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 «Веб технології») на тему:

«Платформа для створення та публікації мемів»

Виконала студентка групи ВТ-21-1
РАДКІВСЬКА Аліна Валеріївна

Керівник роботи
САВІЦЬКИЙ Роман Святославович

Рецензент
ГРОМСЬКИЙ Олександр 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
«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **РАДКІВСЬКА Аліна Валеріївна**
Керівник роботи: **САВІЦЬКИЙ Роман Святославович**
Тема роботи: **«Платформа для створення та публікації мемів»**,
затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**
Вихідні дані для роботи: **розробка та впровадження автоматизованої системи для взаємодії з цифровим контентом за допомогою сучасних інформаційних технологій.**
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	Завдання видав	Завдання прийняв
1	Савіцький Р. С.	17.02.2025	17.02.2025
2	Савіцький Р. С.	19.02.2025	19.02.2025
3	Савіцький Р. С.	21.02.2025	21.02.2025

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновку та списку використаних джерел.

Текстова частина викладена на ХХ сторінках друкованого тексту. Список використаних джерел містить ХХ найменувань і займає ХХ сторінок. В роботі наведено ХХ рисунків та ХХ таблиць. Загальний обсяг роботи – ХХ сторінок.

У роботі поставлені основні завдання на розробку системи для створення та публікації мемів. Система включає модулі для створення кастомних мемів, публікації контенту, управління користувачами, підписками, коментарями, вподобаннями, скаргами, а також модуль рекомендацій. Також було проведено порівняльний аналіз аналогічних платформ. Було обрано клієнт-серверну архітектуру, яка працює у зв'язці з сучасними підходами до веб-розробки. Наведені основні сценарії використання системи, логіка взаємодії між модулями, діаграми даних та алгоритми роботи ключових модулів. Спроєктована база даних з використанням PostgreSQL. Клієнтська частина розроблена за допомогою бібліотека ReactJS із використанням React Query, а серверна частина за допомогою фреймворку NestJS. Для збереження зображень використано UploadThing.

Ключові слова: ПЛАТФОРМА, КОРИСТУВАЧ, БАЗА ДАНИХ, СУТНІСТЬ, МЕМ, МОДЕРАЦІЯ, РЕКОМЕНДАЦІЇ.

					Житомирська політехніка.25.121.21.000 - ПЗ			
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Радківська А.В.*			Пояснювальна записка до випускної кваліфікаційної роботи бакалавра	*Літ.*	*Арк.*	*Аркушів*
Керівник		*Савіцький. Р.С.*					*3*	*ХХ*
						ФІКТ Гр. ВТ-21-1		
Рецензент		*Громський О.О.*						
Зав. каф.		*Вакалюк Т.А*						

ABSTRACT

The explanatory note to the bachelor's final qualification work consists of an introduction, three sections, a conclusion and a list of used sources.

The text part is laid out on XX pages of the printed text. The list of used sources contains 6 names and occupies XX pages. XX images and XX tables are given in the work.

The main tasks for the development of a system for creating and publishing memes are set in the work. The system includes modules for creating custom memes, publishing content, managing users, subscriptions, comments, likes, complaints, recommendations, and a chat module. A comparative analysis of similar platforms was also conducted. A client-server architecture was chosen, which works in conjunction with modern approaches to web development. The main scenarios of using the system, the logic of interaction between modules, data diagrams and algorithms of operation of key modules are given. Designed database using PostgreSQL. The client part is developed using the ReactJS library using React Query, and the server part using the NestJS framework. UploadThing is used to save images.

Keywords: PLATFORM, USER, DATABASE, ESSENCE, MEME, MODERATION, RECOMMENDATIONS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ФОП – фізична особа підприємець

ПЗ – Програмне забезпечення

БД – База даних

API – Application programming interface, дослівно: Інтерфейс прикладного програмування.

JSON – JavaScript Object Notation

JWT – JSON Web Token

RESTFull API – representational state transfer API

JS – Java Script

ВСТУП

У пояснювальній записці до випускної кваліфікаційної роботи бакалавра буде наведено процес розробки платформи для створення та публікації мемів . Система орієнтована на автоматизацію взаємодії між користувачами, публікаціями та модераторами, а також має забезпечити надійну і швидку роботу для великої кількості користувачів.

Актуальність теми: зумовлена зростанням популярності цифрового контенту, зокрема мемів, які стали важливим елементом онлайн-комунікації. Необхідність ефективної взаємодії між користувачами, швидкого обміну контентом, а також якісної модерації публікацій вимагає створення сучасної автоматизованої платформи. Така система сприятиме підвищенню зручності користування, безпеки та ефективності контентного середовища.

Мета роботи: спроєктувати та реалізувати платформу для створення, пошуку та обговорення мемів, що забезпечить швидку взаємодію між користувачами, ефективні інструменти комунікації та якісну систему модерації. Реалізація цієї системи дозволить забезпечити стабільну роботу при великому навантаженні, підвищити комфортність використання та сприяти розвитку інтернет-культури.

Завданням випускної кваліфікаційної роботи бакалавра є виконання таких задач:

- постановка основних задач системи та узагальнення її структури;
- аналіз аналогів;
- побудова архітектури, що буде задовольняти всі вимоги системи;
- вибір та обґрунтування засобів реалізації системи.

Об’єкт дослідження: інформаційні технології для автоматизації взаємодії між користувачами, публікаціями та модераторами, яка включатиме механізми створення та поширення мемів, взаємодії з контентом (коментарі, вподобання), приватного спілкування, а також ефективної модерації. Досліджується можливість оптимізації процесів управління платформою для забезпечення високої продуктивності та безперебійної роботи при великій кількості користувачів.

Предмет дослідження: процес розробки та впровадження автоматизованої системи для взаємодії з цифровим контентом.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано **тезу** на тему **«**Підбір рекомендацій публікацій з використанням content-based та collaborative-based фільтрації» (див. список джерел).

РОЗДІЛ 1. ОСНОВНА ЧАСТИНА

1.1. Визначення та постановка задачі

Основною метою роботи є розробка платформи для створення та публікації мемів, яка забезпечить користувачам можливість швидко та зручно генерувати контент, взаємодіяти з іншими користувачами, а також автоматизувати модерацію та рекомендації. Система спрямована на підвищення якості користувацького досвіду через зручний редактор мемів, систему підписок, сповіщень та рекомендацій.

Для реалізації цієї мети необхідно виконати такі завдання:

1. Розробка архітектури системи:
   - Визначення загальної концепції платформи, її основних функцій та ролей користувачів (user, moderator).
   - Побудова архітектури клієнт-серверної взаємодії на основі сучасних веб-технологій.
   - Проєктування та створення бази даних з використанням PostgreSQL.
2. Розробка функціональних модулів backend частини системи:
   - Модуль користувачів.
   - Модуль авторизації за допомогою JWT-токенів зі змогою авторизуватися за допомогою Google.
   - Модуль публікації мемів та управління контентом.
   - Модуль підписки на акаунти користувачів.
   - Модуль вподобань.
   - Модуль коментування публікацій (включаючи можливість додавання зображення у коментарі).
   - Модуль рекомендацій контенту на основі вподобань користувача.
   - Модуль роботи з і збереженням зображень у віддаленому сховищі.
3. Розробка адміністративного функціоналу:
   - Реалізація модуля скарг на користувачів та публікації.
   - Модуль управління та модерації контенту для модераторів.

- Впровадження механізмів блокування акаунтів та публікацій.

4. Розробка системи сповіщень:
   - Впровадження повідомлень про вподобання, коментарі, підписки.
   - Налаштування можливості отримання push-сповіщень.
5. Розробка додаткових можливостей: генерація мемів за описом за допомогою штучного інтелекту.
6. Тестування та впровадження:
   - Проведення ручного тестування вебзастосунку.
   - Оптимізація продуктивності для забезпечення швидкодії.
   - Аналіз користувацького досвіду та внесення необхідних покращень.

Результатом реалізації поставленого завдання стане багатофункціональна вебплатформа, що дозволить користувачам створювати, публікувати та обговорювати меми, використовуючи сучасні технології фронтенду (Next.js + React Query) та бекенду (NestJS). Система забезпечить зручну взаємодію, ефективне управління контентом, автоматизовану модерацію та рекомендації, що сприятиме створенню активної спільноти навколо платформи.

