Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Вебзастосунок для перегляду та фільтрації спортивних новин
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.2201106.01.12.ПЗ

Виконав студент IV курсу, група ІПЗ-22-1 _____ Денис ПЕТРИК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник канд. техн. наук, доцент _____ Юрій ФОРКУН
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. техн. наук, доцент _____ Оксана ЯШИНА
Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

3 червня 2026 р.

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Інженерії програмного забезпечення

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
2.01. 2026 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Петрику Денису Максимовичу
Прізвище, ім'я, по батькові студента

1. Тема кваліфікаційної роботи Вебзастосунок для перегляду та фільтрації спортивних новин

Керівник кваліфікаційної роботи Форкун Юрій Вікторович, канд. техн. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. №7

2. Строк подання студентом роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області, визначення мети та завдань, постановка задачі, аналіз існуючих рішень, проєктування застосунку, програмна реалізація та тестування

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди для доповіді), діаграма варіантів використання, зображення архітектурного рішення, структурна схема реалізованої бази даних.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина О. М., доцент	04.05.26	20.05.26
Антиплагіат	Форкун Ю. В., доцент	05.05.26	25.05.26

7. Дата видачі завдання « 02 » 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проєктування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12 – 31.12.2025	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	01.01 – 20.02.2026	
3 Проєктування програмного забезпечення	21.02 – 20.03.2026	
4 Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 – 30.04.2026	
5 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2026	
6 Попередній захист КвР	Травень 2026	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2026	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2026	

Студент _____ Денис ПЕТРИК
Підпис Ім'я, прізвище

Керівник роботи _____ Юрій ФОРКУН
Підпис Ім'я, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Вебзастосунок для перегляду та фільтрації спортивних новин».

Автор роботи: Петрик Денис Максимович.

Керівник роботи: Форкун Юрій Вікторович.

Пояснювальна записка: 91 с., 19 рис., 16 табл., 2 дод., 53 джерела.

ВЕБЗАСТОСУНОК, ФІЛЬТРАЦІЯ, АУТЕНТИФІКАЦІЯ, ІНТЕРНЕТ-МАГАЗИН, ЕЛЕКТРОННА КОМЕРЦІЯ, ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ, ПЕРЕГЛЯД СПОРТИВНИХ НОВИН, КОШИК ЗАМОВЛЕНЬ.

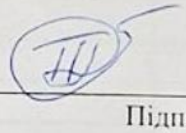
Метою кваліфікаційної роботи є розробка вебзастосунку для перегляду та фільтрації спортивних новин.

У ході роботи над даним програмним продуктом було проаналізовано предметну область, що стосується новинних ресурсів, переглянуто ряд існуючих рішень, сформульовано мету роботи та виділено ряд задач, які необхідно реалізувати для досягнення поставленої мети.

Вебзастосунок розроблявся за допомогою архітектурного рішення RESTful API, бази даних PostgreSQL, фреймворк Next.js, який базується на React.

У результаті проробленої роботи розроблено вебзастосунок для перегляду та фільтрації спортивних новин, що надасть змогу спортивним фанатам знаходити новини спорту, що їх цікавлять, комунікувати один з одним, залишати відгуки та інформаційні повідомлення тощо.

25.05.2026
Дата



Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документу	Найменування документу	К-сть аркушів	№ екз.	Примітка
			Текстові документи	1		
1	А4	КвРІПЗ.2201106.01.12.ПЗ	Пояснювальна записка	91		
2	А4		Завдання на кваліфікаційну роботу	1		
3	А4		Анотація	1		
4	А3		Графічні документи	3		
5	А4		Презентаційні матеріали	17		

Змн.	Арк	№ докум.	Підпис	Дата	КвРІПЗ.2201106.01.12.ВД		
Виконав		Петрик Д. М.		25.05	Вебзастосунок для перегляду та фільтрації спортивних новин Відомість документів	Літ.	Арк. 5 / Аркушів
Керівник		Форкун Ю.В.		25.05			
Реценз.						ХНУ, ІПЗ-22-1	
Н. контр.		Яшина О.М		25.05			
Зав. каф.		Бедратюк Л. П.		25.05			

ЗМІСТ

ВСТУП

Актуальність розробки вебзастосунку для спортивних новин обумовлена стрімкою цифровою трансформацією медіа та зміною звичок споживання контенту. Сьогодні спорт - це не просто результати матчів, а величезна екосистема даних, інтерактивності та персоналізації.

Нижче наведено ключові чинники, що роблять цей проєкт затребуваним. Глобальний спортивний календар 2026 показує, що цей рік є піковим для спортивної індустрії. Підготовка та проведення мегаподій, таких як Чемпіонат світу з футболу 2026 (США, Канада, Мексика) та Зимові Олімпійські ігри в Мілані, створюють безпрецедентний попит на якісні цифрові платформи. Вболівальники шукають єдину точку доступу до розкладів, результатів та аналітики в режимі реального часу.

Ера «Real-time» та інтерактивності сприяє тому, що сучасний користувач не готовий чекати. Актуальність застосунку визначається його здатністю надавати:

- миттєві сповіщення про голи, трансфери чи травми гравців;
- живі текстові трансляції, які доповнюються відеофрагментами та графікою;
- другий екран (Second Screen), що дозволяє використовувати вебзастосунок під час перегляду матчу на ТБ для перевірки статистики та обговорення в чатах.

Гіперперсоналізація через штучний інтелект дає велику перевантаженість інформацією. Вебзастосунок стає актуальним, якщо він пропонує так звану Smart Feed (розумну стрічку), яка:

- фільтрує новини лише за улюбленими командами чи гравцями;
- використовує штучний інтелект для автоматичного створення коротких описів довгих статей чи матчів;
- надає предиктивну аналітику, наприклад, прогнози результатів на основі Big Data).

Зростання нових ніш, таких як жіночий спорт та кіберспорт потребують висвітлення новин про спортивні події. Зокрема, спостерігається вибуховий інтерес

до жіночого футболу, баскетболу та регбі, а також стабільне зростання кіберспорту. Традиційні медіа часто ігнорують ці напрямки, тому створення платформи, що професійно висвітлює ці ніші, є стратегічно вигідним кроком для залучення нової лояльної аудиторії

Гейміфікація та ком'юніті призвели до того, що сучасні вебзастосунки перестали бути лише «газетами». Вони актуальні як платформи для взаємодії:

- фентезі-ліги та прогнози, оскільки інтеграція ігрових механік підвищує час перебування на сайті;
- мікроспільноти дають можливість обговорювати події з фанатами в закритих або публічних чатах безпосередньо в застосунку.

Актуальність проєкту полягає у необхідності створення високотехнологічного інструменту, який об'єднує розрізнені джерела спортивної інформації в персоналізований інтерактивний простір. Це відповідь на запит ринку щодо швидкості, точності даних та інтеграції штучного інтелекту в повсякденне життя вболівальника.

Метою кваліфікаційної роботи є розробка вебзастосунку для перегляду та фільтрації спортивних новин.

Для досягнення мети, необхідно виконати ряд задач даної роботи:

- провести аналітичну роботу із областю новинних ресурсів, визначити її межі та шляхи реалізації;
- проаналізувати існуючі рішення, що стосуються огляду та фільтрування спортивних новин;
- визначити функціональні та нефункціональні вимоги та здійснити постановку задачі;
- спроектувати вебзастосунок;
- здійснити реалізації новинного вебзастосунку;
- протестувати створений новинний застосунок.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Вебзастосунок – це інтерактивна програма, яка працює на сервері та доступ до якої здійснюється через веб-браузер. Ці додатки необхідні для надання послуг користувачам і пропонують цінний зворотний зв'язок розробникам. Цей зворотний зв'язок часто включає дані про використання, що надають уявлення про взаємодію користувачів, уподобання та часто використовувані функції.

Розробка вебзастосунку для спортивних новин охоплює сферу цифрових мас-медіа, яка спеціалізується на зборі, систематизації та оперативному розповсюдженні інформації про світові та локальні спортивні події. Це середовище характеризується надзвичайно високою динамікою оновлення даних, де актуальність контенту може вимірюватися хвилинами або навіть секундами, особливо під час проведення прямих трансляцій чи знакових турнірів.

Інформаційне поле даної галузі складається з різнорідних потоків даних, що надходять від офіційних спортивних федерацій, аналітичних агенцій, журналістів та систем автоматичної фіксації результатів матчів. Головною особливістю є поєднання статичної інформації, а саме архівні дані, біографії атлетів, правила змагань та динамічного контенту, наприклад, поточні рахунки, термінові новини, коментарі в реальному часі. Управління таким обсягом інформації вимагає чіткої класифікації, оскільки користувачі мають різні інтереси – від загального огляду подій дня до глибокого моніторингу конкретного футбольного клубу чи тенісиста.

Структурно предметна область базується на складній системі взаємозв’язків між ключовими сутностями. У центрі системи знаходиться Новина, яка є складеним об'єктом, що містить заголовок, текстовий контент, медіафайли (фото/відео) та метадані (час публікації, автор). Ця сутність обов'язково прив'язується до Категорії спорту (футбол, бокс, кіберспорт), що дозволяє будувати ієрархічну навігацію.

Додатково структура включає систему тегів, які забезпечують перехресні зв'язки: одна новина може стосуватися конкретної команди, окремого гравця та міжнародного турніру одночасно. Важливим структурним елементом є також

Джерело, оскільки в спортивній журналістиці критично важливо відстежувати першоджерело інформації для підтвердження її достовірності. На рівні бази даних це зазвичай реалізується через розгалужену реляційну схему, де таблиці новин пов'язані з довідниками видів спорту та профілями учасників змагань.

Функціональність спортивного вебзастосунку виходить за межі звичайного відображення тексту і зосереджена на ефективному пошуку та висвітленні пріоритетної інформації. Ключовою функцією є багаторівнева фільтрація, яка дозволяє користувачу миттєво відсіяти контент за видом спорту, датою, популярністю, кількістю переглядів або специфічними ключовими словами. Це вимагає розробки складних алгоритмів запитів до бази даних, які забезпечують швидку видачу результатів навіть при великій кількості одночасних звернень.

Ще однією функціональною особливістю є персоналізація стрічки, де система на основі попередніх переглядів користувача пропонує найбільш релевантні новини. Також обов’язковим є функціонал інтеграції з зовнішніми сервісами через API для отримання турнірних таблиць та результатів матчів у реальному часі, що перетворює застосунок із простого новинного ресурсу на повноцінний інформаційний хаб. На рівні адміністратора функціональність розширюється інструментами для управління контентом, модерації коментарів та аналітики читацької активності.

Для того щоб опис предметної області був вичерпним, необхідно вийти за межі суто новинної стрічки та розглянути екосистему спортивного застосунку як інтерактивну та високонавантажену платформу. До вже згаданих структурних елементів варто додати кілька критично важливих аспектів, які визначають успіх такого продукту в сучасних умовах.

Важливою частиною предметної області є чітка диференціація прав доступу, що реалізується через механізм RBAC (Role-Based Access Control). На відміну від простого сайту, спортивний портал передбачає наявність кількох типів користувачів. Адміністратори керують технічними параметрами та структурою категорій. Редактори та журналісти мають доступ до спеціалізованої панелі керування, де вони не лише створюють тексти, а й працюють із чернетками, планують публікації за часом та модерують коментарі. Зареєстровані користувачі

отримують можливість створювати власні добірки обраної інформації, залишати реакції та налаштовувати сповіщення, тоді як анонімні відвідувачі мають доступ лише до перегляду загальної стрічки. Така структура вимагає інтеграції складних механізмів авторизації, що на базі вашого стека зазвичай реалізується через Spring Security.

Оскільки спортивні новини неможливі без якісного візуального супроводу, предметна область включає складну систему управління медіа-активами. Це не просто зберігання картинок у базі даних, а використання зовнішніх сховищ та мереж доставки контенту. Технологічно це означає, що при завантаженні зображення розробник має передбачити генерацію різних розмірів (thumbnail, medium, full-size) для швидкого відображення на мобільних пристроях та десктопах. Також сюди входить робота з метаданими зображень (Alt-тексти, описи), що є критично важливим для доступності інтерфейсу та пошукової оптимізації.

Сучасний вболівальник очікує оновлення рахунку матчу без перезавантаження сторінки. Це вводить у предметну область поняття Live-апдейтів, що технічно реалізується через протоколи WebSockets або Server-Sent Events. У структурі застосунку з'являється необхідність обробки потокових даних, де сервер штовхає інформацію про забитий гол прямо у браузер користувача. Це суттєво змінює функціональну модель: застосунок перетворюється з пасивного отримувача запитів на активне джерело подій. Такий підхід вимагає особливої уваги до продуктивності бекенду на Spring Boot, щоб підтримувати тисячі одночасних з'єднань під час фіналів великих чемпіонатів.

Спортивні новини існують у конкурентному середовищі пошукових систем, тому предметна область включає в себе вимоги до семантичної розмітки. Це використання спеціальних форматів даних, наприклад, JSON-LD за стандартом Schema.org, які пояснюють пошуковим роботам Google, де в тексті вказано назву команди, де результат матчу, а де автор статті. Функціонально це означає, що вебзастосунок повинен автоматично генерувати ці метадані для кожної сторінки. Без цього аспекту навіть найякісніший контент залишиться непоміченим аудиторією,

що робить SEO не просто маркетинговим інструментом, а невід'ємною функціональною вимогою до коду.

Спортивна тематика характерна екстремальними піками навантаження. Коли відбувається знакова подія, наприклад, трансфер зіркового гравця, кількість запитів до бази даних може зрости у сотні разів за лічені хвилини. Тому опис предметної області обов'язково має включати механізми кешування. Використання інструментів на кшталт Redis дозволяє зберігати найбільш популярні новини в оперативній пам'яті, знімаючи навантаження з основної бази даних. Це функціональна особливість, яка гарантує, що застосунок не перестане функціонувати у найвідповідальніший момент, забезпечуючи безперебійний доступ до новин для всіх вболівальників.

Об'єднавши ці аспекти, можна отримати повну картину сучасного вебсервісу, який виступаж не просто сайтом з новинами, а складною інформаційною системою, яка поєднує в собі управління ролями, роботу з медіа, реальний час та високу продуктивність.

1.2 Аналіз рішень в рамках предметної області

Для аналізу сучасних тенденцій у розробці інформаційних систем було обрано сім провідних ресурсів, що спеціалізуються на спортивному та загальному контенті. Кожен із них має унікальні технічні рішення, які варто врахувати при проектуванні вебзастосунку в рамках даної кваліфікаційної роботи. Крім цього, ресурс демонструє високий рівень адаптивності інтерфейсу, що забезпечує комфортне використання як на персональних комп’ютерах, так і на мобільних пристроях.

1. BBC Sport (рисунок 1.1)

BBC Sport – це провідний підрозділ британської медіакорпорації BBC, який відповідає за висвітлення спортивних подій у всьому світі. Це один із найавторитетніших і найстаріших спортивних мовників, відомий своїми якісними трансляціями, аналітикою та оперативними новинами.

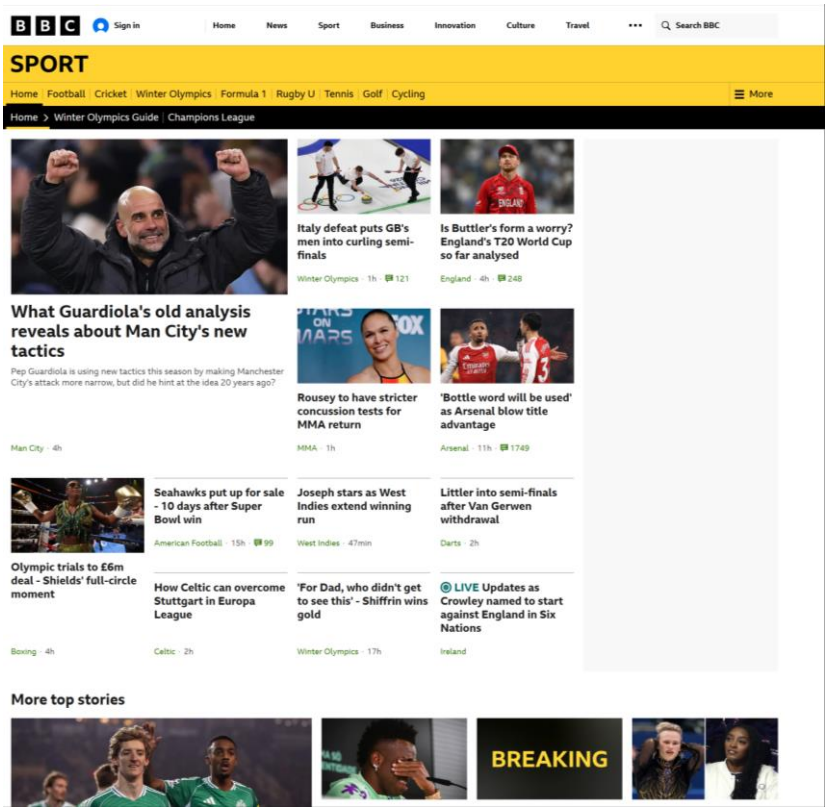


Рисунок 1.1 – BBC Sport

Цей ресурс виділяється серед інших своєю консервативною, але надзвичайно продуманою структурою. Його головною силою є високий рівень доступності (Accessibility) та швидкість завантаження на будь-яких пристроях. З точки зору розробника, архітектура BBC вражає використанням потужних мереж доставки контенту (CDN) та агресивного кешування, що дозволяє витримувати мільйонні стрибки трафіку під час Олімпійських ігор. Слабкою стороною для сучасного користувача може здатися занадто суворий дизайн, який обмежує використання інтерактивних елементів. Можна відзначити їхню модульну систему компонентів, яка забезпечує ідеальну узгодженість інтерфейсу на всіх рівнях.Також варто відзначити ефективну систему категоризації новин та чітку навігацію, завдяки чому користувач може швидко знаходити потрібну інформацію навіть при великому обсязі контенту.Також сайт підтримує єдиний стиль подачі контенту, що позитивно впливає на зручність користування та сприйняття інформації.

2. The Guardian Sports Section (рисунок 1.2).

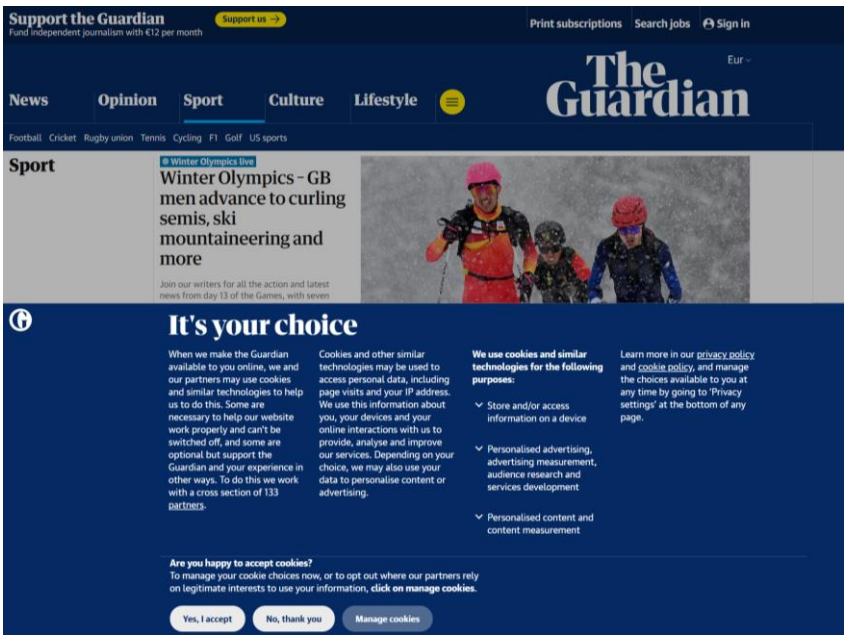


Рисунок 1.2 – The Guardian Sports Section

Цей портал вважається одним із найкращих у використанні типографіки та візуальної ієрархії. Сильна сторона полягає в інтерактивних матеріалах та глибокій аналітиці, що подається через складну інфографіку. Для програміста The Guardian є цікавим прикладом використання сучасних фронтенд-фреймворків, зокрема Svelte та React для створення динамічних сторінок, які відчуваються як живий організм. Слабкою стороною є висока вага сторінок через велику кількість медіафайлів, що може уповільнювати роботу при слабкому інтернет-з'єднанні. З технічного погляду варто звернути увагу на їхню Open Platform API, яка є зразком того, як новини можуть бути доступними для зовнішньої інтеграції. Крім цього, портал демонструє високий рівень адаптивності інтерфейсу та продуману організацію інформації, що забезпечує зручну взаємодію користувача із системою на різних типах пристроїв. Важливою перевагою є використання сучасних підходів до візуалізації даних, завдяки чому навіть складні аналітичні матеріали подаються у зрозумілому та привабливому вигляді. Також ресурс вирізняється ефективною інтеграцією мультимедійного контенту, поєднуючи текстові матеріали, відео, інтерактивні графіки та анімації в єдину інформаційну систему.

3. Новинний ресурс ESPN (рисунок 1.3).

