Вінницький національний технічний університет
факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

КОМПЛЕКСНА МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ВЕБ-ЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ТРЕНУВАНЬ ТА ДІЄТИ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ. ЧАСТИНА 2. «КЛІЄНТСЬКА ЧАСТИНА З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ QWIK»

Виконав: студент 2 курсу, групи 1КІ-22м
спеціальності 123 — Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Іванов В.М.
(прізвище та ініціали)
Керівник к.т.н., доцент каф. ОТ
Городецька О.С.
(прізвище та ініціали)
«07» 12 2023 р.
Опонент к.т.н., доцент каф. ПЗ
Майданюк В.П.
(прізвище та ініціали)
«08» 12 2023 р.

Допущено до захисту
Зав. кафедри д.т.н., проф. Азаров О.Д.
«11» 12 2023 р.

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Галузь знань — Інформаційні технології

Освітній рівень — магістр

Спеціальність — 123 «Комп'ютерна інженерія»

Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ, д.т.н, проф.

Азаров О.Д.

«26» 09 2023року

З А В Д А Н Н Я

НА КОМПЛЕКСНУ МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Іванову Владиславу Миколайовичу

1 Тема роботи «Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik»» керівник роботи Городецька Оксана Степанівна, к.т.н., доцент, затверджені наказом вищого навчального закладу від **«26» 09 2023** року № **247**

2 Строк подання студентом роботи **9 грудня 2023** року.

3 Вихідні дані до роботи: Контент для наповнення веб-сторінок веб-застосунку для організації тренувань та дієти з інтеграцією штучного інтелекту, технології реалізації клієнтської частини.

4 Зміст текстової частини: аналітичний огляд та обгрунтування вибору технологій для розробки клієнтської частини веб-застосунку для організації тренувань та дієти з інтеграцією штучного інтелекту, вдосконалений метод відображення веб-сторінки, розробка клієнтської частини веб-застосунку, тестування клієнтської частини.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема процесу рендеренгу веб-сторінки різними методами, структурна схема процесу відображення веб-сторінки з гідратацією і її відсутністю, структурна схема вдосконаленого методу відображення веб-сторінки із припиненням і відновленням рендерингу, структурна схема клієнтської частини веб-застосунку.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Городецька О.С., к.т.н., доцент каф. ОТ		
5	Небава М.І., К.Е.Н., ПРОФЕСОР КАФ. ЕПВМ		

7 Дата видачі завдання **19.09.2023**.

8 Календарний план виконання КМКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз завдання	26.09.23	27.09.23	виконано
2	Аналіз технологій для розробки клієнстької частини веб-застосунку	28.09.23	05.10.23	виконано
3	Аналіз макету	05.10.23	20.10.23	виконано
4	Аналіз методів відображення веб-сторінок	20.10.23	30.10.23	виконано
5	Покращення методу відображення веб-сторінок	31.10.23	12.11.23	виконано
6	Аналіз вимог до клієнтської чатини веб-застоснку	12.11.23	15.11.23	виконано
7	Проектування клієнстької частини	15.11.23	20.11.23	виконано
8	Розробка клієнстької частини та налаштування взаємодії з сервером	20.11.23	25.11.23	виконано
9	Тестування та перевірка працездатності клієнтської частини веб-застосунку	25.11.23	26.11.23	виконано
10	Оформлення пояснювальної записки та ілюстративного матеріалу	26.11.23	05.12.23	виконано
11	Перевірка якості виконання комплексної магістрської роботи та усунення недоліків	05.12.23	08.12.23	виконано

Студент _____ Іванов В.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Городецька О.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.4

Іванов В.М. Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik». Комплексна магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна Інженерія, Вінниця: ВНТУ, 2023 — 107 с. На укр. Мові. Біблогр. 24 назв; рис: 30;

У комплексній магістрській кваліфікаційній роботі проведено аналітичний огляд технологій, які використовуються для розробки клієнтської частини веб-застосунку для організації дієти та тренувань. Обгрунтовано вибір фреймворку для розробки клієнтської частини веб-застосунку. Вдосконалено метод відображення веб-сторінок, що дало можливість підвищити швидкодію завантаження веб-сторінки. Розроблено структуру клієнтської частини веб-застосунку. Реалізовано клієнстьку частину веб-застосунку для організації тренувань та дієти фреймворком Qwik. Під час розробки було використано підхід поділу на компоненти для подальшого їх перевикористання. Організовано маршрутизацію у клієнтській частині веб-застосунку. Проведено тестування клієнтської частини веб-застосунку для організації тренувань та дієти. Здійснено порівняння продуктивності веб-застосунку розробленого на Qwik та React.

Ключові слова: веб-застосунок, клієнтська частина, фреймворк, Qwik, React, компонент.

ABSTRACT

УДК 004.4

Ivanov V.M. Web Application for Organizing Training and Diet with Artificial Intelligence Integration. Part 2. "Client-side using the Qwik framework." Comprehensive master's thesis in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2023 — 107 p. In Ukrainian. Bibliography: 24 titles; figures: 30.

The comprehensive master's thesis conducts an analytical overview of technologies used in developing the client-side of a web application for organizing diet and training. The choice of the Qwik framework for developing the client-side of the web application is justified. The method of displaying web pages has been improved, allowing for faster page loading. The structure of the client-side of the web application has been developed. The client-side of the web application for organizing training and diet has been implemented using the Qwik framework. The development employed a component-based approach for future reuse. Routing has been organized in the client-side of the web application. Testing of the client-side of the web application for organizing training and diet has been conducted. A performance comparison has been made between the web application developed on Qwik and React.

Keywords: web application, client-side, framework, Qwik, React, component.

ЗМІСТ

ВСТУП

У сучасному світі існує велика різноманітність методів для передачі інформації, але веб-застосунок стали одними з найпопулярніших та найефективніших джерел інформації. На сьогоднішній день, коли потрібно знайти інформацію, багато людей миттєво звертаються до Інтернету. Якщо під час пошуку користувач потрапляє на сайт із заплутаним інтерфейсом або обмеженою продуктивністю, то ймовірність того, що він швидко перейде на інший ресурс, зростає. Таким чином, важливо надавати належну увагу аспектам дизайну інтерфейсу користувача (UI) та користувацького досвіду (UX). Це сприятиме створенню веб-застосунків, який буде комфортним і зрозумілим для користувача.

Зазвичай доступ до веб-ресурсів можливий з різних типів пристроїв, включаючи як телефони з малими екранами, так і стаціонарні комп'ютери. Тому важливо розробляти веб-додатки з урахуванням їх адаптивності для різних пристроїв.

Крім того, важливим фактором є швидкодія сайту. Навіть якщо сайт має зручний та привабливий дизайн, якщо він завантажується повільно, то користувачі, ймовірно, шукатимуть альтернативні ресурси.

На сьогоднішній день існує чимало онлайн та оффлайн ресурсів, які надають розписане харчування та програми тренувань. Проте деякі застосунки не надають достатньо персоналізованих рекомендацій, що зменшує їх ефективність. Також якщо веб-застосунок не оптимізований, він завантажується повільно, що погіршує користувацький досвід. Тому розробка веб-застосунку для організації тренувань та дієти з підвищеною швидкодією є актуальною задачею. Користувачі отримують зручний інструмент для досягнення своїх цілей здоров’я та фітнесу, а розробники отримують можливість залучити широку аудиторію та створити позитивний вплив на життя людей.

Метою комплексної магістерської роботи є підвищення швидкодії завантаження клієнтської частини веб-застосунку для організації тренувань та дієти шляхом вдосконалення методу відображення веб-сторінок.

Задачі дослідження комплексної магістрської кваліфікаційної роботи:

— здійснити огляд та обгрунтування вибору технологій для розробки клієнтської частини;

— розробити структуру клієнтської частини веб-застосунку;

— вдосконалити метод відображення веб-сторінок;

— реалізувати клієнтську частину веб-застосунку;

— здійснити порівняльний аналіз продуктивності Qwik та React;

— здійснити тестування клієнтської частини.

Об’єкт дослідження — процес створення клієнтського інтерфейсу за допомогою популярних технологій, які підвищують продуктивність взаємодії користувача з веб-застосунком.

Предметом дослідження є методи рендерингу веб-сторінок та сучасні фреймворки для клієнтського інтерфейсу.

Новизна одержаних результатів — вдосконалено метод відображення веб-сторінки, в якому на відміну від існуючого, відсутня гідратація та використовується припинення рендерингу на сервері та його відновлення на стороні клієнта, що дає можливість підвищити швидкодію завантаження веб-сторінки.

Практичне значення одержаних результатів комплексної магістерської роботи: реалізована клієнтська частина веб-застосунку, який надає можливість створення дієт та програм тренування для індивідуальних потреб користувачів.

Апробація результатів роботи — зроблено доповіді на LI науково-технічній конференції підрозділів ВНТУ, 2022 р. та на Міжнародній науково-практичній інтернет-конференції Молодь в науці: дослідження, проблеми, перспективи (МН—2024).

Матеріали роботи доповідались та опубліковувались [1, 2]:

В. М. Іванов, О. В. Войцеховська / Аналіз технологій створення веб-клієнту за допомогою реактивного фреймворка React // Тези доповіді. Матеріали LI наукової-технічної конференції підрозділів Вінницького національного технічного університету. Вінниця 2022 р. Режим доступу: https://conferences.vntu.edu.ua /index.php/all-fitki/all-fitki-2022/author/submission/15736.

Городецька О. С., Войцеховська О. В., Іванов В. М. / Аналіз технологій розробки клієнтської частини веб-додатків // Матеріали Міжнародної науково-практичної інтернет-конференції Молодь в науці: дослідження, проблеми, перспективи (МН-2024). 2024 р. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/ viewFile/19089/15839

1 АНАЛІТИЧНИЙ ОГЛЯД ТА ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ТРЕНУВАНЬ ТА ДІЄТИ З ІНТЕГРАЦІЄЮ ШТУЧНОГО ІНТЕЛЕКТУ

Розробка клієнтської частини інтерфейсу користувача або фронт-енд розробка в даний час є дуже популярною. Велика кількість веб-застосунків, сукупність яких тільки зростає з року в рік, потребує великої кількості розробників для їх підтримання та розробки нових сайтів. Бек-енд розробник відповідає за розробку серверної частини веб-додатку, на якій відбуваються основні обрахунки та робота з даними. Бек-енд розробник працює разом з розробником баз даних, як результат їхньої співправці є можливість обмінюватись даними між клієнтською частиною та базою даних.

1.1 Базові технології розробки клієнтської частини

Кожен із сучасних веб-застосунків використовує комбінацію таких основних технологій, як HTML, CSS та JavaScript [3].

HTML, відомий як HyperText Markup Language, служить основною мовою розмітки для створення структури веб-сторінок. Вона визначає компоненти, такі як заголовки, абзаци, графічні елементи, посилання та інші частини контенту, що знаходяться на сторінці.

CSS, або Cascading Style Sheets, представляє собою мову, що визначає стилі та зовнішній вигляд веб-сторінок. За допомогою CSS можна задавати кольори, шрифти, відступи, розташування та інші параметри, щоб створити бажаний зовнішній вигляд веб-застосунку.

JavaScript використовується як мова програмування для додавання інтерактивності та динамічної поведінки на веб-сторінках. JavaScript дозволяє створювати такі функції, як слайд-шоу, виїзні меню, перевірку форм, анімацію та інші інтерактивні елементи.

1.1.1 Особливості мови гіпертекстової розмітки HTML

Інформація, що міститься у форматі HTML-документа, подібна до

звичайного текстового файлу, але має особливість у використанні символів, які інтерпретуються як розмітка. Ці спеціальні символи відомі як теги. Теги грають важливу роль у створенні структурованості документа та розділенні його на різні елементи, такі як параграфи, розділи, малюнки, таблиці, абзаци, списки, індекси, зміст та інше. Кожен елемент може також бути стилізованим, зокрема зміненою розміром шрифту, кольором тексту, наданим напівжирного чи курсивного виду.

Основною особливістю HTML є можливість використовувати гіперпосилання, завдяки яким створюються посилання та переходи з поточної веб-сторінки на інші документи, як локальні (документи, що розташовані на поточному комп'ютері або сервері), так і на ті, що розміщені на серверах у будь-якому куточку світу [4].

