МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

НАУМІК СЕРГІЙ ВОЛОДИМИРОВИЧ

ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ФРЕЙМВОРКУ REACT ДЛЯ ПРОЄКТУВАННЯ ВЕБ-САЙТІВ ЗІ ЗБІРКАМИ ПОТОКОВИХ МЕДІАДАНИХ

Спеціальність: 122 «Комп'ютерні науки»
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технологій
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
ПАСТЕРНАК ЯРОСЛАВ МИХАЙЛОВИЧ
Професор кафедри комп'ютерних наук
та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №
засідання кафедри комп'ютерних наук
та кібербезпеки
від _____ 2024 р.
Завідувач кафедри
(_____) Гришанович Т. О.

Луцьк – 2025

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі веброзробки фреймворки відіграють ключову роль, дозволяючи значно пришвидшити процес створення вебдодатків, мінімізувати кількість помилок та забезпечити високу стабільність і масштабованість проєктів. Це особливо важливо для розробки складних систем, таких як вебсайти, які вимагають ефективної роботи з великими обсягами інформації та динамічним інтерфейсом.

React є одним із провідних фреймворків для побудови користувацьких інтерфейсів у світі веброзробки. Він забезпечує високу продуктивність завдяки віртуальному DOM, ефективну розробку компонентів та гнучкість у створенні складних інтерактивних елементів. Його архітектура і підхід до роботи з даними роблять його ідеальним кандидатом для проєктів, що оперують потоковими медіаданими, такими як відео- та аудіохостинги, онлайн-кінотеатри або стрімінгові платформи.

Дослідження використання React дозволить виявити його ключові переваги та потенційні недоліки в контексті цієї специфічної задачі. Також це дасть змогу з'ясувати, які архітектурні рішення та підходи до інтеграції сторонніх сервісів є найбільш ефективними для створення швидких, стабільних та функціональних вебдодатків у цій галузі.

Таким чином, дослідження одного з провідних фреймворків є надзвичайно актуальним у сучасному цифровому просторі, де домінує споживання медіаконтенту. Результати цього дослідження можуть стати цінним внеском у розвиток практик веброзробки та надати рекомендації для створення високопродуктивних і зручних медіа-платформ.

Метою даної роботи є розробка зручного, адаптивного та функціонального веб-сайту для перегляду аніме, що забезпечить ефективне управління контентом, зручну взаємодію з користувачем та відповідатиме сучасним вимогам до онлайн-платформ потокових медіаданих.

Для досягнення визначеної мети потрібно виконати такі **завдання**:

- дослідити сучасні тенденції розвитку платформ для потокового перегляду відеоконтенту;
- проаналізувати сучасні вебтехнології, що використовуються для створення веб-ресурсів із медіаданими, з особливим акцентом на React;
- визначити особливості та вимоги до вебсайтів, орієнтованих на перегляд аніме;
- провести огляд аналогічних вебрішень (аніме-платформ) та проаналізувати їхні переваги та недоліки;
- сформулювати вимоги до фукціональності та дизайну вебсайту;
- обґрунтувати вибір моделі процесу розробки та інструментальних засобів, зокрема фреймворку React;
- спроєктувати структуру сайту та розробити його основні компоненти;
- реалізувати програмний засіб відповідно до визначених вимог, використовуючи React длвя розробки інтерфейсної частини;
- провести тестування та налагодження вебзастосунку для забезпечення його стабільної та коректної роботи.

Об'єкт дослідження – процес створення вебзастосунків для потокового відтворення медіаданих.

Предмет дослідження – розробка вебсайту для перегляду аніме з урахуванням сучасних вимог до онлайн-платформ із медіаконтентом та ефективного використання фреймворку React.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ВЕБСАЙТІВ ДЛЯ ПОТОКОВОГО ПЕРЕГЛЯДУ МЕДІАДАНИХ

1.1. Сучасні тенденції розвитку вебсайтів для перегляду медіаданних

Онлайн-платформи потокового перегляду медіаданих продовжують активно розвиватися, змінюючи традиційні підходи до споживання контенту та взаємодії з користувачами. Основні тенденції, що спостерігаються останніми роками, свідчать про глибоку цифрову трансформацію ринку розваг. Поява нових технологій, зміна споживацької поведінки та зростаючі вимоги до зручності, швидкості доступу до контенту та його якості стимулюють розробників до постійного оновлення своїх вебплатформ. Водночас, висока конкуренція в онлайн-середовищі змушує компанії впроваджувати інноваційні рішення, щоб утримувати увагу користувачів і забезпечувати винятковий досвід споживання контенту.

Однією з ключових тенденцій є персоналізація сервісу. Користувачі очікують, що платформа буде розуміти їхні вподобання, попередній досвід перегляду та пошукові запити, пропонуючи саме той контент (наприклад, аніме), який може їх зацікавити. Для цього розробники активно використовують аналітику, алгоритми машинного навчання та системи рекомендацій, що формують індивідуальні добірки. Компанії, що впроваджують технології штучного інтелекту, скорочують витрати на обслуговування клієнтів до 30%, водночас покращуючи швидкість реагування та рівень задоволеності клієнтів [1]. Персоналізований підхід дозволяє не лише підвищити рівень задоволення користувачів, а й сприяє зростанню їхньої лояльності.

Помітним став розвиток мобільного доступу до **контенту**, адже користувачі дедалі частіше переглядають медіадані за допомогою смартфонів або планшетів. Щоб задовольнити потреби мобільних користувачів, компанії впроваджують прогресивні вебдодатки (PWA**)**. PWA – це вебдодатки, які

забезпечують нативний інтерфейс, подібний до додатків, на мобільних пристроях. Вони поєднують гнучкість вебу з функціональністю та продуктивністю нативних додатків, пропонуючи безперебійний та адаптивний користувацький досвід [2]. Для платформ потокового відео, PWA можуть забезпечувати офлайн-перегляд, push-сповіщення про нові серії та швидкий доступ з головного екрана пристрою.

Значну актуальність для сучасних онлайн-платформ потокового контенту набуває автоматизована взаємодія з користувачами. Широке впровадження чатботів та віртуальних помічників дозволяє ефективно обробляти численні звернення: від надання інформації про наявність аніме та допомоги у виборі за жанром, до інформування про графік виходу нових епізодів і оперативного вирішення типових технічних питань з відтворенням медіаданих. Ця автоматизована підтримка дає можливість сервісам обслуговувати значно більшу аудиторію без зниження стандартів якості. Завдяки передовим інструментам автоматизації, компанії можуть забезпечувати цілодобову доступність допомоги, швидко реагувати на стандартні запити та оптимізувати завантаженість штатного персоналу. Це також гарантує безперервний доступ до необхідної інформації для користувачів, незалежно від часу доби [3].

Ще однією важливою характеристикою сучасних онлайн-платформ є омніканальність. Це підхід, за якого взаємодія з користувачем здійснюється через різні канали – вебсайт, мобільний застосунок, смарт-ТВ додатки, соціальні мережі, месенджери – з єдиною метою: забезпечити безшовний та зручний досвід. Наприклад, користувач може почати перегляд аніме на комп'ютері, продовжити на смартфоні в дорозі, а завершити на Smart TV вдома, при цьому прогрес перегляду синхронізується. Усі ці канали повинні бути інтегровані між собою, щоб користувач не відчував розриву у процесі споживання контенту.

Крім того, зростає попит на прозорість умов доступу (наприклад, підписка), гнучкі способи оплати (якщо це платна платформа), швидке завантаження та зручні інтерфейси. Споживачі очікують, що весь процес буде максимально простим, швидким і передбачуваним. У зв’язку з цим

вебплатформи впроваджують оптимізовані механізми доставки контенту (CDN), інтегрують різноманітні платіжні сервіси та спрощують реєстрацію та авторизацію. Також важливо, щоб користувачі мали можливість знайти та почати перегляд контенту в кілька кліків, без зайвих етапів [4].

Таким чином, розробка онлайн-платформ для потокового перегляду медіаданих сьогодні – це не просто публікація відео в інтернеті, а ціла екосистема, що постійно вдосконалюється та реагує на виклики часу. Ключовими напрямками її розвитку є персоналізація контенту, автоматизація взаємодії, омніканальність та клієнтоорієнтованість. Усі ці фактори формують нову якість взаємодії між постачальником контенту та користувачем, підвищують залученість аудиторії та сприяють формуванню довгострокових відносин.

1.2. Огляд технологій для створення вебсайтів

У сучасну цифрову епоху створення вебсайтів давно вийшло за рамки простого інструменту для представлення компаній чи послуг. Цей процес став ключовим і самостійним напрямком у сфері інформаційних технологій. Успішний вебсайт — це результат гармонійного поєднання інтуїтивно зрозумілого інтерфейсу, естетично привабливого дизайну та ефективно продуманої логіки взаємодії з користувачем. Саме від злагодженого функціонування технічних рішень та дизайнерських концепцій безпосередньо залежить, чи зможе ресурс утримати відвідувача та успішно конвертувати його у цільову дію. Технології веброзробки постійно еволюціонують, відкриваючи нові горизонти для створення ще більш функціональних та адаптивних цифрових рішень. Усі вебтехнології традиційно поділяють на дві основні складові: фронтенд, що відповідає за клієнтську частину та безпосередню взаємодію з користувачем, та бекенд, який забезпечує внутрішню логіку та функціональність на сервері.

Фронтенд охоплює всю видиму для користувача частину вебсайту, що відображається безпосередньо у веббраузері. Ця складова включає текстовий контент, інтерактивні кнопки, графічні зображення, навігаційні меню, а також різноманітні анімації та інші інтерактивні елементи, що забезпечують взаємодію. Серед ключових технологій, що використовуються для розробки фронтенду, виділяють наступні:

1. HTML (HyperText Markup Language) – це мова розмітки гіпертексту, яка використовується для створення структури та макету вебсторінок. Основними компонентами є елементи, теги та атрибути. HTML використовується для побудови структури та розмітки вебсторінок, відображення тексту зображень та мультимедійного контенту [5].
2. CSS (Cascading Style Sheets) – це мова стилів, призначена для візуального оформлення документів, написаних мовами розмітки, зокрема HTML. CSS використовується для стилізації вебсторінок (кольори, шрифти, відступи тощо) та для адаптивного дизайну, що робить коректним відображення на різних пристроях та екранах [6].
3. JavaScript – це мова програмування, яку зазвичай використовують веброзробники для додавання динамічних взаємодій вебсторінок, програм, серверів і навіть ігор. Ця мова ідеально працює разом з HTML та CSS, доповнюючи CSS у форматуванні елементів HTML [7].

У сфері фронтенд-розробки провідні позиції впевнено утримують три ключові фреймворки: Vue, React та Angular. Кожен з них був створений з метою ефективної реалізації зовнішнього функціоналу вебсайтів, відповідаючи за візуальну частину та взаємодію з користувачем. Для наочного ілюстрування їхньої поширеності та затребуваності серед розробників, нижче представлено графік порівняння популярності цих фреймворків (Рис. 1.1).

