes1(User user) {
        List<Message> likedMessages = messageRepos.findByLikesContaining(user);

        Set<String> uniqueTags = new HashSet<>();
        for (Message message : likedMessages) {
            uniqueTags.add(message.getTag());
        }

        return messageRepos.findByTagIn(uniqueTags);
    }


    public List<MessageDto> getMessagesByUserLikes(User user) {
        List<Message> likedMessages = messageRepos.findByLikesContaining(user);

        Set<String> uniqueTags = new HashSet<>();
        for (Message message : likedMessages) {
            uniqueTags.add(message.getTag());
        }

        List<Message> messagesByTags = messageRepos.findByTagIn(uniqueTags);

        List<MessageDto> messagesByUserLikes = messagesByTags.stream()
                .map(message -> new MessageDto(
                        message,
                        (long) message.getLikes().size(),
                        message.getLikes().contains(user)))
                .collect(Collectors.toList());

        return messagesByUserLikes;
    }

    public List<MessageDto> getMessagesBySimilarText1(User user) {
        List<Message> exceptUserMessages = messageRepos.findAllExceptUser(user);

        Set<String> phrases = new HashSet<>();
        for (Message message : exceptUserMessages) {
            String[] words = message.getText().split("\\s+");
```

```
        for (int i = 0; i < words.length - 1; i++) {
            phrases.add(words[i] + " " + words[i + 1]);
        }
    }

    Set<Message> messagesByPhrases = new HashSet<>();
    for (String phrase : phrases) {
        messagesByPhrases.addAll(messageRepos.findByTextContainingPhrase(phrase));
    }

    List<MessageDto> messagesBySimilarText = messagesByPhrases.stream()
            .map(message -> new MessageDto(
                    message,
                    (long) message.getLikes().size(),
                    message.getLikes().contains(user)))
            .collect(Collectors.toList());

    return messagesBySimilarText;
}


public List<MessageDto> getMessagesByUserLikesAndSimilarText(User user) {
    List<Message> likedMessages = messageRepos.findByLikesContaining(user);

    Set<String> uniqueTags = new HashSet<>();
    for (Message message : likedMessages) {
        uniqueTags.add(message.getTag());
    }

    List<Message> exceptUserMessages = messageRepos.findAllExceptUser(user);

    Set<String> phrases = new HashSet<>();
    for (Message message : exceptUserMessages) {
        String[] words = message.getText().split("\\s+");
        for (int i = 0; i < words.length - 1; i++) {
            phrases.add(words[i] + " " + words[i + 1]);
        }
    }

    Set<Message> combinedMessages = new HashSet<>();
    for (String phrase : phrases) {
        combinedMessages.addAll(messageRepos.findByTextContainingPhraseAndTagIn(phrase,
uniqueTags));
    }

    List<MessageDto> messagesByUserLikesAndSimilarText = combinedMessages.stream()
            .map(message -> new MessageDto(
                    message,
                    (long) message.getLikes().size(),
                    message.getLikes().contains(user)))
            .collect(Collectors.toList());
```

```
        return messagesByUserLikesAndSimilarText;
    }
}
```

Клас WebSecurityConfig.java:

```
package com.example.learnNook.config;

import com.example.learnNook.service.UserService;
import com.example.learnNook.service.UserServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuil
der;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurit
y;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;

import javax.sql.DataSource;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public PasswordEncoder getPasswordEncoder(){
        return new BCryptPasswordEncoder(8);
    }

    @Autowired
    private UserServiceImpl userServiceImpl;

    @Autowired
    private AccountActivationLoginFilter accountActivationLoginFilter;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .authorizeRequests()
                .antMatchers("/", "/registration", "/static/**", "/img/**", "/activate/*",
```

```
"/user/profile").permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login")
                .failureUrl("/login?error=true")
                .permitAll()
                .and()
                .logout()
                .permitAll()
                .and()
                .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.IF_REQUIRED)
                .and()
                .addFilterBefore(accountActivationLoginFilter,
UsernamePasswordAuthenticationFilter.class);
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userServiceImpl)
                .passwordEncoder(getPasswordEncoder());
    }
}
```

Клас MvcConfig.java:

```
package com.example.learnNook.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Value("${upload.path}")
    private String uploadPath;

    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/login").setViewName("login");
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/static/**")
                .addResourceLocations("classpath:/static/");
        registry.addResourceHandler("/img/**")
                .addResourceLocations("file:" + uploadPath + "/");
    }
```

```
}
```

Клас AccountActivationLoginFilter.java:

```
package com.example.learnNook.config;

import com.example.learnNook.domain.User;
import com.example.learnNook.exception.RedirectWithModelException;
import com.example.learnNook.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.GenericFilterBean;
import org.springframework.web.servlet.FlashMap;
import org.springframework.web.servlet.support.RequestContextUtils;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class AccountActivationLoginFilter extends GenericFilterBean {

    @Autowired
    private UserService userService;

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,
                         FilterChain filterChain) throws IOException, ServletException {

        try {
            HttpServletRequest request = (HttpServletRequest) servletRequest;
            HttpServletResponse response = (HttpServletResponse) servletResponse;

            if (request.getRequestURI().equals("/login") && request.getMethod().equals("POST")) {
                String name = request.getParameter("username");
                User user = userService.findByUsername(name);
                System.out.println("Intercepted /login POST request NAME:" + name);

                if (user != null) {
                    if (!user.isActive()) {
                        System.out.println("User is not active");
                        request.getSession().setAttribute("error", "user-not-activated");
                        response.sendRedirect(request.getContextPath() + "/login");
                        return;
                    }
                }
            }
```