1.1.2 Каскадні таблиці стилів CSS та препроцесори CSS

CSS використовується для форматування зовнішнього вигляду веб-документів, написаних на HTML. За допомогою CSS можна налаштовувати параметри, такі як кольори тексту, шрифти, відстань між абзацами, розміри та розташування елементів, встановлювати фонові зображення або кольори, а також створювати адаптивний дизайн для відображення на різних типах пристроїв та розмірах екранів [5].

CSS-препроцесори, такі як Less, Sass і Stylus, додають можливості розширення стандартного CSS. Вони дозволяють використовувати різні корисні функції, такі як змішування (mixins), вкладені стилі, змінні, імпорт стилів прямо в інші файли тощо. Це сприяє більш організованому та компактному коду CSS, роблячи його більш читабельним та зручним для розробників. CSS-препроцесори також відомі як динамічні мови стилів.

Браузер спроектований таким чином, що він не може інтерпретувати синтаксис CSS-препроцесорів напряму, тому необхідно перетворити вихідний код в звичайний CSS. Для компіляції стилів у звичайний CSS існують три можливих способи:

— в браузері, за допомогою інструментів JavaScript;

— на серверній стороні, використовуючи мови програмування;

— локально на вашому комп'ютері, за допомогою спеціальних програм.

Sass використовує розширення .scss (наприклад, style.scss), Less використовує .less, а Stylus — .styl.

LESS — це препроцесор CSS, який прискорює процес оформлення HTML-сторінок сайту шляхом розширення можливостей звичайного CSS. Препроцесор LESS — це мова, яка, хоча і виглядає подібно до мови програмування, виражена набагато простіше, і вона значно розширює функціональність CSS-коду.

Використання препроцесора LESS вносить ряд переваг, основні з яких включають створення змінних, можливість вкладеності та імпорту стилів.

Щодо Stylus, цей препроцесор може бути менш популярним в порівнянні з іншими, але він все ж має свої особливості, які не поступаються іншим аналогам. Як і інші препроцесори, Stylus має численні переваги над звичайним CSS, включаючи можливість використовувати змінні, міксини, оператори, функції, імпорт стилів та інше [6].

— sass є повністю сумісним з усіма версіями CSS, завдяки уважному ставленню розробників до сумісності;

— sass пропонує більше можливостей, ніж будь-який інший препроцесор, що розширює можливості створення стилів;

— крім того, Sass активно підтримується та розробляється високотехнологічними компаніями та залучає багато розробників, що робить його надзвичайно життєздатним.

1.1.3 Огляд мови програмування JavaScript та порівняння її з мовою TipeScript

JavaScript — це мова програмування, яка надає можливість створювати динамічні та інтерактивні веб-сторінки. Вона дозволяє розробникам створювати та керувати оновленнями окремих блоків веб-сторінок, інтерактивними картами, анімацією графіки, прокруткою відео в медіапрогравачах і багато іншим [7].

JavaScript є інтерпретованою мовою програмування, що означає, що для її виконання необхідний інтерпретатор. У випадку веб-розробки, браузери виступають в ролі інтерпретаторів, і вони виконують JavaScript-код, який вбудований в веб-сторінки.

Майже будь-який веб-розробник повинен мати розуміння мови JavaScript, оскільки вона є невід'ємною частиною веб-розробки. Ця мова дозволяє розробникам динамічно змінювати вміст сторінок та надавати їм інтерактивні можливості.

За допомогою JavaScript також легко взаємодіяти з об'єктною моделлю документа (DOM) веб-сторінки. JavaScript DOM включає в себе численні функції, які спрощують роботу з елементами сторінки, включаючи їх знаходження, створення, видалення, зміну вмісту, доступ до стилів та зміну їх параметрів. Це також дозволяє надавати логіку елементам, таким як зміна вмісту, стилів та поведінки при взаємодії користувача зі сторінкою.

TypeScript — це розширення мови JavaScript, що означає, що весь ваш існуючий JavaScript-код можна використовувати без змін. TypeScript додає статичну типізацію, яка допомагає виявляти помилки на етапі компіляції та робить додатки більш надійними. Проте використання TypeScript може вимагати певного часу для вивчення та адаптації до його особливостей [8].

З іншого боку, JavaScript є базовою мовою програмування для веб-розробки. Він використовує динамічну типізацію, що означає, що типи змінних визначаються автоматично під час виконання. JavaScript надає широкий вибір інструментів і бібліотек, і дозволяє швидко прототипувати та розробляти додатки.

Переваги TipeScript перед іншими мовами програмування:

— по-перше, статична типізація допомагає виявляти помилки на етапі розробки, що дозволяє попереджати безліч поширених помилок, пов'язаних з типами даних, і підвищує загальну надійність коду, перенаправлення (компонент <Redirect>);

— по-друге, TypeScript має високий рівень підтримки інструментів розробки, таких як інтегровані середовища розробки (IDE) та текстові редактори,

завдяки статичній типізації, IDE можуть надавати функції автодоповнення, підказки за типами та інші корисні інструменти, які спрощують процес розробки;

— крім того, TypeScript має високоякісну документацію та активну спільноту розробників, що дає можливість знайти допомогу та рішення проблем в онлайн-ресурсах, форумах та блогах.

Основні причини використання TypeScript:

— великі проєкти та командна розробка: якщо у великій команді розробників працює кілька спеціалістів над різними модулями веб-додатка, TypeScript зі статичною типізацією допомагає кожному розробнику чітко розуміти очікувану структуру даних та типи аргументів функцій, спрощуючи спільну роботу та запобігаючи можливим помилкам інтеграції;

— підтримка та оновлення коду: наприклад, ви маєте великий веб-додаток, написаний на JavaScript, і розробляєте його далі, додаючи TypeScript, додавання типів до наявного коду допомагає виявити потенційні помилки та спрощує підтримку, оскільки типи надають чітке розуміння очікуваної структури даних;

— інтеграція з фреймворками та бібліотеками: наприклад, ви розробляєте веб-додаток на React і використовуєте TypeScript, з типізованими оголошеннями для React ви отримуєте підказки типів під час розробки компонентів, що допомагає уникнути помилок у використанні пропсів і забезпечує надійну взаємодію між компонентами;

— покращення продуктивності та надійності: якщо у вашому додатку є складна логіка, що включає безліч функцій і класів, TypeScript дозволяє визначити типи вхідних та вихідних даних для кожної функції та методу, що допомагає запобігти помилкам при використанні неправильних аргументів або невірних типів даних, тим самим підвищуючи продуктивність та надійність додатка.

1.2 Обгрунтування вибору технології для розробки клієнтської частини веб-застосунку

Сучасна індустрія веб-розробки пропонує багатий вибір фреймворків та інструментів, які розробники використовують для створення веб-додатків. Це сфера, що постійно розвивається, і нові інструменти завжди привертають увагу спеціалістів.

Один із таких відносно нових учасників у цій галузі — "Qwik". Цей фреймворк наразі активно привертає увагу та стає предметом інтересу для веб-розробників. Його актуальність полягає у властивостях і можливостях, які він пропонує для розробки веб-додатків.

1.2.1 Огляд технології Qwik

Qwik — це сучасний веб-фреймворк, який дозволяє розробникам створювати високопродуктивні односторінкові додатки (SPA) та статичні сайти (SSG) з використанням звичайних технологій веб-розробки, таких як JavaScript, HTML і CSS. Фреймворк Qwik пропонує інноваційний підхід до розробки, покладаючи основний акцент на продуктивність, безпеку та простоту розробки. Ось декілька ключових аспектів фреймворку Qwik:

— висока продуктивність: Qwik прагне надавати найкращу продуктивність завдяки підтримці швидкого відтворення сторінок, кешуванню та іншим оптимізаціям для зменшення завантаження сторінок;

— простота розробки: Qwik надає простий та легкий спосіб розробки веб-додатків. Розробники можуть використовувати стандартні технології, такі як JavaScript та HTML, і з легкістю вивчати фреймворк;

— швидке відтворення сторінок: фреймворк Qwik надає можливість швидкого відтворення окремих сторінок, замість повного перезавантаження додатку, що робить взаємодію користувача більш зручною;

— SSR (Server-Side Rendering): Qwik підтримує серверний рендеринг, що дозволяє відображати сторінки на сервері для поліпшення продуктивності та оптимізації SEO;

— розділення додатків: Qwik допомагає розділити ваш додаток на окремі частини, що сприяє більшій розширюваності і обслуговуваності;

— надійність і безпека: фреймворк Qwik розроблений з урахуванням вимог щодо безпеки та надійності, що дозволяє створювати стійкі до атак додатки;

— загалом, фреймворк Qwik дозволяє розробникам швидко створювати продуктивні та безпечні веб-додатки та сайти, використовуючи простий та ефективний підхід до розробки [9].

1.2.1.1 Модуль маршрутизації Qwik

Маршрутизація в Qwik відрізняється від підходів, використованих у React. У Qwik сторінки розділяються на окремі "компоненти сторінок," і маршрутизація здійснюється за допомогою параметрів шляху.

Маршрути в Qwik створюються на основі структури каталогів, де імена каталогів використовуються для відповідності вхідним запитам та сторінкам або точкам доступу. Маршрутизація в Qwik ґрунтується на структурі файлової системи, що означає, що створення нового файлу index.tsx у каталозі src/routes/ створить новий маршрут.

Також можливо створювати динамічні маршрути, додавши каталог із параметром [Id] у шляху маршруту (рисунок 1.1).

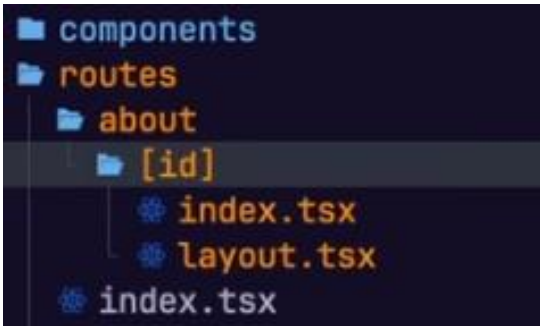

Рисунок 1.1 — Приклад створення сторінки з динамічним параметром

Де [Id] — це параметр шляху, який можна використовувати для відображення різних продуктів. Параметри шляху передаються в компонент About для обробки і відображення конкретного елементу.

Отже, маршрутизація в Qwik базується на розділенні сторінок на компоненти і використанні параметрів шляху для навігації між ними. Цей підхід робить код більш структурованим і гнучким у відображенні різних сторінок вашого додатку [10].

1.2.1.2 Стан компонентів Qwik

Стан в Qwik може бути розглянутий у двох основних аспектах: реактивний стан (Reactive State) і статичний стан (Static State).

Реактивний стан створюється за допомогою функцій useSignal() та useStore(). Він є динамічним і може автоматично оновлювати компоненти, які залежать від нього.

useSignal — це функція що використовується для створення реактивного сигналу з початковим значенням. Реактивний сигнал має властивість .value, і зміна значення цієї властивості призводить до автоматичного оновлення компонентів, які використовують цей сигнал. Приклад використання функції useSignal зображено на лістингу 1.1.

Лістинг 1.1 — Приклад використання функції useSignal

```
import { component$, useSignal } from '@builder.io/qwik';
export default component$(() => {
  const count = useSignal(0);
  return (
    <button onClick$={() => count.value++}>
      Increment {count.value}
    </button>
  );
});
```

Функція useStore дуже схожа на useSignal, але дозволяє створювати реактивний об'єкт. Реактивний об'єкт може включати в себе вкладені об'єкти та

масиви, і зміни в будь-якому з цих об'єктів призводять до автоматичного оновлення компонентів.

Статичний стан (Static State): Статичний стан охоплює будь-які дані, які можуть бути серіалізовані, такі як рядки, числа, об'єкти та масиви. Він представляє дані, які не підлягають реактивному оновленню і можуть бути використані у компонентах для відображення даних.

Реактивний стан може бути використаний для створення і керування динамічними компонентами, які автоматично реагують на зміни даних. Статичний стан може бути використаний для зберігання незмінних даних, таких як заголовки, сталі рядки тощо.

