

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Розробка онлайн-форуму»

Виконав студент групи ІПЗк-23-1
НЕГЕРЕЙ Владислав Васильович

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **НЕГЕРЕЙ Владислав Васильович**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: **«Розробка онлайн-форуму»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **створення та управління цифровою спільнотою в рамках класичної монолітної архітектури, яка використовує фреймворк Spring Boot для побудови надійного та функціонального серверного додатку.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Морозов А.В.	27.02.2025 р.	27.02.2025 р.
2	Морозов А.В.	27.02.2025 р.	27.02.2025 р.
3	Морозов А.В.	27.02.2025 р.	27.02.2025 р.

РЕФЕРАТ

Кваліфікаційна робота складається з програмного продукту та пояснювальної записки. Пояснювальна записка містить опис розробленого онлайн-форуму.

Текстова частина викладена на 71 сторінках друкованого тексту. Пояснювальна записка містить 6 сторінок додатків. Список використаних джерел містить 31 найменування та займає дві сторінки. У роботі наведено 56 рисунків та 12 таблиць. Загальний обсяг роботи – 77 сторінок.

Метою роботи є створення вебдодатку у вигляді онлайн-форуму, що забезпечує зручну платформу для спілкування користувачів, обміну повідомленнями та обговорення різних тем. Мета включає розробку зручного інтерфейсу користувача, реалізацію системи реєстрації та авторизації, створення функціоналу для публікації та коментування тем, а також забезпечення безпеки та стабільної роботи додатку шляхом тестування.

У роботі визначено основні завдання зі створення онлайн-форуму, обґрунтовано вибір інструментів та технологій для веброзробки, описано архітектуру системи та реалізацію ключових модулів, зокрема взаємодію з базою даних, механізмів сортування тем, керування ролями користувачів. Проведено тестування працездатності форуму з метою перевірки його функціональності та надійності.

Ключові слова: ВЕБДОДАТОК, ОНЛАЙН-ФОРУМ, JAVA, HTML, CSS, JAVASCRIPT, MYSQL, REACT, SPRING BOOT, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ВЕБ-РОЗРОБКА.

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Онлайн-форум	Літ.	Арк.	Акрушів
Розроб.		В.В. Негерей						
Перевір.		А.В. Морозов					3	77
Рецензент		Ю.О. Годлевський				Житомирська політехніка, група ІПЗк-23-1		
Зав. каф.		Т.А. Вакалюк						

ABSTRACT

The qualification project consists of a software product and an explanatory note. The explanatory note contains a description of the developed online forum. The textual part is presented on 71 pages of printed text. The explanatory note includes 6 pages of appendices. The list of references includes 31 items and occupies two pages. The work contains 56 figures and 12 tables. The total volume of the project is 77 pages.

The goal of this project is to develop a web application in the form of an online forum that provides a convenient platform for user communication, message exchange, and discussion of various topics. The objective includes the development of a user-friendly interface, implementation of a registration and authentication system, creation of functionality for posting and commenting on topics, as well as ensuring the security and stable operation of the application through testing.

The work outlines the main tasks for creating the online forum, justifies the choice of tools and technologies for web development, describes the system architecture and implementation of key modules, including interaction with the database, topic sorting mechanisms, and user role management. The application was tested to verify its functionality and reliability.

Keywords: WEB APPLICATION, ONLINE FORUM, JAVA, HTML, CSS, JAVASCRIPT, MYSQL, REACT, SPRING BOOT, SOFTWARE, WEB DEVELOPMENT.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

Зміст

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	9
1.1. Постановка задачі.....	9
1.2. Аналіз існуючого програмного забезпечення.....	10
1.3. Обґрунтування вибору інструментальних засобів	14
Висновки до першого розділу	15
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	16
2.1. Технічне завдання на розробку системи	16
2.2. Аналіз вимог до програмного продукту	18
2.3. Алгоритм роботи та стани програмної системи	20
2.4. Об'єктно-орієнтована модель системи	23
2.5. Взаємодія об'єктів системи.....	25
2.6. Розробка бази даних системи.....	27
Висновок до другого розділу	33
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	34
3.1. Порядок встановлення та налаштування системи.....	34
3.2. Розробка основних функцій вебдодатку.....	35
3.3. Опис інструкції користувача.....	45
3.4. Тестування роботи програмного продукту	63
Висновки до третього розділу.....	67
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТКИ.....	71

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – програмне забезпечення;

HTML – HyperText Markup Language, мова розмітки для створення вебсторінок;

CSS – Cascading Style Sheets, мова стилів для оформлення вебсторінок;

JS – JavaScript, мова програмування для створення інтерактивних елементів на стороні клієнта;

React – JavaScript-бібліотека для створення користувацьких інтерфейсів;

Spring Boot – фреймворк Java для спрощеної розробки серверної частини застосунків;

REST API – Representational State Transfer Application Programming Interface, інтерфейс для обміну даними між клієнтом і сервером;

JWT – JSON Web Token, токен для безпечної передачі інформації про користувача та аутентифікації;

БД – база даних, організована сукупність даних для зберігання та обробки;

SQL – Structured Query Language, мова запитів до реляційних баз даних.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми.

У сучасному цифровому середовищі онлайн-форуми залишаються важливою формою комунікації, яка дозволяє користувачам обмінюватися знаннями, досвідом та ідеями в зручному та структурованому форматі. Незважаючи на розвиток соціальних мереж, форуми зберігають свою актуальність завдяки тематичному спрямуванню, високому рівню модерації, можливості створення довготривалих дискусій та зручному пошуку інформації.

Онлайн-форуми активно використовуються в освітніх, технічних, професійних і хобі-спільнотах. Вони є джерелом вирішення проблем, самоосвіти, професійної підтримки та розвитку спільнот за інтересами. Актуальність теми підсилюється зростаючою потребою у децентралізованих, доступних і функціонально гнучких платформах, що дозволяють користувачам створювати, модерувати й підтримувати власні спільноти.

Розробка сучасного онлайн-форуму потребує врахування вимог до безпеки даних, ефективного адміністрування, зручного інтерфейсу користувача та масштабованості. Це завдання є актуальним у контексті розвитку вебтехнологій і необхідності створення стабільних платформ, здатних обробляти велику кількість контенту та користувацьких дій [11].

Метою кваліфікаційної роботи є дослідження методів і принципів побудови сучасних вебфорумів, аналіз архітектури подібних систем і створення власного вебдодатку з інтуїтивним інтерфейсом і розширеним функціоналом для користувачів і адміністраторів.

Для досягнення цієї мети необхідно виконати наступні завдання:

1. Здійснити огляд та аналіз існуючих онлайн-форумів;
2. Визначити вимоги до функціоналу та інтерфейсу;
3. Обрати відповідні технології для реалізації проєкту;
4. Спроекувати базу даних для зберігання повідомлень, тем, користувачів та прав доступу;
5. Побудувати UML-діаграми для відображення логіки системи;

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

6. Реалізувати клієнтську та серверну частину додатку, включаючи адміністративну панель;

7. Провести тестування вебдодатку для перевірки його надійності та зручності використання.

Об'єктом дослідження є процес проєктування та розробки онлайн-форуму, який включає в себе аналіз потреб цільової аудиторії, створення технічного завдання, розробку архітектури системи, програмну реалізацію та тестування.

Предметом дослідження є використання мов програмування HTML, CSS, JavaScript, Java (Spring Boot) та бібліотеки React для розробки вебдодатку, дослідження їхньої взаємодії, інтеграційних можливостей, забезпечення безпеки та створення зручного користувацького досвіду.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Постановка задачі

Онлайн-форум — це вебдодаток, що надає користувачам можливість створювати теми для обговорення, залишати повідомлення, брати участь у дискусіях та взаємодіяти в рамках спільноти за спільними інтересами. Такий ресурс функціонує через веббраузер і зберігається на віддаленому сервері, що забезпечує його доступність з будь-якого пристрою, підключеного до Інтернету.

Використання форумів сприяє налагодженню комунікації між користувачами, дозволяє вирішувати проблеми спільно, ділитися знаннями, ідеями та досвідом. Крім того, форум може виступати як інструмент зворотного зв'язку, технічної підтримки або джерело корисного контенту, створеного самими користувачами.

Вебфорум не потребує встановлення програмного забезпечення на клієнтські пристрої, оскільки вся логіка реалізована на серверній стороні. Це спрощує підтримку та оновлення — зміни вносяться централізовано на сервері й одразу стають доступними всім користувачам.

Для повноцінної роботи форуму необхідно реалізувати взаємодію між вебсервером, що обробляє запити користувачів, і базою даних, у якій зберігається інформація про користувачів, теми, повідомлення, категорії та ролі доступу. База даних повинна забезпечувати швидкий і надійний доступ до великої кількості структурованих даних з можливістю масштабування [21].

Метою розробки є створення сучасного вебдодатку у вигляді форуму, який дозволить користувачам спілкуватися в режимі реального часу, створювати нові теми та повідомлення, брати участь у дискусіях, а адміністраторам — ефективно керувати контентом і користувачами через окрему панель керування.

Основні вимоги до вебдодатку:

1. Сучасний дизайн інтерфейсу, виконаний у приємній світлій кольоровій гамі;

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Інтуїтивно зрозумілий та логічно структурований інтерфейс користувача;
3. Висока швидкодія та оптимізований час завантаження сторінок;
4. Адаптивність інтерфейсу для мобільних пристроїв, планшетів і комп'ютерів;
5. Реалізація основного функціоналу форуму: створення тем, повідомлень, редагування та коментування;
6. Можливість реєстрації, авторизації та керування профілем користувача.

Реалізація вебфоруму сприятиме створенню ефективної комунікаційної платформи з широким функціоналом, зручною навігацією та сучасними підходами до веброзробки.

1.2. Аналіз існуючого програмного забезпечення

Сьогодні в Інтернеті існує велика кількість популярних онлайн-форумів, які охоплюють різноманітні тематичні напрямки — від технологій та ігор до загальних дискусій і наукових обговорень. Аналіз найпопулярніших форумів дозволяє виявити загальні тренди, корисні функції та особливості реалізації, які можуть бути використані при створенні власного вебдодатку.

Одним із таких прикладів є NeoGAF — популярний форум, присвячений відеоіграм, новинам індустрії та загальним темам. Він має чітку структуру з поділом на тематичні підфоруми, як-от ігрові новини, платформи та загальні обговорення, де кожен користувач може створювати нові теми або брати участь у дискусіях. До ключових особливостей платформи належать інтуїтивний інтерфейс з адаптивним дизайном, підтримка лайків, цитування та звітів про порушення, а також система рангу користувачів за активність. NeoGAF вирізняється активною спільнотою геймерів та строгим модераторським контролем, що забезпечує високу якість обговорень. Такий підхід дозволяє зберігати високий рівень дискусій та запобігати поширенню токсичної поведінки. Саме ці аспекти роблять його цікавим об'єктом для аналізу при розробці схожих проєктів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

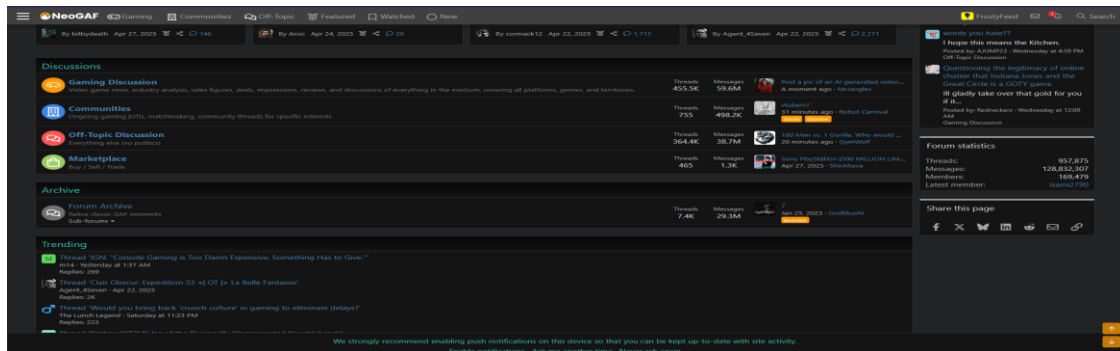


Рис. 1.1 – Головна сторінка сайту NeoGAF

Reddit — це одна з найбільших онлайн-платформ для обговорень, що поєднує в собі елементи форуму та соціальної мережі. Вона складається з безлічі окремих спільнот, відомих як "subreddit", кожна з яких присвячена певній темі — від IT і науки до гумору та порад. До ключових особливостей платформи належить система голосування за пости й коментарі (upvote/downvote), розгалужена структура обговорень та можливість створювати дописи різного типу: текстові, з зображеннями, посиланнями чи опитуваннями. Активність користувачів відображається через рейтинг, що називається "карма". Reddit є чудовим прикладом динамічного, саморегульованого форуму з величезною кількістю тематичних дискусій.



Рис. 1.2 – Головна сторінка сайту Reddit

Quora — це платформа запитань і відповідей, яка функціонує як форум для обговорення різних тем, від точних наук і бізнесу до особистого розвитку. Хоча технічно це не класичний форум, його функціональність та спільнотні дискусії дуже подібні до тематичних форумів. Ключові особливості включають можливість ставити запитання й отримувати відповіді від експертів, систему

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

підписки на теми та користувачів, рейтинг найкращих відповідей, а також обговорення в коментарях під ними. Quora вирізняється високою якістю контенту, оскільки відповіді часто дають фахівці з відповідної сфери.

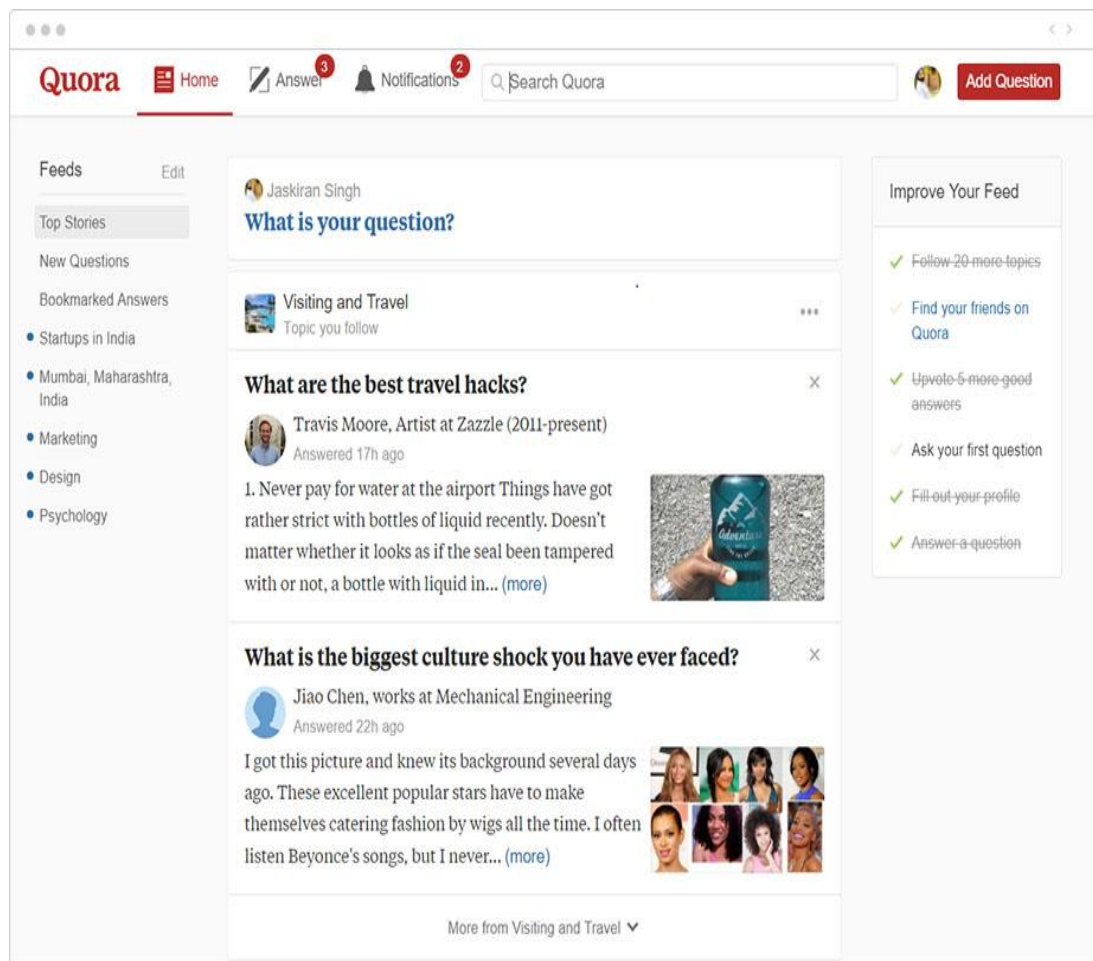


Рис. 1.3. – Головна сторінка сайту Quora

Stack Overflow — це професійний форум для розробників, де користувачі ставлять технічні питання та отримують відповіді, пов'язані з програмуванням. Його ключові особливості включають систему голосування за питання й відповіді, репутаційну систему для користувачів, модерацію з боку самої спільноти та підтримку тегів для зручної фільтрації за темами. Завдяки цьому Stack Overflow є зразком ефективної організації знань у форматі питань і відповідей. Строгі правила щодо формату контенту допомагають підтримувати високий рівень корисності, запобігаючи перетворенню дискусій на суб'єктивні обговорення. Таким чином, платформа слугує не лише для вирішення поточних завдань, а і як надійна, постійно оновлювана база знань для всієї ІТ-спільноти.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

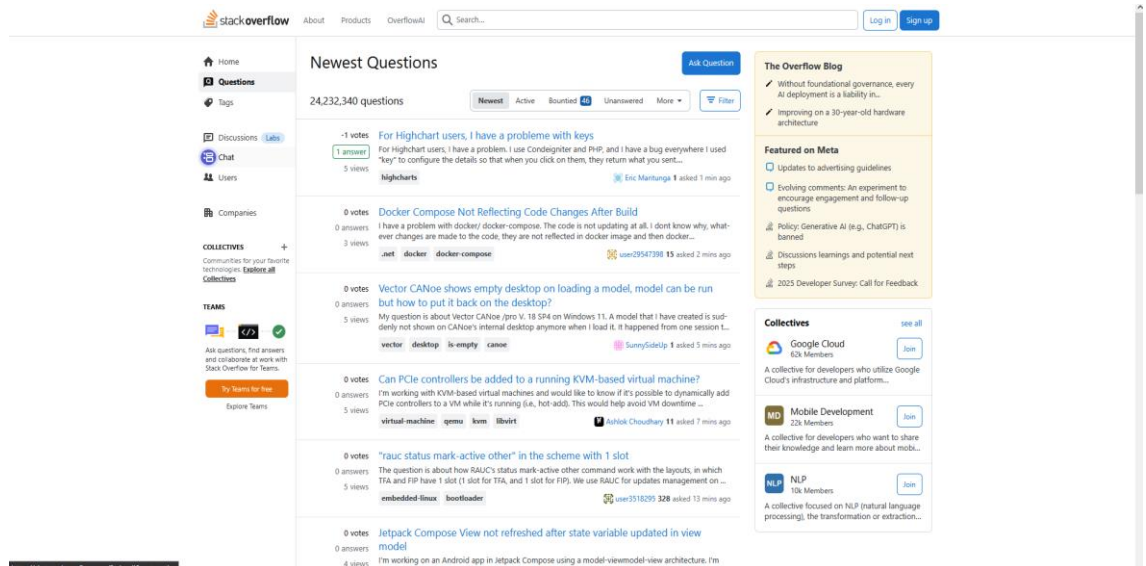


Рис. 1.4. – Головна сторінка сайту Stack Overflow

Таблиця 1.1

Порівняння існуючих аналогів

Критерії	NeoGAF	Reddit	Quora	Stack Overflow
Основна тематика	Відеоігри, медіа	Універсальна (за темами)	Запитання-відповіді	Програмування
Система голосування	Так (лайки)	Так (up/downvotes)	Так	Так
Коментарі до постів	Так	Так	Так	Так
Система рейтингів	Ранг користувача	Карма	Авторитет	Репутація
Адаптивність	Так	Так	Так	Так
Додаткові функції	Приватні повідомлення	Меми, зображення, АМА	Топіки, підписки	Код-сніпети, теги, модерація

На основі аналізу можна зробити висновок, що сучасні форуми поєднують елементи соціальних мереж (рейтинги, підписки, карма) та класичних дискусійних систем. У власному вебфорумі доцільно реалізувати:

- інтуїтивний адаптивний інтерфейс;
- підтримку голосувань та системи рейтингу;
- модерацію контенту;
- категоризацію та фільтрацію тем;
- багаторівневу систему коментарів.