1.2. Аналіз існуючого ПЗ

Існує багато платформ для створення, поширення та обговорення контенту, включаючи меми. Завдяки розвитку соціальних мереж та онлайн-спільнот, користувачі отримали можливість не лише переглядати, а й активно взаємодіяти з контентом, залишаючи коментарі, ставлячи оцінки та поширюючи вподобані матеріали. Для порівняння ми розглянемо три популярні сервіси: Instagram, Reddit та 9GAG. Ці платформи відрізняються своєю цільовою аудиторією, алгоритмами ранжування контенту, можливостями монетизації та форматом взаємодії між користувачами.

Розглянемо кожен аналог ближче.

Розпочнімо із соцмережі Instagram (рис. 1.1.). Instagram є найбільш універсальною платформою, яка дозволяє користувачам створювати візуальний

контент, застосовувати фільтри та взаємодіяти через підписки, лайки та коментарі. Проте він не має спеціалізованих функцій для створення мемів.

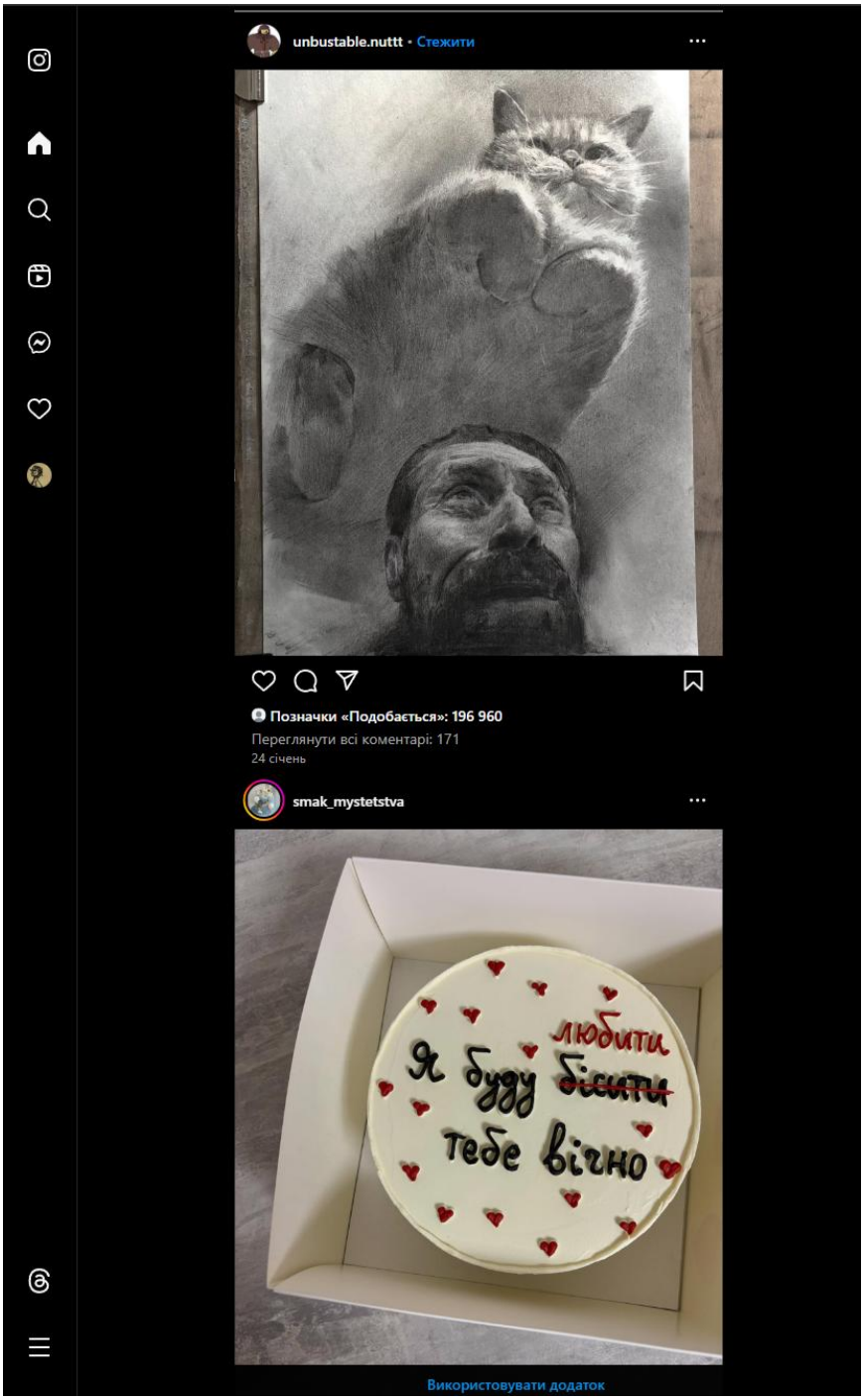

Рисунок 1.1. Публікації у соцмережі Instagram

Instagram вирізняється своєю універсальністю та приваблює широку аудиторію завдяки можливості створювати яскравий візуальний контент, який легко редагується за допомогою фільтрів. Платформа забезпечує активну взаємодію між користувачами — через підписки, вподобайки та коментарі. Також

відкриває можливості для просування через рекламу та рекомендаційні алгоритми. Однак, попри свою популярність, Instagram не надає спеціалізованих інструментів для створення мемів. Його алгоритми також можуть впливати на охоплення публікацій, що ускладнює органічне поширення певного контенту, особливо нішевого або сатиричного спрямування.

Reddit, на відміну від Instagram, побудований на принципі спільнот за інтересами, де користувачі голосують за контент. Це дає можливість організованого обговорення, проте на платформі немає вбудованих інструментів для створення мемів (рис. 1.2).

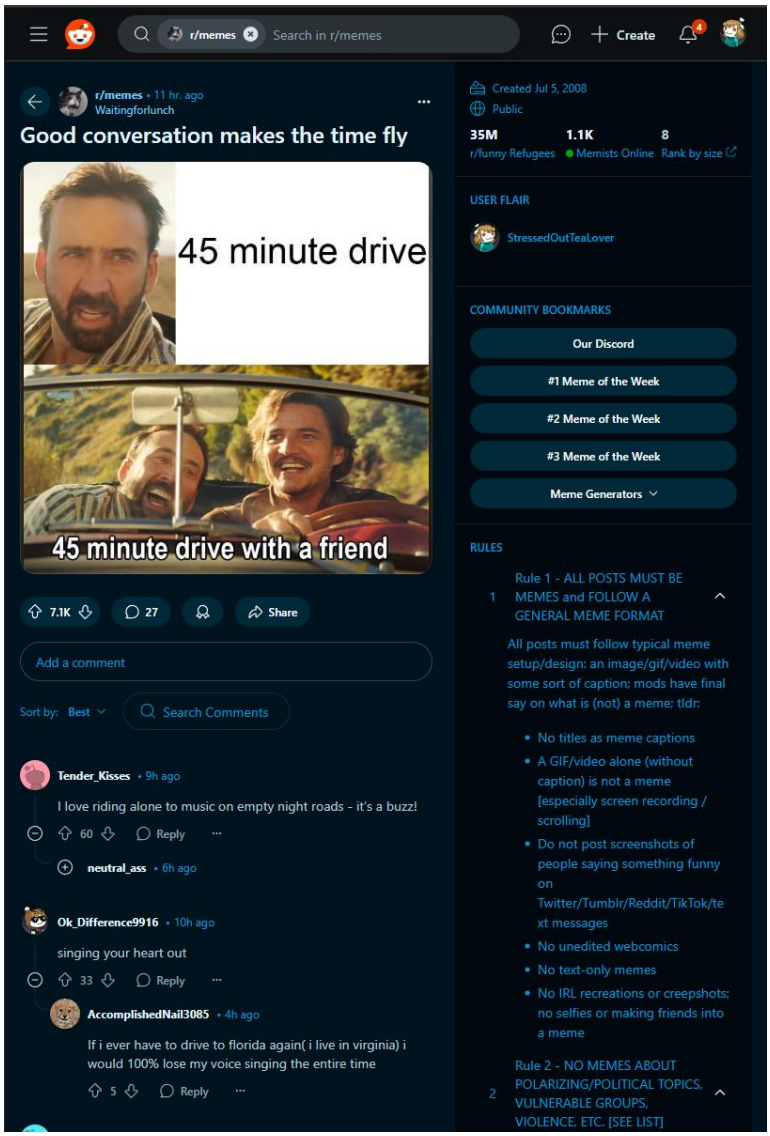

Рисунок 1.2. Обговорення публікації на платформі Reddit

Reddit вирізняється унікальною структурою, що базується на тематичних спільнотах, де кожен користувач може знайти контент за власними інтересами.