```
            filterChain.doFilter(request, servletResponse);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Клас MainController.java:

```
package com.example.learnNook.controller;

import com.example.learnNook.domain.Message;
import com.example.learnNook.domain.User;
import com.example.learnNook.domain.dto.MessageDto;
import com.example.learnNook.repos.MessageRepos;
import com.example.learnNook.repos.UserRepos;
import com.example.learnNook.service.LikeService;
import com.example.learnNook.service.MessageService;
import com.example.learnNook.service.UserServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageImpl;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import org.springframework.web.util.UriComponents;
import org.springframework.web.util.UriComponentsBuilder;

import java.io.File;
import java.io.IOException;
import java.security.Principal;
import java.util.*;
import java.util.stream.Collectors;

@Controller
public class MainController {
    @Autowired
    private MessageRepos messageRepos;

    @Autowired
    private UserRepos userRepos;

    @Autowired
```

```
    private UserServiceImpl userServiceImpl;

    @Autowired
    private MessageService messageService;

    @Autowired
    private LikeService likeService;

    @Value("${upload.path}")
    private String uploadPath;

    @GetMapping("/")
    public String learnNook(Map<String, Object> model) {

        return "learnNook";
    }

    @GetMapping("/main")
    public String main(

            @RequestParam(required = false) String filter,
            Model model,
            @AuthenticationPrincipal User user,
            @RequestParam(required = false) Message messageToEdit,
            @PageableDefault(sort = { "id" }, direction = Sort.Direction.DESC) Pageable pageable) {
        Page<MessageDto> messages = messageService.messageList(pageable, filter, user);

        List<MessageDto> messagesByUserLikes = messageService.getMessagesByUserLikes(user);
        List<MessageDto> messagesBySimilarText =
messageService.getMessagesBySimilarText1(user);
        List<MessageDto> messagesByUserLikesAndSimilarText =
messageService.getMessagesByUserLikesAndSimilarText(user);

        Set<Long> usedMessageIds = new HashSet<>();

usedMessageIds.addAll(messagesByUserLikesAndSimilarText.stream().map(MessageDto::getId).c
ollect(Collectors.toSet()));

        List<MessageDto> filteredMessagesByUserLikes = messagesByUserLikes.stream()
                .filter(message -> !usedMessageIds.contains(message.getId()))
                .collect(Collectors.toList());


usedMessageIds.addAll(filteredMessagesByUserLikes.stream().map(MessageDto::getId).collect(C
ollectors.toSet()));

        List<MessageDto> filteredMessagesBySimilarText = messagesBySimilarText.stream()
                .filter(message -> !usedMessageIds.contains(message.getId()))
                .collect(Collectors.toList());


usedMessageIds.addAll(filteredMessagesBySimilarText.stream().map(MessageDto::getId).collect(
```

```
Collectors.toSet()));

        List<MessageDto> filteredMessages = messages.getContent().stream()
                .filter(message -> !usedMessageIds.contains(message.getId()))
                .collect(Collectors.toList());

        List<MessageDto> combinedMessages = new ArrayList<>();
        combinedMessages.addAll(messagesByUserLikesAndSimilarText);
        combinedMessages.addAll(filteredMessagesByUserLikes);
        combinedMessages.addAll(filteredMessagesBySimilarText);
        combinedMessages.addAll(filteredMessages);


        model.addAttribute("messagesByUserLikesAndSimilarText",
messagesByUserLikesAndSimilarText);
        model.addAttribute("messagesBySimilarText", messagesBySimilarText);
        model.addAttribute("messagesByUserLikes", messagesByUserLikes);
        model.addAttribute("messageToEdit", messageToEdit);

        model.addAttribute("messages", new PageImpl<>(combinedMessages, pageable,
messages.getTotalElements()));
        model.addAttribute("filter", filter);

        boolean isAdmin = user.isAdmin();
        model.addAttribute("isAdmin", isAdmin);
        model.addAttribute("currentUserId", user.getId());

        User userWithId7 = userRepos.findById(5L).orElse(null);

        if (userWithId7 != null) {
            System.out.println("User with ID 7 found: " + userWithId7.getUsername());
        }
        model.addAttribute("userWithId7", userWithId7);
        return "main";
    }

    @PostMapping("/main")
    public String add(
            @AuthenticationPrincipal User user,
            @RequestParam String text,
            @RequestParam String tag, Map<String, Object> model,
            @RequestParam("file") MultipartFile file
    ) throws IOException {
        Message message = new Message(text, tag, user);

        saveFile(file, message);

        messageRepos.save(message);

        Iterable<Message> messages = messageRepos.findAll();

        model.put("messages", messages);
```

```
    return "main";
}

private void saveFile(MultipartFile file, Message message) throws IOException {
    if (file !=null && !file.getOriginalFilename().isEmpty()) {
        File uploadDir = new File(uploadPath);

        if (!uploadDir.exists()) {
            uploadDir.mkdir();
        }

        String uuidFile = UUID.randomUUID().toString();
        String resultFilename = uuidFile + "." + file.getOriginalFilename();

        file.transferTo(new File(uploadPath + "/" + resultFilename));
        message.setFilename(resultFilename);
    }
}

@GetMapping("/user-messages/{user}")
public String userMessges(
        @AuthenticationPrincipal User currentUser,
        @PathVariable User user,
        @AuthenticationPrincipal User userCurrent,
        Model model,
        @RequestParam(required = false) Message message,
        @RequestParam(required = false) Long messageId
) {
    Set<Message> messages = user.getMessages();
    User userWithId7 = userRepos.findById(5L).orElse(null);

    if (messageId != null) {
        Message messageToEdit = messageRepos.findById(messageId).orElse(null);
        model.addAttribute("messageToEdit", messageToEdit);
    }

    model.addAttribute("userChannel", user);
    model.addAttribute("subscriptionsCount", user.getSubscriptions().size());
    model.addAttribute("subscribersCount", user.getSubscribers().size());
    model.addAttribute("isSubscriber", user.getSubscribers().contains(currentUser));
    model.addAttribute("messages", messages);
    model.addAttribute("messageToEdit1", message);
    model.addAttribute("isCurrentUser", currentUser.equals(user));
    User author = userRepos.findById(user.getId()).orElseThrow();
    model.addAttribute("author", author);
    model.addAttribute("subscribers", user.getSubscribers());
    model.addAttribute("hasSubscribers", !user.getSubscribers().isEmpty());
    model.addAttribute("subscriptions", user.getSubscriptions());

    if (userWithId7 != null) {
        System.out.println("User with ID 7 found: " + userWithId7.getUsername());
```

```
        }

        model.addAttribute("userWithId7", userWithId7);

        return "userMessages";
    }

    @PostMapping("/user-messages/{user}")
    public String updateMessage(
            @AuthenticationPrincipal User currentUser,
            @PathVariable Long user,
            @RequestParam("id") Long messageId,
            @RequestParam("text") String text,
            @RequestParam("tag") String tag,
            @RequestParam("file") MultipartFile file
    ) throws IOException {

        Message message = messageRepos.findById(messageId).orElseThrow(() -> new
IllegalArgumentException("Invalid message Id:" + messageId));

        if (message.getAuthor().equals(currentUser)) {
            if (!StringUtils.isEmpty(text)) {
                message.setText(text);
            }

            if (!StringUtils.isEmpty(tag)) {
                message.setTag(tag);
            }

            saveFile(file, message);

            messageRepos.save(message);
        }

        return "redirect:/user-messages/" + user;
    }

    @GetMapping("/messages/{messages}/like")
    public String like(
            @AuthenticationPrincipal User currentUser,
            @PathVariable Message messages,
            RedirectAttributes redirectAttributes,
            @RequestHeader(required = false) String referer
    ) {
        Set<User> likes = messages.getLikes();

        if (likes.contains(currentUser)) {
            likes.remove(currentUser);
        } else {
            likes.add(currentUser);
        }
        UriComponents components = UriComponentsBuilder.fromHttpUrl(referer).build();
```

```
        components.getQueryParams()
                .entrySet()
                .forEach(pair -> redirectAttributes.addAttribute(pair.getKey(), pair.getValue()));

        messageRepos.save(messages);

        return "redirect:" + components.getPath();
    }
}
```