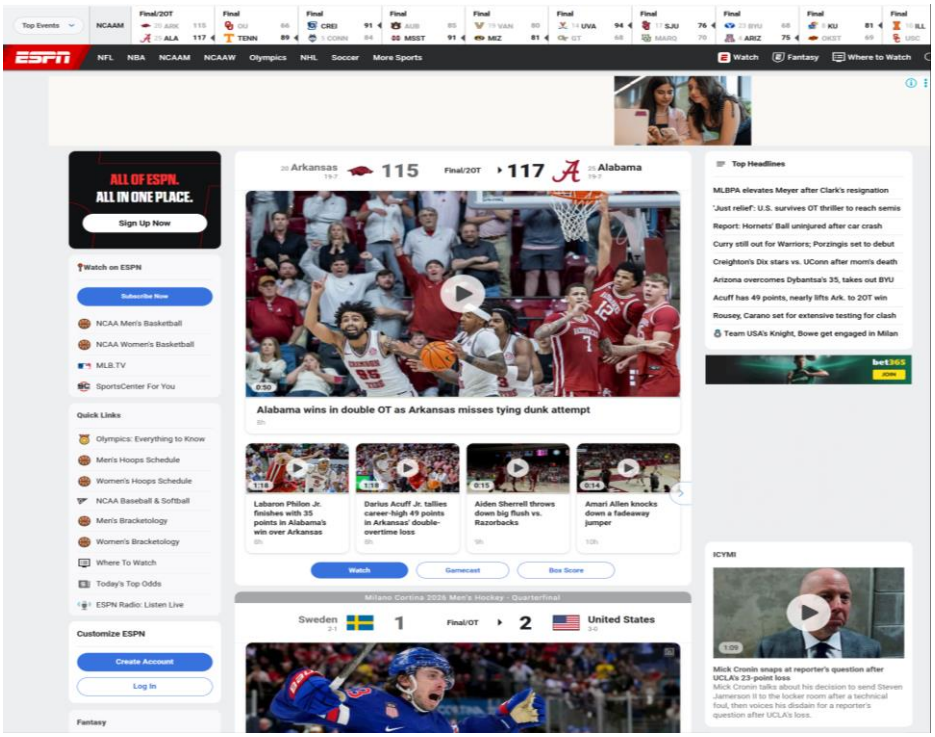


Рисунок 1.3 – ESPN

ESPN є лідером у наданні статистичної інформації в реальному часі. Їхня сильна сторона - це інтеграція живих результатів матчів безпосередньо в новинну стрічку. Проте, для користувача мінусом є надмірна перевантаженість інтерфейсу рекламними блоками та банерами. Тут простежується складна мікросервісну архітектура, де бекенд на Java або Go обробляє тисячі запитів на секунду від різних постачальників даних. Їхня система фільтрації за видами спорту та лігами є надзвичайно деталізованою, що можна взяти за приклад для розроблюваного проекту, але варто уникати такої щільності елементів, яка погіршує сприйняття коду сторінки.Окремо варто відзначити якісно реалізовану систему навігації між матеріалами, яка дозволяє користувачам швидко переходити між пов’язаними новинами та аналітичними статтями. Завдяки цьому портал підтримує високий рівень залучення аудиторії та покращує загальний користувацький досвід.Також важливою перевагою є підтримка сучасних стандартів веброзробки та постійне оновлення функціональних можливостей платформи відповідно до актуальних тенденцій цифрових медіа.

4. The Athletic (рисунок 1.4).

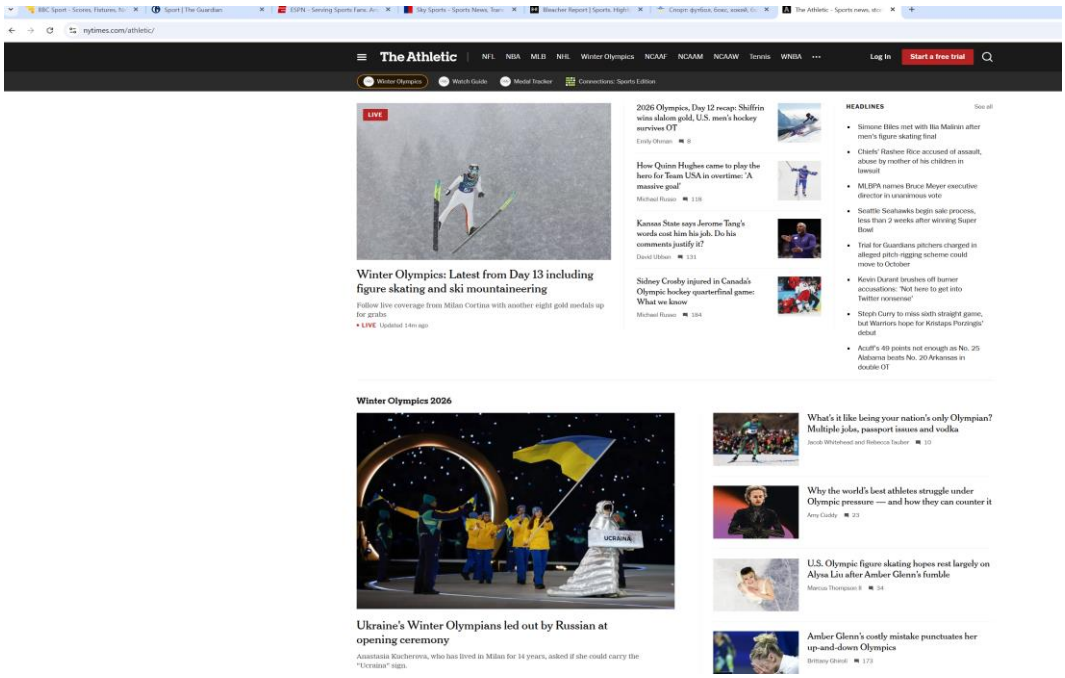


Рисунок 1.4 – The Athletic

Цей ресурс зробив ставку на чистий досвід читання без реклами, де сильною стороною є бездоганний інтерфейс (UI) та персоналізація, тобто користувач обирає улюблені команди, і весь сайт підлаштовується під нього. Слабкість для загального загалу - жорсткий paywall, тобто платний доступ. З позиції розробника, The Athletic є ідеальним кейсом реалізації системи профілів та гнучкого управління правами доступу RBAC. Технологічно сайт виглядає як дуже швидкий Single Page Application (SPA), де переходи між новинами відбуваються миттєво завдяки розумному префетчінгу, тобто попередньому завантаженню даних. Крім того, архітектура оптимізована під високі навантаження під час важливих матчів, оскільки кешування контенту на рівні CDN мінімізує затримки та запити до центральної бази даних. Фронтенд-частина, ймовірно побудована на базі сучасних фреймворків на кшталт React або Next.js, забезпечує плавну анімацію інтерфейсу, що створює відчуття користування нативним мобільним додатком прямо у браузері. З боку бекенду ж реалізовано мікросервісну структуру, яка дозволяє ізольовано оновлювати модулі аналітики, коментарів та трансляцій, не порушуючи стабільність усього ресурсу.

5. Sky Sports (рисунок 1.5)

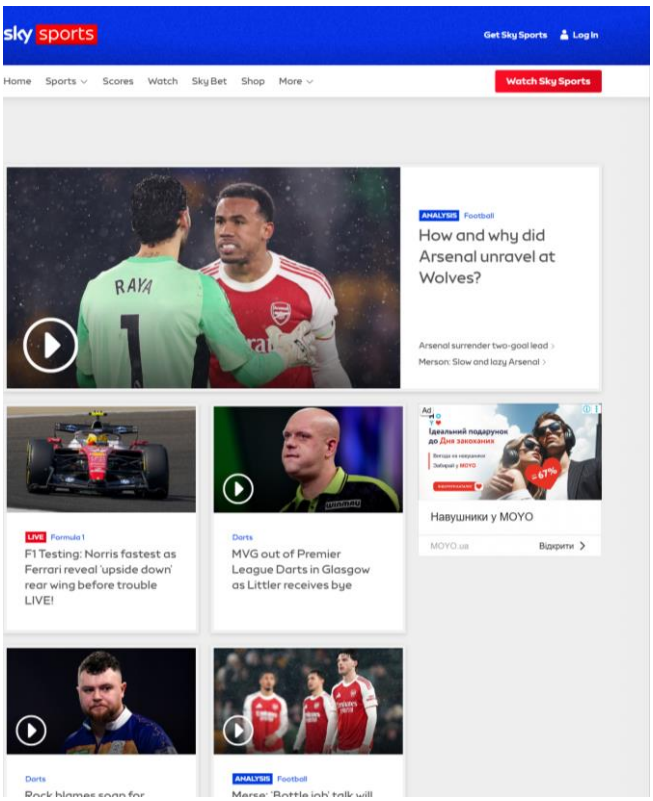


Рисунок 1.5 – Мультимедійний хаб Sky Sports

Мультимедійний хаб Sky Sports спеціалізується на відеоконтенті та прямих ефірах. Головна перевага - безшовна інтеграція відеопрогравачів та потокового мовлення. Слабкою стороною є навігація, яка іноді здається заплутаною через величезну кількість вкладених категорій. Для розробника цей сайт є цікавим з огляду на обробку медіа-потоків та використання складних скриптів для оновлення результатів на льоту через WebSockets. Це яскравий приклад того, як потрібно проектувати бекенд для підтримки великої кількості одночасних з'єднань, що важливо для теми роботи у частині свіжих та актуальних новин. Для забезпечення стабільної роботи за такої архітектури розробники використовують технології адаптивного стрімінгу, як-от HLS або DASH, що дозволяє автоматично підлаштовувати якість відео під швидкість інтернету користувача. З боку фронтенду навантаження знижують за допомогою концепції "lazy loading" для важких медіа-елементів, завдяки чому відеоплеєри ініціалізуються лише тоді, коли потрапляють у зону видимості екрана.

6. Bleacher Report (рисунок 1.6).

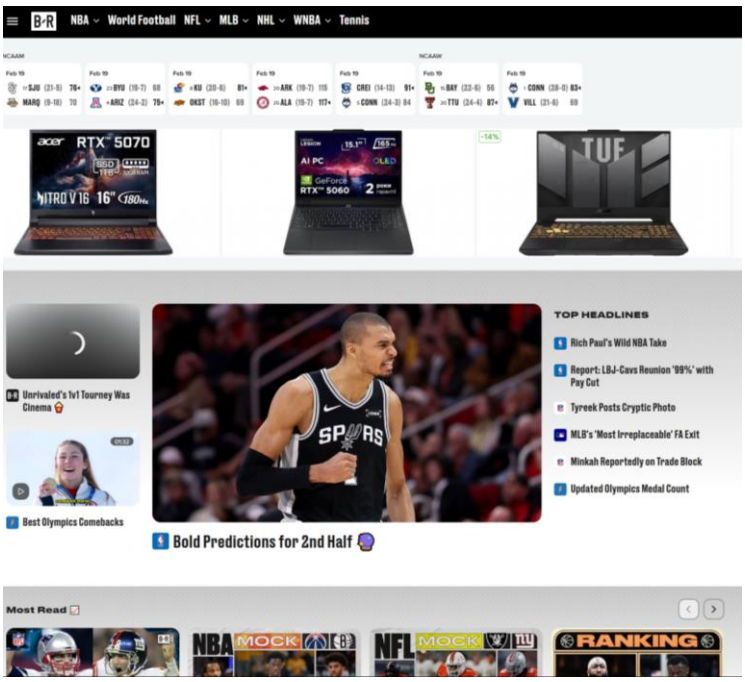


Рисунок 1.6 – Bleacher Report

Даний ресурс орієнтований на молоду аудиторію та формат «швидких» новин та забезпечує соціальну інтеграцію та мобільність. Сильною стороною є агрегація думок із соцмереж та коротка подача фактів. Мінусом можна відзначити низьку глибину аналітики. З точки зору розробника, Bleacher Report демонструє майстерне використання push-повідомлень та інтеграції із зовнішніми API соціальних платформ. Їхня фронтенд-частина оптимізована під принцип Mobile First, що є критично важливим для сучасного вебзастосунку. Внутрішня структура сайту нагадує потік подій Event-driven architecture, де кожна новина є окремою одиницею контенту, яку легко поширювати. Крім того, така архітектура дозволяє реалізувати асинхронну обробку даних на бекенді, де взаємодія з API таких платформ, як X (Twitter), TikTok та Instagram, відбувається через черги повідомлень, наприклад, RabbitMQ або Apache Kafka, що запобігає блокуванню основного потоку програми. Фронтенд, побудований за принципом Mobile First, активно використовує Progressive Web App (PWA) технології, що дозволяє кешувати критично важливі компоненти інтерфейсу на пристрої користувача та забезпечувати працездатність навіть за нестабільного інтернету.

7. Tribuna.com (рисунок 1.7).

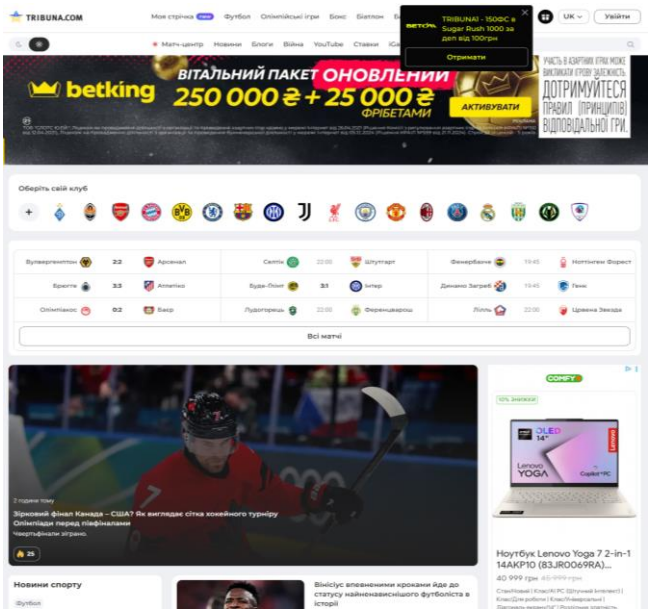


Рисунок 1.7 – Tribuna.com

Український ресурс, який вдало поєднує редакційний контент та блоги користувачів. Сильна сторона - потужна ком'юніті-платформа. Слабкою стороною є іноді застарілий візуальний стиль деяких розділів. Слід відзначити їхню систему коментування та рейтингів контенту, що є прикладом реалізації складних зв'язків у базі даних багато до багатьох між користувачами, постами, тегами та лайками. Для кваліфікаційної роботи це корисний зразок того, як можна організувати архітектуру для зберігання призначеного для користувача контенту паралельно з офіційними новинами.

Аналізуючи ці сім ресурсів, можна виділити кілька технологічних трендів, які варто застосувати у розроблюваному вебзастосунку на Spring Boot та PostgreSQL:

- API-first підхід, тобто розділення бекенду та фронтенду дозволяє легко масштабувати систему та додавати мобільні додатки в майбутньому;
- розумне кешування, а саме використання Redis або вбудованих засобів Spring для зниження навантаження на PostgreSQL при запитах популярних новин;
- динамічне оновлення через реалізацію так званої живої стрічки через WebSockets для підвищення інтерактивності, як це зроблено у Sky Sports чи ESPN;

– семантична база даних завдяки структуруванню PostgreSQL так, щоб забезпечити миттєву фільтрацію за перехресними тегами (вид спорту + команда + дата), що є у BBC та The Athletic.

1.3 Визначення вимог до рішення

Для формування вимог до програмного застосунку важливо не просто перелічити функції, а описати логіку взаємодії користувача з платформою та технічні стандарти, яким вона має відповідати.

Загалом вимоги до будь-якого програмного забезпечення під час його проектування та розробки розподіляються на функціональні та нефункціональні вимоги.

Основна мета застосунку полягає в забезпеченні безперебійного доступу до спортивного контенту. Система розмежування прав передбачає наявність трьох ролей: неавторизований відвідувач, зареєстрований користувач та адміністратор. Гість має можливість переглядати загальну стрічку та користуватися базовим пошуком. Натомість авторизована особа отримує доступ до персонального кабінету, де зберігаються налаштування фільтрації, наприклад, за улюбленими лігами та перелік збережених матеріалів для перегляду в режимі офлайн.

Механізм фільтрації є ключовим компонентом і має працювати динамічно. Користувач повинен мати змогу комбінувати параметри: обирати конкретний вид спорту, наприклад, футбол, баскетбол, теніс, часовий проміжок публікації та джерело походження новини. Система повинна миттєво оновлювати вміст сторінки без повного її перезавантаження, що забезпечується засобами асинхронних запитів до API.

Адміністративна частина системи відповідає за керування джерелами агрегації. Це включає можливість додавання нових RSS-каналів або налаштування парсерів для конкретних спортивних порталів. Також адміністратор повинен мати інструменти для модерації контенту та керування обліковими записами користувачів у разі порушення правил спільноти.

Технічна якість продукту визначається його стабільністю та зручністю. У таблиці 1.1 наведено ключові характеристики.

Таблиця 1.1 – Ключові характеристики згідно вимоги

Характеристика	Опис вимоги
Продуктивність	Час відповіді сервера не повинен перевищувати 500 мс, а завантаження основної стрічки новин має тривати до 2 с.
Адаптивність	Інтерфейс повинен коректно відображатися на всіх типах пристроїв (mobile-first approach) з мінімальною роздільною здатністю 320px.
Безпека	Усі персональні дані та паролі мають бути зашифровані. Використання протоколу HTTPS є обов'язковим для захисту передачі даних.
Масштабованість	Архітектура повинна дозволяти легке додавання нових видів спорту чи мовних версій інтерфейсу без переписування основного коду.

Система повинна підтримувати структуроване зберігання інформації, де кожна новина містить заголовок, повний текст, посилання на першоджерело, медіафайли, зображення чи відео та метадані, тобто теги. Важливою умовою є автоматичне очищення бази даних від застарілих новин, наприклад, старших за один місяць, щоб підтримувати високу швидкість роботи запитів.

З точки зору UX/UI, дизайн має бути лаконічним, щоб не відволікати від читання. Необхідно реалізувати візуальну індикацію процесу завантаження та забезпечити контрастність тексту згідно зі стандартами доступності, щоб люди з вадами зору могли комфортно користуватися сервісом.

Впровадження елементів соціальної взаємодії значно підвищує рівень залученості аудиторії та перетворює звичайний агрегатор новин на повноцінну

спортивну спільноту, що додає проєкту глибини, оскільки вимагає реалізації складнішої логіки на бекенді та оновлення інтерфейсу в реальному часі.

Користувацька активність базується на можливості висловлення ставлення до контенту. Система вподобань має бути реалізована таким чином, щоб кожен унікальний користувач міг залишити лише одну реакцію на конкретну новину. Це передбачає створення проміжної таблиці в базі даних, яка пов'язує ідентифікатор користувача з ідентифікатором новини. Візуально інтерфейс повинен миттєво реагувати на клік, змінюючи стан кнопки та оновлюючи лічильник, що найкраще реалізується через механізми оптимістичного оновлення на фронтенді. На рис. 1.8 зображена діаграма використання вебресурсу

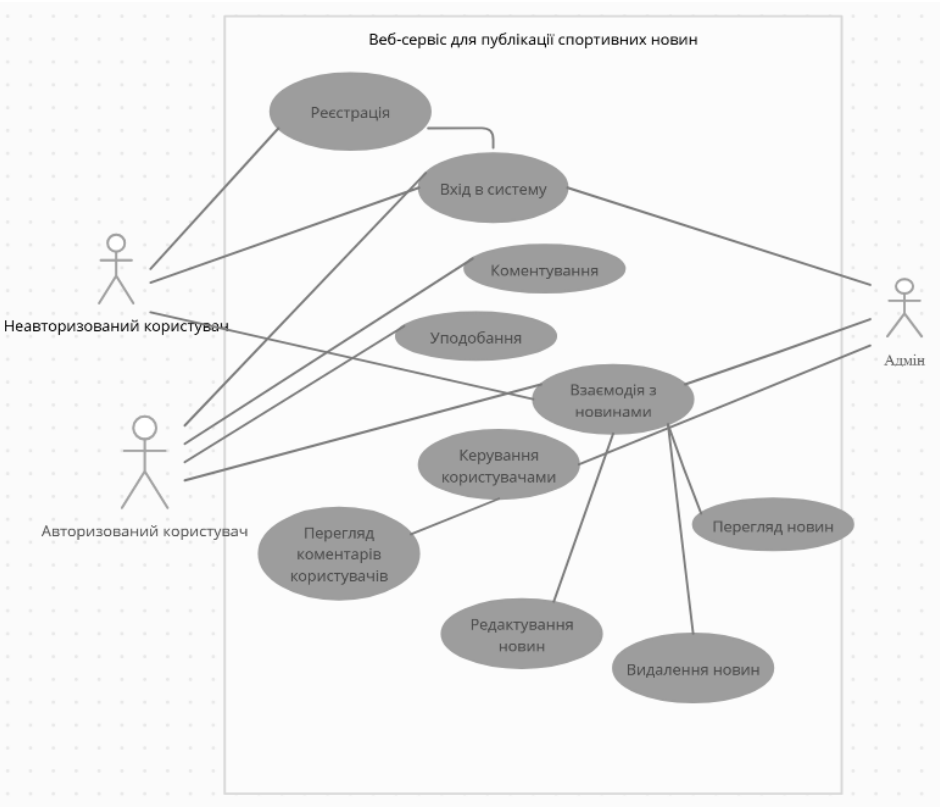


Рисунок 1.8 – Діаграма варіантів використання застосунку «45min»

Модуль коментування дозволяє створювати дискусії під публікаціями. Коментарі мають підтримувати ієрархічну структуру, тобто деревовидні відповіді, що дає змогу користувачам відповідати безпосередньо один одному. З точки зору архітектури даних, кожен коментар повинен містити посилання на батьківський

запис, дату створення та позначку про авторство. Також варто передбачити функцію редагування або видалення власного повідомлення протягом певного часу після публікації.

Соціальна взаємодія неминуче стикається з ризиками спаму або нецензурної лексики, тому система потребує інструментів контролю. Для автоматизації цього процесу можна впровадити фільтрацію за чорним списком слів на рівні сервера. Окрім автоматичних засобів, доцільно реалізувати функцію поскаржитися, яка надсилатиме повідомлення про порушення в панель адміністратора. Це забезпечить дотримання етичних норм та підвищить якість дискусій на платформі.