Такий формат сприяє глибшим дискусіям, а система голосування дозволяє спільноті самостійно визначати релевантність та якість публікацій. Крім того, механізми модерації в більшості спільнот дають змогу підтримувати порядок та специфіку кожного підрозділу платформи. Водночас Reddit не передбачає інструментів для безпосереднього створення візуального гумористичного контенту, як-от меми. А ще, через відкритість і різноманітність аудиторії, у певних спільнотах можуть виникати конфлікти, що впливають на комфорт користувачів.

9GAG спеціалізується саме на мемах та розважальному контенті (рис. 1.3). Взаємодія здійснюється через лайки, коментарі та репости. Однак платформа більше орієнтована на споживання контенту, ніж на його створення, оскільки вбудованого редактора мемів немає.

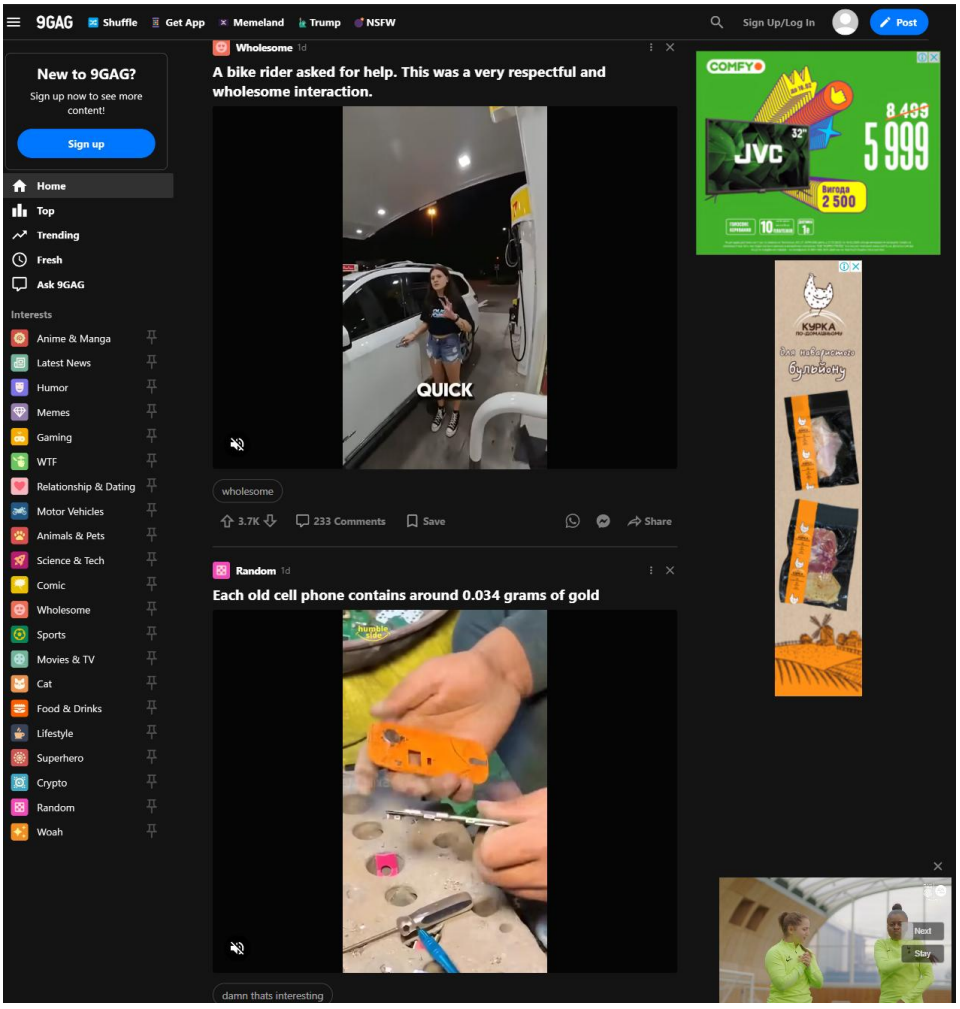

Рисунок 1.3. Головна сторінка 9GAG

9GAG є платформою, яка цілком зосереджена на розважальному контенті, особливо мемах, що робить її привабливою для користувачів, які шукають легкий

гумор і візуальні жарти. Завдяки простому інтерфейсу, перегляд контенту не потребує зусиль, а популярні публікації швидко набирають оберти завдяки системі лайків, коментарів і репостів. Це середовище стимулює вірусне поширення жартів, однак воно менш сприятливе для активної творчості, адже платформа не пропонує вбудованих інструментів для створення власних мемів. Основна аудиторія споживає контент радше пасивно, ніж активно долучається до його продукування. Крім того, значна кількість реклами може відволікати або знижувати загальне враження від користування сайтом.

Короткий опис цих сервісів можна переглянути в порівняльній таблиці 1.1.

Таблиця 1.1

Короткий опис аналогів

Характеристика	**Instagram**	**Reddit**	**9GAG**
Основний контент	Фото, відео	Текст, зображення, відео	Мем-картинки, GIF, відео
Формат взаємодії	Підписки, лайки, коментарі	Голосування, коментарі, підписки на субредіти	Лайки, коментарі, репости
Система рекомендацій	Алгоритмічний фід	Врахування голосів та активності	Популярні пости за рейтингом
Редактор контенту	Фільтри, ефекти	Відсутній	Відсутній
Модерація	Алгоритми та репорти користувачів	Модератори субредитів	Адміністрація платформи
Цільова аудиторія	Загальна, блогери, бренди	Спільноти за інтересами	Любителі мемів, гумору

Насамкінець, порівняймо функціональні вимоги аналогів у таблиці 1.2.

Таблиця 1.2

Функціональні можливості

Функціонал	**Instagram**	**Reddit**	**9GAG**	**Розроблювана система**
Створення мемів	ні	ні	ні	так
Публікація контенту	так	так	так	так
Взаємодія з публікаціями	так	так	так	так
Взаємодія з іншими користувачами	так	так	ні	так
Пошук та рекомендації	так	так	Так	так
Генерація мемів за допомогою ШІ	ні	ні	ні	так

Аналіз аналогічних платформ допоміг визначити сильні та слабкі сторони існуючих рішень. Виявлено, що жодна з проаналізованих платформ не пропонує повноцінного редактора мемів або автоматизованої модерації, а функції взаємодії з контентом (підписки, лайки, коментарі) мають свої обмеження. Таким чином, аналіз аналогів допоміг визначити оптимальний набір функцій та розставити пріоритети у розробці платформи.

1.3. Визначення архітектури ПЗ

Для розробки платформи для створення та публікації мемів було обрано клієнт-серверну архітектуру, у якій клієнтська частина взаємодіє із сервером для отримання, обробки та збереження даних (див. рис.1.4). Такий підхід забезпечує масштабованість, модульність та можливість розширення функціональності.

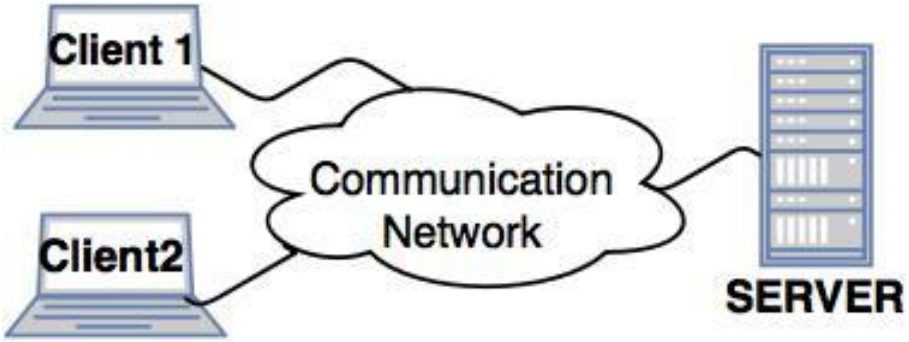

Рисунок 1.4. Схематичне зображення клієнтсько-серверного архітектури

Клієнтська частина реалізована за допомогою React, що відповідає за інтерфейс користувача, взаємодію з редактором мемів та відправку запитів до сервера. У свою чергу, сервер, створений на базі NestJS, обробляє запити, реалізує бізнес-логіку та зберігає дані у PostgreSQL.