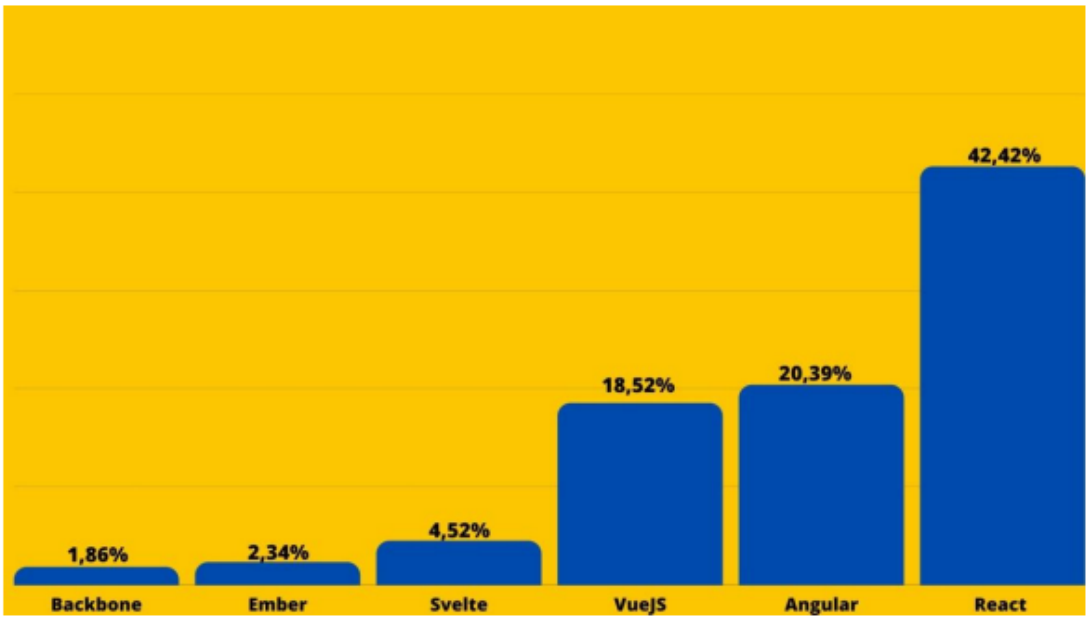

Рисунок 1.1 – Графік порівняння популярності фронтенд фреймворків

Для зіставлення були обрані Vue та React. Обидва фреймворки надають розробникам потужні засоби для створення високопродуктивних і легко масштабованих вебдодатків, ефективно використовуючи компонентний підхід до архітектури та механізм віртуального DOM.

Vue.js вирізняється своєю винятково простою та логічною документацією, що істотно полегшує його освоєння початківцями та прискорює процес розробки порівняно з React. Додатковою зручністю для україномовних розробників є можливість переключення мови документації на українську, що значно оптимізує роботу над вебсайтом, особливо для тих, хто не вільно володіє англійською. Як фронтенд-фреймворк, Vue забезпечує вражаючу швидкість функціонування, зокрема завдяки вбудованій системі реактивності. Це усуває потребу в написанні додаткових скриптів для відстеження змін у стані, на відміну від React. Крім того, базове встановлення Vue вже містить низку корисних плагінів, таких як Vue Router для організації маршрутизації чи Vuex, що слугує централізованим сховищем стану застосунку [8].

Незважаючи на лідерство React за популярністю серед фреймворків, для повноцінної роботи з ним часто виникає необхідність інтеграції додаткових

зовнішніх бібліотек, наприклад, Redux для управління глобальним станом застосунку [9].

Важливо відзначити, що як Vue, так і React ефективно використовують концепцію віртуального DOM, що дозволяє оптимізовано оновлювати лише ті елементи інтерфейсу, які безпосередньо змінилися, забезпечуючи високу продуктивність. Крім того, обидва фреймворки базуються на компонентній моделі розробки, де функціональні блоки коду є незалежними та можуть багаторазово використовуватись у різних частинах вебдодатку.

Отже, кінцевий вибір між Vue.js та React має визначатися сукупністю факторів: специфікою проєкту, поточним рівнем кваліфікації команди розробників та їхніми індивідуальними перевагами. Кожен із цих інструментів має свої унікальні переваги та потенційні недоліки, тому критично важливо ретельно зважити всі ці аспекти перед ухваленням рішення щодо вибору основного фреймворку.

У той час як фронтенд зосереджений на забезпеченні взаємодії з користувачем, бекенд виконує фундаментальні операції: обробку даних, реалізацію бізнес-логіки та налагодження зв'язку між різноманітними компонентами системи. На графіку (рис. 1.2) представлено найпопулярніші фреймворки, які програмісти найчастіше використовують для розробки бекенду станом на січень 2025 року.

Розглянемо одного з лідерів по виористанню розробниками при створеннні бекенду – Nest.js.

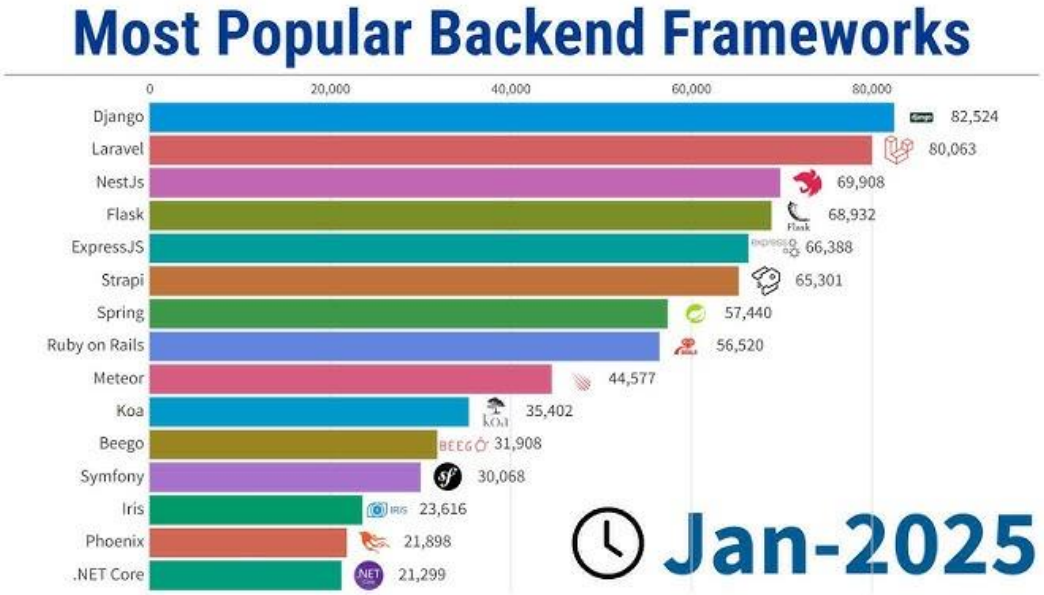

Рис. 1.2 – Графік популярності бекенд фреймворків

1.2.1. Nest.js

Nest.js – це сучасний, прогресивний фреймворк для Node.js, розроблений з метою надати розробникам надійний інструмент для створення масштабованих, легко підтримуваних та високоефективних серверних вебзастосунків. Він базується на архітектурних принципах, що поєднують сучасні підходи та активно використовують TypeScript. Натхнений Angular, Nest.js інтегрує елементи об'єктно-орієнтованого, функціонального та реактивного програмування, що дозволяє створювати надзвичайно стійкі та надійні серверні рішення [10].

Переваги Nest.js [10, 11]:

- завдяки архітектурі, що ґрунтується на модулях, Nest.js значно спрощує структурування коду, розділяючи його на чіткі модулі, сервіси та контролери. Це сприяє кращому супроводу та масштабуванню вебдодатків, особливо у проєктах з великими командами;
- використання TypeScript забезпечує строгу типізацію коду, що надає можливості для автозавершення, статичної перевірки та покращеної інтеграції з інтегрованими середовищами розробки (IDE). Такий підхід

знижує ймовірність помилок на початкових стадіях розробки, підвищує продуктивність команди та спрощує подальше обслуговування кодової бази;

- Nest.js перейняв ключові архітектурні принципи Angular, такі як декоратори, ін'єкція залежностей (DI) та логічна структура проєкту. Ця особливість істотно полегшує адаптацію для фронтенд-розробників або команд, які вже мають досвід роботи з Angular, оскільки архітектура є знайомою та передбачуваною;
- Nest.js надає потужні можливості для інтеграції з GraphQL. Завдяки використанню декораторів та класів, розробники можуть швидко створювати схеми, мінімізуючи обсяги повторюваного коду;
- фреймворк демонструє швидкий прогрес завдяки відкритому коду, активній спільноті розробників та регулярним оновленням. Доступна велика кількість пакетів, генераторів та шаблонів, що спрощує інтеграцію з популярними технологіями та розширює функціональні можливості.

Недоліки Nest.js [10, 11]:

- хоча архітектура **Nest.js** є ретельно продуманою та структурованою, початківці можуть стикнутися з певними труднощами у засвоєнні таких концепцій, як декоратори, DI-контейнери або життєвий цикл компонентів;
- для невеликих проєктів Nest.js іноді може виявитися занадто "об'ємним", його розгалужена модульна структура, велика кількість файлів та класів можуть ускладнити процес розробки, не надаючи при цьому суттєвих переваг у порівнянні з легшими та більш мінімалістичними фреймворками;
- складність інтеграції зі застарілими системами: Оскільки Nest.js розроблений з фокусом на сучасні технології (TypeScript, модульність), його інтеграція у застарілі проєкти, написані на чистому JavaScript, може бути досить складною. Це часто вимагає повного перероблення архітектури, що не завжди є практичним чи доцільним.

Підсумовуючи, Nest.js – це потужний фреймворк, що пропонує передовий підхід до створення серверних вебзастосунків на базі Node.js. Він успішно поєднує в собі переваги об'єктно-орієнтованого програмування, реактивності та строгої типізації. Його архітектура ідеально підходить для розробки масштабних проєктів, мікросервісів, монорепозиторіїв та корпоративного програмного забезпечення. Однак застосування Nest.js є найбільш обґрунтованим, коли команда розробників вже має досвід роботи з TypeScript або Angular, або ж існує чітка потреба у створенні добре структурованого та масштабованого проєкту. У випадку ж простих сервісів чи обмежених ресурсів, доцільніше розглянути легші альтернативи. Важливо зазначити, що Nest.js активно використовується такими відомими компаніями, як Adidas, BMW, IBM, JetBrains, GitLab та Mercedes-Benz [12].

1.3. Особливості розробки вебсайтів для потокового перегляду медіаданих

Створення вебсайтів, призначених для потокової трансляції медіаданих, зокрема аніме, вимагає врахування ряду специфічних умов та нюансів. Це зумовлено необхідністю гарантувати користувачам легкий доступ до контенту, забезпечити високу продуктивність системи, її надійність, належний рівень безпеки, а також ефективну обробку значних обсягів інформації. Такий вебсайт покликаний не лише розважати та інформувати, а й активно залучати аудиторію, стимулюючи регулярне використання платформи. Детальніше розглянемо ключові функціональні можливості, що є невід'ємними для сучасних веб-ресурсів, орієнтованих на потокове відтворення медіа.

Каталог медіаконтенту виступає центральним елементом будь-якої веб-платформи для перегляду контенту, оскільки саме через нього користувачі здійснюють основну навігацію по доступній бібліотеці. Якісний каталог не повинен мати обмежень щодо кількості категорій чи жанрів. Він має бути архітектурно гнучким, дозволяючи обробляти як невеликі колекції, так і

масштабні бібліотеки, залежно від можливостей серверної інфраструктури. Важливою складовою є наявність деталізованих характеристик контенту, що дає змогу налаштовувати гнучкі фільтри за жанром, роком випуску, студією-виробником, рейтингом, статусом перегляду та іншими критеріями. Це значно спрощує пошук бажаного контенту, мінімізуючи кількість кроків. Система каталогу мусить підтримувати динамічне відображення пов'язаних матеріалів (наприклад, різних сезонів, OVA, повнометражних фільмів з одного всесвіту). Інтерфейс каталогу повинен інтегрувати функції попереднього перегляду (як-от трейлери, короткі синопсиси), можливість додавання до списків "обраного" або "до перегляду", швидке застосування фільтрів без необхідності перезавантаження сторінки, а також відображення актуальної інформації про статус виходу епізодів.

