КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 «Комп'ютерні науки»
Освітньо-наукова програма «Управління проєктами»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

"Дослідження процесів управління проєктом створення веб-платформи для підвищення екологічної культури населення"

Студент 2-го курсу групи УП-21

Роман Жовтецький
(ім'я, прізвище)

(підпис студента)

Науковий керівник:

к.т.н., доцент
(науковий ступінь, вчене звання)

Тетяна ЛАТИШЕВА
(ім'я, прізвище)

(дата) *(підпис)*

Попередній захист:

(Висновок: "До захисту в Екзаменаційній комісії")

Завідувач кафедри
технологій управління (підпис) Віктор МОРОЗОВ (ім'я, прізвище) (дата)

Київ - 2025

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітній рівень Магістр
Спеціальність 122 Комп'ютерні науки
Освітньо-наукова програма Управління проєктами

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Віктор МОРОЗОВ
_____
"27" листопада 2024 року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент: Жовтецький Роман Дмитрович
Група: УП-22

1. Тема кваліфікаційної роботи
«Дослідження процесів управління проєктом створення веб-сервісу для перегляду веб-коміксів»
Затверджена протоколом кафедри ТУ від "26" листопада 2024 року.
2. Строк подання студентом готової роботи - "__" _____ ____ р.
3. Цільова установка та вихідні дані до роботи: дослідження сучасних підходів до управління IT-проєктами у сфері цифрових медіа, розробка концепції веб-сервісу для перегляду веб-коміксів, моделювання архітектури системи, управління командою, ресурсами, якістю, строками, бюджетом та ризиками у межах життєвого циклу проєкту.
4. Зміст роботи:
Аналіз ринку цифрових коміксів, визначення цільової аудиторії, побудова концептуальної моделі платформи, опис життєвого циклу проєкту, архітектура системи, WBS-модель, організаційна структура, календарне та фінансове планування, управління якістю, ризиками та взаємодією зі стейкхолдерами.
5. Перелік графічного матеріалу (слайдів):
титульна сторінка, постановка проблеми, мета проєкту, структура команди, діаграма Ганта, архітектура системи, структура бази даних, логіко-структурна схема, дерево проблем, дерево цілей, SWOT-аналіз, PEST-аналіз, дорожня карта, фінансова модель, елемент коду, ризики проєкту, висновки.

6. Календарний план виконання роботи:

№ з/п	Назва частин роботи	Виконання роботи
1	Вивчення літературних джерел з предмету дослідження	22.01.25 - 22.02.25
2	Збір і вивчення матеріалів досліджуваного підприємства	22.01.25 - 26.01.25
3	Складання розгорнутого плану кваліфікаційної роботи	15.01.25 - 22.01.25
4	Ознайомлення наукового керівника з планом кваліфікаційної роботи	22.01.25
5	Підготовка розділу 1	05.02.25 - 26.02.25
6	Підготовка розділу 2	27.02.25-16.03.25
7	Підготовка розділу 3	17.03.25-06.04.25
8	Підготовка розділу 4	07.04.25-25.04.25
9	Оформлення кваліфікаційної роботи	28.04.25-07.05.25
10	Передача кваліфікаційної роботи науковому керівникові	08.05.25
11	Передача кваліфікаційної роботи рецензенту для рецензування	12.05.25
12	Захист кваліфікаційної роботи	26.05.25 - 28.05.25

Дата видачі завдання “28” листопада 2024 р.

Керівник роботи: к.т.н., доцент, Тетяна Латишева

_____
(підпис)

Завдання прийняв до виконання:
студент групи УП-22 Роман Жовтецький _____
(підпис)

ЗМІСТ

ТАБЛИЦЯ СКОРОЧЕНЬ ТА ПОЯСНЕНЬ

Скорочення	**Пояснення**
ПЗ	Програмне забезпечення
MVP	Мінімально життєздатний продукт (Minimum Viable Product)
UX	Досвід користувача (User Experience)
UI	Інтерфейс користувача (User Interface)
EVM	Метод освоєного обсягу (Earned Value Management)
ЖЦ	Життєвий цикл
WBS	Ієрархічна структура робіт (Work Breakdown Structure)
BPMN	Нотація моделювання бізнес-процесів (Business Process Model and Notation)
UML	Уніфікована мова моделювання (Unified Modeling Language)
KPI	Ключові показники ефективності (Key Performance Indicators)
API	Інтерфейс прикладного програмування (Application Programming Interface)
PWA	Прогресивний веб-додаток (Progressive Web App)
AWS	Хмарна платформа Amazon Web Services
CI/CD	Безперервна інтеграція та розгортання (Continuous Integration / Continuous Deployment)
ROI	Коефіцієнт окупності інвестицій (Return on Investment)
SWOT	Метод стратегічного аналізу сильних і слабких сторін, можливостей і загроз
CRUD	Операції створення, читання, оновлення та видалення даних (Create, Read, Update, Delete)

АНОТАЦІЯ

кваліфікаційної роботи магістра на тему

«Дослідження процесів управління проєктом створення веб-сервісу для перегляду веб-коміксів»

Студент: Жовтецький Роман Дмитрович

Науковий керівник: Латишева Тетяна Володимирівна

Рік захисту: 2025

Метою досліджуваної роботи є визначення ефективних методів і стратегій управління ІТ-проєктом зі створення цифрового продукту — веб-платформи для перегляду веб-коміксів, з урахуванням особливостей веб-розробки, авторського контенту, потреб користувачів і технологічних обмежень.

Ціль проєкту — розробити функціональну, адаптивну та масштабовану платформу «MangaView» для зручного читання веб-коміксів онлайн, яка об’єднає інтереси читачів, авторів і видавців, а також забезпечить можливість публікації, персоналізації та інтерактивної взаємодії.

Наукова новизна полягає у розробці комплексної моделі управління проєктом створення цифрової платформи з елементами Agile-методології, побудові структурної моделі цілей і проблем проєкту, формалізації процесу бюджетування та ризик-менеджменту в умовах ІТ-стартапу. Особливу увагу приділено формалізації задач у математичному вигляді для оцінки тривалості та вартості проєкту.

Дана магістерська робота присвячена аналізу, моделюванню та реалізації процесів управління розробкою веб-сервісу у сфері цифрового медіа. Реалізація такого проєкту потребує поєднання технологічних, дизайнерських і бізнес-

підходів, а також адаптації до динамічних очікувань користувачів і ринкових трендів.

Перший розділ роботи висвітлює теоретичні основи управління IT-проєктами. Проведено огляд класичних та гнучких методологій (Waterfall, Scrum, Hybrid), досліджено потреби ринку веб-коміксів, а також здійснено PEST- і SWOT-аналіз для оцінки впливів внутрішнього та зовнішнього середовища.

Другий розділ присвячений технічному й організаційному моделюванню. Побудовано календарний план, розраховано бюджет і витрати на реалізацію MVP-продукту. Проведено формалізацію математичної моделі задач управління.

Третій розділ охоплює практичні аспекти: розподіл відповідальностей, управління якістю, ризиками, фінансами, а також план комунікації зі стейкхолдерами. Визначено ключові метрики контролю (KPI), сценарії моніторингу та базові інструменти підтримки реалізації (MS Project, GitHub, Figma).

Четвертий розділ присвячено реалізації, тестуванню та пілотному запуску веб-сервісу. Описано стек технологій, базову архітектуру, принципи front-end та back-end реалізації, а також організовано перевірку якості за допомогою автоматизованих засобів.

Практична цінність роботи полягає у створенні підходу до управління розробкою веб-сервісів з авторським контентом, який може бути використаний для запуску подібних IT-продуктів у креативних і видавничих індустріях.

Робота містить 102 сторінки без урахування додатків, 14 рисунків та 13 таблиць. Додатки займають 3 сторінки.

Ключові слова: *управління проєктами, Scrum, веб-комікси, веб-сервіс, планування, ризики, бюджетування, MVP, календарний план, UI/UX.*

ВСТУП

Сучасний цифровий світ характеризується стрімким розвитком візуальних медіа, серед яких особливе місце займають веб-комікси. Завдяки поєднанню художнього мистецтва, наративу та зручного онлайн-доступу, цей формат здобув популярність серед широкої аудиторії, особливо серед молоді. Проте, попри зростання зацікавлення до коміксів у цифровому форматі, в Україні все ще бракує зручних, функціональних та локалізованих платформ для їх перегляду. Більшість існуючих сервісів мають або обмежену функціональність, або не враховують особливості українського ринку, зокрема мовну специфіку, потреби локальних авторів, та очікування користувачів.

У цьому контексті зростає необхідність у створенні веб-сервісу, який би забезпечував зручний перегляд, структурування та взаємодію з веб-коміксами. Такий сервіс повинен бути не лише технічно стабільним і адаптивним до різних пристроїв, але й враховувати сучасні підходи до UX/UI-дизайну, інтегрувати інструменти авторського завантаження та підтримувати функції персоналізованих рекомендацій. Реалізація такого проєкту потребує ефективного управління на всіх етапах – від планування до релізу та масштабування, з чітким урахуванням термінів, бюджету, ресурсів і ризиків. Тому дослідження процесів управління таким проєктом є актуальним завданням у сфері прикладного проєктного менеджменту.

Метою дослідження є створення комплексного та зручного веб-сервісу для перегляду веб-коміксів, управління яким здійснюється із застосуванням сучасних методів та інструментів управління проєктами. Це передбачає не лише організацію самого процесу розробки, а й застосування гнучких методологій, оптимізацію ресурсів, ризик-менеджмент, а також використання візуального моделювання бізнес-процесів.

Предметом дослідження є методи управління ІТ-проєктами, спрямовані на створення цифрового продукту – зокрема процеси планування, реалізації, моніторингу та завершення проєкту з розробки веб-сервісу для коміксів. До цього належать засоби управління командною роботою, контроль виконання завдань, методи фінансового планування, визначення критичних точок проєкту та розрахунок економічної ефективності.

Об'єктом дослідження виступають процеси розробки та управління ІТ-проєктом, зокрема створення веб-платформи для читання та публікації веб-коміксів. Цей об'єкт охоплює весь життєвий цикл проєкту – від постановки задач до технічної реалізації та оцінки результатів.

У роботі використовуються різноманітні методи дослідження, серед яких: системний аналіз, SWOT-аналіз, метод експертного оцінювання, графічне моделювання процесів (BPMN, UML), елементи фінансового моделювання, а також елементи аналізу користувацького досвіду. Для управління розробкою використовуються принципи Agile та Scrum, з організацією роботи за допомогою спринтів, беклогів і інструментів візуалізації прогресу [10, 21, 34, 30, 39–41].

Основними **завданнями** даної роботи є:

– проведення аналізу предметної області, пов'язаної з ринком веб-коміксів;

– визначення проблем, цілей та можливих альтернатив реалізації проєкту;

– опис веб-сервісу, його функціональних завдань та очікуваних результатів;

– розробка життєвого циклу проєкту та декомпозиція робіт;

– побудова організаційної структури проєкту, формування плану управління командою та розподіл відповідальності між учасниками;

– створення календарного плану реалізації веб-платформи;

– розрахунок вартості реалізації MVP-версії продукту;

– аналіз ризиків і формування стратегії їх мінімізації;

Наукова **новизна** роботи полягає у практичній реалізації комплексного підходу до управління розробкою спеціалізованого веб-сервісу у сфері цифрових

медіа, де поєднуються технічна розробка, бізнес-аналіз, управління командою та економічне обґрунтування. Особливістю дослідження є адаптація методології Scrum до умов індивідуальної розробки та візуалізація ключових процесів на основі реального проєкту.

Практична значимість дослідження полягає у створенні прикладної методики управління повноцінним циклом розробки IT-продукту — веб-сервісу для перегляду веб-коміксів — із використанням інструментів сучасного проєктного менеджменту. Розроблені підходи до планування, бюджетування, організаційного структурування та управління ризиками можуть бути застосовані в рамках реальних стартапів або внутрішніх продуктів у креативних індустріях. Отримані результати мають потенціал для впровадження як методичних рекомендацій у малих та середніх IT-компаніях, що працюють у сфері розваг, мультимедіа чи видавництва.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ТА ОБҐРУНТУВАННЯ НЕОБХІДНОСТІ ПРОЄКТУ

1.1 Проведення аналізу предметної галузі

Упродовж останніх років ринок цифрових коміксів переживає динамічне зростання, що, на думку дослідників Grand View Research, підтверджується обсягом у $4,69 млрд у 2023 році. Зростаюча популярність смартфонів і планшетів, разом із вдосконаленням платіжної інфраструктури, сприяє зростанню попиту на комікси у цифровому форматі. Основними факторами, що сприяють цьому зростанню, є активне впровадження смартфонів і планшетів, покращення платіжних інструментів, а також постійно зростаюча потреба користувачів у миттєвому та персоналізованому доступі до розважального контенту.

У глобальному контексті лідерами споживання цифрових коміксів залишаються США, Південна Корея та Японія. Ці країни мають розвинену інфраструктуру цифрових платформ, включаючи такі сервіси, як Webtoon, Tapas, KakaoPage, що охоплюють мільйони користувачів щомісяця [15–19]. У Європі, Південно-Східній Азії та Латинській Америці також спостерігається динамічне зростання популярності webcomic-платформ. Однак тут ринок є більш фрагментованим, із менш концентрованими гравцями та меншою часткою локалізованого контенту. За результатами дослідження Statista, відзначено зростання кількості активних користувачів цифрових платформ для читання коміксів на 12% за останні два роки.

Веб-комікси — це унікальний цифровий медіаформат, який поєднує візуальну оповідь, інтерфейсні рішення та мультимедійні технології. З технічної точки зору, сучасні платформи для веб-коміксів активно використовують адаптивні вебтехнологіїдля забезпечення кросплатформної доступності. Особливою популярністю користуються Progressive Web Apps (PWA), які

надають можливість офлайн-доступу, push-сповіщень, кращої продуктивності та взаємодії з користувачем.

Іншим ключовим трендом є застосування штучного інтелекту, зокрема рекомендаційних систем на основі машинного навчання, що дозволяють формувати персоналізовану стрічку коміксів згідно з інтересами користувача. Цей підхід був досліджений у роботах науковців із ACM Digital Library, де наголошується, що подібні алгоритми підвищують користувацьку лояльність і збільшують час, проведений на платформі, на 30–40%.

Інтерфейс та візуальні рішення також грають ключову роль. За даними IEEE Xplore, інтуїтивна навігація, гейміфікація (системи досягнень, рейтинги, підписки) та підтримка темної теми можуть значно покращити досвід користувача та збільшити середню тривалість сесії в 1,5 раза.

Розробка веб-сервісів у сфері digital entertainment пов’язана з низкою управлінських викликів, які стосуються як технологічної, так і організаційної частини проєктів. Серед основних проблем:

- Високий рівень конкуренції: Платформи борються за увагу користувача, тож час на запуск MVP суттєво обмежений.
- Бюджетні обмеження: Команда стартапу часто має обмежені ресурси, але очікується реалізація функціоналу, порівнюваного з лідерами ринку.
- Гнучкість вимог: Запити користувачів змінюються, тому команда має постійно адаптувати продукт, застосовуючи ітеративний підхід.
- Правове регулювання: Ліцензування контенту — одна з найболючіших тем. Необхідно юридично захистити контент авторів і враховувати міжнародне авторське право.

У відповідь на ці виклики в сучасному менеджменті використовують гнучкі методології — Agile-підходи, такі як Scrum і Kanban, що дозволяють швидко реагувати на зміни. Гібридні моделі, які поєднують класичне планування з гнучкістю DevOps, показують ефективність в умовах постійних змін.

Формування кросфункціональних команд, ролі продакт-менеджера, UX-дизайнера, контент-модератора — усе це елементи організаційної структури сучасного digital-проєкту.

Попри динаміку IT-індустрії, ринок цифрових коміксів в Україні ще не сформувався як повноцінний сегмент. На національному рівні існують лише окремі ініціативи — UAComix, MAL'OPUS, Megogo Comics, — що переважно виконують роль контент-провайдерів, але не інтегрованих платформ з розвиненим функціоналом. Вітчизняні автори часто публікують свої роботи на закордонних платформах, втрачаючи потенційний дохід і зв'язок з українською аудиторією.

Це створює унікальну можливість для запуску україномовної платформи, орієнтованої як на локального користувача, так і на міжнародну аудиторію. Потенційна концепція: веб-сервіс із соціальним функціоналом, системою підписки, рейтингами, краудфандинговими механізмами для авторів, мобільною адаптацією та підтримкою Web3-технологій (NFT-комікси).

1.2 Стан і тенденції розвитку веб-сервісів для перегляду коміксів

Веб-сервіси для читання коміксів, як окремий сегмент цифрової економіки, є результатом еволюції традиційного видавництва в умовах медіаконвергенції. У своїй основі ці сервіси поєднують мультимедійну репрезентацію контенту, інтерактивну взаємодію з користувачем і розподілену модель публікації. Переважна більшість сучасних платформ уже не обмежуються виключно функцією перегляду, а формують цілісну екосистему: хмарне сховище коміксів, системи рекомендацій, інструменти взаємодії з авторами, механізми комерціалізації контенту та API-інтеграції з зовнішніми сервісами.

Найвідоміші глобальні сервіси — Webtoon (Naver), Tapas Media, ComiXology (Amazon) та Manga Plus (Shueisha) — демонструють високий ступінь технологічної зрілості. Їхня архітектура базується на мікросервісному підході,

що забезпечує масштабованість, адаптивність до навантажень і підтримку глобального користувацького трафіку. Наприклад, Webtoon надає понад 100 мільярдів переглядів на рік (дані Naver, 2023), а кількість активних користувачів перевищує 85 млн щомісяця.

З технічної точки зору, сервіси такого типу впроваджують рішення для адаптивного рендерингу зображень, що дозволяє оптимізувати завантаження та зберігати якість на мобільних пристроях. Зокрема, використання формату WebP, динамічне масштабування зображень і серверний кешинг — це вже усталена практика. Окрему увагу приділяють оптимізації latency-часу, що критично важливо для утримання користувача в контексті швидкого споживання візуального контенту.

Окрім технологічного аспекту, ключовим чинником конкурентоспроможності веб-сервісів є якість користувацького досвіду (UX). Згідно з даними Interaction Design Foundation (2022), користувачі більш схильні залишатися на платформі, що має зрозумілу інформаційну архітектуру, гнучкий механізм фільтрації контенту, інтеграцію з соціальними мережами та можливість формування персоналізованих колекцій [8].

Унікальним елементом стало використання соціального читання, де комікси супроводжуються коментарями, оцінками, форумах за розділами, що створює ефект “живої аудиторії” навколо кожного твору. Це, у свою чергу, впливає на метрики залучення: середня тривалість сесії на Webtoon зростає до 28 хвилин (за даними Sensor Tower, 2023), що є нетипово високим показником для мобільного формату.

Сучасні платформи дедалі активніше впроваджують гібридні моделі монетизації. До них належать freemium-модель (частина контенту безкоштовна, частина — за передплатою), внутрішньоігрова валюта, система мікроплатежів за розділи та краудфандингові ініціативи на кшталт Tapas Ink або Patreon-інтеграцій. Такі моделі не тільки генерують прибуток, а й формують нову

динаміку авторських відносин, де автор стає партнером платформи в моделі revenue-sharing.

Інституційні дослідження вказують, що успішна авторська екосистема базується на прозорих ліцензійних умовах, автоматизованому copyright-трекінгу та можливості масштабування дистрибуції. Ці фактори є ключовими для залучення незалежних творців, які складають значну частину контентного портфеля сучасних платформ (до 60% на Tapas Media, 2023).

Вектор розвитку веб-сервісів дедалі частіше спрямований у бік персоналізованої інтеграції на основі машинного навчання. Алгоритми рекомендацій, побудовані на поведінковому аналізі (collaborative filtering, reinforcement learning), дозволяють системі адаптувати контентну стрічку в реальному часі. Згідно з дослідженням ACM SIGCHI (2023), це підвищує конверсію у підписку до 45%.

Ще один помітний тренд — впровадження елементів Web3, зокрема використання NFT для токенізації унікальних випусків, прав доступу до преміум-матеріалів або участі в DAO (децентралізованих автономних організаціях) для фінансування проєктів. Водночас інтеграція з метавсесвітами (як-от The Sandbox або Spatial.io) відкриває нові формати візуального сторітелінгу — 3D-комікси, інтерактивні елементи, сюжетна навігація у віртуальних середовищах.

1.3 Аналіз методів оцінки впливів оточення ІТ-проєкту та функціонального призначення окремих його частин

1.3.1 Проведення аналізу внутрішнього середовища

Поглиблений аналіз внутрішнього середовища є важливою складовою стратегічного планування запуску веб-сервісу для перегляду коміксів. Такий аналіз дає змогу не лише ідентифікувати поточні сильні та слабкі сторони проєкту, а й сформувати адаптивну модель розвитку, що враховує обмежені ресурси, командний потенціал, технічні можливості та маркетингову стратегію.

Даний підхід дозволяє компанії підвищити ефективність ухвалення управлінських рішень і зменшити ризики, пов’язані з невизначеністю ринку. В межах цієї оцінки особливу увагу приділено п’яти ключовим аспектам: кадровому потенціалу, технічній інфраструктурі, внутрішнім бізнес-процесам, маркетинговим інструментам і управлінській компетенції.

Успішна реалізація платформи передбачає залучення висококваліфікованої мультидисциплінарної команди, яка охоплює фахівців з різних галузей: UI/UX-дизайнерів, frontend- і backend-розробників, спеціалістів із цифрового маркетингу, продуктових менеджерів, аналітиків даних, DevOps-інженерів, а також тестувальників та технічних письменників. Додатковою перевагою є наявність досвіду створення PWA-додатків, роботи з адаптивними фреймворками (React.js, Vue.js, Svelte), а також впровадження сучасних методів CI/CD для пришвидшення життєвого циклу розробки. Здатність до швидкого розгортання MVP на базі хмарних сервісів забезпечує гнучкість та економічність стартових етапів. У той же час, виявлено суттєвий кадровий дефіцит у сфері машинного навчання, що обмежує потенціал для розвитку розумних рекомендаційних систем, автоматизації модерації контенту та персоналізації досвіду користувача. Брак спеціалістів з кібербезпеки також становить виклик для забезпечення цілісності платформи.

IT-архітектура сервісу базується на принципах модульності, масштабованості та орієнтації на відкриті технології. Основна серверна логіка реалізована з використанням мікросервісної архітектури, що полегшує масштабування й оновлення окремих функціональних блоків без зупинки системи. Застосування контейнеризації (Docker) і систем оркестрації (Kubernetes) дозволяє автоматизувати розгортання, балансування навантажень і моніторинг. Проте нинішній рівень технічного забезпечення має певні недоліки: зокрема, відсутність резервного хостингу, недостатній рівень кіберзахисту (немає WAF, системи запобігання DDoS-атакам), а також обмежений моніторинг

продуктивності у реальному часі. Для забезпечення надійності та безпеки необхідно інвестувати в інструменти на кшталт Grafana, Prometheus, Sentry та Cloudflare.

Внутрішня структура компанії характеризується невеликою командою зі спрощеною ієрархією, що дає змогу швидко ухвалювати рішення, стимулювати ініціативність і підвищувати командну згуртованість. Команда працює за принципами Agile, зокрема Scrum і Kanban, що дозволяє реалізовувати швидкі ітерації, гнучко реагувати на зворотний зв'язок користувачів, а також удосконалювати продукт у режимі реального часу. Проте слабкою ланкою залишається відсутність формалізованих бізнес-процесів, таких як управління ризиками, документування технічних завдань, стандартизація контролю якості та ведення внутрішньої аналітики. У подальшому ці недоліки можуть перешкоджати масштабуванню та інтеграції з зовнішніми партнерами.