Стан у Qwik не обов'язково є локальним станом компонентів; він може бути доступним для будь-якого компонента в додатку. Це робить Qwik потужним інструментом для управління станом та реакцією на зміни даних у великих додатках [11].

1.2.2 Огляд технології React

React — це бібліотека JavaScript, яка використовується для створення інтерактивних та ефективних користувацьких інтерфейсів для веб-додатків. Ця технологія дозволяє розробникам легко створювати складні веб-сторінки та додатки, розділяючи їх на компоненти та керуючи їх станом. React розроблений так, щоб бути незалежним від конкретного браузера і може використовуватися як для створення веб-додатків, так і для мобільних додатків за допомогою React Native. Однією з ключових особливостей React є можливість перевикористовувати компоненти в різних частинах проекту, що спрощує розробку і покращує продуктивність [12].

1.2.2.1 Virtual DOM

DOM (Document Object Model) — це структуроване подання всіх HTML-елементів, присутніх на веб-сторінці або в додатку. При будь-якій зміні інтерфейсу

DOM також оновлюється, щоб відобразити ці зміни. Проте маніпулювання DOM може негативно позначитися на продуктивності [13].

DOM відображається як структура дерева даних, тому оновлення та зміни самого DOM відбуваються досить швидко. Але після оновлення потрібно повторно завантажувати (рендерити) змінений елемент та всі його дочірні елементи для оновлення інтерфейсу користувача. Цей процес повторного рендерингу є досить повільним. Отже, чим більше компонентів інтерфейсу користувача, тим більше ресурсів потрібно для оновлення DOM, що негативно впливає на продуктивність.

У React, замість прямої взаємодії з реальним DOM, розробник працює з його віртуальною копією. Розробник може внести зміни до цієї копії відповідно до потреб, а потім застосувати ці зміни до реального DOM.

Під час цього процесу проводиться порівняння DOM-дерева з його віртуальною копією, визначається різниця та виконується оновлення того, що було змінено. Такий підхід працює набагато швидше, оскільки він уникне використання ресурсів на повний рендеринг реального DOM

1.2.2.2 Модуль маршрутизації React Router

Кожен веб-застосунок з багатьма сторінками повинен мати систему маршрутизації, щоб користувач завжди знав своє поточне місцезнаходження на сайті. Цю інформацію він може бачити у адресній стрічці браузера. Таким чином, сайт повинен здати взнати, яка адреса URL відповідає кожній сторінці [14].

Зокрема, сайт повинен підтримувати історію, що дозволить користувачу легко переходити між різними сторінками веб-застосунку. Модуль маршрутизації забезпечує сторінкам відповідні URL-адреси і автоматично використовує історію користувача за замовчуванням.

Цей модуль маршрутизації є популярним та досить простим у використанні. Він надає такі основні можливості:

— навігація при кліку (за допомогою компонента <Link>);
— перенаправлення (за допомогою компонента <Redirect>);
— маршрутизація (за допомогою компонента <Route>);

— керування історією (за допомогою властивості history).

Ця технологія дозволяє переходити між сторінками без повного перезавантаження додатку. React Router використовує динамічну маршрутизацію, що дозволяє змінювати елементи інтерфейсу користувача без повторного завантаження статичних файлів

1.2.3 Порівняння технології Qwik та React

React і Qwik — це дві різні технології для розробки користувацьких інтерфейсів. React використовує віртуальний DOM для оптимізації оновлень інтерфейсу, тоді як Qwik обходиться без нього, фокусуючись на обмеженні використання JavaScript та уникненні гідратації.

Одностороннє прив'язування даних в React означає, що зміна стану моделі призводить до оновлення елементів інтерфейсу. У випадку Qwik ставиться акцент на односторонньому зв'язку, де стан моделі синхронізується з інтерфейсом.

React — це бібліотека з великою кількістю зовнішніх бібліотек і інструментів для розширення функціональності. Qwik — це фреймворк, спрямований на досягнення максимальної продуктивності, і має менший набір функціональності.

React надає розробникам більше гнучкості у розширенні функціональності, в той час як Qwik спрямований на покращення продуктивності і зменшення розміру інструментарію.

Qwik зорієнтований на досягнення максимальної продуктивності і пропонує рішення для уникнення завантаження JavaScript. React також може бути оптимізований для продуктивності, але вимагає більше додаткової роботи та оптимізації.

React має одностороннє прив'язування даних, що означає, що зміна стану моделі призводить до оновлення відповідних частин інтерфейсу. Qwik, натомість, спрямований на відсутність гідратації та подовження інтерактивності сторінки, що значно пришвидшує роботу веб-додатку.

В цілому ці дві технології схожі але є основна відмінність між ними — це те що фреймворк Qwik рендерить сторінку на сервервері, але намагається уникнути

гідрації. В результаті це дає високу продуктивність і швидке завантаження сторінок для користувача.

Тому, проаналізувавши вимоги та завдання для реалізації клієнтської частини веб-платформи, було обрано фреймворк Qwik. Фактори, які враховувались при виборі, включають швидкодію та ефективність, архітектурну ясність, легкість впровадження та розширення, а також інтеграцію з іншими частинами системи. Qwik, як сучасний фреймворк, надав позитивний відгук у всіх цих аспектах, дозволяючи забезпечити швидку та надійну роботу клієнтської частини веб-платформи.

1.3 Аналіз засобів для комплектування модулів програмного продукту

Для кращого розуміння переваг використання інструментів зі збірки (комплектування) в проектах, давайте розглянемо такий сценарій. Уявімо, що маємо великий проект з великою кількістю складових, над яким працюють декілька розробників, кожен з яких має свої власні файли JavaScript і CSS. Результатом розробки є велика кількість різних файлів, неоптимізованих зображень та CSS-коду на тисячі рядків.

Саме в такому випадку і стає актуальним використання інструментів зі збірки, які допомагають об'єднати всі файли CSS, SCSS, або Stylus в один мінімізований CSS-файл. Такий же підхід використовується для скриптів, стискання зображень та організації їх в потрібні папки. У результаті отримуємо оптимізований проект, який має значно менший обсяг.

Далі цей готовий проект завантажується на сервер. Як результат, на сервері зберігається оптимізований продукт, і кінцевий користувач отримує той варіант, який не містить зайвих елементів і завантажується швидко та ефективно.

1.3.1 Огляд технології комплектування Webpack

Webpack є сучасним комплектувальником модулів для JavaScript-додатків. Цей інструмент проводить аналіз усіх модулів у додатку, визначає їх залежності і

об'єднує їх у один або декілька бандлерів (bundles), на які можна посилатися з файлу index.html.

Зазвичай, при розробці JavaScript-додатків, код розділяють на окремі модулі (Modules). У файлі index.html потрібно вказати посилання на кожен із цих модулів. При такому підході важливо не лише враховувати кожен фрагмент коду, але і дотримуватися правильної послідовності підключення модулів. Наприклад, якщо завантажити код, що залежить від React, перед самим React, це може призвести до збою додатку. Webpack вирішує ці проблеми, оскільки ви можете не турбуватися про послідовність підключення модулів.

Проте збірка модулів — це лише одна з можливостей, яку надає webpack. При необхідності він може виконувати додаткові перетворення модулів перед додаванням їх у бандл. Наприклад, ви можете використовувати файли з розширенням SCSS, які потрібно перетворити в CSS. Налаштування таких перетворень у webpack здійснюється зручніше, ніж в інших інструментах комплектування додатків [15].

1.3.2 Огляд технології комплектування Vite

Технологія комплектування Vite відіграє важливу роль у розробці сучасних веб-додатків. Вона пропонує інноваційний підхід до компіляції та виконання коду, що робить процес розробки більш ефективним і швидким.

Зазвичай під час створення веб-додатків, особливо на JavaScript, код розділяють на окремі модулі. У файлі index.html необхідно вказати посилання на кожен з цих модулів. Це вимагає правильної організації та послідовності підключення модулів, щоб уникнути проблем залежностей. Vite спрощує цей процес, роблячи його автоматизованим та швидким.

Окрім об'єднання модулів, Vite може виконувати інші корисні завдання, такі як оптимізація зображень, компіляція коду з інших мов програмування, і багато інших. Все це робить Vite потужним інструментом для розробників, допомагаючи їм прискорити розробку та оптимізувати робочий процес [16].

1.3.3 Порівняння технологій Vite та Webpack

Комплектувальник Vite має ряд переваг над комплектувальником Webpack:

— швидкість розробки;

— конфігурація за замовчуванням;

— швидкість збірки;

— оптимізація для продакшену.

Однією з основних переваг Vite є надзвичайно швидкий час перезавантаження під час розробки. Він використовує сучасну систему "HMR" (Hot Module Replacement), яка дозволяє змінювати код без перезавантаження сторінки. У порівнянні з Webpack, Vite забезпечує набагато більш швидку зміну та перегляд результатів миттєво;

Webpack вимагає складної налаштування для початку роботи з проектом, в той час як Vite надає конфігурацію "з коробки". Ви можете почати розробку без необхідності великих налаштувань, що робить процес старту значно простішим;

Vite забезпечує надзвичайно швидкий час збірки завдяки використанню модульних систем ECMAScript (ES) та використанню сучасного браузера під час розробки. В порівнянні з Webpack, який може бути повільним при великих проектах, Vite працює значно швидше;

Хоча Vite підходить для розробки, він також надає засоби для оптимізації проекту для продакшену. Ви можете легко налаштувати інструмент для мінімізації, об'єднання файлів та кешування ресурсів для отримання оптимізованої версії додатку.

Проаналізувавши обидві технології комплектування в роботі обрана технологія Vite, оскільки швидкість розробки стала ключовим критерієм при виборі, і система "HMR" в поєднанні зі значною швидкістю перезавантаження дозволяє ефективно працювати з кодом без затримок.

Також Vite може бути ефективно використаний для оптимізації проекту для продакшену за допомогою інструментів, таких як мінімізація, об'єднання файлів та кешування ресурсів.

2 ВДОСКОНАЛЕНИЙ МЕТОД ВІДОБРАЖЕННЯ ВЕБ-СТОРІНКИ

Згідно з даними web.dev, швидкість відображення веб-сторінок визначає, наскільки ефективним є веб-додаток та які прибутки він приносить [17]. Приклади з реального життя показують, що навіть невеликі зміни в швидкості завантаження сторінок можуть призвести до значних приростів у конверсії, продажах та виведенні сайту на новий рівень. Таким чином, збільшення швидкості відображення веб-сторінки є важливою та актуальною задачею.

2.1 Метод відображення та реалізація рендерингу веб-сторінок

Більшість популярних фреймворків таких як React, Angular, Vue, Next використовують такі методи рендерингу, як SSR та CSR. Результатами досліджень було встановлено, що два основні методи відображення веб-сторінок, а саме Серверний рендеринг (SSR) та Клієнтський рендеринг (CSR), володіють власними унікальними перевагами і особливостями.

Схема процесу рендеренгу веб-сторінки різними методами наведена на рис. 2.1. Основна відмінність між SSR та CSR полягає в тому, де відбувається генерація HTML сторінок. У випадку SSR вона відбувається на сервері, а в CSR — на клієнтському боці. SSR зазвичай надає більшу швидкість завантаження та покращену пошукову оптимізацію, а CSR дозволяє ефективно створювати інтерактивні додатки [18].

Метод рендерингу CSR (Клієнтський Рендеринг) передбачає отримання пустого файлу HTML клієнтом при завантаженні сторінки. Після отримання HTML, браузер клієнта повинен запитати сервер щодо CSS-стилів та JavaScript-бібліотеки React. Після завантаження всіх необхідних ресурсів, виконується JavaScript-код, і тільки після цього сторінка рендериться та стає інтерактивною.

У випадку методу рендерингу SSR (Серверний Рендеринг), браузер одразу отримує готовий HTML-код з сервера, що містить всю необхідну інформацію для відображення сторінки.