Клас UserController.java:

```
package com.example.learnNook.controller;
import com.example.learnNook.domain.Role;
import com.example.learnNook.domain.User;
import com.example.learnNook.repos.UserRepos;
import com.example.learnNook.service.UserServiceImpl;
import com.example.learnNook.service.UserServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.security.Principal;
import java.util.Map;

@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    private UserRepos userRepos;

    @Autowired
    private UserServiceImpl userServiceImpl;

    @PreAuthorize("hasAnyAuthority('ADMIN')")
    @GetMapping
    public String userList(Model model) {
        model.addAttribute("users", userServiceImpl.findAll());
        return "userList";
    }

    @PreAuthorize("hasAnyAuthority('ADMIN')")
    @GetMapping("{user}")
    public String userEditForm(@PathVariable User user, Model model) {
        model.addAttribute("user", user);
        model.addAttribute("roles", Role.values());
        return "userEdit";
    }
```

```
@PreAuthorize("hasAnyAuthority('ADMIN')")
@PostMapping
public String userSave(
        @RequestParam String username,
        @RequestParam Map<String, String> form,
        @RequestParam("userId") User user) {
    userServiceImpl.saveUser(user, username, form);
    return "redirect:/user";
}

@GetMapping("profile")
public String getProfile(Model model, @AuthenticationPrincipal User user) {
    model.addAttribute("username", user.getUsername());
    model.addAttribute("email", user.getEmail());

    return "profile";
}

@PostMapping("profile")
public String updateProfile(
        @AuthenticationPrincipal User user,
        @RequestParam String username,
        @RequestParam String password,
        @RequestParam String email
) {
    userServiceImpl.updateProfile(user, username, password, email);
    return "redirect:/user/profile";
}

@PostMapping("unsubscribe/{userId}/{subscriptionUserId}")
public String unsubscribe(
        Model model,

        @PathVariable Long userId,
        @PathVariable Long subscriptionUserId,
        @AuthenticationPrincipal User currentUser
) {
    System.out.println(userId + "=" + subscriptionUserId);
    userServiceImpl.unsubscribeId(subscriptionUserId, userId);
    return "redirect:/user-messages/" + currentUser.getId();
}

@PostMapping("subscribe/{id}")
public String subscribe(
        @PathVariable User id,
        Model model,
        @AuthenticationPrincipal User currentUser) {
    userServiceImpl.subscribe(currentUser, id);
    return "redirect:/user-messages/" + id.getId();
}
```

```
    @PostMapping("unsubscribe/{id}")
    public String unsubscribe(@PathVariable User id,
                              @AuthenticationPrincipal User currentUser,
                              Model model) {
        userServiceImpl.unsubscribe(currentUser, id);
        return "redirect:/user-messages/" + id.getId();
    }

    @GetMapping("{type}/{user}/list")
    public String userList(
            Model model,
            @PathVariable User user,
            @PathVariable String type
    ) {
        model.addAttribute("userChannel", user);
        model.addAttribute("type", type);

        if ("subscriptions".equals(type)) {
            model.addAttribute("users", user.getSubscriptions());
        } else {
            model.addAttribute("users", user.getSubscribers());
        }

        return "subscriptions";
    }
}
```

Клас RegistrationController.java:

```
package com.example.learnNook.controller;
import com.example.learnNook.domain.User;
import com.example.learnNook.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.Map;

@Controller
public class RegistrationController {

    @Autowired
    private UserService userService;
```

```
    @GetMapping("/registration")
    public String registration() {
        return "registration";
    }

    @PostMapping("/registration")
    @ResponseBody
    public Map<String, String> addUser(User user) {
        User userFromDb = userService.getUser(user);
        Map<String, String> response = new HashMap<>();

        if (user.getUsername() == null || user.getUsername().isEmpty() ||
                user.getEmail() == null || user.getEmail().isEmpty() ||
                user.getPassword() == null || user.getPassword().isEmpty()) {
            response.put("message", "Не всі поля заповненні!");
            return response;
        }

        if (userFromDb != null) {
            response.put("message", "Користувач з таким нікнеймом вже існує!");
            return response;
        }

        User userByEmail = userService.getUserByEmail(user.getEmail());
        if (userByEmail != null) {
            response.put("message", "Email зайнятий!");
            return response;
        }

        userService.registerUser(user);
        response.put("redirect", "/login");
        return response;
    }

    @GetMapping("/activate/{code}")
    public String activate(Model model, @PathVariable String code) {
        boolean isActivated = userService.activateUser(code);

        if (isActivated) {
            model.addAttribute("message", "User successfully activated");
        } else {
            model.addAttribute("message", "Activation code is not found!");
        }
        return "login";
    }
}
```