Суттєвою складовою вебсайту для потокового перегляду контенту є система управління списками перегляду та персоналізованими колекціями. Вона надає користувачам можливість зберігати аніме для подальшого перегляду, відстежувати прогрес перегляду серій та формувати власні тематичні добірки. Сучасні вебсайти мають реалізовувати інтерактивні списки, що дозволяють користувачу легко додавати або вилучати позиції, відмічати вже переглянуті епізоди, а також забезпечувати синхронізацію прогресу між різними пристроями. Функціонал безпосереднього перегляду має бути максимально зручним: це охоплює адаптивний відеопрогравач, можливість зміни якості відтворення, вибір аудіодоріжок та субтитрів, а також функцію автоматичного переходу до наступного епізоду.

Особливу увагу варто приділити адміністративній панелі вебсайту, яка забезпечує всебічне керування медіаконтентом, даними користувачів та загальними системними налаштуваннями. За допомогою цієї панелі адміністратор може додавати нові аніме-серіали до каталогу, оновлювати описи та метадані, завантажувати нові епізоди, керувати обліковими записами користувачів та здійснювати модерацію коментарів. Система адміністрування повинна бути ергономічною, інтуїтивно зрозумілою та безпечною, з чітким

розмежуванням доступу для сторонніх осіб. Безпека є ще одним фундаментальним аспектом, оскільки вебсайт обробляє особисті дані користувачів та конфіденційну інформацію. Необхідно впровадити захищене з'єднання через HTTPS, використовувати надійні механізми авторизації та автентифікації, а також забезпечити ефективний захист від типових веб-атак.

Крім того, важливою особливістю є адаптивний дизайн. Зважаючи на те, що значна частина аудиторії отримує доступ до потокового контенту з мобільних пристроїв, інтерфейс має бездоганно відображатися на смартфонах, планшетах та інших екранах. Елементи повинні автоматично адаптуватися під різні розміри дисплеїв, забезпечуючи оптимальний досвід перегляду незалежно від пристрою. Оптимізація швидкодії вебсайту є критично важливою, оскільки повільне завантаження сторінок або буферизація відео негативно позначаються на користувацькому досвіді та можуть призвести до втрати аудиторії. Для залучення нових користувачів та аналізу ефективності функціонування вебсайту доцільно інтегрувати аналітичні інструменти (наприклад, Google Analytics). Вони дозволяють відстежувати поведінку користувачів, джерела трафіку, найбільш популярні аніме та інші метрики, що є надзвичайно цінними для подальшого розвитку платформи.

Таким чином, розробка вебсайту для потокового перегляду медіаданих виходить за рамки простої реалізації базового функціоналу відтворення. Вона передбачає створення надійної, масштабованої, захищеної системи, яка враховує сучасні вимоги до дизайну, продуктивності, безпеки та аналітичних можливостей. Успішний вебсайт у цій ніші має забезпечувати інтуїтивно зрозумілу взаємодію з користувачем, ефективну обробку великих обсягів контенту та бути готовим до постійного зростання та вдосконалення.

1.4. Огляд та аналіз аналогічних розробок

Для створення високоякісного та функціонального вебсайту для потокового перегляду відеоконтенту надзвичайно важливо ретельно вивчити

досвід вже існуючих програмних рішень аналогічного призначення. З цією метою було здійснено аналіз двох обраних онлайн-платформ. Дослідження кожної з них охоплює ключові характеристики, реалізовані функціональні можливості, а також їхні сильні та слабкі сторони. Такий методологічний підхід дає змогу ідентифікувати успішні рішення, які варто застосувати у власному проєкті, а також виявити недоліки, яких слід уникнути під час розробки власного вебресурсу.

На сьогоднішній день ринок потокового аніме представлений як великими глобальними гравцями, так і спеціалізованими платформами, орієнтованими виключно на аніме. До найбільш значущих можна віднести Crunchyroll та Netflix (у контексті їхнього аніме-розділу).

1.4.1 Crunchyroll

Crunchyroll є однією з провідних світових платформ, що спеціалізуються виключно на аніме, манзі та японській драмі (рис. 1.3).

Для створення цього сайту використані такі технології, як мови програмування Node.js для бекенду, для бази даних MongoDB та PostgreSQL.

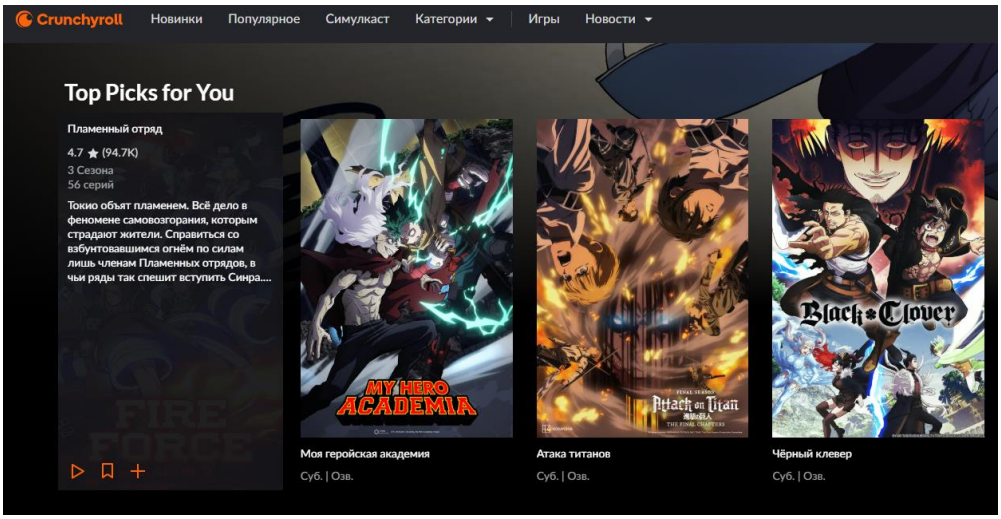

Рис. 1.3 – Головна сторінка сайту «Crunchyroll»

Переваги:

- платформа пропонує одну з найбільших бібліотек аніме, включаючи як класичні твори, так і найновіші релізи;
- нові епізоди часто доступні за лічені години після їхнього виходу;
- орієнтованість виключно на аніме дозволяє глибше опрацьовувати контент, надавати деталізовані метадані;
- наявність субтитрів багатьма мовами, а також постійне розширення бібліотеки з професійним дубляжем;
- інтегровані функції для взаємодії користувачів, такі як коментарі до серій, форуми;
- забезпечує коректне відображення та функціональність на різних пристроях.

Недоліки:

- деякі користувачі відзначають, що якість відео не завжди досягає рівня 4K, обмеження у 1080p є поширеним явищем для значної частини контенту;
- хоча вебінтерфейс є функціональним, його візуальна складова та користувацький досвід не завжди відповідають найсучаснішим стандартам, іноді здаючись дещо застарілим;
- наявність реклами для користувачів без преміум-підписки може бути відволікаючою.

1.4.2 Netflix

Netflix — глобальний лідер серед потокових сервісів, який активно інвестує у розширення своєї аніме-бібліотеки, включаючи ліцензування існуючих та створення оригінальних аніме-проєктів (рис. 1.4).

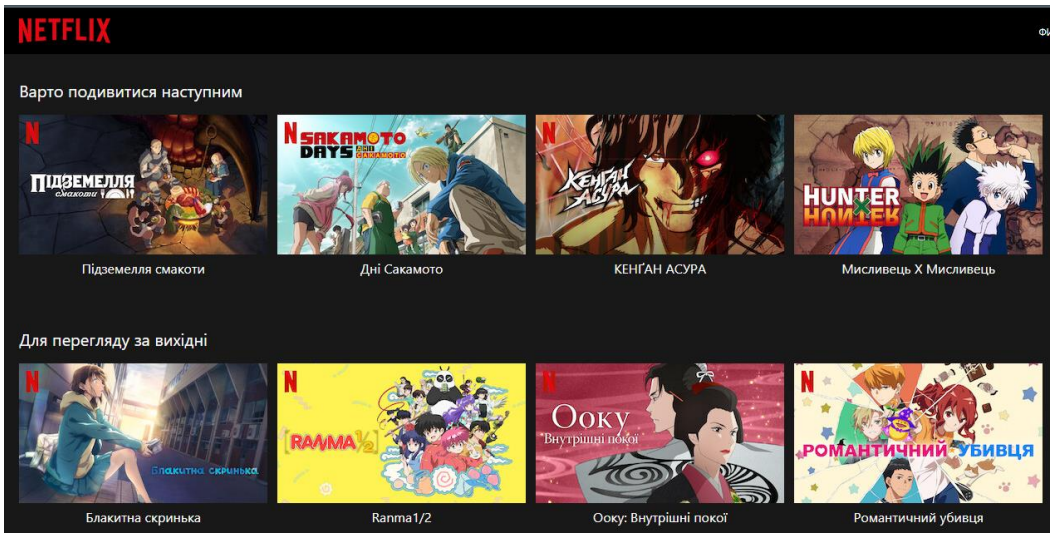

Рис. 1.4 – Головна сторінка сайту «Netflix»

Переваги:

- часто пропонує контент у 4K Ultra HD з підтримкою HDR, що забезпечує відмінну візуальну якість;
- широкий вибір мовних доріжок та субтитрів, включаючи український дубляж або субтитри для значної частини аніме;
- інтуїтивно зрозумілий веб-інтерфейс, ефективні алгоритми рекомендацій та плавна робота плеєра;
- активне виробництво ексклюзивних аніме-серіалів та фільмів;
- доступність на величезній кількості пристроїв (Smart TV, консолі, мобільні пристрої, веб-браузери) із синхронізацією прогресу перегляду.

Недоліки:

- хоча бібліотека аніме зростає, вона все ще не така обширна, як у спеціалізованих платформ типу Crunchyroll, і може не включати деякі нішеві або старіші тайтли;
- нові епізоди аніме, які не є оригінальними проєктами Netflix, з'являються з певною затримкою;

- через широкий профіль, Netflix не пропонує таких глибоких фанатських функцій (наприклад, коментарі до конкретних епізодів або детальні метадані про сейю), як спеціалізовані платформи.

Аналіз двох аналогічних програмних рішень дозволяє зробити висновок, що більшість сучасних вебсайтів для потокового перегляду аніме мають подібну базову структуру: деталізований каталог, сторінки з описом окремих аніме (включно з серіями), функціонал для управління списками перегляду. Найбільш насиченими функціоналом виявилися такі платформи, як Crunchyroll та Netflix, які пропонують персоналізовані рекомендації, розширені фільтри та зручні системи управління профілями користувачів. Натомість, деякі менш складні ресурси можуть слугувати прикладом спрощеної реалізації з мінімальним функціоналом, що може бути корисним для початкових етапів розробки. Для створення власного вебсайту доцільно врахувати такі ключові аспекти: забезпечення простої та інтуїтивно зрозумілої навігації, реалізація ефективної системи фільтрації контенту, включення розділу з новинками, а також передбачення зручного функціоналу для відстеження прогресу перегляду та формування персоналізованих списків.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБСАЙТУ ЗІ ЗБІРКАМИ ПОТОКОВИХ МЕДІАДАННИХ

2.1. Постановка задачі, призначення та вимоги до програмного засобу

Ключовим завданням поточного проєкту є розробка вебсайту, призначеного для потокового перегляду аніме. Цей ресурс покликаний забезпечити користувачам інтуїтивно зрозумілий та безпечний процес пошуку, вибору й відтворення бажаного контенту. Вебсайт має запропонувати ефективний інтерфейс, що дозволить швидко знаходити аніме-серіали чи фільми, додавати їх до персональних списків перегляду та безперешкодно розпочинати відтворення. Важливим аспектом, що забезпечить доступність та зручність використання платформи для широкої аудиторії, є реалізація адаптивного дизайну. Це означає, що вебсайт повинен коректно та естетично відображатися на будь-яких пристроях – від великих моніторів до планшетів і смартфонів, автоматично підлаштовуючи свої елементи під різні розміри екранів.