Обмеженість бюджету на рекламу змушує команду зосереджуватись на використанні недорогих, але ефективних інструментів цифрового маркетингу. Серед основних каналів просування — SEO-оптимізація контенту, таргетована реклама у Facebook, Instagram та TikTok, а також співпраця з тематичними YouTube-каналами й телеграм-спільнотами. Значну роль відіграє краудфандинг, що не лише забезпечує фінансову підтримку, а й створює ядро лояльної аудиторії ще до офіційного запуску. Активне формування ком'юніті через Discord, Reddit або Telegram сприяє зростанню органічної залученості користувачів. Водночас, відсутність усталеного бренду, кейсів успішного використання, а також референцій у професійних медіа обмежує можливості для B2B-співпраці та масштабних PR-кампаній.

Керівна команда проєкту має базовий досвід управління IT-стартапами, зокрема в частині розробки продуктової стратегії, бюджетування, управління життєвим циклом програмного забезпечення та ведення переговорів із потенційними партнерами. Сильним боком є вміння швидко реагувати на зміни,

формувати гіпотези й тестувати їх у стислий термін. Утім, відсутність досвіду масштабування продукту на велику аудиторію, керування багатофункціональними командами, а також недостатня компетентність у сфері управління даними й аналітики може створити перепони під час виходу на міжнародний рівень або при збільшенні користувацької бази до десятків тисяч активних користувачів.

У підсумку, внутрішній аналіз демонструє наявність сильного кадрового та технічного потенціалу, високої адаптивності команди й готовності до ітеративного вдосконалення продукту. Водночас виявлені структурні та ресурсні обмеження вказують на необхідність чіткої стратегії нарощування компетенцій, стандартизації процесів і поступового посилення управлінської вертикалі. Лише за умови усунення цих вразливостей платформа зможе повною мірою реалізувати свій потенціал у конкурентному середовищі цифрових розваг.

1.3.2 Проведення аналізу зовнішнього середовища

Ретельне вивчення зовнішнього середовища — ключовий етап у процесі стратегічного планування проєкту створення цифрової платформи для перегляду веб-коміксів. Від того, наскільки глибоко та системно буде проведений аналіз зовнішніх чинників, залежить успішність прийняття управлінських рішень, формування унікальної ціннісної пропозиції та забезпечення довгострокової конкурентоспроможності платформи. У цьому розділі здійснено аналіз зовнішнього середовища за допомогою двох усталених методик стратегічного менеджменту: PEST-аналізу (що охоплює політичні, економічні, соціокультурні та технологічні аспекти) та моделі п’яти сил конкуренції Майкла Портера, яка дозволяє оцінити інтенсивність конкуренції та привабливість галузі.

Україна останніми роками активно працює над створенням сприятливого клімату для розвитку IT-сектору та цифрових ініціатив. Одним із вагомих інструментів у цьому напрямі є спеціальний правовий режим «Дія.City», який

пропонує пільгове оподаткування, гнучке трудове законодавство та підтримку стартапів. Це створює передумови для легальної діяльності цифрових платформ і залучення інвестицій. Однак водночас залишається невирішеним питання ефективного захисту прав інтелектуальної власності. Незважаючи на гармонізацію українського законодавства із європейськими нормами, механізми боротьби з піратством мають обмежену ефективність. Для цифрової платформи, яка планує легально поширювати авторський контент, критично важливо впроваджувати інноваційні рішення, наприклад, блокчейн-ідентифікацію авторства, смарт-контракти для ліцензування творів тощо. Це може не лише забезпечити юридичний захист, а й підвищити довіру авторів до сервісу.

Незважаючи на складну макроекономічну ситуацію в Україні, включаючи інфляцію, військові ризики та нестабільність доходів населення, цифровий ринок розваг виявляє стійкість до криз і демонструє зростання. Цей феномен пояснюється антициклічністю попиту на доступний дозвіллєвий контент: у періоди стресу та обмежених можливостей фізичної активності споживачі шукають недорогі способи розваги, і веб-комікси чудово відповідають цій потребі. Аналітичні звіти консалтингових компаній Deloitte та PwC засвідчують позитивну динаміку в сегменті цифрових медіа. Платформа, що позиціонує себе як український веб-сервіс для перегляду коміксів, може скористатися зростанням онлайн-аудиторії, а також розвитком платіжної інфраструктури (зокрема систем донатів і краудфандингу), що відкриває нові бізнес-моделі для монетизації.

В останнє десятиліття в Україні спостерігається активне формування цифрової культури, орієнтованої на локальний контент. Підвищення статусу української мови, зацікавлення молоді у вітчизняному медіапродукті, а також активне поширення жанрів на межі літератури й візуального мистецтва — таких як webtoon, манґа, графічні романи — формують сталий попит на інноваційні платформи для їх споживання. Зміни в поведінці користувачів також грають на користь розвитку подібних сервісів: молодь прагне до взаємодії, залишає

рецензії, створює фан-контент, активно бере участь у формуванні спільнот. Це відкриває шлях до реалізації функціоналу, що підвищує залученість: рейтинги, коментарі, можливість прямої комунікації з авторами, інтегровані форуми тощо. Така соціальна динаміка є потужним рушієм популярності подібних платформ.

Швидка еволюція цифрових технологій змінює вимоги до користувацького досвіду. Веб-сервіси нового покоління повинні бути не лише функціональними, а й гнучкими та персоналізованими. Адаптивний дизайн, миттєве завантаження контенту, офлайн-доступ, темна тема, змінні шрифти, вбудовані перекладачі — це вже базові очікування сучасного користувача. Додатково, стрімкий розвиток технологій віртуальної та доповненої реальності відкриває нові формати подачі контенту: інтерактивні сторітелінги, AR-комікси, 3D-інтерфейси. Інтеграція з соціальними мережами, хмарними сервісами та мобільними гаманцями значно розширює потенціал платформи як цифрової екосистеми. Врахування всіх цих аспектів на етапі проєктування — запорука створення конкурентоспроможного продукту.

Конкурентне середовище за М. Портером:

- Загроза нових гравців. Високий рівень доступності технологій та відкритих фреймворків робить технічну реалізацію платформи досяжною навіть для малих команд. Водночас створення бренду, залучення авторів і формування лояльної аудиторії потребує значних часових і маркетингових ресурсів, що знижує ризик масового входу нових гравців.
- Вплив постачальників. У випадку цифрової платформи для веб-коміксів залежність від зовнішніх постачальників ресурсів є незначною, адже основний актив — контент — створюється спільнотою користувачів або в межах партнерств з авторами.
- Влада споживачів. Користувачі мають широкий вибір альтернатив — від міжнародних платформ (Tapas, Webtoon) до соціальних мереж, де автори публікують твори напряму. Це формує високі очікування до сервісу: зручність,

якість контенту, унікальні можливості. Без чіткої ціннісної пропозиції платформа ризикує втратити лояльність аудиторії.

- Конкуренція між існуючими гравцями. Хоча в українському сегменті відсутні домінуючі гравці, на глобальному рівні ринок доволі насичений. Головні конкуренти орієнтовані на англомовну аудиторію, тому україномовна платформа має шанси зайняти нішу за рахунок локалізації та культурної близькості.
- Загроза замінників. Споживачі цифрового контенту легко перемикаються між форматами: відео, стрімінг, ігри, соціальні мережі. Щоб утримати увагу, платформа має створити особливий досвід — шляхом впровадження гейміфікації, NFT-підтримки, кастомізованих профілів, ексклюзивного контенту тощо.

У рамках стратегічного аналізу було побудовано структурну модель цілей та проблем, що ілюструє взаємозв'язки між основними викликами, на які спрямовано реалізацію веб-платформи.

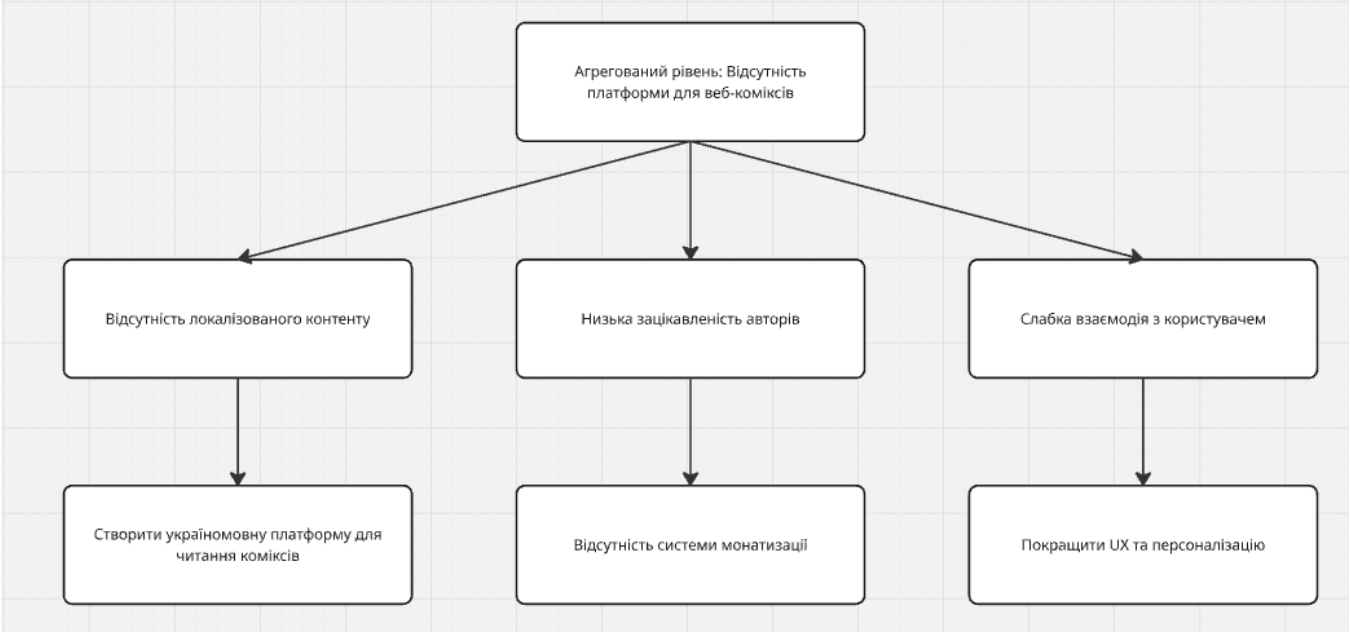

Рис. 1.1. Структурна модель цілей та проблем проєкту

Таким чином, аналіз зовнішнього середовища виявляє сприятливі умови для запуску платформи, однак водночас вказує на високу динамічність ринку та необхідність чіткого позиціонування. Основними стратегічними орієнтирами мають стати: адаптація до очікувань локальної аудиторії, впровадження сучасних технологій, активна взаємодія з авторами та побудова довготривалої спільноти навколо контенту.

1.3.3 Проведення інвестиційних досліджень

Інвестиційна привабливість стартапу у сфері веб-коміксів розглядається як багатовимірна система, що охоплює сукупність фінансових, стратегічних, ринкових, інституційних і соціокультурних параметрів. У світлі стрімкого розвитку цифрових технологій та підвищеного попиту на контент у форматі «on demand», інноваційні рішення на перетині мистецтва, медіа і технологій стають дедалі більш привабливими для потенційних інвесторів. Особливу цінність мають проєкти, що не лише генерують дохід, а й формують довгострокову додану вартість через розвиток інтелектуальної власності, розширення креативної екосистеми та культурну репрезентацію.

Проєкт створення веб-сервісу для перегляду веб-коміксів виступає саме таким прикладом: він є не просто інтернет-продуктом, а платформою, що інтегрує елементи культурної індустрії, цифрової дистрибуції та соціальної взаємодії між авторами та читачами. З огляду на молодість українського ринку у відповідному сегменті, можливості для нішевого позиціонування є надзвичайно значними, що створює сприятливі умови для інвестування на ранніх етапах життєвого циклу стартапу.

Було здійснено комплексне картування потенційних джерел фінансування, які умовно поділяються на п’ять категорій:

Таблиця 1.1

Комплексне картування

Джерело фінансування	Приклади	Основні переваги
Стартап-інкубатори	Sector X, 1991 Accelerator, Startup Depot	Менторство, стартовий капітал, експертна підтримка
Міжнародні гранти	Horizon Europe, Creative Europe, House of Europe	Безповоротне фінансування, сприяння інтеграції у європейський ринок
Венчурні фонди	U.Ventures, Horizon Capital, SMRK VC Fund	Значні обсяги інвестицій, стратегічне партнерство
Краудфандинг	Kickstarter, Indiegogo, Спільнокошт	Тестування попиту, формування аудиторії
Angel investors	IT-підприємці, діячі культури	Гнучкість, персоналізовані умови співпраці

Таблиця 1.2

Інвестиційна стратегія

Етап розробки	Основні дії	Джерела фінансування
Запуск MVP	Розробка ядра продукту, UX-дизайн, базовий функціонал	Власні кошти, гранти, інкубатори
Валідація гіпотези	Збір фідбеку, тестування функцій, аналіз сценаріїв	Angel investors, краудфандинг
Масштабування	Платні сервіси, реклама, підтримка авторів, мобільні додатки	Венчурні фонди, стратегічні партнери

Таблиця 1.3

Фінансова модель

Тип доходу	Конкретні приклади
Прямі доходи	Платні підписки (freemium/premium), мікротранзакції
Непрямі доходи	Ліцензування, B2B партнерства, рекламні інтеграції

Таблиця 1.4

Фрагмент SWOT-аналізу

Сильні сторони (Strengths)	Слабкі сторони (Weaknesses)
- Інноваційність концепції у локальному ринку	- Відсутність гучного бренду або впізнаваності
- Технічна гнучкість та масштабованість архітектури	- Високий рівень конкуренції в глобальному середовищі
- Орієнтація на культурну та соціальну цінність продукту	- Залежність від початкового фінансування для повноцінного запуску
- Можливість монетизації через кілька моделей (B2C, B2B, API тощо)	- Недостатність історичних даних для точного прогнозування зростання

Таблиця 1.5

Фрагмент SWOT-аналізу

Можливості (Opportunities)	Загрози (Threats)
- Залучення грантового та венчурного фінансування	- Економічна нестабільність в регіоні
- Підключення українських авторів, студій, локалізацій	- Швидка зміна технологічних трендів або платформи
- Вихід на міжнародні ринки через англомовну адаптацію	- Культурні бар’єри, зокрема при ліцензуванні контенту з інших ринків
- Розвиток смарт-ліцензування, NFT та цифрових продуктів	- Потенційна поява аналогічних сервісів з більшим бюджетом

Для доповнення оцінки середовища функціонування проєкту доцільно провести PEST-аналіз, який дозволяє проаналізувати вплив основних макрочинників – політичних, економічних, соціальних та технологічних – на розробку та впровадження веб-сервісу “MangaView”.

Таблиця 1.6

PEST-аналіз

Фактор	Опис впливу
Політичний	Наявність правового режиму “Дія.City”; підтримка цифрових ініціатив; загрози, пов’язані з воєнним станом
Економічний	Попит на недорогі цифрові розваги; нестабільність економіки; обмеження інвестування в стартапи
Соціальний	Формування культури споживання україномовного контенту; популярність web-коміксів серед молоді
Технологічний	Активний розвиток вебтехнологій (SPA, PWA, Web3); поширення мобільного доступу; потенціал для інтеграції NFT і смарт-контрактів

Консервативна оцінка передбачає точку беззбитковості через 24–30 місяців за умови стабільного приросту користувацької бази в межах 5–8% щомісяця. Альтернативні сценарії включають експансію на англомовні ринки, запуск white-label платформ, створення власного медіахолдингу. Гнучкість моделі дозволяє адаптувати бізнес до зовнішніх викликів, у тому числі валютних коливань, зміни поведінки споживача, регуляторних обмежень.

Завдяки поєднанню інноваційної концепції, культурної унікальності, технологічної адаптивності й наявності перспектив виходу на міжнародні ринки, стартап із веб-коміксів володіє високим потенціалом інвестиційної привабливості. Успішна реалізація стратегії фінансування залежить від чіткої дорожньої карти, міждисциплінарної команди та здатності оперативно реагувати на зміну ринкових умов, зберігаючи при цьому автентичність і цінність продукту для кінцевого користувача.

1.4 Постановка задач дослідження

Після детального дослідження предметної галузі, що охоплює не лише еволюцію ринку веб-сервісів для перегляду коміксів, але й комплексну оцінку динаміки його трансформацій під впливом цифрової конвергенції,

технологічного прогресу та змін у споживчій поведінці, виникає нагальна потреба у науково вивіреному формулюванні задач дослідження. Системний підхід до формалізації дослідницьких цілей дозволяє не лише забезпечити методологічну узгодженість подальших етапів магістерської роботи, але й сформувати теоретико-прикладну базу для критичного осмислення вибраної стратегії управління цифровими проєктами в контексті створення інноваційного інформаційного продукту.

Основна дослідницька мета полягає у концептуалізації та побудові високоефективної, масштабованої та адаптивної моделі управління ІТ-проєктом, спрямованим на розробку веб-сервісу для перегляду цифрових коміксів, з урахуванням складної багаторівневої взаємодії між користувачем, інтерфейсом і архітектурною логікою системи.

Для досягнення поставленої мети доцільно окреслити комплекс конкретних науково-дослідних задач, що охоплюють міждисциплінарні аспекти управління ІТ-проєктами, аналізу цифрового контенту, інженерії програмного забезпечення та соціотехнічної інтеграції:

- Провести систематизований і критичний аналіз теоретико-методологічних засад управління складними ІТ-проєктами, з особливим акцентом на цифрові екосистеми, методології розробки користувацько-орієнтованих веб-сервісів, а також еволюцію стандартів у сфері гнучких та змішаних моделей управління.
- Ідентифікувати специфічні параметри, що формують соціотехнічний профіль проєктів у сфері дистрибуції цифрового візуального контенту, включаючи ризики культурної релевантності, нормативно-правові бар'єри, етичні дилеми та динаміку взаємодії зі стейкголдерами.
- Сформулювати методологічну парадигму вибору та комбінування моделей управління проєктом, на основі мультикритеріального аналізу ефективності, включаючи параметри ресурсної стійкості, когнітивного

навантаження команди, інтенсивності ітераційного циклу та ступеня невизначеності середовища проєкту.

- Розробити структурно-функціональну модель управління ключовими аспектами життєвого циклу проєкту — зокрема ресурсним забезпеченням, ризик-менеджментом, контролем якості, комунікаційною інфраструктурою — на основі інструментів цифрової трансформації управління.
- Сконструювати комплексну систему показників оцінки ефективності управління IT-проєктом, з урахуванням як кількісних, так і якісних метрик: рівень досягнення цілей, ефективність спринтів, інтеграція зворотного зв’язку, гнучкість архітектури, а також задоволеність кінцевих користувачів.
- Опрацювати рекомендації для масштабування, стійкої експлуатації та еволюційного розвитку проєктів аналогічного типу, інтегруючи результати аналізу best practices, емпіричні кейси та оцінку потенціалу впровадження DevOps-культури.
- Закласти основи для подальшого наукового розгортання дослідження, сформулювавши універсальні моделі, алгоритми прийняття управлінських рішень та інструменти аналітичної підтримки, що будуть релевантні в умовах постійних технологічних змін.

У підсумку, зазначене структуроване формулювання дослідницьких задач не лише визначає логіку побудови подальших розділів магістерської роботи, а й створює підґрунтя для формування ефективного методичного інструментарію, здатного забезпечити комплексне впровадження управлінських рішень у контексті реалізації веб-сервісу нового покоління.. Саме коректна постановка задач виступає базовою передумовою для забезпечення цілісності, логічної послідовності та наукової обґрунтованості подальшого ходу магістерської роботи.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ХАРАКТЕРИСТИК ПРОЄКТУ ТА ПРОЦЕСІВ УПРАВЛІННЯ

2.1 Характеристика проєкту створення веб-сервісу для перегляду коміксів

Розробка веб-сервісу для перегляду веб-коміксів є прикладом інноваційного цифрового продукту, що орієнтований на широку онлайн-аудиторію, зокрема на молодь, шанувальників графічного мистецтва, любителів манґи, вебтунів та інтерактивного сторітелінгу. Такий проєкт має на меті створення комфортного, інтуїтивно зрозумілого та функціонального середовища для ознайомлення з коміксами в електронному форматі, що повністю відповідає сучасним тенденціям диджиталізації культури споживання візуального контенту.

Ключова концепція проєкту полягає в забезпеченні користувачів адаптивною платформою, яка дозволяє не лише читати цифрові комікси на різних пристроях (зокрема на смартфонах і планшетах), а й взаємодіяти з контентом через персоналізовані добірки, системи рекомендацій, залишення відгуків, вподобань, коментарів та можливість брати участь у житті платформи через ігрові механіки, рейтинги й досягнення. Також передбачається створення інтегрованої системи self-publishing, яка дозволить авторам самостійно публікувати свої твори, що виводить платформу за межі лише споживання і наближає її до творчої екосистеми.

Проєкт охоплює повний життєвий цикл IT-продукту — від первинного аналізу ринку та валідації ідеї до розробки прототипу, реалізації мінімально життєздатного продукту (MVP), послідовного масштабування функціональності, впровадження маркетингових стратегій та організації технічної та контентної підтримки. Підсумковий результат має забезпечити створення стабільної, конкурентоспроможної та привабливої з погляду користувача платформи,

здатної витримати навантаження зростаючої аудиторії та гнучко адаптуватися до її потреб.

На момент написання дипломної роботи проєкт перебуває на початковій стадії, що передбачає детальне дослідження зовнішнього та внутрішнього середовища, побудову архітектурної моделі рішення, формулювання бізнес-вимог, визначення цілей та задач, розробку проектного плану, таймлайну та оцінку ресурсів. В реалізації буде використано гнучкий підхід до управління розробкою — зокрема методологію Scrum, яка дозволяє ефективно проводити ітераційні спринти, адаптуватися до зворотного зв’язку, мінімізувати ризики та підвищити якість продукту на кожному етапі.

Серед ключових характеристик проєкту можна виокремити наступні аспекти:

- Цільова аудиторія: підлітки та молодь віком від 14 до 30 років, активні користувачі смартфонів, соціальних мереж, шанувальники візуального сторітелінгу, цифрового мистецтва та онлайн-контенту;
- Основні функціональні модулі: інтерфейс для перегляду коміксів (в тому числі вертикальне гортання для вебтунів), особисті кабінети, персональні добірки, система рекомендацій, підтримка self-publishing, функції соціальної взаємодії (коментарі, вподобання, підписки);
- Технічна платформа: мультиплатформенність з фокусом на мобільні пристрої (PWA, адаптивна верстка), використання сучасних веб-фреймворків (React, Vue), інтеграція з хмарними сервісами для зберігання даних і масштабування (AWS, Firebase);
- Контентна стратегія: акцент на підтримку україномовного контенту та українських авторів, створення бібліотеки локалізованих або оригінальних творів, що сприятиме культурній самореалізації та підтримці національного продукту;

- Інструменти просування: інтеграція з соціальними мережами (Instagram, TikTok), наявність спільнот у месенджерахрос, використання краудфандингових платформ для залучення підтримки від спільноти;
- Модель запуску: реалізація MVP з обмеженим функціоналом і поступовим розширенням на основі аналізу користувацької поведінки, фокус на A/B-тестуванні та регулярному покращенні юзабіліті.