NestJS дотримується шаблону модульної архітектури, який значною мірою натхненний принципами модульного моноліту та предметно-орієнтованого проєктування (DDD). Він також включає елементи впровадження залежностей і інверсії управління (IoC) через декоратори та рефлексію метаданих. В архітектурі використано такі патерни:

- Modularization – застосунок розділено на окремі модулі (редактор мемів, система коментарів, модуль рекомендацій, система сповіщень тощо). Це дозволяє легше розширювати функціональність та підтримувати кодову базу.
- Dependency Injection – впроваджено для управління залежностями між компонентами серверної частини. Це дозволяє гнучко налаштовувати сервіси, підвищує тестованість та забезпечує слабке зв’язування між модулями.

Серверна частина реалізує модерацію контенту, рекомендації, систему підписок та сповіщень, взаємодіючи з клієнтською частиною через REST API та WebSocket для реального часу (наприклад, для миттєвих сповіщень та оновлень контенту).

Обрана архітектура забезпечує гнучкість, масштабованість та можливість подальшого розширення функціоналу без значних змін у базовій структурі застосунку. Завдяки модульному підходу кожен компонент системи розробляється окремо, що полегшує тестування, обслуговування та внесення змін без ризику порушення роботи всього застосунку. Масштабованість досягається завдяки можливості додавання нових модулів або оптимізації існуючих без необхідності переробляти всю систему. Крім того, використання базових патернів NestJS дозволяє легко інтегрувати нові сервіси та адаптувати застосунок до змінюваних вимог бізнесу. Таким чином, архітектура сприяє ефективному розвитку проєкту, дозволяючи розширювати його функціональність поступово, без значного збільшення складності коду та збереження високої продуктивності.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до ПЗ

У процесі розробки системи було використано кілька ключових інструментів, що забезпечують ефективність, продуктивність та зручність роботи. Одним із головних виборів стало середовище розробки Visual Studio Code. Це легке та водночас потужне середовище, яке підтримує широкий набір розширень для різних мов програмування та технологій. Завдяки вбудованим можливостям підсвічування синтаксису, автоматичного доповнення коду та підтримки систем контролю версій, Visual Studio Code дозволяє розробникам працювати швидше та ефективніше.

Для тестування API-запитів було обрано Postman. Цей інструмент значно полегшує розробку та налагодження API, дозволяючи створювати, тестувати та аналізувати запити до сервера в зручному графічному інтерфейсі. Використання Postman допомагає автоматизувати перевірку роботи API, що сприяє швидкому виявленню помилок та покращенню якості сервісу.

Для роботи з базами даних були проаналізовані найпопулярніші варіанти: MySQL, PostgreSQL та MongoDB. Порівняння цих баз даних можна побачити у таблиці 1.3. обрано PostgreSQL, що є потужною реляційною системою керування базами даних. Вона забезпечує високу продуктивність, підтримує складні запити, транзакції та механізми безпеки. Для зручного керування базою даних та її

розгортання використовується Docker, що дозволяє створювати ізольовані контейнери, полегшуючи управління середовищем розробки та підготовку до розгортання в продакшн. Використання контейнеризації гарантує стабільність роботи та однакові умови для всіх середовищ.

Таблиця 1.3

Порівняння баз даних

Система	**Продуктивність**	**Безпека**	**Масштабованість**
PostgreSQL	Висока	Висока	Гнучка
MySQL	Середня	Середня	Середня
MongoDB	Висока	Середня	Висока

При розробці інтерфейсу застосовувався інструмент Figma. Це зручний сервіс для створення дизайну та прототипів, що підтримує командну роботу та інтеграцію з іншими сервісами. Figma дозволяє швидко розробляти та тестувати макети, що сприяє ефективній взаємодії між розробниками та дизайнерами.

Вибір кожного інструмента обумовлений його популярністю, функціональністю та відповідністю поставленим вимогам. Використання сучасних рішень дозволяє досягти високої продуктивності, гнучкості та масштабованості проєкту.

Висновки до першого розділу

У першому розділі було проведено детальний аналіз існуючих аналогів, визначено архітектуру програмного забезпечення, обґрунтовано вибір інструментальних засобів і сформульовано основні завдання.

Аналіз аналогів дозволив визначити ключові функціональні вимоги до системи, зокрема інтеграцію спеціалізованого редактора мемів, впровадження алгоритмів рекомендацій і модерації.

Для реалізації проекту було обрано клієнтно-серверну архітектуру, яка забезпечує масштабованість і гнучкість. Клієнтська частина створена на основі React, що дозволяє розробити зручний інтерфейс користувача, а серверна частина базується на NestJS, що забезпечує ефективну обробку запитів і взаємодію з базою

даних PostgreSQL. Використання патернів модульної архітектури та впровадження залежностей значно спрощує підтримку та розвиток платформи. При виборі технологічного стеку враховувалися продуктивність, безпека та масштабованість.

Загалом, задумана архітектура та вибрані технології забезпечують високу продуктивність, надійність і можливість подальшого розширення функціоналу платформи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПЛІТВОРМИ ДЛЯ СТВОРЕННЯ ТА ПУБЛІКАЦІЇ МЕМІВ

2.1. Визначення варіантів використання об’єктно-орієнтованої структури системи

Система є веб-платформою для створення, редагування, публікації та взаємодії з мемами. Вона використовує архітектуру та логіку взаємодії основних об'єктів у веб-додатку для створення і публікації мемів. Система базується на стеку Next.js та ReactQuery для фронтенду, NestJS для бекенду, базі даних PostgreSQL, сервісі UploadThing для зберігання зображень та бібліотеці Konva для редактора зображень. Основними функціями системи є створення мемів, їх публікація, взаємодія користувачів через лайки, коментарі, підписки, сповіщення Ai можливість скарг на користувачів або контент.

У системі визначено дві основні ролі: користувач і модератор.

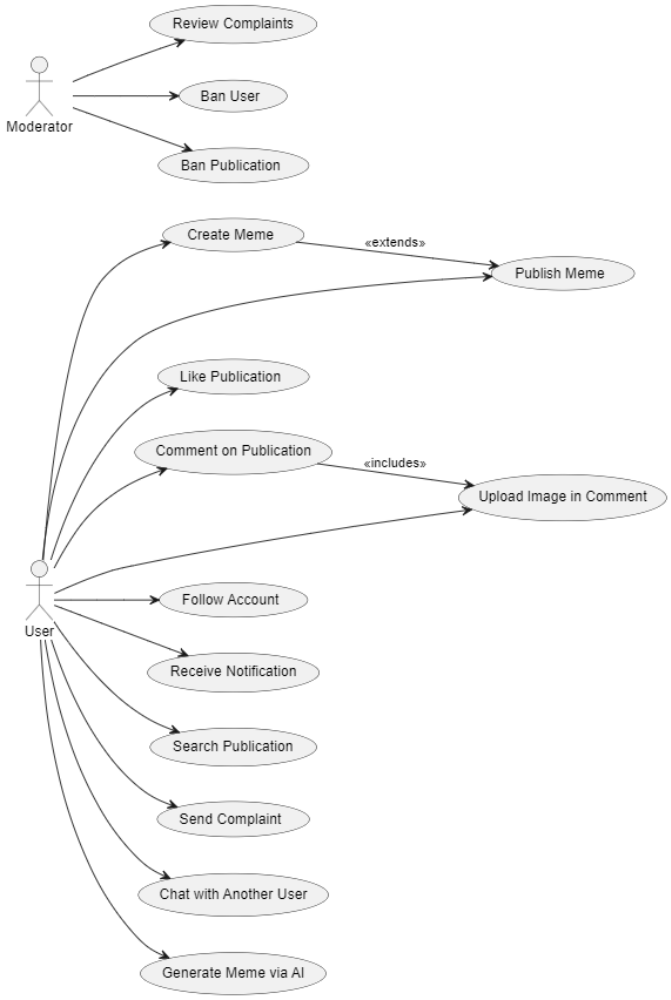

Рис. 2.1. Діаграма варіантів використання

Користувачі можуть створювати меми, коментувати, вподобувати публікації та підписуватися на інші акаунти. Модератори мають доступ до скарг, можуть банити публікації та користувачів, а також здійснювати перегляд і обробку скарг через адміністративний інтерфейс.