1.3. Обґрунтування вибору інструментальних засобів

Для розробки клієнтської частини вебдодатку для вивчення іноземних мов було обрано сучасні технології, зокрема HTML, CSS, JavaScript та React.js. HTML є основною мовою розмітки, що визначає структуру вебсторінок, дозволяючи створювати заголовки, абзаци, зображення та інші елементи. CSS використовується для визначення візуального оформлення сайту, таких як кольори, шрифти та макет. JavaScript дозволяє реалізувати інтерактивність та динамічність сторінок, а React.js дає змогу будувати високопродуктивні інтерфейси з компонентами, які повторно використовуються, що значно підвищує ефективність розробки [6].

Для серверної частини обрано Java в поєднанні з Spring Boot, оскільки цей стек технологій є одним з найбільш популярних для створення масштабованих та надійних вебдодатків. Java є потужною та стабільною мовою програмування, яка відома своєю здатністю працювати з великими обсягами даних і високими навантаженнями. Вона також добре підтримує багато бібліотек та фреймворків, що дозволяє спростити процес розробки. Spring Boot забезпечує швидку настройку та запуск проекту, що дозволяє зосередитись на бізнес-логіці, а не на налаштуванні сервера. Зручна інтеграція з Spring Data JPA дає змогу легко працювати з базами даних [11].

Вибір серверної мови був зроблений після порівняння кількох популярних варіантів. Ось основні критерії порівняння:

Таблиця 1.2

Порівняння серверних мов програмування

Мова програмування	Синтаксис і читабельність	Взаємодія з базами даних	Асинхронність і швидкодія
PHP	Простий для освоєння	Підтримка MySQL, PostgreSQL	Менше підтримує асинхронність
Python	Лаконічний синтаксис	Підтримка SQLite, PostgreSQL	Має асинхронне програмування

JavaScript (Node.js)	Модульний синтаксис	Підтримка MongoDB, MySQL, PostgreSQL	Основна підтримка асинхронності
Java	Надійний, підходить для великих проектів	Підтримка баз даних через JDBC, JPA	Підтримка асинхронності через CompletableFuture

З огляду на стабільність, масштабованість і підтримку високих навантажень, було вирішено вибрати Java як серверну мову програмування. Spring Boot надає зручні механізми для розробки RESTful API та забезпечує швидку інтеграцію з базами даних, що є важливим для нашого проєкту.

Висновки до першого розділу

У першому розділі було розглянуто основні вимоги до вебдодатку для вивчення іноземних мов, проведено аналіз існуючих вебплатформ та визначено їхні ключові особливості. На основі дослідження аналогів було вирішено вибрати HTML, CSS, JavaScript, React.js та Java з фреймворком Spring Boot як інструменти для розробки вебдодатку.

Перший розділ кваліфікаційної роботи показав важливість ретельного аналізу ринку онлайн-платформ для навчання мовам та допоміг сформулювати правильну стратегію для подальшої розробки. На основі цього аналізу було забезпечено вибір таких технологій, які максимально відповідають вимогам до функціональності, зручності використання та масштабованості проєкту.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Технічне завдання на розробку системи

Система має назву «Форум DarkForum». Повне найменування системи — вебфорум для обговорення тем, пов'язаних із технологіями, програмуванням та культурою. Коротка назва — форум або програмне забезпечення.

Призначення системи — створити зручну та живу онлайн-платформу, де можна вільно та без зайвого офіціозу спілкуватися на теми, що дійсно хвилюють. Це простір для людей, захоплених технологіями, програмуванням, кібербезпекою та всім, що з цим пов'язано. Чи то ви обговорюєте найновіший фреймворк для веброзробки, ділитеся думками про останні новини у світі гаджетів, чи просто шукаєте поради щодо налаштування свого домашнього сервера — тут для цього знайдеться місце.

Форум створений, щоб стати тим куточком в інтернеті, куди можна прийти з конкретною проблемою — чи то баг у коді, що не вдається відловити, чи питання вибору технології для нового проєкту — і отримати дружню пораду від тих, хто вже проходив через це. Головна мета — надати можливість легко обмінюватися знаннями, допомагати один одному та вчитися разом.

Це також майданчик для формування спільнот за інтересами. Тут можна знайти однодумців, щоб обговорити улюблену гру, зібрати команду для участі в хакатоні або просто стежити за темами, які вам найбільше до душі. Користувачі зможуть легко створювати нові обговорення, щоб поділитися власними напрацюваннями чи поставити питання, активно брати участь у вже наявних дискусіях, залишаючи коментарі, та ділитися корисними посиланнями.

До системи висуваються такі **функціональні вимоги**:

- **Надійність та доступність:** Програмне забезпечення повинно забезпечувати стабільну роботу 24/7 та бути доступним із будь-якого сучасного веббраузера.

- **Управління контентом (для користувачів):**

- Створення, редагування та видалення власних тем і постів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

- Можливість прикріплювати файли та додавати посилання.
- Підтримка різних типів контенту: текст, зображення, відео.
- Перегляд новин сайту та можливість реагувати на них (лайками).
- Спільнота та взаємодія:
 - Система репутації: Наявність таблиці лідерів для перегляду користувачів із найвищою репутацією.
 - Правила форуму: Забезпечення легкого доступу всіх користувачів до перегляду актуальних правил форуму.
- Адміністрування та модерація (для адміністраторів/модераторів):
 - Модерація контенту: Наявність інструментів для редагування, переміщення та видалення тем/постів, а також блокування користувачів.
 - Управління сайтом: Можливість створювати, редагувати та публікувати новини сайту і правила форуму.
 - Статистика: Доступ до панелі зі статистикою сайту (наприклад, кількість нових користувачів, повідомлень, активність за періодами).

Вимоги до інтерфейсу

Інтерфейс форуму "DarkForum" має бути зручним і доступним для широкого кола користувачів. Він повинен включати ясну навігацію з швидким доступом до основних категорій форуму та пошукової системи. Користувачі повинні мати можливість з легкістю переміщатися між темами, категоріями та обговореннями. Важливо, щоб дизайн був логічно організованим, з чітко виділеними кнопками для додавання нових постів, відповідей, лайків та повідомлень.

Для забезпечення зручності використання важливо, щоб елементи навігації були інтуїтивно зрозумілими та легко розпізнаваними. Кожна сторінка повинна мати чітко виражені заголовки, достатньо простору між елементами для зручності перегляду, а інтерактивні елементи, такі як кнопки для лайків або коментарів, повинні миттєво реагувати на дії користувача.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2. Аналіз вимог до програмного продукту

Форум "DarkForum" має відповідати кільком важливим вимогам, щоб забезпечити користувачам зручність, ефективність та надійність у використанні.

Бізнес-вимоги включають необхідність забезпечення конкурентоспроможності продукту на ринку, що досягається через надання унікальних функцій, високоякісного контенту та сучасного дизайну. Важливою складовою є підтримка різних моделей монетизації, наприклад, через рекламу, підписки або платні функції для преміум-користувачів.

Вимоги користувачів для форуму включають можливість реєстрації нових облікових записів, входу в систему, а також відновлення забутого пароля, якщо це необхідно. Користувачі повинні мати можливість редагувати свою особисту інформацію, наприклад, ім'я, пароль та інші параметри профілю, для забезпечення безпеки та зручності використання. Важливим аспектом є надання можливості користувачам переглядати особисту статистику, таку як активність на форумі, кількість створених тем, коментарів і лайків. Користувачі повинні мати змогу писати нові пости, відповідати на існуючі теми та взаємодіяти з іншими учасниками.

Для забезпечення зручності, форум повинен мати функціонал пошуку за фільтрами для швидкого знаходження тем, постів чи користувачів за певними критеріями. Важливою є можливість переглядати історію своїх публікацій та коментарів, а також мати доступ до архіву старих обговорень.

Функціональні вимоги включають забезпечення авторизації користувачів через систему логіну та пароллю, з можливістю збереження даних в базі даних та за допомогою cookies. Авторизовані користувачі повинні мати доступ до персоналізованого функціоналу, такого як редагування профілю та перегляд своєї активності на форумі. Важливим є забезпечення безпеки даних користувачів.

Нефункціональні вимоги включають забезпечення високої швидкодії системи. Система повинна оптимізувати час відповіді на запити, гарантуючи швидкість завантаження сторінок не більше 6 секунд для основних функцій.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

Також важливою є підтримка одночасної роботи 100 користувачів без значного зниження продуктивності. Вебсайт має бути здатний масштабуватися при необхідності, щоб обробляти більшу кількість запитів у разі зростання кількості користувачів.

Системні вимоги для форуму "DarkForum" включають:

Для роботи системи необхідне використання СУБД MySQL версії 5.6 або вище. Забезпечується стабільна взаємодія з сервером за умови доступу до Інтернету зі швидкістю не менше 2 Mbps. Сервер повинен мати щонайменше 1 ГБ оперативної пам'яті. Для роботи з клієнтської сторони передбачається застосування сучасного веббраузера, зокрема Chrome версії 86 або новішої, Firefox версії 82 або новішої, Safari версії 12 або новішої чи Edge версії 86 або новішої. Для комфортного перегляду контенту мінімальна рекомендована роздільна здатність екрану складає 1280x800 пікселів.

На основі цих вимог можна створити діаграму варіантів використання (use case diagram), яка відображатиме взаємодію користувачів з системою, а також ключові функції, які вони можуть виконувати, такі як реєстрація, вхід в систему, публікація постів та коментарів (рис 2.1) [28].

У даній діаграмі використання представлено три типи користувачів: Гість, Авторизований користувач та Адміністратор.

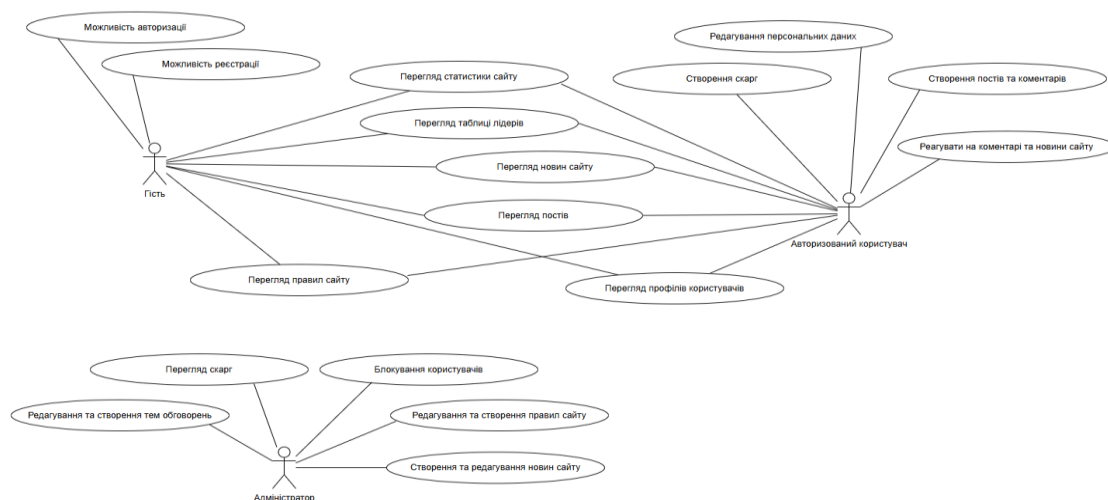


Рис. 2.1. – Діаграма варіантів використання

Гість має базові можливості для ознайомлення з сайтом. Він може переглядати основний контент, такий як статистика сайту, таблиця лідерів, новини, пости та правила сайту. Ключовою функцією для гостя є можливість реєстрації та авторизації, щоб отримати доступ до розширеного функціоналу.

Авторизований користувач успадковує всі права Гостя, але додатково отримує можливості для активної взаємодії з платформою. Він може створювати пости та коментарі, реагувати на новини та коментарі інших, редагувати свої персональні дані та переглядати профілі інших користувачів. Також авторизований користувач може створювати скарги у випадку порушень.

Адміністратор має найвищий рівень доступу та відповідає за управління сайтом і його контентом. До його обов'язків входить перегляд скарг від користувачів, блокування порушників, редагування та створення правил сайту, а також створення та редагування новин і тем для обговорень.

Дана діаграма дозволяє візуалізувати, які дії доступні для кожної ролі, та забезпечує чітке уявлення про функціональність системи.

2.3. Алгоритм роботи та стани програмної системи

Діаграма діяльності ілюструє процес авторизації та реєстрації користувача у вебдодатку (рис. 2.2), поділяючи ролі між гостем і системою. Початок процесу полягає в переході гостя на сторінку авторизації. Далі користувач має визначити, чи має вже існуючий акаунт. Якщо акаунт відсутній, система відображає форму реєстрації, після чого гість вводить необхідні дані [28].

Система виконує валідацію введених даних: якщо дані є некоректними — користувач повертається до етапу введення. У разі правильного введення система зберігає інформацію про нового користувача в базі даних. Після реєстрації або при наявності акаунта, користувач вводить дані для входу.

На цьому етапі система знову виконує перевірку правильності введених даних. Якщо інформація вірна — користувач авторизується і система відображає головну сторінку. Якщо ж дані невірні — користувач повертається до етапу введення авторизаційної інформації.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Ця діаграма дозволяє візуально простежити логіку процесу, ідентифікувати критичні точки взаємодії між користувачем і системою, а також гарантувати, що всі гілки логіки враховані при проєктуванні функціоналу авторизації та реєстрації.

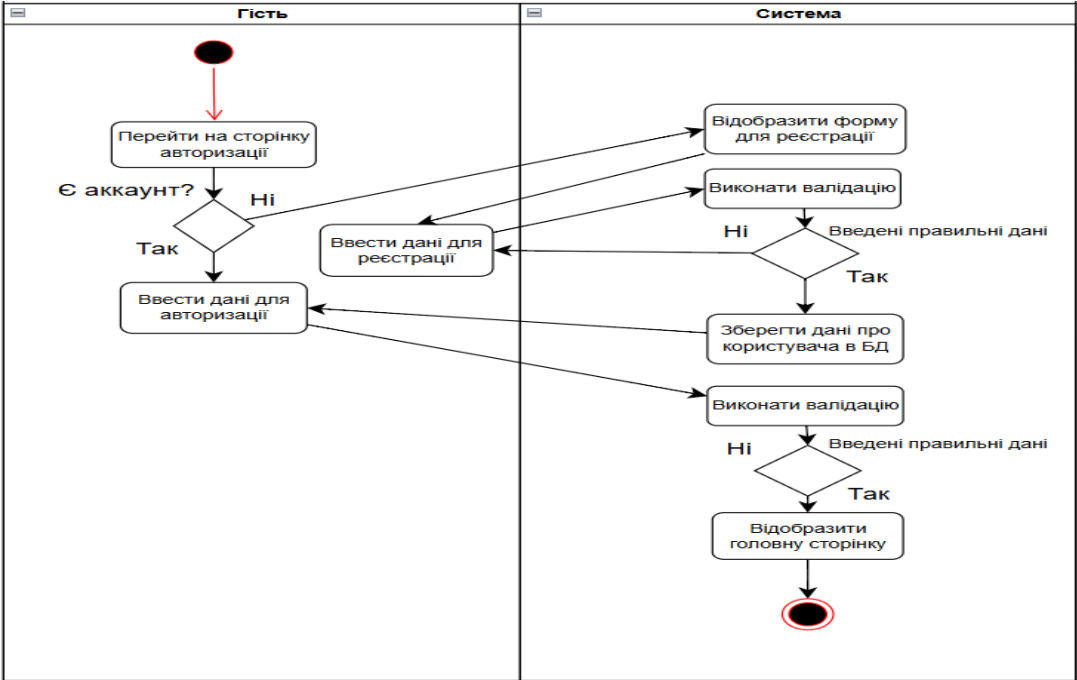


Рис. 2.2 – Діаграма процесу реєстрації та авторизації

Діаграма діяльності ілюструє процес створення посту або коментаря авторизованим користувачем у вебдодатку (рис. 2.3), розділяючи ролі між користувачем і системою. Початок процесу полягає у введенні даних для створення посту або коментаря з боку користувача. Обидва сценарії передбачають подальшу взаємодію із системою для перевірки коректності введених даних.

Система виконує валідацію отриманих даних. У разі виявлення помилки система повертає користувача до етапу введення, відображаючи відповідне повідомлення про помилку. Якщо дані відповідають вимогам, система зберігає їх у базі даних.

Після успішного збереження даних система автоматично оновлює інтерфейс, відображаючи новостворений пост або коментар. На цьому процес завершено.

Ця діаграма дозволяє візуально простежити логіку створення контенту, окреслити точки перевірки даних і реакції системи на дії користувача, а також гарантувати цілісність та правильність роботи функціоналу публікацій.