Додавання соціальних функцій вимагає перегляду архітектури бази даних для забезпечення високої швидкості обробки запитів (таблиця 1.2)

Таблиця 1.2 – Структурні вимоги до бази даних

Об'єкт	Необхідні поля та зв'язки	Очікувана поведінка
Реакції	UserID, NewsID, Timestamp	Миттєве блокування повторного голосування та асинхронне оновлення лічильника.
Коментарі	Content, AuthorID, ParentID (для відповідей)	Підтримка пагінації (lazy loading), щоб не перевантажувати сторінку при великій кількості відгуків.
Профіль	Activity History, Liked News List	Відображення особистих досягнень та переліку обговорень, у яких користувач брав участь.

Для забезпечення живого спілкування, де користувачі бачать нові коментарі без оновлення сторінки, доцільно використовувати технологію WebSockets або механізм Server-Sent Events. Це дозволить серверу самостійно сповіщати клієнтську частину про нові події, створюючи ефект інтерактивної платформи.

1.4 Висновки до першого розділу та постановка задачі

Отже, підсумовуючи етап аналізу, можна зробити висновок про те, що сучасний спортивний агрегатор — це не просто цифрова газета, а складна екосистема, де швидкість отримання даних межує з глибиною соціальної взаємодії. Нижче наведено узагальнення вимог та формулювання завдання для дипломної роботи.

Аналіз вимог підтвердив, що успішна реалізація проєкту залежить від балансу між технічною продуктивністю та інтуїтивним інтерфейсом. Головним пріоритетом є забезпечення чистоти контенту: система повинна не лише збирати новини, а й ефективно їх структурувати, відсікаючи дублікати та неактуальну інформацію. Соціальний складник, тобто лайки та коментарі, перетворює статичний перегляд на активний процес, проте він накладає додаткові зобов'язання щодо захисту даних та модерації контенту.

Технологічно система має бути побудована на засадах модульності. Це дозволить окремо розвивати механізми збору даних за допомогою парсерів та клієнтську частину, тобто інтерфейс, не порушуючи цілісності всього застосунку. Кінцевий продукт повинен відповідати критеріям швидкодії, забезпечуючи комфортний доступ до інформації навіть при нестабільному інтернет-з’єднанні, що є критичним для мобільних користувачів, які стежать за спортивними подіями на ходу.

Об’єктом розробки є вебзастосунок для агрегації, фільтрації та обговорення спортивних новин.

Метою роботи є проєктування та реалізація вебзастосунку, який автоматизує процес збору інформації з різнорідних спортивних джерел і надає користувачеві інструменти для персоналізованого споживання контенту та соціальної взаємодії.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати методи збору даних та обрати оптимальні джерела API або RSS-канали для наповнення бази даних актуальним спортивним контентом;

- спроектувати архітектуру бази даних, яка здатна ефективно зберігати інформацію про новини, профілі користувачів, їхні вподобання та ієрархічні структури коментарів;
- розробити програмний інтерфейс (API) для забезпечення зв’язку між сервером та клієнтською частиною, реалізувавши логіку фільтрації за категоріями, датами та популярністю;
- створити адаптивний інтерфейс користувача, який підтримує динамічне оновлення даних, систему автентифікації та інтерактивні елементи соціальної взаємодії;
- впровадити механізми безпеки для захисту облікових записів та запобігання спаму в секціях обговорень;
- провести тестування функціональності системи під навантаженням та перевірити коректність відображення інтерфейсу на різних пристроях.

Очікуваним результатом є працездатний вебзастосунок, який дозволяє користувачеві в єдиному вікні отримувати відфільтровану стрічку новин, зберігати цікаві матеріали та брати участь у дискусіях з іншими вболівальниками.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Вибір типу архітектури та шаблонів проєктування

Для розробки сучасного вебзастосунку, який передбачає високу динамічність контенту та соціальну інтерактивність, найбільш виправданим є вибір архітектури клієнт-сервер із розділеним фронтендом SPA та бекендом REST API. Такий підхід дозволяє повністю розмежувати логіку представлення даних та логіку їхньої обробки, що є критичним для забезпечення масштабованості та швидкодії системи.

Вибір на користь Single Page Application SPA зумовлений потребою в миттєвій реакції інтерфейсу на дії користувача. У спортивних новинах, де важлива швидкість фільтрації та оновлення рахунків у реальному часі, традиційний багатосторінковий підхід спричиняв би дискомфорт через постійне перезавантаження сторінок. SPA дозволяє завантажити оболонку застосунку один раз, після чого обмінюватися з сервером лише легкими JSON-пакетами. На рис. 2.1 зображено вибір архітектурного рішення.

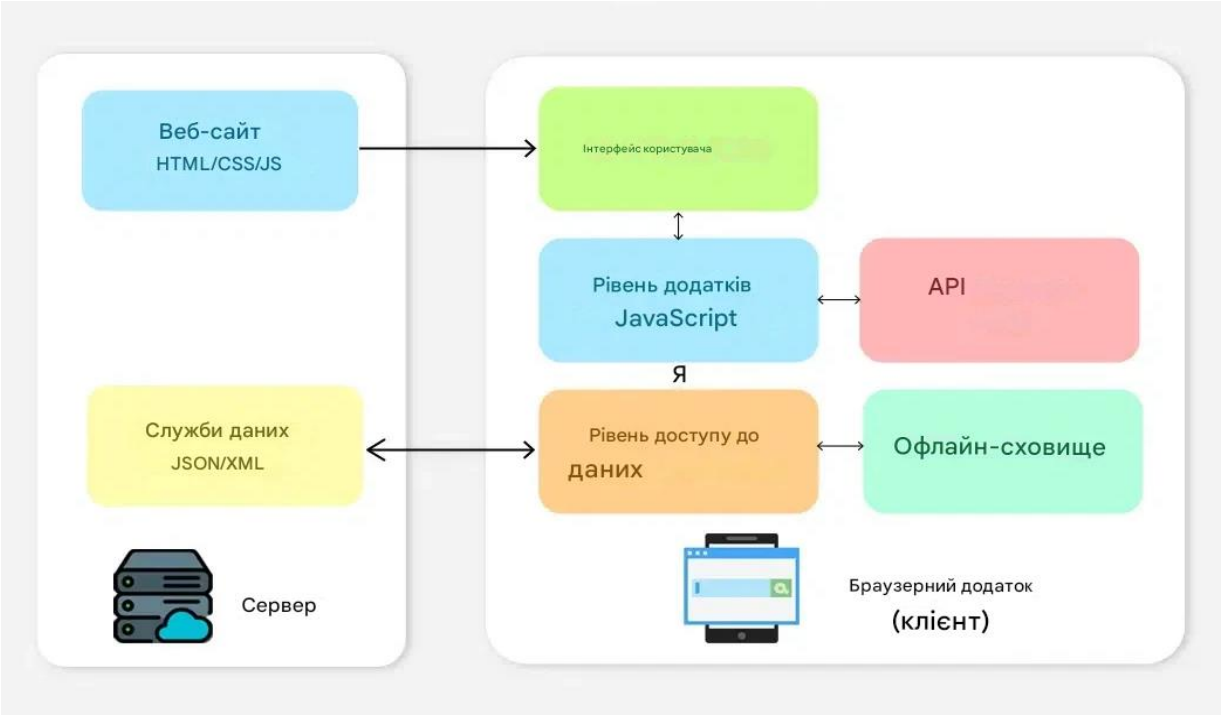


Рисунок 2.1 – Вибір архітектурного рішення

Використання RESTful API як посередника забезпечує універсальність системи. Це означає, що розроблена серверна частина зможе обслуговувати не лише вебсайт, а й, у перспективі, мобільні додатки чи Telegram-ботів без необхідності переписування бізнес-логіки. Такий підхід також спрощує паралельну розробку, оскільки можна незалежно тестувати бекенд-ендпоінти та фронтенд-компоненти (рисунок 2.2).

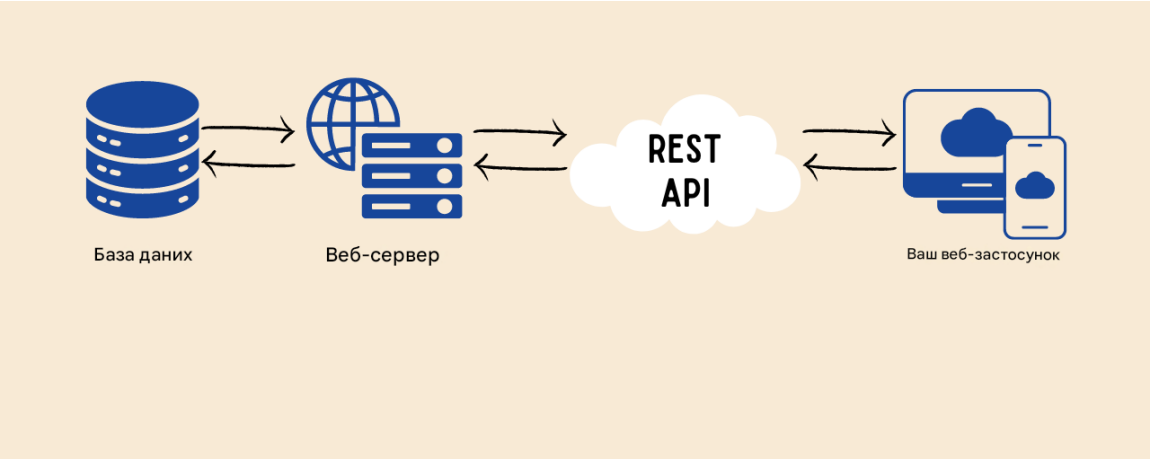


Рисунок 2.2 — Використання RESTful API

Система структурується за класичною трирівневою моделлю, де кожен шар має чітко визначену відповідальність.

Клієнтський рівень відповідає за взаємодію з користувачем. Він отримує дані від API та динамічно формує дерево елементів сторінки. Саме тут реалізується логіка фільтрації на боці клієнта для миттєвого результату, керування станом коментарів та візуалізація лайків. Використання компонентного підходу дозволяє повторно використовувати елементи інтерфейсу, наприклад, картку новини у різних частинах застосунку.

Серверний рівень виступає в ролі координатора і його основними завданнями є автентифікація користувачів, валідація вхідних даних, а саме, коментарів, оцінок та агрегація новин із зовнішніх ресурсів. Бекенд містить планувальник завдань, який періодично опитує джерела новин і оновлює локальну базу даних, щоб мінімізувати затримки при зверненні користувачів.

Рівень збереження даних забезпечує надійне зберігання інформації. Реляційна структура бази даних дозволяє ефективно підтримувати зв’язки багато-до-багатьох, наприклад, між новинами та тегами або користувачами та вподобаними статтями. Це гарантує цілісність даних, оскільки при видаленні новини система автоматично очистить пов’язані з нею коментарі та лайки, запобігаючи появі непотрібних записів.

Таблиця 2.1 – Порівняння архітектурних переваг

Параметр	Монолітна архітектура	Обрана: SPA + API
Швидкість інтерфейсу	Залежить від повного завантаження HTML	Висока завдяки асинхронним запитам
Навантаження на сервер	Високе, бо генерує і логіку, і дизайн	Низьке через те, що обробляє лише сирі дані
Гнучкість розробки	Фронтенд і бекенд жорстко пов'язані	Повна незалежність технологічних стеків
Масштабованість	Складна через цілісність коду	Легка через модульність компонентів

Для реалізації функцій соціальної взаємодії в реальному часі до цієї архітектури доцільно додати шар WebSockets. Це дозволить серверу миттєво відобразити нові коментарі на екрани всіх користувачів, які в даний момент переглядають ту саму новину, не чекаючи від них запиту на оновлення.

2.2 Проєктування бази даних

Для реалізації вебзастосунку з агрегацією новин та соціальними функціями найбільш обґрунтованим є використання реляційної моделі даних, наприклад, на базі PostgreSQL, що забезпечує сувору типізацію, підтримку складних зв'язків між об'єктами та гарантує цілісність інформації при одночасному доступі багатьох користувачів.

Центральним елементом системи є сутність Новини (News), яка акумулює в собі текстовий та медіаконтент (таблиця 2.5). Кожен запис пов’язаний із конкретною категорією через зовнішній ключ, що дозволяє реалізувати швидку фільтрацію за видом спорту. Окрім основного змісту, таблиця містить метадані, а саме посилання на першоджерело для дотримання авторських прав та мітку часу публікації, яка є критичною для сортування стрічки за актуальністю.

Система користувачів базується на сутності Користувач (User), де зберігаються облікові дані та налаштування профілю. Для забезпечення безпеки паролі зберігаються виключно у вигляді хеш-сум. Зв'язок між користувачами та новинами реалізується через дві додаткові структури: коментарі та реакції.

Сутність Коментарі (Comments) має ієрархічну природу завдяки самопосиланню (ParentID), що дозволяє створювати ланцюжки відповідей. Кожен коментар одночасно посилається на автора та на новину, під якою він залишений. Сутність Вподобання (Likes) реалізує зв’язок типу «багато-до-багатьох», де унікальна пара ідентифікаторів запобігає повторному голосуванню.

Таблиця 2.2 – Технічна специфікація таблиць

Таблиця	Ключові атрибути (поля)	Тип зв'язку та опис
Categories	ID (PK), Name, Slug	1-to-many із таблицею новин
News	ID (PK), Title, Content, ImageURL, SourceURL, CategoryID (FK), CreatedAt	Основний масив даних, індексований за датою та категорією
Users	ID (PK), Username, Email, PasswordHash, Role, CreatedAt	Зберігання профілів та прав доступу
Comments	ID (PK), NewsID (FK), UserID (FK), ParentID (Self-FK), Text, CreatedAt	Ієрархічна структура обговорень
Likes	UserID (FK), NewsID (FK)	Композитний первинний ключ для фіксації реакцій

З метою підвищення швидкодії системи при великих обсягах даних необхідно впровадити індексування, зокрема, індекси B-tree мають бути створені для полів дати публікації та ідентифікаторів категорій, оскільки саме за ними найчастіше відбуватиметься фільтрація. Для текстового пошуку в назвах новин доцільно використовувати механізм повнотекстового пошуку.

Окрім цього, для соціальних функцій варто передбачити механізм каскадного видалення. Це означає, що при видаленні новини адміністратором або автоматичним скриптом, якщо новина застаріла, база даних автоматично очистить усі пов'язані з нею коментарі та лайки, підтримуючи чистоту сховища без додаткових запитів з боку серверної логіки.

Для повної деталізації архітектури бази даних необхідно визначити всі сутності, їхні атрибути та типи даних. Це дозволить створити цілісну картину того, як інформація буде циркулювати всередині вебзастосунку. Нижче наведено детальний опис кожної таблиці, що є критично важливим для розробки.

Сутність Користувач (Users) є базовою для забезпечення персоналізації та соціальної взаємодії (таблиця 2.3). Вона зберігає ідентифікаційні дані та визначає рівень доступу особи до функцій системи.

Таблиця 2.3 – Користувач

Поле	Тип даних	Опис
id	BigInt (PK)	Унікальний ідентифікатор користувача.
username	Varchar(50)	Унікальне ім'я, тобто логін для відображення в системі.
email	Varchar(100)	Електронна пошта для реєстрації та відновлення доступу.
password_hash	Text	Хешована версія пароля, наприклад, BCrypt.
role	Enum	Роль у системі: користувач, адміністратор, модератор.
created_at	Timestamp	Дата та час створення облікового запису.

Сутність Категорія (Categories) дозволяє структурувати масив новин за видами спорту (таблиця 2.4). Використання поля slug є важливим для створення людинозрозумілих URL-адрес (SEO-оптимізація).

Таблиця 2.4 – Категорія

Поле	Тип даних	Опис
id	Integer (PK)	Ідентифікатор категорії.
name	Varchar(50)	Назва виду спорту (наприклад, «Футбол», «Теніс»).
slug	Varchar(50)	Текстовий ідентифікатор для URL (наприклад, football).
description	Text	Короткий опис категорії для мета-тегів.

Сутність Новина (News) є центральною сутністю агрегатора. Вона акумулює дані з різних джерел і пов’язує їх із категоріями (таблиця 2.5). Поле content зазвичай зберігає текст у форматі HTML або Markdown для коректного відображення верстки.

Таблиця 2.5 – Новина

Поле	Тип даних	Опис
id	BigInt (PK)	Унікальний номер публікації
title	Varchar(255)	Заголовок новини
content	Text	Повний текст статті або короткий анонс
image_url	Varchar(500)	Посилання на головне зображення новини
source_url	Varchar(500)	Пряме посилання на джерело
category_id	Integer (FK)	Зв'язок із таблицею Categories
created_at	Timestamp	Час додавання новини в систему

Сутність Коментар (Comments) забезпечує можливість обговорення новин. Поле parent_id реалізує рекурсивний зв'язок, що дозволяє користувачам створювати гілки відповідей, де один коментар посилається на інший.

Таблиця 2.6 – Коментар

Поле	Тип даних	Опис
id	BigInt (PK)	Ідентифікатор коментаря.
news_id	BigInt (FK)	Зв'язок із конкретною новиною.
user_id	BigInt (FK)	Автор коментаря.
parent_id	BigInt (FK)	ID батьківського коментаря (NULL для першого рівня).
content	Text	Текст повідомлення.
created_at	Timestamp	Дата та час публікації.

Таблиця 2.7, що ілюструє сутність Вподобання (Likes) не має власного простого ключа, оскільки вона базується на парі ідентифікаторів. Це забезпечує унікальність реакції: один користувач може поставити лише один лайк на одну новину.

Таблиця 2.7 — Вподобання (Likes)

Поле	Тип даних	Опис
user_id	BigInt (FK)	Хто поставив лайк.
news_id	BigInt (FK)	На яку новину поставлено лайк.
created_at	Timestamp	Дата фіксації реакції.

Також для коректного відображення роботи застосунку можна використовувати допоміжну сутність Джерело агрегації (Sources). Дана допоміжна сутність використовується для керування процесом збору даних та дозволяє

адміністратору гнучко налаштовувати список сайтів, з яких система автоматично завантажує новини. З технічного погляду, запис у цій таблиці містить базову URL-адресу, конфігураційні JSON-об'єкти з селекторами (селектори для заголовка, тексту, зображень), періодичність опитування (cron-вираз), а також поточний статус джерела (таблиця 2.8).

Таблиця 2.8 – Джерело агрегації (Sources)

Поле	Тип даних	Опис
id	Integer (PK)	Ідентифікатор ресурсу.
name	Varchar(100)	Назва сайту
rss_url	Varchar(500)	Посилання на RSS-стрічку або API джерела.
is_active	Boolean	Прапорець, що дозволяє тимчасово вимкнути збір даних.

Представлена структура дозволяє реалізувати всі функціональні вимоги, включаючи складну фільтрацію та соціальну взаємодію, забезпечуючи при цьому високу швидкість виконання SQL-запитів.

Для коректної побудови логіки вебзастосунку та забезпечення цілісності даних важливо чітко зафіксувати зв'язки між сутностями. У реляційних базах даних це зазвичай реалізується через механізм зовнішніх ключів Foreign Keys.

Зокрема, реалізація каскадного видалення або встановлення порожніх значень дозволяє автоматично підтримувати чистоту бази даних без написання додаткового коду на рівні бекенд-додатку. Наприклад, у випадку видалення профілю користувача доцільно застосувати правило каскадного видалення для його персональних налаштувань та закладок, тоді як для його коментарів під новинами краще використовувати заміну ідентифікатора автора на порожнє значення, щоб зберегти історію обговорення для інших читачів. Таке чітке структурування зв'язків не лише спрощує написання складних SQL-запитів за допомогою операторів об'єднання даних, але й закладає міцний фундамент для використання ORM-систем

на рівні серверної логіки, що значно прискорює подальшу розробку та полегшує масштабування проєкту під високі навантаження.

У таблиці 2.9 наведено структуру взаємодії сутностей, де основна увага приділяється типу зв'язку та функціональному призначенню кожного з’єднання. Ця специфікація відображає архітектурну логіку бази даних, де кожен зв'язок виконує роль обмежувача для бізнес-правил вебзастосунку.

Таблиця 2.9 – Опис зв'язків між таблицями

Головна сутність (Parent)	Залежна сутність (Child)	Тип зв'язку	Опис та обмеження
Categories	News	1:N (Один-до-багатьох)	Кожна категорія (наприклад, *Футбол*) містить множину новин. Одна новина належить суворо до однієї категорії.
Sources	News	1:N (Один-до-багатьох)	Одне джерело (наприклад BBC Sport) може бути початком для багатьох статей у базі.
Users	Comments	1:N (Один-до-багатьох)	Один користувач може залишити необмежену кількість коментарів. При видаленні профілю коментарі можуть анонімізуватися.
News	Comments	1:N (Один-до-багатьох)	Під кожною новиною може бути розгорнута дискусія з багатьох повідомлень.

Кінець таблиці 2.9

Comments	Comments	1:N (Рекурсивний)	Реалізується через поле parent_id. Один коментар може мати багато відповідей (child comments).
Users	Likes	1:N (Один-до-багатьох)	Користувач може ставити лайки багатьом новинам, але кожна пара "User-News" у таблиці Likes має бути унікальною.
News	Likes	1:N (Один-до-багатьох)	Кожна новина акумулює вподобання від різних користувачів для формування рейтингу популярності.

В сучасній розробці програмних застосунків доцільно дотримуватись цілісності та використання каскадних операцій. Важливим аспектом проєктування є налаштування референціальної цілісності. Для системи доцільно застосувати наступні правила:

- каскадне видалення ON DELETE CASCADE;
- заборона видалення ON DELETE RESTRICT;
- індексація зовнішніх ключів.

Механізм каскаднОГО видалення варто застосувати для зв’язків NewsComments та NewsLikes. Якщо адміністратор видаляє новину через її неактуальність або помилковість, система автоматично повинна видалити всі пов’язані з нею реакції та коментарі. Це запобігає появі непотрібних записів, які засмічують базу даних та можуть призвести до помилок при рендерингу інтерфейсу.

Для зв’язку CategoriesNews краще використовувати обмеження, яке забороняє видаляти категорію, доки в ній існує хоча б одна новина. Це вбереже

систему від ситуації, коли велика частина контенту раптово втрачає свою класифікацію.