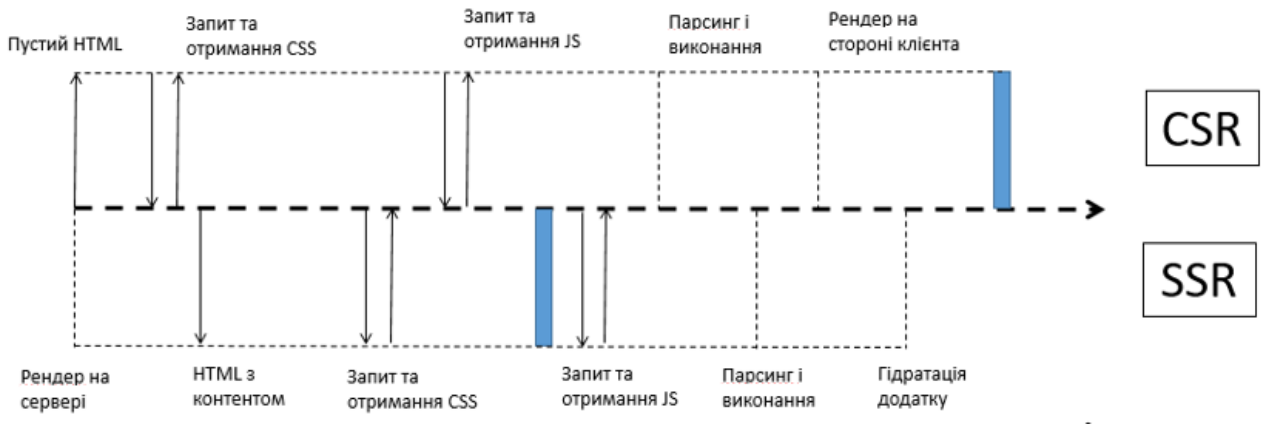

Рисунок 2.1 — Схема процесу рендеренгу сторінки різними методами

Це дозволяє показати сторінку користувачу швидше, оскільки не потрібно чекати завантаження JavaScript-бібліотеки для інтерактивності

2.1.1 Реалізація відновлення

У роботі запропоновано використати метод відновлення, який дозволяє зупиняти виконання додатку на сервері і відновлювати його виконання на клієнтському боці без необхідності повторного завантаження і відтворення всього коду застосунку [19]. Це робить процес рендерингу та виконання додатків більш ефективним та продуктивним в порівнянні з гідратацією.

На рис. 2.2 наведено структурну схему процесу відображення веб-сторінки з гідратацією та її відсутністю. Під час гідратації веб-сторінки відбувається асинхронне завантаження та відображення додаткового контенту або функціональності на сторінці, після того як базовий HTML та CSS вже були завантажені.

Спочатку веб-браузер відправляє запит на сервер для отримання HTML-документу. Сервер обробляє запит та відправляє назад HTML-код, який містить основну структуру сторінки.

Після отримання HTML-документу, браузер починає завантажувати всі скрипти JavaScript, які вказані у тезі <script> в HTML-коді. Це можуть бути як внутрішні скрипти, так і зовнішні файли, на які є посилання.

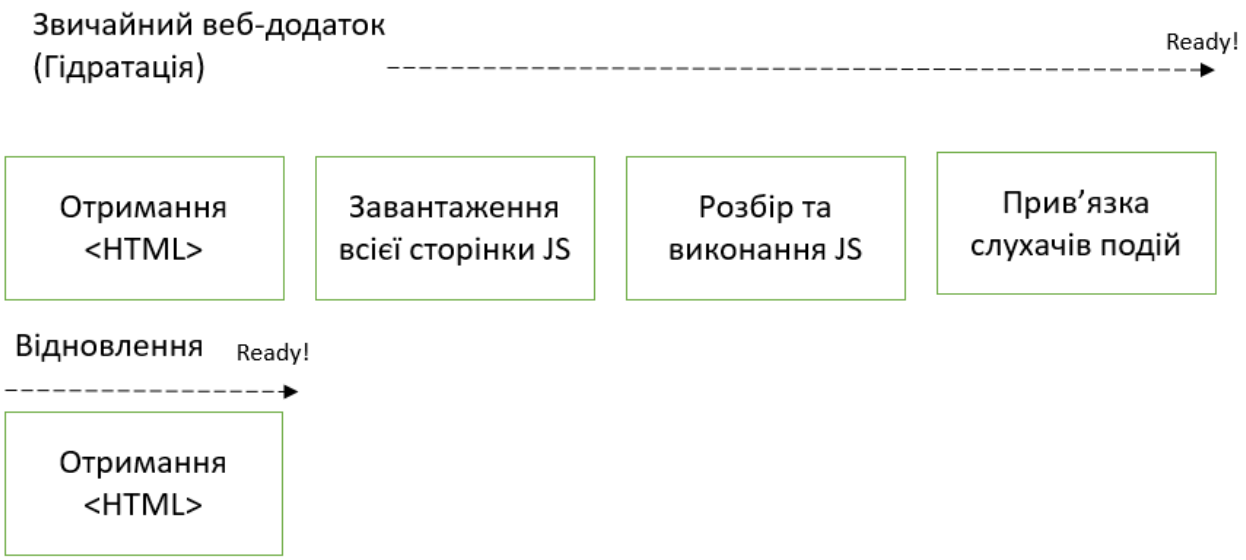

Рисунок 2.2 — Структурна схема процесу відображення веб-сторінки з гідратацією та її відсутністю

Коли всі скрипти завантажені, браузер розбирає і виконує JavaScript-код. Цей етап включає інтерпретацію та виконання інструкцій з коду, створення об'єктів та взаємодію з DOM (Document Object Model).

Також під час виконання JavaScript можуть бути встановлені слухачі подій на різних елементах сторінки. Наприклад, це може бути обробка подій клікання, введення тексту, руху миші тощо. Слухачі подій визначають, як сторінка повинна реагувати на взаємодію користувача.

В роботі запропоновано використовувати метод відображення веб-сторінки, в якому відсутня гідратація. При цьому додаток на сервері зупиняє виконання на певному етапі, і цей стан переноситься на клієнтську сторону, де він може бути відновлений. Таким чином, на клієнті виконання додатку просто продовжується з того місця, де воно було зупинено на сервері. І найважливіше, це відбувається без необхідності повторного завантаження всього коду (гідратації).

Можливість відновлення дозволяє досягти високої продуктивності та миттєвого запуску додатків. Коли користувач взаємодіє з додатком, він одразу отримує відповідь та інтерактивність, без зайвих затримок на завантаження коду.

2.1.2 Метод рендерингу

Концепція "відновлення" представляє собою інноваційний підхід до рендерингу веб-додатків. Основна ідея полягає в максимальному використанні можливостей серверного рендерингу, при цьому решта процесу може бути відновлена на стороні клієнта.

В роботі запропоновано вдосконалений метод відображення веб-сорінок із припиненням і відновленням рендерингу, структура якого зображена на рис. 2.3.

При отриманні запиту від користувача проміжний сервер ініціює процес рендерингу, формуючи серіалізований стан додатка, обробники подій, HTML-коди компонентів та фрагменти коду. Сервер надсилає клієнту відповідь у вигляді HTML-коду сторінки та серіалізованого стану.

Ліниве завантаження включає у себе відкладене завантаження компонентів, слухачів подій та HTML, використовуючи їх лише тоді, коли вони є необхідними для відображення конкретної частини сторінки. Замість того, щоб попередньо завантажувати всі ресурси при старті, динамічно завантажує лише ті компоненти та дані, які вперше потрібні користувачу. Це дозволяє зменшити час завантаження сторінки та зробити її більш ефективною для користувача, особливо в умовах обмеженої швидкості Інтернету або на мобільних пристроях.

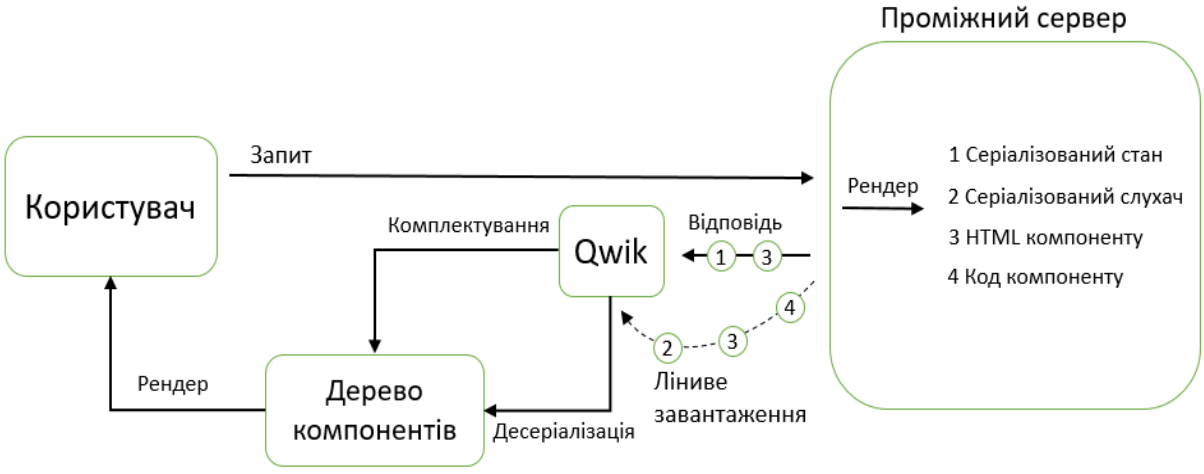

Рисунок 2.3 — Структурна схема вдосконаленого методу відображення веб-сторінки із припиненням і відновленням рендерингу

На клієнті браузер обробляє отриману сторінку і відображає інтерфейс. Qwik десеріалізує стан після завантаження сторінки, що містить локальні стани кожного компонента. Компоненти з відкладеним завантаженням можуть бути динамічно імпортовані, адже вони можуть посилатися на десеріалізований стан для своїх локальних станів.

Також Qwik серіалізує обробники подій, що дозволяє їх динамічно завантажувати та використовувати при взаємодії з користувачем.

У відгуках на взаємодію користувачів з інтерактивними елементами, Qwik використовує посилання для динамічного завантаження фрагмента з сервера та виклику події за допомогою обробника подій усередині фрагмента.

2.1.3 Концепція реактивності

Qwik використовує концепцію реактивності для відстеження взаємозв'язків між станом і компонентами. Це означає, що Qwik знає, які компоненти підписані на конкретний стан. Ця інформація дозволяє Qwik оновлювати лише ті компоненти, які дійсно пов'язані зі змінами у стані, уникнувши повного перерендерингу всього дерева компонентів.

Зазвичай, без детальної реактивності, зміна стану призвела б до повторного рендерингу всіх компонентів, включаючи навіть ті, які не пов'язані зі змінами стану. Однак завдяки реактивності в Qwik, лише відповідні компоненти оновлюються при зміні стану, що підвищує продуктивність та ефективність додатка.

Важливо відзначити, що Qwik не використовує виявлення змін, подібно до Angular. Замість цього Qwik використовує сигнали для безпосереднього оновлення шаблонів компонентів при зміні стану, що дозволяє уникнути необхідності виснажливо перевіряти весь стан на наявність змін [20].

Компоненти Qwik не вимагають послідовного порядку відтворення. Це означає, що компонент може бути відтвореним незалежно від статусу його батька або дочірніх компонентів. Це важливо, оскільки дозволяє Qwik відтворювати лише

ті компоненти, які потребують оновлення через зміни стану, і уникнути повного перерендерингу всього дерева компонентів.

Коли компоненти відтворюються, вони мають доступ до своїх атрибутів через реквізити. Оскільки компоненти можуть бути відтвореними незалежно від їх батьківських компонентів, реквізити повинні бути серіалізованими, щоб забезпечити правильну взаємодію між ними.

2.2 Використання компонентів в Qwik

В Qwik компонент — це багаторазово використовуваний фрагмент коду, який відповідає за структуру та поведінку певної частини веб-додатка. Кожен компонент може мати власну роль та відповідальність, і ці компоненти можуть використовуватися для побудови складних інтерфейсів та додавання функціональності [21].

Компоненти Qwik базуються на трьох основних принципах:

— оптимізація;

— відновлення;

— реактивність.

Qwik автоматично оптимізує завантаження та відображення компонентів. Кожен компонент розбивається на окремі фрагменти, що завантажуються відкладено. Це означає, що лише необхідні частини компонентів завантажуються, що сприяє швидкості та ефективності додатка.