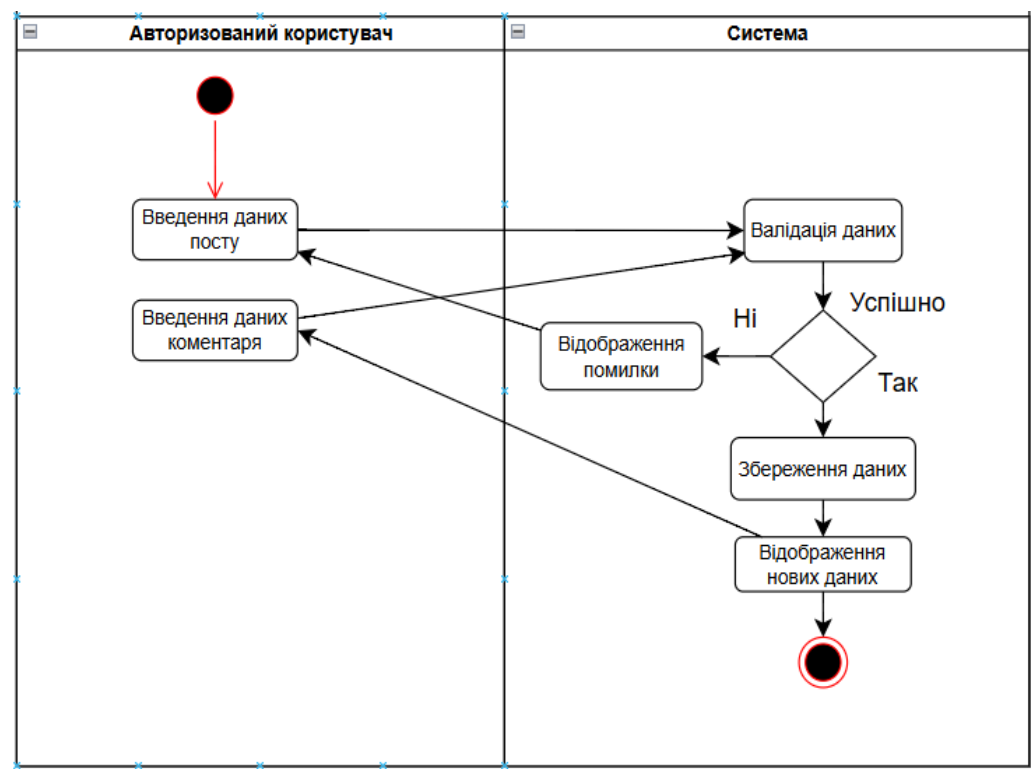


Рис. 2.3. – Діаграма процесу створення постів та коментарів

Процес створення та обробки скарги (рис. 2.4) у вебдодатку починається з того, що користувач переглядає публікації. Якщо під час перегляду користувач виявляє публікацію, що порушує правила платформи, він ініціює створення скарги. У разі відсутності порушень процес завершується.

Після надходження скарги адміністратор вебсайту переходить до її перегляду. Адміністратор перевіряє вказану публікацію на наявність дійсного порушення правил. Якщо порушення не підтверджується, скарга відхиляється, і процес завершується.

Якщо ж публікація справді порушує встановлені правила, адміністратор блокує користувача, що її створив, та видаляє відповідну публікацію. Після завершення цих дій процес обробки скарги вважається завершеним.

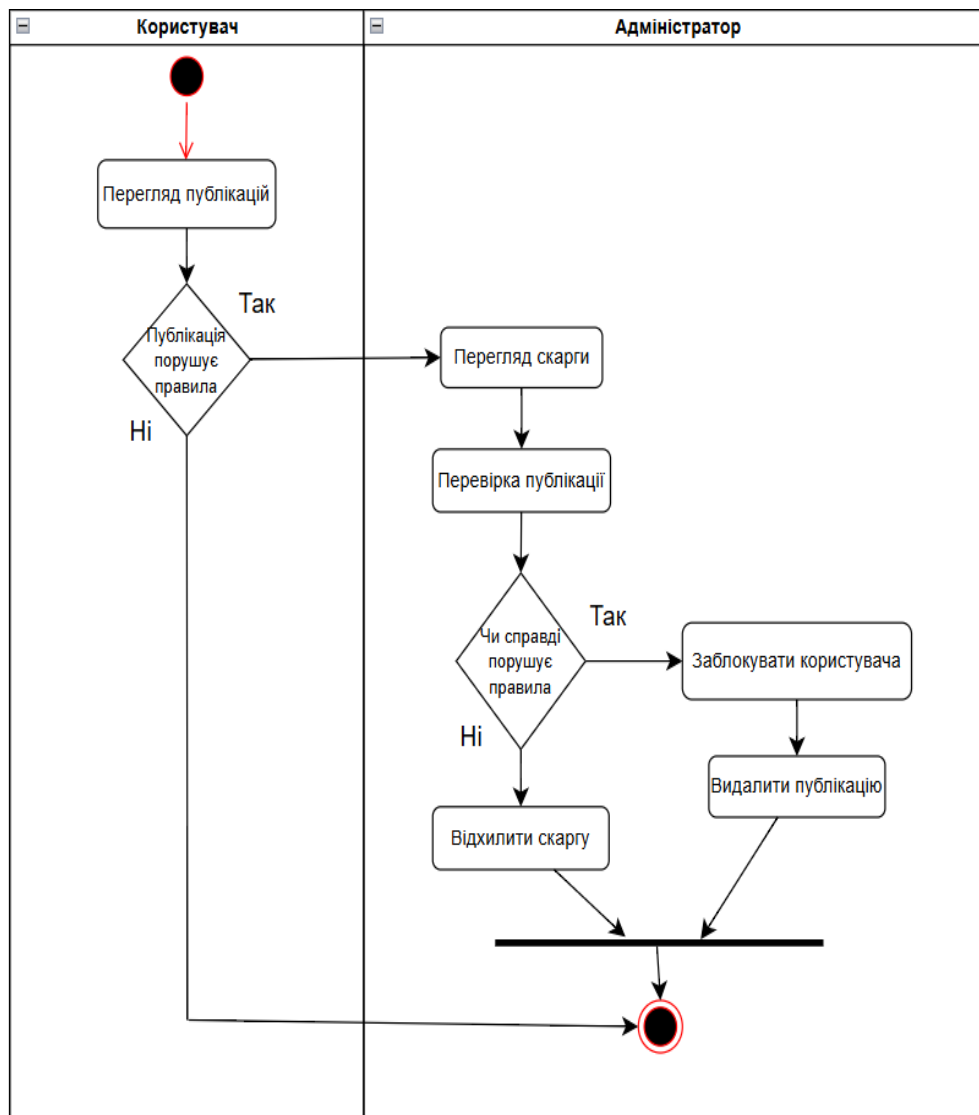


Рис. 2.4 – Діаграма процесу створення та обробки скарг

2.4. Об'єктно-орієнтована модель системи

Об'єктно-орієнтована модель системи представлена у вигляді діаграми класів (рис. 2.5). Діаграма класів наочно відображає структуру та зв'язки між класами в системі, що дозволяє зрозуміти їх взаємодію та залежності. Вона містить інформацію про атрибути класів, методи, які вони реалізують, та типи асоціацій між ними. Діаграма містить вичерпну інформацію про основні компоненти системи. Вона описує атрибути, що визначають стан і властивості кожного класу, а також методи, які інкапсулюють їхню поведінку та реалізують основну бізнес-логіку. Такий структурований підхід дозволяє виявити потенційні архітектурні проблеми на ранніх етапах проєктування, значно знижуючи вартість їх подальшого виправлення.

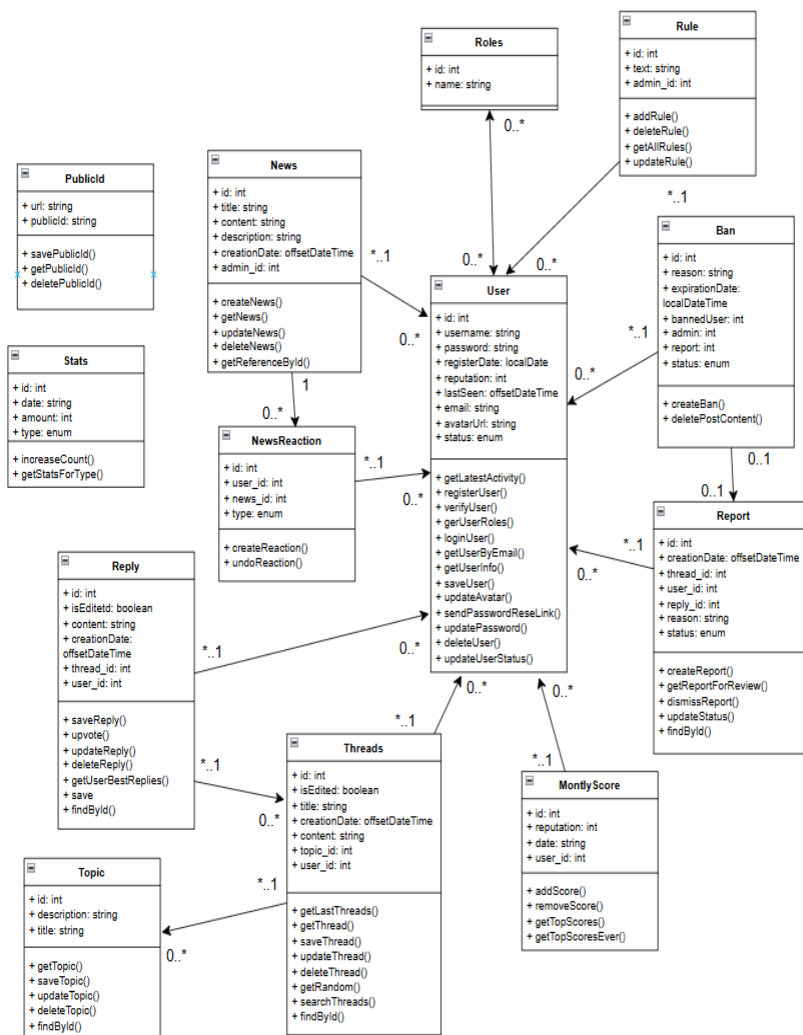


Рис. 2.5 – Діаграма класів

Діаграма класів для вебфоруму розкриває ключові сутності та їхні взаємозв'язки, що формують архітектурну основу додатку. Кожен модуль відіграє критичну роль у функціонуванні платформи.

Центральним елементом системи є модуль користувачів, представлений класом User. Він зберігає всю ключову інформацію про учасників форуму: логін, пароль, репутацію, дату реєстрації та поточний статус. Цей клас є основою для всієї взаємодії на платформі. Він має зв'язок із класом Roles (багато-до-багатьох), що визначає права доступу та привілеї користувачів (наприклад, звичайний користувач чи адміністратор).

Основний функціонал форуму реалізовано через модуль контенту. Клас Topic слугує для групування обговорень за тематикою. Кожен Topic може містити безліч гілок обговорень (Threads), які створюються користувачами. У

свою чергу, кожна гілка (Thread) складається з відповідей (Reply), що дозволяє вести структуровані дискусії. Користувачі є авторами як гілок, так і відповідей, і можуть взаємодіяти з контентом, наприклад, голосуючи за відповіді (upvotes).

Модуль модерації та адміністрування забезпечує порядок на форумі. Користувачі можуть скаржитися на неналежний контент, створюючи Report, який пов'язаний з конкретним Thread або Reply. Адміністратори розглядають ці скарги та можуть вжити заходів, наприклад, видати Ban користувачеві. Клас Ban фіксує інформацію про причину, термін блокування, а також хто з адміністраторів його видав. Крім того, адміністратори керують правилами форуму через клас Rule.

Додаткові, але важливі функції, реалізовані через модулі новин та статистики. Клас News дозволяє адміністраторам публікувати оголошення, а користувачі можуть на них реагувати за допомогою NewsReaction. Для відстеження активності та репутації використовуються класи MonthlyScore, що зберігає щомісячні бали користувачів, та Stats, який агрегує загальну статистику на форумі (наприклад, кількість постів, нових користувачів тощо).

2.5. Взаємодія об'єктів системи

Діаграма послідовності – це графічне зображення, яке ілюструє взаємодії між об'єктами у певній системі, упорядковані за часом. Вона представляє послідовність повідомлень, які передаються між об'єктами для виконання конкретного сценарію.

Основна мета діаграми послідовності полягає у візуалізації порядку виконання дій і взаємодій у часі, що дозволяє краще зрозуміти і задокументувати роботу системи. Діаграма послідовності також допомагає виявити можливі проблеми у взаємодії об'єктів і оптимізувати процеси.

У діаграмі послідовності (рис. 2.6) створення нового допису на вебфорумі, процес починається з того, що Користувач ініціює перехід до обраної теми. Інтерфейс користувача (View) перехоплює цю дію та надсилає запит на обробку події переходу до Контролера (Controller). Контролер, у свою чергу, звертається до Моделі (Model), щоб отримати дані про всі дописи, які містяться в цій темі.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Модель повертає запитані дані Контролеру, який передає їх у View для завантаження відображення. Після цього View відображає користувачу повний список дописів у темі.

Наступним кроком є створення нового допису. Користувач взаємодіє з інтерфейсом, відкриваючи форму та вводячи дані нового повідомлення. Після відправки форми, View обробляє подію створення допису та передає дані до Контролера. Контролер надсилає команду Моделі, яка відповідає за додавання нового допису до бази даних. Після успішного збереження, Контролер інформує View про результат операції. View оновлює інтерфейс, миттєво додаючи щойно створений допис до загального списку, який бачить Користувач.

Таким чином, дана діаграма послідовності наочно ілюструє повний цикл взаємодії в системі: від запиту даних для перегляду до створення нового контенту та його відображення. Вона демонструє чіткий розподіл відповідальності за логікою MVC: View відповідає за взаємодію з користувачем, Controller керує потоком даних та логікою, а Model управляє даними та бізнес-правилами [29].



Рис. 2.6 – Діаграма послідовності створення нового допису

Діаграма кооперації використовується для показу взаємодії об’єктів, вона відображає як об’єкти взаємодіють для виконання завдань у конкретному варіанті використання. Діаграма кооперації корисна для зображення простих взаємодій між меншою кількістю об’єктів.

У діаграмі кооперації (рис. 2.7) взаємодії на вебфорумі відбувається наступне: процес починається з «Головної сторінки форуму», яка є точкою входу для користувача. За допомогою дії «1: Перегляд», користувач переходить до

«Списку тем обговорень», де може ознайомитися з усіма доступними на форумі дискусіями.

Наступним кроком є «2: Вибір теми». Користувач обирає конкретну тему зі списку, що призводить до переходу на сторінку «Список дописів теми». На цьому етапі користувач може прочитати всі повідомлення, залишені в рамках обраного обговорення.

Щоб взяти участь у дискусії, користувач ініціює дію «3: Створення допису». Цей процес завершується створенням об'єкта «Новий допис», який представляє собою повідомлення користувача, готове для додавання до теми.

Таким чином, ця послідовність дій демонструє ключовий сценарій використання вебфоруму: від загального перегляду тем до вибору конкретної та створення власного контенту. Діаграма чітко ілюструє логічний шлях користувача та взаємозв'язок основних об'єктів системи.

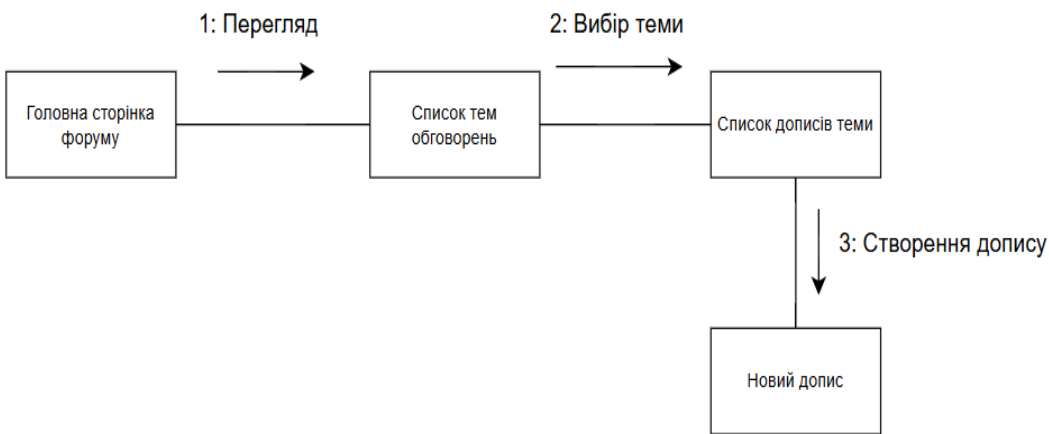


Рис. 2.7 – Діаграма кооперації створення допису

2.6. Розробка бази даних системи

База даних вебфоруму реалізована за допомогою MySQL та містить 15 таблиць, що забезпечують повну функціональність платформи. На рисунку [Вставте номер вашого рисунка] показана загальна схема бази даних. Детальніші рисунки головних таблиць бази даних, включаючи їх структуру, взаємозв'язки та атрибути, наведені у Додатку А.

Продовження таблиці 2.1

reputation	INT	—	—	Репутація користувача на форумі
last_seen	TIMESTAMP	—	—	Час останньої активності
email	VARCHAR(100)	—	—	Електронна пошта користувача
status	ENUM(...)	—	—	Статус акаунта
avatar_url	VARCHAR(20...)	—	—	URL-адреса аватара користувача

Таблиця «threads» зберігає дані про дописи (треди), створені на форумі, які є основними одиницями контенту. Кожен запис у цій таблиці представляє унікальну тему для обговорення та слугує відправною точкою для всіх подальших коментарів.

Таблиця 2.2

Опис полів таблиці «threads»

Назва	Тип даних	ПК	ЗК	Опис поля
id	INT	+	—	Унікальний ідентифікатор допису
tittle	VARCHAR(255)	—	—	Заголовок
topic_id	INT	—	+	Ідентифікатор розділу
user_id	INT	—	+	Ідентифікатор автора допису
creation_date	TIMESTAMP	—	—	Дата та час створення допису
content	TEXT	—	—	Основний вміст
is_edited	TINYINT(1)	—	—	Прапорець, що вказує, чи редагувалася тема

Таблиця «reply» містить відповіді користувачів у межах конкретних тем.

Таблиця 2.3

Опис полів таблиці «reply»

Назва	Тип даних	ПК	ЗК	Опис поля
id	INT	+	–	Унікальний ідентифікатор відповіді
thread_id	INT	–	+	Ідентифікатор теми, до якої належить відповідь
user_id	INT	–	+	Ідентифікатор автора відповіді
content	TEXT	–	–	Текст відповіді
creation_date	TIMESTAMP	–	–	Дата та час створення відповіді
is_edited	TINYINT(1)	–	–	Прапорець, що вказує, чи редагувалася відповідь

Таблиця «bans» використовується для обліку блокувань (банів) користувачів, що є важливою частиною системи модерації.

Таблиця 2.4

Опис полів таблиці «bans»

Назва	Тип даних	ПК	ЗК	Опис поля
id	INT	+	–	Унікальний ідентифікатор запису про бан
reason	TEXT	–	–	Причина блокування
expiration_date	TIMESTAMP	–	–	Дата та час закінчення терміну бану
user_id	INT	–	+	Ідентифікатор заблокованого користувача
admin_id	INT	–	+	Ідентифікатор адміністратора, який видав бан
status	ENUM(...)	–	–	Статус бану

Продовження таблиці 2.4

report_id	INT	–	+	Ідентифікатор скарги, що призвела до бану (опційно)
-----------	-----	---	---	---

Таблиця «topics» організовує форум у тематичні розділи, що дозволяє структурувати контент.

Таблиця 2.5

Опис полів таблиці «topics»

Назва	Тип даних	ПК	ЗК	Опис поля
id	INT	+	–	Унікальний ідентифікатор розділу
tittle	VARCHAR(200)	–	–	Назва розділу
description	VARCHAR(25...)	–	–	Короткий опис розділу

Таблиця «reports» призначена для системи модерації, дозволяючи користувачам надсилати скарги на контент, що порушує правила. Вона фіксує інформацію про об'єкт скарги (пост або коментар), ініціатора звернення та причину порушення.