Основні прецеденти описані в таблицях 2.1 - 2.9.

Табл. 2.1

Опис прецеденту «Створення мему»

Назва	Створення мему
Опис	Користувач створює зображення через редактор, додає текст та оформлення
Виконавці	User

Табл. 2.2

Опис прецеденту «Публікація мему»

Назва	Створення мему
Опис	Користувач публікує створений мем для перегляду іншими користувачами
Виконавці	User

Табл. 2.3

Опис прецеденту «Пошук публікацій»

Назва	Пошук публікацій
Опис	Користувач шукає публікації за ключовими словами або іншими фільтрами
Виконавці	User

Табл. 2.4

Опис прецеденту «Генерація мему за описом»

Назва	Генерація мему за описом
Опис	Користувач вводить опис, система створює мем за допомогою ШІ
Виконавці	User

Табл. 2.5

Опис прецеденту «Лайк публікації»

Назва	Лайк публікації
Опис	Користувач ставить вподобання під обраною публікацією
Виконавці	User

Табл. 2.6

Опис прецеденту «Перегляд скарг»

Назва	Лайк публікації
Опис	Модератор переглядає скарги користувачів на публікації або акаунти
Виконавці	Moderator

Табл. 2.7

Опис прецеденту «Блокування публікації»

Назва	Блокування публікації
Опис	Модератор блокує публікацію, що порушує правила платформи
Виконавці	Moderator

Табл. 2.8

Опис прецеденту «Підписка на користувача»

Назва	Підписка на користувача
Опис	Користувач підписується на інший акаунт для відстеження нових публікацій
Виконавці	User

Табл. 2.9

Опис прецеденту «Створення скарги»

Назва	Створення скарги
Опис	Користувач скаржиться на користувача або публікацію у випадку порушення правил
Виконавці	User

Розглянемо діаграму класів, що на рисунку 2.2.

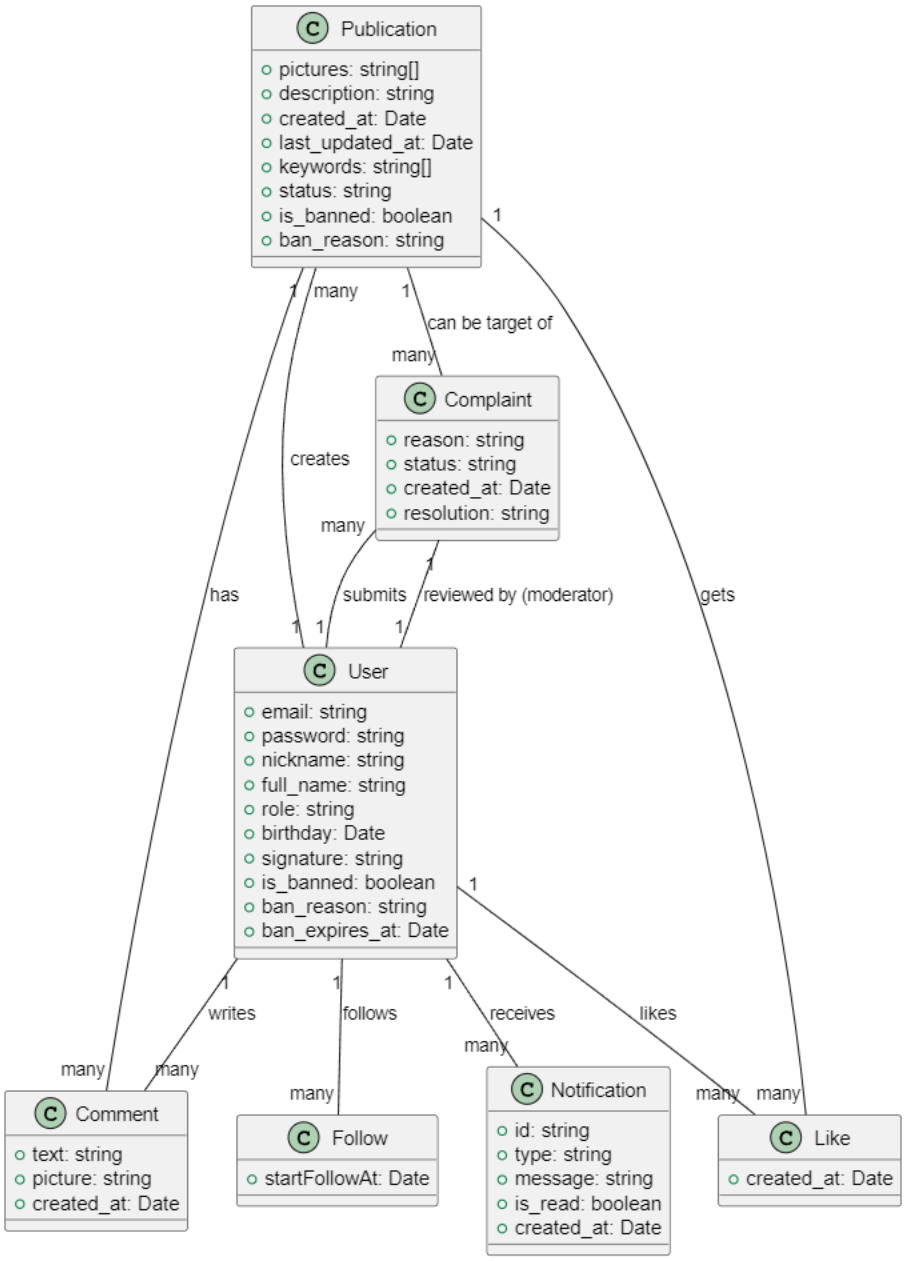

Рис. 2.2. Діаграма класів

Діаграма класів описує основні сутності системи для платформи створення та публікації мемів, а також взаємозв'язки між ними. Користувач (User) є центральною фігурою: він може створювати публікації (Publication), залишати коментарі (Comment), ставити вподобання (Like), підписуватися на інших користувачів (Follow), отримувати сповіщення (Notification) та надсилати скарги (Complaint). Кожна публікація містить масив зображень, опис, ключові слова та дані про її статус, при цьому вона може отримувати коментарі, вподобання та

скарги. Ця структура забезпечує гнучку та розширювану модель даних, необхідну для розвитку соціальної активності на платформі.

2.2. Розробка бази даних системи

Для зберігання усієї критичної інформації системи використовується база даних PostgreSQL. Коротки опис можна побачити в таблицях 2.10 - 2.16.