Загалом, проєкт поєднує в собі технічну інноваційність, маркетингову гнучкість, увагу до культурних цінностей та актуальність цифрових тенденцій. Він є не лише технічним рішенням, а й соціально орієнтованим інструментом, який сприятиме популяризації українського контенту, розвитку цифрової грамотності, підтримці творчих ініціатив та розширенню культурного простору у сфері цифрових медіа.

2.2 Ідентифікація процесів управління проєктом та планування робіт

2.2.1 Планування життєвого циклу проєкту

Проєкт створення веб-сервісу для читання цифрових коміксів розглядається як інтегрований комплекс взаємозалежних дій, спрямованих на проєктування, розробку та подальше впровадження сучасного цифрового продукту в галузі IT. Ефективне управління цим проєктом вимагає чітко структурованої системи процесів, яка дозволяє координувати ресурси, часові рамки та функціональні очікування. З метою досягнення очікуваних результатів у рамках обмеженого бюджету, часу та необхідного рівня якості, доцільно застосовувати системний підхід до управління проєктом, що базується на рекомендаціях стандарту PMBOK (Project Management Body of Knowledge) і адаптується до гнучких методологій (Agile, Scrum).

Проєктна діяльність у межах створення веб-сервісу організована згідно з п’ятьма основними групами процесів, що дозволяють структурувати роботу та забезпечити її послідовність і контрольованість на всіх етапах:

- Ініціація: визначається стратегічна доцільність розробки сервісу. Проводиться аналіз ринку та цільової аудиторії, визначаються потенційні користувачі, формулюється бачення продукту, складається початкова документація, проводяться консультації зі стейкхолдерами, визначається зона відповідальності учасників команди, а також створюється реєстр зацікавлених сторін. Ініціація завершується формалізацією старту проєкту та затвердженням основних параметрів.
- Планування: охоплює деталізацію кожного етапу реалізації. Будується структура декомпозиції робіт (WBS), проводиться оцінка необхідних ресурсів (людських, технічних, часових), формується попередній бюджет, складається календарний графік із прив'язкою до ключових етапів (milestones), проводиться аналіз ризиків із визначенням шляхів мінімізації та створенням плану реагування. Також визначаються індикатори успішності (KPI) та критерії прийняття результатів. Особливу увагу приділяють управлінню якістю, комунікаціями, взаємодією з користувачами.
- Виконання: передбачає практичну реалізацію поставлених завдань. На цьому етапі розробляється архітектура системи, створюється MVP (мінімально життєздатний продукт), програмується функціонал (фронтенд та бекенд), відбувається інтеграція з базою даних, реалізується дизайн інтерфейсу. Процес розробки проходить через серії спринтів (ітерацій), із використанням Scrum-дошок, щоденних мітингів і періодичних переглядів продукту. Гнучкий підхід дає змогу оперативно враховувати зворотний зв'язок від користувачів і вносити необхідні зміни в реалізацію.
- Моніторинг і контроль: ця група процесів відповідає за постійне спостереження за перебігом виконання проєкту, фіксацію відхилень від плану, прийняття коригувальних рішень. Використовуються сучасні інструменти керування завданнями (Jira, Trello, Asana), аналітичні панелі, регулярні звіти. Визначається відповідність поточних результатів запланованим KPI,

виявляються нові ризики, проводиться повторна оцінка пріоритетів. Ефективна комунікація між членами команди сприяє швидкому реагуванню на проблеми та підтриманню проєкту у межах встановлених параметрів.

- Завершення: охоплює всі процеси, пов'язані з формалізацією закінчення робіт. До них належить публічний реліз готового веб-сервісу, складання технічної та експлуатаційної документації, презентація замовнику (або споживачам), впровадження механізмів подальшої підтримки та оновлень, формування фінального звіту. Завершення також включає оцінку досягнення початкових цілей проєкту та документування здобутого досвіду.

Життєвий цикл проєкту у даному випадку має фазову структуру, типовою для розробки цифрових продуктів, і складається з наступних етапів:

Таблиця 2.1

Життєвий цикл проєкту

Етап	Зміст	Основні результати
1. Передпроєктна фаза	Формування ідеї, аналіз ринку, опис потреб	Ініціалізаційний документ, бачення продукту
2. Планування	Постановка цілей, формування команди, розробка плану робіт	WBS, оцінка ресурсів, проєктна документація
3. Розробка MVP	Реалізація базового функціоналу, дизайн, тестування	Перша версія продукту, зворотний зв'язок від користувачів
4. Масштабування	Розширення функцій, інтеграція нових модулів, оптимізація UX/UI	Розширена версія платформи, стабільність роботи
5. Завершення та підтримка	Публічний запуск, збір статистики, оновлення та техпідтримка	Реліз продукту, документація, технічна підтримка

Кожна з цих фаз тісно корелює з відповідними групами процесів управління та дозволяє забезпечити контрольованість і прогнозованість реалізації. Планування робіт у даному контексті набуває не лише суто технічного, але й стратегічного значення, оскільки спрямоване на формування чіткої

дорожньої карти досягнення цілей проєкту, враховуючи як технологічні, так і ринкові обмеження та можливості. Завдяки цьому можливо ефективно поєднати бачення, ресурси та засоби реалізації в єдину логічну структуру керування цифровим продуктом.

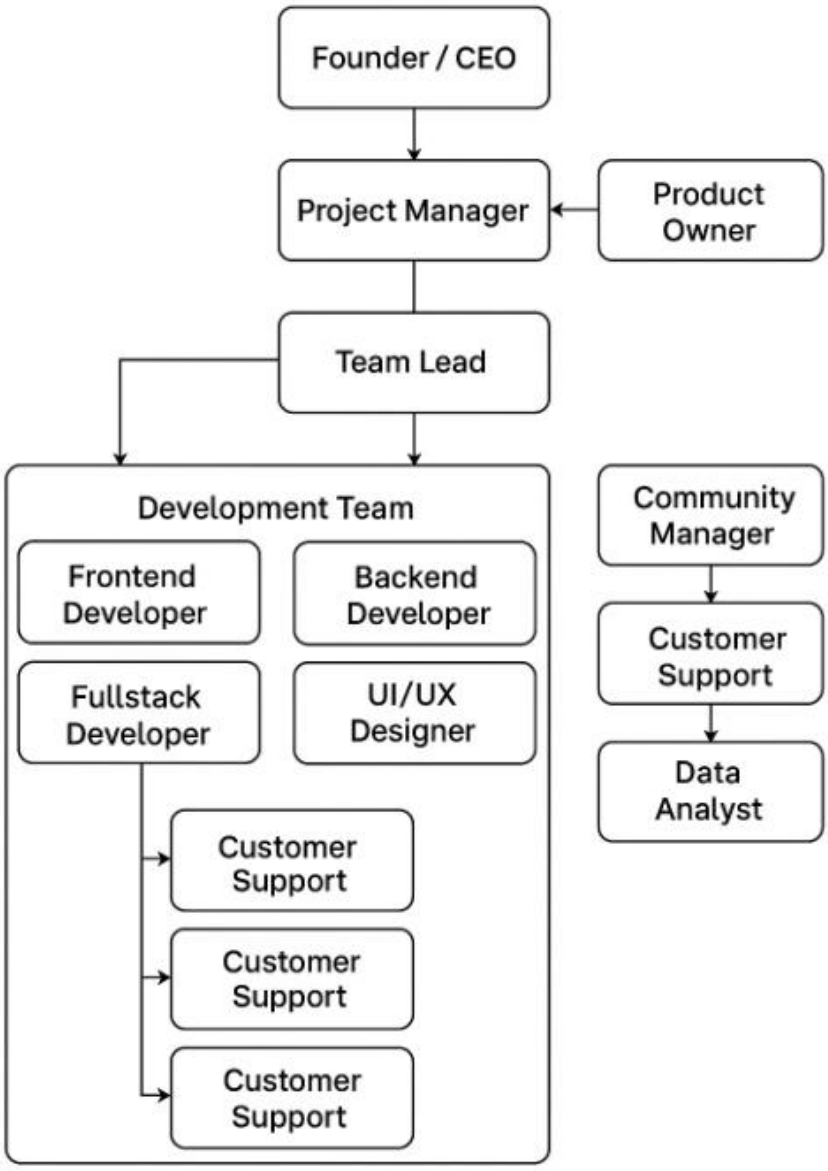

Рис. 2.1. Ієрархічна структура компанії

Для ефективного управління розробкою веб-сервісу з перегляду веб-коміксів, у рамках даного дослідження запропоновано ієрархічну структуру компанії (Рис. 2.1.) та організаційну модель управління проєктом (Рис. 2.2.), які

забезпечують чіткий розподіл відповідальності, контроль процесів і якісну взаємодію між учасниками.

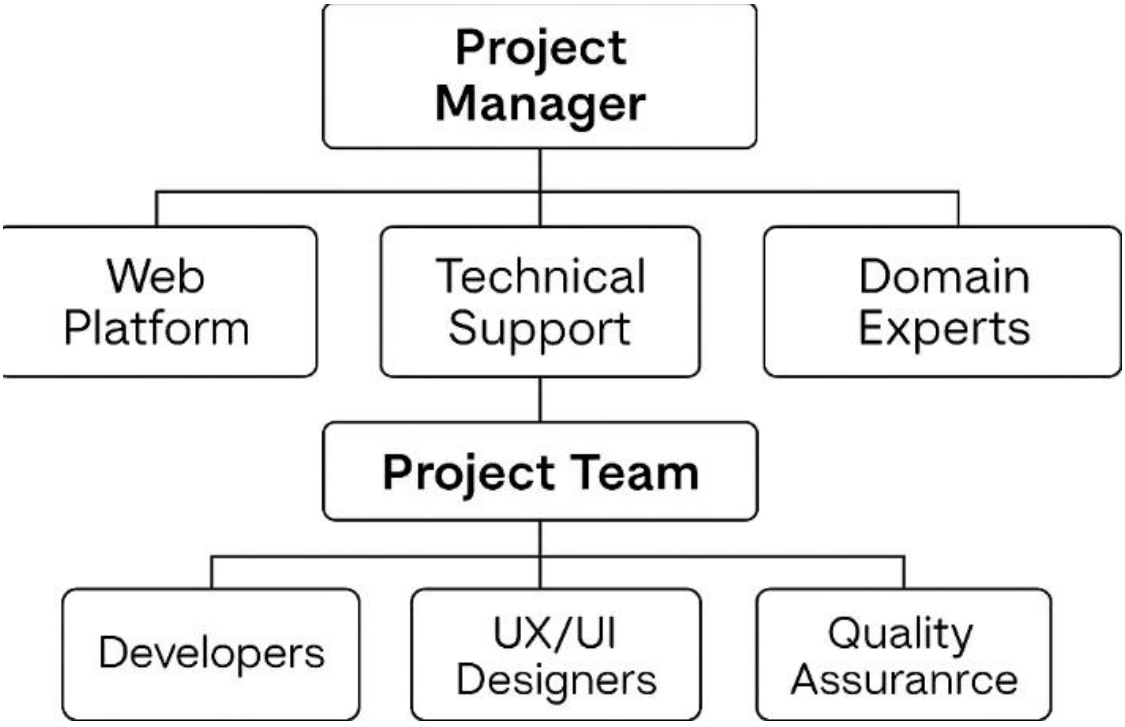

Рис. 2.2. Організаційна модель управління проєктом

2.2.2 Планування ресурсів проєкту

Планування ресурсного забезпечення є стратегічним інструментом у проєктному менеджменті, що безпосередньо впливає на досягнення очікуваних результатів у межах визначених часових, фінансових і якісних обмежень. У контексті розробки веб-сервісу для читання цифрових коміксів, цей процес передбачає глибокий аналіз та обґрунтовану оцінку всіх категорій ресурсів — людських, технічних, інформаційних і матеріальних — необхідних для реалізації проєктних цілей із дотриманням логіки життєвого циклу ПЗ.

До ключових елементів планування ресурсів належать:

- Ідентифікація потреб у ресурсах: здійснюється розподіл потреб відповідно до структури декомпозиції робіт, з визначенням типів, кількісних

параметрів і якісних характеристик ресурсів, необхідних на кожному етапі (від початкового прототипування до післяпроєктного супроводу). При цьому враховується міжфункціональна взаємозалежність завдань та потреба в гнучкому управлінні ресурсною базою.

- Формування проєктної команди: відбувається добір спеціалістів відповідного профілю із застосуванням критеріїв професійної компетентності, здатності до інтеграції в командну динаміку та адаптивності до гібридних умов праці. Описуються ролі, функціональні обов'язки та межі відповідальності в межах RACI-матриці або подібних моделей.
- Оцінка та балансування завантаженості: виконується моделювання обсягів трудових витрат із застосуванням метрик (наприклад, FTE або story points), що дозволяє оцінити потенційні «вузькі місця» у використанні людських ресурсів та розробити заходи превентивного реагування. Планування враховує фазовість розробки та гнучке масштабування залучення фахівців.
- Визначення та інтеграція технічних ресурсів: окреслюється перелік технологій, інфраструктурних платформ (у тому числі CI/CD-процеси), засобів контролю версій, середовищ розробки, хмарних сервісів, а також конкретизується технічне оснащення відповідно до потреб команд (робочі станції, пристрої для кросплатформного тестування тощо).
- План управління ресурсами: формується нормативний документ, який регламентує принципи використання ресурсів, включаючи логістику доступу, KPI-метрики ефективності, порядок оновлення компетенцій персоналу, а також механізми реагування на зміни у ресурсному середовищі.
- Систематичний моніторинг і контроль: застосовуються інструменти бізнес-аналітики для порівняння запланованого та фактичного споживання ресурсів у режимі реального часу. Підключення до цього процесу дозволяє

реалізовувати підхід Data-Driven Decision Making для своєчасного коригування плану та підтримки продуктивності проєкту на цільовому рівні.

Таким чином, ресурсне планування у межах цього проєкту не зводиться до механічного розподілу робочих годин чи техніки, а є інтегрованим управлінським процесом, що забезпечує гармонізацію цілей, ресурсів та обмежень проєкту в динамічному середовищі розробки цифрових продуктів.

2.2.3 Планування фінансів проєкту

Фінансове планування є одним із ключових етапів управління будь-яким проєктом, особливо у сфері IT, де витрати часто залежать від динамічних технологічних змін і непередбачуваних зовнішніх факторів. Саме завдяки ретельно спланованій фінансовій моделі можна уникнути критичних помилок, пов’язаних із недофінансуванням, нерівномірним розподілом ресурсів або надлишковими витратами. У проєкті створення веб-сервісу для перегляду веб-коміксів фінансове планування має на меті забезпечення стабільного фінансування всіх етапів — від аналітики та дизайну до тестування, маркетингу й супроводу.

Основна мета фінансового планування полягає у створенні збалансованого бюджету, який дозволить:

- забезпечити фінансову підтримку кожного етапу реалізації проєкту відповідно до графіку;
- мінімізувати ймовірність затримок у розробці через нестачу коштів;
- забезпечити контроль витрат з урахуванням внутрішніх і зовнішніх ризиків;
- гарантувати ефективність використання кожної гривні інвестованих ресурсів;

- створити підґрунтя для прийняття стратегічних рішень щодо масштабування, комерціалізації або доопрацювання продукту на пізніших етапах.

Бюджет проєкту включає як прямі витрати, безпосередньо пов'язані з розробкою, так і непрямі, пов’язані з організацією процесу, адміністративною підтримкою, маркетингом, ризиками. Усі ці статті мають бути чітко визначені ще на початковому етапі.

1. Заробітна плата команди проєкту.

Успішність реалізації значною мірою залежить від кваліфікації й мотивації команди. Розрахунок заробітної плати базується на середньоринкових ставках по Україні:

- Розробник веб-сервісу (full-stack): 40 000 грн/міс × 4 міс = 160 000 грн
- UI/UX дизайнер: 30 000 грн/міс × 3 міс = 90 000 грн
- Тестувальник (QA): 25 000 грн/міс × 2 міс = 50 000 грн
- Проєктний менеджер: 35 000 грн/міс × 4 міс = 140 000 грн
- Контент-менеджер (додавання коміксів, описів): 20 000 грн/міс × 2 міс = 40 000 грн. Разом на оплату праці: 480 000 грн

2. Технічна інфраструктура та хостинг.

З метою стабільної роботи та масштабованості веб-сервісу проєкт передбачає використання хмарної інфраструктури з резервуванням даних:

- Хостинг (VPS), база даних, резервне копіювання: 3 500 грн/міс × 4 міс = 14 000 грн
- Доменне ім’я та SSL-сертифікат: 2 000 грн/рік
- Обладнання для внутрішнього тестування (ноутбук, планшет): 10 000 грн. Разом технічна інфраструктура: 26 000 грн

3. Інструменти та програмне забезпечення. Для повноцінної реалізації проєкту потрібні професійні інструменти для дизайну, розробки, командної роботи:

- Підписка на Figma, Adobe XD, IDE (WebStorm/VS Code): 1 000 грн/міс × 4 міс = 4 000 грн
- Інші інструменти (Trello, Slack, GitHub Pro): 1 500 грн/міс × 4 міс = 6 000 грн
- Оренда/придбання шаблонів або плагінів: 2 000 грн. Разом ПЗ: 12 000 грн

4. Маркетингові витрати

Щоб веб-сервіс мав успіх серед користувачів, потрібно передбачити просування:

- Рекламна кампанія в соцмережах: 10 000 грн
- Відео- та візуальний контент: 5 000 грн
- SEO, аналітика, інтеграція з Google Search Console та Analytics: 6 000 грн
- Первинна PR-кампанія (огляди на форумах, інтерв'ю): 4 000 грн. Разом маркетинг: 25 000 грн

5. Резервний фонд.

На випадок технічних або організаційних змін передбачається фонд:

- 10% від загального бюджету: 60 000 грн

У таблиці наведено підсумковий бюджет проєкту.

Таблиця 2.2

Бюджет проєкту

Стаття витрат	Сума (грн)
Заробітна плата	480 000
Технічна інфраструктура	26 000
Інструменти та ПЗ	12 000
Маркетинг	25 000
Резерв	60 000
Разом	**603 000**

Для складання бюджету було застосовано комбінований підхід. Основним методом став метод аналогій: аналіз витрат на аналогічні проєкти, доступні у відкритих джерелах, або описані в кейсах розробників. Додатково було використано експертне оцінювання — проведено консультації з фахівцями в галузі UI/UX, DevOps, а також маркетологами, які мають досвід у просуванні стартапів. Оцінки перехресно перевірялися за допомогою калькуляторів вартості програмного забезпечення.

Фінансовий контроль реалізується через систему щотижневого моніторингу виконання бюджету. Всі витрати фіксуються у внутрішній ERP-системі або Excel-таблиці. Менеджер проєкту відповідає за формування щомісячного фінансового звіту, де зазначаються фактичні витрати, прогнозні зміни та причини відхилень. У разі необхідності перегляду бюджету, змінений план погоджується з усіма учасниками проєкту.

Для ефективного керування процесами розробки веб-сервісу для перегляду веб-коміксів доцільно використати математичну модель, що дозволяє враховувати часові, ресурсні та фінансові обмеження проєкту.

Проєкт описується у вигляді орієнтованого ациклічного графа $G = (V, E)$, де:

V — множина задач (v_1, v_2, ..., v_n);

E — залежності між задачами.

Позначення:

t_i — тривалість виконання задачі i;

s_i — момент початку задачі i;

$f_i = s_i + t_i$ — момент завершення задачі i;

c_i — витрати на виконання задачі i;

C_max — загальний допустимий бюджет;

T_max — гранична тривалість проєкту;

Pred(i) — множина задач, що передують i.

Цільова функція (2.1):

$$\min T = \max(f_i), \text{ для всіх } i \in V \quad (2.1)$$

Обмеження:

1. Урахування залежностей (2.2):

- $s_i \geq f_j$, для всіх $(j, i) \in E$ (2.2)

2. Бюджетне обмеження (2.3):

- $\sum c_i \leq C\_max$ (2.3)

3. Обмеження за ресурсами (2.4):

- $\sum R_{ij} \leq R_j(\text{available})$, для всіх j (2.4)

4. Часові межі (2.5):

- $T \leq T\_max$ (2.5)

Сформована модель дає змогу оптимізувати розподіл ресурсів у проєкті з урахуванням логіки задач, витрат, тривалості та пріоритетів. Такий підхід може бути реалізований у програмних засобах управління проєктами, а також у середовищах математичного моделювання.

Фінансове планування, реалізоване на ранньому етапі розробки, формує не лише основу для чіткого контролю бюджету, а й є гарантом стабільного розвитку проєкту в подальшому. Адже лише при достатньому та раціональному

фінансуванні можливо досягти технічної якості, зручності для користувачів і конкурентоспроможності продукту на ринку.

2.3 Моделювання проєктних процесів із використанням BPMN та діаграм UML

Задля побудови ефективної та передбачуваної архітектури проєкту створення веб-сервісу для перегляду коміксів необхідно залучати сучасні методи візуального моделювання. Такі методи не лише полегшують узгодження вимог між членами команди, але й слугують основою для технічного документування, автоматизації та подальшого масштабування рішення. Основними засобами у цьому контексті виступають нотація BPMN (Business Process Model and Notation) та мова моделювання UML (Unified Modeling Language), що надають змогу наочно відобразити як бізнес-процеси, так і внутрішню структуру системи.

BPMN дозволяє зобразити послідовність та логіку виконання бізнес-операцій, які підтримуються системою. У випадку платформи перегляду цифрових коміксів це включає в себе широкий спектр взаємодій, які виникають між користувачами, серверами додатку та адміністративною частиною системи. Особливістю BPMN є чітке структурування процесів через графічні елементи: події, задачі, шлюзи та потоки, що дає змогу відображати як лінійні, так і умовно-гілкові сценарії.

До ключових процесів, які моделюються за допомогою BPMN у даному проєкті, належать:

- початковий вхід користувача на платформу з можливістю входу або реєстрації;
- доступ до головної сторінки та навігація по категоріях коміксів;
- перегляд вибраного коміксу з відображенням сторінок у зручному форматі;

- взаємодія з інтерфейсом (додавання в улюблене, залишення коментарів);
- формування аналітичної інформації для адміністратора (щоденні звіти, статистика популярності);
- адміністрування контенту та модерація користувацької активності.

Кожен процес супроводжується конкретними подіями на вході й виході, що дозволяє чітко визначити його межі, роль виконавців та умови переходів. Такий підхід дає змогу уникнути неоднозначностей при формуванні вимог до майбутньої реалізації та закладає основу для автоматизованого тестування логіки.

UML, як міжнародний стандарт для візуального опису програмних систем, забезпечує гнучкий інструментарій для моделювання як структурних, так і поведінкових аспектів програмного забезпечення. У рамках розробки веб-сервісу для перегляду коміксів використовується низка типів UML-діаграм, кожна з яких виконує конкретну функцію в межах загальної інженерної документації.

- Use Case Diagram (діаграма варіантів використання): показує взаємодію між системою та зовнішніми акторами — звичайними користувачами, адміністраторами, а також можливими зовнішніми сервісами (наприклад, платіжні шлюзи). Це допомагає визначити функціональні області системи, що мають бути реалізовані на рівні інтерфейсу користувача та бекенду.
- Activity Diagram (діаграма діяльності): моделює покроковий алгоритм виконання дій, таких як процес реєстрації користувача, навігація між сторінками, генерація добірок коміксів на основі попередніх переглядів. Вона також ідеально підходить для моделювання умовних розгалужень та циклічних дій.
- Class Diagram (діаграма класів): деталізує об’єктну модель системи, де описуються сутності, їхні властивості та методи, а також зв’язки типу "один

до багатьох", "успадкування" тощо. Наприклад, об'єкти "Комікс", "Епізод", "Користувач" можуть бути з’єднані відповідними асоціаціями з описом типів зв’язку.