Таблиця 2.6

Опис полів таблиці «reports»

Назва	Тип даних	ПК	ЗК	Опис поля
id	INT	+	–	Унікальний ідентифікатор скарги
thread_id	INT	–	+	ID теми, на яку подано скаргу (якщо скарга на тему)
user_id	INT	–	+	ID користувача, який подав скаргу

Продовження таблиці 2.6

reply_id	INT	–	+	ID відповіді, на яку подано скаргу (якщо скарга на відповідь)
reason	TEXT	–	–	Причина скарги
creation_date	TIMESTAMP	–	–	Дата та час створення скарги
status	ENUM(...)	–	–	Статус скарги

Таблиця «user_roles» є сполучною таблицею, яка реалізує зв'язок "багато-до-багатьох" між користувачами та їх ролями.

Таблиця 2.7

Опис полів таблиці «user_roles»

Назва	Тип даних	ПК	ЗК	Опис поля
user_id	INT	–	+	Ідентифікатор користувача
role_id	INT	–	+	Ідентифікатор ролі

Таблиця «verification» зберігає токени для верифікації акаунтів користувачів після реєстрації.

Таблиця 2.8

Опис полів таблиці «verification»

Назва	Тип даних	ПК	ЗК	Опис поля
id	INT	+	–	Унікальний ідентифікатор запису
user_id	INT	–	+	Ідентифікатор користувача, для якого створено токен

token	VARCHAR(150)	—	—	Унікальний токен верифікації
experation_date	DATETIME	—	—	Дата та час закінчення терміну дії токена

Висновок до другого розділу

У другому розділі кваліфікаційної роботи було розглянуто і проаналізовано технічне завдання на розробку вебфоруму. Проведено детальний аналіз вимог до програмного продукту, охоплюючи як функціональні (створення тем та дописів, система ролей, модерація), так і нефункціональні вимоги (продуктивність, безпека, масштабованість). Розроблено алгоритм роботи системи та проаналізовано можливі стани програмної системи, що дозволило визначити ключові аспекти її функціонування.

Створено об'єктно-орієнтовану модель вебфоруму, яка включає опис основних об'єктів, таких як «Користувач», «Тема», «Допис», та їх взаємодію. Взаємодія об'єктів системи була проілюстрована за допомогою UML діаграм, що демонструють послідовність і кооперацію між різними компонентами системи під час виконання основних процесів, зокрема створення нового допису та перегляду обговорень.

Особлива увага була приділена розробці реляційної бази даних системи. Було спроектовано логічну модель, визначено структуру ключових таблиць, їх атрибути та зв'язки для забезпечення цілісності та ефективного зберігання даних. У результаті було створено комплексну технічну базу для подальшої розробки та впровадження вебфоруму, що забезпечує надійну основу для його успішної реалізації.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1. Порядок встановлення та налаштування системи

Вебфорум розроблено за сучасною клієнт-серверною архітектурою, що забезпечує гнучкість, масштабованість та незалежність розробки окремих компонентів. Застосунок використовує принципи архітектурного шаблону MVC (Model-View-Controller), які розподілені між клієнтською та серверною частинами. Серверна частина (backend), реалізована на Spring Boot, відповідає за Model (робота з даними та бізнес-логіка) та Controller (обробка HTTP-запитів через REST API). Клієнтська частина (frontend), реалізована на React, відповідає за View (відображення інтерфейсу користувача та взаємодія з ним) [30].

Структура серверної частини (backend) на Spring Boot (рис. 3.1) організована за пакетами для чіткого розділення відповідальності. Пакет controller містить REST-контролери, які приймають запити від клієнта. Вони, у свою чергу, викликають бізнес-логіку, що зосереджена в пакетах service. Для взаємодії з базою даних сервіси використовують інтерфейси з пакета repository. Основою моделі даних (Model) є класи в пакеті entity, які описують структуру таблиць. Для передачі даних між клієнтом та сервером використовуються об'єкти з пакета dto (Data Transfer Objects), а за безпеку та авторизацію відповідає конфігурація в пакеті security.

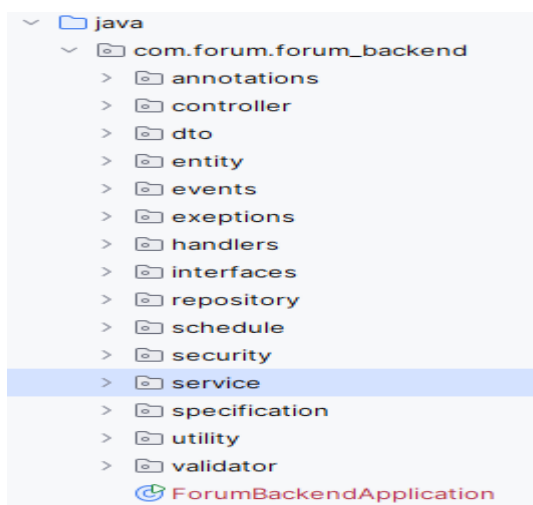


Рис. 3.1 – Файлова структура серверної частини

Структура клієнтської частини (frontend) на React (рис. 3.2) побудована навколо компонентного підходу. Основу інтерфейсу складають компоненти в папці pages, що представляють собою окремі сторінки вебдодатку. Ці сторінки будуються з менших, перевикористовуваних UI-елементів, розташованих у папці components. Для глобального управління станом, наприклад, для зберігання даних про користувача, використовується React Context API, реалізація якого знаходиться в папці context. Статичні файли, такі як зображення, зберігаються у папці assets, а точкою входу в додаток є файл main.jsx, тоді як основна логіка маршрутизації міститься в App.jsx.

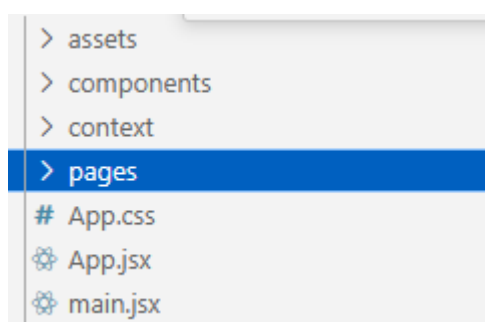


Рис. 3.2 – Файлова структура клієнтської частини

Таке чітке розділення на клієнтську та серверну частини, а також логічна структуризація кожного з проєктів, сприяє організації та зрозумілості коду. Це полегшує незалежну розробку, тестування та розгортання компонентів, а також забезпечує високу масштабованість та легкість у підтримці вебфоруму.

3.2. Розробка основних функцій вебдодатку

Зайшовши на вебсайт, користувач в першу чергу проходить процес аутентифікації. Вхід на сайт реалізований через заповнення форми, яка містить поля для електронної пошти та пароля, а також опцію «Запам'ятати мене» для збереження сесії при наступних візитах з цього пристрою.

На клієнтській стороні (frontend), перед відправкою даних на сервер, відбувається їх валідація. Для цього в React-компоненті використовується бібліотека zod, яка перевіряє вхідні дані на відповідність заздалегідь визначеній схемі loginSchema (рис. 3.3). Це дозволяє миттєво повідомити користувача про помилки, такі як недійсний формат електронної пошти або пароль, що не відповідає вимогам безпеки.

```
const loginSchema = z.object({
  email: z.string().email('Введіть дійсну адресу електронної пошти'),
  password: z
    .string()
    .min(8, 'Пароль має містити щонайменше 8 символів')
    .regex(/^[a-zA-Z0-9]+$/, 'Пароль може містити лише літери та цифри')
});
```

Рис. 3.3 – Схема валідації даних для входу

Після успішної клієнтської валідації, функція `handleSubmit` (рис. 3.4) за допомогою бібліотеки `axios` надсилає асинхронний POST-запит на серверний ендпоінт `/api/auth/login`. У тілі запиту передаються дані форми у форматі JSON. Функція також обробляє відповіді від сервера, відображаючи повідомлення про помилки (наприклад, невірні дані для входу або блокування акаунту) або, у разі успіху, оновлюючи стан додатку та перенаправляючи користувача на головну сторінку.

```
loginSchema.parse({
  email: formData.email,
  password: formData.password
});

setIsError(false);
setErrorMessage('');
const res = await axios.post('http://localhost:8080/api/auth/login',
  formData,
  { headers: { 'Content-Type': 'application/json' },
    withCredentials: true });
localStorage.setItem('rememberMe', formData.remember);
login(res.data);
navigate('/');
```

Рис. 3.4 – Функція обробки відправки форми входу

На сервері (backend), цей запит приймає `AuthController` (рис 3.5). Контролер делегує обробку логіки `UserService` та, у разі успішної аутентифікації, використовує `jwtUtility` для генерації JWT (JSON Web Token). Цей токен встановлюється у HTTP-only cookie, що захищає його від доступу через JavaScript (XSS-атаки) і забезпечує безпеку сесії.

```

@PostMapping("/login") no usages new *
public ResponseEntity<LoginRequestResponseDTO> login(@RequestBody LoginRequestDTO loginRequestDAO, HttpServletResponse response){
    User user = userService.loginUser(loginRequestDAO);
    String token = jwtUtility.generateToken(user);
    jwtUtility.setJwtTokenCookie(response, token, loginRequestDAO.remember());
    return ResponseEntity.ok(new LoginRequestResponseDTO(user));
}

```

Рис. 3.5 – Метод контролера для обробки запиту на вхід

Основна логіка перевірки даних користувача зосереджена в UserService (рис. 3.6). Сервіс знаходить користувача в базі даних за електронною поштою. Для перевірки пароля використовується PasswordEncoder, який порівнює хеш введеного пароля з хешем, що зберігається в базі даних. Такий підхід гарантує, що паролі ніколи не зберігаються і не передаються у відкритому вигляді.

```

public User loginUser(LoginRequestDTO loginRequestDAO){ 1 usage new *
    User user = userRepository.getUserByEmail(loginRequestDAO.email()).orElseThrow(InvalidCredentialsException::new);
    if(!passwordEncoder.matches(loginRequestDAO.password(), user.getPassword())){
        throw new InvalidCredentialsException();
    }else{
        user.setLastSeen(OffsetDateTime.now());
        return user;
    }
}

```

Рис. 3.6 – Метод сервісу для аутентифікації користувача

Функціональність «Запам'ятати мене» реалізована за допомогою localStorage на клієнті та довготривалих cookie на сервері. При завантаженні додатку AuthContext перевіряє наявність відповідного запису та автоматично надсилає запит на сервер для відновлення сесії, забезпечуючи безшовний досвід для користувача (рис. 3.7).

Спочатку код перевіряє значення rememberMe у localStorage. Якщо воно true, додаток автоматично надсилає GET-запит на захищений ендпоінт /api/auth/me. Завдяки опції withCredentials: true, браузер автоматично додає до цього запиту раніше збережене довготривале HTTP-only cookie. Сервер отримує цей запит, валідує JWT-токен з cookie і, якщо він дійсний, повертає дані про користувача. Клієнтський додаток, отримавши ці дані, оновлює свій глобальний стан, фактично "входячи" користувача в систему без будь-яких видимих дій з

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

його боку. Якщо ж токен невалідний або його термін дії закінчився, сервер поверне помилку, і користувач залишиться неавторизованим.

```
useEffect(() => {
  const fetchUser = async () => {
    const rememberMe = localStorage.getItem('rememberMe') === "true";
    if (rememberMe) {
      try {
        const response = await axios.get(
          "http://localhost:8080/api/auth/me",
          {
            withCredentials: true,
          }
        );
        setUser(response.data);
      } catch (err) {
        setUser(null);
        if (err.response.data.errorCode === "USER_BANNED") {
          setBanDetails(err.response.data);
        }
      }
    }
  };

  fetchUser();
}, []);
```

Рис. 3.7 – Відновлення сесії користувача в AuthContext

Якщо користувач забув пароль, система надає безпечний механізм для його відновлення. Цей процес починається на спеціальній сторінці, де користувачу пропонується ввести свою електронну адресу. Як і у випадку з формою входу, на клієнтській стороні (frontend) застосовується валідація за допомогою бібліотеки zod. Схема requestPasswordResetSchema (рис. 3.8) перевіряє, що поле не є порожнім і містить адресу у правильному форматі.

```
const requestPasswordResetSchema = z.object({
  email: z
    .string()
    .min(1, "Електронна пошта є обов'язковою")
    .email("Будь ласка, введіть дійсну електронну адресу."),
});
```

Рис. 3.8 – Схема валідації для запиту на відновлення пароля

Після успішної валідації функція handleSubmit у компоненті RequestPasswordReset (рис 3.9) надсилає POST-запит на серверний ендпоінт /api/user/password-reset-request. Важливим аспектом безпеки є те, що користувач отримує однакове повідомлення про успіх незалежно від того, чи існує акаунт з

такою поштою. Це запобігає атакам типу "email enumeration", коли зловмисник може перевірити наявність зареєстрованих поштових адрес у системі.

```
requestPasswordResetSchema.parse({ email });
setIsSubmitting(true);

await axios.post(
  "http://localhost:8080/api/user/password-reset-request",
  { email: email },
  { headers: { "Content-Type": "application/json" } }
);

setSuccessMessage(
  "Якщо акаунт з такою електронною поштою існує, посилання для скидання пароля було надіслано. Будь ласка, перевірте свою поштову скриньку."
);
setEmail("");
```

Рис. 3.9 – Функція обробки запиту на відновлення пароля

На серверній стороні (backend) запит обробляється відповідним контролером, який викликає метод `sendPasswordRestLink` у `UserService` (рис. 3.10). Сервіс спочатку перевіряє, чи існує користувач з вказаною електронною поштою. Якщо користувач знайдений, система генерує новий, короткотривалий JWT. Цей токен слугує безпечним та унікальним ідентифікатором запиту на скидання пароля. Далі `emailService` відправляє на пошту користувача лист, що містить спеціальне посилання, до якого включено цей згенерований токен.

```
public String sendPasswordRestLink>PasswordResetRequestDTO requestDTO) { 1usage new *
    User user = userRepository.getUserByEmail(requestDTO.email());
    orElseThrow(() -> new UserNotFoundException("Користувача не знайдено"));
    emailService.sendPasswordRestEmail(user.getEmail(), jwtUtility.generateToken(user));
    return "OK";
}
```

Рис. 3.10 – Метод для відправки посилання на відновлення пароля

Коли користувач переходить за посиланням з листа, клієнтський додаток зчитує токен з URL-адреси. Цей токен надсилається на сервер для валідації. Якщо токен дійсний і не протермінований, сервер надає дозвіл на зміну пароля, і користувачу відображається форма для введення нового пароля. Після успішного оновлення пароля в базі даних, старий токен стає недійсним.

Таким чином, у всіх частинах вебдодатку, які вимагають авторизації, кожен запит до API супроводжується JWT-токеном, що дозволяє серверу

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

ідентифікувати користувача та перевіряти його права доступу до захищених ресурсів, забезпечуючи надійну та безпечну роботу системи.

У вебфорумі реалізовано функціонал оцінки дописів (лайків), що є важливою частиною системи репутації. Взаємодія реалізована за допомогою React на клієнтській стороні та асинхронних запитів до REST API за допомогою бібліотеки axios, що забезпечує динамічне оновлення інтерфейсу без перезавантаження сторінки.

При кліку на іконку лайка біля допису, викликається функція, яка ініціює асинхронний POST-запит до серверного ендпоінту `/api/reply/{id}/upvote`, де `{id}` – це ідентифікатор відповідного допису (рис. 3.11). Запит надсилається з опцією `withCredentials: true`, що дозволяє серверу ідентифікувати користувача за допомогою JWT-токена, який зберігається в cookie.

```
await axios.post(
  `http://localhost:8080/api/reply/${id}/upvote`,
  {},
  {
    headers: {
      "Content-Type": "application/json",
    },
    withCredentials: true,
  }
).then((response) => {
  if (user.upvotedReplies.includes(id)) {
    setUserHasUpvoted(false);
    setUpvote((prevUpvotes) => prevUpvotes - 1);
    user.upvotedReplies = user.upvotedReplies.filter((replyId) => replyId !== id);
  } else {
    setUserHasUpvoted(true);
    setUpvote((prevUpvotes) => prevUpvotes + 1);
    user.upvotedReplies.push(id);
  }
})
```

Рис. 3.11 – Клієнтська логіка обробки лайка

Після отримання успішної відповіді від сервера, клієнтський додаток не перезавантажує сторінку, а миттєво оновлює інтерфейс. Логіка в блоці `.then()` перевіряє, чи ставив користувач лайк раніше. Залежно від цього, стан компонента оновлюється: змінюється вигляд іконки лайка, лічильник лайків збільшується або зменшується, а також оновлюється локальний масив

user.upvotedReplies. Цей підхід, відомий як "оптимістичне оновлення UI", забезпечує плавну та швидку взаємодію для користувача.

На серверній стороні запит обробляється методом upvote у відповідному сервісі (рис. 3.12). Цей метод реалізує логіку "перемикача" (toggle). Спочатку система знаходить користувача, що надіслав запит, та допис, який оцінюють.

```
public ResponseEntity<String> upvote(String email, int id) { 1 usage new *
    User user = userService.getUserByEmail(email)
        .orElseThrow(() -> new EntityNotFoundException("Користувача не знайдено"));
    Reply reply = repliesRepository.findById(id);
    if(user.getUpvotes().contains(reply)){
        user.getUpvotes().remove(reply);
        userService.decreaseUserReputation(reply.getUser().getId());
        monthlyScoreService.removeScore(reply.getUser().getId());
        reply.getUpvotedBy().remove(user);
    }else{
        user.getUpvotes().add(reply);
        userService.increaseUserReputation(reply.getUser().getId());
        monthlyScoreService.addScore(reply.getUser().getId());
        reply.getUpvotedBy().add(user);
    }
    userService.save(user);
    return ResponseEntity.ok( body: "ok");
}
```

Рис. 3.12 – Серверна логіка обробки лайка

Якщо користувач вже оцінив цей допис, система видаляє зв'язок між ними, зменшує загальну репутацію автора допису, а також його місячний рейтинг. У протилежному випадку, якщо лайк ставиться вперше, система створює новий зв'язок та, навпаки, збільшує репутацію та місячний рейтинг автора. Після виконання цих операцій зміни зберігаються в базі даних.

Такий підхід забезпечує не тільки динамічну взаємодію на клієнті, але й реалізує складну бізнес-логіку системи репутації на сервері, гарантуючи цілісність даних.

Для підвищення залученості користувачів та спрощення навігації по контенту, у вебфорумі реалізовано функціонал переходу до випадкової теми

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

обговорення. Цей механізм дозволяє користувачам відкривати для себе нові або старі дискусії, на які вони могли б не натрапити під час звичайного перегляду.

Реалізація цієї функції на клієнтській стороні (frontend) використовує можливості бібліотеки react-router-dom, зокрема loaders для отримання даних перед рендерингом компонента. Коли користувач ініціює перехід на спеціальний маршрут, пов'язаний з компонентом RandomThread, спочатку виконується функція RandomThreadLoader (рис 3.13).

```
export async function RandomThreadLoader() {  
  const response = await axios.get('http://localhost:8080/api/thread/random');  
  return response.data;  
}
```

Рис. 3.13 – Функція-завантажувач для отримання випадкового допису

Ця асинхронна функція надсилає GET-запит на серверний ендпоінт /api/thread/random. Сервер, у свою чергу, обробляє цей запит та повертає у відповіді унікальний ідентифікатор (ID) однієї випадково обраної теми.