Основне призначення цієї розробки полягає у створенні такого вебсайту, який стане зручним інструментом для шанувальників аніме. Платформа має гарантувати комфортний процес ознайомлення з доступним асортиментом, вибору та безперешкодного перегляду улюблених серіалів чи фільмів, усуваючи необхідність пошуку контенту на різноманітних, часто розрізнених, ресурсах.

Функціональні вимоги до програмного засобу:

- управління доступом користувача;
- перегляд медіаконтенту;
- навігація та пошук контенту;
- вибір випадкового аніме;
- перехід на окрему сторінку конкретного аніме;
- зміна теми сайту на темну;
- адаптація під різне розширення.

2.2. Вибір моделі розробки програмного засобу

У рамках даного проєкту для створення вебзастосунку була застосована інкрементна модель життєвого циклу програмного забезпечення. Ця методологія передбачає послідовне формування системи через невеликі, керовані сегменти, або ж інкременти. Кожен такий інкремент втілює певну функціональну частину системи, а після завершення відповідної фази розробки отримуємо робочу версію продукту [13]. При розробці вебсайту для потокового перегляду аніме було реалізовано кілька ключових ітерацій, кожна з яких охоплювала окремий блок функціоналу.

Першим кроком у розробці стало створення детального дизайну в Figma. На цьому етапі було ретельно візуалізовано кожний аспект майбутнього вебсайту. Це включало розробку макетів для головної сторінки, що задає тон усьому ресурсу, а також продуману структуру навігаційного меню для інтуїтивно зрозумілого переміщення по сайту. В рамках дизайн-процесу також було визначено стилістичні рішення: палітру кольорів, типографіку, розташування елементів інтерфейсу, що забезпечить привабливий та єдиний візуальний стиль вебресурсу**.**

Наступним етапом, після завершення дизайну, стало концептуальне проєктування структури вебсайту. На цій фазі було сформовано базовий каркас вебресурсу, що включав архітектуру головної сторінки, системи навігації, а також окремих сторінок для кожного аніме. Водночас відбувалася розробка оптимальної файлової структури проєкту та остаточний вибір технологічного стеку, що охоплює основні мови програмування, необхідні бібліотеки та середовище розробки. Головною метою цього етапу було закладення міцного фундаменту, на якому в подальшому будуть надбудовуватися всі функціональні складові вебзастосунку.

Третьою ітерацією у процесі розробки стало впровадження темної теми оформлення, широко відомої як чорна тема. Це рішення було обумовлене не лише прагненням забезпечити значне зниження навантаження на зір користувачів під

час тривалого перегляду контенту, що особливо актуально для вебсайтів із потоковими медіаданими. Окрім ергономічних переваг, темна тема також істотно підкреслює естетичну привабливість вебсайту, надаючи йому сучасний та стильний вигляд, що є особливо виразним в умовах низького освітлення або у вечірній час.

Четверта ітерація – створення слайдера. Цей інтерактивний елемент дозволяє ефективно демонструвати рекомендовані аніме, новинки або популярні тайтли на головній сторінці вебсайту. Слайдер забезпечує динамічну подачу візуального контенту, привертаючи увагу користувачів та спонукаючи їх до взаємодії з платформою.

Наступним кроком стала реалізація кнопки "Random". Цей функціональний елемент дозволяє користувачам миттєво переходити на сторінку випадково обраного аніме з бібліотеки вебсайту. Таке рішення додає елемент несподіванки та допомагає користувачам відкривати для себе новий контент, коли вони не впевнені у своєму виборі.

Фінальним етапом реалізації стала ітерація, спрямована на візуальну досконалість вебсайту та всебічне тестування його функціональних можливостей. У процесі цього етапу була перевірена бездоганність роботи всіх реалізованих функцій, а також оперативно усунуті виявлені недоліки. Особлива увага зосереджувалася на підвищенні зручності інтерфейсу, забезпеченні легкості сприйняття текстового контенту, логічності розміщення елементів та загальній продуктивності вебсайту, що має гарантувати користувачам безперебійний та комфортний досвід взаємодії.

Таким чином, використання інкрементної моделі виявилося оптимальним для розробки повноцінного вебзастосунку. Вона дозволила послідовно інтегрувати ключові можливості, забезпечити гнучкість процесу розробки та своєчасно виявляти й ефективно усувати будь-які недоліки. Кожен з етапів був логічно завершеним і гармонійно поєднувався із загальним баченням функціоналу вебсайту, що створює надійну основу для його легкого розширення та подальшого розвитку в майбутньому.

2.3. Загальний опис проєкту

Розроблене програмне забезпечення є вебзастосунком, орієнтованим на потоковий перегляд аніме. Архітектура вебсайту ґрунтується на перевіреній клієнт-серверній моделі. Взаємодія між клієнтською частиною та сервером забезпечує оперативну обробку запитів користувачів, відображення актуального контенту та підтримання стабільної роботи всієї системи. Цей підхід дозволяє створити зручний, надійний та ефективний сервіс для доступу до колекції аніме.

У системі передбачений один основний тип користувачів – відвідувачі платформи. Діаграма використання вебсайту, що представлена на рисунку 2.1, ілюструє ключові функціональні можливості, доступні для цієї категорії користувачів.

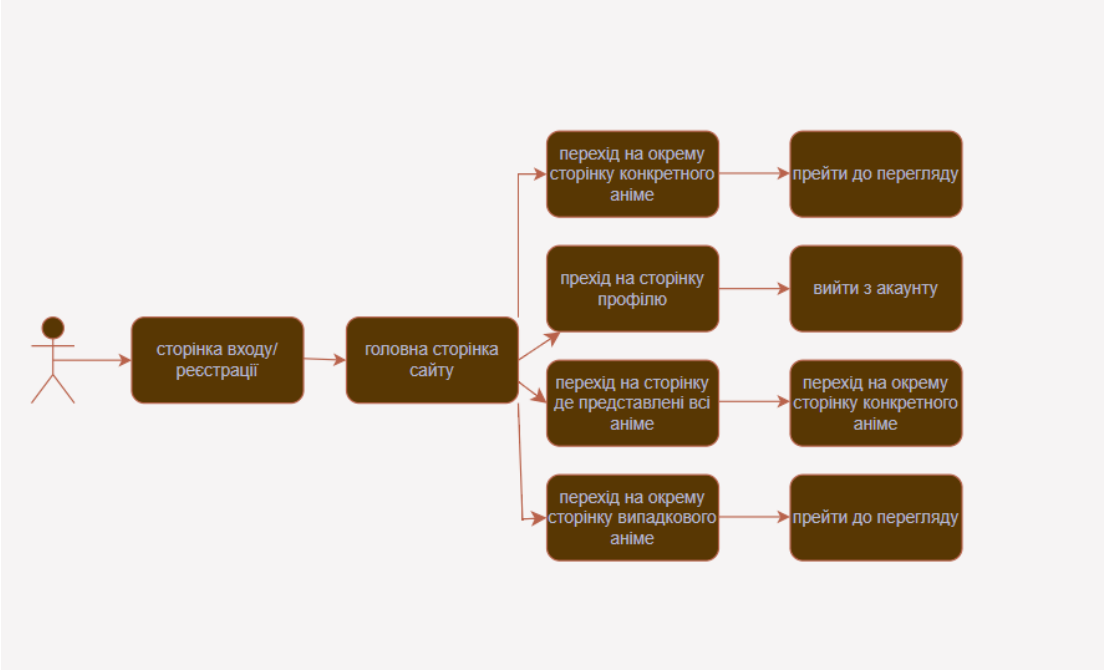

Рис. 2.1 – Діаграма використання сайту

З боку користувача вебсайт надає низку ключових можливостей, починаючи з сторінки входу/реєстрації, що забезпечує авторизований доступ.

Після успішного входу користувач потрапляє на головну сторінку сайту, звідки може здійснювати різні функціональні дії.

Серед основних сценаріїв взаємодії виділяються:

- перехід на окрему сторінку конкретного аніме – для детального ознайомлення з обраним тайтлом та подальшого переходу до його перегляду;
- доступ до сторінки профілю – де користувач може керувати своїми даними та, за потреби, вийти з акаунту;
- навігація на сторінку, що містить повний перелік усіх аніме, з можливістю подальшого переходу до окремої сторінки конкретного аніме;
- вибір випадкового аніме за допомогою спеціальної функції, яка перекидає користувача на сторінку випадково обраного тайтлу для його подальшого перегляду.

У даному розділі наведено детальний опис реалізації ключових компонентів та функціональних можливостей розробленого вебзастосунку для потокового перегляду аніме. Зокрема, розглянуто структуру вебсайту**,** взаємодію між його клієнтською частиною, а також логіку роботи основних елементів, що забезпечують зручний доступ до контенту та інтерактивну взаємодію з користувачем. Також висвітлюються принципи організації даних, що лежать в основі відображення аніме, та підходи до забезпечення стабільності функціонування системи.

2.4. Обґрунтування вибору інструментальних засобів розробки

2.4.1. Середовище розробки VS Code

Visual Studio Code (VS Code) — це надзвичайно потужний та водночас легкий редактор коду з розширеними функціональними можливостями інтегрованого середовища розробки (IDE), що активно застосовується для створення вебзастосунків різної складності [14]. Вибір саме цього середовища

розробки зумовлений його винятковою гнучкістю, кросплатформністю, високою продуктивністю та величезною екосистемою розширень, що робить його оптимальним вибором для сучасних вебпроєктів, зокрема тих, що базуються на JavaScript/TypeScript та фреймворку React.

Основні функціональні переваги VS Code [15]:

- VS Code надає передове автодоповнення коду, що охоплює змінні, функції, методи, класи та модулі. Ця функція адаптується до контексту проєкту, особливо ефективно працюючи з TypeScript, значно прискорюючи процес написання коду;
- середовище оперативно виявляє потенційні помилки та синтаксичні неточності безпосередньо під час написання коду. Це забезпечується завдяки вбудованим механізмам та інтеграції з лінтерами і компілятором TypeScript, що дозволяє усувати проблеми на ранніх стадіях розробки;
- VS Code оснащений потужним вбудованим налагоджувачем. Він дозволяє встановлювати точки зупину, відстежувати значення змінних у режимі реального часу та покроково виконувати код, що є незамінним для ефективного пошуку та виправлення помилок;
- завдяки широкому спектру доступних розширень, VS Code підтримує популярні фреймворки для тестування (наприклад, Jest, React Testing Library). Це дозволяє запускати, переглядати та аналізувати результати тестів безпосередньо у середовищі розробки, інтегруючи тестування у робочий процес;
- VS Code надає надійні інструменти для автоматизованого рефакторингу, включаючи безпечне перейменування, витягування функцій та інші операції. Це допомагає оптимізувати структуру коду, підвищити його читабельність та підтримуваність без ризику порушення логіки роботи;
- хоча VS Code не має власної вбудованої СУБД, за допомогою відповідних розширень він може інтегруватися з різними системами управління базами даних. Це дозволяє розробникам переглядати їхню структуру, виконувати

SQL-запити та взаємодіяти з даними, що розширює його можливості для повностекової розробки;

- вбудована підтримка HTML, CSS, JavaScript, а також відмінна робота з TypeScript та фреймворками на кшталт React для фронтенду та Node.js для бекенду, дозволяє ефективно розробляти обидві частини вебпроєктів в одному уніфікованому середовищі.

Таким чином, Visual Studio Code є оптимальним вибором для розробки сучасних вебзастосунків, забезпечуючи всі необхідні інструменти для високоефективної та комфортної роботи.