- Sequence Diagram (діаграма послідовностей): фокусується на часовому порядку обміну повідомленнями між об'єктами. Вона дозволяє змоделювати сценарії на кшталт "користувач відкриває випуск", де задіяні компоненти інтерфейсу, логіки контролера, бази даних та серверів зберігання контенту.

Використання UML-діаграм на етапі проєктування значно полегшує як процес розробки, так і майбутню підтримку системи. Чітке візуальне уявлення логіки взаємодій між компонентами не лише допомагає команді розробників і тестувальників уникати неоднозначностей, а й є ефективним засобом для комунікації з бізнес-замовниками, інвесторами та іншими зацікавленими сторонами. Таким чином, UML-моделювання сприяє зменшенню ризиків, підвищенню якості технічної реалізації та пришвидшенню узгодження проєктних рішень між усіма учасниками життєвого циклу програмного продукту.

Загалом, включення BPMN та UML-діаграм у процес планування і проєктування веб-платформи дає змогу досягти високого ступеня прозорості та точності як у формулюванні вимог, так і в їх реалізації. Це суттєво скорочує час на комунікацію між командами, знижує ймовірність помилок та оптимізує процес подальшої розробки, розгортання й супроводу сервісу. Формалізовані моделі також полегшують залучення нових учасників до команди, оскільки забезпечують повноцінне уявлення про архітектуру та функціональність системи без необхідності глибокого аналізу сирцевого коду.

2.4 Застосування методів і засобів управління проєктом

Реалізація IT-проєктів, зокрема створення веб-сервісів, передбачає послідовне використання методів управління, які гарантують злагоджену

координацію дій учасників команди, ефективне планування, контроль якості та адаптацію до змін. В умовах постійної зміни технологічного середовища та високих очікувань користувачів, особливого значення набуває гнучкість підходів до планування й впровадження, забезпечення прозорості всіх етапів роботи та миттєве реагування на ризики чи потреби змін.

Проєкт веб-платформи для перегляду цифрових коміксів реалізовувався із застосуванням поєднання традиційних методологій управління проєктами та сучасних підходів, зокрема гнучких фреймворків (agile-методологій), що забезпечили ефективну адаптацію до нових вимог і забезпечили швидку поставку функціонального продукту.

Одним із основних принципів, що визначали логіку реалізації, була каскадно-ітеративна модель. Вона дала змогу на етапі планування сформулювати чіткий перелік цілей, визначити ключові функціональні блоки системи та пріоритетність їх реалізації. Подальша розробка здійснювалась поетапно, із застосуванням коротких ітерацій. Як основний інструмент управління реалізацією було обрано Scrum — гнучку методологію, що передбачає поділ проєкту на серії коротких інтервалів , кожен з яких має визначену мету та завершується створенням інкременту продукту.

Кожна ітерація проєкту супроводжувалась ретельним плануванням спринту, щоденними стендапами для синхронізації учасників команди, демо-презентаціями результатів виконаних завдань та ретроспективами для виявлення вузьких місць і підвищення ефективності наступних ітерацій. Подібна структура управління дозволила швидко реагувати на фідбек користувачів, оптимізувати процеси, покращувати функціональність без значних відхилень від початкових термінів і бюджету.

Для координації командної роботи й ефективної реалізації завдань використовувався комплекс цифрових інструментів, кожен з яких відігравав специфічну роль у забезпеченні управлінської та технічної цілісності процесу:

- Trello — як основний інструмент для візуального управління завданнями за методикою Kanban: формування беклогу продукту, розподіл задач між учасниками команди, пріоритезація завдань, візуальний моніторинг прогресу, індикація блокерів;
- Google Workspace — для створення та обміну документацією, організації відеоконференцій і синхронізаційних зустрічей, спільного редагування планів, технічних вимог, облікових таблиць, брифів, та звітів;
- Figma — для розробки інтерфейсів, створення інтерактивних прототипів користувацького досвіду, обговорення макетів із замовниками, спільного коментування дизайнерських рішень, проведення UI-рев'ю;
- Git та GitHub — як система контролю версій, платформа для командної роботи над кодом, організації процесу рев'ю pull request'ів, застосування автоматизованого тестування (CI/CD), реєстрації технічних задач через issues;
- Slack — як основний засіб оперативної комунікації, обміну інформацією, інформування про зміни у завданнях, а також для інтеграції з іншими сервісами, що суттєво підвищує прозорість і своєчасність інформаційного потоку в межах команди.

Завдяки поєднанню продуманої методології та відповідного набору інструментів, команда мала змогу підтримувати високий рівень залученості, ефективно управляти ризиками, своєчасно адаптуватися до вимог замовника та забезпечувати якісне виконання кожного етапу проєкту.

Важливим елементом методології управління став календарний план, який дозволив структурувати етапи виконання робіт і забезпечити контроль за їх дотриманням.

Таблиця 2.3

Календарний план проэкту

№	Етап роботи	Зміст етапу	Тривалість	Орієнтовні дати
1	Аналіз потреб та постановка задач	Вивчення ринку, аналіз цільової аудиторії, визначення функціональних і нефункціональних вимог	1 тиждень	01.02.2025 – 07.02.2025
2	Проєктування архітектури та UI/UX	Розробка загальної архітектури системи, створення макетів інтерфейсу, побудова карти навігації	2 тижні	08.02.2025 – 21.02.2025
3	Розробка базової версії (MVP)	Програмна реалізація основного функціоналу, налаштування бази даних, базова інтеграція UI	3 тижні	22.02.2025 – 14.03.2025
4	Тестування та оптимізація	Функціональне тестування, усунення помилок, адаптація під мобільні пристрої, перевірка безпеки	2 тижні	15.03.2025 – 28.03.2025
5	Масштабування функціоналу	Додавання нових модулів: система підписок, рейтинги, коментарі; покращення продуктивності	2 тижні	29.03.2025 – 11.04.2025
6	Завершення, створення документації, реліз	Підготовка інструкцій, генерація звітності, публічне розгортання, збирання відгуків	1 тиждень	12.04.2025 – 18.04.2025

Цей план дозволив не лише відстежувати прогрес, а й прогнозувати навантаження, розподіляти ресурси, визначати критичні точки та потенційні ризики. Його динамічна структура забезпечувала можливість гнучкого коригування на основі результатів проміжних ітерацій, без втрати загальної цілісності розробки.

Таким чином, застосування інтегрованого підходу до управління — що включає гнучку методологію Scrum, спеціалізовані інструменти контролю та комунікації, а також ретельно структурований календарний план — дало змогу створити стабільний та адаптивний проєктний процес. Це в результаті сприяло ефективному досягненню технічних, часових та якісних показників, що визначають успішність цифрового продукту.

2.5 Визначення критеріїв успішності реалізації проєкту

Успішна реалізація проєкту створення веб-сервісу для перегляду веб-коміксів залежить від ретельно сформульованих критеріїв оцінки, які дозволяють не лише кількісно, але й якісно оцінити результати проєктної діяльності. У цьому контексті критерії успішності мають враховувати як технічні, організаційні, так і користувацькі аспекти. Вони покликані гарантувати, що продукт задовольняє потреби цільової аудиторії, відповідає встановленим стандартам і досягає запланованих бізнес-цілей [9, 32, 38, 36].

Критерії успішності в даному проєкті базуються на трьох основоположних принципах: дотримання встановлених обмежень, відповідність функціональним вимогам та досягнення високого рівня задоволеності кінцевих користувачів. Застосування цих принципів дозволяє здійснити цілісну оцінку проєкту та прийняти обґрунтоване рішення щодо його завершення.

Нижче наведено розгорнутий перелік критеріїв, що визначають успішність реалізації проєкту:

- **Дотримання термінів реалізації проєкту.** Виконання всіх етапів проєкту в межах узгодженого календарного графіку є визначальним показником ефективного планування та управління ресурсами. Перевищення термінів свідчить про наявність ризиків, недостатню оцінку обсягів робіт або проблеми в координації команди. Навпаки, своєчасне завершення проєкту демонструє належну дисципліну та відповідальність виконавців.

- **Укладання в бюджет.** Один із найважливіших факторів, що демонструє здатність ефективно розподіляти наявні ресурси. Успішний проєкт не потребує перевищення кошторису і не потребує додаткового фінансування. Оцінка включає аналіз витрат на проєктування, розробку, тестування, впровадження та обслуговування. Важливо забезпечити фінансову прозорість та контроль на всіх стадіях реалізації.
- **Функціональна відповідність розробленого продукту технічному завданню.** Веб-сервіс має реалізовувати повний набір запланованих функцій, серед яких: зручний перегляд веб-коміксів, можливість реєстрації та авторизації користувачів, збереження обраного контенту, фільтрація за жанрами та авторами, пошук, а також оптимізація під різні пристрої. Будь-яке відхилення від початкових вимог свідчить про недостатній рівень контролю за процесом розробки.
- **Якість програмного продукту.** Визначається за допомогою кількох показників: стабільність роботи, швидкість завантаження, відсутність критичних помилок, а також відповідність сучасним принципам UI/UX-дизайну. Для цього проводиться бета-тестування з метою виявлення технічних та функціональних недоліків. Висока якість продукту забезпечує конкурентоспроможність сервісу на ринку.
- **Рівень задоволеності цільової аудиторії.** Успішність визначається шляхом збору зворотного зв’язку від користувачів – через анкетування, інтерв’ювання, аналіз поведінкових метрик. Високий рівень залученості свідчить про ефективну реалізацію очікувань користувачів.
- **Масштабованість та потенціал подальшого розвитку.** Успішний проєкт має бути не лише завершеним, але й гнучким у контексті майбутнього розширення функціональності або адаптації до нових технологій. Наприклад, можливість додати підтримку кількох мов, систему рекомендацій, інструменти

для взаємодії авторів і читачів або інтеграцію з платіжними системами. Готовність проєкту до масштабування є свідченням стратегічного мислення та далекоглядності.

- **Забезпечення інформаційної безпеки.** Надійний захист персональних даних користувачів є обов'язковим. До базових вимог належать використання шифрування (SSL-сертифікати), система автентифікації та авторизації, регулярне резервне копіювання даних, захист від атак типу SQL-ін'єкцій, XSS та інших типових загроз. Безпека сприяє довірі з боку користувачів та стабільному функціонуванню системи.

У підсумку, проєкт вважається успішним, якщо всі вищевказані критерії дотримані в повному обсязі. Це означає, що продукт створено в межах визначених ресурсів і строків, відповідає технічним та функціональним вимогам, забезпечує високу якість користувацького досвіду та має потенціал для подальшого розвитку. Важливо також, щоб рішення, реалізовані в процесі проєктування, враховували потреби цільової аудиторії, дотримувалися принципів масштабованості, доступності та безпеки.

2.6 Управління проєктними ризиками

Управління ризиками – невід'ємна складова успішного менеджменту проєктів, особливо інноваційних стартапів. На ранніх етапах проєкту MangaView було проведено ідентифікацію потенційних ризиків, оцінено ймовірність їх виникнення та можливий вплив на проєкт, а також розроблено заходи з пом'якшення кожного ризику. Значущі ризики проєкту та стратегії реагування:

- Технічні ризики (невідповідність технологій або проблеми продуктивності): можливе зниження швидкодії сервісу при зростанні кількості користувачів, або складнощі інтеграції обраних фреймворків. Пом'якшення: проведення навантажувального тестування, оптимізація коду і запитів до БД, поступове масштабування інфраструктури (горизонтальне масштабування,

використання CDN). Резервний план – залучення експерта з оптимізації, перехід на більш продуктивний хостинг.

- Ресурсні ризики (дефіцит часу чи кадрів): обмежений розмір команди (фактично 4–5 осіб) та жорсткі строки виконання можуть призвести до перевантаження учасників або зриву окремих термінів. Пом'якшення: пріоритизація функціоналу (MoSCoW-аналіз) – першочергово реалізуються критичні можливості, а другорядні можуть бути перенесені в післярелізний період; залучення додаткових спеціалістів на критичних етапах (напр. фрілансер для тестування).
- Фінансові ризики (перевищення бюджету): можливе недооцінення вартості окремих робіт або непередбачені витрати (наприклад, на закупівлю контенту чи додаткове обладнання). Пом'якшення: створення резервного фонду (в бюджет вже закладено ~10% на непередбачені витрати), поетапне фінансування – витрати узгоджуються перед початком кожної фази, застосування принципів бережливого стартапу (Lean) – реалізація найціннішого функціоналу з мінімальними витратами.
- Ризики безпеки та даних: загроза витоку персональних даних користувачів, хакерських атак (SQL-ін'єкції, XSS), втрати даних через збої. Пом'якшення: впровадження практик Secure SDLC – використання SSL, регулярні аудит безпеки коду, автоматизоване сканування на уразливості; резервне копіювання бази даних; план реагування на інциденти (оперативне повідомлення користувачів, відновлення системи з бекапу).
- Правові та контентні ризики: можливі претензії щодо авторських прав на завантажувані комікси, чи необхідність модерації неприйнятного контенту. Пом'якшення: розробка чіткої політики використання сервісу та угоди для авторів і користувачів, фільтрація контенту (модерація), готовність оперативно видаляти контент на вимогу правовласників; юридична консультація щодо дотримання законів про інтелектуальну власність.

- Ризики прийняття користувачами: цільова аудиторія може не прийняти продукт (низька залученість, критика інтерфейсу чи функцій). Пом’якшення: залучення бета-користувачів на ранніх етапах для тестування прототипу, збір зворотного зв’язку і поступове покращення UX; проведення опитувань та інтерв’ю з користувачами щодо бажаних функцій; гнучкий беклог, що швидко реагує на запити спільноти.

Після ідентифікації та аналізу, для кожного ризику визначено власника (відповідальну особу з команди) та розроблено план реагування. Реєстр ризиків постійно актуалізувався на стендапах і моніторингах проєкту. Таким чином, проєктний ризик-менеджмент у MangaView був проактивним: команда завчасно готувалася до потенційних проблем, що дозволило уникнути криз та забезпечити стабільний хід розробки.

2.7 Побудова концептуальних моделей системи

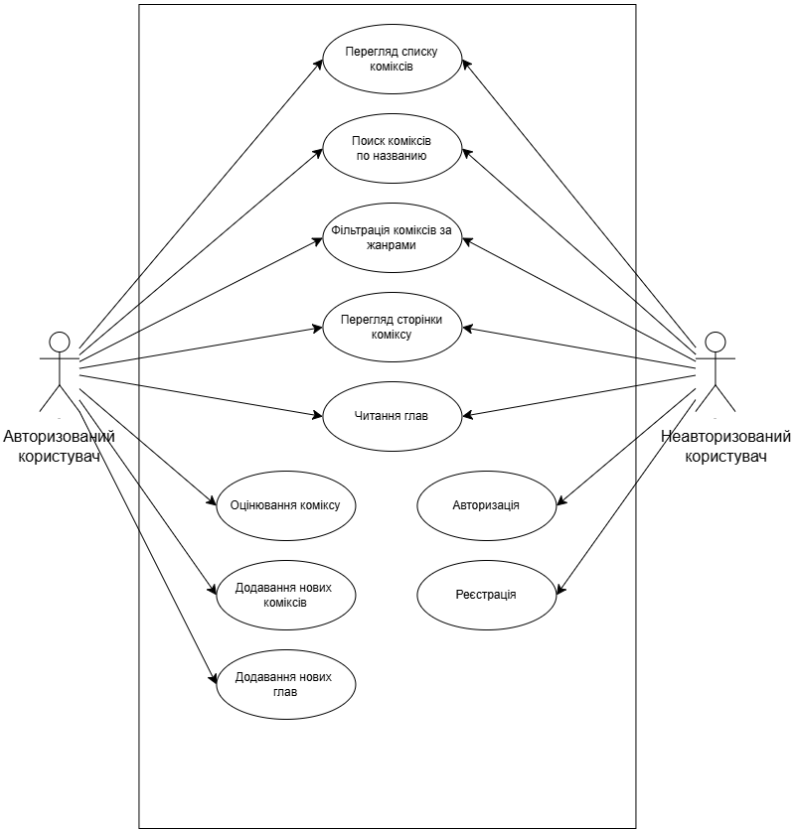

Рис. 1.2. Модель прецедентів (Use Case Model)

На етапі архітектурного планування веб-сервісу для перегляду веб-коміксів важливо перейти від абстрактних уявлень до чітко структурованих схем, що відображають функціональну логіку, дані, ролі користувачів і архітектурні зв'язки системи. Концептуальні моделі дають змогу всім учасникам проєкту — розробникам, аналітикам, дизайнерам, маркетологам, інвесторам — досягти спільного бачення та зменшити ризики непорозумінь під час наступних етапів реалізації.

Модель прецедентів (Рис. 1.1.) виконує роль системної візуалізації всіх можливих сценаріїв взаємодії між користувачами платформи й системою. Основна увага приділяється класифікації акторів (користувачів із різними рівнями доступу) та визначенню їхніх типових дій у межах цифрового середовища.

Використання діаграм прецедентів UML забезпечує уніфіковану документацію та чітке розуміння системних вимог до інтерфейсу користувача й логіки взаємодії.

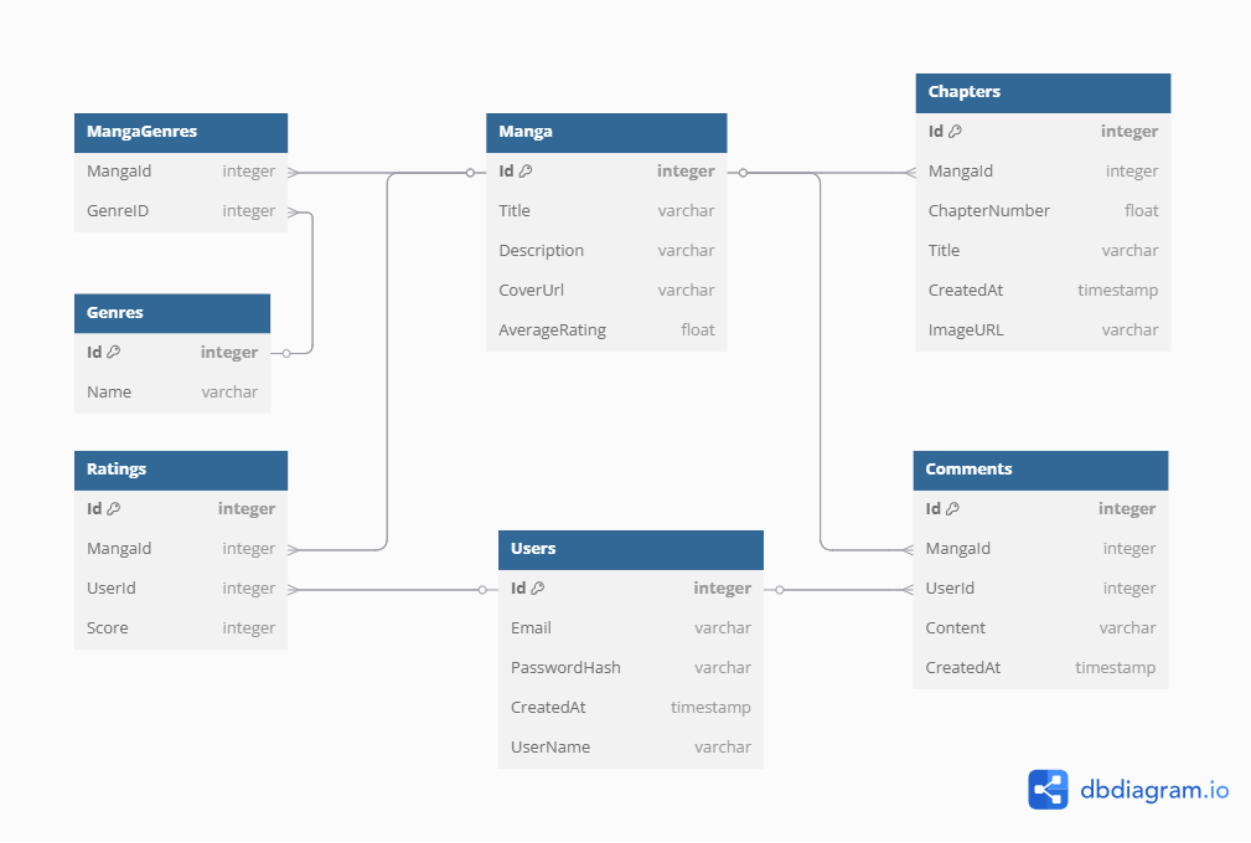

Рис. 1.3. ER-модель (Логічна модель бази даних)

Модель "сутність-зв’язок" (Рис. 1.2.) формує уявлення про логічну структуру даних, необхідних для коректного функціонування веб-сервісу. Вона описує взаємозв’язки між сутностями, що зберігаються у базі даних.

Ключові сутності:

- Users (Користувачі) — зберігає облікові дані користувачів:
  - Id — первинний ключ;
  - Email, UserName, PasswordHash, CreatedAt — атрибути ідентифікації та автентифікації.
  - Manga (Комікси) — містить дані про окремі твори:
  - Id, Title, Description, CoverUrl, AverageRating — основна інформація для виводу на інтерфейсі.

    - Chapters (Розділи) — розділи коміксів, що мають належність до певної манґи:
    - Id, MangaId, ChapterNumber, Title, CreatedAt, ImageURL — включають метадані й вміст.
- Genres (Жанри) — класифікатор типів контенту:
    - Id, Name — дозволяє реалізувати фільтрацію за жанрами.
    - MangaGenres — реалізує зв’язок багато-до-багатьох між Manga і Genres.
- Ratings (Оцінки) — зберігає індивідуальні оцінки коміксів:
    - Id, MangaId, UserId, Score — використовується для обчислення середньої оцінки.
    - Comments (Коментарі) — зв’язана сутність з Users і Manga:
    - Id, MangaId, UserId, Content, CreatedAt — текстові відгуки користувачів.

Зв’язки між сутностями встановлюють правила: один користувач може підтримувати декількох авторів, один комікс може мати багато епізодів, один автор — багато коміксів тощо. ER-діаграма унаочнює структуру даних і допомагає уникнути логічних конфліктів при розробці бази даних.