Після того, як RandomThreadLoader успішно отримує ID, цей ідентифікатор передається як data у компонент RandomThread (рис. 3.14). Компонент RandomThread не рендерить жодного видимого інтерфейсу. Його єдине завдання – виконати програмну навігацію (перенаправлення) на сторінку обраної теми. Для цього використовується хук useNavigate з react-router-dom.

```
export default function RandomThread() {  
  const data = useLoaderData();  
  const navigate = useNavigate();  
  navigate('/thread/' + data);  
}
```

Рис. 3.14 – Компонент для перенаправлення на випадкову тему

На серверній стороні (backend) логіка є простою та ефективною. Запит обробляється методом getRandom у відповідному контролері, який викликає метод findRandomRecord з ThreadRepository (рис 3.15).

```
public String getRandom() { 1 usage new *  
  int randomId = threadRepository.findRandomRecord();  
  return Integer.toString(randomId);  
}
```

Рис. 3.15 – Серверна логіка для вибору випадкового допису

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

Для вибору випадкового запису використовується нативний SQL-запит, який є високопродуктивним рішенням для баз даних, таких як MySQL. Команда `ORDER BY RAND() LIMIT 1` ефективно обирає один випадковий рядок з таблиці `threads` та повертає його `id`. Цей ID потім конвертується у рядок і відправляється клієнту.

Таким чином, комбінація клієнтської логіки з використанням `react-router-dom` та ефективного SQL-запиту на сервері дозволяє реалізувати швидку та безшовну для користувача функцію переходу до випадкової теми.

Важливою складовою будь-якого онлайн-співтовариства є система модерації, яка дозволяє підтримувати порядок та дотримуватися правил платформи. У розробленому вебфорумі реалізовано функціонал подання скарг (репортів) на контент, що порушує правила. Цей механізм дозволяє користувачам повідомляти адміністрацію про неналежні теми або дописи.

Процес ініціюється користувачем через спеціальну форму, яка зазвичай відкривається у модальному вікні. Перед відправкою скарги на сервер, введені дані проходять валідацію на клієнтській стороні (frontend). Для цього використовується бібліотека `zod` та схема `reportSchema` (рис. 3.16), яка перевіряє, що причина скарги має мінімально та максимально допустиму довжину.

```
const reportSchema = z.object({
  reason: z.string()
    .min(4, { message: "Причина повинна містити щонайменше 4 символи." })
    .max(500, { message: "Причина не може перевищувати 500 символів." }),
});
```

Рис. 3.16 – Схема валідації для форми скарги

Якщо валідація пройдена успішно, клієнтський додаток надсилає асинхронний POST-запит на ендпоінт `/api/reports` (рис. 3.17). У тілі запиту передається причина скарги, а також тип контенту (`thread` або `reply`) та його ідентифікатор (`postId`). Це дозволяє серверу точно ідентифікувати об'єкт скарги. Після отримання успішної відповіді, користувачу відображається повідомлення про успішне надсилання, а форма автоматично закривається через короткий проміжок часу.

```

        axios.post('http://localhost:8080/api/reports', {
            reason: result.data.reason,
            type: isMainPost ? 'thread' : 'reply',
            postId: postId
        }, {
            withCredentials: true,
            headers: {
                "Content-Type": "application/json",
            }
        })
        setSuccessMessage(`Звіт успішно надіслано!`);
        setReason('');
        setIsReportedSuccessfully(true);
        setTimeout(() => {
            onClose();
        }, 1000);
    }
}

```

Рис. 3.17 – Надсилання запиту на створення скарги

На серверній стороні (backend) запит обробляється методом `createReport` у відповідному сервісі (рис. 3.18). Цей метод є універсальним для обробки скарг як на теми, так і на дописи.

```

public String createReport(CreateReportRequestDTO createReportRequestDTO){ 1 usage
    boolean isReply = createReportRequestDTO.type().equals("reply");
    Post post = isReply ? replyService.findById(createReportRequestDTO.postId())
        : threadService.findById(createReportRequestDTO.postId())
        .orElseThrow(() -> new EntityNotFoundException("Пост не знайдено"));
    User user = post.getUser();
    Report report = Report.builder().user(user)
        .creationDate(OffsetDateTime.now())
        .reason(createReportRequestDTO.reason())
        .reply(isReply ? (Reply) post : null)
        .thread(isReply ? null : (Threads) post)
        .status(ReportStatus.REVIEW).build();
    reportRepository.save(report);
    return "OK";
}

```

Рис. 3.18 – Серверна логіка створення скарги

Спочатку метод визначає тип контенту на основі поля `type` з запиту і завантажує відповідний об'єкт (тему або допис) з бази даних. Далі, використовуючи патерн `Builder`, створюється новий об'єкт `Report`. У ньому заповнюються всі необхідні поля: користувач, на якого подано скаргу, причина, посилання на відповідний допис чи тему, а також встановлюється початковий

статус REVIEW (на розгляді). Після цього новий запис про скаргу зберігається в базі даних.

Ця реалізація забезпечує надійний механізм для модерації контенту, дозволяючи адміністраторам ефективно реагувати на порушення правил спільноти.

3.3. Опис інструкції користувача

Спочатку було обрано дизайн та розмітку сторінок у темній стилістиці, і на основі цього було розроблено повноцінний вебфорум «DarkForum».

Головна сторінка вебфоруму (рис. 3.19) спроектована для інтуїтивної навігації та швидкого доступу до ключового функціоналу. Верхня частина сторінки містить навігаційне меню (хедер), яке включає логотип «DarkForum» та основні посилання: «Головна сторінка», «Таблиця лідерів», «Статистика», «Правила», «Новини» та «Випадковий пост». Праворуч розташований блок аутентифікації: для неавторизованих користувачів доступні кнопки «Авторизація» та «Реєстрація», які після входу змінюються на посилання на профіль користувача та кнопку виходу.

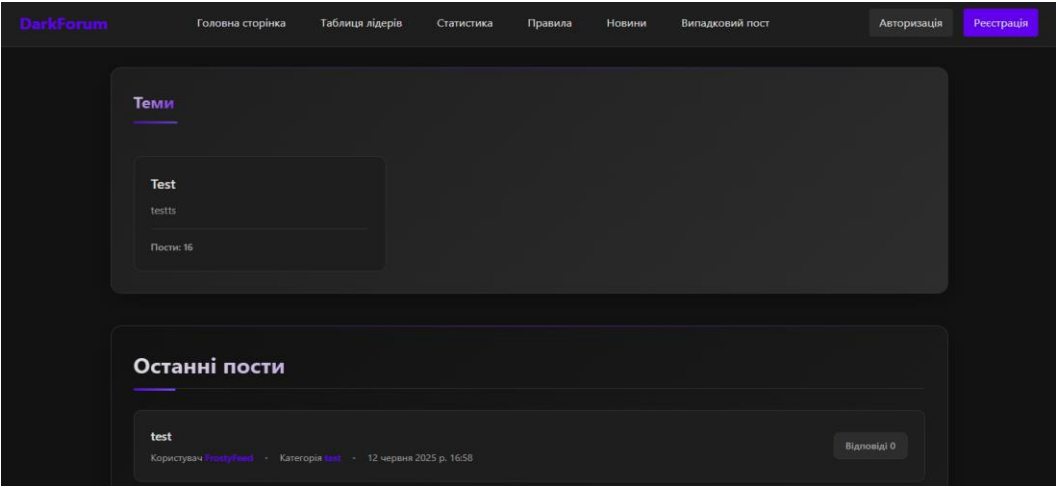


Рис. 3.10 – Головна сторінка

На головній сторінці представлено ключові розділи для взаємодії з контентом. Перший блок «Теми» (рис. 3.11) відображає основні категорії обговорень у форматі карток. Кожна картка містить назву теми, її короткий опис та кількість дописів у ній, що дозволяє користувачам швидко оцінити популярність та зміст кожної категорії.

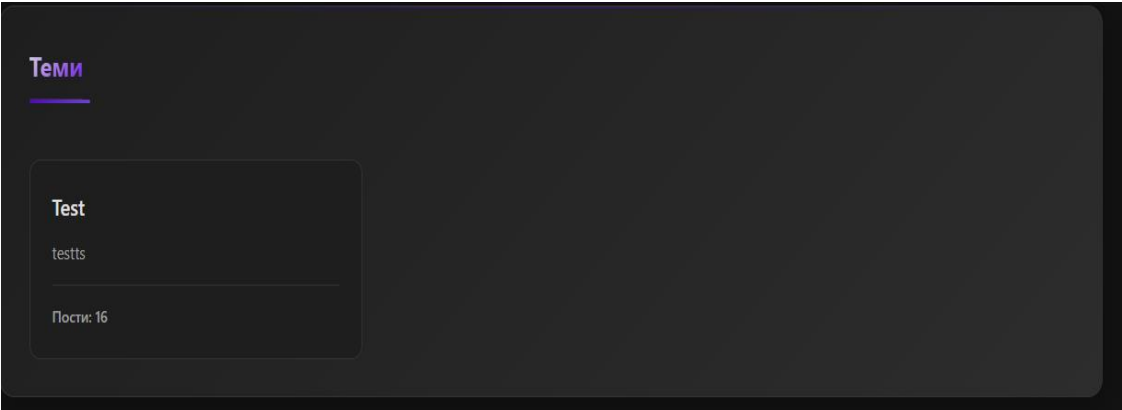


Рис. 3.11 – Блок «Теми»

Нижче розташований блок «Останні пости» (рис. [вставте номер рисунка]), який надає користувачам швидкий огляд найновішої активності на форумі. Кожен запис у списку містить заголовок допису, ім'я автора, категорію, до якої він належить, дату публікації, а також кнопку з лічильником відповідей, що дозволяє одразу перейти до обговорення.

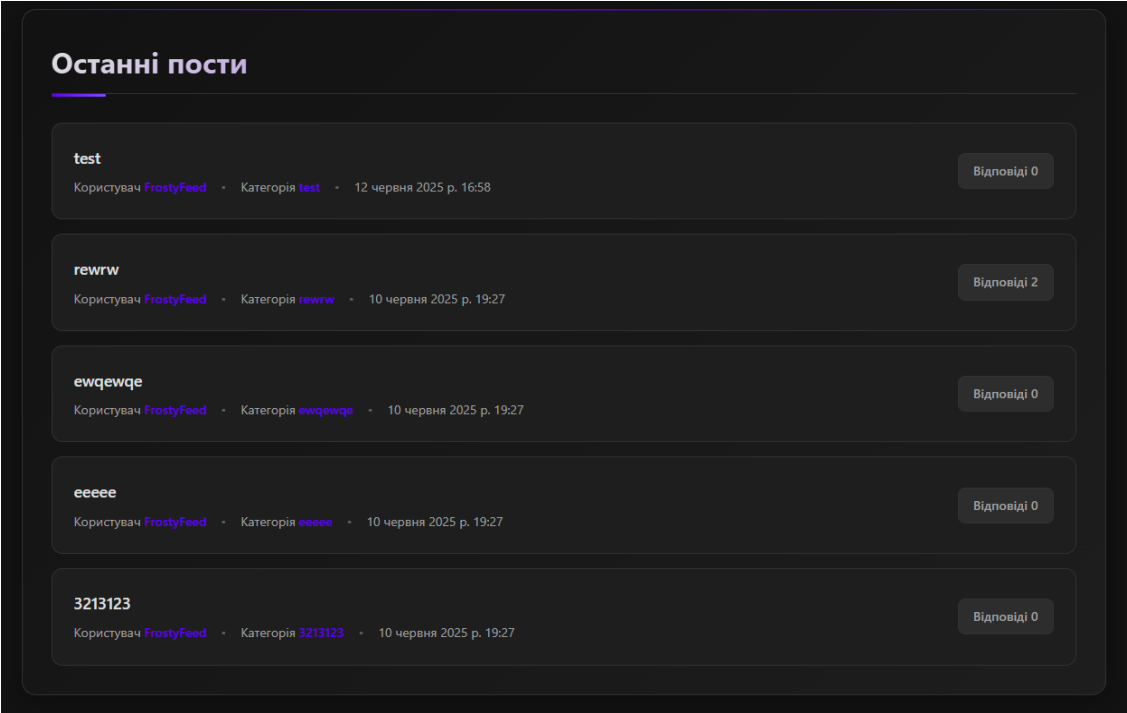


Рис. 3.12 – Розділ «Останні пости»

Завершує сторінку мінімалістичний футер (нижній колонтитул) (рис. 3.13) Він містить інформацію про авторські права: «© 2025 DarkForum. Всі права захищені».



Рис. 3.13 – Нижній колонтитул

Для того, щоб повною мірою використовувати функціонал вебфоруму, зокрема створювати теми та дописи, користувачу необхідно увійти на сайт. Якщо у користувача немає облікового запису, він може зареєструватися, натиснувши відповідну кнопку в навігаційному меню.

Процес реєстрації (рис. 3.14) передбачає заповнення детальної форми. Користувачу необхідно ввести унікальне ім'я користувача, яке буде відображатися на форумі, дотримуючись вимог до довжини від 3 до 20 символів. Далі вказується дійсна електронна адреса, на яку буде надіслано лист для підтвердження акаунту. Також є можливість завантажити зображення профілю (аватар) з обмеженнями щодо формату та розміру файлу. Останнім кроком є введення та підтвердження пароля у відповідних полях. Система надає підказки щодо вимог до пароля, а для зручності біля полів є кнопка «Показати», яка тимчасово відображає введені символи.

Реєстрація

Ім'я користувача

 Виберіть унікальне ім'я користувача довжиною від 3 до 20 символів

Електронна адреса

 Ми надішлемо підтвердження на цю електронну адресу

Зображення профілю

 Зображення не вибрано
 Вибрати зображення
 Завантажте фото профілю (JPG, PNG, макс. 5 МБ)

Пароль

 Показати
 Використовуйте принаймні 8 символів, тільки літери та цифри (без символів)

Підтвердити пароль

 Показати

Зареєструватися

Вже маєте акаунт? [Увійти](#)

Рис. 3.14 – Форма реєстрації

Після заповнення всіх необхідних полів користувач натискає кнопку «Зареєструватися» для завершення процесу. На цьому етапі система проводить валідацію введених даних. Якщо користувач введе невідповідний пароль, використає вже зареєстровану електронну адресу або ім'я користувача, система виведе відповідне повідомлення про помилку біля проблемного поля. Користувач зможе успішно завершити реєстрацію тільки після виправлення всіх помилок. Після реєстрації на вказану пошту буде надіслано лист для верифікації акаунту. Внизу форми також є посилання «Вже маєте акаунт? Увійти» для швидкого переходу на сторінку авторизації.

Після успішної реєстрації та верифікації акаунту користувач може увійти на сайт. Для цього необхідно перейти на сторінку «Увійдіть у свій акаунт» (рис. 3.15) та заповнити форму, вказавши свою адресу електронної пошти та пароль. Для зручності користувача, біля поля для пароля розташована кнопка «Показати», яка дозволяє тимчасово переглянути введені символи. Якщо користувач не бажає вводити свої дані щоразу при вході, він може поставити галочку біля опції «Запам'ятати мене». Це дозволить системі зберегти сесію на тривалий час, і при наступних візитах вхід буде виконано автоматично. Внизу форми також є посилання для переходу на сторінку реєстрації.

Рис. 3.15 – Форма авторизації

Іноді користувачі забувають свої паролі, і для таких випадків передбачено безпечний механізм відновлення доступу. Натиснувши на посилання «Забули пароль?» на сторінці входу, користувач переходить на сторінку «Скинути Ваш Пароль» (рис. 3.16). Тут необхідно ввести електронну адресу, яка була використана при реєстрації, і натиснути кнопку «Надіслати посилання для скидання». Важливим аспектом безпеки є те, що система не повідомляє, чи існує акаунт з такою поштою. Замість цього, після натискання кнопки, на сторінці з'являється уніфіковане повідомлення, що інструкції були надіслані, якщо такий акаунт існує.

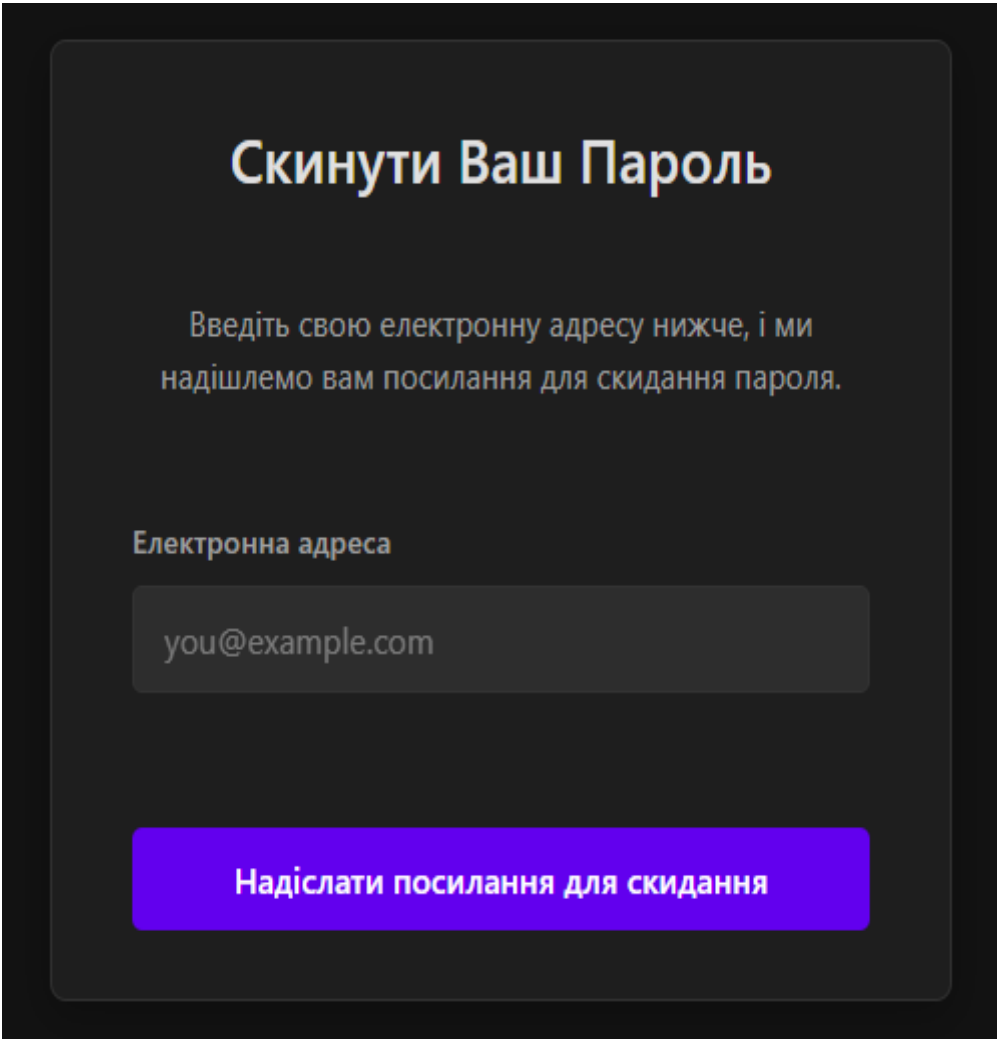


Рис. 3.16 – Форма скидання пароля

Після цього на вказану електронну адресу надсилається лист (рис. [вставте номер рисунка]), який містить унікальне та обмежене в часі посилання для скидання пароля. Це посилання містить довгий, криптографічно згенерований

Скидання пароля

From  vlad.negerey1@gmail.com

To vlad.negerey1@protonmail.com

     ...