Клас AnalyticsController.java:

```
package com.example.learnNook.controller;
```

```
import com.example.learnNook.domain.Message;
import com.example.learnNook.domain.Role;
import com.example.learnNook.domain.User;
import com.example.learnNook.repos.MessageRepos;
import com.example.learnNook.repos.UserRepos;
import com.example.learnNook.service.AnalyticsService;
import com.example.learnNook.service.UserServiceImpl;
import com.fasterxml.jackson.core.JsonProcessingException;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.fasterxml.jackson.databind.ObjectMapper;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

@Controller
public class AnalyticsController {

    @Autowired
    private MessageRepos messageRepos;

    @Autowired
    private UserRepos userRepos;

    @Autowired
    private AnalyticsService analyticsService;

    @GetMapping("/analytics")
    public String analytics(Model model) {

        List<Object[]> tagData = messageRepos.countTags();
        List<String> tags = new ArrayList<>();
        List<Long> counts = new ArrayList<>();

        for (Object[] data : tagData) {
            tags.add((String) data[0]);
            counts.add((Long) data[1]);
        }
        System.out.println(tags);
        System.out.println(counts);
        ObjectMapper mapper = new ObjectMapper();
        try {
            String tagsJson = mapper.writeValueAsString(tags);
            String countsJson = mapper.writeValueAsString(counts);

            System.out.println(tagsJson);
```

```
            System.out.println(countsJson);
            model.addAttribute("tagsJson", tagsJson);
            model.addAttribute("countsJson", countsJson);