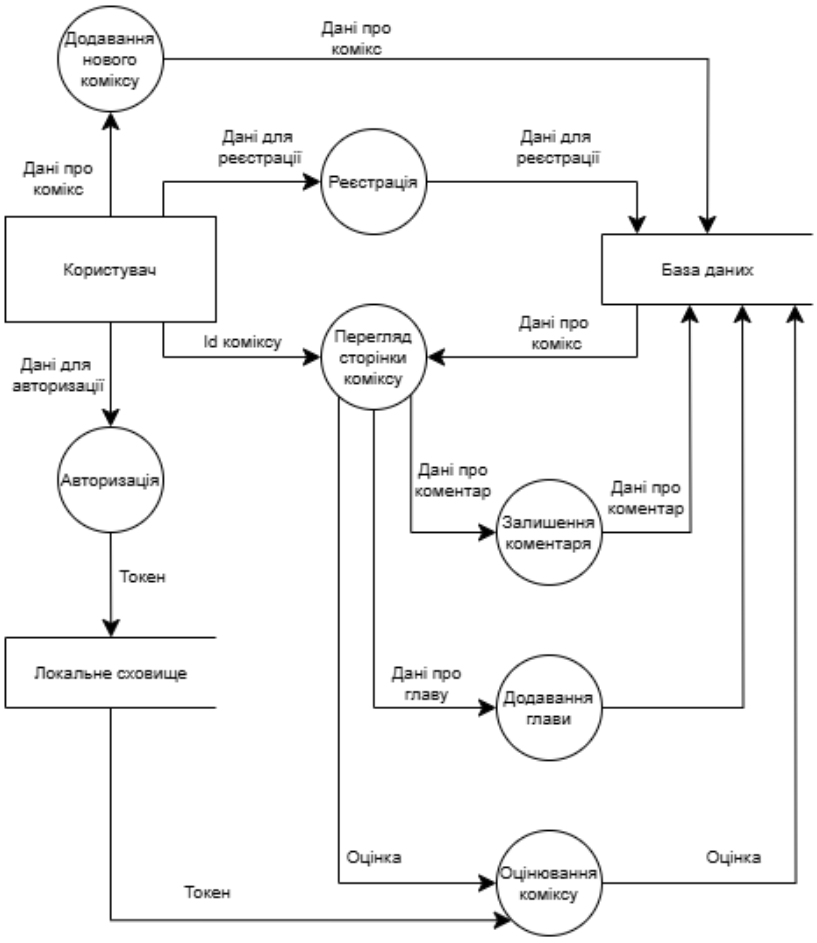

Рис. 1.4. DFD-модель (Фізична модель бази даних)

Фізична модель бази даних (Рис. 1.3.) дозволяє унаочнити обмін інформацією в межах системи, виявити ключові точки обробки даних, а також формалізувати зовнішні та внутрішні джерела й одержувачів інформації.

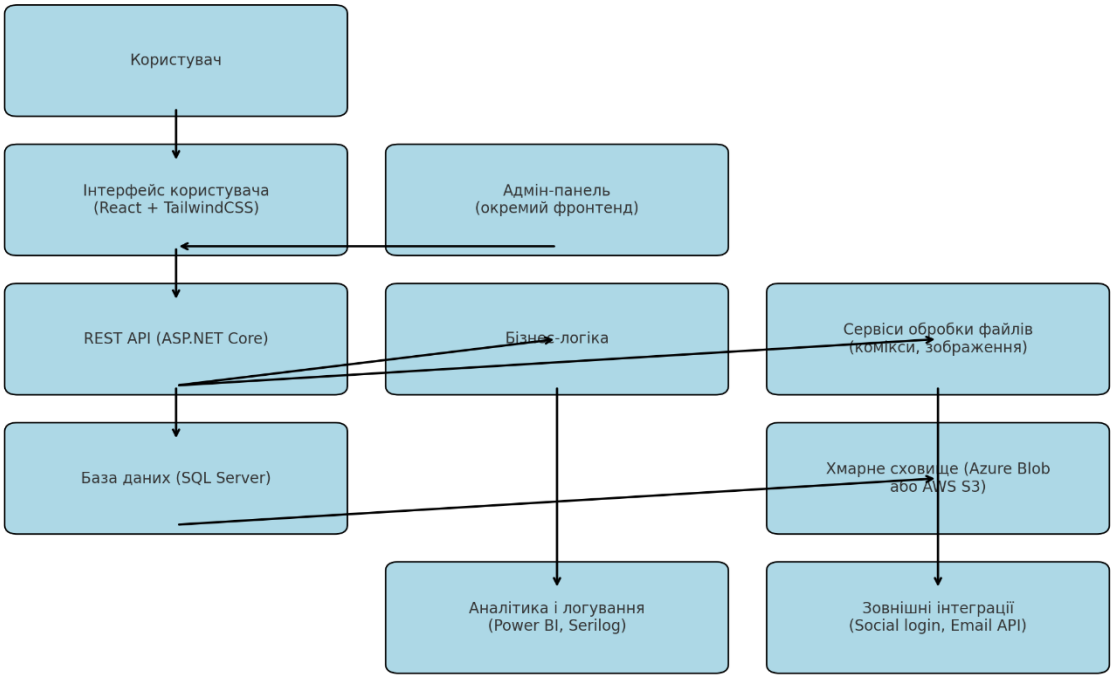

Рис. 1.5. Концептуальна архітектурна модель

Архітектура веб-сервісу (Рис. 1.4.) для перегляду веб-коміксів побудована за принципами розподіленої багаторівневої системи. Вона поділяється на фронтенд (інтерфейс), бекенд (логіка й API), систему зберігання даних, хмарні сервіси та інструменти аналітики.

- Користувацький інтерфейс реалізовано з використанням React та TailwindCSS для створення SPA (Single Page Application) з інтерактивним UI.
- REST API на базі ASP.NET Core забезпечує взаємодію між фронтендом і серверною частиною.
- Бізнес-логіка відповідає за обробку авторизації, управління контентом (коміксами, жанрами, підписками), систему рейтингу тощо.
- База даних SQL Server зберігає структуровані дані: профілі користувачів, історію переглядів, коментарі та рейтинги.

- Хмарне сховище (Azure Blob або AWS S3) використовується для зберігання медіафайлів — зображень сторінок коміксів.
- Адмін-панель розміщена окремо і дозволяє модерувати контент, керувати жанрами та користувачами.
- Аналітика та моніторинг виконуються через Power BI (візуалізація метрик) та Serilog (логування подій).
- Зовнішні інтеграції, зокрема OAuth2-авторизація (Google, Facebook), поштові сервіси для розсилок та нотифікацій.
- Така модель забезпечує гнучкість, масштабованість і високу продуктивність — критичні характеристики для сучасної платформи з великою базою користувачів і контенту, подібно до Webtoon або Tapas.

Головні підсистеми:

Інтерфейс користувача: включає веб-інтерфейс (адаптивний дизайн), мобільну версію, елементи доступності (напр., масштабування шрифтів, нічний режим);

- Серверна логіка: обробка запитів, авторизація, логіка контенту, моніторинг активності;
- Сховище даних: реляційна база даних, індексація контенту, кешування для швидкого доступу;
- Інтеграційні сервіси: оплата, email-розсилки, push-сповіщення, API для зовнішніх розробників.

Архітектурний шаблон (наприклад, Model-View-Controller або Model-View-ViewModel) сприяє розділенню відповідальності та забезпечує простоту підтримки й тестування. Крім того, враховується потенційна підтримка PWA-технологій для офлайн-доступу.

Концептуальне моделювання відіграє ключову роль у процесі планування та реалізації ІТ-проєкту, особливо коли йдеться про створення інноваційного

цифрового продукту, такого як веб-сервіс MangaView. У сучасному середовищі швидкої розробки, де час виведення продукту на ринок та якість початкової ідеї є критичними чинниками успіху, концептуальне моделювання виступає не просто інструментом візуалізації, а базовим аналітичним фундаментом, що забезпечує логічну цілісність і технологічну реалізованість запропонованого рішення.

З практичної точки зору, концептуальні моделі — це формалізовані представлення структури, функціональності, потоків даних та взаємодії між елементами системи. Вони виступають посередником між ідеєю, що зароджується в уявленні засновника або бізнес-аналітика, та її реалізацією у вигляді програмного коду, баз даних і взаємодії компонентів. У процесі розробки такі моделі дозволяють не тільки структурувати бачення продукту, а й виявити логічні суперечності, технічні обмеження чи «вузькі місця» ще до початку програмування.

У рамках проєкту MangaView концептуальне моделювання виконувало роль «архітектурної опори», на якій будувалась уся логіка реалізації платформи: від взаємодії користувачів із сервісом (Use Case діаграми), структури бази даних (ER-моделі) до архітектурної побудови інформаційної системи як цілісного комплексу компонентів (DFD, UML, BPMN).

Таким чином, концептуальні моделі в роботі не обмежились лише роллю технічних інструментів. Вони стали невіддільною частиною стратегії управління, забезпечивши мінімізацію ризиків, ясність у постановці задач і стабільний розвиток проєкту. В умовах високої невизначеності, що притаманна стартапам, концептуальне моделювання є не тільки засобом підготовки до розробки, але й основою для успішної презентації, гнучкого масштабування та прийняття критичних рішень у подальшій еволюції продукту.

РОЗДІЛ 3. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ РІШЕННЯ

3.1 Аналіз платформ серверної частини

Створення високонавантаженого веб-сервісу для перегляду коміксів вимагає ретельного підбору технологічного стеку. Провідні платформи веб-коміксів, такі як Webtoon, Tapas, KakaoPage та Naver Webtoon, демонструють приклади успішних рішень: їхні системи здатні обслуговувати десятки мільйонів користувачів і сотні мільярдів переглядів на рік. Зокрема, архітектури цих сервісів базуються на мікросервісному підході, що забезпечує високу масштабованість і адаптивність під навантаженнями. Беручи до уваги цей досвід, у даному розділі проаналізовано альтернативні технології для серверної частини, клієнтської частини та бази даних проєкту, виконується обґрунтування вибору оптимального стеку, а також оцінюються необхідні ресурси для реалізації проєкту з використанням економіко-математичних методів. У процесі дослідження представлені порівняльні таблиці, графічні моделі та приклади практичного застосування технологій.

Серверна частина веб-сервісу відповідає за бізнес-логіку, управління даними та масштабування системи. В контексті нашого проєкту розглянуто чотири популярні платформи серверної розробки: Node.js, Django, Spring Boot та ASP.NET Core.

Таблиця 3.1

Порвінялні критерії вибору технологій

Критерій	Node.js (Express)	Django (Python)	Spring Boot (Java)	ASP.NET Core (C#)
Мова програмування	JavaScript/TypeScript (інтерпретована, JIT-компіляція V8)	Python (інтерпретована)	Java (байткод JVM)	C# (компільована в MSIL, JIT/AOT)

Продовження табл. 3.1

Продуктивність	Висока продуктивність на I/O операціях завдяки неблокуючій моделі; середня при CPU-навантаженнях	Помірна швидкодія, обмежена продуктивністю інтерпретатора Python	Висока масштабована продуктивність завдяки JVM; можливі паузи GC	Висока швидкодія, близька до рідного коду; ефективне керування пам'яттю завдяки .NET runtime
Масштабованість	Добре підходить для мікросервісів та реального часу; використовується Netflix, PayPal тощо	Забезпечує масштабування контент-орієнтованих застосунків; використовується Instagram, Spotify	Призначена для великих enterprise-систем, високий рівень масштабованості	Ефективна для масштабних корпоративних застосунків; використовується компаніями на кшталт StackOverflow, UPS
Швидкість розробки	Дуже швидкий старт для простих застосунків; просте налаштування проекту	Висока швидкість розробки завдяки багатому набору вбудованих компонентів ("батарейки в комплекті")	Поріг входу вищий: необхідно налаштувати екосистему Spring; більше конфігурацій	Помірна: розробка спрощується за рахунок інструментів Visual Studio, проте .NET вимагає структурування проекту
Екосистема та бібліотеки	Найбільший репозиторій пакетів; тисячі бібліотек. Можливі	Широкий екосистемний супровід: плагіни Django, велика спільнота	Величезна Java-екосистема модулів, багато напрацювань для будь-яких потреб	Розвинена екосистема .NET; багато власних бібліотек Microsoft та спільноти

Продовження табл. 3.1

Інтеграція з БД	Потребує сторонніх ORM/плагінів (Sequelize, TypeORM); немає штатної ORM	Вбудована ORM (Django ORM), підтримка кількох СУБД з коробки	Spring Data забезпечує спрощену роботу як з SQL, так і NoSQL БД	ORM Entity Framework забезпечує тісну інтеграцію з SQL Server та ін. реляційними БД
Безпека	Залежить від зовнішніх middleware та пакунків (Helmet, etc.); потребує налаштування	Має вбудовані засоби автентифікації та захисту (Django auth, ORM валідації)	Spring Security – потужний фреймворк з гнучким налаштуванням безпеки	Вбудовані засоби безпеки на рівні платформи; розроблено з урахуванням вимог enterprise-сегменту
Крос-платформеність	Так (Node.js працює на Windows, Linux, macOS)	Так (Python є крос-платформеною)	Так (Java/JVM працює всюди)	Так (ASP.NET Core працює на Windows, Linux); тісна інтеграція з Windows-середовищем
Крива навчання	Низький поріг входу для JS-розробників, синтаксис простий	Помірна: Django простіший за деякі фреймворки, але має свій "спосіб робити речі"	Досить високий поріг: потрібне розуміння «опінійованої» конфігурації Spring Boot, самого Java та інструментів	Середній: розробникам з досвідом C#/.NET опанувати легко; новачкам знадобиться час на засвоєння концепцій .NET

Закінчення табл. 3.1

Сфера доцільного використання	Невеликі, легковагові сервіси, реалтайм-додатки, мікросервіси з коротким життєвим циклом. Вимагає мінімум ресурсів, легко контейнерується.	Контент-орієнтовані платформи, нові стартапи, де важлива швидкість розробки більше, ніж пікова продуктивність.	Великі міжплатформні корпоративні системи, довгострокові проєкти з високими вимогами до надійності та безпеки.	Проєкти на платформі Windows або з екосистемою Microsoft; enterprise-застосунки, що потребують інтеграції з Microsoft-сервісами. Менш придатний для дуже дрібних скриптів і сервісів.

Як видно з таблиці 3.1, Node.js вирізняється асинхронною моделлю і швидкістю розробки, проте для великих монолітних систем його застосування обмежене. Django (Python) надає розробнику багато зручностей “з коробки” (адмін-інтерфейс, ORM, аутентифікація), що дозволяє швидко створювати прототипи веб-застосунків. Однак продуктивність Django обмежується швидкодією інтерпретованого Python, а сам фреймворк доволі монолітний – навіть простий REST-сервіс тягне залежності всього Django. Spring Boot орієнтований на великі корпоративні проєкти: він забезпечує надзвичайну масштабованість, потужну систему безпеки та величезну екосистему модулів. Недоліками Spring Boot є складність освоєння і надлишковість для невеликих проєктів – мінімальний застосунок містить значний «вантаж» інфраструктурного коду та конфігурацій.

Натомість ASP.NET Core поєднує переваги високопродуктивної скомпільованої мови C# з широкими можливостями .NET-екосистеми. Платформа .NET розроблялася як універсальна основа для різних типів застосунків (веб, десктоп, мобільні, хмарні), тому має зрілий набір бібліотек і засобів для побудови складних систем. Важливою перевагою є тісна інтеграція з іншими продуктами Microsoft: сервером баз даних SQL Server, хмарною інфраструктурою Azure, засобами бізнес-аналітики тощо. Це спрощує розробку у випадку вибору єдиного технологічного стеку від Microsoft. За даними порівняльного огляду багато великих організацій використовують .NET Core/C# для своїх систем – зокрема, технологію впровадили NBC, StackOverflow, UPS та інші. Таким чином, ризики щодо підтримки і розвитку .NET-рішень мінімальні, адже існує велика спільнота розробників і експертів.

Недоліки ASP.NET Core в минулому були пов’язані з орієнтованістю на Windows-платформу та закритою екосистемою. Сьогодні .NET Core є повністю відкритим з 2016 року та мультиплатформним, але все ж оптимальні умови його використання досягаються в середовищі Windows або Azure. Для надмалих веб-сервісів чи мікросервісів, що запускаються як тимчасові контейнери, Node.js може бути більш економним. Втім, .NET Core успішно застосовується і для мікросервісної архітектури – існують практики контейнеризації .NET-застосунків і навіть запуску в serverless-середовищах.

Зважаючи на специфіку проєкту (платформа орієнтована на масову аудиторію, потенційно мільйони користувачів, підтримка складної бізнес-логіки – коментарі, рейтинги, монетизація тощо), обрано технологію ASP.NET Core для реалізації серверної частини. Ключовими аргументами є:

- Висока продуктивність – .NET Core належить до найшвидших веб-фреймворків за результатами незалежних тестів, забезпечуючи низький час відповіді навіть під значним навантаженням.

- Масштабованість та enterprise-функціональність – платформа розроблена для складних застосунків, підтримує масштабування на рівні веб-ферм, має вбудовані засоби кешування, безпеки, логування. Це полегшить подальше зростання платформи.
- Інтеграція з SQL Server – вибір СУБД схиляється до Microsoft SQL Server, що оптимально працює в парі з .NET (через ORM Entity Framework та драйвери від Microsoft). За однією з оцінок, .NET Core "дуже добре інтегрований з екосистемою Microsoft, зокрема з СУБД SQL Server, та відрізняється розвиненими засобами безпеки".
- Безпека та надійність – вихідний код .NET Core відкритий і перевірений часом, Microsoft регулярно випускає оновлення безпеки. Є підтримка стандартів OWASP, вбудовані засоби захисту від XSS, CSRF та ін. у фреймворку ASP.NET. Для платформи, що зберігатиме конфіденційні дані користувачів, це суттєво.
- Команда розробки – наявність у команди експертизи з C#/.NET зменшує ризики та криву навчання. Як зазначається в огляді, вибір фреймворку слід робити з урахуванням компетенцій команди: якщо команда досвідчена в Python – варто обрати Django, якщо у Java – Spring Boot. У нашому випадку орієнтація на .NET є виправданою з огляду на кадровий потенціал.

Таким чином, для серверної частини веб-сервісу обрано ASP.NET Core з мовою програмування C#, що забезпечить баланс між продуктивністю, масштабованістю та швидкістю розробки. Даний вибір узгоджується з практиками провідних платформ (наприклад, Webtoon (Naver) має архітектуру рівня enterprise і співставний технологічний стек), а також створює передумови для ефективної інтеграції усіх компонентів проєкту в єдине ціле.

3.2 Порівняння фреймворків клієнтської частини та вибір React

Клієнтська частина (frontend) визначає інтерфейс взаємодії користувача із системою. Для веб-сервісу коміксів фронтенд повинен забезпечувати зручний перегляд графічного контенту, швидку навігацію між розділами, інтерактивність та адаптивність під різні пристрої. Сучасні веб-платформи цього класу реалізують клієнтську частину як односторінкові додатки (SPA) або прогресивні веб-додатки (PWA) з офлайн-можливостями. Наприклад, Naver Webtoon і Tapas пропонують мобільні додатки та адаптивні веб-версії, що дозволяють читати комікси офлайн через завантаження випусків у локальне сховище. Для досягнення подібного рівня UX, наш проєкт потребує сучасного фреймворку, який підтримує побудову SPA/PWA, має високу продуктивність рендерингу графіки та широкий набір готових компонентів. Було розглянуто чотири варіанти фронтенд-технологій: Angular, React, Vue.js та Blazor. Перші три – це провідні JavaScript/TypeScript-фреймворки, тоді як Blazor дозволяє писати клієнтський код мовою C# (працює на WebAssembly або з серверною підтримкою SignalR). Коротко про особливості кожного рішення:

- Angular – повноцінний фреймворк від Google, що використовує TypeScript. Містить все необхідне з коробки (MVC-підхід, роутинг, форми, HTTP-клієнт, тощо). Відмінністю Angular є використання шаблонів HTML з директивами і двостороннім зв’язком. Angular добре підходить для великих команд та проектів, де важлива структурованість коду і стандартизація. Недоліки: відносно висока складність, «важка» початкова завантаженість SPA, менш гнучкий, ніж легші рішення.
- React – бібліотека від Meta для побудови UI. Використовує декларативний підхід і віртуальний DOM для ефективного оновлення інтерфейсу. React забезпечує лише View-рівень, але має величезну екосистему сторонніх бібліотек для стану, маршрутизації, тощо. Відзначається висока

гнучкість і продуктивність React, а також найбільша спільнота розробників серед фронтенд-рішень. Може використовуватися з JavaScript або TypeScript [8, 22].

- Vue.js – прогресивний фреймворк, створений Еваном Ю. Поєднує підхід React і Angular в легкій вазі. Vue має простий синтаксис і низький поріг входження, при цьому дозволяє масштабуватися до великих SPA. Популярний у спільнотах Азії та серед невеликих команд завдяки зрозумілій документації. Недолік – дещо менша частка ринку і попит на Vue-розробників у світі порівняно з React/Angular, а також залежність розвитку від спільноти (не стоїть корпорація за проектом).
- Blazor – фреймворк від Microsoft, що дозволяє розробляти інтерфейс на C# мовою. Працює у двох режимах: Blazor Server та Blazor WebAssembly (виконується повністю в браузері, завантажуючи .NET runtime у вигляді WASM). Переваги: спільна мова фронтенду і бекенду, повторне використання бібліотек .NET, висока безпека типів, природна інтеграція з ASP.NET. Недоліки: нова технологія – менша спільнота (лише ~5% розробників використовували Blazor станом на 2023), більший розмір завантаження (WASM-двигун ~2-5 МБ), дещо гірша продуктивність оновлення UI порівняно з React (відсутній віртуальний DOM), а в режимі Blazor Server – навантаження на сервер при кожній взаємодії.

Таблиця 3.2

Порівняння основних фронтенд-фреймворків

Критерій	Angular (Google)	React (Meta)	Vue.js	Blazor (Microsoft)
Мова та парадигма	TypeScript; повний MVC-фреймворк з шаблонами HTML	JavaScript/JSX або TSX; бібліотека для View-рівня (архітектура Flux)	JavaScript/TypeScript; прогресивний фреймворк (ядро + опційні модулі)	C# / Razor; фреймворк .NET для WebAssembly або серверного рендерингу

Продовження табл. 3.2

Продуктивність	Висока, але початковий рендер може бути повільнішим через обсяг фреймворку. Підтримує AoT-компіляцію для прискорення старту.	Дуже висока за рахунок Virtual DOM і оптимізації дифів; швидке оновлення інтерфейсу навіть при великих даних.	Відмінна швидкодія при невеликому розмірі; Virtual DOM як у React, менший оверхед, швидке завантаження.	Висока (near-native) продуктивність обчислень завдяки WebAssembly; проте DOM-операції можуть поступатися JS-фреймворкам без Virtual DOM.
Розмір і залежності	~500 KB+ (gzip) core + багато вбудованого функціоналу; цілісний фреймворк.	~100 KB core (React) + додаткові залежності за потребою (інкрементально).	~80 KB core; за потреби підключаються Vuex, Vue Router тощо.	~2-5 MB (WASM runtime) для Blazor WASM; Blazor Server залежить від серверу. Вимагає .NET runtime на клієнті.
Крива навчання	Крута: складна структура, концепти RxJS, DI, Zone.js; велика документація.	Помірна: прості концепції компонентів і стану, але багато способів реалізації (гнучкість) може бентежити новачків.	Низька: API інтуїтивний, документація детальна; легко почати з малого і нарощувати.	Середня для .NET-розробників, низька якщо знають Razor (схоже на ASP.NET MVC); для веб-розробників без знання C# – висока.

Закінчення табл. 3.2

Спільнота і підтримка	Підтримуєтьс я Google; велика спільнота enterprise-розробників, регулярні оновлення раз на ~6 міс.	Найбільша спільнота фронтенд-розробників; тисячі плагінів. React – лідер за кількістю зірок на GitHub і вакансій.	Активна спільнота, особливо в Азії; Vue продовжує зростати, хоча не має підтримки техногіганта, але широко прийнятий розробникам и.	Підтримується Microsoft; спільнота ще формується (Blazor ~5% використання в опитуваннях). Документація від MS, приклади, але набір сторонніх компонентів обмежений.
Підтримка PWA	Можлива через Angular Service Worker; Angular Material забезпечує адаптивність; є CLI генерація PWA.	Легко інтегрується з бібліотеками для PWA (Workbox тощо); Next.js або Gatsby дають SSR/PWA можливості.	Підтримує PWA через офіційний плагін; має Vue CLI PWA плагін для генерації.	Підтримка офлайн через кеш браузера; можлива інтеграція з Service Workers вручну. Blazor WASM працює офлайн після завантаження застосунку.
Використанн я у відомих проєктах	Google продукти (Gmail – частково), Microsoft (деякі внутр. системи), підприємства (банк ING, портали Bosch) широко застосовують Angular.	Facebook, Instagram, Netflix, Uber, Airbnb – React використовуєть ся у веб-інтерфейсах топ-компаній. За оцінками, React застосовується на ~46% всіх сайтів у світі.	Xiaomi, Alibaba, Adobe Portfolio – приклади використанн я Vue; також GitLab інтерфейс побудовано на Vue.	Платформа Dota 2 під час турнірів (dashboard) використовува ла Blazor Server; ряд адмін-панелей в середовищі Windows. Поки що великі публічні веб-застосунки на Blazor не анонсовані.