Табл. 2.10

Таблиця «User»

Поле	**Тип**	**Nullable**
email	string	ні
password	string	ні
nickname	string	ні
full_name	string	так
role	string	ні
birthday	Date	ні
signature	string	ні
is_banned	boolean	ні
ban_reason	string	так
ban_expires_at	Date	так

Табл. 2.11

Таблиця «Publication»

Поле	**Тип**	**Nullable**
pictures	string[]	ні
description	string	ні
author_id	FK → user	ні
created_at	Date	ні
last_updated_at	Date	ні

Продовження таблиці 2.11

Поле	Тип	Nullable
keywords	string[]	ні
status	string	ні
is_banned	boolean	ні
ban_reason	string	так

Табл. 2.12

Таблиця «Comment»

Поле	Тип	Nullable
text	string	ні
picture	string	так
user_id	FK → user	ні
publication_id	FK → publication	ні
created_at	Date	ні

Табл. 2.13

Таблиця «Like»

Поле	Тип	Nullable
liker_id	FK → user	ні
publication_id	FK → publication	ні
created_at	Date	ні

Табл. 2.14

Таблиця «Follow»

Поле	Тип	Nullable
follower_id	FK → user	ні
following_id	FK → user	ні
created_at	Date	ні

Табл. 2.15

Таблиця «Notification»

Поле	**Тип**	**Nullable**
user_id	FK → user	ні
type	string	ні
message	string	ні
is_read	boolean	ні
created_at	Date	ні

Табл. 2.16

Таблиця «Complaint»

Поле	**Тип**	**Nullable**
complainer_id	FK → user	ні
target_user_id	FK → user	так
target_publication_id	FK → publication	так
reason	string	ні
status	string	ні
created_at	Date	ні
reviewed_by_id	FK → user	так
resolution	string	так

Центральною таблицею є таблиця користувачів, яка зберігає унікальну електронну пошту, зашифрований пароль, нікнейм, а також за потреби повне ім’я. Роль користувача визначає рівень доступу в системі (наприклад, адміністратор чи звичайний користувач). Додатково враховуються дата народження, персональний підпис, а також можливість фіксації бану, його причини та дати завершення.

Публікації є ще однією ключовою сутністю, що містить зображення у вигляді масиву посилань, опис, ключові слова, дату створення та останнього оновлення. До кожної публікації прив’язано автора, її статус (наприклад, опублікована, прихована

або на розгляді), а також інформацію про можливий бан публікації й причину цього.

Коментарі дозволяють користувачам взаємодіяти з контентом, маючи текстовий вміст, за потреби зображення, і прив’язку до публікації та користувача, що його залишив. Також зберігається дата створення коментаря.

Функціональність вподобань реалізується через окрему таблицю, яка фіксує зв’язок між користувачем, що вподобав публікацію, та самою публікацією разом із часом взаємодії.

Соціальний аспект системи реалізовано через таблицю підписок, у якій зазначено, хто саме підписався на кого та коли розпочалося відстеження. Це дозволяє будувати мережу зв’язків між користувачами.

Для забезпечення інформування користувачів створено механізм сповіщень. Кожне сповіщення має унікальний ідентифікатор, пов’язується з конкретним користувачем, має тип (наприклад, бан, коментар, лайк, скарга), текст повідомлення, статус прочитаності та дату створення.

Механізм скарг дозволяє фіксувати звернення користувачів щодо порушень. У скарзі зазначається особа, яка подала скаргу, її об'єкт (користувач або публікація), причина, статус розгляду, дата створення, модератор, що розглядав скаргу, та резолюція за результатами перевірки.

Загальна структура бази даних спрямована на забезпечення стабільної та безпечної взаємодії користувачів із платформою, надаючи можливість ефективно реагувати на порушення, стимулювати соціальну активність і формувати прозоре інформаційне середовище.

2.3. Проєктування та реалізація алгоритмів роботи платформи

Для більшості функцій на платформі, наприклад створення мемів, публікації, коментування і тому подібне, користувачам потрібно ввійти на вебсайт. Діаграма активності на рис. 2.1 показує процес авторизації та реєстрації в системі, від введення логіну користувача та пароля, а також реєстраційної форми, до успішного входу в систему.

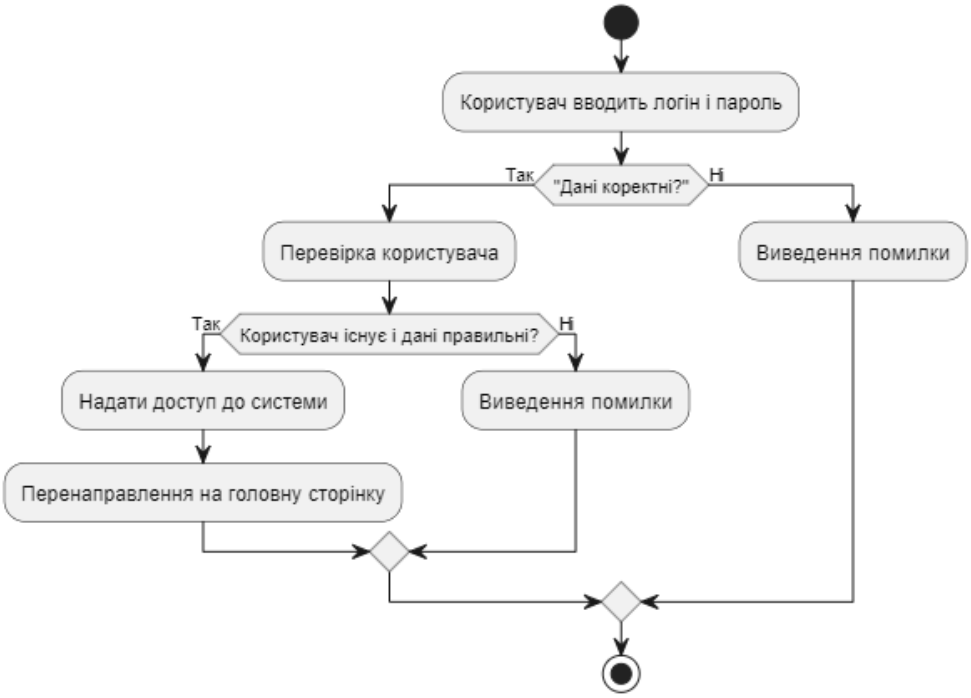

Рис. 2.3. Діаграма активності для авторизації

Під час входу в систему користувачеві буде виведено сторінку з формою для входу, де він має ввести свої облікові дані. Якщо введені дані є неправильними, то користувач залишиться на сторінці для входу, для повторного введення свого логіну та паролю. Якщо ж користувач ввів коректні дані, то він отримує доступ до функціоналу системи відповідно до його ролі. Варто згадати, що користувач має попередньо існувати в системі, тому перед авторизацією користувач має зареєструватися. Аналогічно до авторизації, під час реєстрації має відобразиться реєстраційна форма, де користувач має заповнити всі обов'язкові поля так, щоб було пройдено валідацію. Після валідації даних на клієнті, вони відправляються на сервер і там також відбувається перевірка на правильність даних.

На рис. 2.4 зображено діаграму активності для процесу створення та публікації мему, що містить 2 доріжки. Доріжки використовуються для групування дій, які виконуються одним актором. У даній діаграмі першу доріжку представляє Користувач, а другу – Система.

Якщо користувач на етапі перегляду мему не задоволений результатом, він може повернутися до редагування та внести зміни перед повторною публікацією. Після підтвердження мему та успішної публікації процес завершується.

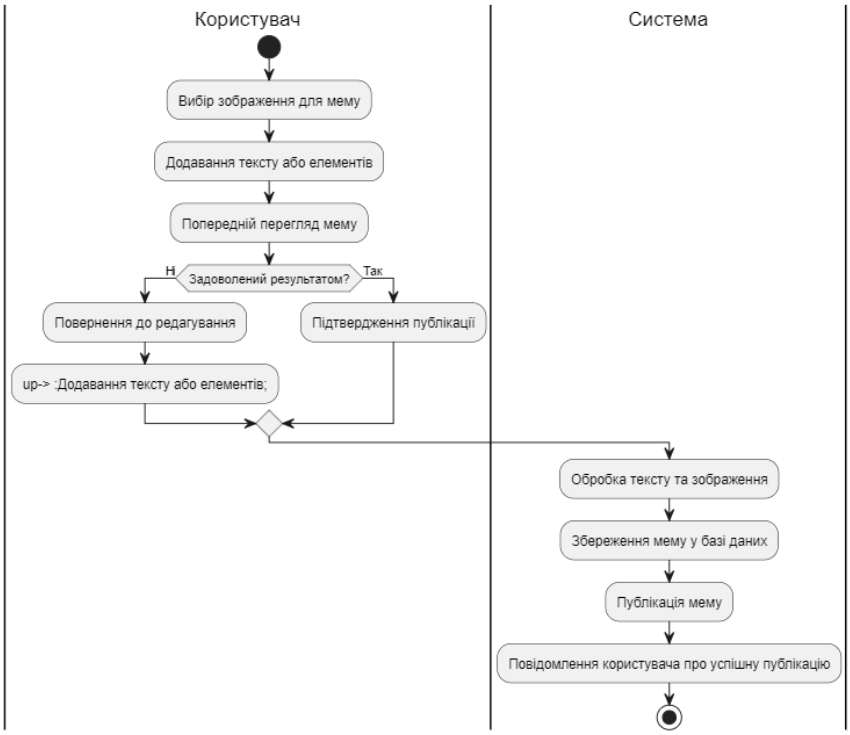

Рис. 2.4. Діаграма активності для створення й публікації мему

Діаграма активності для процесу коментування публікацій відображає взаємодію між користувачем і системою, розділену на дві доріжки: одна для дій користувача, інша для обробки даних системою.