        } catch (Exception e) {
            e.printStackTrace();
        }

        long countUser = userRepos.count();
        long countAdmin = userRepos.countByRole(Role.ADMIN);
        long countPost = messageRepos.count();

        model.addAttribute("countUser", countUser);
        model.addAttribute("countAdmin", countAdmin);
        model.addAttribute("countPost", countPost);

        List<AnalyticsService.UserPostCount> userPostCounts =
analyticsService.getUserPostCounts();
        try {
            String userPostCountsJson = mapper.writeValueAsString(userPostCounts);
            model.addAttribute("userPostCountsJson", userPostCountsJson);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return "analytics";
    }
}
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МІКРОБЛОГІНГОВОЇ СОЦІАЛЬНОЇ МЕРЕЖІ ДЛЯ ПІДТРИМКИ ОСВІТНІХ ІНІЦІАТИВ

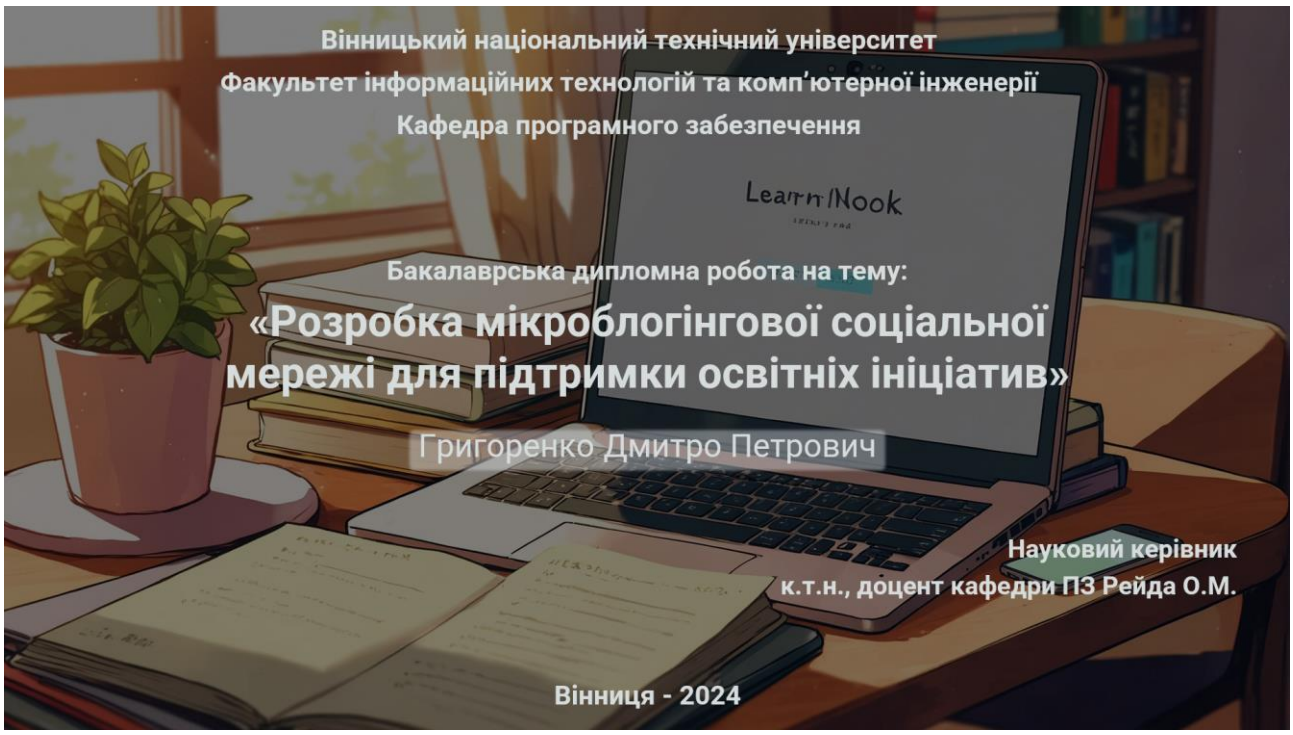

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність розробки

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

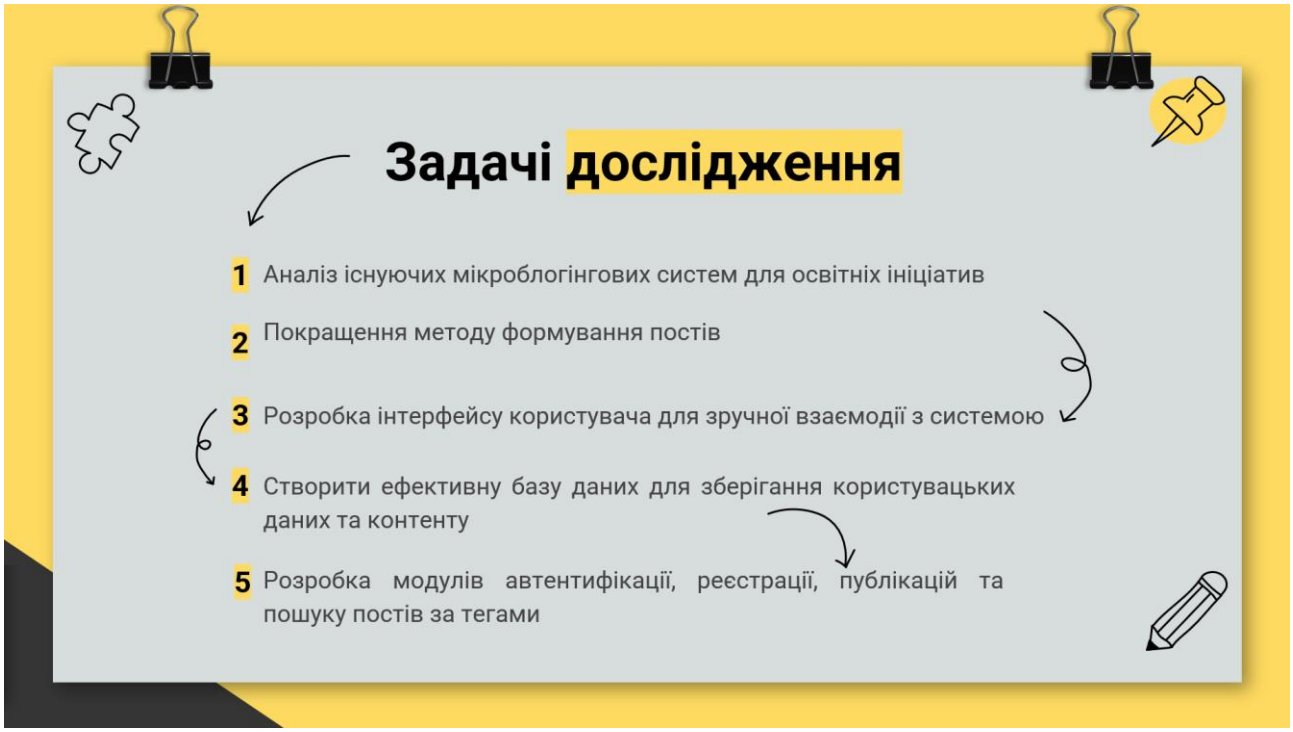

Рисунок Г.4 – Задачі дослідження

Рисунок Г.5 – Наукова новизна

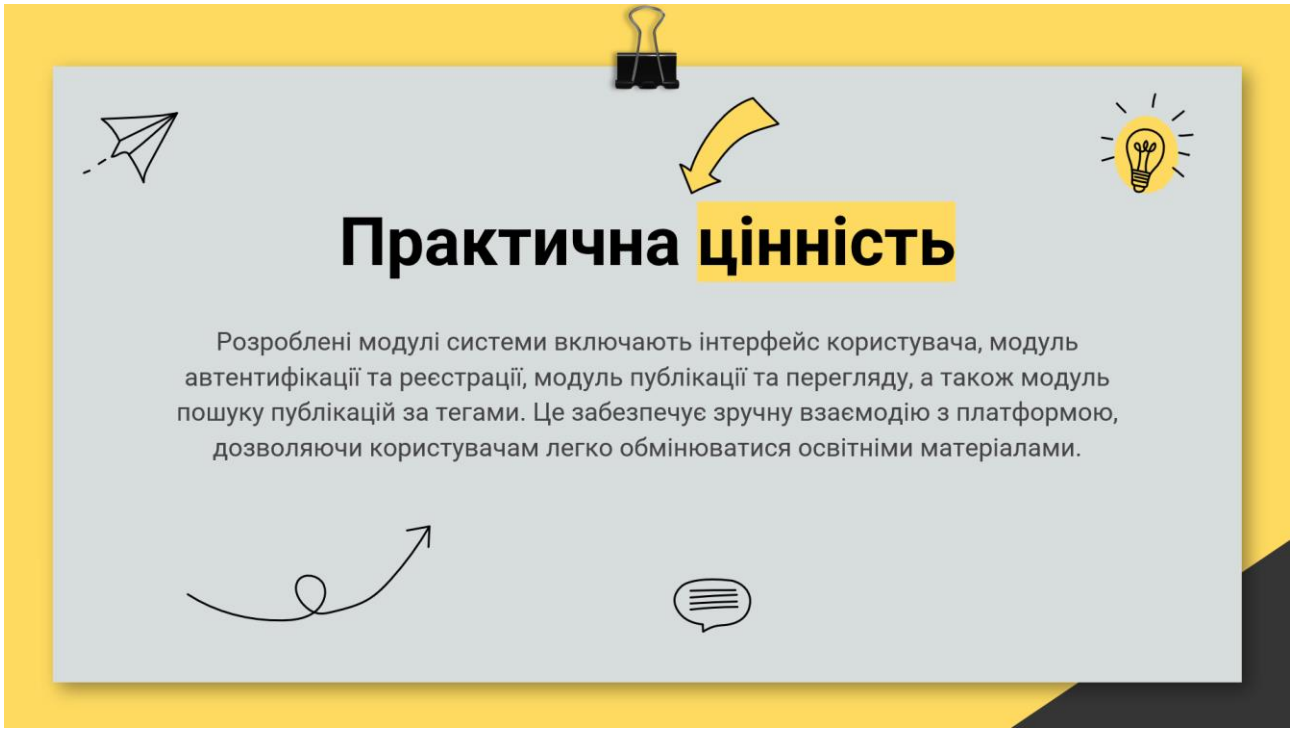

Рисунок Г.6 – Практична цінність

Порівняння з аналогами

Характеристика	Edmodo	Schoology	LearnNook
Можливість управління повідомленнями	1	1	1
Інтелектуальні рекомендації на основі алгоритмів уподобань	0	0	1
Можливість оцінювання постів	1	0	1
Аналіз активності користувачів та візуалізація даних	0	1	1
Можливість розширеного пошуку публікацій за тегами	0	0	1
Сумарний коефіцієнт	2	2	5

Рисунок Г.7 – Порівняння з аналогами

Розробка структури інтерфейсу