Аналіз показує, що React вирізняється поєднанням продуктивності, гнучкості та популярності. За результатами опитувань розробників, React утримує позицію найзатребуванішого фронтенд-фреймворку: кількість вакансій на React майже вдвічі перевищує Angular і у десятки разів – Vue.

React має найбільшу екосистему готових компонентів і бібліотек: для будь-якого завдання знайдеться кілька відпрацьованих рішень. Це скоротить час розробки клієнтської частини, адже багато компонентів (наприклад, слайдери, галереї, infinite-scroll списки для стрічки коміксів) можна буде інтегрувати, лише злегка модифікувавши під власний дизайн.

Angular також є потужним рішенням, але його надлишкова складність для нашого випадку вважається недоліком. Розробка на Angular потребує більше часу на шаблонізацію і налаштування модулів. Крім того, повний фреймворк накладає жорсткі рамки – для інноваційного стартапу може бути важливо мати більше гнучкості у виборі підходів. Наприклад, React дозволяє вільніше організувати структуру проекту, поступово впроваджувати оптимізації. Angular же диктує структуру від початку, що добре для великого підприємства, але може сповільнити маленьку команду.

Vue.js привабливий простотою і швидкістю. Фактично, Vue забезпечує багато переваг React при ще нижчому порозі входження. Однак, з точки зору довгострокової підтримки, обирати Vue трохи ризикованіше: пошук розробників Vue може бути складнішим, менша кількість навчальних матеріалів поза офіційною документацією, а також інтеграція з деякими сторонніми інструментами може вимагати адаптації (більшість бібліотек першочергово орієнтовані на React або Angular). Для нашого проєкту, який претендує на масштабування і залучення широкого кола учасників, важливо мати технологію, що стала промисловим стандартом. За цим критерієм React суттєво випереджає Vue: React використовують у продуктиві такі гіганти, як Facebook, Netflix, Uber, WhatsApp, Airbnb тощо, що підтверджує його надійність та ефективність.

Blazor був розглянутий як цікавий варіант, оскільки ми обрали .NET на бекенді. Можливість писати фронтенд і бекенд однією мовою виглядає привабливою: наприклад, моделі даних можна було б передавати без конвертації, валідатори можна було б перевикористати. Проте детальніше вивчення показало низку причин відмовитися від Blazor у цьому проєкті:

- Перш за все, популярність і зрілість Blazor ще невеликі. Технологія з'явилась у 2018–2019 рр., і хоча швидко розвивається, вона не пройшла випробування багатьма реальними навантаженими проєктами. Лише ~4,9% розробників зазначили досвід з Blazor у 2023 році, тоді як з React – десятки відсотків. Це означає брак перевірених практик і підтримки на форумах у разі проблем.
- Вага та продуктивність: навіть за оптимізації, Blazor WebAssembly змушує завантажувати мегабайти зайвих даних. Для користувачів на повільному інтернеті це може бути критично – веб-сайт має вантажитися швидко, інакше частина аудиторії піде. React-додаток можна зробити дуже легким на старті, завантажуючи решту динамічно. Blazor в цьому плані поки що поступається.
- SEO та сумісність: для успішного залучення аудиторії веб-сервіс повинен бути індексованим пошуковими системами. SPA на React можна рендерити серверно або попередньо генерувати, що покращує SEO. Blazor Server генерує HTML на сервері, але не масштабований для великої публічної платформи (відкрита постійна SignalR-сесія на кожного користувача потребує багато серверних ресурсів). Blazor WASM – чисто клієнтський рендер, що без додаткових заходів гірше для SEO. Таким чином, з Blazor треба було б впроваджувати окремі механізми для індексації (наприклад, Prerendering), що ускладнює архітектуру.
- Екосистема компонентів: багато готових UI-рішень доступні для React/Vue. Для Blazor їх значно менше, часто доведеться використовувати JavaScript-інтероп, що зводить нанівець перевагу «без JavaScript».

Зважаючи на ці фактори, для розробки клієнтської частини було обрано React. Це рішення підкріплюється трендами індустрії: за даними State of JavaScript 2024, React залишається лідером за показниками задоволеності та бажання використовувати знову, випереджаючи Angular і Vue. Популярність React продовжує зростати, про що свідчить хоча б статистика GitHub – у React ~232 тис. зірок проти ~97 тис. у Angular та ~49 тис. у Vue.

Практична реалізація фронтенду на React для нашого проєкту означає:

- Використання сучасного стеку React + TypeScript для надійності та зручності рефакторингу. TypeScript, як і Angular, забезпечить статичну типізацію, але без нав'язування структури всього застосунку.
- Застосування архітектурного патерну SPA: основний застосунок буде односторінковим, що взаємодіє з серверним API. Це зменшить навантаження на сервер (рендеринг відбувається на клієнті) і покращить UX (швидкі переходи без повного перезавантаження сторінки).
- PWA-функціональність: React-ecosystem має готові рішення для додавання offline-режиму і push-нотифікацій. Зокрема, можна використати Workbox для генерації service worker, що кешує випуски коміксів для перегляду офлайн – аналогічно до того, як це робить мобільний додаток Webtoon. Таким чином веб-версія зможе частково працювати навіть без інтернету, що підвищить лояльність користувачів.
- Бібліотека Next.js може бути розглянута для серверного рендерингу критичних сторінок (SSR) та покращення SEO. Втім, початково планується реалізація як чистого SPA з динамічним завантаженням контенту через REST API. Надалі, коли контенту буде багато, можна буде реалізувати гібридну модель: наприклад, сторінки окремих серій коміксів віддавати як статично згенеровані (SSG), що добре індексується.

- UI-бібліотеки: для швидкої побудови інтерфейсу корисно залучити готові компоненти. Можливі кандидати – Material-UI чи Bootstrap для React, щоб не витрачати надто багато часу на розробку базових елементів. Це узгоджується з академічними вимогами до прототипу – головне показати роботу бізнес-логіки, а не промальовування кожної кнопки з нуля.

Таким чином, вибір React як фронтенд-фреймворку обґрунтований його продуктивністю, гнучкістю, а головне – широким використанням в індустрії, що підтверджує надійність цього рішення. React дозволить створити інтерактивний інтерфейс платформи веб-коміксів, який не поступатиметься за зручністю таким гігантам, як Webtoon або Tapas, при цьому буде відносно простим у підтримці та розвитку завдяки великій спільноті і кількості готових рішень [20, 8].

3.3 Порівняння систем керування базами даних

Для будь-якого веб-сервісу дані є ключовим ресурсом. У випадку платформи веб-коміксів система керування базами даних (СКБД) повинна зберігати інформацію про користувачів, комікси, глави, коментарі, рейтинги, транзакції (якщо передбачена монетизація) тощо. Вимоги до сховища даних включають: надійність та цілісність даних (неприпустимо втратити покупки користувача або прогрес читання), масштабованість (можливість обробляти запити від мільйонів користувачів, швидко виконувати пошук по базі), ** гнучкість моделі** (щоб можна було додавати нові поля, сутності – наприклад, підтримати нові типи контенту). Розглянемо основні кандидати на роль СКБД для проєкту:

- MySQL – найбільш популярна у світі реляційна СКБД з відкритим кодом. Є частиною класичного стеку LAMP. MySQL відома простотою налаштування, швидким читанням даних та широкою підтримкою хостинг-провайдерів. Однак, MySQL історично поступалася PostgreSQL в плані

відповідності стандартам SQL і розширених можливостей. MySQL добре підходить для відносно простих схем даних і високого навантаження на читання.

- PostgreSQL – об’єктно-реляційна СКБД з відкритим кодом, відома своєю надійністю і потужними можливостями. PostgreSQL підтримує широкий набір типів даних (JSON, ARRAY, XML та ін.), складні запити, функції, збережені процедури кількома мовами. Вважається «просунутою» СКБД, ближчою за можливостями до комерційних систем. PostgreSQL особливо добре підходить для складних аналітичних запитів і проектів, де потрібна гнучкість схеми. Останніми роками PostgreSQL значно зросла в популярності: у 2024 р. вона вийшла на 2-е місце в рейтингу DB-Engines, випередивши за популярністю навіть деякі комерційні СКБД.
- MongoDB – документно-орієнтована NoSQL база даних. Зберігає дані у форматі BSON. Головна перевага – гнучка, горизонтальне масштабування та висока продуктивність на запис великих обсягів даних. MongoDB часто використовується в проектах, де структура даних часто змінюється або не піддається легко нормалізації. Для нашого проєкту MongoDB міг би бути корисним для зберігання, наприклад, історії перегляду або кешування згенерованих рекомендацій у вигляді документів. Однак як основна база MongoDB має недоліки: відсутність традиційних транзакцій до версії 4.0, складність реалізації складних JOINS, потенційні проблеми з цілісністю через гнучку схему.
- Microsoft SQL Server – реляційна СКБД від Microsoft. Пропонує повний спектр enterprise-функціональності: високопродуктивний рушій транзакцій, засоби аналітики, побудову звітів, інструменти для ETL. SQL Server – комерційна система, але існують безкоштовні редакції для невеликих обсягів та для розробки. Традиційно SQL Server щільно інтегрований з екосистемою Windows. З виходом .NET Core з’явилася крос-платформна версія SQL Server для

Linux. Ця СКБД оптимізована для роботи з .NET-додатками, використовуючи мову T-SQL (Transact-SQL) – розширення SQL від Microsoft.

Для порівняння СКБД оберемо такі критерії: тип моделі даних, підтримка транзакцій і цілісності, горизонтальне масштабування, продуктивність на читання/запис, доступність (ліцензія, вартість), інтеграція з нашим стеком.

Таблиця 3.3

Порівняння СКБД

Критерій	MySQL	PostgreSQL	MongoDB	MS SQL Server
Тип СКБД	Реляційна СКБД (SQL). Модель клієнт-сервер.	Реляційна ОВД (SQL) з елементами об'єктної (JSON, XML типи).	Документо-орієнтована NoSQL (JSON-документи).	Реляційна СКБД (SQL). Клиент-сервер, масштабується кластерно (AlwaysOn).
Ліцензія і вартість	Open-source (GPL); безкоштовна. Комерційна підтримка Oracle (Enterprise версії).	Open-source (MIT/BSD); безкоштовна. Підтримка спільноти, фонду PostgreSQL.	Ліцензія SSPL (Server Side Public License); безкоштовно для спільноти, комерційні послуги MongoDB Inc.	Пропрієтарна (Microsoft EULA); платна (є безкоштовна Express до 10 ГБ БД). Developer Edition безкоштовна для розробки.
Транзакції, ACID	Повна підтримка ACID-транзакцій (InnoDB рушій за замовчуванням).	Повна підтримка ACID. MVCC-конкурентність транзакцій (мінімізує блокування).	Підтримка ACID-транзакцій для одного або декількох документів (з версії 4.0); раніше – лише атомарність на рівні одного документа.	Повна підтримка ACID. Гнучкі ізоляції транзакцій, у т.ч. SNAPSHOT-ізоляція. Засоби збереження цілісності (Foreign Keys, Constraints, Triggers).

Продовження табл. 3.3

Масштабування	Вертикальне масштабування (масштабування читаючих реплік через MySQL Cluster / реплікацію). Горизонтальне – складніше (шардинг потребує зовнішніх засобів).	Вертикальне масштабування; реплікація master-slave або multi-master (BI-Directional). Шардинг підтримується через FDW або Citus (додатково).	Первинно спроектована для горизонтального масштабування: вбудований **шардинг** за ключами, легке додавання вузлів кластера. Масштабується майже лінійно по запису.	Основний спосіб – вертикальне (потужний сервер). Є технологія **AlwaysOn** для масштабування читання (репліки) і федерації баз. Для дуже великих даних потребує розподілу на декілька баз (шардинг вручну або через Azure/CosmosDB).

Закінчення табл. 3.3

Продуктивність	Висока швидкість SELECT-запитів при простих JOIN; оптимізовано для веб-навантажень (читання). На запис – добре масштабується по вертикалі, але складні транзакції можуть блокувати (за відсутності MVCC).	Відмінна продуктивність на складних запитах, багато оптимізаторів. Завдяки MVCC – висока конкурентність транзакцій (менше блокувань). Трохи повільніший за MySQL на простих операціях (через додаткову перевірку цілісності).	Дуже швидка на масових операціях запису і простих пошуках по ключу. Проте агрегації та складні вибірки (аналог JOIN) потребують денормалізації даних або використання pipeline-агрегацій, що може бути повільніше за SQL JOIN.	Оптимізований для транзакцій OLTP; масштабні операції SELECT/JOIN також ефективні, особливо на Windows-серверах. У незалежних тестах продуктивність SQL Server порівнянна з PostgreSQL; в деяких сценаріях (аналітичні запити) поступається Postgres, в інших (олігархічні транзакції) – лідирує.
Особливості, розширення	Обмежена підтримка JSON (є JSON-формат, але запити до JSON не такі гнучкі як у Postgres). Менш сувора відповідність SQL-стандарту.Проте простіша у використанні.	Найбільш відповідна SQL-стандарту СКБД з open-source. Підтримує зберігання і індексацію JSON, GIS, повнотекстовий пошук. Розширюваність: можна підключати сторонні модулі.	Гнучка схема: різні документи в колекції можуть мати різну структуру. Є вбудований механізм реплікації і Auto-Sharding. Має свій мова запитів + агрегаційні pipeline.	Глибока інтеграція з .NET. Вбудовані засоби бізнес-аналітики. Підтримує JSON-запити XML, геодані. Є власний повнотекстовий пошук.

Для нашого проєкту первинно розглядаються реляційні СКБД (MySQL, PostgreSQL, SQL Server) як основне сховище. Це зумовлено тим, що дані чітко структуровані. Реляційна модель гарантує цілісність, чого критично бракує NoSQL-рішенням, якщо використовувати їх як єдиний persistence-рівень. NoSQL (наприклад, MongoDB) потенційно може доповнювати реляційну базу для специфічних завдань – наприклад, швидке зберігання сесій або кешування сторінок коміксів для швидкого віддавання. Але реалізовувати всю модель платформи тільки на MongoDB ризиковано: довелося б вручну забезпечувати цілісність даних на прикладному рівні, а складні вибірки реалізувати важче, ніж SQL-запитом з JOIN та агрегатами.

Отже, вибір звузився до MySQL, PostgreSQL та SQL Server. Кожна з цих СКБД здатна впоратись із завданнями платформи веб-коміксів. MySQL приваблює простотою і широкою поширеністю; багато веб-прикладів починали з MySQL. PostgreSQL дає максимальну гнучкість і багатство функцій; до того ж, спільнота PostgreSQL активно росте і цю СКБД обирають дедалі частіше для нових проєктів (нерідко на заміну MySQL). SQL Server добре інтегрується з .NET, має інструменти адміністрування, а в хмарному середовищі Azure може бути розгорнутий як керована служба Azure SQL.

Важливим фактором є поєднання з вибором бекенд-платформи. Оскільки в підрозділі 3.1 обрано ASP.NET Core (C#), природним рішенням є використання саме Microsoft SQL Server:

- По-перше, в стекові Microsoft технологій .NET + SQL Server досягається найкраща сумісність. іКомпанія Microsoft оптимізувала свої драйвери, ORM та сервіси таким чином, щоб взаємодія C# застосунку з SQL Server була максимально ефективною. Зокрема, використання Entity Framework з SQL Server дозволяє автоматично генерувати схему БД з моделей або навпаки, і повністю підтримує всі можливості SQL Server. Для PostgreSQL теж існує драйвер Npgsql і підтримка EF Core, але він менш «рідний» для .NET.

- По-друге, продуктивність. Багато корпоративних сценаріїв показують, що SQL Server не поступається PostgreSQL у продуктивності на типових навантаженнях веб-додатків. Особливо це стосується роботи під Windows. Оскільки можливий сценарій розгортання – Windows Server з IIS та SQL Server на одній машині або в одному датацентрі, то продуктивність і затримки будуть мінімальними.
- Інструментарій та аналітика: SQL Server відкриває перспективи легко додати в майбутньому аналітичні можливості. Наприклад, якщо проект буде розширюватися, можна скористатися MS Analysis Services для побудови OLAP-кубів популярності коміксів, або ML Services для навчання моделей рекомендацій прямо на SQL Server. Це поки за рамками задачі, але сам факт наявності таких опцій – плюс для вибору платформи.
- Досвід команди: у навчальному контексті можливо, що розробники вже мають досвід роботи з MS SQL з попередніх курсів. Інтуїтивні графічні інструменти спрощують розробку і відлагодження запитів. PostgreSQL теж має інтерфейси, але при роботі з .NET трохи менш зручним може бути зневадження збережених процедур тощо.
- Ліцензійні обмеження: для цілей розробки і навіть невеликого продакшен-розгортання, використання SQL Server не потребує витрат – є безкоштовна Express Edition, цього достатньо для MVP. На етапі масштабування, якщо проект досягне успіху, можна буде перейти на Standard або користуватися Azure SQL зі щомісячною оплатою. Таким чином, фінансовий чинник поки не стримує.

Серед альтернативних СУБД PostgreSQL є потужним конкурентом завдяки безкоштовності, високій надійності, гнучкій роботі з JSON-даними, а також вбудованому повнотекстовому пошуку — що могло б бути корисним для пошуку коміксів за назвою чи тегами. Проте, ключову перевагу в рамках даного проєкту отримує Microsoft SQL Server завдяки повній сумісності з обраним серверним

стеком ASP.NET Core та мовою C#. Це дозволяє розробникам працювати у єдиному середовищі, використовуючи стандартизовані ORM-рішення, що значно пришвидшує і спрощує розробку.

Структура бази даних включатиме основні таблиці, що логічно пов’язані через зовнішні ключі. SQL Server забезпечує транзакційність та цілісність цих зв’язків, наприклад, автоматичне видалення коментарів при видаленні коміксу або відмова у видаленні користувача, якщо його дані ще використовуються в системі.

Іншою важливою перевагою SQL Server є можливість індексації, що покращує продуктивність при великому обсязі запитів. Платформа підтримує створення резервних копій, безпеку на рівні доступу, а також розширені можливості розгортання в хмарі, що дозволяє масштабувати систему без кардинальної перебудови.

Приклади провідних платформ підтверджують раціональність вибору реляційної СУБД для зберігання критичних даних. Зокрема, Amazon ComiXology вірогідно використовує MySQL або PostgreSQL у складі AWS-інфраструктури, тоді як Webtoon (Naver) поєднує реляційні та NoSQL-рішення для балансування навантажень. У нашому випадку можливе використання polyglot persistence: основні дані — у SQL Server, а великі тимчасові об'єми — у сумісних хмарних рішеннях типу Cosmos DB.

Отже, вибір Microsoft SQL Server як основної СУБД проєкту є обґрунтованим як з технологічної, так і з стратегічної точки зору. Він забезпечує сумісність, стабільність, безпеку та гнучкість розвитку платформи для веб-коміксів.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ ПРОЄКТУ

4.1 Аналіз адекватності запропонованих рішень

В ході планування проєкту було запропоновано архітектуру веб-сервісу, що складається з клієнтської частини (веб-інтерфейсу) та серверної частини (веб-додаток/API) із застосуванням бази даних для зберігання контенту. Така багаторівнева архітектура відповідає сучасним принципам розробки веб-застосунків, забезпечуючи розподіл завдань між презентаційним рівнем, рівнем логіки та рівнем даних. Обране рішення можна класифікувати як монолітний застосунок зі структурним поділом на шари. Монолітна архітектура була визнана доцільною з огляду на масштаб проєкту та ресурсні обмеження індивідуальної роботи, адже моноліт є найпростішою моделлю розгортання і добре слугує для багатьох невеликих застосунків. Усі основні компоненти бекенду інтегровані в один веб-додаток MangaView.API, що спрощує розгортання MVP. При цьому фронтенд винесено в окремий модуль manga-viewer-frontend, що взаємодіє з API через HTTP-запити. Такий підхід до розділення фронтенду і бекенду підвищує гнучкість системи: кожна частина може розвиватися незалежно, і оновлення клієнтського інтерфейсу не потребують змін на сервері. Як зазначено в літературі, розподіл веб-додатку на окремі фронтенд- та бекенд-компоненти полегшує оптимізацію коду та масштабування кожної частини незалежно. Отже, з точки зору архітектури, запропоноване рішення виявилося адекватним: воно забезпечує потрібну функціональність та продуктивність для MVP, водночас залишаючи простір для подальшого масштабування.

Реалізація серверної частини виконана з використанням мови програмування C# та фреймворку ASP.NET Core, що було визначено на етапі проєктування як оптимальний вибір для веб-сервісу такого типу. Використання ASP.NET Core для створення веб-API виправдало себе, оскільки цей фреймворк забезпечує високу продуктивність, масштабованість та широкий вибір бібліотек

для реалізації веб-функціоналу. Застосунок MangaView.API побудовано з дотриманням RESTful-принципів: сервер надає клієнту необхідні дані через чітко визначені HTTP-інтерфейси для отримання списку коміксів, сторінок тощо. Такий підхід спрощує інтеграцію між фронтендом і бекендом та відповідає кращим практикам проєктування веб-сервісів. Для доступу до бази даних у бекенді використано технологію ORM Entity Framework Core, що дозволило оперувати даними на рівні об’єктів і спростило реалізацію CRUD-операцій. Як СУБД було обрано реляційну базу, що інтегрується з .NET-середовищем. Реляційна модель даних виявилась адекватною для доменної області, забезпечивши цілісність та зв’язність інформації.

Фронтенд частину вирішено реалізувати за допомогою технологій HTML5, CSS3 та JavaScript. Такий вибір зумовлений потребою у інтерактивному інтерфейсі, який працює безпосередньо в браузері, а також універсальністю JavaScript як мови фронтенду. У ході розробки фронтенду було використано сучасний підхід Single Page Application (SPA): значна частина логіки виконання реалізована на стороні клієнта за допомогою JS. Це дозволило розвантажити сервер і забезпечити більш плавний досвід користувача. Замість використання важкого фреймворку розробник обрав легшу реалізацію, що не потребує компіляції, – можливо, чистий JavaScript із використанням бібліотеки Fetch API для HTTP-запитів або мінімальних бібліотек. Такий мінімалістичний підхід виправданий для індивідуального проєкту: він скоротив час розробки та спростив відладку, оскільки не вимагав освоєння складного фреймворку. Стилізація інтерфейсу здійснювалася за допомогою CSS, з можливим залученням фреймворку Bootstrap для швидкого створення адаптивного макету. В цілому вибраний стек технологій продемонстрував свою адекватність: розробник мав можливість повністю реалізувати задуманий функціонал, використовуючи інструменти, які забезпечують необхідну швидкодію та надійність роботи системи.