Qwik дозволяє створювати компоненти, які можуть бути створені на сервері та продовжувати виконання на клієнті. Це надає можливість покращити продуктивність та відкриває двері для серверного рендерингу.

Компоненти в Qwik є реактивними, що означає, що вони автоматично оновлюються після зміни стану. Всі зміни, які впливають на компонент, спричиняють автоматичне оновлення, що спрощує управління станом та відображенням.

Ці принципи роблять Qwik компоненти потужними та ефективними будівельними блоками для створення високопродуктивних веб-додатків.

2.2.1 Функція component$

component$ — це ключова функція в Qwik, призначена для створення компонентів. Вона дозволяє визначити логіку та інтерфейс користувача, які будуть відображені на сторінці вашого додатку. Компоненти створені за допомогою component$ володіють рядом важливих властивостей, таких як відкладене завантаження, можливість відновлення, реактивність та автоматична оптимізація завантаження, які роблять їх потужними будівельними блоками для розробки продуктивних та швидких веб-додатків.

Дана функція повертає JSX (JavaScript XML), який представляє опис того, як виглядає інтерфейс користувача частини вашого додатку. JSX включає в себе HTML-подібну розмітку та може містити компоненти, логіку, властивості та обробники подій для створення візуальної частини вашого додатку. Приклад створення компоненту можна побачити на лістингу 2.1.

Лістинг 2.1 — Створення компоненту

```
import { component$ } from '@builder.io/qwik';
export default component$(() => {
  return <div>Hello World!</div>;
});
```

Символ $ у component$() в Qwik є ключовим, оскільки він визначає, як оптимізатор Qwik розділяє компоненти на окремі частини. Ці окремі частини, або фрагменти, завантажуються лише в тому випадку, якщо вони дійсно потрібні на поточній сторінці. Іншими словами, клієнтська частина не завантажуєте заздалегідь всі компоненти, а лише ті, які необхідні для даної конкретної сторінки.

2.2.2 Відкладене завантаження

Відкладене завантаження в Qwik — це важлива концепція, яка сприяє оптимізації додатків та покращує їх продуктивність.

Однією з ключових концепцій в Qwik є розділення компонентів на окремі частини, які завантажуються лише в тому випадку, коли користувач дійсно

взаємодіє з ними на сторінці. Це важливо для оптимізації завантаження додатка, оскільки не завжди потрібно завантажувати всі компоненти разом.

Для реалізації відкладеного завантаження Qwik використовує маркери, такі як <!--qv--> та <!--/qv--> (рисунок 2.4). Ці маркери вказують, де саме на сторінці потрібно вставити вміст відкладених компонентів після їх завантаження. Вони допомагають оптимізатору визначити, де розміщувати вміст компонентів відповідно до їх контексту на сторінці.

Рисунок 2.4 — Маркери завантаження компонентів

Використовуючи дану концепцію, Qwik дозволяє розбити додаток на окремі частини та завантажувати їх лише тоді, коли це дійсно необхідно, що робить додаток більш продуктивним і швидким у використанні.

2.2.3 Композиція компонентів

Композиція компонентів — це ключова концепція розробки в Qwik, яка дозволяє поєднувати декілька компонентів, щоб створити складні інтерфейси та додатки. Іншими словами, ви можете будувати додаток, комбінуючи невеликі та прості компоненти, які мають обмежену відповідальність, в одному інтерфейсі.

Поєднання компонентів для створення складних інтерфейсів: Qwik дозволяє поєднувати компоненти, які виконують певні функції, для створення складних інтерфейсів. Наприклад можна використовувати окремі компоненти для заголовку, абзаців, списків тощо, і потім об'єднувати їх, щоб створити сторінку або додаток (лістинг 2.2).

Лістинг 2.2 — Приклад поєднання компонентів

```
import { component$ } from '@builder.io/qwik';
export const Child = component$(() => {
  return <p>child</p>;
});
export const Parent = component$(() => {
  return (
    <section>
      <Child />
    </section>
  );
});
export default Parent;
```

Вбудовані компоненти: Окрім звичайних компонентів, Qwik також підтримує вбудовані компоненти, які можна використовувати в батьківських компонентах. Вбудовані компоненти — це легкі компоненти, які включені безпосередньо в батьківський компонент і не можуть бути завантажені відокремлено. Вони корисні для створення малих фрагментів інтерфейсу, таких як кнопки або інші елементи (лістинг 2.3).

Лістинг 2.3 — Приклад вбудованого компоненту

```
export const ParentComponent = component$(() => {
  return (
    <div>
      <MyButton text="Click me" />
    </div>
  );
```

Вбудовані компоненти полегшують використання і обслуговування одноразових елементів інтерфейсу, не потребуючи окремого завантаження так як вони є частиною фреймворку Qwik.

2.2.4 Реалізація стану в компонентах

Реалізація стану компонентів є важливою частиною розробки будь-якого додатку. У Qwik існує два типи стану: реактивний і статичний.

Статичний стан (Static state) — це будь-яке значення, яке може бути серіалізованим, таке як рядок, число, об'єкт або масив.

Реактивний стан (Reactive state), навпаки, створюється за допомогою функцій useSignal() або useStore(). Реактивний стан відрізняється тим, що зміни в ньому автоматично сповіщають усі компоненти, які залежать від нього. Важливо відзначити, що в Qwik стан — це не обов'язково локальний стан компоненту; це стан додатка, який може бути інстанційований будь-яким компонентом [22].

Функція useSignal()` використовується для створення реактивного сигналу (форми стану). Вона приймає початкове значення і повертає реактивний сигнал (лістинг 2.4). У цьому прикладі функція useSignal() створює реактивний сигнал, що відповідає лічильнику. Зміни значення count.value автоматично відображаються в компоненті.

Лістинг 2.4 — Використання функції useSignal

```
import { component$, useSignal } from '@builder.io/qwik';
export default component$(() => {
  const count = useSignal(0);
  return (
    <button onClick$={() => count.value++}>
      Increment {count.value}
    </button>
  );
});
```

Функція useStore() працює схоже до useSignal()`, але приймає об'єкт як початкове значення, і реактивність розповсюджується на вкладені об'єкти та масиви за замовчуванням (лістинг 2.5).

Лістинг 2.5 — Використання функції useStore

```
import { component$, useStore } from '@builder.io/qwik';
export default component$(() => {
  const state = useStore({ count: 0 });
  return (
    <>
      <button onClick$={() => state.count++}>Increment</button>
      <p>Count: {state.count}</p>
    </>
  );
});
```

Методи useStore() також включають в себе глибоку реактивність, що означає, що масиви та об'єкти в межах стору також є реактивними. Таким чином, зміни в глибоких структурах автоматично відобразяться в компонентах.

2.3 Реалізація життєвого циклу компонентів

В Qwik, життєвий цикл компонента розпочинається з моменту його створення і включає в себе різні етапи, які можуть бути виконані як на сервері, так і на клієнті. Порядок цих етапів однаковий як на сервері, так і на клієнті. Все це можливо завдяки принципу "повернення до початку" (resumability), коли виконання коду на сервері "призупиняється" і потім "продовжується" на клієнті [23].

У Qwik існує всього три етапи життєвого циклу компонента:

— завдання (Task);

— відображення (Render);

— видиме завдання (VisibleTask).

Виконується до відображення і при зміні відслідковуваного стану. Завдання можуть бути асинхронними. Вони можуть виконуватися як на сервері, так і на

клієнті. Якщо виконується декілька завдань, вони виконуються послідовно в порядку реєстрації, і наступне завдання чекає завершення попереднього.

Виконується після завдань і перед видимим завданням. Виконується на сервері та на клієнті. Цей етап виконується для побудови віртуального дерева DOM і створення відповідного HTML на сервері.

Виконується після відображення і тільки на клієнті. Цей етап виконується, коли компонент стає видимим в області видимості (Рисунок 2.5).

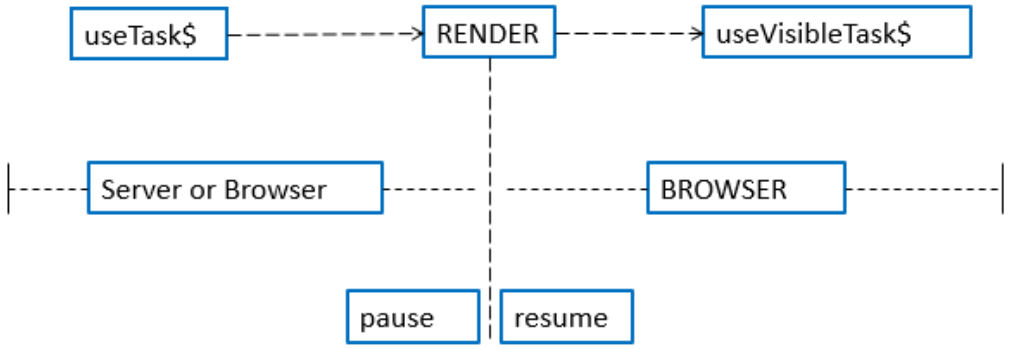

Рисунок 2.5 — Життєвий цикл компонента

Зазвичай життєвий цикл компонента починається на сервері, і в цьому випадку етапи "Завдання" і "Відображення" виконуються на сервері. Потім, коли компонент стає видимим, виконується етап "Видимого завдання".

Але іноді компонент може бути спочатку відображений на клієнті, наприклад, коли користувач переходить до нової сторінки або коли "модальний" компонент спершу з'являється на сторінці. У цьому випадку життєвий цикл виконується тільки в браузері (рисунок 2.6).

Життєвий цикл залишається однаковим, незалежно від того, де спочатку відбувається рендеринг. Це сприяє відновленню інтерфейсу користувача та забезпечує єдність виконання як на сервері, так і на клієнті.

Функція useTask$() реєстру хук, який буде запущений під час створення компонента і виконається принаймні один раз. Місце виконання цієї функції залежить від того, де спочатку рендериться компонент на сервері або в браузері.

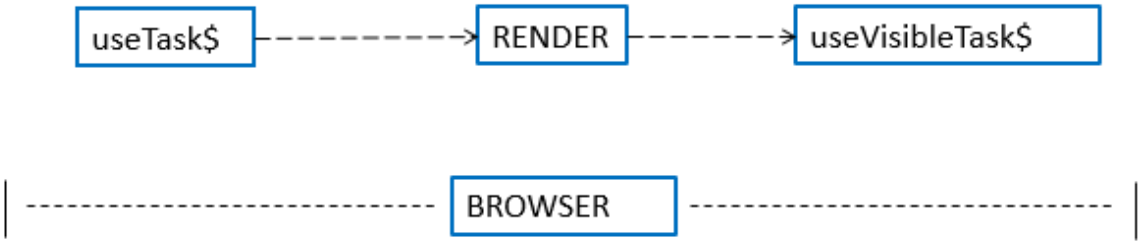

Рисунок 2.6 — Життєвий цикл компонента в браузері

Крім того, useTask$() може бути реактивним і виконуватиметься повторно, коли відстежуються зміни стану (Рисунок 2.7).