2.4.2. Середовище розробки дизайну Figma

Figma — це інноваційне хмарне інтегроване середовище, спеціалізоване на проєктуванні користувацьких інтерфейсів (UI) та користувацького досвіду (UX), а також на створенні прототипів вебзастосунків та мобільних програм [16]. Вибір цього інструменту для розробки дизайну вебсайту для потокового перегляду аніме зумовлений його унікальними можливостями для спільної роботи, високою доступністю, потужним набором інструментів для візуалізації та функціоналом прототипування, що є критично важливим для сучасного вебдизайну.

Основні функціональні переваги Figma [17]:

- Figma дозволяє кільком дизайнерам одночасно працювати над одним проєктом, бачачи зміни в реальному часі. Це значно оптимізує командну взаємодію та прискорює ітерації, дозволяючи швидко отримувати зворотний зв'язок;
- будучи хмарним рішенням, Figma не потребує встановлення додаткового програмного забезпечення та доступна безпосередньо через веббраузер. Це забезпечує кросплатформність та дозволяє працювати над проєктом з будь-якого пристрою з доступом до інтернету;

- середовище оснащене широким спектром інструментів для векторної графіки, макетування, роботи з текстом та зображеннями. Особливо цінними є функції Auto Layout, які автоматично підлаштовують елементи під зміни контенту, та Variant, що дозволяє створювати різні стани одного компонента.
- Figma надає можливість створювати інтерактивні прототипи без необхідності використання сторонніх програм. Це дозволяє симулювати взаємодію користувача з майбутнім вебсайтом, перевіряти логіку переходів та анімацій ще до етапу розробки.
- інструмент підтримує створення багаторазових компонентів (кнопок, полів введення, іконок) та їх організацію в бібліотеки. Це забезпечує єдність дизайну, прискорює роботу та спрощує внесення глобальних змін до проєкту;
- генерує фрагменти CSS, Swift або XML коду для розробників, що значно спрощує перетворення дизайну на робочий веб-інтерфейс. Розробники можуть легко переглядати специфікації, розміри, кольори та інші параметри елементів безпосередньо у Figma;
- велика екосистема плагінів розширює функціональність Figma, дозволяючи автоматизувати рутинні завдання, інтегруватися з іншими сервісами та додавати нові можливості для дизайну та прототипування.

Таким чином, Figma є ідеальним вибором для проєктування веб-інтерфейсу аніме-платформи, забезпечуючи ефективну спільну роботу, швидке прототипування та зручну передачу дизайну для подальшої розробки.

2.4.3. Клієнтська частина

Клієнтська частина (фронтенд) вебзастосунку відповідає за візуальне відображення інтерфейсу користувача та забезпечення інтуїтивної взаємодії з функціоналом вебсайту. Саме ця складова є видимою для користувача під час його взаємодії з платформою. У межах даного проєкту клієнтська частина реалізована за допомогою фреймворку React, який використовує та розширює

можливості HTML, CSS та JavaScript для побудови динамічних та інтерактивних веб-інтерфейсів.

React (або React.js) — це декларативна, компонентно-орієнтована JavaScript-бібліотека з відкритим кодом, спеціально розроблена для ефективної побудови інтерфейсів користувача [18]. Вона є основою для розробки високопродуктивних та масштабованих односторінкових вебзастосунків (SPA), де весь контент завантажується лише один раз, а подальші зміни відбуваються динамічно, без потреби у перезавантаженні сторінки, забезпечуючи безперервний користувацький досвід.

Роль JSX

У контексті React, традиційний HTML набуває нового вигляду через використання JSX (JavaScript XML). JSX – це синтаксичне розширення JavaScript, яке дозволяє розробникам писати HTML-подібну розмітку безпосередньо у файлах JavaScript. Хоча це не є чистим HTML, JSX компілюється у виклики JavaScript-функцій, які маніпулюють DOM. Такий підхід дозволяє органічно поєднувати логіку інтерфейсу та його структуру в одному місці, що значно підвищує читабельність коду, спрощує розробку та підтримуваність компонентів [19].

Підходи до стилізації

Інтеграція CSS у React-проєкти реалізується різними підходами, кожен з яких сприяє модульності та інкапсуляції стилів, що є ключовим для компонентної архітектури React. Замість глобальних таблиць стилів, які можуть спричиняти конфлікти, у React часто використовуються:

- CSS Modules – підхід, при якому файли CSS імпортуються як об'єкти JavaScript, а всі імена класів автоматично перетворюються на унікальні, локальні імена. Це гарантує, що стилі, визначені для одного компонента, не впливатимуть на інші, запобігаючи небажаним колізіям;
- CSS-in-JS бібліотеки, зокрема, такі інструменти, як Styled Components або

Emotion, дозволяють писати CSS-правила безпосередньо в JavaScript-коді компонентів. Це дає можливість використовувати динамічні властивості JavaScript для стилізації та забезпечує повну ізоляцію стилів на рівні компонента [20];

- традиційні CSS або Sass, хоч і менш інкапсульовані, все ще використовуються, часто у поєднанні з методологіями іменування (наприклад, BEM) або препроцесорами для кращої організації коду.

Ці методи стилізації у React дозволяють створювати ізольовані, гнучкі та легко підтримувані візуальні компоненти, що є критично важливим для комплексних вебзастосунків.

Основа інтерактивності

React повністю побудований на JavaScript і активно використовує його можливості для створення динамічних та інтерактивних веб-інтерфейсів. JavaScript у React є основою для [21]:

- завдяки JavaScript розробляються функціональні або класові компоненти, які визначають поведінку та відображення елементів інтерфейсу;
- React надає механізми (такі як useState, useReducer) для ефективного управління станом компонентів, дозволяючи інтерфейсу динамічно реагувати на зміни даних;
- за допомогою хуків (наприклад, useEffect) розробники можуть контролювати поведінку компонентів на різних етапах їхнього життєвого циклу – від монтування до оновлення та розмонтування [22];
- JavaScript дозволяє обробляти події користувача (кліки, введення, наведення курсору), реалізуючи інтерактивну поведінку веб-елементів;
- React-застосунки часто використовують JavaScript для виконання асинхронних запитів до серверу (наприклад, за допомогою fetch або Axios) для отримання даних без перезавантаження сторінки [23].

2.5. Особливості програмної реалізації та основні режими роботи

Програмна реалізація вебзастосунку для потокового перегляду аніме зосереджена на клієнтській частині, використовуючи сучасні технології для побудови динамічного та інтерактивного веб-інтерфейсу. Проєкт ініційовано за допомогою інструменту Vite, який забезпечує швидкий старт та оптимізовану розробку для React-застосунків. Структура проєкту чітко організована, поділяючи функціонал на логічні директорії та файли, що сприяє його масштабованості та легкості супроводу (рис. 2.2).

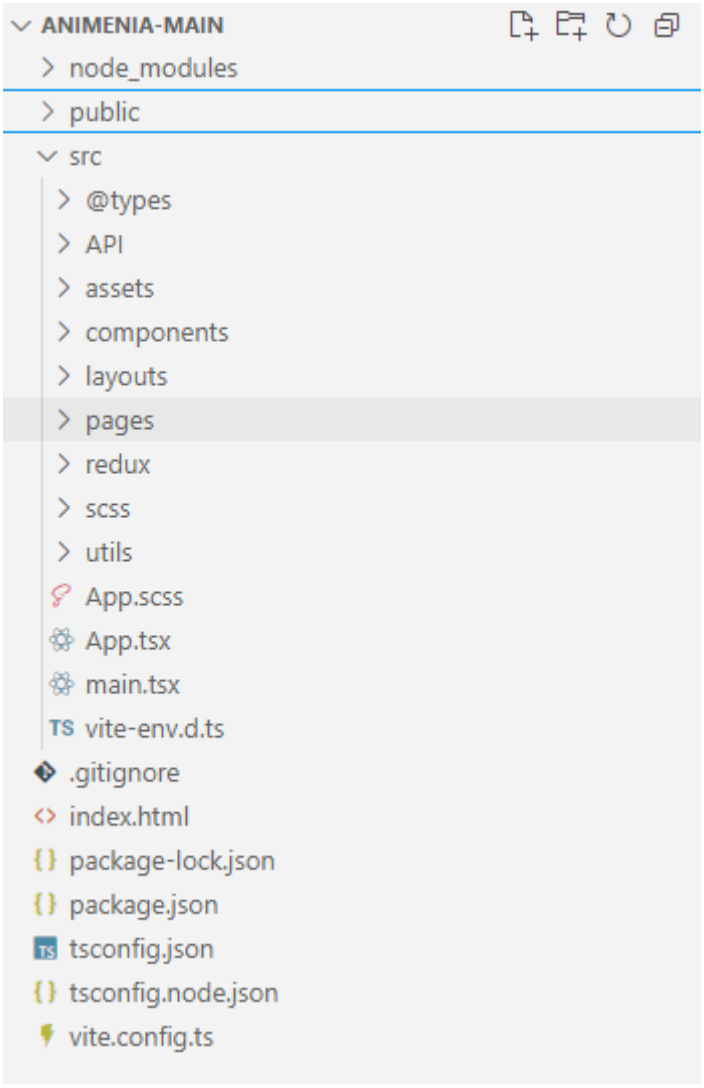

Рис. 2.2 – Структура проєкту

Короткий опис основних файлів і директорій та їхнє призначення:

- node_modules/ – директорія, що містить усі встановлені сторонні бібліотеки та залежності, необхідні для роботи проєкту.

- public/ – каталог для статичних файлів, що не піддаються обробці збіркою (наприклад, index.html, глобальні іконки, файли маніфесту).
- src/ – основний робочий каталог, де знаходиться весь вихідний код вебзастосунку.
- API/ – містить логіку для взаємодії із зовнішніми веб-API (формування та виконання запитів на отримання даних про аніме, функції автентифікації тощо).
- assets/ – містить статичні асети проєкту, такі як зображення, шрифти, SVG-іконки, що використовуються в компонентах.
- components/ – директорія для багаторазових, незалежних від конкретної сторінки UI-компонентів (наприклад, кнопки, картки аніме, елементи навігації, слайдери, компоненти веб-плеєра).
- layouts/ – містить компоненти, що визначають загальну розмітку сторінок (наприклад, структуру заголовків, футерів та бічних панелей для різних типів сторінок).
- pages/ – містить компоненти, які представляють окремі сторінки вебсайту (наприклад, Головна сторінка, Каталог аніме, Детальна сторінка аніме, Профіль, Сторінка входу/реєстрації).
- redux/ – директорія, призначена для управління глобальним станом вебзастосунку за допомогою бібліотеки Redux (включає actions, reducers, store).
- scss/ – містить файли стилів, написані з використанням препроцесора Sass (SCSS), що дозволяє застосовувати змінні, вкладені правила та міксини для ефективної стилізації.
- utils/ – директорія для допоміжних функцій та утиліт, які використовуються в різних частинах застосунку.
- App.tsx – основний компонент застосунку, що агрегує інші компоненти та задає загальну структуру веб-інтерфейсу.
- main.tsx – точка входу в застосунок, де React монтує кореневий компонент в index.html.

- index.html – головний HTML-файл, який є точкою входу для вебзастосунку, куди завантажується та монтується React-додаток.

Важливо зазначити, що у межах цього проєкту не передбачено власної постійної бази даних та адміністративної панелі для управління контентом. Взаємодія з даними (наприклад, інформацією про аніме) відбувається через зовнішні API-сервіси або ж на основі фіксованих (мокованих) даних, інтегрованих безпосередньо у фронтенд. Це означає, що посилання на аніме прописуються безпосередньо в коді сайту. Додавання, зміна або видалення інформації про аніме, яка відображатиметься на вебсторінці, відбувається шляхом модифікації коду сторінки. Такий підхід зумовлений фокусом проєкту на клієнтській частині та вебінтерфейсі.

Розширені аспекти програмної реалізації.