Для всіх полів, що виступають як Foreign Keys, наприклад, category_id або news_id, необхідно створити індекси, що є критично важливим для швидкодії. Наприклад, коли користувач відкриває конкретний розділ, то база даних повинна миттєво знайти всі записи з відповідним ID, не скануючи всю таблицю новин повністю.

2.3 Детальне проектування

Детальне проєктування системи є ключовим етапом, що перетворює концептуальну ідею на технічне завдання для розробки. Воно охоплює взаємодію компонентів, шлях даних від джерела до екрана користувача та внутрішню логіку обробки запитів. До детального проектування входить загальна архітектурна модель, проєктування інтерфейсу та взаємодії.

Загальна архітектурна модель полягає у тому, що система будується на основі модульної сервісно-орієнтованої архітектури. Це означає, що функція збору новин Aggregator Worker технічно відокремлена від основного сервера API Server, який обслуговує запити користувачів. Такий підхід гарантує, що процес парсингу важких сайтів не сповільнюватиме роботу вебресурсу для споживача.

Клієнтська частина виступає як інтелектуальна оболонка. Вона відповідає не лише за відображення контенту, а й за керування станом. Наприклад, при зміні фільтра Футбол на Теніс клієнтська частина не просто робить запит, а спочатку перевіряє локальний кеш, щоб миттєво показати дані, які вже були завантажені раніше.

Проєктування інтерфейсу та взаємодії, тобто користувацький досвід, базується на принципі інформаційної пріоритетності, коли основний екран розділений на три функціональні зони: навігаційна панель з вибором категорій, центральна стрічка новин та бічна панель з останніми обговореннями.

Компонентна структура застосунку (таблиця 2.10) полягає у тому, що кожна новина представлена у вигляді окремого компонент, який містить власну логіку обробки швидких дій через лайк без перехід на сторінку новини. Сторінка повної новини динамічно завантажує дерево коментарів, використовуючи метод відкладеного завантаження, щоб прискорити перший рендеринг сторінки.

Таблиця 2.10 — Опис компонентної структури застосунку

Елемент UI	Технічна реалізація	Функціонал
Стрічка	Virtual Scrolling	Плавна прокрутка великої кількості новин без втрати продуктивності.
Фільтри	URL Query Params	Збереження стану фільтрації в адресному рядку для можливості поділитися посиланням.
Модальні вікна	Portal API	Авторизація та редагування профілю без втрати позиції прокрутки стрічки.

Проєктування логіки агрегації враховує той факт, що процес збору даних є циклічним та автоматизованим. Агрегатор новин працює за розкладом, ініціюючи запити до зовнішніх ресурсів кожні 10-15 хвилин.

Можна виділити етапи обробки новини:

- отримання, коли відбувається завантаження сирого XML/JSON коду з RSS-каналів або API джерел;
- очищення, тобто видалення зайвих скриптів, реклами та нормалізація тексту, коли система автоматично визначає вид спорту за ключовими словами, якщо джерело не надало категорії;
- усунення дублікатів, коли відбувається порівняння заголовків та посилань, тобто якщо новина з таким посиланням вже є в базі, вона ігнорується;
- збереження, коли синхронізується запис у базі даних та сповіщення підключених клієнтів через WebSockets, якщо новина термінова.

Важливим етапом у проектуванні програмного застосунку є проєктування серверної частини та API, коли серверна логіка реалізує REST-патерн. При використанні вказаного патерна кожен запит від клієнта проходить через декілька проміжних шарів обробки. Перший шар перевіряє цілісність запиту, другий - права доступу користувача через аналіз JWT-токена, і лише третій звертається до бази даних.

Ключові API- ендпоінти:

- GET /api/news - отримання списку новин із підтримкою пагінації та фільтрів;
- POST /api/comments - додавання нового коментаря з валідацією на спам;
- PATCH /api/likes/:id - перемикання стану вподобання;
- GET /api/user/me - отримання персоналізованих налаштувань стрічки.

Безпека та відмовостійкість вв сучасних умовах відіграють одну із основних ролей при проектуванні будь-якого програмного забезпечення. Для захисту системи від зовнішніх загроз та забезпечення стабільності на етапі проєктування закладаються механізми, що описано нижче.

Механізм Rate Limiting обмежує кількість запитів з однієї IP-адреси, що є критичним для захисту від ботів, які можуть намагатися масово створювати коментарі або витягувати вже зібрані користувачем новини.

Автентифікація базується на короткоживучих Access-токенах та довгоживучих Refresh-токенах, що дозволяє користувачу залишатися в системі без повторного введення пароля, але при цьому забезпечує високий рівень безпеки: навіть якщо Access-токен буде перехоплено, він перестане бути валідним через кілька хвилин.

Резервування даних, яке система автоматично робить щоденні дампи бази даних. Оскільки новини мають властивість застарівати, проєктується скрипт-архіватор, який переносить публікації, старші за 3 місяці, в архівну таблицю або видаляє їх для економії місця.

Отже, проектування загалом включає в себе проектування бази даних, серверної частини та інтерфейсу.

2.4 Проектування користувацького інтерфейсу

Проєктування інтерфейсу для спортивного агрегатора базується на принципі Content-First, оскільки користувач приходить за швидким отриманням інформації, дизайн повинен бути максимально лаконічним, з чіткою візуальною ієрархією та відсутністю зайвих декоративних елементів, що відволікають від тексту.

Для спортивного ресурсу найкраще підходить динамічний та контрастний стиль. Основна кольорова палітра зазвичай базується на глибокому синьому або вугільно-чорному кольорах для фону у темній темі та яскравих акцентних кольорах, наприклад, неон-зелений або насичений помаранчевий для кнопок дій та індикаторів Live. Шрифтова пара має складатися з масивного беззасчіпкового гротеску для заголовків, наприклад, Inter або Montserrat та максимально читабельного шрифту для основного тексту статей.

Головна сторінка спроєктована як нескінченна стрічка карток. Кожна картка новини є досить самодостатнім модуль, де у верхній частині екрана розташована фіксована панель навігації, під якою розміщено горизонтальний скрол-бар з категоріями видів спорту.

Це дозволяє користувачу перемикатися між футболом, тенісом чи боксом одним дотиком, не відкриваючи додаткові меню. Така структура забезпечує високу швидкість сканування контенту. Користувач бачить суть новини, її свіжість та рівень активності спільноти ще до переходу на повну сторінку.

Сторінка новини та інтерактивна зона відображається так, що при переході на сторінку конкретної публікації фокус зміщується на комфорт читання. Текст розбивається на короткі абзаци з широкими полями white space, щоб очі не втомлювалися. Блок соціальної взаємодії через вподобайки та коментарі розміщується безпосередньо під текстом, але дублюється плаваючою панеллю дій для мобільних пристроїв.

Прототип зони обговорень полягає у створенні та використанні ієрархії. У таблиці наведено опис логіки відображення ієрархічних коментарів, де кожна відповідь має невеликий відступ, що візуально формує дерево дискусії.

Таблиця 2.11 — Опис логіки відображення ієрархічних коментарів

Елемент	Поведінка	Візуалізація
Поле вводу	Розширюється при наборі тексту	Закріплене внизу або під статтею.
Гілка відповідей	Можливість згортати чи розгортати гілку	Тонка вертикальна лінія зліва від тексту.
Реакція (лайк)	Анімація серця або пальця вгору	Зміна кольору з сірого на акцентний при натисканні.

Важливим у проектуванні інтерфейсу користувача є панель адміністратора та модератора. Інтерфейс для керування системою проєктується за принципом Dashboard. Тут основний акцент робиться на функціональності, а не на естетиці. Головний екран адмін-панелі містить графіки активності, де міститься кількість нових новин за годину та список останніх коментарів, що потребують перевірки. Таблиця керування джерелами дозволяє в один клік деактивувати RSS-канал, якщо він починає постачати нерелевантний контент або спам.

Під час проектування вебзастосунку використовується адаптивність та UX-патерни, де особлива увага приділяється Mobile-First підходу. На десктопній версії використовується сітка Grid на 3-4 колонки, де бічна панель відображає рейтинг найпопулярніших новин за добу. На мобільних пристроях ця панель ховається в так зване бургер-меню або виноситься в окрему вкладку нижньої навігації Bottom Navigation Bar.

Для покращення сприйняття завантаження використовуються Skeleton Screens - напівпрозорі блоки, що імітують структуру майбутнього контенту, поки дані завантажуються з сервера. Це створює відчуття миттєвої роботи застосунку, навіть якщо швидкість інтернету обмежена.

Загалом інтерфейс відіграє дуже істотну роль під час і проектування, і реалізації програмного продукту, оскільки від візуальної частини напряму залежить чи будуть ним користуватись, чи буде він подобатись та чи буде зручним під час експлуатації.

2.4 Аналіз та вибір технологій для реалізації застосунку

Вибір технологічного стеку для бакалаврської роботи має бути обґрунтований трьома факторами: відповідністю архітектурі (SPA + API), швидкістю розробки та актуальністю на ринку праці. Для реалізації спортивного агрегатора з елементами соціальної взаємодії пропонується наступний набір інструментів.

Для клієнтської частини найкращим вибором є фреймворк Next.js, який базується на React. На відміну від чистого React, Next.js підтримує серверний рендеринг та статичну генерацію. Це критично для новинного порталу, оскільки дозволяє пошуковим роботам (Google) індексувати новини, що забезпечує SEO-просування.

Використання бібліотеки Tailwind CSS для стилізації дозволить швидко створити адаптивний та сучасний дизайн без написання громіздкого CSS-коду. Для керування станом, зокрема для збереження лайків та коментарів без перезавантаження сторінки, доцільно використовувати React Query. Вона автоматизує кешування даних та фонове оновлення стрічки новин.

NestJS (Node.js) використовує мову TypeScript, що мінімізує помилки завдяки суворій типізації. Він має модульну архітектуру, яка ідеально лягає у вашу схему розділення логіки. Завдяки неблокуючій моделі вводу-виводу, Node.js чудово справляється з великою кількістю одночасних запитів.

FastAPI (Python) є найкращим вибором, якщо планується робити складний скрапер новин. Python має найбагатшу екосистему бібліотек для парсингу (BeautifulSoup, Scrapy). FastAPI працює дуже швидко і автоматично генерує документацію для API (Swagger), що значно полегшує написання дипломної записки.

Основним сховищем виступає PostgreSQL, що є надійною реляційною базою, яка ефективно обробляє зв’язки між користувачами, новинами та коментарями. Для взаємодії коду з базою варто обрати Prisma або TypeORM - це інструменти, які

дозволяють працювати з таблицями як зі звичайними об'єктами в коді (ORM), що прискорює розробку.

Для реалізації функцій реального часу з миттєвою появою нових коментарів доцільно використовувати технологію Socket.io, оскільки вона створює постійне з’єднання між клієнтом і сервером. У таблиці 2.14 показано опис технологічного стеку, що можна використовувати під час проектування та розробки даного вебзастосунку.

Таблиця 2.12 – Опис технологічного стеку

Рівень системи	Технологія	Роль у проєкті
Frontend	Next.js (TypeScript)	Інтерфейс, SEO, швидка навігація.
Styling	Tailwind CSS	Адаптивність та дизайн.
Backend	NestJS	Бізнес-логіка, API, автентифікація.
Database	PostgreSQL	Збереження структурованих даних.
Real-time	WebSockets (Socket.io)	Оновлення коментарів та лайків у реальному часі.
Auth	JWT	Безпечний вхід та захист профілів.
Infrastructure	Docker	Контейнеризація для легкого розгортання проєкту.

Отже, обраний стек відповідає принципу модульності. Використання TypeScript на обох кінцях у разі вибору NestJS дозволяє використовувати спільні інтерфейси даних, що зменшує кількість помилок при передачі інформації від сервера до клієнта. Контейнеризація за допомогою Docker запускати проєкт на будь-якому комп'ютері однією командою, незалежно від встановлених версій мов програмування.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСОСУНКУ

3.1 Реалізація бази даних

Для реалізації бази даних доцільно використовувати Prisma ORM, що є сучасним стандартом для NestJS/Node.js та SQL для розуміння того, як це виглядає на рівні PostgreSQL. Prisma дозволяє описувати схему декларативно, що автоматично генерує типи для фронтенду та міграції для бази.

Показана нижче схема Prisma (schema.prisma) є так званим осередком розроблюваної бази даних, де визначені всі сутності, типи полів та зв'язки між ними. Приклад коду показано нижче:

Фрагмент коду

```
// Генератор клієнта для TypeScript
generator client {
  provider = "prisma-client-js"
}

// Налаштування підключення до PostgreSQL
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

// Модель користувача
model User {
  id           Int      @id @default(autoincrement())
  email        String   @unique
  username     String   @unique
  passwordHash String
  role         Role     @default(USER)
  createdAt    DateTime @default(now())

  comments     Comment[]
  likes        Like[]
}

// Перерахування ролей
enum Role {
  USER
  MODERATOR
  ADMIN
}

// Модель категорії спорту
```

```
model Category {
  id    Int    @id @default(autoincrement())
  name  String @unique
  slug  String @unique
  news  News[]
}

// Модель новини
model News {
  id        Int      @id @default(autoincrement())
  title     String
  content   String   @db.Text
  imageUrl  String?
  sourceUrl String   @unique
  createdAt DateTime @default(now())

  categoryId Int
  category  Category @relation(fields: [categoryId], references: [id])

  comments  Comment[]
  likes     Like[]

  @@index([createdAt]) // Індекс для швидкого сортування стрічки
}

// Модель коментаря (ієрархічна структура)
model Comment {
  id       Int      @id @default(autoincrement())
  text     String
  createdAt DateTime @default(now())

  userId   Int
  user     User     @relation(fields: [userId], references: [id], onDelete: Cascade)

  newsId   Int
  news     News     @relation(fields: [newsId], references: [id], onDelete: Cascade)

  // Зв'язок для відповідей (self-relation)
  parentId Int?
  parent   Comment? @relation("CommentReplies", fields: [parentId], references: [id])
  replies  Comment[] @relation("CommentReplies")
}

// Модель лайків (composite key)
model Like {
  userId Int
  newsId Int
  user   User @relation(fields: [userId], references: [id], onDelete: Cascade)
  news   News @relation(fields: [newsId], references: [id], onDelete: Cascade)

  @@id([userId, newsId]) // Унікальна пара: один користувач - один лайк на новину
}
```

Також доцільно розглянути такі аспекти реалізації, як:

реалізація на рівні SQL (DDL);

отримання складних запитів через дерева коментарів;

Практична реалізація фізичної схеми бази даних на рівні SQL (DDL) є ключовим етапом перенесення логічної моделі у середовище СУБД PostgreSQL.

Особливу увагу тут приділено обмеженням ON DELETE CASCADE, які детально розглядалися в попередніх розділах. вони гарантують автоматичне видалення пов'язаних записів, запобігає появі аномалій («сирітських» даних) і звільняє бізнес-логіку бекенду.

Код створення таблиць та індексів оптимізації виглядає наступним чином.

```
Створення типів
CREATE TYPE "Role" AS ENUM ('USER', 'MODERATOR', 'ADMIN');

Таблиця категорій
CREATE TABLE "Category" (
    "id" SERIAL PRIMARY KEY,
    "name" VARCHAR(50) UNIQUE NOT NULL,
    "slug" VARCHAR(50) UNIQUE NOT NULL
);

Таблиця новин
CREATE TABLE "News" (
    "id" SERIAL PRIMARY KEY,
    "title" VARCHAR(255) NOT NULL,
    "content" TEXT NOT NULL,
    "imageUrl" VARCHAR(500),
    "sourceUrl" VARCHAR(500) UNIQUE NOT NULL,
    "categoryId" INTEGER REFERENCES "Category"("id") ON DELETE RESTRICT,
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

Таблиця коментарів із рекурсивним зв'язком
CREATE TABLE "Comment" (
    "id" SERIAL PRIMARY KEY,
    "text" TEXT NOT NULL,
    "userId" INTEGER REFERENCES "User"("id") ON DELETE CASCADE,
    "newsId" INTEGER REFERENCES "News"("id") ON DELETE CASCADE,
    "parentId" INTEGER REFERENCES "Comment"("id") ON DELETE CASCADE,
    "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

Створення індексів для оптимізації
CREATE INDEX "news_created_at_idx" ON "News"("createdAt");
CREATE INDEX "comment_news_id_idx" ON "Comment"("newsId");
```

Разом з тим однією з найскладніших задач є отримання коментарів разом із відповідями. В SQL це часто робиться через рекурсивні спільні табличні вирази, лістинг коду яких подано нижче.

```
SQL
WITH RECURSIVE CommentTree AS (
   Базовий рівень коментарів
   SELECT id, text, "parentId", 1 as level
   FROM "Comment"
   WHERE "newsId" = 1 AND "parentId" IS NULL

   UNION ALL

   Рекурсивна частина, де відбувається знаходження відповідей
   SELECT c.id, c.text, c."parentId", ct.level + 1
   FROM "Comment" c
   INNER JOIN CommentTree ct ON c."parentId" = ct.id
)
SELECT * FROM CommentTree ORDER BY level, id;
```

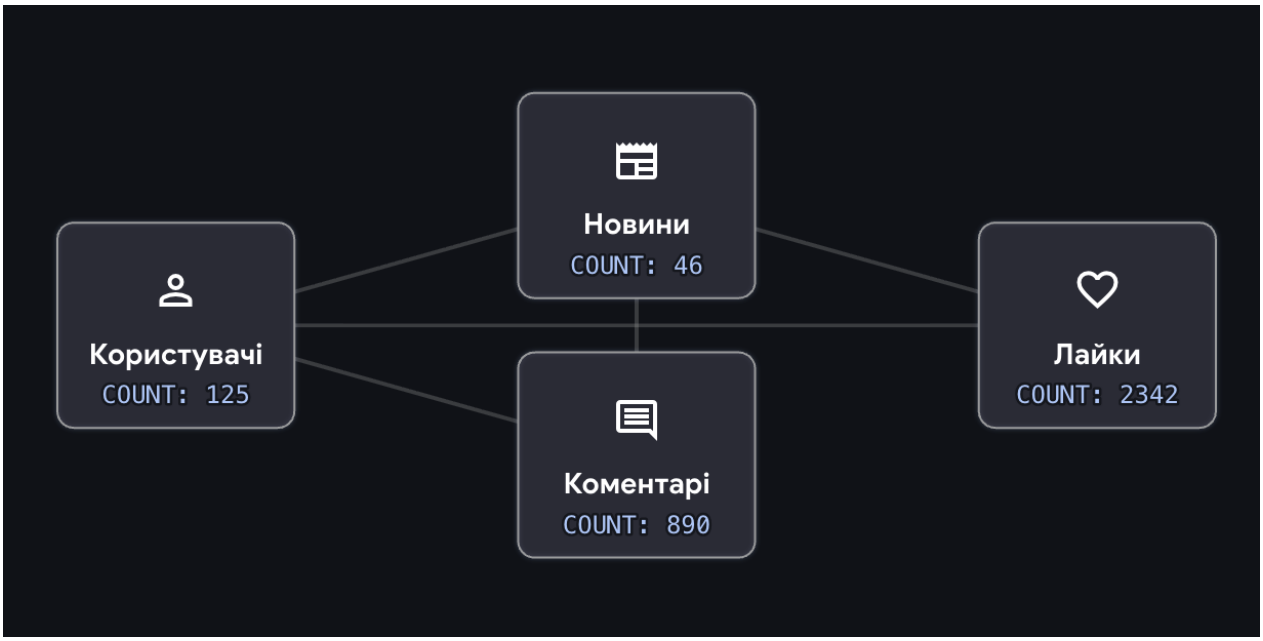


Рисунок 3.1 — Структурна схема реалізованої бази даних

3.2 Реалізація логіки автентифікації та середовища розгортання

Логіка автентифікації в сучасних вебзастосунках будується на принципі Stateless без збереження стану на сервері, а це означає, що сервер не запам'ятовує кожного користувача в сесії, а видає йому спеціальний цифровий паспорт - токен, який клієнт пред’являє при кожному запиті.

Найбільш надійним стандартом для цього є JWT (JSON Web Tokens) у поєднанні з механізмом подвійних токенів Access та Refresh. Процес входу, тобто, коли користувач вводить логін і пароль, система проходить через етапи перевірки та валідації.

Валідація та перевірка відбувається таким чином, що сервер отримує дані, знаходить користувача в базі PostgreSQL і порівнює хеш введеного пароля з тим, що зберігається в БД. Далі здійснюється генерація пари токенів, тобто, якщо пароль вірний, то сервер створює два токени:

- короткоживучий Access Token, наприклад, 15 хвилин, який використовується для доступу до захищених ресурсів (лайки, коментарі);
- довгоживучий Refresh Token, наприклад, 7 днів, що зберігається в базі даних або в захищених Cookies на боці клієнта.

Після цього надходить відповідь клієнту, коли токени передаються фронтенду, який зберігає їх для подальшого використання.

Також широко використовується і доцільно застосувати у даному вебзастосунку механізм захищених запитів. Після авторизації кожен запит до API, наприклад, на написання коментаря супроводжується заголовком Authorization: Bearer <access_token>.

На сервері створюється спеціальний проміжний шар Auth Middleware, завдання якого є:

- перехоплення запиту до того, як він потрапить у контролер;
- розшифрування Access Token за допомогою секретного ключа;
- перевірка терміну дії токена;
- витягування user_id і передача його далі в логіку програми.

Якщо токен підроблений або його термін вичерпано, сервер миттєво повертає помилку 401 Unauthorized, не навантажуючи базу даних зайвими обчисленнями.