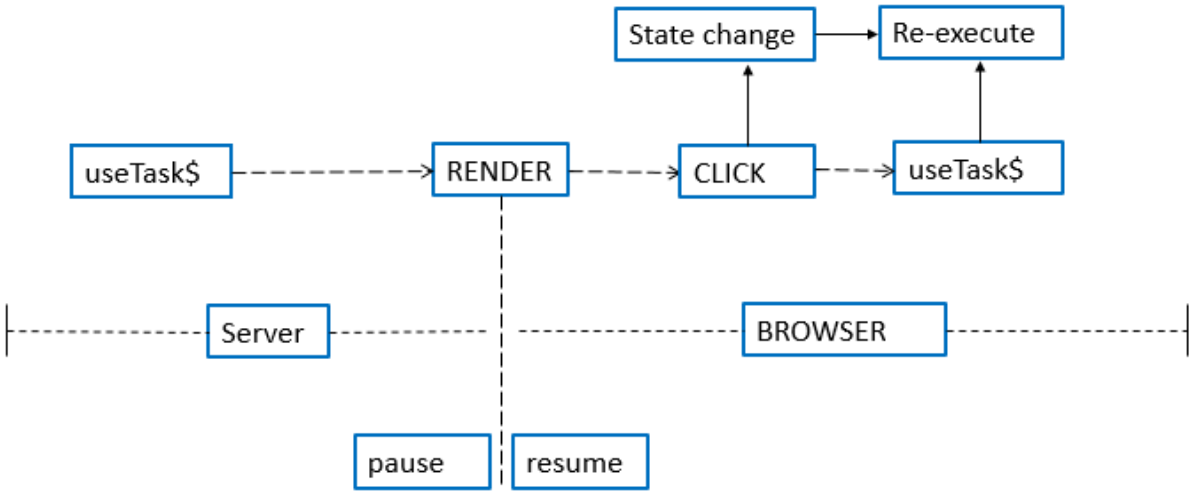

Рисунок 2.7 — Життєвий цикл компонента зі змінним станом

Якщо useTask$() не відстежує жоден стан, він запуститься рівно один раз або на сервері, або в браузері (не в обох), залежно від того, де компонент спочатку відтворюється.

3 РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ-ЗАСТОСУНКУ

3.1 Проектування структури клієнтської частини веб-застосунку

При аналізі вимог до клієнтської частини застосунку були визначені ключові функціональні та нефункціональні вимоги для забезпечення ефективної та зручної роботи системи.

Застосунок повинен надавати можливість користувачам зберігати та отримувати особисті дані, такі як ім'я, електронна адреса, вік, стать та інші важливі параметри. Додаток повинен забезпечувати безпечний та надійний механізм реєстрації та автентифікації користувачів для забезпечення конфіденційності та безпеки даних.

Дизайн інтерфейсу повинен бути зручним та інтуїтивно зрозумілим для користувачів будь-якого рівня технічної підготовки, забезпечуючи комфортну навігацію та взаємодію з системою.

Застосунок повинен коректно відображатись на різних пристроях, забезпечуючи адаптивний та відзивчивий дизайн для мобільних телефонів, планшетів та настільних комп'ютерів.Враховуючи потребу у високій швидкодії для взаємодії з штучним інтелектом, система повинна бути оптимізованою для мінімізації часу відгуку та максимізації продуктивності.

Структура взаємодії з сервером повинна бути грамотно розробленою, забезпечуючи легкість розширення та масштабування системи для додавання нового функціоналу та модулів. Аналіз вимог підкреслює необхідність створення компактної, швидкої та гнучкої клієнтської частини застосунку для забезпечення ефективної взаємодії з користувачем та високого рівня задоволення від використання продукту.

Проаналізувавши всі вимоги до застосунку, в роботі розроблено загальну структуру клієнтської частини веб-застосунку, яка наведена на рис. 3.1. При цьому було враховано вимоги до продуктивності, безпеки, доступності, масштабованості, надійності та зручності інтерфейсу. Зазначена структура передбачає чітку

організацію функціональних частин системи та забезпечує їх взаємодію для досягнення визначених цілей.

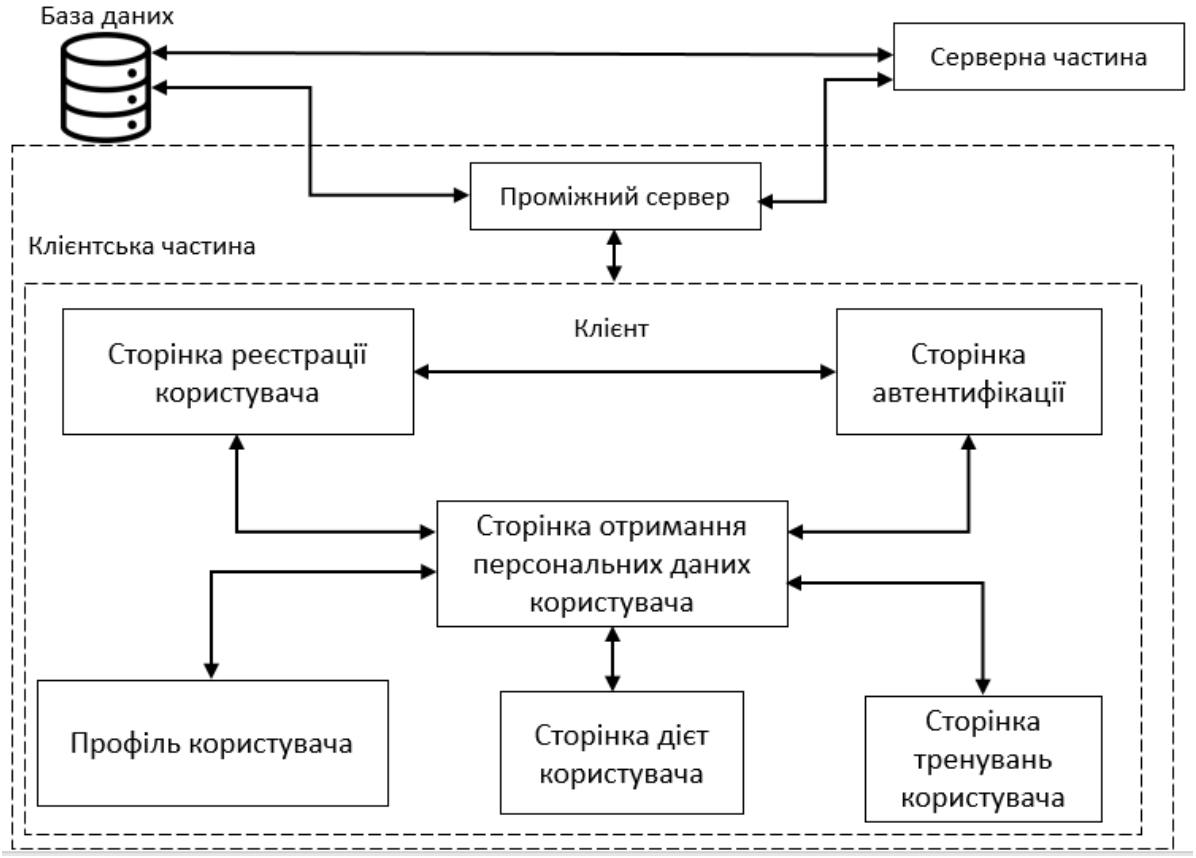

Рисунок 3.1 — Структурна схема клієнтської частини веб-застосунку

Головні компоненти клієнтської частини включають сторінки реєстрації та логіну, які надають користувачам можливість створення або входу в обліковий запис.

Після входу користувач має можливість ввести свої фізіологічні дані та деталі про свій образ життя, вибрати стать, ціль для досягнення (схуднення, підтримка ваги, набір ваги) та перейти до додаткової форми для введення інших важливих даних.

Друга форма дозволяє користувачеві вказати алергії, вподобання в їжі та інші деталі, які будуть враховані при побудові дієти. Це дозволяє персоналізувати рекомендації та забезпечити користувача збалансованою дієтою.

Після введення всіх необхідних даних користувач натискає кнопку "Створити дієту", і система відправляє дані на сервер для обробки. Після обробки

сервером користувач отримує персоналізовані рекомендації з харчування та переходить на сторінку, де відображаються розклад та харчування на кожен день.

Всі сторінки мають спільні компоненти, такі як хедер та футер, які забезпечують єдність дизайну та навігації на всьому веб-застосунку. Такий підхід допомагає уникнути дублювання коду та підтримує консистентність інтерфейсу.

Важливою особливістю клієнтської сторони веб-застосунку є її взаємодія з проміжним сервером. Qwik дозволяє частковий рендеринг сторінок на проміжному сервері, що поліпшує продуктивність та забезпечує швидкий відгук для користувачів. Це особливо важливо при великій кількості даних або складних візуальних елементах.

Крім того, використання проміжного серверу дало можливість напряму взаємодіяти з базою даних. Це дозволило клієнтській частині звертатися до бази даних без необхідності робити додаткові запити на серверну частину. Наприклад, якщо користувач хоче змінити свої фізіологічні дані, клієнт може одразу взаємодіяти з базою даних через проміжний сервер, уникнувши зайвого трафіку та зменшивши навантаження на сервер.

3.2 Програмна реалізація клієнтської частини веб-застосунку

Грамотно структурований проект є необхідною складовою розробки будь-якого веб-застосунку. Організація грає ключову роль у спрощенні процесу створення додатку, надаючи можливість швидко розпочати роботу, не витрачаючи час на пошук потрібних файлів. Крім того, належна структура проекту значно зменшує ймовірність помилок під час компіляції, що важливо для ефективного виконання завдань. Фізична структура веб-застосунку представлена на рисунку 3.2.

Папка "Api" відіграє ключову роль у взаємодії із сервером та управлінні серверною-логікою. В цій папці зосереджена функціональність, пов'язана з взаємодією з сервером, включаючи реалізацію наступного функціоналу:

— запити до сервера;
— автентифікація та реєстрація користувача;
— управління токенами;

— збереження даних користувача, дієт та тренувань;

— обробка помилок.

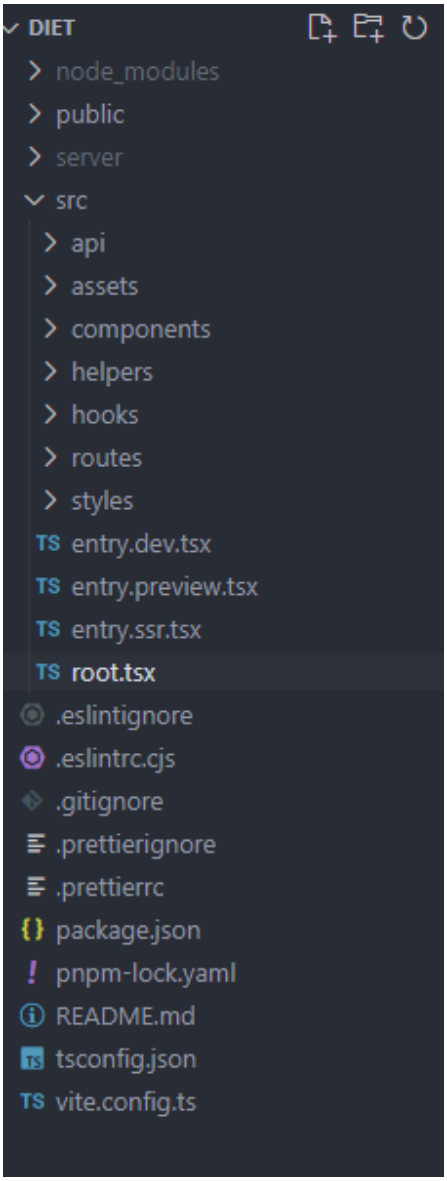

Рисунок 3.2 — Структура каталогів веб-застосунку

Ця структура дозволяє ефективно організувати логіку взаємодії із сервером у вигляді окремої папки, забезпечуючи легкість управління та розширенням функціоналу.

Структура каталогу Api представлена на рисунку 3.3

Проект використовує TypeScript, тому у папці "dto" розташований файл "auth.tsx", де зберігаються інтерфейси які використовуються під час запитів до сервера. Використання TypeScript дозволяє користуватися інтерфейсами для

типізації взаємодії із сервером, що сприяє зручності виявлення та усунення помилок на етапі розробки веб-застосунку.

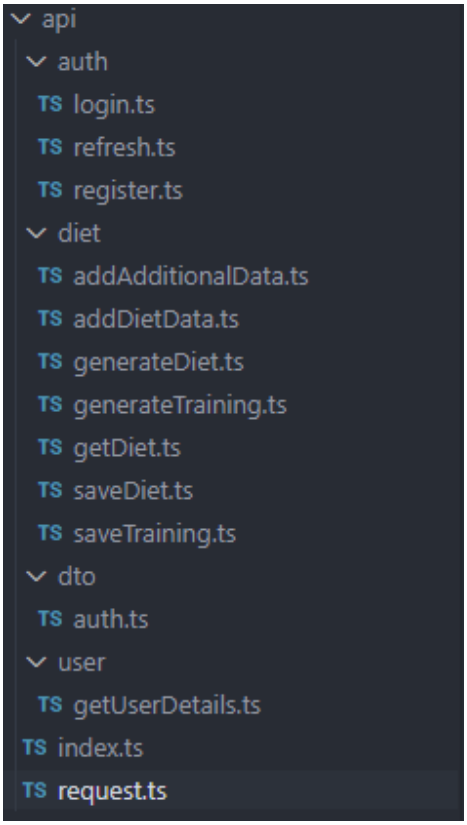

Рисунок 3.3 — Структура каталогу Api

Також одна з основних каталогів це папка components (рисунок 3.4). У папці "components" розташовані всі необхідні компоненти, які будуть використовуватися на різних сторінках проекту, такі як: кнопки, заголовки, форми і тд. Ця папка містить модульні компоненти, які можна використовувати одноразово або використовувати на декількох сторінках для забезпечення єдності стилю, функціональності та економії ресурсів користувача під час використання веб-застосунку.