Для забезпечення високої продуктивності, інтерактивності та зручності користувача, програмна реалізація вебзастосунку ґрунтується на кількох ключових принципах та технологічних рішеннях:

- маршрутизація у клієнтській частині (Frontend Routing) – оскільки застосунок є односторінковим (SPA), навігація між різними розділами вебсайту відбувається без повного перезавантаження сторінки. Це реалізовано за допомогою бібліотеки React Router, яка дозволяє зіставляти URL-адреси з відповідними React-компонентами. Такий підхід забезпечує швидкі переходи та плавну взаємодію, імітуючи роботу традиційного багатосторінкового вебсайту;
- управління станом застосунку (State Management) – для ефективного управління глобальним станом застосунку, таким як статус автентифікації користувача, дані про аніме, налаштування фільтрів тощо, використовується бібліотека Redux. Вона надає централізоване сховище стану та передбачуваний потік даних, що спрощує налагодження та підтримку складних веб-інтерфейсів;
- взаємодія з даними та зовнішніми API – отримання даних про аніме, а також обробка автентифікаційних запитів, здійснюється шляхом взаємодії

із зовнішніми API-сервісами. Це реалізується за допомогою вбудованого Fetch API JavaScript або бібліотеки Axios для виконання асинхронних HTTP-запитів. Завдяки цьому, вебзастосунок динамічно завантажує та оновлює контент без необхідності перезавантаження сторінки;

- компонентна архітектура – дизайн та розробка вебзастосунку повністю відповідає компонентній філософії React. Кожен елемент інтерфейсу (від кнопки до цілої сторінки) розглядається як окремий, незалежний компонент. Це сприяє високій модульності, можливості багаторазового використання коду та значно спрощує процес розробки, тестування та подальшої підтримки;
- підходи до стилізації за допомогою SCSS – для організації та застосування стилів використовується препроцесор SCSS (Sass). Це дозволяє писати більш структурований та підтримуваний CSS-код за рахунок використання змінних, вкладеності правил, міксинів та функцій. Такий підхід забезпечує гнучке керування візуальним оформленням компонентів та полегшує створення єдиного дизайн-системи вебсайту.
- автентифікація та авторизація (користувацька сесія) – реалізація автентифікації користувачів здійснюється шляхом взаємодії із зовнішнім сервісом (наприклад, через API). Після успішної авторизації, вебзастосунок отримує токени (наприклад, JWT), які зберігаються у локальному сховищі браузера (localStorage або sessionStorage). Ці токени використовуються для підтвердження ідентичності користувача при подальших запитах до захищених ресурсів, забезпечуючи безпечну та персоналізовану сесію.

Основні режими роботи та функціональні сценарії

Основними режимами роботи вебзастосунку є ключові сценарії взаємодії користувача з платформою для перегляду аніме, які детально відображені у діаграмі варіантів використання (рис. 2.1). Ці режими забезпечуються вищеописаними програмними рішеннями та включають:

- перегляд контенту – доступ до головної сторінки з основними рекомендаціями та динамічним слайдером, а також детальних сторінок конкретних тайтлів з усією необхідною інформацією.
- використання функції "Рандом" для відкриття нового, випадково обраного аніме, що розширює горизонти перегляду;
- управління профілем – можливість перегляду персональної інформації користувача та безпечного виходу з облікового запису;
- відтворення аніме – безпосередній перегляд обраних серій або фільмів за допомогою вбудованого веб-плеєра, який надає базові функції керування відтворенням, забезпечуючи зручний перегляд.

Розглянемо реалізацію інтерактивного компонента "Акордеон" (Accordion) для відображення розширеної інформації на вебсайті (рис. 2.3). Цей компонент дозволяє компактно приховувати та розкривати блоки контенту, оптимізуючи простір на сторінці та покращуючи користувацький досвід.

```
const Accordion: React.FC<AccordionProps> = ({ title, genre, date, children }) => {

    const [isActive, setIsActive] = useState(false);

    return (
        <div className={styles.accordion}>
            <div className={styles.accordion__main} onClick={() => setIsActive(!isActive)}>
                <div className={styles.accordion__main__head}>
                    <div className={styles.accordion__main__head__title}>
                        <h3
                            className={isActive ? styles.accordion__main__head__title__active : styles.accordion__main__head__title__default}
                        >
                            {title}
                        </h3>
                        <ArrowDownIcon className={isActive ? styles.icon__active : styles.icon__default} />
                    </div>
                    <p>{genre}</p>
                </div>
                <p>{date}</p>
            </div>
            {isActive &&
                <div className={styles.accordion__content}>
                    <Link to=''>{children}</Link>
                </div>
            }
        </div>
    )
}
```

Рис. 2.3 – Реалізація інтерактивного компонента "Акордеон"

На початку реалізації компонента Accordion визначаються його властивості (props): title, genre, date та children. Ці властивості дозволяють передавати динамічні дані в компонент, такі як назва аніме, його жанр, дата виходу та довільний внутрішній контент (епізоди, опис тощо).

Всередині компонента використовується хук стану React useState для управління внутрішнім станом isActive. Змінна isActive є булевим значенням, що вказує, чи відкритий (активний) акордеон, чи закритий (неактивний). За замовчуванням акордеон закритий (false).

Реалізація компонента "Акордеон" таким чином забезпечує високу модульність та багаторазове використання у різних частинах вебсайту. Це дозволяє ефективно керувати великими обсягами інформації, надаючи користувачам можливість розгортати лише ті блоки, які їх цікавлять, зберігаючи інтерфейс чистим та не перевантаженим (рис. 2.4).

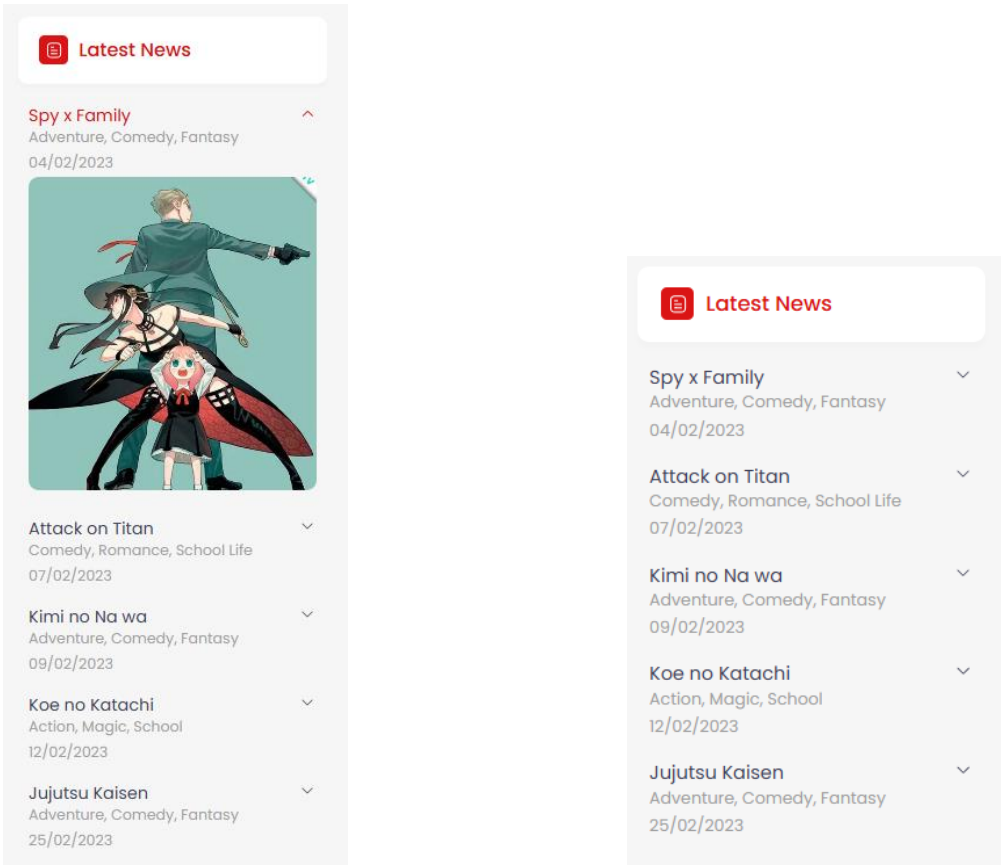

Рис. 2.4 – Відображення компоненту "Акордеон" на сторінці

Розглянемо реалізацію компонента "Скелетон" (Skeleton Loader), зокрема AccordionTabletSkeleton, який використовується для покращення користувацького досвіду під час завантаження даних на вебсайті (рис. 2.5). Скелетон є візуальним плейсхолдером, що імітує структуру та розміри вмісту, який ще завантажується, запобігаючи різким стрибкам макета та надаючи користувачу візуальний зворотний зв'язок.

Реалізація AccordionTabletSkeleton є простим функціональним React-компонентом (React.FC). Його основна функція полягає у відображенні статичної структури, яка візуально повторює форму оригінального компонента "Акордеон", але без реальних даних (рис. 2.6).

```
1   //styles
2   import styles from './AccordionTabletSkeleton.module.scss'
3
4   const AccordionTabletSkeleton: React.FC = () => {
5       return (
6           <div className={styles.accordion}>
7               <div className={styles.accordion__title} />
8               <div className={styles.accordion__info}>
9                   <div className={styles.accordion__info__genre} />
10                  <div className={styles.accordion__info__date} />
11              </div>
12              <div className={styles.accordion__image} />
13          </div>
14      )
15  }
16
17  export default AccordionTabletSkeleton
18
```

Рис. 2.5 – Реалізація компонента "Скелетон"

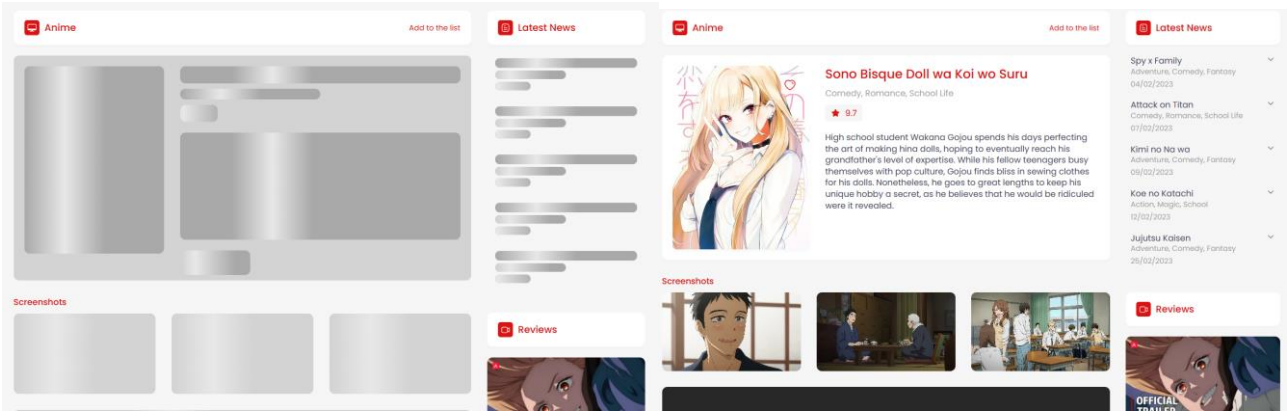

Рис. 2.6 – Відображення компоненту "Скелетон" на сайті

Розглянемо реалізацію функції рандомного вибору аніме для перегляду, яка представлена у фрагменті коду (рис. 2.7). Ця функція є ключовим елементом інтерактивності вебсайту, дозволяючи користувачеві швидко переходити до перегляду випадкового аніме.

```
<nav>
    <ul className={styles.header__main__links}>
        {links.map((link, index) => (
            <li key={index} className={styles.header__main__links__item}>
                {link.name === 'Random' ?
                    <button onClick={(e) => handleClickRandom(e)}>{link.name}</button>
                    :
                    <Link to={link.path}>
                        {link.name}
                    </Link>
                }
                {links.slice(1, 4).includes(link) && <ArrowDownIcon />}
            </li>
        ))}
    </ul>
</nav>
```

Рис. 2.7 – Реалізація функції рандомного вибору аніме

Основна логіка випадкового вибору зосереджена у функції handleClickRandom, яка, судячи з контексту, викликається при натисканні на кнопку "Random" у навігаційному меню.