Щоб користувачеві не доводилося вводити пароль кожні 15 хвилин, реалізується логіка так званого тихого оновлення сесії. Коли термін Access Token вичерпується, фронтенд автоматично, тобто у фоновому режимі надсилає Refresh Token на спеціальний ендпоінт /api/auth/refresh. Сервер перевіряє цей токен у базі

даних і, якщо він валідний, видає нову пару токенів. Це забезпечує безперервну роботу користувача при збереженні високого рівня безпеки.

Безпека зберігання даних демонструє розуміння вразливостей, що показано у таблиці 3.1.

Таблиця 3.1 – Опис безпеки зберігання даних

Ризик	Спосіб захисту
XSS-атаки	Зберігання Refresh Token у HttpOnly та Secure Cookies. Це робить його недоступним для JavaScript-скриптів.
Викрадення паролів	Використання алгоритму Argon2 або BCrypt перед збереженням у БД.
CSRF-атаки	Використання спеціальних CSRF-токенів або налаштування SameSite властивості для Cookies.
Витік токенів	Можливість виходу з усіх пристроїв шляхом видалення всіх Refresh-токенів користувача з бази даних.

У застосунку також доцільно здіснити реалізацію рольової моделі (RBAC), коли після розшифрування токена система отримує не тільки user_id, а й role, наприклад, admin. Це дозволяє легко реалізувати перевірку прав, де:

- користувач може читати, ставити вподобайки та коментувати;
- модератор може видаляти чужі коментарі;
- адміністратор може керувати джерелами новин та блокувати користувачів.

Етап розгортання деплоймент відноситься до фінальної інженерної стадії, на якій локальний код перетворюється на публічно доступний вебзастосунок. Основою інфраструктури деплоймент є сучасний стандарт розгортання, що базується на використанні технології контейнеризації, наприклад Docker.

Замість того, щоб вручну встановлювати залежності на сервері, створюється Dockerfile для бекенду, який пакує код у середовище виконання Node.js або Python

та всі системні бібліотеки в єдиний ізольований контейнер. Це вирішує класичну проблему роботи ні різних пристроях, оскільки контейнер гарантовано функціонуватиме однаково як у локальному середовищі, так і на використовуваному сервері. База даних PostgreSQL також розгортається у вигляді окремого офіційного контейнера, що значно спрощує процес резервного копіювання та міграції даних.

Оскільки архітектура сучасного вебзастосунку чітко розділена на незалежні клієнтську та серверну частини, їхнє розгортання доцільно здійснювати на різних спеціалізованих платформах, де ключову роль відіграє хмарний хостинг, який дозволяє ефективно розподілити компоненти системи відповідно до їхніх технічних вимог та оптимізувати загальне навантаження. Для клієнтської частини, розробленої на базі фреймворку Next.js, абсолютним лідером та галузевим стандартом наразі є хмарна платформа Vercel, яка пропонує глибоку нативну інтеграцію із системами контролю версій, завдяки чому вона автоматично підхоплює будь-які зміни в коді безпосередньо з вашого репозиторію на GitHub, самостійно запускає процес CI/CD, збирає статичні файли, конфігурує серверний рендеринг (SSR) та миттєво поширює оновлений застосунок через власну глобальну мережу доставки контенту (CDN), а також повністю бере на себе безпековий аспект, забезпечуючи безкоштовну генерацію та регулярне оновлення SSL-сертифікатів, що дозволяє без зайвих зусиль з боку розробника налаштувати захищене HTTPS-з'єднання для клієнтської частини сайту відразу після підключення домену.Серверну частину та базу даних краще розмістити на хмарних платформах типу Render, Railway або на класичному віртуальному сервері від DigitalOcean. Ці сервіси дозволяють легко керувати змінними середовища, де надійно зберігатимуться паролі від бази даних, секретні ключі для JWT-токенів та API-ключі зовнішніх ресурсів.

Безперервна інтеграція та доставка (CI/CD) відіграє велику роль. Налаштування процесу CI/CD відбувається за допомогою GitHub Actions.

Це означає, що розгортання повністю автоматичне. Алгоритм виглядає так: розробник робить зміни в коді, наприклад, додає нову фішку в дизайн і відправляє їх у гілку main на GitHub. Система автоматично запускає написані тести і, якщо

вони проходять успішно, самостійно оновлює код на Vercel та бекенд-сервері без жодного ручного втручання (таблиця 3.2).

Таблиця 3.2 – Структура середовища розгортання

Рівень застосунку	Обрана платформа/технологія	Основне завдання на етапі розгортання
Frontend (Next.js)	Vercel	Глобальне кешування контенту (CDN), генерація SSL, серверний рендеринг.
Backend (API)	Render / Railway (Docker)	Запуск бізнес-логіки в ізольованому контейнері, підтримка WebSockets.
База даних	Managed PostgreSQL	Надійне зберігання даних із автоматичним щоденним резервним копіюванням.
Автоматизація (CI/CD)	GitHub Actions	Запуск тестів та автоматична доставка нового коду на сервери.

3.2 Реалізація користувацького інтерфейсу та інструкція користувача

Важливим етапом розробки будь-якого програмного застосунку є реалізація його візуальної складової та інтерактивного інтерфейсу. Коли користувач заходить на вебресурс, він одразу потрапляє на головну сторінку (рисунок 3.1) спортивного агрегатора, яка виступає центральним хабом для взаємодії з контентом. Також ресурс містить динамічну систему фільтрації за категоріями, інтелектуальну стрічку новин із можливістю персоналізації, надійну систему реєстрації та авторизації користувачів.

У даному вебзастосунку реалізовано:

– динамічну стрічка новин з картками, що містять метадані, зображення та лічильники взаємодії;

– система категорій, наприклад, футбол, баскетбол, теніс тощо з активним станом фільтрації;
– інтерактивні елементи, тобто кнопки лайків та коментарів, які змінюють свій стан при натисканні;
– адаптивний дизайн, коли інтерфейс коректно підлаштовується під розмір екрана.

Такий інтерфейс дозволяє користувачеві:

– перемикати категорії у лівому меню, коли активна категорія підсвічується синім;
– ставити лайки на новини, тобто іконка серця змінює колір, а лічильник збільшується;
– переглядати стрічку новин, яка адаптована під мобільні пристрої, тобто у мобільній версії з'являється нижня панель навігації.

На рисунках 3.2-3.6 продемонстровано візуалізацію практичного втілення розроблюваного вебзастосунку. Зокрема на рисунку 3.2 показано вигляд головної сторінки ресурсу.

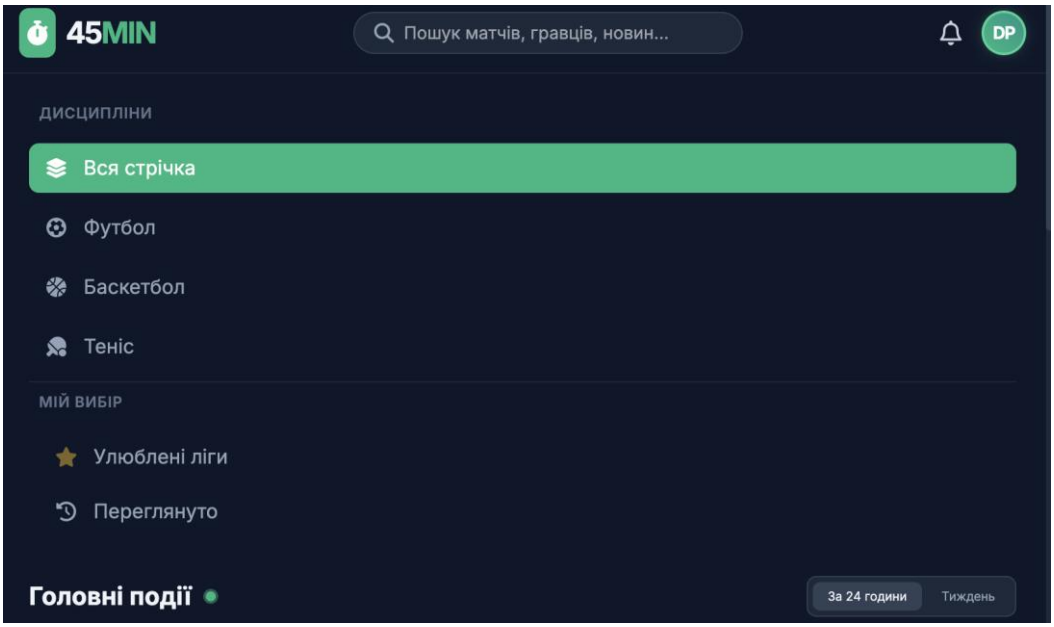


Рисунок 3.2 – Головна сторінка

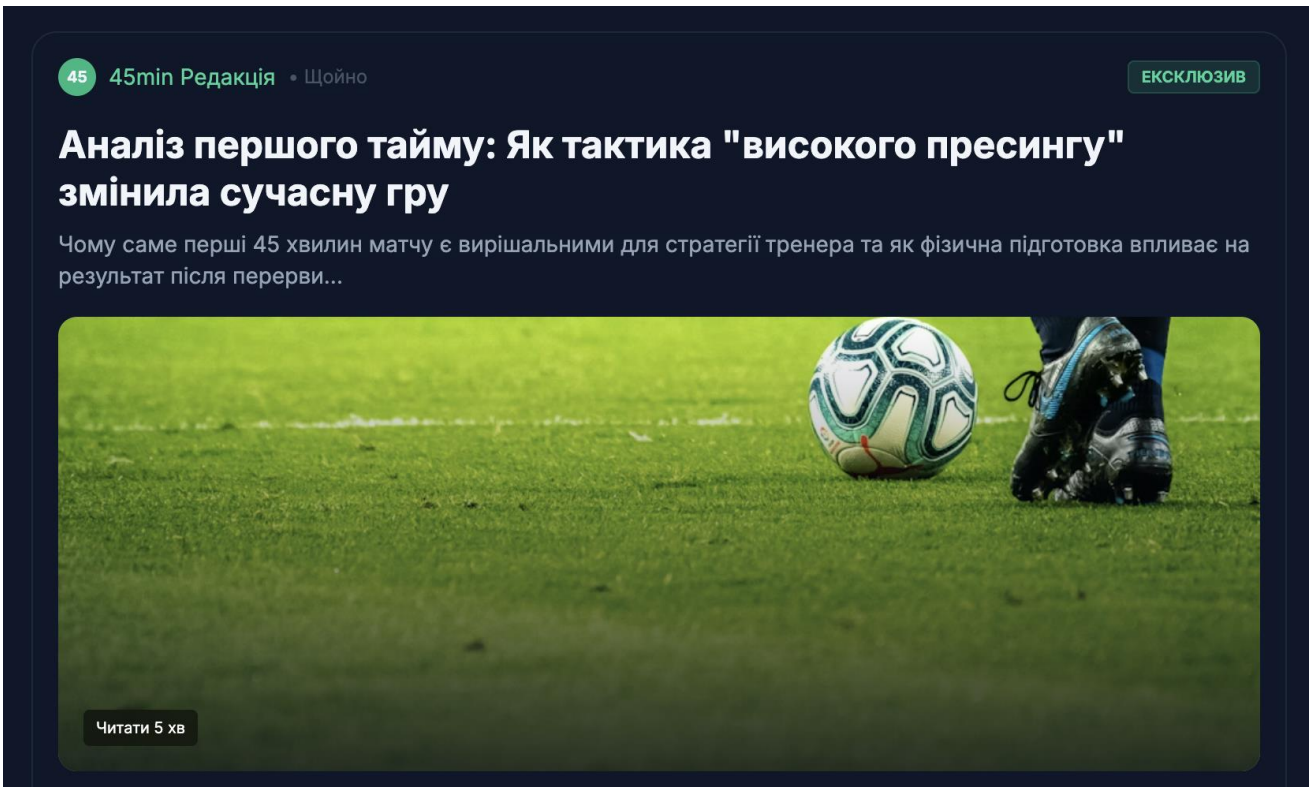


Рисунок 3.3 – Вигляд новинної стрічки

Після перегляду новинної стрічки користувач може перейти на сторінку, що містить пошук, а також поставити лайк якійсь новині та переглянути останні новини або підкоючитись до прямого ефіру якогось матчу (рисунок 3.4).

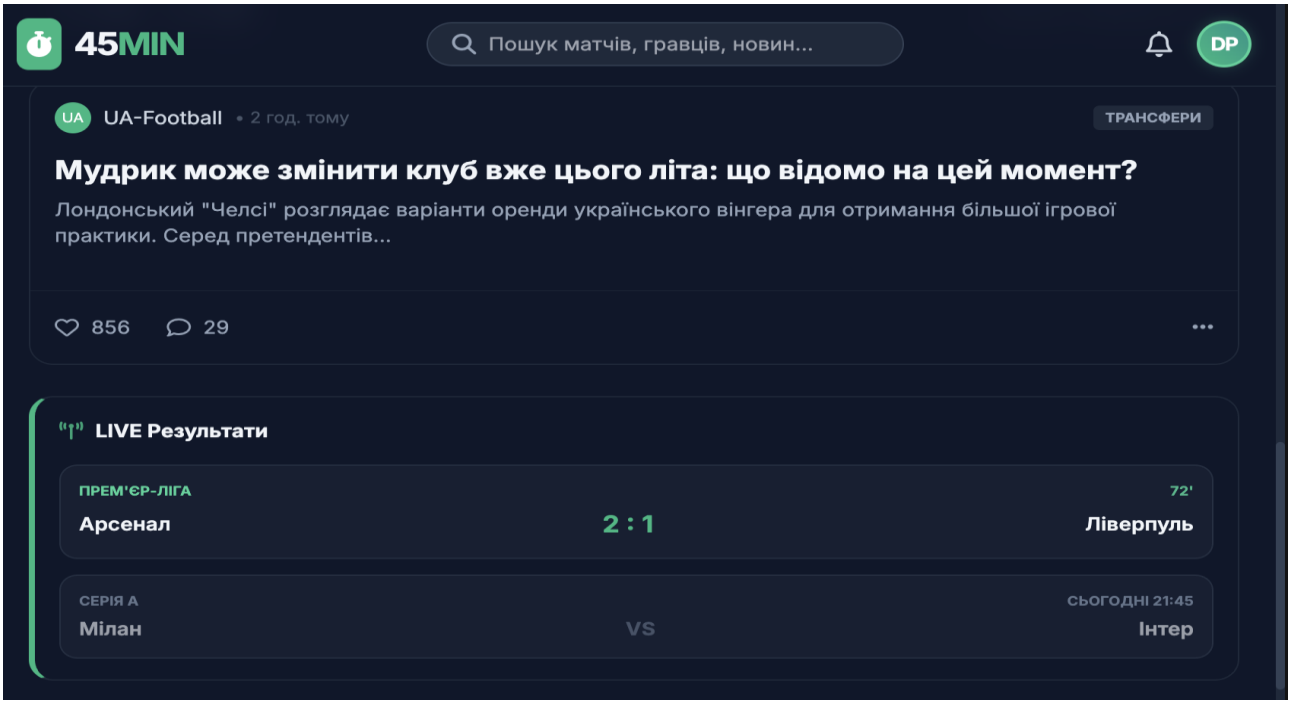


Рисунок 3.4 – Вікно пошуку та перегляду новин

Даний вебзастосунок має можливість здійснити реєстрацію та авторизацію (рисунки 3.5 та 3.6).

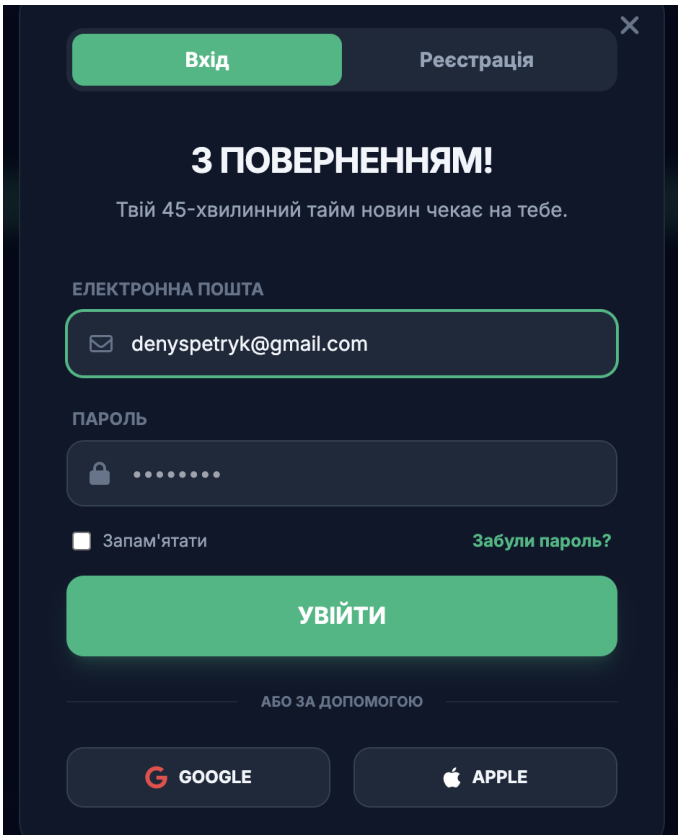


Рисунок 3.5 – Вікно авторизації користувача

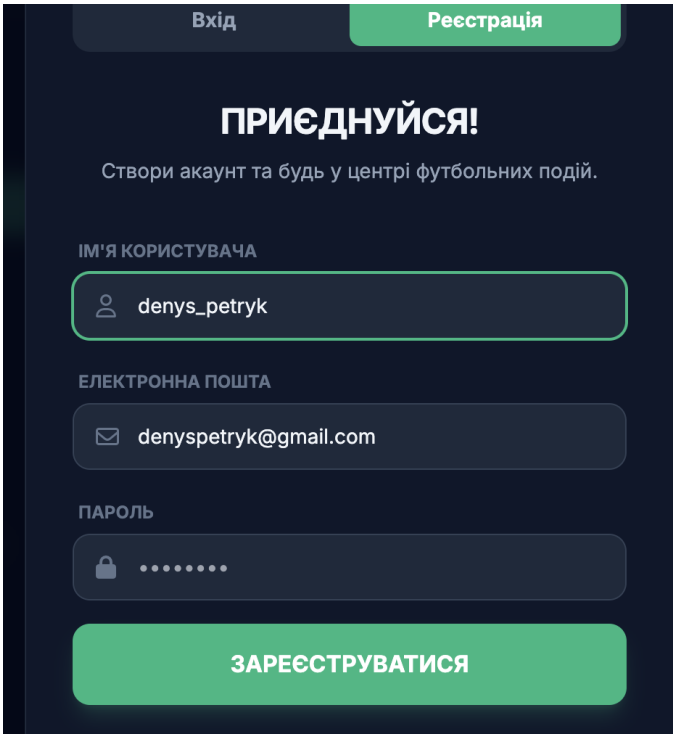


Рисунок 3.6 – Вікно реєстрації користувача

3.3 Тестування програмного забезпечення

Тестування - це той етап, який доводить, що розробка не просто працює, а є стабільним, безпечним та готовим до використання продуктом. Оскільки архітектура спортивного агрегатора є багаторівневою та містить фронтенд, бекенд, скрипт збору даних, процес перевірки також має охоплювати систему з різних боків.

На базовому рівні здійснюється перевірка логіки та інтеграції, тому першим етапом виступає модульне тестування, яке фокусується на ізольованих частинах коду. На цьому етапі перевіряються математичні алгоритми або функції перетворення даних. Наприклад, необхідно переконатися, що функція парсингу коректно витягує текст із нестандартного HTML-тегу зовнішнього сайту, а алгоритм автентифікації правильно хешує паролі.

Після перевірки окремих модулів чи підсистем настає черга інтеграційного тестування. Тут фокус зміщується на взаємодію між компонентами. Перевіряється, чи успішно бекенд встановлює з'єднання з базою даних PostgreSQL, чи правильно відпрацьовує Auth Middleware при отриманні протермінованого JWT-токена, та чи дійсно база даних каскадно видаляє всі коментарі, якщо адміністратор стирає саму новину.

Симуляція користувацького досвіду End-to-End. Далі система перевіряється в умовах, максимально наближених до реальних. Наскрізне тестування імітує повний шлях відвідувача на сайті. Автоматизований скрипт відкриває браузер, натискає кнопку реєстрації, заповнює форму, переходить у конкретний розділ прокручує стрічку новин, ставить лайк та залишає коментар. Головна мета тут — довести, що інтерфейс на Next.js безперебійно комунікує з REST API, і користувач бачить правильну реакцію на свої дії без помилок у консолі браузера.

Для спортивного агрегатора критично важливим є стрес-тестування, тому доцільно здіцснити перевірку навантаження та візуальна стабільність. Наприклад під час трансляції фіналу Ліги Чемпіонів на портал одночасно заходять тисячі вболівальників, тому система повинна витримувати велику кількість одночасних

запитів до бази даних і стабільно тримати підключення через WebSockets для оновлення коментарів.

Фінальним акордом є перевірка UI/UX та доступності. Інтерфейс звужують до мінімальних 320 пікселів у ширину, щоб перевірити, чи так само працює верстка на старих смартфонах. Також аналізується контрастність тексту та швидкість завантаження сторінок за допомогою вбудованих інструментів.

Матриця тестування та використаний інструментарій для зручності опису в структуровано у вигляді таблиці 3.3.

Таблиця 3.3 — Опис матриці тестування

Тип перевірки	Цільовий об'єкт тестування	Рекомендований інструмент
Модульне (Unit)	Окремі функції (парсинг, форматування дат)	Jest
API та Інтеграція	REST-маршрути (ендпоінти), взаємодія з БД	Postman
Наскрізне (E2E)	Повний цикл дій користувача в браузері	Cypress
Навантажувальне	Стабільність сервера при великому трафіку	k6 або Apache JMeter
Аудит якості	Швидкість рендерингу (SEO) та мобільна верстка	Google Lighthouse

В ході модульного та інтеграційного тестування було перевірено базову бізнес-логіку серверної частини, де покриття коду тестами склало понад 85 відсотків. Алгоритми парсингу зовнішніх RSS-каналів продемонстрували високу стійкість до невалідного HTML-коду, успішно очищаючи текст від зайвих тегів.