[http://localhost:5173/user/password-reset/
eY3hbGciOiJIUzI1NiIsbnVlZ2dtIiwiaWF0IjoxNzUwMDA0ODM1CjleHAoIjE3NTcyMTg0WzV9.Gda1NFVaNVtrfXINBao1VF3D50t2Vcrr4IResYT3fk](http://localhost:5173/user/password-reset/eY3hbGciOiJIUzI1NiIsbnVlZ2dtIiwiaWF0IjoxNzUwMDA0ODM1CjleHAoIjE3NTcyMTg0WzV9.Gda1NFVaNVtrfXINBao1VF3D50t2Vcrr4IResYT3fk)

Перейшовши за посиланням з листа, користувач потрапляє на сторінку «Встановіть Ваш Новий Пароль» (рис. [вставте номер рисунка]). Тут йому необхідно ввести новий пароль та підтвердити його. Після успішного введення та підтвердження, пароль для його акаунту буде оновлено, а використане посилання для скидання стане недійсним, що гарантує безпеку процесу.

Встановіть Ваш Новий Пароль

Новий Пароль

Показати

Повинен містити щонайменше 8 символів. Тільки літери та цифри.

Підтвердьте Новий Пароль

Показати

Встановити Новий Пароль

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Після авторизації користувач отримує повний доступ до функціоналу форуму. Однією з основних сторінок є сторінка категорії (теми), яка містить список всіх обговорень у її межах (рис. 3.19). У верхній частині сторінки відображається назва категорії та кнопка «Створити пост», яка дозволяє користувачам ініціювати нове обговорення. Нижче розташований інструмент для фільтрації, що дозволяє сортувати пости, наприклад, «Спочатку новіші», для зручності перегляду. Кожен пост у списку представлений у вигляді картки, що містить заголовок, ім'я автора, дату публікації та лічильник коментарів, який вказує на активність дискусії.

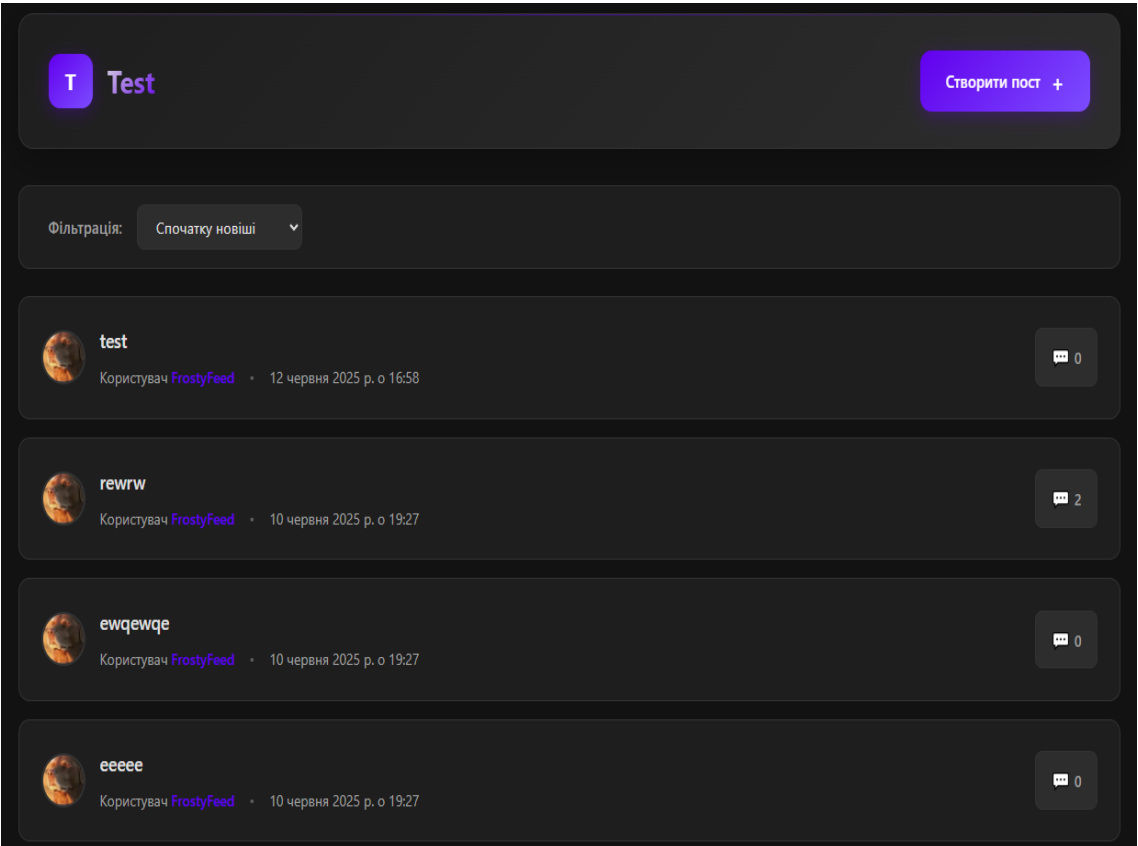


Рис. 3.19 – Сторінка теми

При натисканні на кнопку «Створити пост» з'являється модальне вікно (рис. 3.20), яке містить форму для створення нової теми. Користувачу необхідно ввести заголовок та основний текст повідомлення. Для форматування тексту надається простий візуальний редактор з базовими опціями: жирний шрифт, курсив, підкреслення, створення списків, додавання посилань та зображень. Після заповнення форми, користувач може натиснути кнопку «Створити», щоб опублікувати новий пост, або «Скасувати», щоб закрити вікно.

Рис. 3.20 – Модальне вікно

При переході на сторінку конкретного поста (рис. 3.21), користувач бачить детальну інформацію про нього. У верхній частині сторінки відображається заголовок поста, а також інформаційні блоки з кількістю коментарів, датою створення та датою останнього коментаря. Нижче представлений основний вміст поста, інформація про автора та дата публікації. Поруч із постом розташована кнопка «Поскаржитись», яка дозволяє повідомити адміністрацію про порушення правил.

Під основним постом знаходиться форма «Відповісти на пост». Вона має такий самий візуальний редактор, як і при створенні теми, що дозволяє користувачам зручно формувати свої відповіді. Після введення тексту відповіді, користувач натискає кнопку «Створити», щоб додати свій коментар до обговорення.

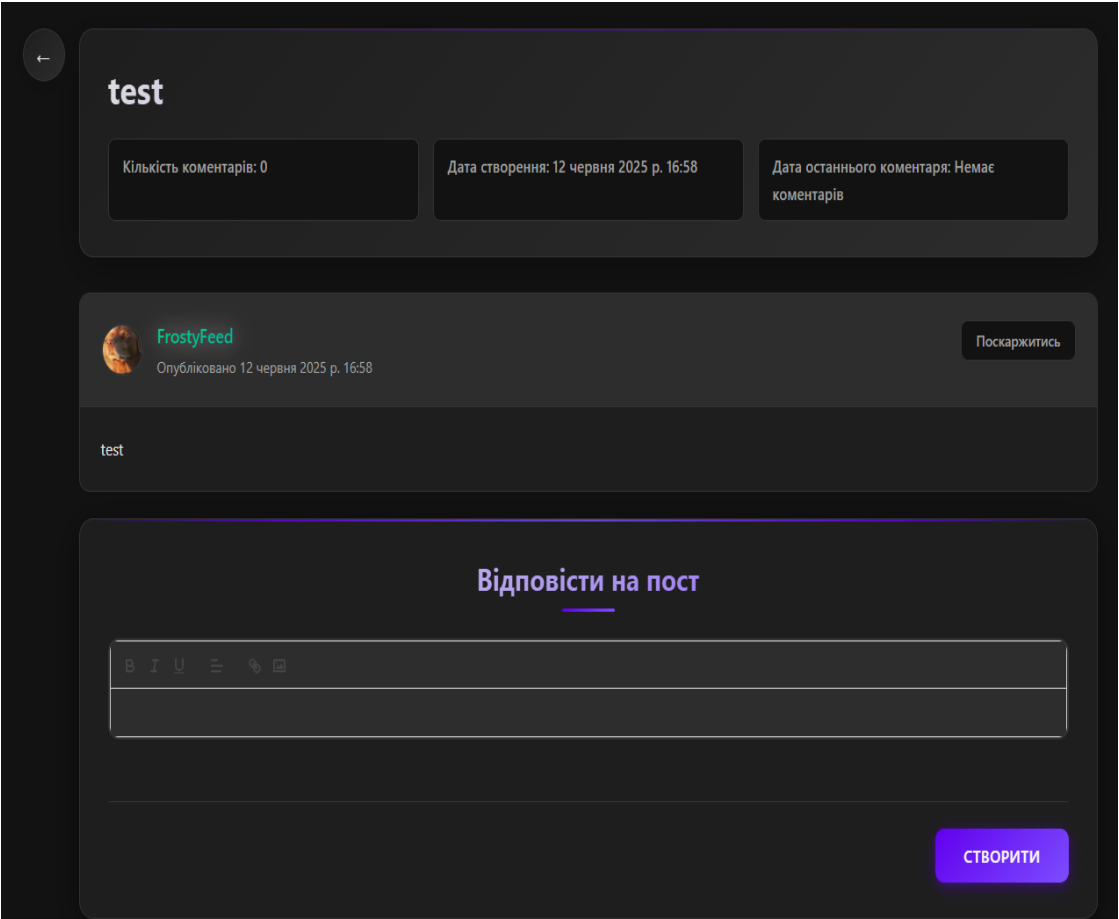


Рис. 3.21 – Сторінка допису

Нижче форми розташований список всіх відповідей на даний пост (рис. 3.22). Кожна відповідь містить аватар та ім'я автора, дату публікації та текст самого повідомлення. Поруч з кожною відповіддю є кнопка для оцінки (лайка) з лічильником, що відображає поточну кількість голосів, а також кнопка «Поскаржитись» для модерації.

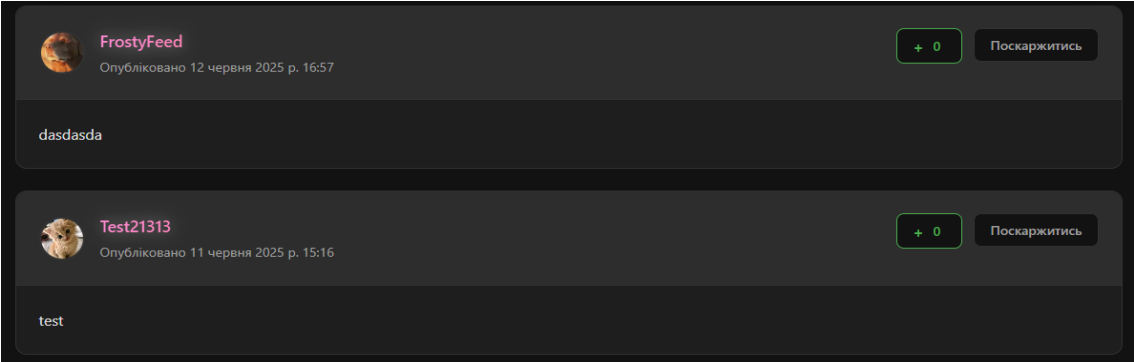


Рис. 3.22 – Список відповідей до допису

Важливою частиною гейміфікації та заохочення активності користувачів є сторінка «Таблиця лідерів» (рис. 3.23). Ця сторінка призначена для відображення

рейтингів найактивніших та найавторитетніших учасників форуму на основі їхньої репутації, яка накопичується завдяки лайкам на їхніх дописах.

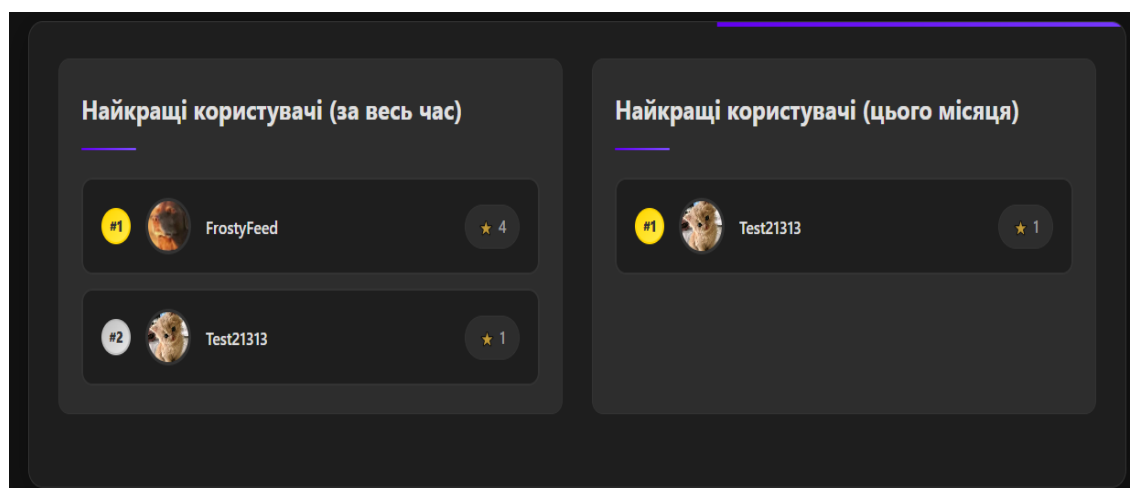


Рис. 3.23 – Таблиця лідерів

Сторінка розділена на два основні блоки. Зліва розташований рейтинг «Найкращі користувачі (за весь час)», який показує загальний топ учасників за всю історію існування форуму. Справа знаходиться рейтинг «Найкращі користувачі (цього місяця)», який відображає лідерів за поточний календарний місяць. Це дозволяє новим користувачам також змагатися за перші місця та отримувати визнання.

Кожен запис у рейтингу містить позицію користувача (#1, #2 і т.д.), його аватар, ім'я та загальну кількість заробленої репутації (лайків). Такий формат дозволяє швидко ідентифікувати лідерів спільноти.

Сторінка є інтерактивною. При натисканні на будь-якого користувача у списку, нижче з'являється додатковий блок «Найкращі відповіді за весь час від [Ім'я користувача]» (рис. 3.24). Цей блок динамічно завантажує та відображає список найпопулярніших відповідей обраного користувача. Кожен запис у цьому списку містить текст відповіді, кількість отриманих вподобань та назву теми, в якій було залишено коментар. Якщо відповідь була видалена модератором, замість тексту відображається відповідне повідомлення. Це дозволяє іншим користувачам ознайомитися з найціннішим внеском лідерів та зрозуміти, за що вони отримали свою репутацію. Користувач може закрити цей додатковий блок, натиснувши на іконку "х".

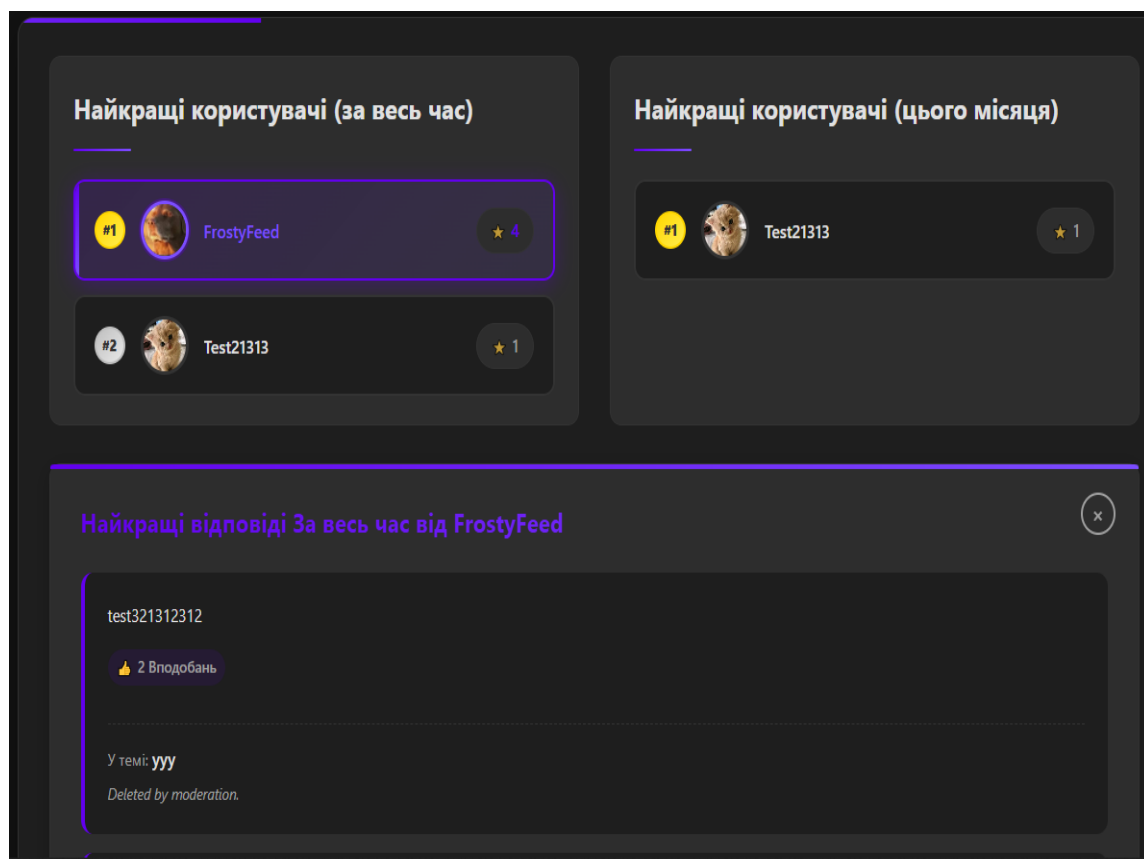


Рис. 3.24 – Блок найкращих відповідей користувача

Така реалізація таблиці лідерів не тільки стимулює здорову конкуренцію, але й слугує інструментом для пошуку якісного контенту та визнання авторитетних учасників форуму.

Для аналізу та моніторингу активності на форумі розроблено сторінку «Статистика» (рис. 3.25), яка надає візуальне представлення ключових метрик росту та залученості спільноти. Цей розділ є важливим інструментом для адміністрації для відстеження стану платформи. У верхній частині сторінки розташований інтерактивний випадаючий список «Оберіть рік», який дозволяє фільтрувати дані на графіках для перегляду статистики за конкретний рік, роблячи аналіз гнучким та сфокусованим.