Діаграма активності для процесу скарги на публікацію (див. рис. 2.5) демонструє, як користувач взаємодіє із системою для надсилання скарги, та як система обробляє цей запит.

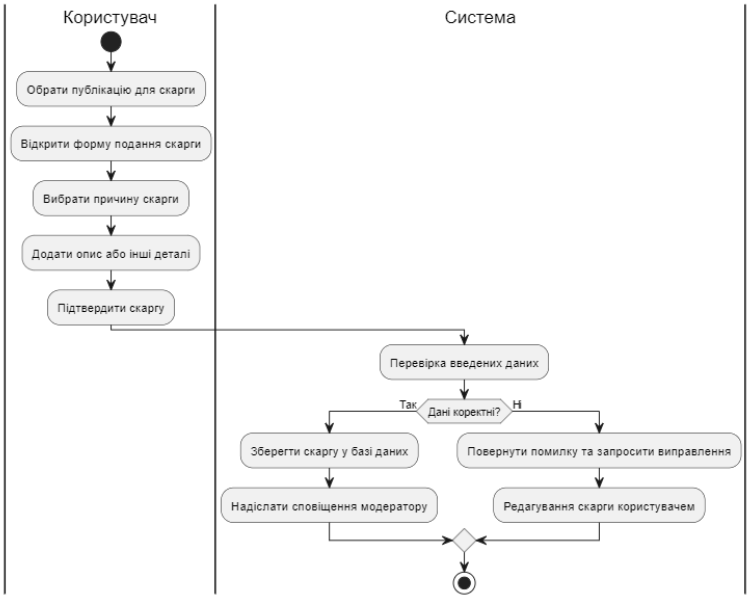

Рис. 2.5. Діаграма активності подачі скарги користувача на публікацію

Процес починається з того, що користувач обирає публікацію, на яку хоче поскаржитися. Далі користувачу надається можливість вказати причину скарги, обравши її зі списку запропонованих варіантів (наприклад, неналежний контент або порушення правил платформи) або ввівши власний текст для уточнення причини. Користувач також може, за бажанням, додати додаткові докази – наприклад, скріншот або інше зображення.

Після заповнення всіх необхідних даних користувач підтверджує скаргу. На цьому етапі система перевіряє введені дані на наявність обов'язкових полів, зокрема - чи обрано причину скарги та чи введено опис (якщо необхідно). Якщо обов’язкові дані відсутні або завантажене зображення некоректне, користувач повертається до редагування скарги для внесення виправлень.

Таким чином, діаграма активності показує злагоджений процес обробки скарг на публікації, забезпечуючи можливість внесення коригувань на етапі формування скарги та гарантуючи, що всі необхідні дані будуть передані для подальшого розгляду модераторами.

Процес реагування на скаргу (див. рис. 2.6) розпочинається з моменту, коли модератор отримує сповіщення про нову скаргу. Далі модератор переглядає деталі скарги та публікацію, на яку вона надійшла. Після цього модератор має ухвалити рішення про обґрунтованість скарги.

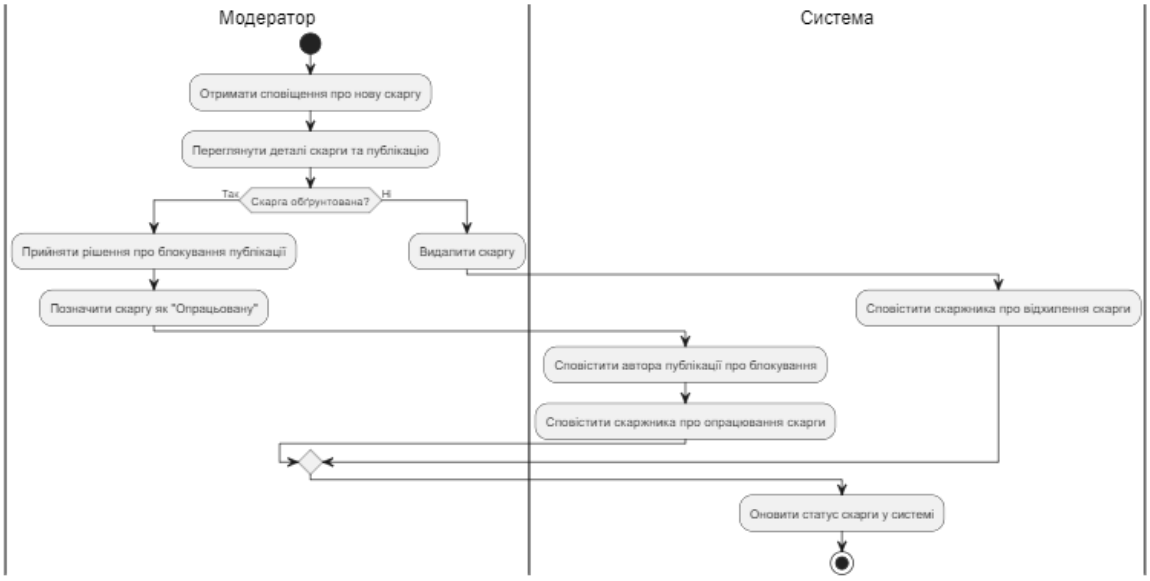

Рис. 2.6. Діаграма активності реагування скарги модератором

Якщо скарга визнана обґрунтованою, модератор блокує публікацію, позначає скаргу як "Опрацьовану", а система виконує важливі дії: сповіщає автора публікації про блокування контенту та інформує скаржника про успішне опрацювання скарги.

Якщо ж скарга виявляється необґрунтованою, модератор її відхиляє та видаляє її. У цьому випадку система надсилає скаржнику повідомлення про відхилення скарги.

Завершальним етапом є оновлення статусу скарги у системі, що фіксує результат роботи модератора. Після цього процес завершується.

Таким чином, у діаграмі чітко розмежовані дії модератора та системи: модератор ухвалює рішення і виконує адміністративні функції, а система автоматично сповіщає учасників про результат розгляду скарги.

2.4. Реалізація функціоналу платформи

Уся серверна логіка побудована на основі фреймворку NestJS, що забезпечує масштабованість і модульність. У реалізації застосовано патерн Singleton, що гарантує єдиний екземпляр кожного сервісу в межах усього життєвого циклу додатку, підвищуючи ефективність роботи з ресурсами та стабільність поведінки ін’єкованих залежностей. Візуалізація цього патерну можна побачити на рис. 2.7.

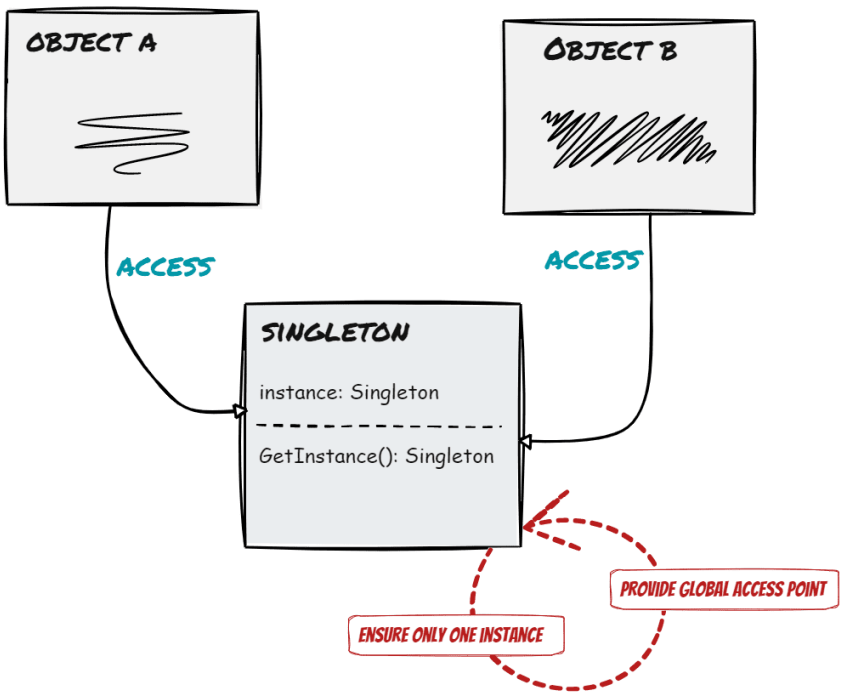

Рис. 2.7. Візуалізація патерну Singleton

Центральним елементом платформи є модуль обробки публікацій.