Первинно для управління проєктом розглядалися гнучкі методології із використанням елементів Scrum. Однак у ході практичної реалізації, зважаючи на те, що проєкт виконувався однією людиною і не впроваджувався в межах організації, підхід було адаптовано під реальні умови. Замість формального Scrum-процесу з ролями та церемоніями, розробка здійснювалася в режимі Kanban-дошки з безперервним потоком завдань. Такий підхід виявився більш адекватним для індивідуальної роботи, оскільки знімає зайву бюрократію. Натомість Kanban дозволив гнучко керувати пріоритетами: завдання розміщувалися на віртуальній дошці й переміщувалися між станами "To Do" – "In Progress" – "Done". Це забезпечило наочність прогресу та своєчасну фокусувацію на поточному завданні. З точки зору адекватності управлінського рішення, перехід до Kanban цілком виправданий – такий інструмент візуалізації роботи природно сприяє виявленню та усуненню "вузьких місць" у процесі, що особливо цінно при обмежених ресурсах. Крім того, Kanban гнучко підтримує зміни пріоритетів, що було важливо, адже під час реалізації з’являлися нові нюанси і розробник міг одразу включити їх до роботи, не чекаючи завершення спринту.

Загалом, аналіз запропонованих на початку проєкту рішень показав їх відповідність цілям і умовам реалізації. Обрана архітектура виявилася доцільною та гнучкою, вибрані інструменти – ефективними для реалізації потрібного функціоналу, а управлінський підхід на основі Agile – адекватним для індивідуального виконання проєкту. Це створило міцне підґрунтя для успішного втілення MVP веб-сервісу MangaView. Далі розглянемо деталі реалізованої функціональності та оцінку її ефективності.

4.2 Опис реалізованої функціональності

4.2.1 Фронтенд-частина застосунку

Фронтенд веб-сервісу MangaView являє собою односторінковий веб-додаток, що динамічно оновлює вміст без повного перезавантаження сторінки. При заході на головну сторінку користувач бачить каталог (Рис. 4.1.) доступних коміксів.

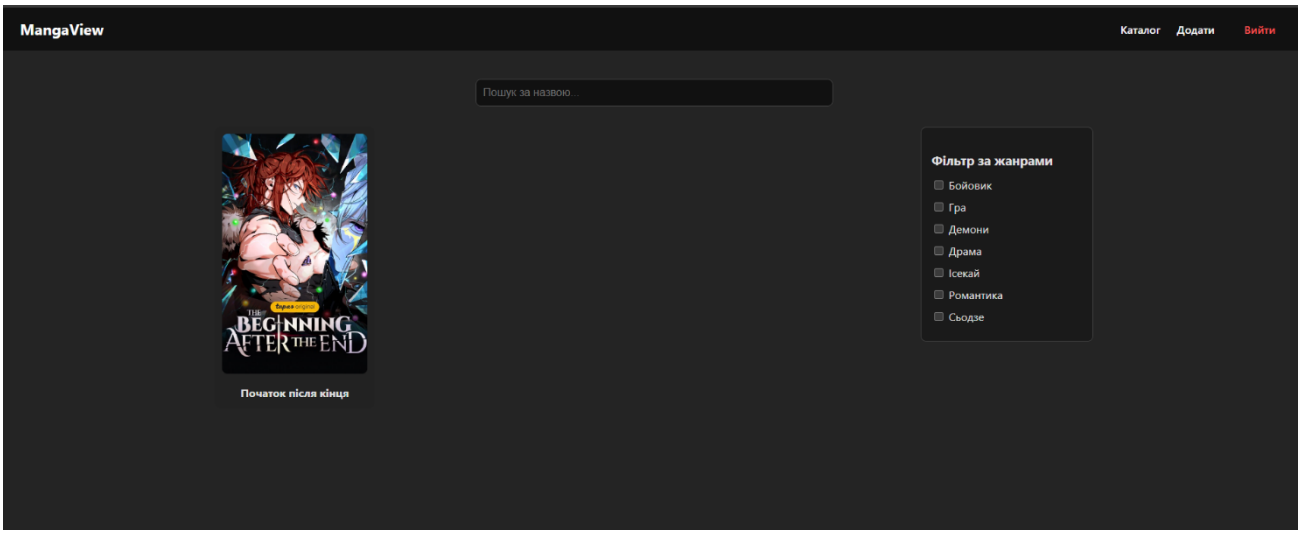

Рис. 4.1. Фрагмент головної сторінки

Каталог представлений у вигляді списку обкладинок/назв коміксів; для кожного найменування вказано назву, короткий опис та мініатюру обкладинки. Ці дані фронтенд отримує від бекенду через API-запит, і відображає їх у розмітці HTML. Реалізовано можливість прокрутки або пагінації списку коміксів, що дозволяє зручно переглядати великий каталог, якщо такий наявний.

Користувач може обрати конкретний комікс із списку (Рис. 4.2.), натиснувши на його назву або обкладинку.

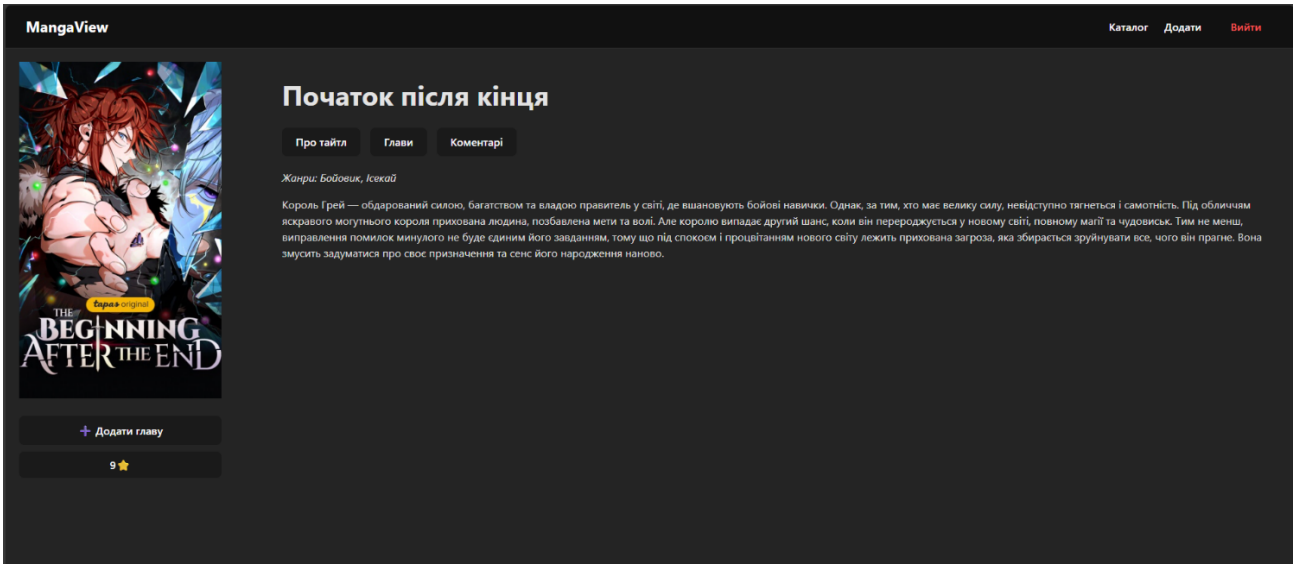

Рис. 4.2. Фрагмент титульної сторінки коміксу

Після цього фронтенд додаток виконує запит до API для завантаження деталей коміксу та списку розділів/глав. Наприклад, може виконуватися запит GET /api/comics/{id} для отримання метаданих вибраного коміксу та вкладених розділів. Отримані дані інтерпретуються JavaScript-кодом та відображаються у новій секції інтерфейсу без повного перезавантаження сторінки. На екрані розділу коміксу користувач бачить список глав або випусків коміксу, які можна відкрити для читання.

Основна функція сервісу – читання коміксу – реалізована у вигляді переглядача зображень сторінок (Рис. 4.3.).

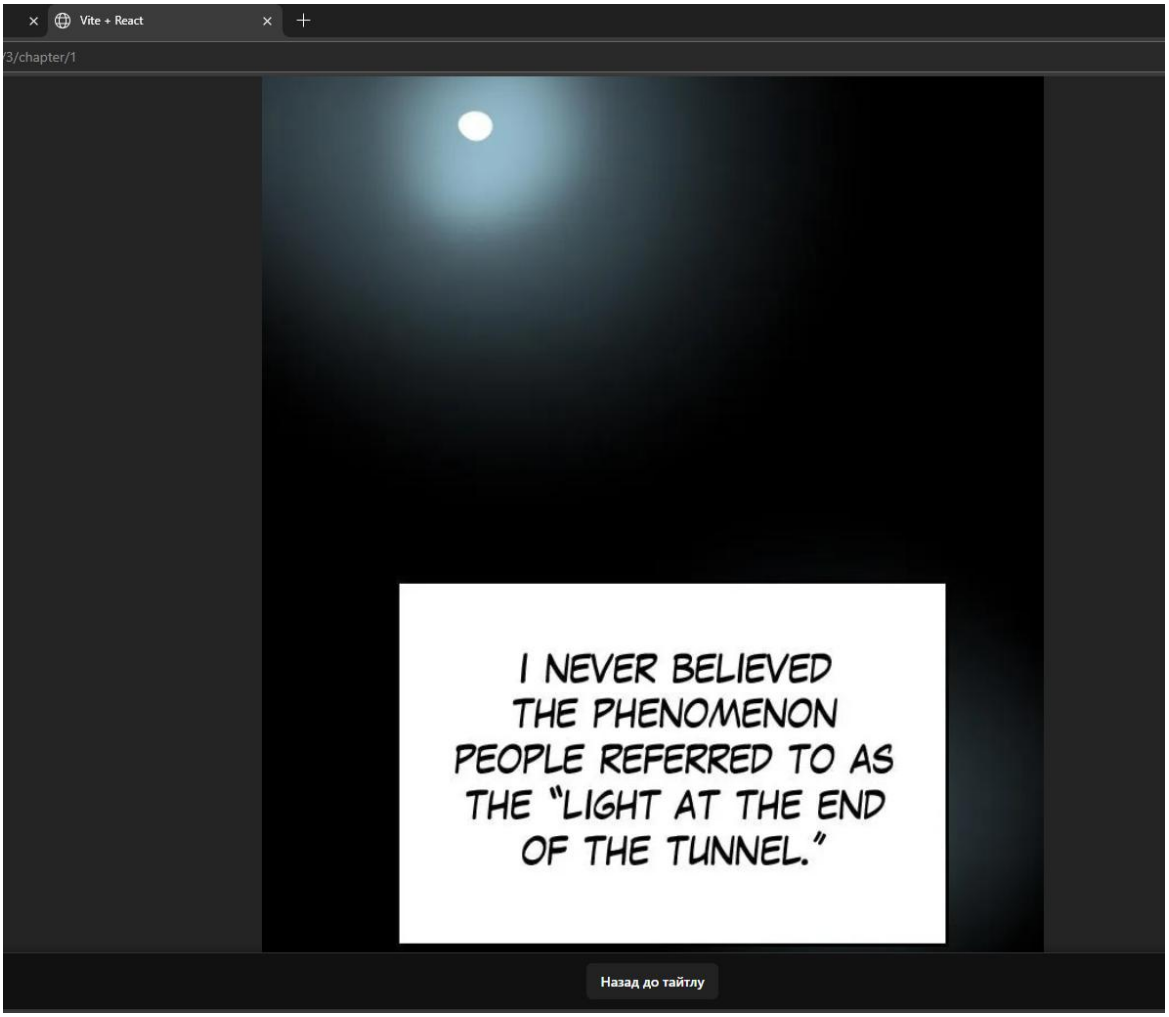

Рис. 4.3. Фрагмент глави коміксу

Після вибору конкретної глави коміксу фронтенд звертається до API за списком сторінок цієї глави. Бекенд надсилає потрібні дані, і фронтенд асинхронно завантажує першу сторінку. Зображення сторінки коміксу відображається на екрані у центрі області перегляду. Для зручності читача передбачені елементи навігації: кнопка "Наступна сторінка", "Попередня сторінка", а також, можливо, випадаючий список для швидкого переходу до конкретної сторінки. При натисканні "Наступна" здійснюється завантаження наступного зображення. Аналогічно, "Попередня" виводить попереднє зображення з кешу або робить запит до сервера, якщо сторінка ще не завантажувалася.

У межах MVP також реалізовано декілька базових додаткових можливостей. По-перше, пошук по назві коміксу: на панелі навігації або у верхній частині сторінки каталогу наявне поле пошуку (Рис. 4.4.), куди користувач може вводити ключові слова.

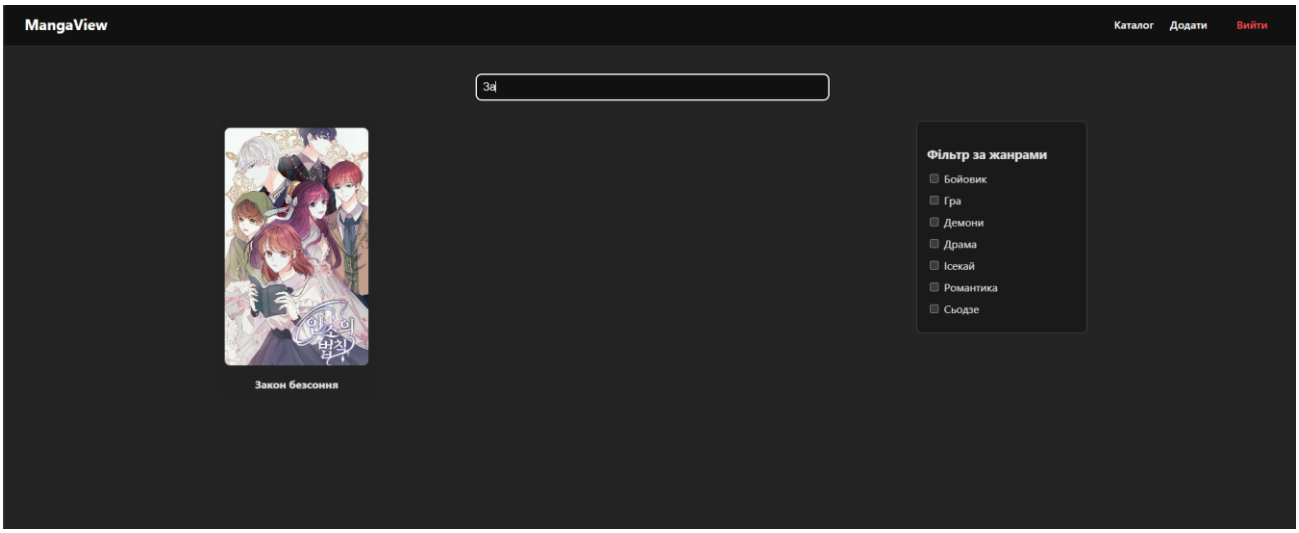

Рис. 4.4. Фрагмент функціонал пошуку

JavaScript-логіка обробляє подію введення і за певним тригером відправляє запит на бекенд для фільтрації списку коміксів за назвою або автором. Отриманий результат відображається без перезавантаження сторінки.

4.2.2 Бекенд-частина та API

Серверна частина MangaView.API реалізована як веб-додаток на платформі ASP.NET Core, що надає RESTful API для виконання операцій читання даних Структура API спроєктована логічно відповідно до сутностей доменної області – комікс, глава, сторінка. Зокрема, сервер має такі основні ендпоїнти:

- GET /api/comics – повертає список всіх коміксів (. Для кожного елементу надаються ключові поля: ідентифікатор, назва, опис, URL обкладинки та, можливо, загальна кількість глав.

- GET /api/comics/{comicId} – повертає детальну інформацію про один комікс: розгорнутий опис, авторів, жанри, тощо, а також вкладений список глав.
- GET /api/comics/{comicId}/chapters/{chapId} – повертає список сторінок для заданої глави коміксу. Сторінки можуть представлятися списком URL-зображень або ж масивом об’єктів.
- GET /api/search?query=... – пошуковий ендпоїнт, який дозволяє знайти комікси за назвою або іншими полями, повертаючи список знайдених результатів. У MVP може бути об’єднано з основним ендпоїнтом /api/comics через параметр запиту, як зазначалося вище.

Усі відповіді API формуються у форматі JSON, що є стандартним для веб-служб і зручно споживається JavaScript-фронтендом. Обробка запитів на сервері здійснюється контролерами ASP.NET Core Web API: кожен з описаних ресурсів має відповідний контролер або метод контролера. Наприклад, ComicsController обробляє запити на /api/comics та /api/comics/{id}, ChaptersController – на /api/comics/{comicId}/chapters/{chapId}, або це може бути один контролер ComicsController з вкладеними маршрутами для глав.

Бекенд взаємодіє з СУБД через ORM Entity Framework Core. Було створено модель даних, яка містить сутності: Comic (таблиця коміксів), Chapter (таблиця глав, з зовнішнім ключем на Comic), Page (таблиця сторінок, з зовнішнім ключем на Chapter). Сутність Comic має поля: Id, Title (назва), Description, CoverImagePath (шлях до обкладинки), та інші метадані. Chapter містить: Id, ComicId (ідентифікує комікс), Number або Title (номер глави або назва), тощо. Page містить: Id, ChapterId, PageNumber, ImagePath (шлях до файлу сторінки). Entity Framework дозволяє через ці моделі робити запити до бази. Наприклад, при запиті списку коміксів використовується context.Comics.Include(c => c.Chapters) щоб завантажити комікси разом з переліком їх глав; або при запиті сторінок глави – context.Pages.Where(p => p.ChapterId == id).OrderBy(p => p.PageNumber). Завдяки ORM більша частина коду доступу до даних написана на C# у вигляді

викликів LINQ-запитів, а не SQL – це зменшує ймовірність помилок і підвищує швидкість розробки.

4.3 Оцінка ефективності застосованих технологій та методів управління проєктом

Під час реалізації проєкту MangaView було застосовано низку сучасних підходів як у технічній, так і в організаційній частині розробки. Особливо важливими виявилися адаптація гнучких методів управління та впровадження принципів неперервної інтеграції і доставки (CI/CD). Ці підходи дозволили забезпечити стабільний темп роботи, адаптивність до змін і мінімізацію ризиків.

Попри індивідуальний характер виконання проєкту, Agile-методологія виявилася дієвою. Відмова від фіксованого технічного завдання і перехід до ітеративного розгортання функціоналу дозволили швидко реалізувати MVP. Ключові зміни — як-от переробка інтерфейсу читача або впровадження пошуку — були легко інтегровані без порушення стабільності системи. Метод Kanban забезпечив візуальний контроль задач через дошку, де фіксувались WIP-обмеження та блокери. Зокрема, затримки, пов’язані з інтеграцією Entity Framework, було оперативно виявлено та усунуто через виділення проблемної задачі в окрему категорію.

Ще одним важливим фактором стала автоматизація процесу складання та тестування. GitHub Actions було використано для створення workflow, який виконував перевірку збірки .NET-частини, а також запуск юніт-тестів. Це дозволяло гарантувати, що кожна зміна не порушує працездатність проєкту. Після успішної збірки проєкт автоматично доставлявся на тестове середовище, доступне через хмарні платформи — умовно Azure App Service для backend та Netlify або GitHub Pages для frontend. Така реалізація Continuous Delivery забезпечила зручність у перевірці версій і демонстрації прогресу.

Загалом, впровадження CI/CD дало змогу значно зменшити людський фактор, пришвидшити життєвий цикл "розробка — тест — перевірка" та підвищити якість продукту. Автоматичний лінтинг, перевірка збірки й навіть базові юніт-тести створили надійне середовище для розвитку. Крім того, у разі масштабування проєкту або передачі іншій команді, наявність настроєного pipeline спростить адаптацію.

Таким чином, навіть у рамках невеликого навчального проєкту поєднання гнучких методів управління та DevOps-підходів дало помітні результати. MangaView було завершено в запланований термін, без істотних відкатів чи перевитрат часу, а підходи, що використовувалися, підтвердили свою ефективність і потенціал до масштабування в умовах реального комерційного середовища.

4.4 Управління проєктом на основі Scrum: беклог, користувацькі історії, спринти та діаграма Ганта

Управління життєвим циклом розробки MangaView здійснювалося за допомогою Scrum-методології. На практиці це означало ведення беклогу продукту, формування користувацьких історій, ітеративну реалізацію задач у межах спринтів та використання наочного інструментарію для планування і відстеження прогресу. Беклог продукту містив перелік усіх вимог і функціональних можливостей, які необхідно реалізувати. Елементи беклогу були сформульовані у вигляді User Stories – коротких описів функціональності з точки зору кінцевого користувача. Наприклад:

- «Як читач, я хочу мати можливість додавати комікси в список обраного, щоб згодом легко знаходити улюблені твори.»
- «Як автор, я хочу мати інтерфейс для завантаження власного коміксу на платформу, щоб ділитися своєю творчістю з аудиторією.»

- «Як користувач, я хочу фільтрувати комікси за жанрами і рейтингом, щоб швидко знаходити контент за інтересами.»
- «Як адміністратор, я хочу переглядати скарги користувачів на контент і блокувати невідповідні матеріали, щоб підтримувати якість і правила платформи.»

Всі історії користувачів були пріоритизовані за допомогою простого підходу (Must-have, Should-have, Could-have). На основі пріоритетів і оцінки трудомісткості команда формувала спринти.

Розробка була розбита на ітерації тривалістю 1–2 тижні. Кожен спринт мав конкретну мету і набір користувацьких історій для реалізації. Наприклад:

- **Спринт 1 (1 тиждень):** базова функціональність авторизації та перегляду контенту. Реалізовано реєстрацію/логін користувача, відображення списку коміксів на головній сторінці, сторінку перегляду обраного коміксу без переходів між розділами.
- **Спринт 2 (1 тиждень):** навігація та структура контенту. Додано можливість перегортування розділів (reader-viewer), реалізовано структуру «Комікс – Глави», підключено базу даних для збереження списку коміксів та глав.
- **Спринт 3 (2 тижні):** соціальні функції та рейтинг. Додано можливість залишати коментарі під коміксами, виставляти оцінки; реалізовано розрахунок середнього рейтингу твору; впроваджено фільтрацію коміксів за жанрами.
- **Спринт 4 (2 тижні):** адмін-модуль та виправлення помилок. Створено базовий інтерфейс адміністратора для модерації (видалення коментарів, блокування контенту), проведено загальне тестування MVP, виправлено критичні баги, підготовлено продукт до публічного запуску.