Основна частина сторінки містить два блоки з графіками, що візуалізують різні аспекти діяльності. Лівий блок, «Активність нових користувачів», містить лінійний графік, що демонструє динаміку реєстрацій нових учасників за місяцями протягом обраного року. Правий блок, «Активність нових дописів», представляє аналогічний графік, який візуалізує кількість створених тем та дописів, що є ключовим індикатором залученості спільноти.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

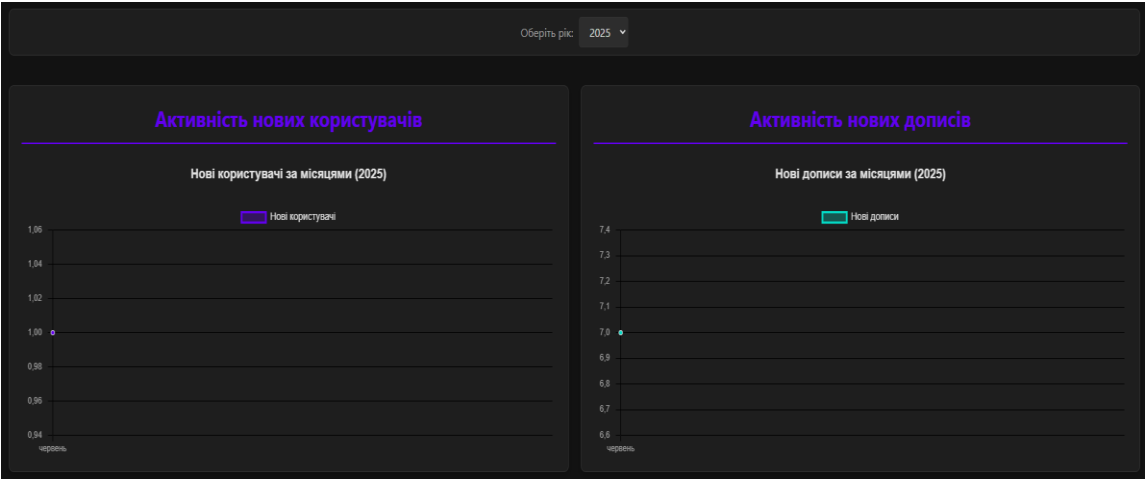


Рис. 3.25 – Сторінка статистики

Таким чином, сторінка «Статистика» слугує важливим аналітичним інструментом для адміністрації форуму, дозволяючи наочно відстежувати тенденції росту, оцінювати ефективність заходів по залученню нових учасників та аналізувати загальну активність спільноти в часі.

Для забезпечення здорової та конструктивної атмосфери на вебфорумі розроблено сторінку «Правила». Цей розділ є фундаментальним для всіх учасників спільноти, оскільки він встановлює норми поведінки та спілкування.

Сторінка (рис. 3.26) має лаконічний та мінімалістичний дизайн, що відповідає загальній темній стилістиці сайту та сприяє легкому сприйняттю текстової інформації. У верхній частині розміщено чіткий заголовок «Правила форуму», який одразу інформує користувача про зміст сторінки. Нижче розташований основний блок контенту, де у доступній та структурованій формі представлено звід правил, що регулюють усі аспекти взаємодії на платформі: від створення контенту до спілкування між користувачами.

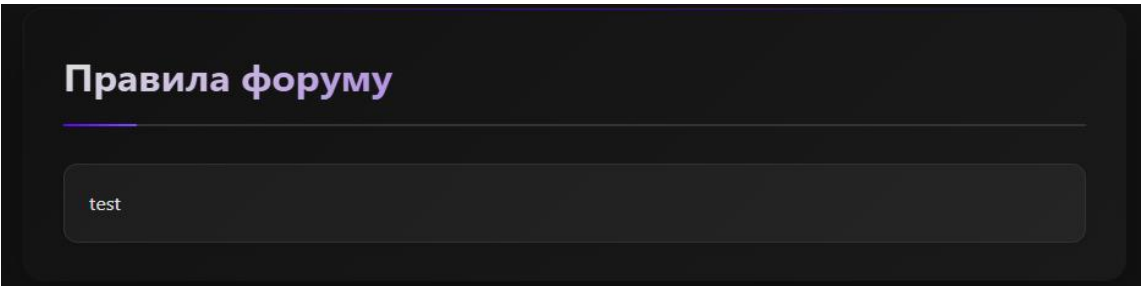


Рис. 3.26 – Сторінка правил

Ознайомлення з правилами є обов'язковою умовою для участі у житті форуму, а посилання на цей розділ доступне з головного навігаційного меню для

швидкого доступу. Ця сторінка слугує основою для роботи системи модерації, оскільки на її положеннях базуються рішення щодо попереджень, блокувань та видалення контенту. Таким чином, сторінка «Правила» виконує критично важливу функцію у підтримці порядку та створенні безпечного і поважного середовища для всіх учасників.

Для інформування спільноти про важливі події, оновлення платформи та інші офіційні повідомлення від адміністрації, на вебфорумі передбачено розділ «Новини» (рис. 3.27). Ця сторінка слугує як офіційний блог або дошка оголошень, забезпечуючи прямий канал комунікації між адміністрацією та користувачами.

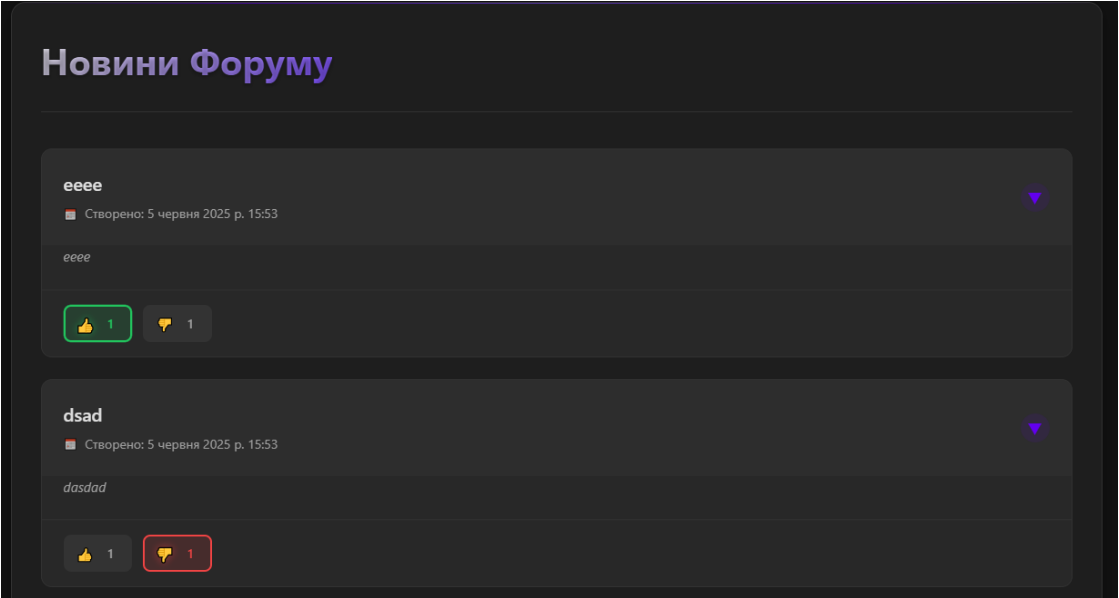


Рис. 3.27 – Сторінка новин форуму

Сторінка має структуру, схожу на акордеон, де кожна новина представлена у вигляді згорнутої картки. У верхній частині сторінки розміщено заголовок «Новини Форуму». Кожна картка новини містить її заголовок, дату створення та іконку-стрілку, що вказує на можливість розгортання для перегляду повного тексту.

При натисканні на картку або іконку-стрілку, вона плавно розгортається (рис. 3.28), відкриваючи повний текст новини. Це дозволяє користувачам ознайомитися з деталями повідомлення, не переходячи на окрему сторінку, що робить перегляд зручним та швидким.

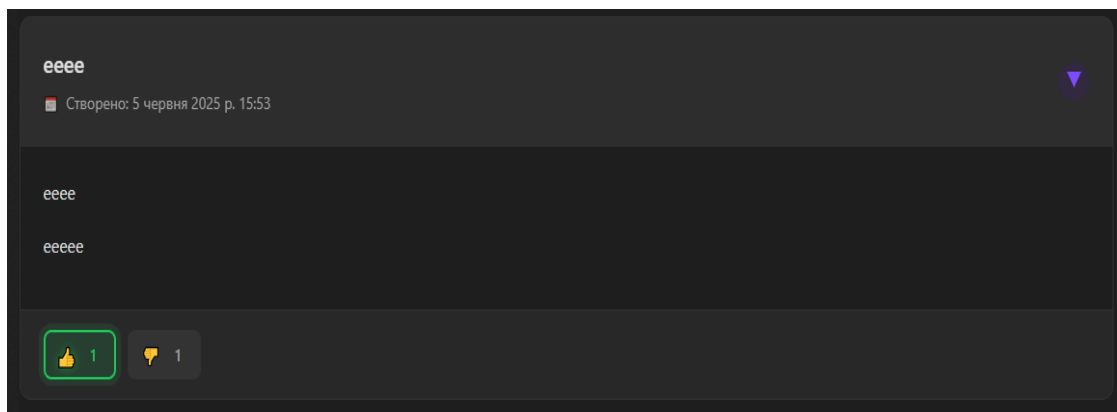


Рис. 3.28 – Розгорнута новина

У нижній частині кожної новини, як у згорнутому, так і в розгорнутому вигляді, розташована система реакцій. Користувачі можуть висловити своє ставлення до новини, натиснувши на кнопки «лайк» (великий палець вгору) або «дизлайк» (великий палець вниз). Поруч з кожною кнопкою відображається лічильник, який показує поточну кількість відповідних реакцій. Це дозволяє адміністрації отримувати зворотний зв'язок від спільноти щодо опублікованих новин та оновлень.

Така інтерактивна та інформативна сторінка новин є важливим інструментом для підтримки прозорості та залучення користувачів до життя форуму.

Профіль користувача є центральною сторінкою, що представляє кожного учасника спільноти. Він надає вичерпну інформацію про діяльність, статус та історію користувача на форумі (рис. 3.29). Сторінка має двоколонну структуру для зручного та логічного розподілу інформації.

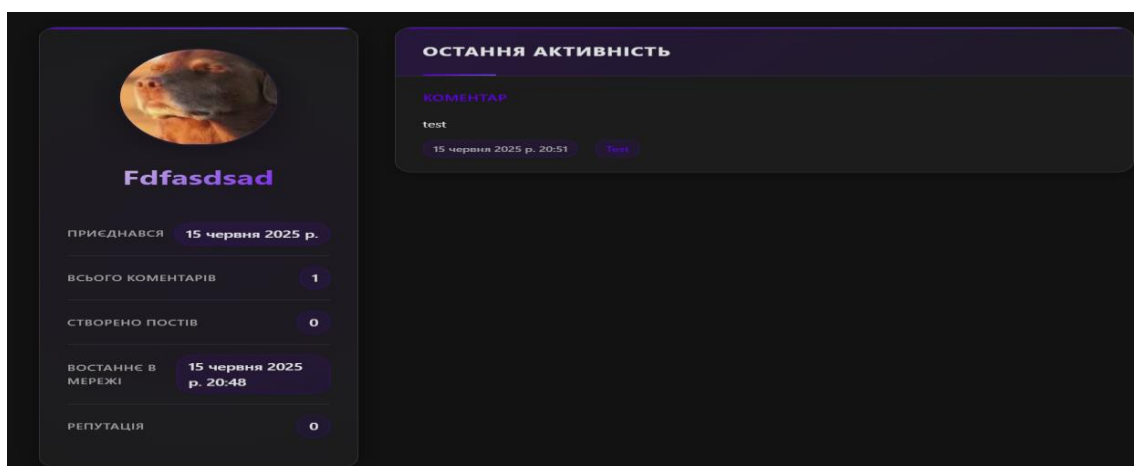


Рис. 3.29 – Профіль користувача

Ліва колонка є інформаційною карткою користувача. У верхній частині розташований великий круглий аватар, який є візуальним представленням учасника. Під ним великим шрифтом відображається його унікальне ім'я (нікнейм). Нижче наведено ключову статистичну інформацію, яка дозволяє швидко оцінити внесок користувача у життя форуму. Сюди входять дата реєстрації («Приєднався»), загальна кількість залишених коментарів, кількість створених ним тем («Створено постів»), час останньої активності на сайті («Востаннє в мережі»), а також загальний рівень репутації, який накопичується за рахунок оцінок інших учасників. Цей блок надає повне уявлення про історію та статус користувача.

Права колонка призначена для відображення останньої активності користувача в реальному часі. Під заголовком «Остання активність» формується стрічка дій, які учасник нещодавно виконав на форумі. Кожен запис у стрічці містить тип дії (наприклад, «Коментар»), короткий уривок тексту, точну дату та час дії. Крім того, поруч із записом є інтерактивне посилання-кнопка (наприклад, «Test»), яка веде безпосередньо до теми, де була здійснена ця дія.

Така організація сторінки дозволяє іншим учасникам не тільки отримати загальну інформацію про користувача, але й бачити його поточний внесок у життя форуму, легко переходити до актуальних обговорень та взаємодіяти з ним.

Для забезпечення повного контролю над своїми даними, кожен користувач має доступ до сторінки «Налаштування акаунту». Ця сторінка є центральним вузлом для управління особистою інформацією, безпекою та налаштуваннями профілю. Вона структурована у вигляді окремих блоків, кожен з яких відповідає за певну функцію.

Перший блок, «Оновити Ваш Нікнейм» (рис. 3.30), дозволяє користувачу змінити своє відображуване ім'я на форумі. Поле для введення вже містить поточний нікнейм, а під ним є підказка щодо вимог до довжини. Після введення нового імені, користувач натискає кнопку «Оновити нікнейм» для збереження змін.

Рис. 3.30 – Блок «Оновити нікнейм»

Наступний блок, «Оновити Зображення Профілю» (рис. 3.31), призначений для зміни аватара. На сторінці відображається поточне зображення профілю, а під ним кнопка «Вибрати нове зображення», яка відкриває файловий діалог для завантаження нового файлу. Система надає інформацію про дозволені формати та максимальний розмір файлу. Після вибору нового зображення користувач натискає кнопку «Оновити зображення профілю».

Рис. 3.31 – Блок «Оновити зображення профілю»

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Третій блок, «Змінити Ваш Пароль» (рис. 3.32), забезпечує безпечний спосіб зміни пароля. Замість того, щоб вимагати введення старого пароля, система пропонує натиснути кнопку «Надіслати посилання для скидання пароля». Це ініціює такий самий процес, як і при відновленні забутого пароля: на зареєстровану пошту надсилається унікальне посилання, перейшовши за яким, користувач може встановити новий пароль.

Рис. 3.32 – Блок «Змінити пароль»

Останній і найбільш критичний блок — «Видалити Ваш Акаунт» (рис. 3.33).

Рис. 3.33 – Блок «Видалити акаунт»

Ця секція оформлена з особливою увагою до безпеки, щоб запобігти випадковому видаленню. На видному місці розташоване попередження про те, що дія є безповоротною і всі дані будуть назавжди видалені. Для підтвердження операції користувачу необхідно ввести свій поточний пароль, а також поставити галочку в полі «Я розумію, що ця дія є безповоротною і мій акаунт не можна буде відновити». Тільки після виконання цих дій кнопка «Видалити мій акаунт», яка для привернення уваги має червоний колір, стає активною. Це забезпечує багаторівневий захист від випадкових чи зловмисних дій.

Для ефективного управління спільнотою та забезпечення дотримання правил, адміністратори мають доступ до спеціалізованої сторінки управління скаргами. Цей розділ є ключовим інструментом модерації, що дозволяє розглядати повідомлення від користувачів про порушення та приймати відповідні заходи. На сторінці відображається список всіх отриманих скарг. У верхній частині розташована пагінація з кнопками «Попередній» та «Наступний» для зручної навігації. Основна частина сторінки присвячена детальному розгляду однієї скарги (рис. 3.34).

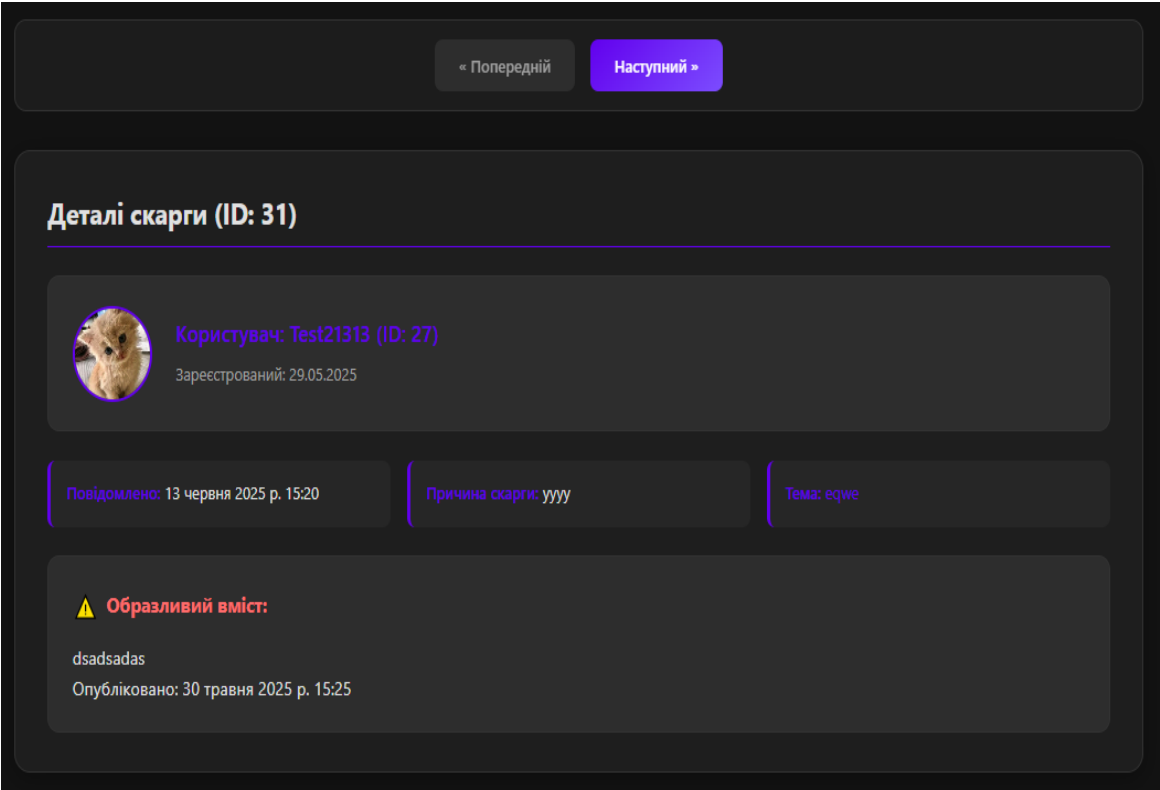


Рис. 3.34 – Основна частина сторінки

Блок «Деталі скарги» надає вичерпну інформацію про інцидент, включаючи унікальний ID скарги, дані про користувача, на якого поскаржились, дату та причину скарги, а також посилання на тему порушення. Нижче у блоці «Образливий вміст» наведено точний текст допису, що дозволяє адміністратору одразу оцінити контекст.

Під деталями скарги розташований блок «Дії адміністратора» (рис. 3.35), який надає інструменти для реагування. Адміністратор може обрати опцію видалення контенту, ввести причину та тривалість бану. На основі аналізу, адміністратор приймає одне з двох рішень: кнопка «Забанити користувача», яка застосовує санкції, або кнопка «Відхилити скаргу», яка закриває її без наслідків.

Рис. 3.35 – Блок «Дії адміністратора»

Така структура сторінки забезпечує адміністраторам всю необхідну інформацію та інструменти для прийняття обґрунтованих та ефективних рішень, що є критично важливим для підтримки порядку та здорової атмосфери на форумі.

3.4. Тестування роботи програмного продукту

Функціональне тестування допомагає перевірити систему на відповідність функціональним вимогам, воно гарантує, що програма належним чином

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

задовольняє вимоги. Мета функціонального тестування полягає у перевірці кожної функції програми із очікуваним результатом, це включає тестування вхідних і вихідних даних програмного забезпечення, маніпулювання даними, взаємодію з користувачем і реакцію системи на різні сценарії та умови [30].