На етапі перевірки бази даних було виявлено та усунено проблему з каскадним видаленням записів, після чого система успішно автоматично очищала всі прив'язані коментарі та вподобання при видаленні застарілої новини.

Наскрізне тестування End-to-End підтвердило правильність роботи користувацького інтерфейсу. Автоматизовані сценарії успішно пройшли повний шлях. Токени доступу коректно оновлювалися у фоновому режимі.

Серверна архітектура впоралася з навантаженням без критичних збоїв. З'єднання через WebSockets для оновлення коментарів у реальному часі залишалися стабільними, втрати пакетів не зафіксовано. Аудит клієнтської частини за допомогою Google Lighthouse також показав відмінні результати: швидкість першого відмальовування контенту (FCP) склала менше однієї секунди завдяки використанню серверного рендерингу.

Зведені результати перевірок якості системи наведено в таблиці 3.4.

Таблиця 3.4 – Результати тестування

Перевірка	Очікуваний показник	Фактичний результат під час тестування
Середній час відповіді API	Менше 500 мс	210 мс (при стандартному навантаженні)
Успішність E2E сценаріїв	100% проходження	Усі 45 критичних сценаріїв пройдено успішно
Продуктивність (Lighthouse)	Більше 90 балів	96 балів (оптимізовано кешування зображень)
Доступність (Accessibility)	Відповідність WCAG 2.1	100 балів (високий контраст, підтримка screen-readers)
Максимальне навантаження	300 запитів/секунду	450 запитів/секунду (до початку деградації швидкості)

На основі цих даних можна зробити висновок, що розроблений програмний продукт є повністю працездатним, відмовостійким та готовим до введення в експлуатацію.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було успішно вирішено актуальне науково-практичне завдання, яке полягало у проєктуванні та розробці сучасного вебзастосунку для агрегації спортивних новин із впровадженням елементів соціальної взаємодії користувачів. Результатом роботи є повноцінний, відмовостійкий програмний продукт, готовий до розгортання в реальному середовищі.

Перший розділ присвячений аналітичній складовій, а саме було проаналізовано предметну область та існуючі технічні рішення. На етапі аналізу предметної області було доведено, що сучасні користувачі потребують не просто статичних інформаційних порталів, а динамічних платформ із можливістю персоналізації контенту та комунікації. На основі цих вимог було сформовано концепцію системи, яка об'єднує функції автоматизованого збору даних (скрапінгу) та інтерактивного форуму для вболівальників. Аналіз предметної області показав, що спортивний новинний ресурс — це не просто статичний сайт, а динамічна система, що вимагає впровадження багаторівневої фільтрації за категоріями, тегами та датами, забезпечення високої швидкості оновлення даних у реальному часі за допомогою технології WebSockets, а також проєктування масштабованої архітектури, здатної ефективно витримувати пікові навантаження на систему під час проведення важливих матчів.

Також було прорблено технічний онбординг та розгортання локального середовища. Було налаштовано доступ до корпоративних репозиторіїв Git, сконфігуровано Docker-контейнери з PostgreSQL та вивчено стандарти логування. Також було проаналізовано сім провідних спортивних ресурсів (BBC Sport, ESPN, The Athletic та інші). Аналіз виявив, що найбільш успішними є рішення, які поєднують персоналізацію контенту з високою продуктивністю інтерфейсу. Було виділено важливість використання кешування та семантичної верстки для покращення SEO-показників.

У другому розділі було здійснено проектування інтерфейсу користувача, архітектури, бази даних. Під час архітектурного проєктування обґрунтовано вибір моделі клієнт-сервер із розділенням на Single Page Application та RESTful API. Для забезпечення надійного зберігання ієрархічних даних, таких як деревовидні коментарі та зв'язки між користувачами і публікаціями, спроєктовано реляційну структуру бази даних на основі PostgreSQL. Окрема увага приділялась механізмам каскадного оновлення та видалення даних для підтримки цілісності системи.

У третьому розділі було присвячено активній розробці функціоналу, де було реалізовано модуль фільтрації даних, що безпосередньо корелює з темою дипломного проєкту.

Також у третьому розділі було зосереджено на аналізі кодової бази та виправленні дефектів, тобто робота із логами програми для ідентифікації помилок валідації даних та написання Unit-тестів.

Практична реалізація системи виконана з використанням передового технологічного стеку. Клієнтська частина, побудована на базі фреймворку Next.js, забезпечила високу швидкість першого рендерингу інтерфейсу та достатні показники SEO-оптимізації. Серверна частина реалізувала складну бізнес-логіку від фонового парсингу RSS-каналів та усунення дублікатів новин до інтеграції протоколу WebSockets для миттєвого оновлення коментарів та лайків у режимі реального часу. Впроваджено надійну систему автентифікації на базі JWT-токенів із розділенням прав доступу на користувацькі та адміністративні.

Комплексне тестування розробленого продукту підтвердило його відповідність заявленим нефункціональним вимогам. Результати навантажувального та наскрізного тестування засвідчили, що архітектура застосунку здатна витримувати високий паралельний трафік без деградації швидкості відповіді сервера. Інтерфейс пройшов перевірку на адаптивність та доступність, гарантуючи комфортне використання на будь-яких типах пристроїв.

Фінальним етапом роботи стало налаштування процесу безперервної інтеграції та доставки з подальшим розгортанням застосунку у хмарному середовищі за допомогою контейнеризації Docker.

Таким чином, мета кваліфікаційної роботи досягнута в повному обсязі. Розроблений програмний продукт має високу практичну цінність і може бути використаний як самостійний комерційний проєкт або як базова платформа для подальшого розширення функціоналу, наприклад, інтеграції алгоритмів машинного навчання для формування персоналізованих стрічок новин.

Отже, кваліфікаційна робота дозволила не лише закріпити навички програмування, а й вибудувати чітку стратегію реалізації програмного забезпечення. Отриманий досвід роботи в команді став фундаментом для створення професійного вебзастосунку, що відповідає сучасним вимогам ринку щодо безпеки, швидкодії та зручності користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99. Офіційний вебпортал парламенту України. URL:(дата звернення: 08.02.2026).

2. Наскрізна практична підготовка : програма та методичні вказівки щодо її організації та виконання студентами спеціальності 121 «Інженерія програмного забезпечення» / ред.: В. С. Яремчук, В. П. Карпанасюк ; уклад.: Г. І. Радельчук, Л. П. Бедратюк. Хмельницький : Хмельн. нац. ун-т, 2021. 40 с. (дата звернення: 16.01.2026).

3. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень (34094). ДНАОП - Нормативно-правовая библиотека инструкции документы. URL: https://dnaop.com/html/34094/doc%20-ДСН_3.3.6.042-99 (дата звернення: 08.02.2026).

4. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання : ДСТУ 8302:2015. [Чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 17 с.

5. Spring Boot Reference Documentation. *Spring.io*. URL: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/ (дата звернення: 11.02.2026).

6. PostgreSQL 16.0 Documentation. *PostgreSQL Global Development Group*. URL: https://www.postgresql.org/docs/16/index.html (дата звернення: 11.02.2026).

7. Bloch J. Effective Java. 3rd Edition. Boston : Addison-Wesley, 2018. 412 p.

8. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

9. Schwaber K., Sutherland J. The Scrum Guide. *Scrum.org*. 2020. URL: https://scrumguides.org/scrum-guide.html (дата звернення: 17.02.2026).

10. Hibernate ORM 6.2 User Guide. *Hibernate.org*. URL: https://hibernate.org/orm/documentation/6.2/ (дата звернення: 17.02.2026).

11. OWASP Top 10:2021. The Ten Most Critical Web Application Security Risks. *OWASP Foundation*. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 17.02.2026).

12. Баришполець О. Т. Український сегмент спортивної журналістики: стан та перспективи розвитку. *Наукові записки Інституту журналістики*. 2018. Т. 70. С. 112–120.

13. Бебик В. Г. Спортивна журналістика в системі соціальних комунікацій. *Інформаційне суспільство*. 2019. Вип. 29. С. 45–52.

14. Безручко О. В. Жанрова палітра сучасного спортивного телебачення України. *Вісник Київського національного університету культури і мистецтв. Серія: Аудіовізуальне мистецтво і виробництво*. 2020. № 3(1). С. 15–24.

15. Голубєв В. П. Мультимедійна журналістика: навчальний посібник. Рівне : МЕГУ, 2021. 240 с.

16. Гнатюк С. Л. Трансформація спортивних медіа в епоху цифровізації. *Медіапростір*. 2021. № 14. С. 88–95.

17. Даниленко С. І. Сучасні спортивні інтернет-видання: особливості функціонування. *Діалог: медіастудії*. 2019. Вип. 25. С. 67–76.

18. Житарюк М. Г. Світова журналістика: західна модель. Львів : ПАІС, 2022. 312 с.

19. Квіт С. М. Масові комунікації: підручник. Київ : Вид. дім «Києво-Могилянська академія», 2018. 206 с.

20. Костюк О. Г. Спортивний веб-ресурс як інструмент формування лояльності аудиторії. *Прикладна комунікація*. 2020. № 12. С. 34–41.

21. Лизанчук В. В. Основи журналістики: підручник. Львів : ЛНУ імені Івана Франка, 2019. 480 с.

22. Мащенко І. Г. Світове телебачення: спорт і медіабізнес. Київ : Тетра, 2018. 192 с.

23. Потятиник Б. В. Інтернет-журналістика: навч. посіб. Львів : ПАІС, 2020. 244 с.

24. Салига П. П. Дизайн та юзабіліті медіасайтів: сучасні тренди. *Вісник ЛНУ імені Івана Франка*. 2021. Вип. 50. С. 102–110.

25. Ситник О. В. Побудова спортивного контенту в соціальних мережах. *Сучасний мас-медійний простір*. 2022. № 4. С. 55–62.

26. Тернова А. І. Спортивна преса України: історія та сучасність. Запоріжжя : ЗНУ, 2019. 156 с.

27. Ткаченко В. О. Візуалізація даних у спортивній журналістиці. *Комунікаційні технології*. 2021. Т. 18. С. 120–128.

28. Шевченко В. Е. Формування візуального образу в мультимедійному виданні. *Наукові читання Інституту журналістики*. 2020. № 26. С. 77–84.

29. Шпак В. Г. Спортивна публіцистика та її вплив на молодіжну аудиторію. *Журналістика*. 2019. Вип. 18. С. 22–30.

30. Юричко Ю. В. Технічні аспекти створення новинних агрегаторів. *Інженерія програмного забезпечення*. 2022. № 3. С. 41–49.

31. Ярош О. Б. Монетизація спортивних інтернет-ресурсів: український досвід. *Економіка та менеджмент медіа*. 2021. № 2. С. 12–20.

32. Andrew B. S. Sports Journalism: A Practical Introduction. London : SAGE Publications, 2019. 256 p.

33. Billings A. C. Sports Media: Transformation, Integration, Consumption. New York : Routledge, 2021. 310 p.

34. Boyle R. Sports Journalism: Context and Practice. 2nd ed. London : SAGE, 2020. 224 p.

35. Bradshaw P. The Online Journalism Handbook. 2nd ed. New York : Routledge, 2022. 344 p.

36. Butt G. J. Media and Sports Analytics. Oxford : Oxford University Press, 2021. 288 p.

37. Castells M. The Communication Power. Oxford : Oxford University Press, 2018. 537 p.

38. Daum E. The Digital Future of Football Media. *Journal of Sports Management*. 2022. Vol. 15, No. 2. P. 134–145.

39. Garrison B. Sports Reporting. Ames : Iowa State University Press, 2019. 380 p.

40. Hutchins B., Rowe D. Sport Beyond Television: The Internet, Digital Media and the Rise of Networked Media Sport. New York : Routledge, 2020. 264 p.

41. Jenkins H. Convergence Culture: Where Old and New Media Collide. New York : NYU Press, 2018. 320 p.

42. Kovach B., Rosenstiel T. The Elements of Journalism. 4th ed. New York : Crown, 2021. 352 p.

43. Lambert S. Digital Storytelling: Capturing Lives, Creating Communities. London : Routledge, 2020. 216 p.

44. Mullin B. J., Hardy S. Sport Marketing. 5th ed. Champaign : Human Kinetics, 2021. 520 p.

45. Pedersen P. M. Strategic Sport Communication. 3rd ed. Champaign : Human Kinetics, 2022. 464 p.

46. Sanderson J. It’s a Whole New Ballgame: How Social Media is Changing Sports. New York : Hampton Press, 2021. 240 p.

47. Steen R. Sports Journalism: A Multimedia Primer. London : Routledge, 2020. 328 p.

48. Stofer K. T. Sports Journalism: An Introduction to Reporting and Writing. Lanham : Rowman & Littlefield, 2019. 210 p.

49. Tuchman G. Making News: A Study in the Construction of Reality. New York : Free Press, 2018. 256 p.

50. Whannel G. Media Sport Stars: Masculinities and Moralities. London : Routledge, 2022. 288 p.

51. Zelizer B. What Journalism Could Be. Cambridge : Polity Press, 2021. 240 p.

52. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/rada/show/va037282-99#Text (дата звернення: 08.02.2025).

53. Про охорону праці Стаття 28. Опрацювання, прийняття та скасування нормативно-правових актів з охорони праці. Головна - Законодавство України 2019 рік. URL: https://kodeksy.com.ua/pro_ohoronu_pratsi283_new/statja-28.htm (дата звернення: 08.02.2026).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

1. Перелік умовних скорочень

Користувач: Звичайний користувач, який реєструється на платформі для перегляду новин.

Адмін: Адміністратор, який має доступ до панелі адміністратора для керування контентом, користувачами та іншими даними.

Публікація: Процес публікації новини користувачем або адміністратором.

ТЗ: Технічне завдання.

БД: База даних.

Фронт: Фронтенд частина вебзастосунку.

Бек: Бекенд частина вебзастосунку.

Новина: Інформаційний матеріал про спортивні події.

Категорія: Категорія, до якої відноситься новина (спорт, команда, змагання).

Автор: Автор новини (користувач або адміністратор).

2.Сервер ідентифікації – система для реєстрації та аутентифікації користувачів

3.Загальні відомості

3.1. Призначення документа

Цей документ описує технічні вимоги до розробки веб-додатку "45Min", платформи для публікації спортивних новин, розробленої з використанням ASP.NET Core, Entity Framework Core та Microsoft SQL Server. У цьому розділі описано наступне. Цей документ призначений для використання командами розробників, які створюють системи, що дозволяють користувачам реєструватися, публікувати новини, коментувати їх та дозволяють адміністраторам керувати даними користувачів, новин та категорій. У цьому документі викладено:

- Основні функціональні та нефункціональні вимоги до системи.
- Технологічний стек, використаний для розробки додатку.
- Опис функціоналу для користувачів та адміністраторів.

– Архітектура та системні вимоги додатку. Технічні специфікації є основним керівництвом для створення, реалізації та тестування функціональності веб-додатку, гарантуючи, що результати відповідають очікуванням кінцевого користувача та замовника.

3.2. Загальні положення

Цей документ визначає технічні та якісні характеристики вебдодатку для публікації спортивних новин "45Min". Система повинна відповідати наступним вимогам:

– Функціонально повна: забезпечує реєстрацію, публікацію новин, коментування, пошук.

– Надійний: гарантує безпеку даних і точне виконання завдань.

– Модернізується: легко розширюється новими функціями.

– Модульна: компоненти можна оновлювати окремо.

– Інтуїтивно зрозумілий: простий, зручний інтерфейс.

– Безпечний: забезпечує безпеку даних і захист від зовнішніх загроз.

3.3. Повне найменування системи та її умовне позначення

Повне найменування: Електронна система для реалізації вебдодатку "45Min" – платформи для публікації спортивних новин. Умовне позначення: Система. Планові терміни початку та закінчення робіт Плановий термін початку робіт – 05.09.2024 року, орієнтовний термін завершення робіт – 18.11.2024 року.

3.4. Нормативно правові документи, використані під час створення системи:

Нормативно-правові акти, що були використані при аналізі та описі бізнес-процесів:

– Закон України “Про захист персональних даних”.

– Закон України “Про авторське право і суміжні права”.

– Державні стандарти України (ДСТУ) на програмне забезпечення.

– ISO/IEC 27001.

– ISO 9241.

– WCAG (Web Content Accessibility Guidelines).

Даний перелік не є вичерпним. Вимоги законодавства України, нормативних та керівних документів, що стосуються мети, призначення та цілей надання послуг можуть бути уточнені.

4. ПРИЗНАЧЕННЯ ТА ЦІЛІ СТВОРЕННЯ СИСТЕМИ

4.1. Мета створення системи Метою впровадження системи "45Min" є розробка та реалізація вебсервісу для публікації спортивних новин, який забезпечить ефективний механізм створення та поширення інформаційних матеріалів.

4.2. Призначення системи Система "45Min" покликана надати користувачам єдину платформу для створення, перегляду та обговорення спортивних новин. Користувачі повинні мати можливість швидко знаходити, переглядати та публікувати новини відповідно до своїх інтересів. Система стане надійним помічником для журналістів, блогерів та звичайних користувачів, надаючи зручні інструменти для створення та поширення контенту.

4.3. Характеристики об’єкта автоматизації Автоматизація спрямована на систему "45Min", яка дозволяє користувачам переглядати, сортувати та фільтрувати новини за різними критеріями, такими як категорія, дата публікації та інші параметри. Система інтегрована у вебдодаток і має на меті надати користувачам зручний інструмент для роботи з інформацією про спортивні події. Система повинна забезпечувати доступ до інформації про новини в електронному форматі за допомогою підключення до Інтернету. Користувачі можуть переглядати новини в будь-який час, а також сортувати та фільтрувати дані за заданими параметрами. Користувачами системи є журналісти, блогери та звичайні користувачі, які бажають публікувати або переглядати спортивні новини. Функціональні вимоги до системи:

- Система повинна підтримувати механізм фільтрації та сортування інформації про новини за різними параметрами
- Система повинна зберігати всю введену інформацію в базі даних, гарантувати її цілісність і захищати від несанкціонованого доступу. Інформація, яка повинна зберігатися в системі, включає в себе:

- Заголовок новини;
- Текст новини;
- Категорія новини;
- Автор новини;
- Дата публікації;
- Коментарі до новини.

5. ВИМОГИ ДО СИСТЕМИ

5.1. Вимоги до Системи в цілому Загальні системні вимоги При розробці системи "45Min" слід дотримуватися наступних принципів

- Функціональна достатність: система повинна надавати всі функції, необхідні для публікації та перегляду спортивних новин.
- Зручний веб-інтерфейс: інтерфейс повинен бути зручним і зрозумілим для всіх категорій користувачів.
- Відмовостійкість: система повинна продовжувати працювати без перерви у разі виникнення помилки.
- Надійність зберігання даних: має бути забезпечена цілісність і безпека даних користувачів та новин.
- Модернізація та масштабованість: система повинна бути достатньо гнучкою для впровадження нових можливостей та функцій.
- Зручність і простота використання: користувачі повинні мати можливість легко керувати функціями системи.

5.2. Вимоги до структури та функціонування Система "45Min" повинна забезпечувати наступні функціональні завдання:

Реєстрація та аутентифікація користувачів:

- Користувачі повинні мати можливість зареєструватися, ввівши необхідну інформацію (логін, пароль, email).
- Після реєстрації користувачеві повинен бути наданий доступ до системи за допомогою безпечного механізму аутентифікації.

Публікація новин:

– Зареєстровані користувачі (або адміністратори) повинні мати можливість створювати нові новини, додавати заголовок, текст, зображення та вибирати категорію.

– Система має забезпечувати перевірку опублікованих новин на наявність нецензурної лексики та іншого небажаного контенту.

Перегляд новин:

– Користувачі повинні мати можливість переглядати опубліковані новини, сортувати їх за датою, категорією або популярністю.

– Система має забезпечувати зручну навігацію між новинами.

Коментування:

– Користувачі повинні мати можливість залишати коментарі до опублікованих новин.

– Система має забезпечувати модерацію коментарів для запобігання спаму та образ.

Пошук новин:

– Система повинна мати функцію пошуку новин за ключовими словами.

Панель адміністратора:

Адміністратор повинен мати доступ до панелі управління, де він може:

– Додавати, редагувати та видаляти користувачів.

– Додавати, редагувати та видаляти новини.

– Керувати категоріями новин.

– Переглядати статистику відвідування сайту.

Система управління користувачами:

– Система повинна дозволяти користувачам редагувати свій профіль.

– Система повинна забезпечувати відновлення пароля у випадку його втрати.

6. НЕФУНКЦІОНАЛЬНІ ВИМОГИ

6.1. Вимоги до надійності та відмовостійкості Програмне забезпечення "45Min" повинно функціонувати цілодобово, з можливістю оперативного

відновлення після збоїв чи аварій. У штатному режимі роботи системи має бути забезпечено:

- Цілодобова робота серверів і технічних засобів.
- Витримка пікових навантажень, зокрема під час великих спортивних подій або релізів нових функцій.
- Збереження даних та обробка великих обсягів запитів. У разі збоїв чи аварій система має забезпечувати:
- Автоматичне завершення збереження даних.
- Відновлення з резервної копії з мінімальними втратами.

- 6.2. Вимоги до потужності системи Під час пікових навантажень система повинна витримувати:
  - Запитів на день: ~1000 (з урахуванням пікових днів).
  - Запитів на годину: ~150-200 під час піків.
  - Навантаження подій на секунду: ~20-50 запитів.

6.3. Вимоги до інтерфейсу Інтерфейс системи "45Min" має бути:

- Зручним для навігації і пошуку новин за категоріями, тегами та іншими фільтрами.
- Адаптивним до різних розмірів екранів.
- Інтуїтивно зрозумілим, з мінімальними діями для виконання запитів користувачів.
- Оптимізованим для української мови та роботи в сучасних браузерах (Chrome, Firefox, Edge тощо).