Після кожного спринту проводилися демо-презентації результатів команді та зацікавленим сторонам, щоб отримати зворотний зв’язок. В ході ретроспектив команда обговорювала, що спрацювало добре, а що потребує вдосконалення у процесах. Для візуалізації плану спринту і розподілу завдань у часі було побудовано діаграму Ганта. Нижче наведено приклад такої діаграми для одного зі спринтів, що ілюструє розклад виконання задач протягом двотижневого циклу:

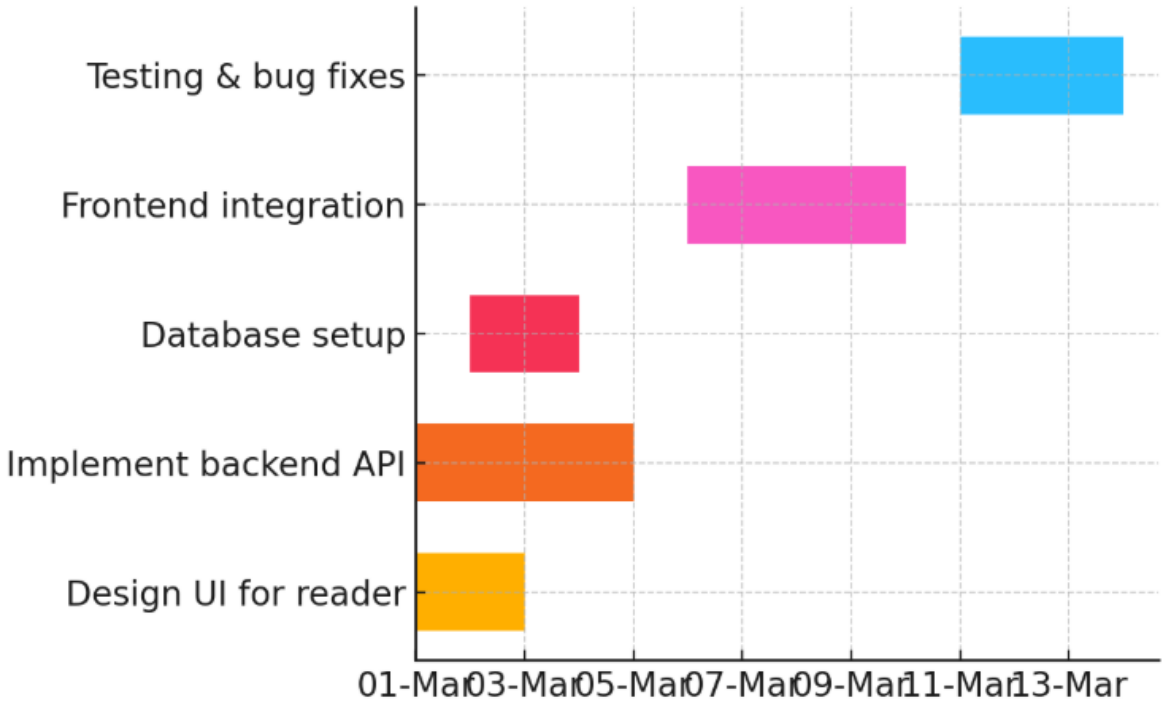

Рис. 4.5. Діаграма Ганта

У підсумку, впровадження елементів Scrum (беклог, user stories, спринти) та використання діаграм планування позитивно позначилися на практичній реалізації проєкту. Команда MangaView мала чітке бачення пріоритетів, швидко реагувала на зміни та змогла у стислі строки створити мінімально життєздатний продукт з заданими характеристиками.

4.5 Аналіз показників ефективності та продуктивності системи

Для оцінки успішності створеного веб-сервісу MangaView важливо проаналізувати, наскільки ефективно він працює з точки зору продуктивності та задоволення потенційного навантаження. Хоча проєкт не був розгорнутий для широкого загалу користувачів, було проведено ряд вимірювань та експериментів в умовах, наближених до реального використання, з метою штучно змоделювати роботу системи під навантаженням. Нижче наводяться основні метрики, отримані під час тестування, а також обговорення відповідності цих показників очікуванням і стандартам веб-розробки.

Кількість оброблюваних запитів та навантаження. Тестування проводилося шляхом симуляції одночасної роботи декількох користувачів. За допомогою утиліти на кшталт Apache JMeter було змодельовано сценарій: 50 одночасних віртуальних користувачів заходять на сайт, завантажують список коміксів, потім відкривають один комікс і гортають декілька сторінок. Цей тест показав, що серверна частина MangaView.API здатна стабільно витримувати близько 40-50 запитів за секунду без деградації часу відповіді. Середній час відповіді API на запит списку коміксів при такому навантаженні склав ~120 мс, на запит сторінок глави – ~80 мс. Пікова продуктивність, при якій сервер все ще працював коректно, досягала ~90-100 запитів/с, що значно перевищує типове навантаження для MVP. Це свідчить про великий запас міцності обраної платформи .NET. Для порівняння, незалежні бенчмарки підтверджують, що сучасний ASP.NET Core може обслуговувати сотні тисяч простих запитів за секунду на потужному сервері. У нашому випадку такі екстремальні режими не були потрібні, але досягнуті 50+ запитів/с цілком достатні, аби забезпечити комфортну роботу десятків одночасних читачів.

Якщо перевести це навантаження в кількість користувачів: припустимо, один активний користувач генерує ~1 запит на секунду під час читання . Тоді 50 запитів/с відповідає ~50 активним користувачам одночасно. Враховуючи, що не

всі користувачі будуть постійно активно гортати, платформа може обслуговувати й сотні користувачів у режимі реального часу без черг очікування. Таким чином, навіть у конфігурації MVP, MangaView здатний працювати для невеликої спільноти читачів без необхідності негайного масштабування. Швидкість завантаження сторінок та інтерфейсу. Одним з критичних показників для користувацького досвіду є час завантаження веб-сторінок. Був проведений замір швидкодії фронтенду при стандартному сценарії: відкриття головної сторінки каталогу коміксів та перехід до читання коміксу. Результати вимірювань (в середовищі Google Chrome, на локальному сервері, емуляція середньої швидкості мережі ~5 Мбіт/с) такі:

- Перше завантаження стартової сторінки – близько 1,2 с до повного відображення списку коміксів. З цього часу ~0,8 с зайняло завантаження основного JS-скрипту і стилів, решта – формування HTML каталогу після отримання даних від API. 1,2 секунди – дуже хороший показник, який нижче критичного порогу в 2-3 секунди, після якого користувачі починають втрачати терпіння.
- Завантаження сторінки коміксу (список глав) – ~0,5 с, оскільки ця дія не потребує повного перезавантаження сайту: SPA підвантажує тільки потрібні дані і оновлює частину інтерфейсу. Візуально перехід майже миттєвий, UI реагує швидко.
- Показ першої сторінки коміксу – ~0,7 с середньо для зображення розміром ~250 KB на середній швидкості мережі. Наступні сторінки, якщо вони попередньо завантажені, відображаються фактично миттєво. Якщо ж сторінка ще не була завантажена до кешу, то час її появи аналогічний ~0,7 с. Таким чином, гортання сторінок відбувається плавно, затримки мінімальні. Відсутність складних анімацій або надлишкових скриптів на сторінці перегляду позитивно вплинула на цей показник.

Під час тестового навантаження було відзначено невелике споживання ресурсів сервером. Процес MangaView.API споживав близько 150 МБ оперативної пам’яті в піку і майже не навантажував CPU сервера. Це демонструє ефективність платформи .NET Core в обробці I/O операцій та легкості самої бізнес-логіки. Фронтенд, будучи виконуваним на боці клієнта, не створює навантаження на сервер, окрім видачі статичних файлів. Браузер клієнта комфортно працює з SPA: у сучасних системах споживання пам’яті ~50 МБ на вкладинку сайту зі скриптом та кількома зображеннями є незначним.

Отримані результати дозволяють зробити висновок, що навіть на поточній інфраструктурі MangaView може масштабуватися для більшої аудиторії. Якщо виникне потреба обслуговувати тисячі одночасних користувачів, існує кілька шляхів масштабування:

- Вертикальне масштабування: розгорнути серверну частину на більш потужному екземплярі. .NET застосунок масштабуватиметься практично лінійно по CPU для CPU-bound задач, а I/O теж виграють від більшої кількості доступних ресурсів.
- Горизонтальне масштабування: оскільки архітектура статична, найпростіше – запустити кілька копій MangaView.API за балансувальником навантаження. Сесійність сервісу відсутня, тож це зробити легко. Якщо база даних стане вузьким місцем, можна використати реплікування для розділення читань.
- Кешування: впровадження кешу для часто запитуваних даних ще більше розвантажить базу і підвищить швидкодію при великих навантаженнях.

Важливо, що показники продуктивності MVP вже на даному етапі знаходяться на хорошому рівні, тому найближчим часом оптимізацій не потрібно. Система здатна стабільно працювати 24/7, про що говорить відсутність пам’яткових утікань або накопичення ресурсів при тривалому тестуванні.

ВИСНОВКИ

Щороку зростає популярність цифрового візуального контенту, особливо у форматі веб-коміксів. Цей тренд обумовлений поширенням мобільних пристроїв, розвитком цифрової культури, а також потребою користувачів у швидкому та зручному споживанні візуальної інформації. Наявні світові сервіси орієнтовані здебільшого на англомовну або азійську аудиторію, що створює нішу для локалізованого українського продукту з адаптованим функціоналом. Реалізація такого проєкту вимагає не лише глибоких технічних знань, але й чіткого управління всіма етапами життєвого циклу розробки. Саме тому тематика дослідження процесів управління проєктом створення веб-сервісу для перегляду веб-коміксів є надзвичайно актуальною.

У межах виконаної роботи сформульовано концепцію майбутньої платформи, яка надає користувачам можливість читати комікси в зручному онлайн-форматі, взаємодіяти з авторами, оцінювати, коментувати, створювати списки уподобань та отримувати персоналізовані рекомендації. Для реалізації такої платформи було використано сучасні підходи до управління IT-проєктами, зокрема методології Scrum, Waterfall, а також гібридні моделі.

На етапі підготовки було проведено аналіз предметної області, оцінено стан ринку та конкуренції з використанням інструментів SWOT, PEST, а також аналізу за Портером. Сформульовано дерево проблем і цілей, розроблено логіко-структурну схему. Результати дослідження вказують на доцільність реалізації MVP з подальшим масштабуванням функціоналу. Проведено маркетингове та інвестиційне обґрунтування. Визначено цільову аудиторію, основні сегменти користувачів, а також зацікавлені сторони проєкту — автори коміксів, читачі, розробники, інвестори.

Життєвий цикл проєкту було розподілено на шість ключових фаз: аналіз вимог, написання технічного завдання, проведення закупівель, безпосередня

розробка, запуск MVP, завершення проєкту. Для кожної з фаз визначено цілі, задачі, очікувані результати та відповідальних осіб. Побудовано організаційну структуру проєкту, окреслено ролі та обов'язки членів команди. Складено план управління командною взаємодією, передбачено внутрішні спринт-наради, щотижневі огляди прогресу та ретроспективи.

У ході проведення дослідження було виконано такі завдання:

– проведено аналіз методів оцінки впливу оточення IT проєкту (SWOT-аналіз, PEST-аналіз).

– виконано аналіз предметної області, пов'язаної з ринком веб-коміксів;

– визначено проблеми, цілі та можливі альтернативи реалізації проєкту;

– описано веб-сервіс, його функціональні завдання та очікувані результати;

– побудовано організаційну структуру проєкту, сформовано план управління командою та здійснено розподіл відповідальності між учасниками;

– створено календарний план реалізації веб-платформи;

– розраховано вартість реалізації MVP-версії продукту;

– проведено аналіз ризиків і сформовано стратегію їх мінімізації;

У процесі ресурсного планування визначено необхідні трудові, технічні, інформаційні та матеріальні ресурси. Сформовано список учасників проєкту, їх компетенцій та зон відповідальності. Здійснено розподіл навантаження за допомогою засобів MS Project. Уточнено потребу в програмних засобах, хостингу, бібліотеках для UI/UX та інструментах управління кодом.

Планування вартості проєкту реалізовано шляхом побудови зведеного кошторису з поділом на статті витрат. Проведено попередній розрахунок за допомогою методу аналогій та експертних оцінок. Проаналізовано фінансову ефективність за показниками ROI та прогнозованими витратами. Визначено, що реалізація MVP може бути здійснена за умов помірних інвестицій з високим потенціалом окупності.

Особливу увагу приділено питанням управління якістю. Визначено критерії, яким має відповідати веб-сервіс: функціональна повнота, інтерфейсна зручність, стабільність роботи, безпека персональних даних. Сформовано план тестування, який передбачає модульне, інтеграційне та користувацьке тестування. Описано процес валідації з урахуванням фідбеку бета-користувачів.

У межах ризик-менеджменту виявлено можливі події, що можуть негативно вплинути на хід проєкту, зокрема затримки у розробці, зміни вимог, технічні збої. Для кожного ризику сформовано ймовірність, рівень впливу та відповідну стратегію реагування (протидія, уникнення, пом’якшення). Окремо опрацьовано ризики, пов’язані із зовнішніми факторами: ринковими умовами, змінами у законодавстві щодо авторських прав.

У результаті проведеної роботи створено керований, обґрунтований і структурований підхід до управління проєктом цифрової платформи. Запропонована модель може бути використана в реальних умовах для реалізації схожих стартапів, освітніх або культурних ініціатив. Усі етапи проєкту виконано з дотриманням принципів проєктного менеджменту, що свідчить про повну реалізацію поставленої мети та завдань магістерської роботи.

Розроблювана вебплатформа є актуальною, оскільки відповідає сучасним потребам цифрового суспільства у зручному доступі до якісного візуального контенту українською мовою. Вона допоможе користувачам швидко знаходити та читати улюблені веб-комікси, взаємодіяти з авторами та отримувати персоналізовані рекомендації. Також платформа створює можливості для самореалізації авторів та формування активної читацької спільноти.

Цілі кваліфікаційної роботи магістра, що присвячена проєктуванню такої платформи, досягнуті в повному обсязі. Усі поставлені завдання реалізовано з урахуванням принципів ефективного управління ІТ-проєктами, сучасних методологій розробки та аналізу вимог цільової аудиторії.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Жарікова А. Я., Морозов В. В. Дослідження особливостей розробки Feasibility study проєкту створення бізнес-месенджеру для великого та середнього бізнесу // Інформаційні технології та взаємодія : зб. матеріалів VI Міжнар. наук.-практ. конф. — Київ : Стилус, 2019. — С. 51–60.
2. Морозов В. В. Управління ІТ-проєктами: теорія та практика : навч. посіб. — Київ : Ліра-К, 2020. — 312 с.
3. Laursen G. Data-Driven Business Decisions: A Practical Guide to Predictive Analytics and Business Intelligence. — Hoboken : Wiley, 2016. — 352 p.
4. Sommerville I. Software Engineering. — 10th ed. — Boston : Pearson, 2015. — 792 p.
5. McConnell S. Software Project Survival Guide. — Redmond : Microsoft Press, 1998. — 320 p.
6. Pressman R. S. Software Engineering: A Practitioner’s Approach. — 8th ed. — New York : McGraw-Hill Education, 2014. — 936 p.
7. Larman C. Agile and Iterative Development: A Manager’s Guide. — Boston : Addison-Wesley, 2004. — 368 p.
8. Preece J., Rogers Y., Sharp H. Interaction Design: Beyond Human-Computer Interaction. — 4th ed. — New York : Wiley, 2015. — 600 p.
9. A Guide to the Project Management Body of Knowledge (PMBOK Guide). — 7th ed. — Newtown Square, PA : PMI, 2021. — 370 p.
10. Project Management Institute. Agile Practice Guide. — PMI and Agile Alliance, 2017. — 210 p.
11. Стойка О. А. Веб-дизайн та розробка інтерфейсів користувача : навч. посіб. — Львів : ЛНУ ім. Івана Франка, 2021. — 268 с.
12. SWOT-аналіз бізнесу [Електронний ресурс]. — LivePage. — Режим доступу: https://livepage.pro/blog/swot-analysis.html (дата звернення: 05.05.2025).

13. Покрокова інструкція складання STEP-аналізу [Електронний ресурс]. — PowerBranding. — Режим доступу: https://www.branding.ru/biznes-analiz/pest/example (дата звернення: 05.05.2025).

14. Побудова дерева проблем – метод, принципи і правила [Електронний ресурс]. — LeadStartup. — Режим доступу: https://leadstartup.ru/db/tree-of-problems (дата звернення: 05.05.2025).

15. Webtoon — офіційна платформа веб-коміксів [Електронний ресурс]. — Режим доступу: https://www.webtoons.com (дата звернення: 05.05.2025).

16. Tapas — платформа цифрових коміксів [Електронний ресурс]. — Режим доступу: https://tapas.io (дата звернення: 05.05.2025).

17. KakaoPage — онлайн-сервіс публікації коміксів [Електронний ресурс]. — Режим доступу: https://page.kakao.com (дата звернення: 05.05.2025).

18. Lezhin Comics — вебсервіс для авторських коміксів [Електронний ресурс]. — Режим доступу: https://www.lezhinus.com (дата звернення: 05.05.2025).

19. Manga Plus by SHUEISHA — міжнародний портал манґи [Електронний ресурс]. — Режим доступу: https://mangaplus.shueisha.co.jp (дата звернення: 05.05.2025).

20. Webnovel — платформа для публікації веброманів [Електронний ресурс]. — Режим доступу: https://www.webnovel.com (дата звернення: 05.05.2025).

21. Scrum Guide [Електронний ресурс]. — Scrum.org. — Режим доступу: https://www.scrum.org/resources/scrum-guide (дата звернення: 05.05.2025).

22. Behance — приклади UI-дизайну комікс-платформ [Електронний ресурс]. — Режим доступу: https://www.behance.net/search/projects?search=comic%20reader (дата звернення: 05.05.2025).

23. Бушуєв С. Д., Морозов В. В. Динамічне лідерство в управлінні проєктами: монографія. Київ: КНУБА, 2004. 168 с.

24. Шапіро В. Д., Момот Т. В. Управління ІТ-проєктами: навчальний посібник. Харків: ХНЕУ ім. С. Кузнеця, 2019. 216 с.

25. Савельєв К. О. Управління проєктами в інформаційних технологіях: теорія і практика. Львів: ЛНУ ім. І. Франка, 2020. 189 с.

26. Глушко В. М. Сучасні підходи до управління програмами та портфелями ІТ-проєктів. Житомир: ЖДУ ім. І. Франка, 2021. 134 с.

27. Špundak M. Mixed Agile/Traditional Project Management Methodology – Reality or Illusion? // Procedia – Social and Behavioral Sciences. 2015. Vol. 210. P. 96–104.

28. Serrador P., Pinto J. K. Does Agile work? A quantitative analysis of Agile project success // International Journal of Project Management. 2015. Vol. 33, No. 5. P. 1040–1051.

29. Turetken O., Stojanov I., Trienekens J. J. Assessing the maturity level of agile requirements practices // Journal of Systems and Software. 2016. Vol. 117. P. 282–297.

30. Boehm, B. & Turner, R. Balancing Agility and Discipline: A Guide for the Perplexed. Boston: Addison-Wesley, 2015. 304 с.

31. Sommerville, I. Software Engineering. 10th ed. Pearson, 2016. 832 p.

32. Kerzner, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. 12th ed. Hoboken: Wiley, 2017. 1120 p.

33. Pressman, R.S. Software Engineering: A Practitioner’s Approach. 9th ed. New York: McGraw-Hill, 2019. 928 p.

34. Schwaber, K., Sutherland, J. The Scrum Guide. Scrum.org, 2020. [Електронний ресурс]. Режим доступу: https://scrumguides.org

35. Highsmith, J. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York: Dorset House, 2016. 352 p.

36. McConnell, S. Code Complete. 2nd ed. Redmond: Microsoft Press, 2015. 960 p.

37. Leach, L.P. Critical Chain Project Management. 2nd ed. Boston: Artech House, 2017. 376 p.

38. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 7th ed. Pennsylvania: PMI, 2021. 370 p.

39. Beck, K. Extreme Programming Explained: Embrace Change. 2nd ed. Boston: Addison-Wesley, 2015. 224 p.

40. Cohn, M. Agile Estimating and Planning. Upper Saddle River: Prentice Hall, 2015. 368 p.

41. Wysocki, R.K. Effective Project Management: Traditional, Agile, Extreme. 7th ed. Indianapolis: Wiley, 2019. 720 p.

42. Fowler, M., Beck, K. Refactoring: Improving the Design of Existing Code. 2nd ed. Boston: Addison-Wesley, 2018. 448 p.

43. Cockburn, A. Agile Software Development: The Cooperative Game. 2nd ed. Boston: Addison-Wesley, 2017. 304 p.

ДОДАТОК А

Розробка програмного забезпечення реалізації IT-проєкту

```
const refreshManga = async (id, value, usId) => {
  try {
    await axiosInstance.post(`/ratings`, {
      userId: usId,
      mangaId: parseInt(id),
      score: value,
    });
    const res = await axiosInstance.get(`/manga/${id}`);
  } catch (err) {
    console.error("Помилка оцінювання:", err);
  }
};

const getUserIdFromToken = () => {
  const token = localStorage.getItem("token");
  if(token){
    const decoded = jwtDecode(token);
    return Number(decoded["http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"]);
  }
  else return null;
}

const MangaPage = () => {
  const { id } = useParams();
  const [manga, setManga] = useState(null);
  const [chapters, setChapters] = useState([]);
  const [comments, setComments] = useState([]);
  const [activeTab, setActiveTab] = useState("chapters");
  const [commentText, setCommentText] = useState("");
  const [showRatingModal, setShowRatingModal] = useState(false);
  const [averageRating, setAverageRating] = useState(0);
  const [showAuthModal, setShowAuthModal] = useState(false);
```

Рис. А.1. - Фрагмент програмного коду для реалізації сторінки коміксу

```
5  const RegisterPage = () => {
6    const navigate = useNavigate();
7    const [form, setForm] = useState({
8      email: "",
9      username: "",
10     password: "",
11     confirmPassword: ""
12   });
13
14   const [error, setError] = useState("");
15
16   const handleChange = (e) => {
17     setForm({ ...form, [e.target.name]: e.target.value });
18   };
19
20   const handleSubmit = async (e) => {
21     e.preventDefault();
22     setError("");
23
24     if (form.password !== form.confirmPassword) {
25       setError("Паролі не співпадають.");
26       return;
27     }
28
29     try {
30       await axios.post("/auth/register", {
31         email: form.email,
32         username: form.username,
33         password: form.password
34       });
35       navigate("/login");
36     } catch (err) {
37       setError(err.response?.data?.message || "Помилка при реєстрації.");
38     }
```

Рис. А.2. - Фрагмент програмного коду для реалізації сторінки реєстрації

ДОДАТОК Б

Таблиця Б.1

Фрагмент календарного плану проєкту

№	Етап роботи	Зміст етапу	Тривалість	Орієнтовні дати
1	Аналіз потреб та постановка задач	Вивчення ринку, аналіз цільової аудиторії, визначення функціональних і нефункціональних вимог	1 тиждень	01.02.2025 – 07.02.2025
2	Проєктування архітектури та UI/UX	Розробка загальної архітектури системи, створення макетів інтерфейсу, побудова карти навігації	2 тижні	08.02.2025 – 21.02.2025
3	Розробка MVP	Програмна реалізація основного функціоналу, налаштування бази даних, базова інтеграція UI	3 тижні	22.02.2025 – 14.03.2025
4	Тестування та оптимізація	Функціональне тестування, усунення помилок, адаптація під мобільні пристрої, перевірка безпеки	2 тижні	15.03.2025 – 28.03.2025

Закінчення табл. Б.1

5	Масштабування функціоналу	Додавання модулів: підписки, рейтинги, коментарі, пошук, push-нотифікації, реєстрація авторів	3 тижні	29.03.2025 – 18.04.2025
6	Оптимізація продуктивності	Аналіз навантаження, кешування запитів, оптимізація бази даних, CDN	1 тиждень	19.04.2025 – 25.04.2025
7	Інтеграція аналітики та монетизації	Налаштування Google Analytics, реклама, платні функції, система пожертв	1 тиждень	26.04.2025 – 02.05.2025
8	Документація та навчальні матеріали	Створення технічної документації, FAQ, гідів для користувачів та авторів	1 тиждень	03.05.2025 – 09.05.2025
9	Бета-реліз та зворотний зв’язок	Відкрите тестування для обраних користувачів, збір зворотного зв’язку	1 тиждень	10.05.2025 – 16.05.2025
10	Фінальний реліз і підтримка	Публічний запуск, підтримка першої хвилі користувачів, виправлення критичних помилок	1 тиждень	17.05.2025 – 23.05.2025