Після генерації випадкового ID, функція Maps (яка є частиною бібліотеки для маршрутизації, наприклад, React Router) програмно перенаправляє

користувача на нову вебадресу. Шаблон URL /Animenia/${rand} означає, що користувач буде перенаправлений на сторінку деталізації аніме з відповідним випадково згенерованим ID. Наприклад, якщо rand дорівнює 5, URL буде /Animenia/5.

Таким чином, функція handleClickRandom забезпечує швидкий та динамічний спосіб для користувача відкривати новий контент на вебсайті, автоматизуючи процес вибору аніме.

Розглянемо реалізацію функції, яка керує "відстеженням сесії" та її відображенням. Ця логіка забезпечує отримання, відображення та керування даними про сесії користувача на вебсайті (рис. 2.9).

Згідно з наданим JSX, ми бачимо використання функцій getOS(), getGeo(), getDate() та handleLogout(). Ці функції не є прямими викликами до API в цьому JSX, а допоміжними функціями, які отримують або формують дані для відображення, а також обробляють дії користувача (рис. 2.8).

```
{tab === 'Security and privacy' &&
    <section className={styles.sessions}>
        <div className={styles.sessions__wrapper}>
            <HeadingBlock title='Sessions' icon={<FolderIcon />} filter />
            <div className={styles.sessions__item}>
                <div className={styles.sessions__item__os}>
                    <div className={styles.sessions__item__os__icon}>
                        {getOS()?.includes('iOS') || getOS()?.includes('Android') ? <MobileIcon /> : <DesktopIcon />}
                    </div>
                    <div className={styles.sessions__item__os__info}>
                        <h3>OS: {getOS()}</h3>
                        <p>Geo: {getGeo()}</p>
                    </div>
                </div>
                <p className={styles.sessions__item__date}>
                    Date: {getDate()}
                </p>
                <p className={styles.sessions__item__current}>
                    Current Session
                </p>
                <div className={`${styles.sessions__item__remove}
    ${theme === 'light' ? styles.sessions__item__remove__light : styles.sessions__item__remove__dark}`}>
                    <button onClick={handleLogout}>
                        <TrashIcon />
                    </button>
                </div>
            </div>
        </div>
    </section>
}
```

Рис. 2.7 – Реалізація функції відстеження сесії

Функція getOS() (Отримання назви операційної системи):

Ця функція відповідає за визначення та повернення назви операційної системи, з якої була ініційована сесія. Вона використовується для відображення тексту "OS: Windows 10" та вибору відповідної іконки (мобільний чи десктопний).

Функція getGeo() (Отримання географічних даних):

Ця функція повертає географічні дані (наприклад, країну або місто) сесії.

Функція getDate() (Отримання дати сесії):

Повертає дату ініціалізації сесії.

Функція handleLogout() (Обробка виходу із сесії):

Ця функція викликається, коли користувач натискає на кнопку "сміттєвий кошик", щоб завершити певну сесію.

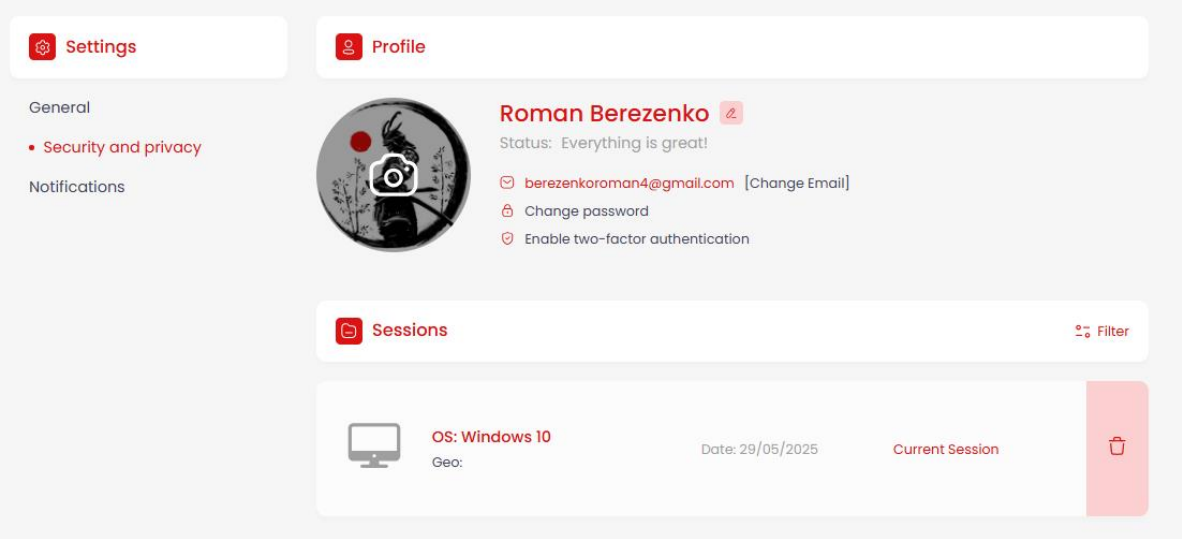

Рис. 2.9 – Відображення компоненту відстеження сесії

2.6. Організація тестування та налагодження програмного засобу

Тестування програмного забезпечення є критично важливим етапом життєвого циклу розробки, спрямованим на ідентифікацію потенційних дефектів та верифікацію відповідності продукту встановленим вимогам.

Тестування передбачає випробування програми на різних сценаріях, що можуть здійснюватися як автоматично, так і вручну. Наприклад, ми провели тестування за такими сценаріями:

- тестування функціональності слайдера: перевірялася коректність відображення, навігації та автоматичного перемикання зображень на вебсторінці;
- тестування адаптивності до зміни розміру вікна браузера: оцінювалася правильність відображення елементів інтерфейсу та їхнього розташування при зміні розмірів вікна браузера, щоб забезпечити коректний вигляд на різних екранах;
- тестування на мобільних застосунках**:** проведено перевірку роботи програми на різних мобільних пристроях (смартфонах, ПК) та операційних системах (iOS), зокрема коректність відображення, взаємодії з елементами та відповідність функціональних вимог.

Цей процес являє собою системне дослідження, метою якого є оцінка якості розробленого рішення в контексті його функціонального призначення. Ефективність тестування досягається шляхом виконання програми з контрольованими тестовими сценаріями та порівняння фактичних результатів із очікуваними.

Для проведення тестування вебсайту «ANIMENIA» було задіяно кілька апаратних конфігурацій з різними технічними характеристиками, що дозволило оцінити продуктивність вебзастосунку у різноманітних умовах експлуатації.

Тестування здійснювалося на таких пристроях:

- Ноутбук, оснащений процесором Intel Core i3-8145U, дискретною відеокартою nVidia GeForce MX150, 8 ГБ оперативної пам'яті, під

керуванням операційної системи Windows 10 Pro;

- ноутбук на базі процесора Intel Pentium Silver N5030, з інтегрованою графікою, 8 ГБ оперативної пам'яті та операційною системою Windows 11;
- ноутбук із процесором Intel Core i5-10300H, відеокартою nVidia GeForce GTX 1650, 8 ГБ оперативної пам'яті, що працював під операційною системою Windows 10 Pro;
- мобільний пристрій iPhone 10 Pro, що забезпечило перевірку адаптивності та функціональності вебсайту на мобільних платформах.

Застосоване ручне тестування дало змогу підтвердити коректність функціонування веб-інтерфейсу «ANIMENIA». Проведені перевірки показали стабільну роботу вебсайту у браузері Google Chrome версії 119.0.6045.200, та у браузері Safari на iPhone, без зафіксованих збоїв чи зависань. Було відзначено, що продуктивність та швидкість відгуку вебзастосунку є вищою на апаратних платформах із потужнішими центральними процесорами, що свідчить про чутливість системи до обчислювальних ресурсів.

Підсумовуючи результати тестування, можна констатувати, що розроблений програмний продукт «ANIMENIA» відповідає встановленим вимогам. Він демонструє необхідний рівень стабільності у роботі та забезпечує належну продуктивність, що є важливим критерієм для якісного функціонування веб-сервісу. Налагодження, що супроводжувало процес тестування, дозволило усунути виявлені недоліки та оптимізувати роботу вебсайту.

2.7. Рекомендації по використанню та впровадженню програмного засобу

Розроблений вебзастосунок «ANIMENIA» створювався як інформаційно-розважальна платформа, призначена для зручного доступу до інформації про аніме, новинні стрічки та іншого тематичного контенту. Основною метою розробки було забезпечення інтуїтивно зрозумілого інтерфейсу для користувачів, які цікавляться аніме, а також надання можливості перегляду основної інформації про тайтли, їх рейтинги та скріншоти. Практичне

застосування програмного продукту дозволяє ефективно організувати доступ до об'ємних даних про аніме, оптимізувати процес пошуку та фільтрації контенту, а також покращити загальний користувацький досвід взаємодії з аніме-спільнотою.

Для забезпечення коректного функціонування вебсайту «ANIMENIA» необхідним є стабільне підключення до мережі Інтернет. Це критично важливо, оскільки застосунок активно взаємодіє із зовнішнім API для отримання даних про аніме та новинний контент. Стосовно серверного обладнання, хоча «ANIMENIA» є фронтенд-застосунком без власної бази даних, для розгортання статичних файлів та коректної роботи веб-сервера, який обслуговуватиме веб-застосунок, рекомендується наступні мінімальні технічні характеристики: щонайменше 4 ГБ оперативної пам'яті, процесор із середнім рівнем продуктивності та достатній обсяг дискового простору для розміщення файлів вебсайту.

Клієнтська частина системи розроблена з використанням адаптивного дизайну, що забезпечує її коректне та естетичне відображення на будь-якому розмірі екрана – від мобільних телефонів до широкоформатних моніторів. Ця адаптивність досягається за рахунок використання сучасних CSS-технік та гнучких компонентів, що дозволяють інтерфейсу динамічно перебудовуватися під різні роздільні здатності та орієнтації пристроїв. Система сумісна з широким спектром сучасних веббраузерів, що підтримують актуальні стандарти HTML5, CSS3 та JavaScript. Це гарантує коректне відображення веб-інтерфейсу та функціональність на різноманітних пристроях, включаючи персональні комп'ютери, планшети та смартфони. Компоненти скелетонів (AccordionTabletSkeleton, AnimeBlockSkeleton, AnimeCardSkeleton) також сприяють поліпшенню користувацького досвіду під час завантаження контенту, мінімізуючи "стрибки макета".

Впровадження програмного засобу «ANIMENIA» можливе як на локальних веб-серверах підприємства (якщо є потреба у внутрішньому хостингу), так і на хмарних платформах, які надають послуги розгортання

статичних вебзастосунків та підтримки взаємодії з API. Архітектура системи, побудована на React та Redux, передбачає високу модульність, можливість подальшого масштабування функціоналу та потенційну інтеграцію з іншими інформаційними системами або сервісами, що може сприяти її адаптації до потреб розширення функціональних можливостей або інтеграції в більші екосистеми.

ВИСНОВОК

У рамках виконання цієї бакалаврської роботи було успішно спроєктовано та реалізовано вебсайт «ANIMENIA», що є інформаційною платформою для любителів аніме. Розроблений програмний засіб забезпечує базову функціональність, необхідну для ефективної взаємодії користувачів з контентом. Користувачам доступний перегляд обширного каталогу аніме-тайтлів, можливість ознайомлення з актуальними новинами індустрії. Реалізовано функціонал, що дозволяє відображати ключову інформацію про кожне аніме.