6.4. Вимоги до захисту інформації Для захисту інформації в системі "45Min" від несанкціонованого доступу вживаються такі заходи:

- Правові: дотримання вимог законодавства та стандартів захисту інформації.
- Адміністративно-організаційні: контроль доступу до серверів та компонентів системи, моніторинг поведінки персоналу, що обслуговує систему.
- Програмно-апаратні: використання захисту від лімітів на кількість запитів до бази даних для запобігання SQL-ін'єкцій, XSS-атак та DDoS-атак.

Основні принципи захисту інформації:

- Централізоване управління: всі аспекти безпеки системи управляються в одному місці.
- Ефективність: для ефективного запобігання атакам необхідно забезпечити адекватний захист від новітніх загроз.
- Надійність: безпека даних повинна зберігатися у випадку часткового виходу системи з ладу.
- Конфіденційність: користувачі не повинні знати про певні механізми безпеки для запобігання потенційним атакам.

6.5. Вимоги до розвитку та модернізації системи Система "45Min" повинна бути захищена від типових атак і забезпечувати належний рівень інформаційної безпеки. Вимоги є наступними:

- Захист від типових атак: система повинна бути захищена від SQL-ін'єкцій, XSS-атак та інших поширених вразливостей.
- Конфіденційність інформації: конфіденційна інформація, така як ідентифікатори сеансів та ідентифікатори користувачів, не повинна бути доступна на веб-сайті або в інтерфейсі користувача.

Фізична безпека:

- Фізичний доступ до серверів та іншого обладнання повинен бути обмежений.
- Усі дії з доступу повинні бути задокументовані та відстежуватися.
- Система повинна включати механізми обмеження кількості запитів до бази даних, щоб запобігти перевантаженню та забезпечити стабільну роботу системи.

Додаткові пропозиції:

- Масштабованість: Система повинна бути спроможна обробляти зростання кількості користувачів та обсягу даних.
- Доступність: Система повинна бути доступна 24/7 з мінімальним часом простою.

– Інтеграція: Система повинна мати можливість інтеграції з іншими системами (наприклад, системами аналітики, соціальними мережами).

– Тестування: Система повинна проходити регулярне тестування на надійність, безпеку та продуктивність.

6.6. Вимоги до патентної чистоти Ліцензійні угоди: все програмне та апаратне забезпечення, що входить до складу системи, повинно поважати право інтелектуальної власності та авторське право і відповідати умовам ліцензійних угод. Патентна чистота: розробник системи не має виключних авторських прав на компоненти системи або систему в цілому. Необхідно переконатися, що всі використовувані технології та компоненти не порушують авторських прав та відповідають вимогам законодавства про інтелектуальну власність.

6.7. Вимоги до розвитку та модернізації системи Якщо система розробляється, вона повинна мати можливість бути модернізованою або розширеною за мінімальний проміжок часу. Розробка системи включає в себе:

– Масштабування системи: збільшення обчислювальної потужності та оптимізація продуктивності бази даних зі збільшенням кількості користувачів та обсягу контенту.

– Інтеграція нових функцій: можливість додавання нових функціональних можливостей, таких як підтримка різних форматів мультимедіа, аналітика даних та персоналізація контенту.

– Розширена функціональність: нові процеси та функції можуть бути додані без значних змін у поточній функціональності. Це включає додавання нових категорій новин, інтеграцію з соціальними мережами та іншими платформами.

– Горизонтальне масштабування: дозволяє розширювати API системи шляхом додавання нових серверів або інфраструктури.

6.8. Вимоги до стандартизації та уніфікації

– Стандартизація системи має бути забезпечена використанням сучасних засобів проектування та розробки. Основні вимоги:

– Уніфікований підхід до проектування: використання стандартних методів для побудови функціонального та інформативного програмного забезпечення, що відповідає міжнародним стандартам.

– Уніфіковане програмне забезпечення: використання програмних компонентів, що підтримують стандартизацію, спрощують обслуговування та зменшують витрати на модернізацію.

– Відповідність міжнародним стандартам: всі програмні засоби повинні відповідати міжнародним та національним IT-стандартам, таким як ISO/IEC 27001 (управління інформаційною безпекою) та ISO/IEC 25010 (якість програмного забезпечення).

6.9. Вимоги до інформаційного забезпечення Інформаційне забезпечення системи повинно враховувати:

– Забезпечення фізичної та логічної цілісності даних: Використання механізмів контролю даних для уникнення їх пошкодження або втрати.

– Мінімізація надмірності даних: Уникнення дублювання інформації для збереження цілісності та ефективності роботи системи.

– Стандартизація представлення даних: Дані повинні бути представлені у стандартизованому форматі для спрощення їх обробки та інтеграції з іншими системами.

– Достовірність та актуальність даних: Регулярне оновлення даних та перевірка їхньої відповідності актуальній інформації.

– Автоматична консолідація даних: Забезпечення цілісності даних у географічно розподілених системах.

7.АДМІНІСТРАТИВНА ІНФРАСТРУКТУРА

7.1 Розміщення системи Доступність і тестування:

– Під час розробки та впровадження система повинна бути доступною для перегляду та тестування користувачами через веб-сервіси.

– Для покращення роботи та тестування система повинна мати окремі середовища для тестування стабільної та нової функціональності. Середовища:

– PROD (production environment): основне середовище для користувачів.

– DEV: для розробки та тестування нових функцій і для проміжного тестування. Розгортання та управління:

– Початкове розгортання середовища DEV повинно бути виконано за допомогою ресурсів, доступних для проекту.

– Середовище PROD розгортається і управляється на апаратному забезпеченні, яке використовується для виробничої системи.

7.2. Логування Призначення та конфігурація Для забезпечення безпеки даних і контролю доступу до функцій системи необхідно впровадити уніфіковане протоколювання дій користувачів і змін даних.

Події для логування:

– Запуск/зупинка сервісу: реєстрація окремих подій запуску/зупинки сервісу.

– Помилки: реєстрація помилок, таких як збої зв'язку, порушення цілісності даних, неочікувані затримки в обробці інформації.

– Критичні події: реєстрація критичних подій системи моніторингу, таких як недостатня кількість пам'яті або місця на диску.

– Дії користувачів: протоколювання дій користувачів, таких як публікація новин, редагування профілю, додавання коментарів.

Захист і форматування логів:

– Форматування журналу: журнали зберігаються у вигляді текстових файлів.

– Формат подій: реєстрація подій повинна відповідати встановленому формату, включаючи дату і час.

– Маскування даних: конфіденційні дані (наприклад, паролі) повинні маскуватися відповідно до встановленого формату.

– Мітки часу: мітки часу реєстру слід зберігати у форматі UTC.

7.3. Підключення до системи моніторингу та алертингу

Сповіщення та алертинг:

– Система повинна сповіщати користувачів про події, що потребують уваги, через визначений канал сповіщення. Це включає:

- Аварійні зупинки або перезапуски сервісів.
- Логи рівня ERROR та вище.
- Повідомлення про підозрілу активність: наприклад, спроби несанкціонованого доступу.
- Сповіщення повинні бути налаштовані таким чином, щоб забезпечити своєчасне реагування на критичні ситуації.

Централізоване зберігання налаштувань:

- Система повинна підтримувати централізоване зберігання налаштувань.
- Налаштування мають бути представлені у вигляді конфігурації у форматі JSON.
- Доступ до цих налаштувань має бути обмежений. Адміністратор повинен мати можливість заповнювати ці налаштування, але звичайні користувачі не повинні мати доступ до їх перегляду або редагування.

8. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

8.1 Приймальні випробування Формування робочих груп Для проведення приймальних випробувань буде сформована робоча група, до складу якої увійдуть представники замовника та підрядника, для проведення приймальних випробувань. Процедура тестування На етапі введення в експлуатацію проводиться тестування системи з метою перевірки відповідності створеної системи вимогам технічного завдання (ТЗ). Випробування проводяться відповідно до програми та методики випробувань, що розробляються виконавцем та затверджуються замовником. Передача результатів розробки:

Після завершення розробки та успішного завершення тестування виконавець передає замовнику:

Виключне право власності на програмне забезпечення

Проектну документацію, включаючи:

- Інструкції по встановленню та налаштуванню системи.
- Інструкції користувача системи.
- Вихідний код програмного забезпечення.

Акт приймання-передачі послуг, що надаються поетапно.

Передача та приймання копій програмного забезпечення.

Гарантійна підтримка:

– На все програмне забезпечення та конфігурації, розгорнуті під час побудови системи, має бути надана гарантійна підтримка, включаючи виправлення помилок та оновлення версій програмного забезпечення, протягом щонайменше 12 календарних місяців з дати розгортання системи.

Ключові зміни:

– Розширення логування: Додано логування дій користувачів для кращого аналізу та безпеки.

– Посилення безпеки: Додано вимогу до маскування конфіденційних даних у логах.

– Моніторинг: Додано вимогу до моніторингу підозрілої активності.

– Технічні деталі: Збережено деталі щодо технологічного стеку, середовищ розробки та процедури приймання.

ДОДАТОК Б

(Презентаційні матеріали)

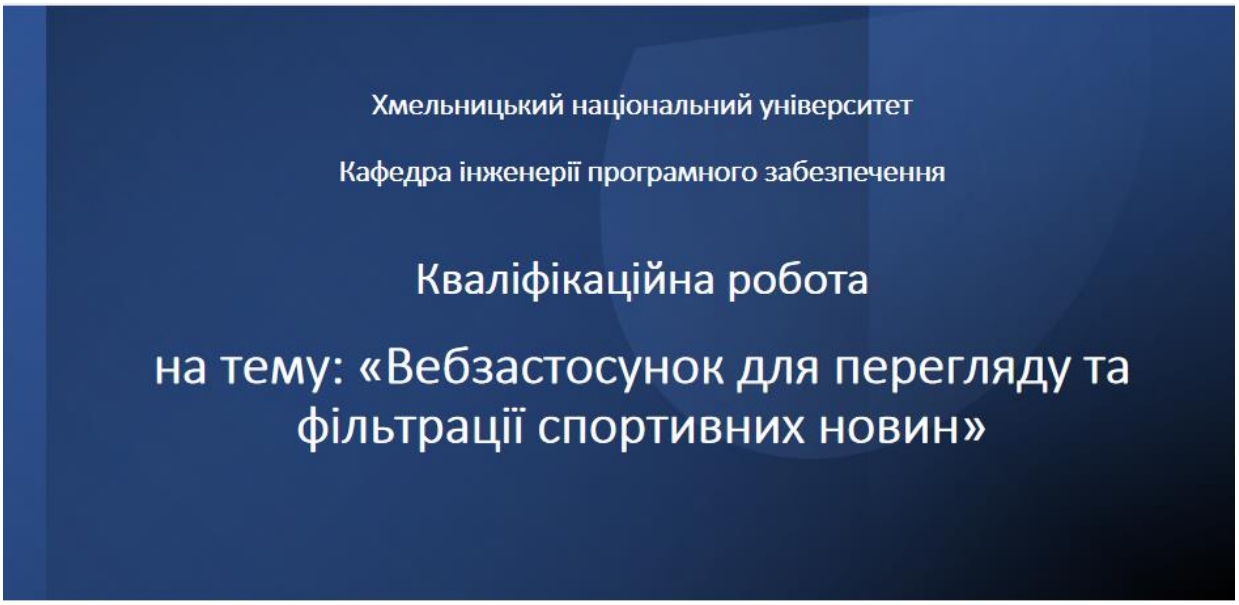


Рисунок А.1

Рисунок А.2

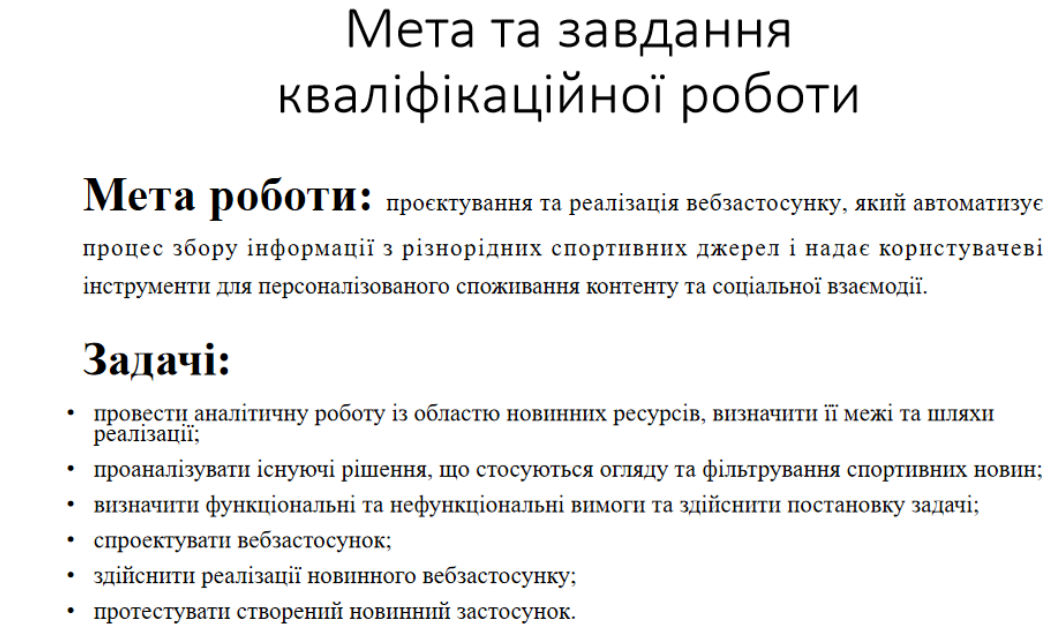


Рисунок А.3

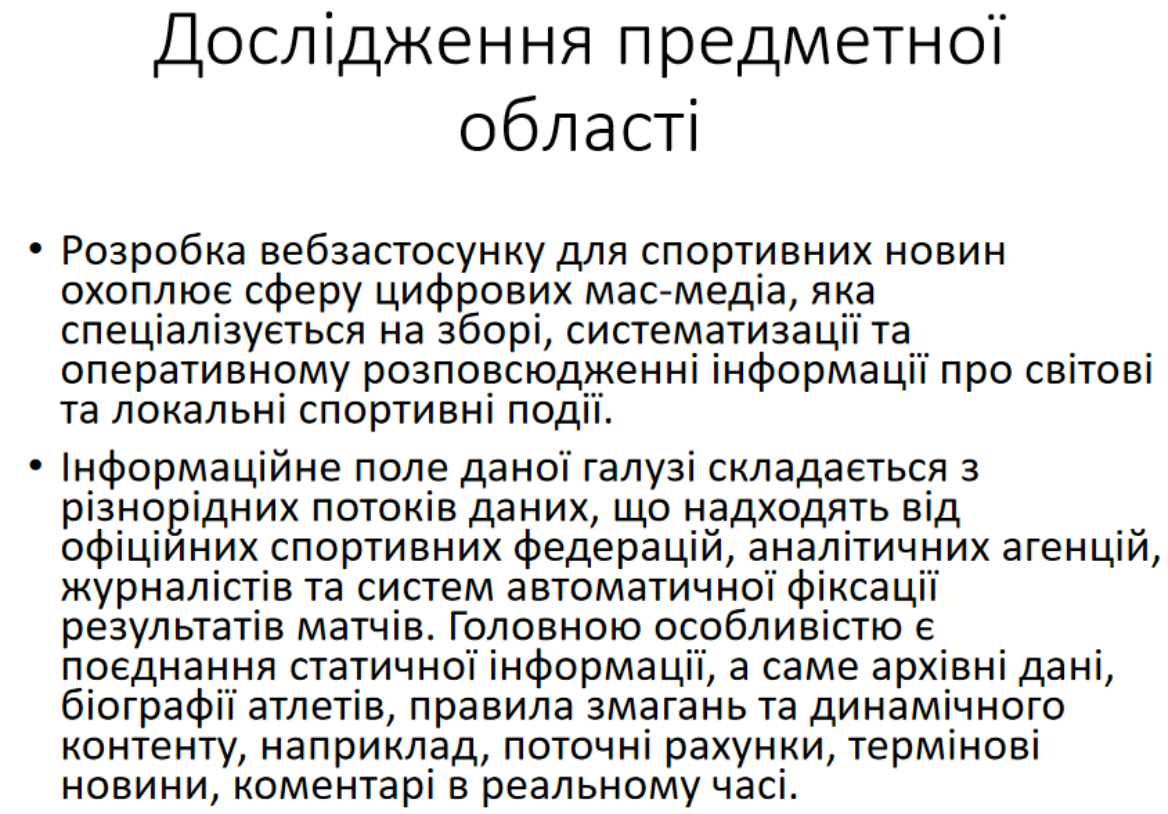


Рисунок А.4

Аналіз наявного програмно-технічного забезпечення

- Важливим етапом під час розробки є аналіз існуючого програмного забезпечення, зокрема новинних вебзастосунків. Визначення їх переваг та недоліків, а також якихось особливостей, для того щоб врахувати під час своєї розробки, є досить важливим моментом.
- Загалом їх є величезна кількість. В рамках кваліфікаційної роботи було виділено лише декілька, що є невеликою частиною від усієї кількості.
- Було проаналізовано такі ресурси як BBC Sport, The Guardian Sports Section, ESPN, The Athletic, Sky Sports, Bleacher Report, Tribuna.com та інші.

Перевагами таких ресурсів є можливість пошуку за такими критеріями, як дата матчу, актуальність новини, можливість залишати відгук, зручність, простота використання тощо.
До **недоліків** відноситься дещо суворий дизайн, велика кількість спливаючих вікон та пропозицій, заплутаність у користуванні.

Рисунок А.5

Визначення вимог

В розроблюваному програмному продукті необхідно здійснити реалізацію класичних функцій, що є обов'язковими для безпрецедентної роботи вебзастосунку, а саме.

- здійснення перегляду потрібної інформації би;
- можливість фільтрування та вибору новин;
- можливість редагування та додавання новини;
- можливість персоналізувати стрічку;
- можливість безперебійного доступу до спортивного контенту;
- наявність розмежування за ролями;

Рисунок А.6

Вибір типу архітектури

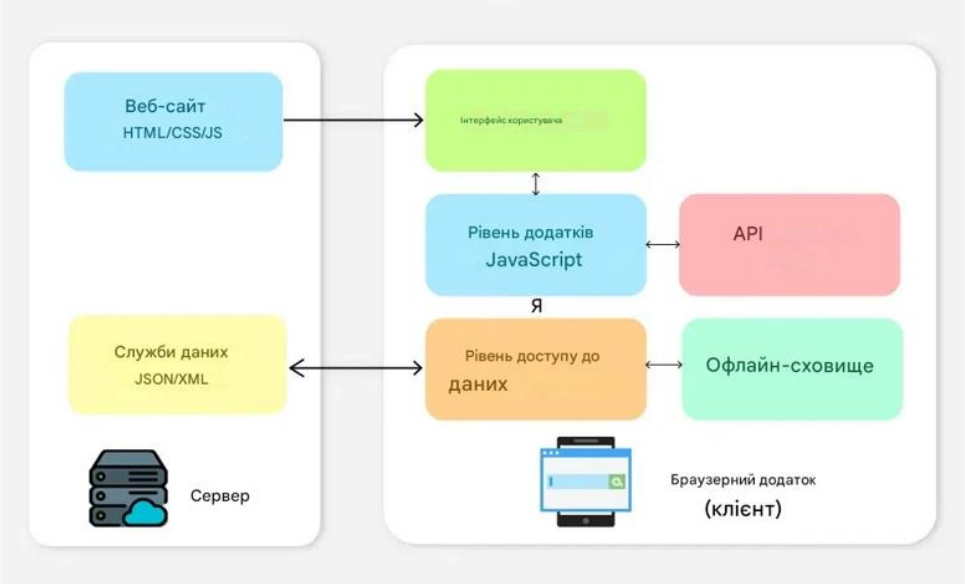


Рисунок А.7

База даних

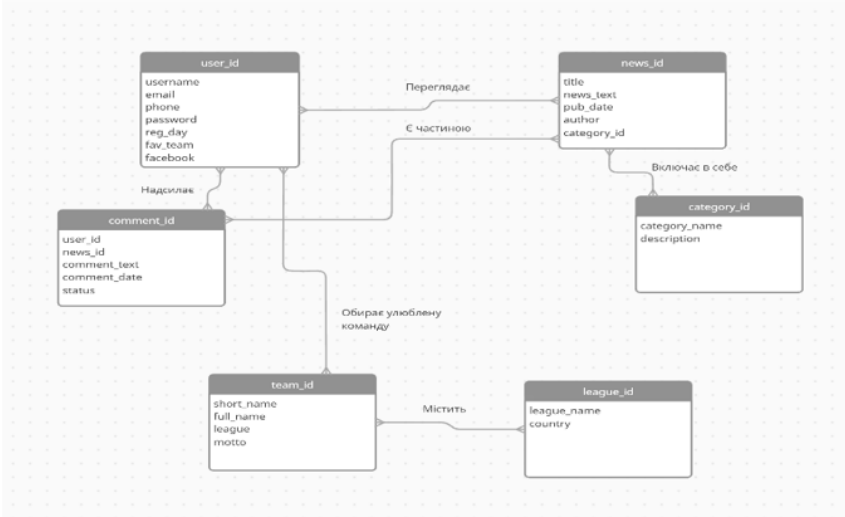


Рисунок А.8

Структура проекту

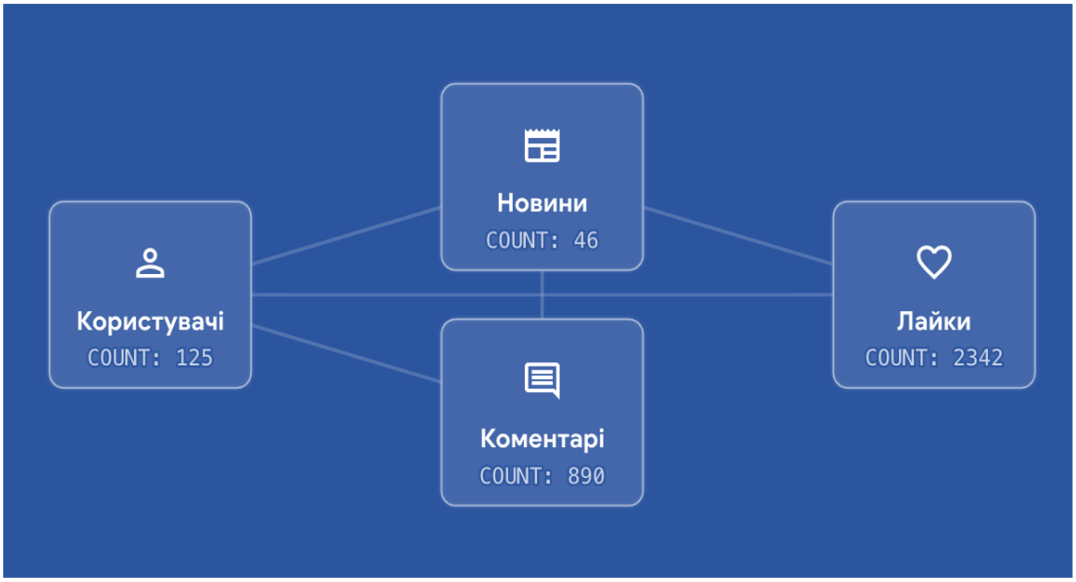


Рисунок А.9

Використані технології

Рівень системи	Технологія	Роль у проєкті
Frontend	Next.js (TypeScript)	Інтерфейс, SEO, швидка навігація.
Styling	Tailwind CSS	Адаптивність та дизайн.
Backend	NestJS	Бізнес-логіка, API, автентифікація.
Database	PostgreSQL	Надійне збереження структурованих даних.
Real-time	WebSockets (Socket.io)	Оновлення коментарів та лайків у реальному часі.
Auth	JWT	Безпечний вхід та захист профілів.
Infrastructure	Docker	Контейнеризація для легкого розгортання проєкту.