Як можна побачити на даному рисунку в каталозі components розміщено багато різних папок. В кожній папці міститься два файли один з логікою поведінки компонента та об’єктом який містить jsx код, а другий містить стилі даного компоненту.

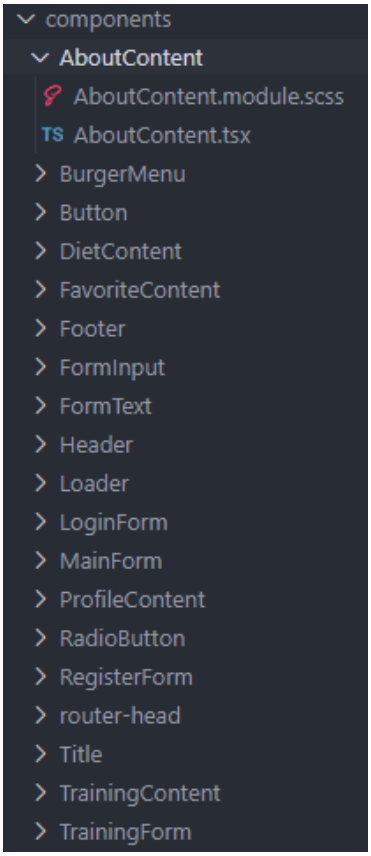

Рисунок 3.4 — Структура каталогу components

Кожна з цих папок є окремим компонентом який буде використовуватись в інших компонентах та\або на сторінках у веб-застосунку.

У папці "assets" та "styles" розміщуються зображення та глобальні стилі, що використовуються в усьому проекті (рисунок 3.5).

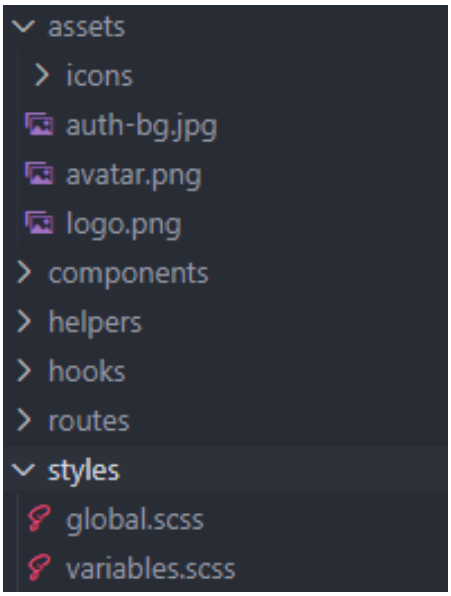

Рисунок 3.5 — Структура каталогів assets та styles

У "assets" знаходяться зображення, іконки і тд, які можуть бути використані в різних частинах додатку. Це дозволяє ефективно керувати та організовувати ресурси проекту.

У папці "styles" розташовані глобальні стилі та змінні, які використовуються для стилізації компонентів та елементів інтерфейсу. Це сприяє єднанню та однорідності дизайну усього веб-застосунку. Використання глобальних стилів також дозволяє зменшити обсяг коду, що позитивно впливає на продуктивність та завантаження ресурсів користувача.

3.3 Реєстрація та автентифікація

3.3.1 Програмна реалізація реєстрації

На початковому етапі розробки реєстраційної функціональності веб-застосунку, створено окремий каталог з назвою "auth", призначений для об'єднання компонентів та логіки, пов'язаної з автентифікацією. У межах цього каталогу створено ще один підкаталог "register", в якому розташований файл із назвою "index.tsx" (рисунок 3.6). Цей файл є відповідальним за реалізацію сторінки реєстрації веб-застосунку.

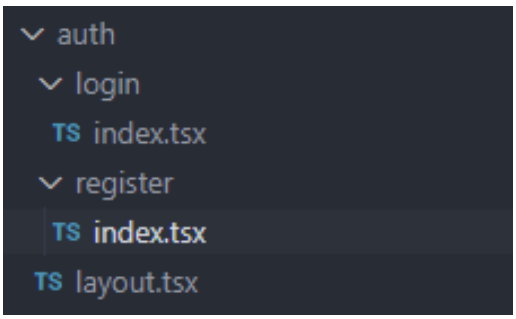

Рисунок 3.6 — Каталог auth

Структура файлу "index.tsx" включає в себе компоненти та логіку, пов'язану зі сторінкою реєстрації. Реалізація цієї сторінки включає в себе форму для введення реєстраційних даних користувача, таких як ім'я, електронна пошта та пароль.

При використанні цієї сторінки, користувач може заповнити необхідні поля та відправити дані для реєстрації. Логіка реєстрації включає в себе відправку

запиту до сервера для обробки реєстраційної інформації. При успішній реєстрації користувача буде перенаправлено на головну сторінку.

Після створення даного файлу в нього було імпортовано усі необхідні бібліотеки та компоненти (лістинг 3.1).

Лістинг 3.1 — Приклад імпорту бібліотек та компонентів

```
import { component$ } from '@builder.io/qwik';
import type { DocumentHead } from '@builder.io/qwik-city';
import { routeLoader$ } from '@builder.io/qwik-city';
import { type InitialValues } from '@modular-forms/qwik';
import RegisterForm from '~/components/RegisterForm/RegisterForm';
```

В цьому розділі використовуються імпорти з бібліотек @builder.io/qwik та @modular-forms/qwik. Основні об'єкти, які імпортуються, це component$, DocumentHead, routeLoader$ та InitialValues. Компонент RegisteForm це безпосередньо форма реєстрації з якою буде взаємодіяти користувач, яка імпортується з каталогу components.

Далі було оголошено тип форми реєстрації та створено функцію яка буде використовуватися для завантаження даних з форми (лістинг 3.2).

Лістинг 3.2 — Приклад створення типу та функції для запису даних

```
export type RegisterForm = {
    name: string;
    email: string;
    password: string;
    repeatPassword: string;
};
export const useFormLoader = routeLoader$<InitialValues<RegisterForm>>(() => ({
    name: '',
    email: '',
    password: '',
    repeatPassword: '',
}));
```

Далі було створено основний компонент, який використовує component$. У цьому компоненті відбувається відображення компонента форми реєстрації RegisterForm.

Усі ці елементи разом створюють структуру та функціонал сторінки реєстрації веб-застосунку

Далі було розроблено компонент RegisterForm, який використовувався у файлі зі сторінкою реєстрації.

Так само як і на сторінці реєстрації виконується імпорт необхідних модулів та ресурсів, таких як компоненти, стилі та іконки. Qwik та Builder.io надають інструменти для ефективної роботи з компонентами та реактивним станом (лістинг 3.3).

Лістинг 3.3 — Приклад імпорту бібліотек, компонентів та зображень

```
import { $, component$, JSXNode, useSignal } from '@builder.io/qwik';
import { SubmitHandler, useForm } from '@modular-forms/qwik';
import { RegisterForm, useFormLoader } from '~/routes/auth/register';
import { useAuth } from '~/hooks/useAuth';
import { api } from '~/api';
import styles from '~/components/LoginForm/LoginForm.module.scss';
import Button from '~/components/Button';
import Title from '~/components/Title';
import Loader from '~/components/Loader';
import FormInput from '~/components/FormInput';
import { Link } from '@builder.io/qwik-city';
import EmailIcon from '~/assets/icons/email.svg?jsx';
import PassWordIcon from '~/assets/icons/PassWord.svg?jsx';
import PassWordRepeatIcon from '~/assets/icons/password-repeat.svg?jsx';
import NameIcon from '~/assets/icons/name.svg?jsx';
```

Перед початком створення безпосередньо форми реєстрації потрібно створити тип, інтерфейс та масив з об’єктами для коректної роботи з формою. Тип буде мати назву RegisterFormKeys, який буде представляти ключі об'єкта RegisterForm, який був створений на сторінці реєстрації. Інтерфейс, який було

названо RegisterFormFeilds, буде представляти структуру кожного поля форми та містити в собі такі властивості як: icon, label, fieldName та type (лістинг 3.4).

Лістинг 3.4 — Приклад створення типу та інтерфейсу

```
type RegisterFormKeys = keyof RegisterForm;
interface RegisterFormFields {
    icon: JSXNode;
    label: string;
    fieldName: RegisterFormKeys;
    type: 'text' | 'email' | 'password';
}
```

Далі було створено масив з об’єктами який має назву formFields, де кожен об'єкт представляє одне поле форми (лістинг 3.5).

Лістинг 3.5 — Приклад створення масиву об’єктів

```
const formFields: RegisterFormFields[] = [
    {
        icon: <NameIcon />,
        label: 'Введіть ваше ім’я:',
        fieldName: 'name',
        type: 'text',
    },
    {
        icon: <EmailIcon />,
        label: 'Введіть ваш email:',
        fieldName: 'email',
        type: 'email',
    },
    {
        icon: <PassWordIcon />,
        label: 'Придумайте пароль:',
        fieldName: 'password',
        type: 'password',
    },
    {
        icon: <PassWordRepeatIcon />,
        label: 'Повторіть пароль:',
```

Продовження лістингу 3.5

```
        fieldName: 'repeatPassword',
        type: 'password',
    },
];
```

У даному випадку об'єкти оголошуються для імені, електронної пошти, пароля та повторення пароля. Кожен об'єкт містить властивості, що визначають його структуру, такі як icon, label, fieldName та type. Властивість icon використовується для вставки відповідної іконки, які ми імпортуємо з каталогу assets. label встановлює відповідний текст до кожного поля, наприклад: «Введіть ваше ім’я» або «Придумайте пароль». Поле fieldname вказує на ключ в RegisterForm, до якого буде призначено значення цього поля. Властивість type буде вказувати на тип у кожного з полів вводу, наприклад у поля де потрібно ввести почту буде використовуватись тип email.

Далі було створено компонент реєстрації. У даному компоненті ми ініціюємо два реактивні сигнали: isLoading та error, призначені для відстеження стану завантаження та відображення можливих помилок, що можуть виникнути під час процесу реєстрації користувача (лістинг 3.6).

Лістинг 3.6 — Приклад створення типу та інтерфейсу

```
const isLoading = useSignal(false);
    const error = useSignal('');
    const auth = useAuth();
    const [, { Form, Field }] = useForm<RegisterForm>({
        loader: useFormLoader(),
    });
```

Змінна isLoading служить для визначення чи триває процес завантаження, тоді як error використовується для зберігання текстового повідомлення про помилку, яке може виникнути при спробі реєстрації.

Також було використано кастомний хук для оптимального управління станом та функціональністю. Ми отримуємо доступ до функцій авторизації це дозволяє використовувати функціонал авторизації, який був реалізований в каталозі hooks.

Далі було створено хук useForm, де отримуємо два значення: порожній масив, який представляє стан форми, та об'єкт який має два значення Form, Field. Цей об'єкт представляє собою функції, пов'язані з визначенням форми та її полів відповідно. Зокрема, Form відповідає за відображення форми, а Field за визначення окремих полів у формі.

Також, через loader: useFormLoader(), ми передаємо дані які були створені на сторінці реєстрації.

Далі було створено функцію handleSubmit яка викликається при підтверджені відправки форми на сервер користувачем (лістинг 3.7).