Проведене тестування підтвердило коректну роботу ключових функцій вебсайту, зокрема, відображення інформації про аніме, новинних стрічок, а також функціонування механізму рандомного вибору аніме. Попри загалом позитивні результати та стабільну роботу вебзастосунку, на поточному етапі розробка має певні обмеження. Зокрема, серед потенційних напрямків для подальшого вдосконалення вебсайту можна виділити наступні: наразі відсутня можливість для користувачів залишати відгуки, коментарі або додавати аніме до особистих списків перегляду. Впровадження цих функцій значно посилить залученість аудиторії та покращить соціальну взаємодію на платформі.

Незважаючи на наявні перспективи для розвитку, усі поставлені завдання в рамках бакалаврської роботи були виконані в повному обсязі. Розроблений вебсайт «ANIMENIA» є міцною та якісною основою, яка може бути використана для подальшого розширення функціоналу, поглиблення інтерактивних можливостей та вдосконалення загального користувацького досвіду, перетворивши його на повноцінний та конкурентоспроможний веб-ресурс для аніме-спільноти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Key Retail Tech Trends Shaping Future E-commerce Growth. *Number Analytics - AI Statistical Software*. URL: https://www.numberanalytics.com/blog/key-retail-tech-trends-shaping-future-ecommerce-growth (дата звернення: 06.05.2025).

2. eCommerce Technology: Top 10 Trends Shaping 2025's Retail Landscape. *Digital Silk*. URL: https://www.digitalsilk.com/digital-trends/ecommerce-technology/ (дата звернення: 06.05.2025).

3. The Future Of E-Commerce Technology. *Forbes*. URL: https://www.forbes.com/councils/forbes'agencycouncil/2023/08/25/the-future-of-e-commerce-technology/ (дата звернення: 06.05.2025).

4. Що таке CDN? Навіщо він потрібен? Що дає його використання?. Best E-Commerce Fully Managed Hosting Platform. URL: https://zahid.host/uk/posts/technical-articles/cdn/ (дата звернення: 29.05.2025).

5. HTML - Hypertext Markup Language. *Freelancer*. URL: https://www.freelancer.com.ua/what-is-html (дата звернення: 08.05.2025).

6. W3Schools.com. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/css/css_intro.asp (дата звернення: 08.05.2025).

7. What Is JavaScript: A Beginner's Guide to the Basics of JS. *Hostinger Tutorials*. URL: https://www.hostinger.com/tutorials/what-is-javascript (дата звернення: 08.05.2025).

8. Вступ | Vue.js. Vue.js - Прогресивний JavaScript фреймворк | Vue.js. URL: https://ua.vuejs.org/guide/introduction.html (дата звернення: 29.05.2025).

9. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/. (дата звернення: 29.05.2025).

10. Advantages and disadvantages ofÂ NestJS. *JavaScript Custom Software Development Company*. URL: https://themobilereality.com/blog/advantages-and-disadvantages-of-nestjs (дата звернення: 10.05.2025).

11. Beighton B. NestJS the Pros and Cons. *Medium*. URL: https://bradbeighton.medium.com/nestjs-the-pros-and-cons-aff714607b07 (дата звернення: 10.05.2025).

12. Documentation | NestJS - A progressive Node.js framework. *Documentation | NestJS - A progressive Node.js framework*. URL: https://docs.nestjs.com/discover/companies (дата звернення: 10.05.2025).

13. Інкрементальна модель життєвого циклу розробки ПЗ: ключові переваги та етапи. *Custom Web & Mobile Development Company - New Line Technologies*. URL: https://newline.tech/inkrementalna-model-zhittyevogo-ciklu-rozrobki-programnogo-zabezpechennya-uk/ (дата звернення: 12.05.2025).

14.Visual Studio Code - Power Apps. Microsoft Learn: Build skills that open doors in your career. URL: http://learn.microsoft.com/uk-ua/power-apps/maker/portals/vs-code-extension (date of access: 29.05.2025).

15. Що таке Visual Studio Code 2025 - ТОП КЕЙС. *ТОП КЕЙС*. URL: https://top-keis.com.ua/shho-take-visual-studio-code/ (дата звернення: 29.05.2025).

16. Kukurudza. Що таке Figma та для кого вона потрібна. CASES. URL: https://cases.media/article/sho-take-figma-ta-dlya-kogo-vona-potribna?srsltid=AfmBOoqhBAHJEGMulTPhp0bhTYtr9WBYgZtt1j9efZPkEEY337U8fUmc (дата звернення: 29.05.2025).

17. Kukurudza. Що таке Figma та для кого вона потрібна. CASES. URL: https://cases.media/article/sho-take-figma-ta-dlya-kogo-vona-potribna (дата звернення: 29.05.2025).

18. Projector Creative & Tech Institute. React: Що таке React? Як почати вивчати Реакт? Основні навички. CASES. URL: https://cases.media/article/sho-take-react-js-yak-pochati-vivchati-reakt-navichki-dlya-react-developer (дата звернення: 29.05.2025).

19. Що таке різниця між JS та JSX? - javascript.org.ua - JS Communities. javascript.org.ua - JS Communities.

URL: https://javascript.org.ua/shho-take-rizniczya-mizh-js-ta-jsx/ (дата звернення: 29.05.2025).

20. Інструменти Reactjs на всі випадки життя. DevZone. URL: https://devzone.org.ua/post/instrumenty-reactjs-na-vsi-vypadky-zyttia (дата звернення: 29.05.2025).

21. Projector Creative & Tech Institute. React: Що таке React? Як почати вивчати Реакт? Основні навички. CASES. URL: https://cases.media/article/sho-take-react-js-yak-pochati-vivchati-reakt-navichki-dlya-react-developer?srsltid=AfmBOorJG3smFWiEHf8qCr52TFuZ_dHeh5en4M4rV5Vx35X5kSM70UWo (дата звернення: 29.05.2025).

22. Використовуємо хук ефекту – React. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/docs/hooks-effect.html (дата звернення: 29.05.2025).

23. React API верхнього рівня – React. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/docs/react-api.html (дата звернення: 29.05.2025).

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ

ВСТУП

Цей проєкт передбачає створення вебсайту «ANIMENIA», що є інформаційною платформою, присвяченою аніме. Розроблений програмний продукт надасть користувачам зручні засоби для перегляду каталогу аніме-тайтлів, ознайомлення з новинами індустрії та інтерактивної взаємодії з контентом. Основною метою розробки є формування сучасної вебплатформи, яка забезпечить ефективний доступ до інформації про аніме та покращить користувацький досвід для фанатів.

ПІДСТАВИ ДЛЯ РОЗРОБКИ

Робота зі створення вебсайту «ANIMENIA» здійснюється в рамках виконання бакалаврської кваліфікаційної роботи.

ПРИЗНАЧЕННЯ РОЗРОБКИ

Програмний продукт призначений для забезпечення зручного та ефективного доступу до структурованої інформації про аніме. Ключові функції включають:

- перегляд розширеного каталогу аніме-тайтлів;
- можливість рандомного вибору аніме для перегляду;
- управління доступом користувача;
- перехід на окрему сторінку конкретного аніме;
- зміна теми сайту на темну;
- адаптація під різне розширення.

Вебсайт розробляється для використання в інформаційно-розважальному вебсередовищі, орієнтованому на спільноту шанувальників аніме.

ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмний продукт повинен забезпечувати виконання наступних функцій:

- можливість перегляду аніме-тайтлів з детальною інформацією (опис, рейтинги, скріншоти);
- можливість рандомного вибору аніме для перегляду;
- перехід на окрему сторінку конкретного аніме;
- зміна теми сайту на темну;
- адаптація під різне розширення.

Вимоги до надійності:

- стабільна робота у різних сучасних веббраузерах (Chrome, Firefox, Safari, Edge);
- коректне та швидке завантаження і відображення контенту, мінімізація "стрибків макета" за допомогою скелетонів;
- висока адаптивність вебінтерфейсу для коректного відображення на всіх розмірах екранів, від мобільних пристроїв до широкоформатних моніторів;
- мінімізація часу відновлення після потенційних збоїв (наприклад, збою API);
- можливість оперативного внесення виправлень та оновлень у програмний код.

Вимоги до експлуатації:

Для забезпечення стабільної роботи програмного продукту необхідне стабільне інтернет-з’єднання, оскільки вебсайт взаємодіє із зовнішніми API. Клієнтська частина системи розрахована на використання в сучасних веббраузерах, що підтримують стандарти HTML5, CSS3 та JavaScript.

ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документів входить текст бакалаврської роботи та це технічне завдання.

ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Розробка вебсайту «ANIMENIA» дозволить підвищити зручність доступу до аніме-контенту для цільової. Створення інтуїтивно зрозумілого інтерфейсу зменшить час, який користувачі витрачають на пошук бажаного контенту, покращить їхній досвід взаємодії та сприятиме залученню нової аудиторії. Хоча пряма економічна вигода від такого інформаційного ресурсу не вимірюється продажами, вона може бути реалізована через підвищення лояльності користувачів, збільшення трафіку та потенційне монетизаційне масштабування у майбутньому (наприклад, через рекламу або партнерські програми). Це робить вебсайт перспективним з точки зору формування активної спільноти та задоволення інформаційних потреб фанатів аніме.

СТАДІЇ І ЕТАПИ РОЗРОБКИ

1. Аналіз вимог та визначення цілей проєкту.
2. Проєктування архітектури системи (зокрема, фронтенду та взаємодії з API).
3. Розробка фронтенд-частини (інтерфейсу користувача та логіки React/Redux).
4. Реалізація ключової функціональності, а саме відображення новин, функція рандомного вибору аніме, вибір теми сайту, перегляд аніме, адаптація сайту, управління доступом користувача, перехід на окрему сторінку конкретного аніме.
5. Тестування, виявлення та усунення виявлених дефектів.
6. Введення вебсайту в тестову експлуатацію.

ПОРЯДОК КОНТРОЛЮ Й ПРИЙМАННЯ

Приймання результатів роботи здійснюється в процесі захисту бакалаврської кваліфікаційної роботи.

АНОТАЦІЯ

Наумік С. В. Дослідження та використання фреймворку React для проєктування вебсайтів зі збірками потокових медіаданних. Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 122 Комп'ютерні науки. Волинський національний університет імені Лесі Українки, Луцьк, 2025 р.

Актуальність теми цієї роботи визначається стрімким зростанням популярності аніме як культурного феномену та потребою у створенні зручних, функціональних онлайн-ресурсів.

Метою дослідження стало створення інтерактивного вебзастосунку, присвяченого аніме, який надає користувачам функції перегляду каталогу аніме-тайтлів.

Розроблений програмний продукт є фронтенд-орієнтованим вебзастосунком, реалізованим з використанням сучасних вебтехнологій: HTML, CSS, JavaScript (з використанням бібліотеки React) та Redux для управління станом. Для отримання інформації про аніме та новинного контенту застосунок взаємодіє із зовнішнім веб-API, що дозволяє йому функціонувати без власної постійної серверної бази даних. У застосунку реалізовано можливості перегляду аніме, зміни теми сайту, вибір рандомного аніме для перегляду, управління доступом користувача та адаптація під різні пристрої.

У роботі викладено принципи архітектури вебзастосунку, розглянуто ключові програмні модулі, структуру інтерфейсу користувача та алгоритми, які забезпечують основну логіку роботи програми. Крім того, представлено перспективи подальшого розвитку функціоналу, зокрема можливість впровадження системи автентифікації користувачів для персоналізації досвіду та інтеграції додаткових інтерактивних можливостей, таких як відгуки та формування власних списків перегляду.

Ключові слова: вебзастосунок, аніме, вебтехнології, React, Redux, API, фронтенд-розробка, інформаційна платформа.