Рисунок А.10

Реалізація вебзастосунку

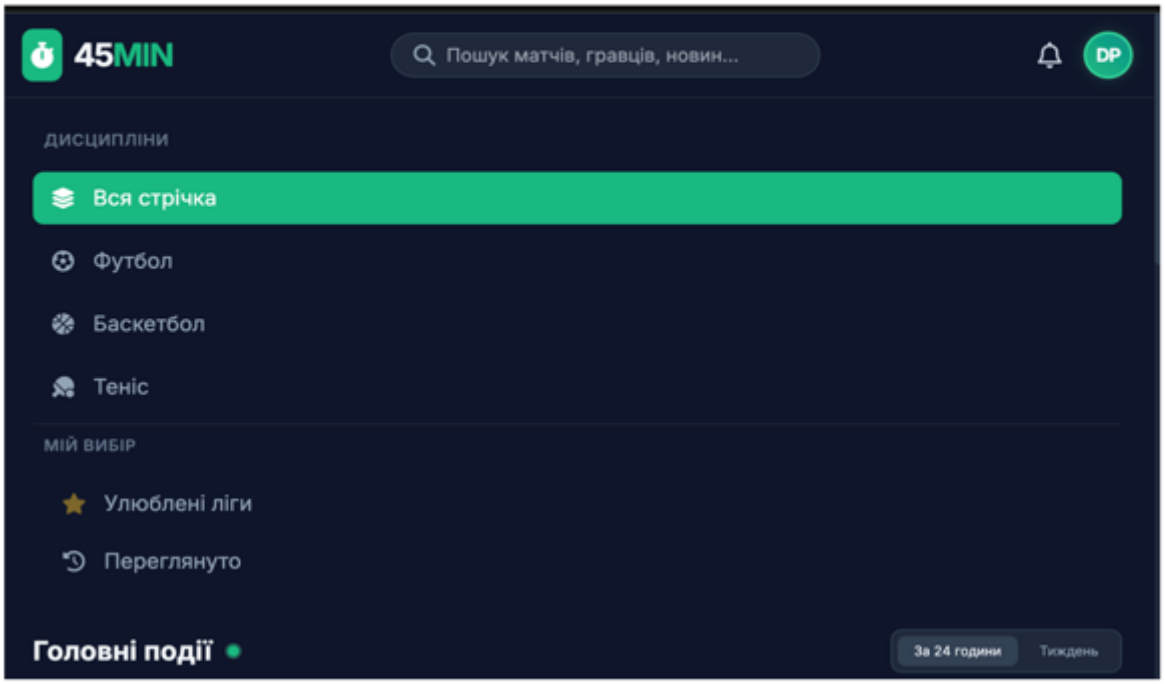


Рисунок А.11

Реалізація вебзастосунку перегляд та коментування

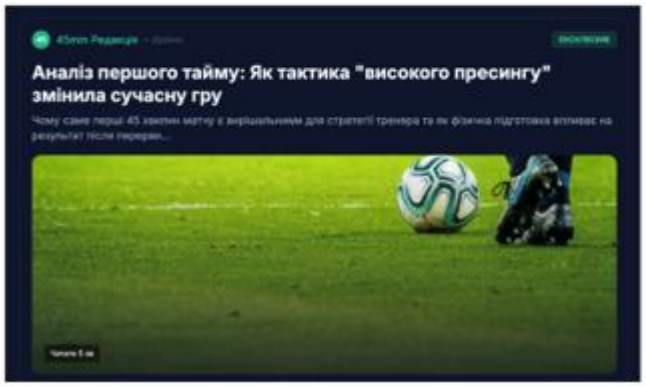


Рисунок А.12

Реалізація вебзастосунку реєстрація та авторизація

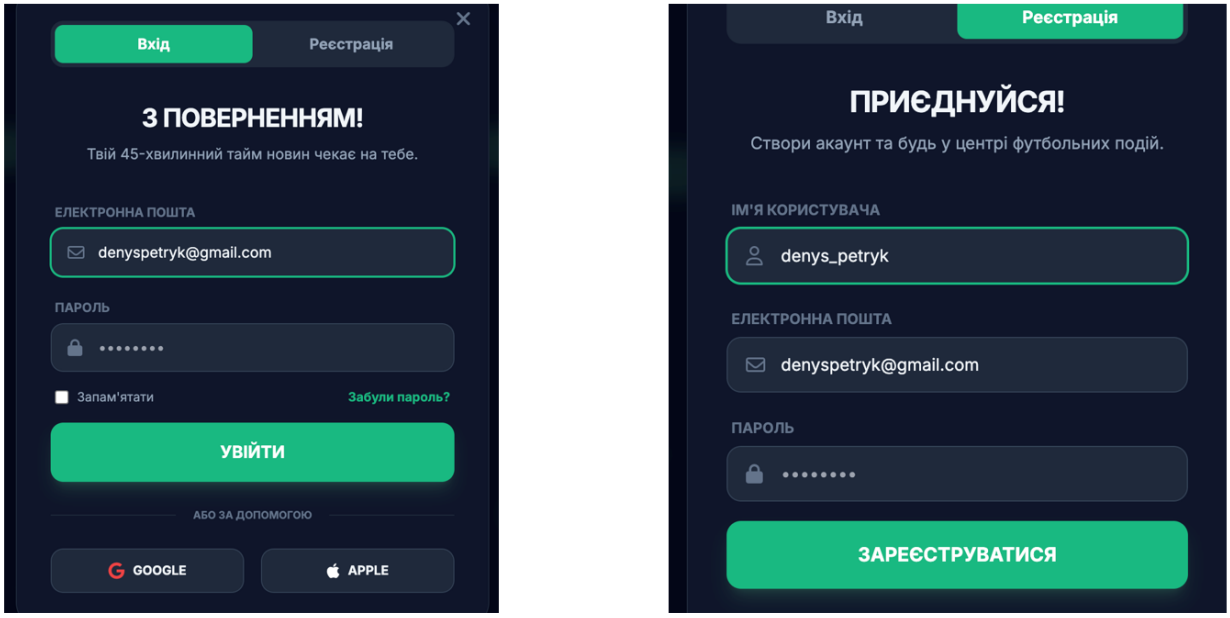


Рисунок А.13

Вимоги до технічного та програмного забезпечення

- Новинним вебзастосунком може користуватись будь-хто з доступом до мережі Інтернет з будь-якого браузера, оскільки дана розробка є крос-браузерною.

Рисунок А.14

Тестування

Перевірка	Очікуваний показник	Фактичний результат під час тестування
Середній час відповіді API	Менше 500 мс	210 мс (при стандартному навантаженні)
Успішність E2E сценаріїв	100% проходження	Усі 45 критичних сценаріїв пройдено успішно
Продуктивність	Більше 90 балів	96 балів (оптимізовано кешування зображень)
Доступність	Відповідність WCAG 2.1	100 балів (високий контраст, підтримка screen-readers)
Максимальне навантаження	300 запитів/секунду	450 запитів/секунду (до початку деградації швидкості)

Рисунок А.15

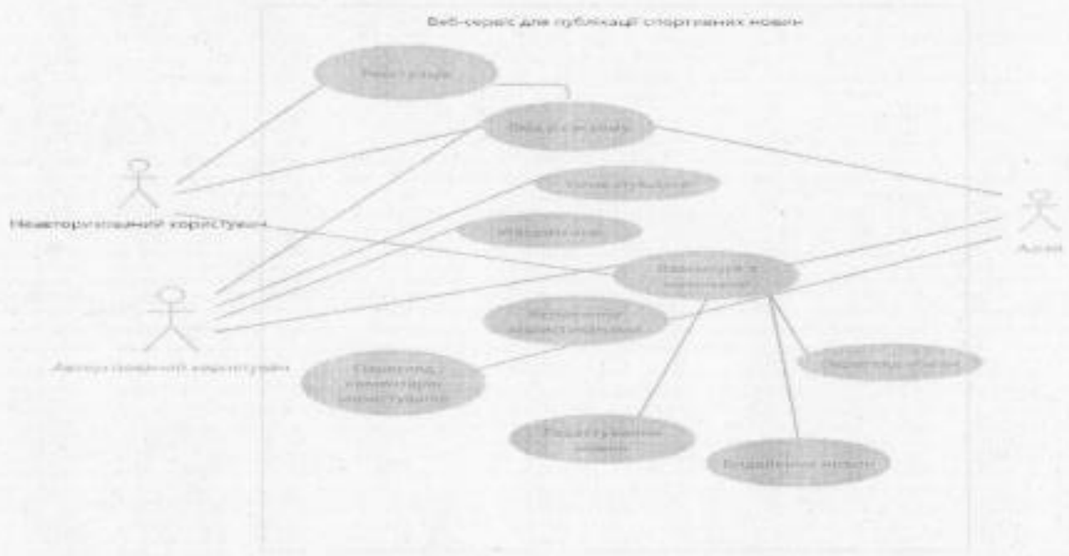
Висновки

Провести аналіз існуючих вебзастосунків;	✓
Здійснити аналіз та надати перелік вимог для проектування та розробки вебзастосунку;	✓
Здійснити вибір та опис архітектури, інтерфейсу, базу даних;	✓
Здійснити розробку вебзастосунку ;	✓
Здійснити тестування вебзастосунку.	✓

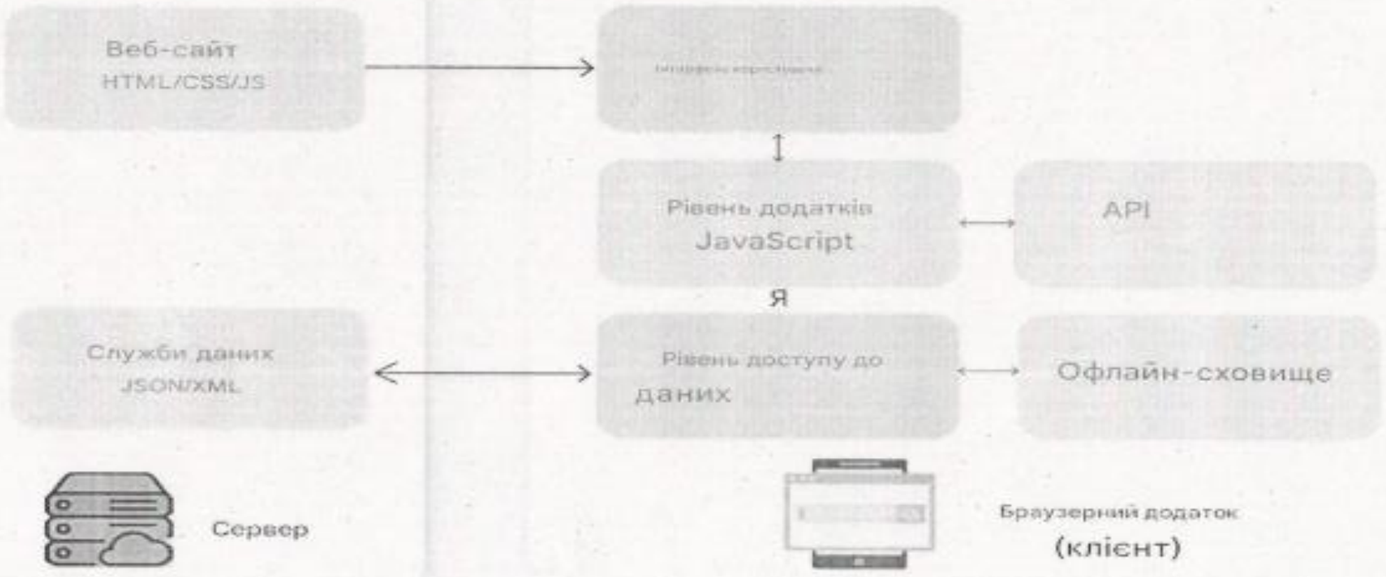
Рисунок А.16

Рисунок А.17

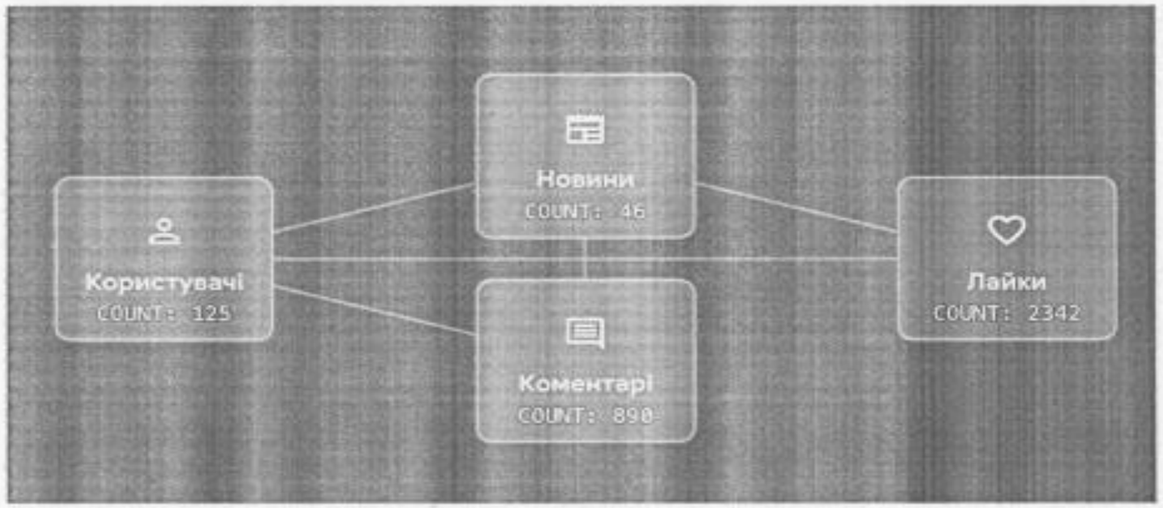
ГРАФІЧНА ЧАСТИНА



					КвРІПЗ.2201106.01.12Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Вебзастосунок для перегляду та фільтрації спортивних новин Діаграма ВВ	Літера	Маса / Масштаб
Розробив		Петрик Д.М.		30.0			
Керівник		Форкун Ю.В.		30.0		Аркуш 1	Аркушів 3
Консульт.				30.0			
Н. контр.		Яцина О.М.		30.0		ХНУ, ІПЗ-22-1	
Зав. каф.		Бездрюк Л. П.		30.0			



					КвРІПЗ.2201106.01.12Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Вебзастосунок для перегляду та фільтрації спортивних новин	Літера	Маса	Масштаб
Розробив		Петрик Д.М.		03.01				
Керівник		Форкун Ю.В.		03.01	Архітектура		Аркуш 1	Аркушів 3
Консультант								
Н. Контр.		Яшина О.М.		03.01				
Зав. каф.		Бездрюк Л. П.		03.01				



					КвРІПЗ.2201106.01.12Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Вебзастосунок для перегляду та фільтрації спортивних новин База даних	Літера	Маса / Масштаб
Розробив		Петрик Д.М.		6.20			
Керівник		Форкун Ю.В.		6.20		Аркуш 1	Аркушів 3
Консульт.							
Н. Контр.		Яцина О.М.		6.20		ХНУ, ІПЗ-22-1	
Зав. каф.		Бедратюк Л.П.		6.20			

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного
забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Петрика Дениса Максимовича
факультет ІТ, IVкурс, група ІПЗ-21-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

09.05.2026р
дата

підпис

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Денис ПЕТРИК

Співавтор:

Назва: Вебзастосунок для перегляду та фільтрації спортивних новин

Науковий керівник: канд. техн. наук, доцент Юрій ФОРКУН

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:5.5%

Коефіцієнт подібності 2:2.48%

Мікропробіли: 15

Заміна букв: 0

Інтервали: 0

Білі знаки: 15

Дата створення звіту: 2026-06-03 13:02:10.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 25.05.26

експерт

Mon Jun 01 12:51:33 EEST 2026, Форкун Юрій Вікторович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (http://ap.km.ua) v-16.718

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: UA, US, RU. **Помилок в документах: 14%**

ID: 272956 Назва: БКР_Вебзастосунок для перегляду та фільтрації спортивних новин Додано в БД: 2026-06-01 Автора: Денис ПЕТРИК Керівники: канд. техн. наук, доцент Юрій ФОРКУН Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	78396	697	3880 (5%)	58 (8%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

ДипломникПетрик Денис Максимович

Тема Вебресурс для перегляду та фільтрації спортивних новин

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 ; кількість сторінок записки 91

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі проведено аналіз предметної області та визначено вимоги до програмного забезпечення. Розглянуто існуючі аналоги, їх переваги та недоліки. У результаті розроблено вебзастосунок для перегляду та фільтрації спортивних новин та проведено його тестування, яке підтвердило коректну роботу системи.

2. Висновок про відповідність роботи поставленому завданню. Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням усіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовано актуальність теми, визначено мету та основні завдання кваліфікаційної роботи. У першому розділі проведено аналіз предметної області, досліджено існуючі спортивні інформаційні системи та визначено функціональні й нефункціональні вимоги до програмного забезпечення. У другому розділі розглянуто сучасні архітектурні підходи та технології веброзробки, обґрунтовано вибір клієнт-серверної архітектури для реалізації системи. У третьому розділі описано процес практичної реалізації програмного забезпечення, створення основних модулів системи, структури бази даних та користувацького інтерфейсу. Також проведено тестування функціональних можливостей вебзастосунку, результати якого підтвердили коректну роботу системи.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки спортивні інформаційні ресурси користуються значним попитом серед користувачів. У роботі використано сучасні технології веброзробки та актуальні підходи до побудови клієнт-серверних систем. Програмний продукт має зручний інтерфейс, структуровану подачу інформації та забезпечує швидкий доступ до спортивних новин.

5. Негативні сторони роботи. У роботі недостатньо реалізовано персоналізацію контенту для користувачів. Також доцільно було б додати систему рекомендацій новин та можливість надсилання сповіщень про нові публікації.

6. Оцінка графічного оформлення та пояснювальної записки. Графічне оформлення виконано відповідно до тематики кваліфікаційної роботи та подано у вигляді схем, діаграм і рисунків. Пояснювальна записка оформлена згідно з вимогами чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому. Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки викладено послідовно, структуровано та зрозуміло. Графічні матеріали наочно демонструють особливості проєктування та реалізації програмного забезпечення. Розроблений вебзастосунок виконує поставлені завдання та може бути використаний на практиці.

8. Інші зауваження. Істотних зауважень до кваліфікаційної роботи немає.

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленим завданням та заслуговує на оцінку «добре» (80/С).

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи)

Капусенко Марія Вікторівна, доцент кафедри КІТС, ХНУ

«25» 05 2026 р. _____

(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.
Назва кваліфікаційної роботи: «Вебзастоссунок для перегляду та фільтрації спортивних новин»
Автор: Петрик Денис Максимович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Форкун Юрій Вікторович, кандидат технічних наук, доцент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована.	
3	Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:
Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:
1) у тексті кваліфікаційної роботи системою перевірки на плагіат Anti-Plagiarism виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках, у структурі змісту, назвах розділів/підрозділів, рамках форм, у назвах та URL-адресах публікацій переліку джерел посилання;
2) запозичення, виявлені у тексті роботи, є фрагментарними.
Загальна сумарна подібність у базі даних складає 2 % за символами та 14 % за лексемами. Крім того за результатами додаткового аналізу системи StrikePlagiarism коефіцієнт подібності 1 становить 5,5 %, коефіцієнт подібності 2 – 2,48 %. З урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 29.05.2026 р.

Завідувач кафедри — Підпис — Леонід БЕДРАТЮК (Ім'я, ПРІЗВИЩЕ)
Гарант освітньої програми — Підпис — Леонід БЕДРАТЮК (Ім'я, ПРІЗВИЩЕ)
Керівник кваліфікаційної роботи — Підпис — Юрій ФОРКУН (Ім'я, ПРІЗВИЩЕ)