У першу чергу необхідно протестувати користувацьку сторону вебфоруму, а саме: працездатність усіх посилань, кнопок, форм для введення даних, можливості входу, реєстрації, відновлення пароля, створення тем та дописів, оцінки дописів (лайки), перегляду статистики, таблиці лідерів та профілів користувачів, а також функціонал налаштувань акаунту, включаючи його видалення.

Таблиця 3.1

Тестування користувацької сторони вебфоруму

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Перевірка форми реєстрації	Ввести коректні та унікальні ім'я, пошту та пароль	Створення нового акаунту, перенаправлення на сторінку входу	Акаунт створено, виконано перенаправлення	Пройдено
2	Перевірка форми авторизації	Ввести коректні пошту та пароль	Успішний вхід, відображення головної сторінки для авторизованого користувача	Успішний вхід в систему	Пройдено
3	Перевірка створення нового поста (теми)	Ввести заголовок та текст у формі створення	Новий пост з'являється у списку відповідної теми	Пост успішно створено та відображено	Пройдено

Продовження таблиці 3.1

4	Перевірка додавання відповіді до поста	Ввести текст у форму відповіді під постом, натиснути "Створити"	Нова відповідь з'являється під постом	Відповідь успішно додано	Пройдено
5	Перевірка функціоналу лайків (upvotes)	Натиснути на кнопку лайка біля відповіді	Лічильник лайків збільшується на 1, іконка змінює вигляд	Лічильник та іконка оновлюються коректно	Пройдено
6	Перевірка процедури відновлення пароля	Ввести пошту, перейти за посиланням з листа, ввести новий пароль	Пароль успішно змінено, можливість увійти з новим паролем	Пароль успішно змінено	Пройдено
7	Перевірка видалення акаунту	Ввести пароль, підтвердити дію, натиснути "Видалити"	Акаунт видаляється, сесія користувача завершується	Акаунт видалено, виконано вихід з системи	Пройдено

Окремо було проведено тестування функціоналу, доступного лише для адміністраторів, а саме: перегляд та обробка скарг від користувачів, а також застосування санкцій, таких як блокування користувачів та видалення контенту.

Таблиця 3.2

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестування адміністративної сторони вебфоруму

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Перевірка відображення сторінки скарг	Перейти на сторінку управління скаргами	Відображається список скарг з детальною інформацією	Список скарг відображається коректно	Пройдено
2	Перевірка відхилення скарги	Натиснути кнопку "Відхилити скаргу" на сторінці скарги	Скарга отримує статус "Відхилено" і більше не відображається як активна	Скарга успішно відхилена	Пройдено
3	Перевірка бану користувача через скаргу	Ввести причину, тривалість бану, натиснути "Забанити користувача"	Користувач отримує бан. При спробі входу користувач бачить повідомлення про бан	Користувача забанено, повідомлення відображається	Пройдено
4	Перевірка видалення контенту через скаргу	Поставити прапорець "Видалити вміст" та застосувати дію (бан/відхилення)	Вміст допису/теми видаляється з форуму	Вміст успішно видалено	Пройдено

Під час проведення тестування було уважно перевірено вебфорум на наявність можливих помилок і недоліків. Завдяки цьому процесу було досягнуто високої якості програмного продукту та його відповідності заявленим вимогам.

Висновки до третього розділу

У третьому розділі кваліфікаційної роботи було проведено комплексний аналіз практичної реалізації розробленого вебфоруму. Детально проаналізовано та описано структуру клієнтської та серверної частин додатку, побудованих на стеку технологій React та Spring Boot, що підкреслює сучасний підхід до веб-розробки. Було визначено основні елементи користувацького інтерфейсу та їх функціональні можливості, що охоплюють увесь життєвий цикл взаємодії користувача з платформою.

Проведено послідовний аналіз використання програмного додатку, в рамках якого детально описано роботу з ключовими сторінками: від головної сторінки з оглядом останньої активності, сторінок реєстрації та авторизації з реалізованими механізмами безпеки, до сторінок створення тем і дописів з візуальним редактором. Окрему увагу приділено інтерактивним елементам, таким як таблиця лідерів з динамічним відображенням статистики, сторінка налаштувань акаунту з функціями зміни даних та безпечного видалення профілю. Також детально описано роботу з адміністративною панеллю для управління скаргами.

Ключовим етапом цього розділу стало проведення функціонального тестування розробленого програмного забезпечення з метою перевірки його працездатності, стабільності та відповідності функціональним вимогам, визначеним у другому розділі. В рамках тестування були ретельно перевірені як панель адміністратора, так і користувацька частина вебфоруму. Тестові сценарії охоплювали всі основні функції: від валідації форм та роботи REST API ендпоінтів до коректного оновлення UI після асинхронних запитів. Результати тестування, задокументовані у відповідних таблицях, підтвердили високу якість розробленого програмного забезпечення, відсутність критичних помилок та його здатність ефективно функціонувати у визначених умовах, що свідчить про успішне завершення етапу реалізації та готовність продукту до розгортання.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено вебфорум «DarkForum», спрямований на створення інтерактивної та сучасної платформи для онлайн-обговорень. Основні цілі та завдання проекту полягали у створенні вебдодатку, який би поєднував у собі передові технології та найкращі практики розробки програмного забезпечення. Аналіз існуючих рішень та сучасних архітектурних підходів надав змогу обрати оптимальний стек технологій, а саме: React для створення динамічного та гнучкого клієнтського інтерфейсу, та Spring Boot (Java) для розробки надійного, безпечного та масштабованого серверного REST API.

Детальний аналіз вимог до програмного продукту був здійснений з метою точного визначення функціональних (реєстрація, створення тем, система репутації, модерація) та нефункціональних (безпека, продуктивність) вимог. На основі цього аналізу було спроектовано логічну модель бази даних та архітектуру REST API, що чітко відображає взаємодію різних компонентів системи.

Особлива увага була приділена розробці ключових аспектів функціонування системи: реалізації безпечної аутентифікації та авторизації з використанням JWT, ефективної взаємодії з базою даних через Spring Data JPA, а також створенню динамічного користувацького інтерфейсу. У третьому розділі роботи було детально описано інтерфейс вебфоруму, визначено основні елементи і їх функціональні можливості, що дозволило користувачам зручно та ефективно взаємодіяти з додатком.

Проведене функціональне тестування підтвердило високу якість програмного забезпечення і його відповідність усім поставленим вимогам.

Отже, у результаті виконання кваліфікаційної роботи було реалізовано повноцінний вебфорум, що відповідає всім встановленим вимогам, демонструє високий рівень функціональності, безпеки та користувацького досвіду, та є готовим до розгортання та подальшого розвитку.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Learning React: Modern Patterns for Developing React Apps, 2nd Edition / Alex Banks, Eve Porcello, 2020 – 352 p.
2. Full-Stack React, TypeScript, and Node / Adam Freeman, 2021 – 854 p.
3. JavaScript: The Definitive Guide, 7th Edition / David Flanagan, 2020 – 704 p.
4. Eloquent JavaScript, 3rd Edition / Marijn Haverbeke, 2018 – 472 p.
5. CSS: The Definitive Guide, 5th Edition / Eric A. Meyer, Estelle Weyl, 2023 – 982 p.
6. React.js Official Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>
7. MDN Web Docs: JavaScript Guide [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
8. React Router Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://reactrouter.com/>
9. Axios NPM Package [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/axios>
10. Zod: TypeScript-first schema validation [Електронний ресурс] – Режим доступу до ресурсу: <https://zod.dev/>
11. Spring in Action, 6th Edition / Craig Walls, 2022 – 528 p.
12. Pro Spring Boot 3 / Felipe Gutierrez, 2023 – 675 p.
13. Effective Java, 3rd Edition / Joshua Bloch, 2018 – 416 p.
14. Java: The Complete Reference, 12th Edition / Herbert Schildt, 2021 – 1344 p.
15. Spring Boot: Up and Running / Mark Heckler, 2021 – 312 p.
16. Java Persistence with Hibernate, 2nd Edition / Christian Bauer, Gavin King, Gary Gregory, 2016 – 656 p.
17. Spring Framework Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://spring.io/projects/spring-framework>
18. Baeldung: Spring Security [Електронний ресурс] – Режим доступу до ресурсу: <https://www.baeldung.com/security-spring>

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

19. Oracle Java Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.oracle.com/en/java/>
20. JSON Web Tokens (JWT) Introduction [Електронний ресурс] – Режим доступу до ресурсу: <https://jwt.io/introduction/>
21. MySQL Crash Course / Rick Silva, 2022 – 288 p.
22. SQL in 10 Minutes, a Day, 5th Edition / Ben Forta, 2020 – 288 p.
23. MySQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>
24. W3Schools SQL Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3schools.com/sql/>
25. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert C. Martin, 2017 – 432 p.
26. RESTful Web Services / Leonard Richardson, Sam Ruby, 2007 – 454 p.
27. Design Patterns: Elements of Reusable Object-Oriented Software / Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 1994 – 416 p.
28. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition / Martin Fowler, 2003 – 208 p.
29. MVC (Model-View-Controller) Architecture Pattern [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/mvc-framework-introduction/>
30. REST API Design Best Practices [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design>
31. Software Functional Testing [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ibm.com/topics/functional-testing>

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

Модель «User»

```

@Entity
@Table(name = "users")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@EntityListeners({StatsEntityListener.class})
public class User implements HasStatsType {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column(name = "username")
    private String username;
    @Column(name = "password")
    private String password;
    @Column(name = "register_date")
    private LocalDate registerDate;
    @Column(name = "reputation")
    private int reputation = 0;
    @Column(name = "last_seen")
    private OffsetDateTime lastSeen;
    @Column(name = "email")
    private String email;
    @Column(name = "avatar_url")
    private String avatarUrl;
    @Enumerated(EnumType.STRING)
    private Status status;
    @OneToMany(mappedBy = "user")
    private List<Threads> threads = new ArrayList<>();
    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval =
true)
    private List<Report> reports = new ArrayList<>();
    @OneToMany(mappedBy = "user")
    private List<Reply> replies = new ArrayList<>();
    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(
        name = "user_roles",
        joinColumns = {@JoinColumn(name = "user_id")},
        inverseJoinColumns = {@JoinColumn(name = "role_id")}
    )
    private Set<Roles> roles = new HashSet<>();
    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(
        name="upvotes",
        joinColumns = {@JoinColumn(name = "user_id")},
        inverseJoinColumns = {@JoinColumn(name = "reply_id")}
    )
    private Set<Reply> upvotes;
    @Formula("(SELECT COUNT(*) FROM reply r WHERE r.user_id=id)")
    private int replyCount;
    @Formula("(SELECT COUNT(*) FROM threads t WHERE t.user_id=id)")
    private int threadCount;
    @OneToMany(mappedBy = "bannedUser", fetch = FetchType.EAGER)
    private List<Ban> bans = new ArrayList<>();
    @Transient
    private final StatsType statsType = StatsType.USER;

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@PrePersist
protected void onCreate(){
    registerDate=LocalDate.now();
}
public User(RegisterRequestDTO registerRequestDAO, String avatarUrl, String
role){
    this.password = registerRequestDAO.getPassword();
    this.roles = new HashSet<>();
    this.roles.add(new Roles(1,role));
    this.email = registerRequestDAO.getEmail();
    this.username = registerRequestDAO.getUsername();
    this.avatarUrl = avatarUrl;
    this.lastSeen = OffsetDateTime.now();
    this.status = Status.UNVERIFIED;
}
}

```

Модель «Reply»

```

@Entity
@Table(name = "reply")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@EntityListeners({CloudCleanupEntityListener.class, StatsEntityListener.class})
public class Reply implements HasCloudResource, Post, HasStatsType,
HasLatestActivity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column(name = "isEdited")
    private boolean isEdited;
    @Column(name = "content")
    private String content;
    @Column(name = "creation_date")
    private OffsetDateTime creationDate;
    @ManyToOne
    @JoinColumn(name = "thread_id")
    private Threads thread;
    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;
    @ManyToMany(mappedBy = "upvotes")
    private Set<User> upvotedBy = new HashSet<>();
    @Formula("(SELECT COUNT(*) FROM upvotes u WHERE u.reply_id = id)")
    private int upvoteCount;
    @OneToMany(mappedBy = "reply", orphanRemoval = true, cascade =
CascadeType.ALL)
    private List<Report> reports = new ArrayList<>();
    @Transient
    private final StatsType statsType = StatsType.POST;

    public Reply(String content, OffsetDateTime dateTime, Threads thread, User
user){
        this.content = content;
        creationDate = dateTime;
        this.thread = thread;
        this.user = user;
    }

    @Override
    public LatestActivityDTO toLatestActivityDTO() {
        return new

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```
LatestActivityDTO(this.getId(),this.getContent(),this.getThread().getTopic().getId(),
this.getThread().getTopic().getTittle(),"Коментар",this.getCreationDate(),this.getThread().getId());
    }
}
```

Модель «Report»

```
@Entity
@Table(name = "reports")
@Data
@NoArgsConstructor
@Builder
@AllArgsConstructor
@EntityListeners({StatsEntityListener.class})
public class Report implements HasStatsType {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column(name = "creation_date")
    private OffsetDateTime creationDate;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "thread_id")
    private Threads thread;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", nullable = false)
    private User user;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "reply_id")
    private Reply reply;
    @Column(name = "reason")
    private String reason;
    @Column(name = "status")
    @Enumerated(EnumType.STRING)
    private ReportStatus status;
    @Transient
    private final StatsType statsType = StatsType.REPORT;
}
```

Контролер «UserController»

```

@RestController
@RequestMapping("/api")
@CrossOrigin(origins = "http://localhost:5173", allowedHeaders = "*",
allowCredentials = "true", methods = {RequestMethod.GET, RequestMethod.POST,
RequestMethod.PUT, RequestMethod.DELETE, RequestMethod.PATCH})
@RequiredArgsConstructor
public class UserController {
    private final UserService userService;
    private final CloudStorageService cloudStorageService;
    @GetMapping("/user/{id}/latest")
    public ResponseEntity<List<LatestActivityDTO>>
getLatestActivity(@PathVariable int id){
        return ResponseEntity.ok(userService.getLatestActivity(id));
    }
    @PostMapping("/user/register")
    public ResponseEntity<String> register(@RequestBody @Valid
RegisterRequestDTO registerRequestDAO){
        return ResponseEntity.ok(userService.registerUser(registerRequestDAO));
    }
    @PostMapping("/user/avatar")
    public ResponseEntity<?> setAndUploadAvatar(@ModelAttribute UploadAvatarDTO
uploadAvatarDTO){
        try{
            cloudStorageService.upload(uploadAvatarDTO.email(),
uploadAvatarDTO.avatar().getBytes());
        } catch (IOException e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
.body("Error reading file before processing.");
        }
        return ResponseEntity.accepted().body("Uploading avatar");
    }
    @GetMapping("/user/email/{token}")
    public ResponseEntity<String> getEmailFromToken(@PathVariable String token){
        return ResponseEntity.ok(userService.getEmailFromToken(token));
    }
    @PostMapping("/user/verification")
    public ResponseEntity<String> verifyUser(@RequestBody @Valid TokenDTO
token){
        return ResponseEntity.ok(userService.verifyUser(token));
    }
    @GetMapping("/user/roles/{email}")
    public ResponseEntity<Set<Roles>> getUserRoles(@PathVariable String email){
        return ResponseEntity.ok(userService.getUserRoles(email));
    }
    @GetMapping("/user/{id}")
    public ResponseEntity<UserDataDTO> getUserInfo(@PathVariable int id){
        return ResponseEntity.ok(userService.getUserInfo(id));
    }
    @PatchMapping("/user/nickname")
    public ResponseEntity<String> updateNickname(@RequestBody
NicknamePatchRequestDTO requestDTO){
        return ResponseEntity.ok(userService.updateNickname(requestDTO));
    }
    @PatchMapping("/user/avatar")
    public ResponseEntity<AvatarChangeResponseDTO> updateAvatar(@ModelAttribute
AvatarChangeRequestDTO requestDTO){
        return ResponseEntity.ok(userService.updateAvatar(requestDTO));
    }
    @PostMapping("/user/password-reset-request")

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        public ResponseEntity<String> sendPasswordResetLink(@RequestBody
        PasswordResetRequestDTO requestDTO) {
            return ResponseEntity.ok(userService.sendPasswordRestLink(requestDTO));
        }
        @PatchMapping("/user/password-reset")
        public ResponseEntity<String> updateUserPassword(@RequestBody @Valid
        ChangePasswordRequestDTO requestDTO) {
            return ResponseEntity.ok(userService.updatePassword(requestDTO));
        }
        @DeleteMapping("/user")
        public ResponseEntity<String> deleteUser(@RequestBody DeleteUserRequestDTO
        requestDTO, HttpServletRequest request) {
            return ResponseEntity.ok(userService.deleteUser(requestDTO, request));
        }
    }
}

```

Контролер «ThreadController»

```

@RestController
@RequestMapping("/api")
@CrossOrigin(origins = "http://localhost:5173")
public class ThreadController {
    private final ThreadService threadService;
    @Autowired
    public ThreadController(ThreadService threadService) {
        this.threadService = threadService;
    }
    @GetMapping("/thread/latest")
    public ResponseEntity<List<ThreadDTO>> getLatestThreads() {
        return
        ResponseEntity.of(Optional.ofNullable(threadService.getLastThreads()));
    }
    @GetMapping("/thread/{id}")
    public ThreadDTO getThread(@PathVariable int id) {
        return new ThreadDTO(threadService.getThread(id).get());
    }
    @PostMapping("/thread")
    public ResponseEntity<ThreadDTO> saveThread(@RequestBody POSTThreadDTO
    postThreadDTO) {
        return ResponseEntity.ok(threadService.saveThread(postThreadDTO));
    }
    @GetMapping("/thread")
    public ResponseEntity<?> searchThreads(
        Integer topicId,
        Integer userId,
        int page,
        int size,
        "creationDate,desc") String sort
        ) {
        String[] sortParams = sort.split(",");
        String sortField = sortParams[0];
        Sort sortOrder = sortParams.length > 1 &&
        sortParams[1].equalsIgnoreCase("desc")
            ? Sort.by(sortField).descending()
            : Sort.by(sortField).ascending();
    }
}

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Pageable pageable = PageRequest.of(page, size, sortOrder);

        SliceResponse<ThreadDTO> result = threadService.searchThreads(topicId,
userId, pageable);

        return ResponseEntity.ok(result);
    }
    @PutMapping("/thread")
    public ResponseEntity<ThreadPUTResponseDTO> updateThread(@RequestBody
RecordUpdateDTO recordUpdatedDTO, HttpServletRequest request){
        return
ResponseEntity.ok(threadService.updateThread(recordUpdatedDTO, request));
    }
    @DeleteMapping("/thread/{id}")
    public ResponseEntity<ThreadDeleteResponseDTO> deleteThread(@PathVariable
int id, HttpServletRequest request){
        return ResponseEntity.ok(threadService.deleteThread(id, request));
    }
    @GetMapping("/thread/random")
    public ResponseEntity<String> getRandomThread(){
        return ResponseEntity.ok(threadService.getRandom());
    }
}

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		