Його основна функція полягає в створенні, оновленні, фільтрації, перегляді та видаленні записів користувачів. Для отримання публікацій реалізовано гнучку систему фільтрації за ключовими словами, авторством, статусом модерації та наявністю блокування. Запити формуються через query builder з урахуванням складених умов пошуку та сортування, наприклад, за датою створення. Також враховується контекст користувача, що дозволяє відображати персоналізовану інформацію, зокрема, позначення "вподобаних" публікацій. У процесі створення публікації здійснюється автоматичне завантаження зображень, після чого отримані URL-адреси зберігаються разом із метаданими в базі даних.

Завантаження файлів здійснюється через окремий сервіс, що взаємодіє з хмарним сховищем, використовуючи API від UploadThing. Завантаження файлів на сховище можна переглянути на рисунку 2.8.

```
async uploadFiles(files: Express.Multer.File[]): Promise<string[]> {
  if (files.length < 1) {
    throw new BadRequestException('No files provided');
  }

  const modifiedFiles = await Promise.all(
    files.map(async (file) => await this.modifyFile(file)),
  );

  try {
    const res = await this.utApi.uploadFiles(modifiedFiles);

    return res.map((file) => file.data.url);
  } catch (error) {
    console.error('Upload failed:', error);
    throw new InternalServerErrorException();
  }
}
```

Рис. 2.8. Завантаження файлів у сховище

Перед передачею даних зображення обробляються бібліотекою sharp – змінюється розмір відповідно до встановлених обмежень, зберігаючи при цьому якість і зменшуючи навантаження на клієнт. Кожен файл отримує унікальну назву, що унеможливлює конфлікти та дублікати. Сервіс також підтримує видалення

файлів за ключами, отриманими з URL, що дозволяє підтримувати чистоту сховища при оновленні чи видаленні публікацій (рис.2.9).

```
async updateOne(
  id: string,
  updatePublicationDto: UpdatePublicationDto,
  user: UserEntity,
  pictures?: Express.Multer.File[],
): Promise<PublicationEntity> {
  const publication = await this.getOne(id, user.id);

  const newPictures =
    pictures?.length > 0
      ? await this.fileUploadService.uploadFiles(pictures)
      : publication.pictures;

  if (pictures?.length > 0)
    await this.fileUploadService.deleteFiles(publication.pictures);

  const updatedPublication = {
    ...publication,
    ...updatePublicationDto,
    lastUpdatedAt: new Date(Date.now()).toISOString(),
    pictures: newPictures,
  };

  await this.publicationRepository.save(updatedPublication);

  return publication;
}
```

Рис. 2.9. Оновлення публікації із заміною файлів

Модуль авторизації забезпечує класичну автентифікацію за email та паролем, а також інтеграцію з Google через OAuth. Для реєстрації нового користувача передбачено перевірку відповідності пароля і його підтвердження, а також захист пароля шляхом хешування через bcrypt. Після успішного входу або реєстрації генерується JWT-токен, який клієнт використовує для аутентифікованих запитів. Для обмеження доступу до певних ендпоїнтів використовується один з кількох захисних декораторів. На рисунку 2.10 наведений приклад використання декоратора, що вимагає обов'язкову авторизацію, на рисунку 2.11 – лише отримує дані авторизованого користувача, якщо такий існує.

```
@HttpCode(HttpStatus.CREATED)
@UseInterceptors(FileInterceptor('picture'))
@UseGuards(JwtAuthGuard)
@Post()
createComment(
  @Body() createCommentDto: CreateCommentDto,
```

Рис. 2.10. Повний захист ендоїнта від неавторизованих користувачів

```
@HttpCode(HttpStatus.OK)
@UseGuards(OptionalJwtAuthGuard)
@Get()
getPublications(
  @GetUser() user?: UserEntity,
```

Рис. 2.11. Частковий захист ендпоїнта від неавторизованих користувачів

У випадку входу через Google, у разі відсутності користувача, створюється новий запис із мінімальним набором обов'язкових полів.

```
async loginWithGoogle(profile: Profile): Promise<AuthResultDto> {
  const email = profile.emails[0]?.value;
  let user: UserEntity | UserEntity;

  try {
    user = await this.userService.getOne(undefined, email);
  } catch (error) {
    console.error(error);
    if (error.status == 404) {
      user = await this.userService.createOne({
        email,
        username: email.split('@')[0],
        fullName: profile.displayName,
        role: 'user',
        password: null,
      });
    } else {
      console.error(error);
      throw new InternalServerErrorException();
    }
  }

  return {
    user: {
      id: user.id,
      email: user.email,
      username: user.username,
    },
    accessToken: await this.generateToken(user.id, user.email),
  };
}
```

Висновки до другого розділу

У другому розділі було детально окреслено архітектурне та функціональне проєктування платформи, визначаючи ключові компоненти та взаємодії системи. Було розроблено варіанти використання системи, де чітко окреслено ролі користувача та модератора, а також типові сценарії їхньої взаємодії з платформою. Зокрема, описані можливості створення, публікації, пошуку, оцінювання та

модерування контенту, що демонструє орієнтацію проєкту на активну участь користувачів і забезпечення контролю за дотриманням правил спільноти.

Класова діаграма та структура бази даних демонструють об’єктно-орієнтований підхід до моделювання системи. Вона включає гнучку і масштабовану модель даних, що забезпечує збереження й обробку основної інформації. Такий підхід дозволяє ефективно обслуговувати основні функції соціальної взаємодії та модерації.

Завдяки побудові діаграм активності були окреслені ключові алгоритми взаємодії користувача із системою, зокрема процеси реєстрації, авторизації, створення і публікації мему. Це забезпечує чітке розуміння логіки платформи як для користувачів, так і для розробників.

Загалом, реалізований функціонал платформи демонструє цілісну архітектуру із чітким розмежуванням обов’язків між сервісами. Такий підхід дозволяє не лише підтримувати чистоту коду, а й легко розширювати платформу в майбутньому.

ВИСНОВКИ

В результаті написання випускної кваліфікаційної роботи бакалавра проаналізовано існуючі аналоги, визначено архітектуру ПЗ, обґрунтовано вибір інструментів та сформульовано основні завдання. Аналіз дозволив визначити ключові вимоги: інтеграцію редактора мемів, алгоритми рекомендацій і модерації.

Обрана клієнтно-серверна архітектура: клієнтська частина на Next.js для зручного інтерфейсу, серверна – на NestJS з базою PostgreSQL для ефективної обробки запитів. Використання модульної архітектури спрощує підтримку. Вибрані технології забезпечують продуктивність, безпеку та масштабованість.

Переддипломна практика була важливим етапом перед написанням дипломної роботи, що дозволила закріпити теоретичні знання, набути практичні навички та підготуватися до професійної діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Instagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.instagram.com/
2. Reddit [Електронний ресурс] – Режим доступу до ресурсу: https://www.reddit.com/
3. 9GAG [Електронний ресурс] – Режим доступу до ресурсу: https://9gag.com/
4. Документація фреймворку NestJS [електронний ресурс] – Режим доступу до ресурсу: https://docs.nestjs.com/
5. Документація бібліотеки React [електронний ресурс] – Режим доступу до ресурсу: https://react.dev/learn
6. Документація бази даних PostgreSQL [електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/
7. Теза на тему «Підбір рекомендацій публікацій з використанням content-based та collaborative-based фільтрації» [електронний ресурс] – Режим доступу до ресурсу: <Посилання на тезу>