Головна сторінка інтерфейсу мікроблогінгової мережі створена для забезпечення інтуїтивно зрозумілої та ефективної навігації

Рисунок Г.8 – Розробка структури інтерфейсу (частина 1)

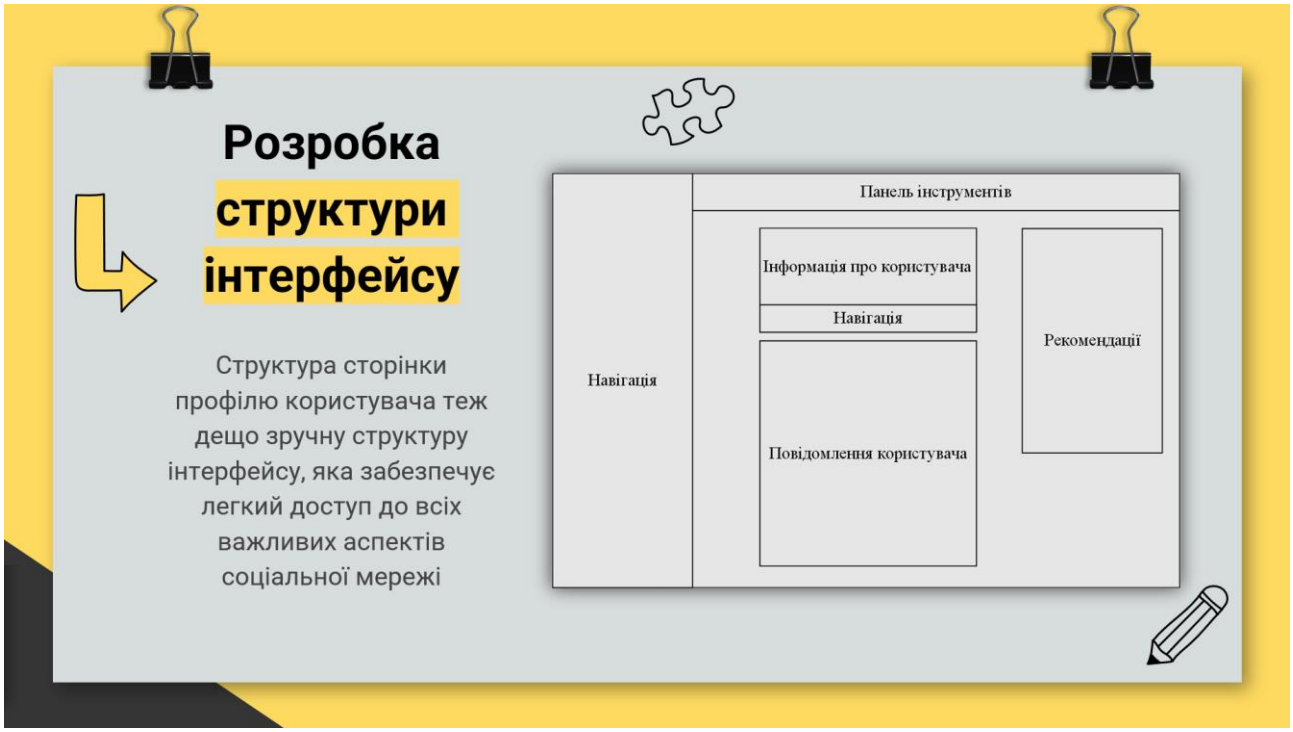

Рисунок Г.9 – Розробка структури інтерфейсу (частина 2)

Рисунок Г.10 – Модель роботи мікроблогінгової вебсистеми

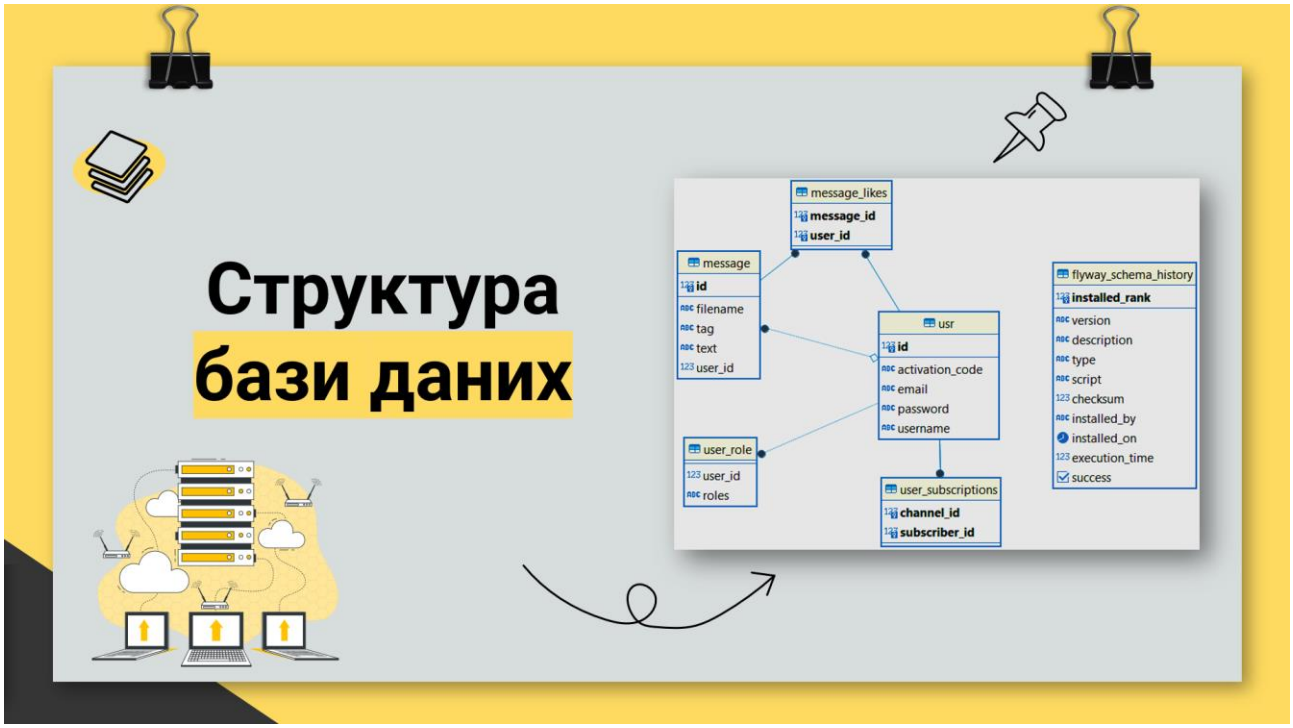

Рисунок Г.11 – Структура бази даних

Рисунок Г.12 – Використані технології

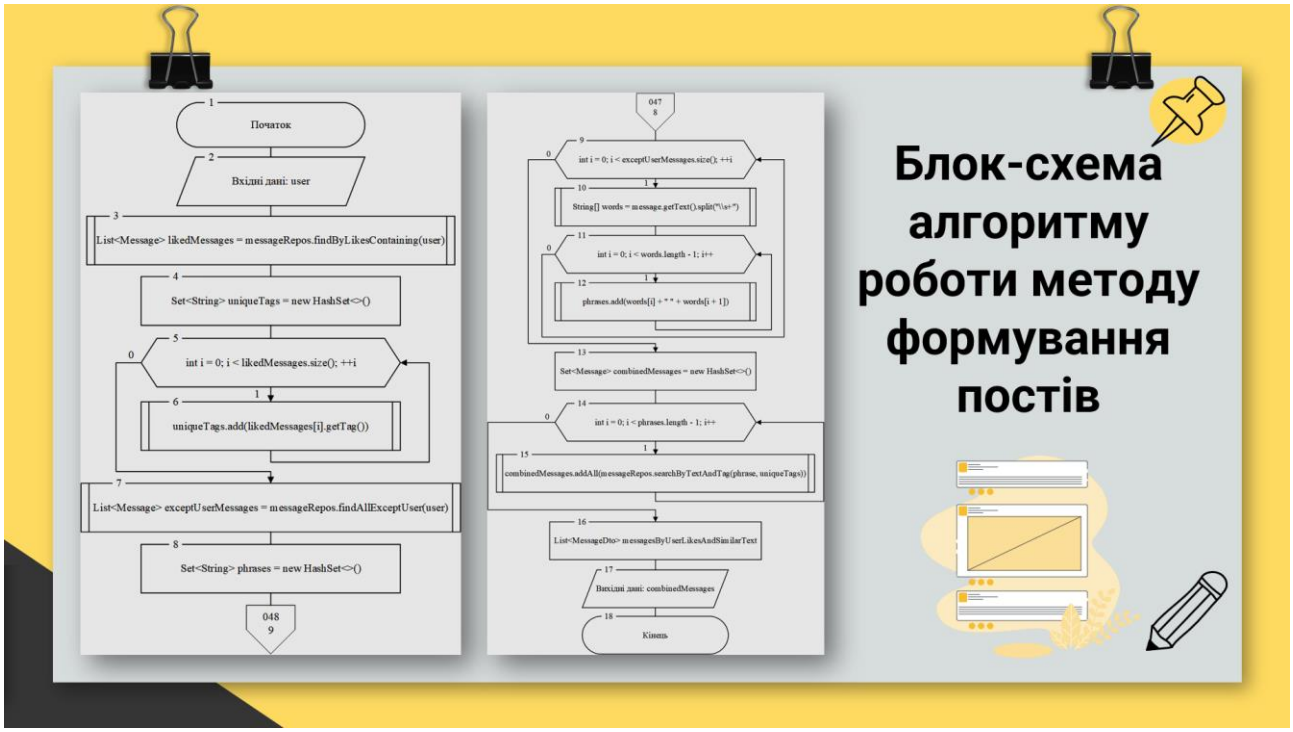

Рисунок Г.13 – Блок-схема алгоритму роботи методу формування постів

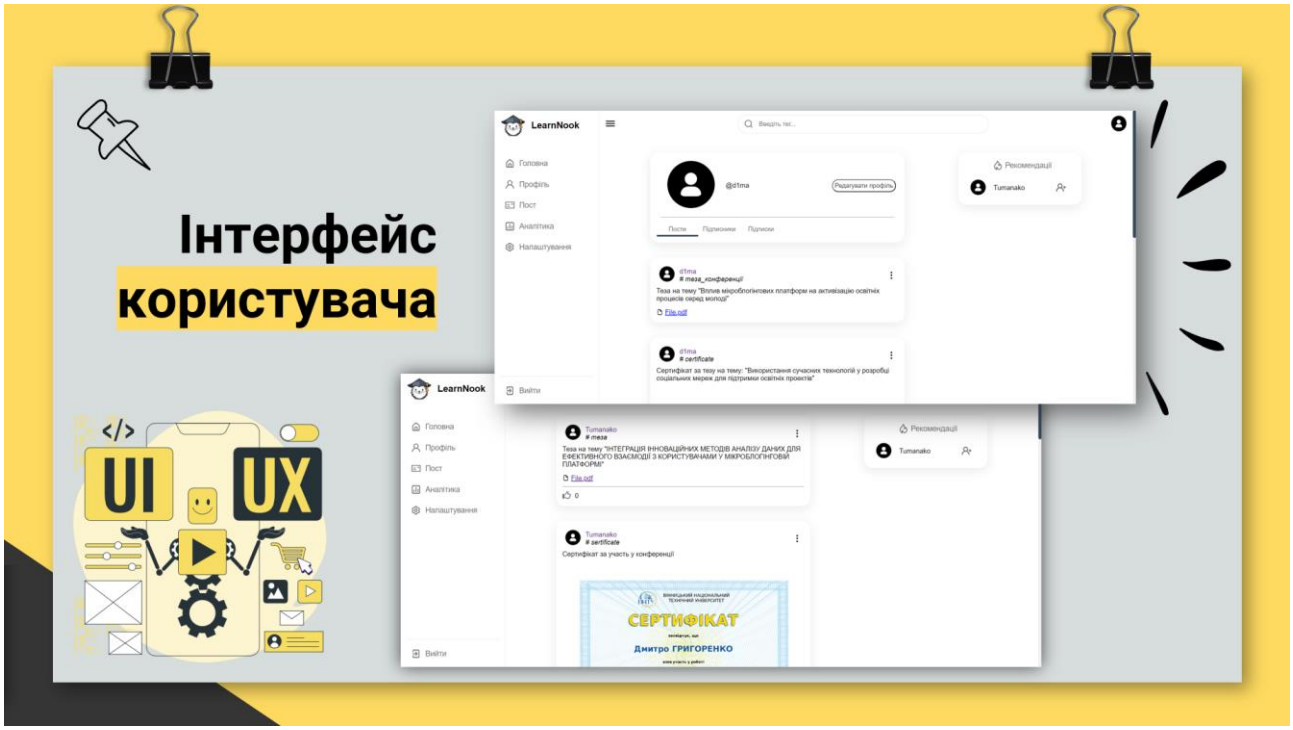

Рисунок Г.14 – Інтерфейс користувача

Рисунок Г.15 – Інтерфейс сторінки аналітики

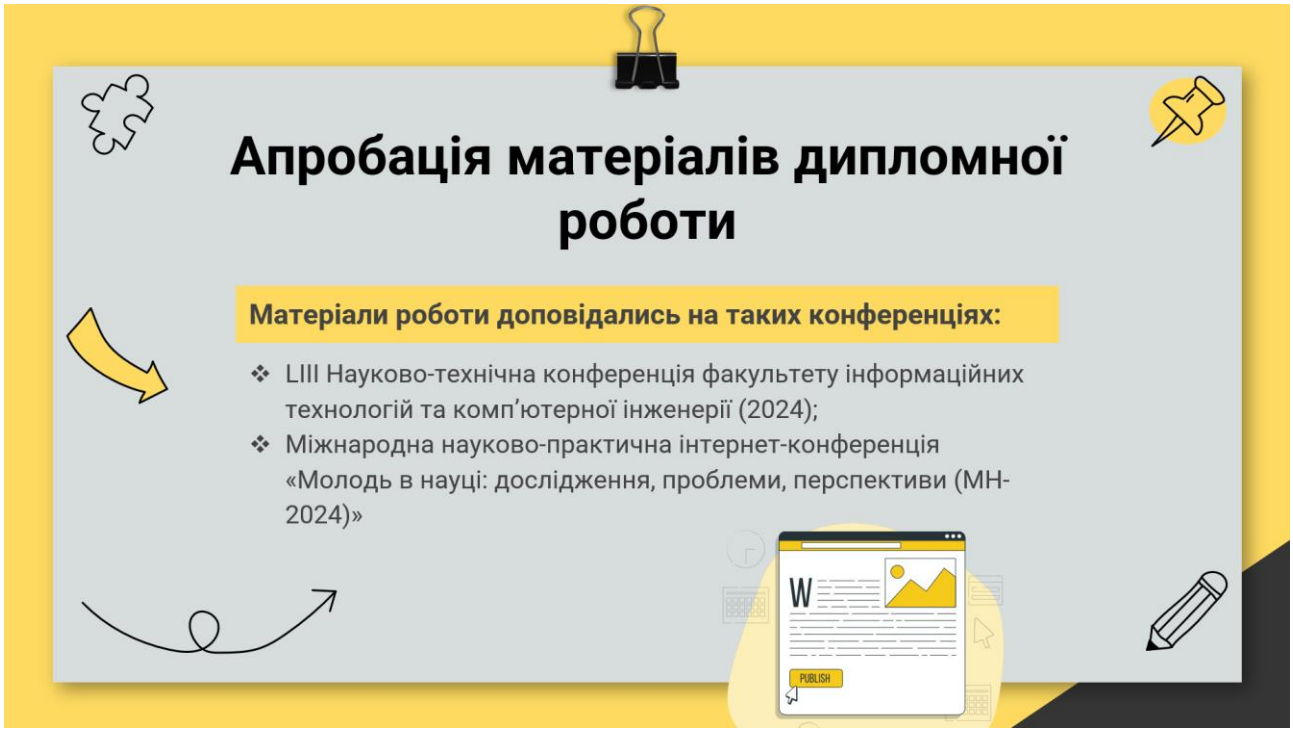

Рисунок Г.16 – Апробація матеріалів дипломної роботи

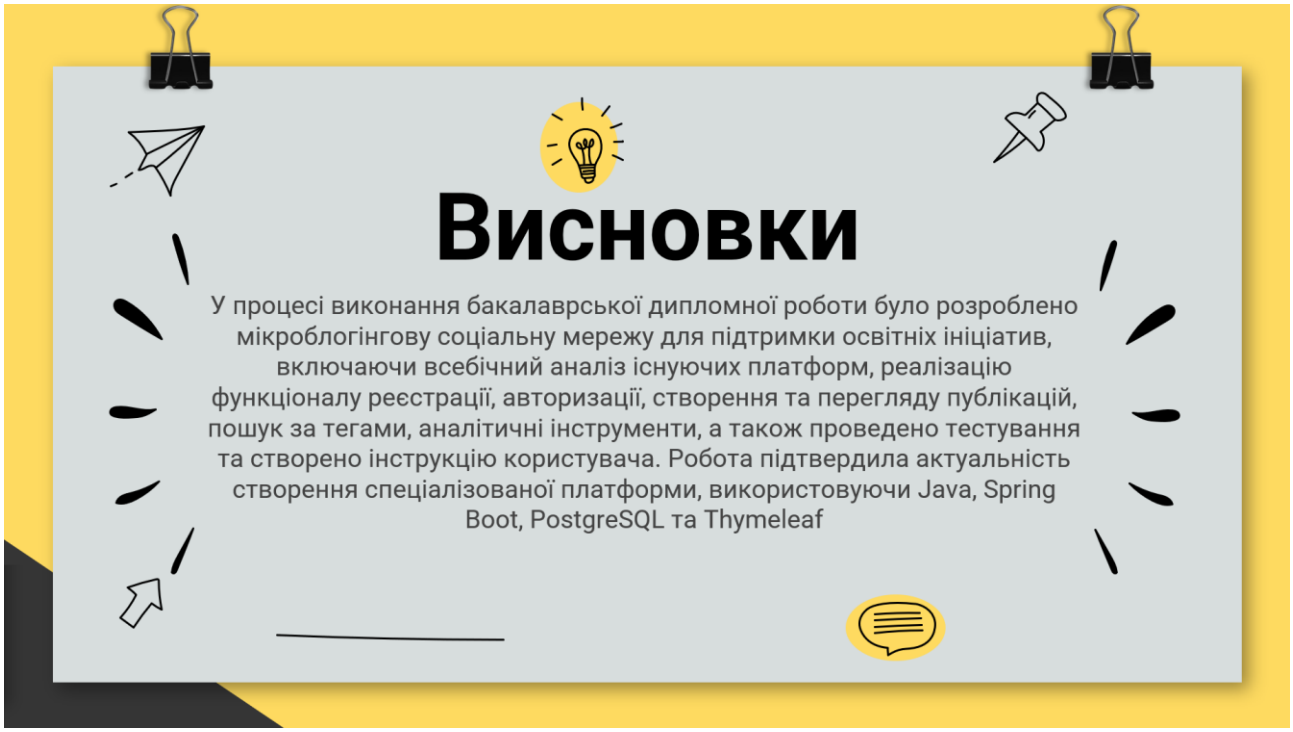

Рисунок Г.17 – Висновки