Лістинг 3.7 — Код функції handleSubmit

```
const handleSubmit = $<SubmitHandler<RegisterForm>>(async (values) =>
{
 if (values.password === values.repeatPassword) {
     isLoading.value = true;
     const response = await api.registerUser(values);
     isLoading.value = false;
     if (response.isError) {
         error.value = response.error?.message ?? 'Error';
         return;
     }
     const loginResponse = await auth.loginUser({
         email: values.email,
         password: values.password,
     });
     if (loginResponse.isError) {
         error.value = loginResponse.error?.message ?? 'Error';
         return;
     }
 } else {
     error.value = 'Паролі не співпадають';
 }
});
```

Після того, як користувач натисне кнопку "Зареєструватися", викликається функція handleSubmit. Перш за все, вона перевіряє, чи паролі, які ввів користувач,

співпадають. Якщо паролі не співпадають, виводиться відповідна помилка, і ніякі дані не відправляються на сервер. У випадку коли паролі співпадають, відбувається виклик функції registerUser (лістинг 3.8) для здійснення запиту на сервер.

Лістинг 3.8 — Код функції registerUser

```
export const registerUser = server$(async (data: RequestRegisterData)
=> {
    return await request('/user/register', 'POST', data);
});
```

Під час очікування відповіді від сервера у користувача з’являється позначка, що вказує користувачеві на те, що триває завантаження. Після отримання відповіді від сервера перевіряється, чи сервер не надіслав помилку. Якщо отримано помилку, виводиться відповідне повідомлення про помилку на екран і функція припиняє виконання. Якщо помилок не виникло, дані користувача автоматично передаються в наступний запит на автентифікацію. Якщо логін також не видає помилок, користувач потрапляє на головну сторінку.

Далі було створено саму форму за допомогою JSX-елементів, таких як <Form>, <Field>, <FormInput>, <Button>, і <Link>, які відповідають за створення різних елементів форми та відображення їх на екрані (Додаток Б).

3.3.2 Програмна реалізація автентифікації

В процесі створення сторінки автентифікації та форми логіну використовувався схожий підхід як і на сторінці реєстрації (Додаток В).

Форма логіну виглядає подібно формі реєстрації. Так само імпортуються усі необхідні компоненти, стилі, іконки та бібліотеки Qwik. Створюються всі ті самі типи, інтерфейси, масиви та об’єкти але тільки для двох полів логіну і паролю. Логіном є пошта користувача, а паролем до цього логіну є пароль який користувач ввів під час реєстрації. Також є кнопка «увійти» на яку користувач натисне після того як заповнить всі поля у формі логіну.

Також в даній формі дещо по іншому працює функція handlSubmit, яка виконується після натискання на кнопку «увійти» (лістинг 3.9).

Лістинг 3.9 — Функція handlSubmit на сторінці автентифікації

```
const handleSubmit = $<SubmitHandler<LoginForm>>(async (values)=>{
        isLoading.value = true;
        const response = await auth.loginUser(values);
        isLoading.value = false;
        if (response.isError) {
            error.value = response.error ?? 'Error';
        }
    });
```

Після введення користувачем даних в форму та їх відправлення, відбувається виклик функції loginUser (лістинг 3.10), яка знаходиться у каталозі Api, для здійснення запиту на сервер. Цей запит відбувається асинхронно та включає передачу об'єкта з даними логіну, такими як email та пароль. Далі йде перевірка на те чи сервер не відправив помилку. Якщо прийшла помилка вона виведеться на екран і функція закінчить своє виконання.

Лістинг 3.10 — Функція loginUser

```
export const loginUser = async (data: RequestLoginData) => {
    return await request<ResponseLoginData>('/auth/login', 'POST', data);
};
```

Функція loginUser використовує інтерфейси RequestLoginData (лістинг 3.11) та ResponseLoginData (лістинг 3.12), які визначають формат даних для запиту на логін та формат відповіді від сервера відповідно.

При успішному виконанні запиту сервер надсилає об'єкт ResponseLoginData, який включає токени доступу, дані користувача та іншу інформацію. Ці дані можуть бути збережені в локальному сховищі для подальших операцій.

Лістинг 3.11 — Інтерфейс RequestLoginData

```
export interface RequestLoginData {
    email: string;
    password: string;
}
```

Цей інтерфейс визначає структуру об'єкту, який передається в запиті на сервер для логіну. Об'єкт містить два поля: email для електронної пошти користувача та password для його пароля.

Лістинг 3.12 — Інтерфейс ResponseLoginData

```
export interface ResponseLoginData {
    accessToken: string;
    firstName: string;
    refreshToken:string;
    userId:string;
    userName:string;
}
```

Цей інтерфейс визначає структуру об'єкту, який може бути отриманий у відповіді від сервера після логіну. Об'єкт містить інформацію про успішну автентифікацію, таку як accessToken (токен доступу), firstName (ім'я), refreshToken (токен оновлення), userId (ідентифікатор користувача) та userName (ім'я користувача).

Ці інтерфейси використовуються для підтримки типізації та створення об'єктів з чітко визначеними властивостями при взаємодії з функцією loginUser та в обробці відповідей від сервера.

3.4 Організація маршрутизації в Qwik

У Qwik, маршрутизація ґрунтується на структурі папок у директорії src/routes/. Кожна папка у цій директорії представляє окрему сторінку або компонент.

Тому для створення сторінки тренування, на якій буде знаходитись форма введення фізіологічних даних користувача, в папці routes створюємо папку з назвою training. У папці сторінки розміщуються всі компоненти та файли, пов'язані з цією сторінкою. Основний компонент сторінки, який відповідає за відображення сторінки, має назву index.tsx (рисунок 3.7).

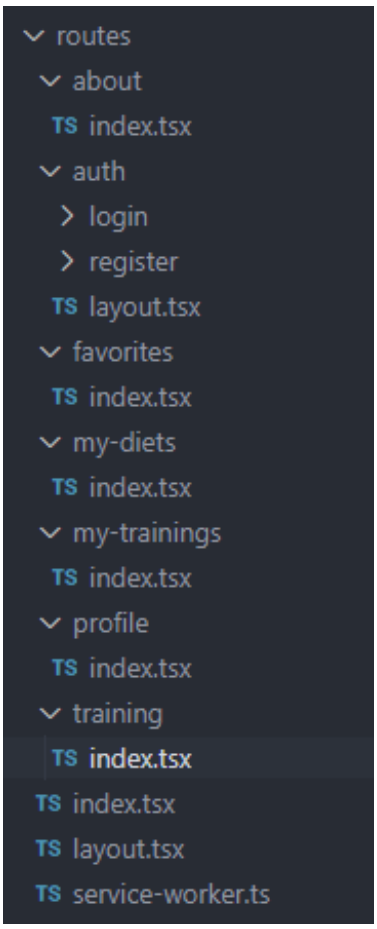

Рисунок 3.7 — Організація сторінок у веб-застосунку

В даній структурі також знаходиться папка auth в якій є ще дві папки login та register. Кожна з цих двох папок містить в собі файл index.tsx, який є основним файлом сторінки. Така організація дозволяє кожній сторінці мати свій власний розділ коду, але при цьому зберігає їх об'єднаними за допомогою спільної батьківської папки auth.

Також в папці auth створений файл layout.tsx. Даний використовується як обгортка для обох сторінок. У цьому файлі використовується тег <slot/> (лістинг 3.13), який дозволяє вставляти вміст конкретної сторінки — логін чи реєстрацію.

Лістинг 3.13 — Приклад використання тегу Slot

```
<>
     <AuthBG class={'authBg'} />
       <div class={'authBar'} />
       <div class={'authContent'}>
```

Продовження лістингу 3.13

```
            <Logo class={'authLogo'} />
            <Slot />
        </div>
</>
```

Використання тега <slot/> дозволяє об'єднати спільні компоненти і використовувати їх для обох сторінок без необхідності повторювати код.

Цей підхід зроблений для полегшення управління кодом, забезпечення однакового вигляду та структури для обох сторінок, а також уникнення дублювання коду в разі змін чи розширення функціональності в обох частинах проекту.

3.5 Основні компоненти клієнтської частини веб-застосунку

Використання веб-платформи є простим та зручним процесом, який передбачає реєстрацію нового облікового запису. Для цього користувачам достатньо заповнити обов'язкові поля форми, такі як ім'я, електронна пошта, пароль та його повторення (рисунок 3.8). Цей етап дозволяє створити особистий обліковий запис і мати повний доступ до усіх функцій та можливостей веб-платформи.

На сторінці реєстрації користувач також може швидко перейти до сторінки логіну, якщо вже має обліковий запис. Це забезпечує гнучкість та зручність використання платформи для користувачів з різними потребами та умовами використання. Послуга надається в інтуїтивно зрозумілому форматі, сприяючи швидкому та ефективному взаємодії з платформою.

Для входу в систему користувачі мають можливість використовувати сторінку логіну, яка надає швидкий та безпечний спосіб отримання доступу до особистого облікового запису. Сторінка логіну містить два основних поля: електронна пошта та пароль.

На цій сторінці користувачі можуть ввести свою зареєстровану електронну адресу та пароль для автентифікації.

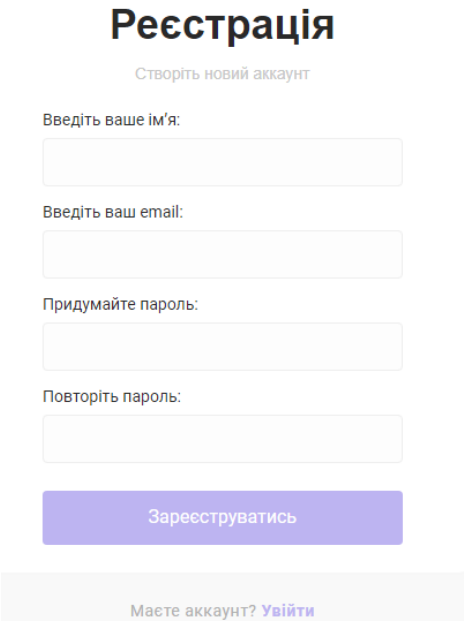

Рисунок 3.8 — Фрагмент форми на сторінці реєстрації

Також, для зручності, сторінка логіну містить посилання на сторінку реєстрації для тих, хто ще не має облікового запису (рисунок 3.9).

Завдяки мінімалістичному та зрозумілому дизайну форми логіну, користувачам легко та швидко увійти в систему, забезпечуючи безпеку та конфіденційність їхніх даних. Використання облікового запису дозволяє отримати повний доступ до усіх функцій та сервісів веб-платформи.

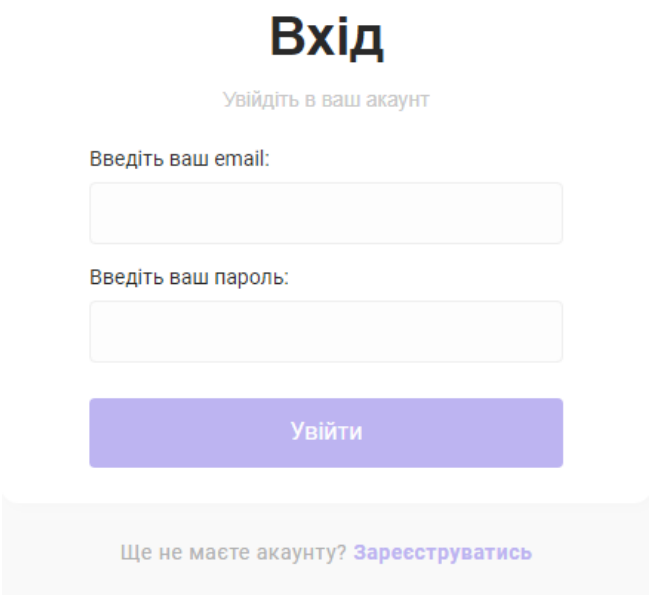

Рисунок 3.9 — Фрагмент форми на сторінці автентифікації

Після успішного входу в систему користувач потрапляє на головну сторінку, де йому пропонується ввести фізіологічні дані та надати деталі щодо свого образу життя. На цій сторінці користувач має можливість визначити такі параметри, як вік, зріст, вага, рівень фізичної активності тощо. Також він може вибрати свою стать (чоловік або жінка) та визначити свою основну ціль: схуднення, підтримка ваги або набір ваги (рисунок 3.10).

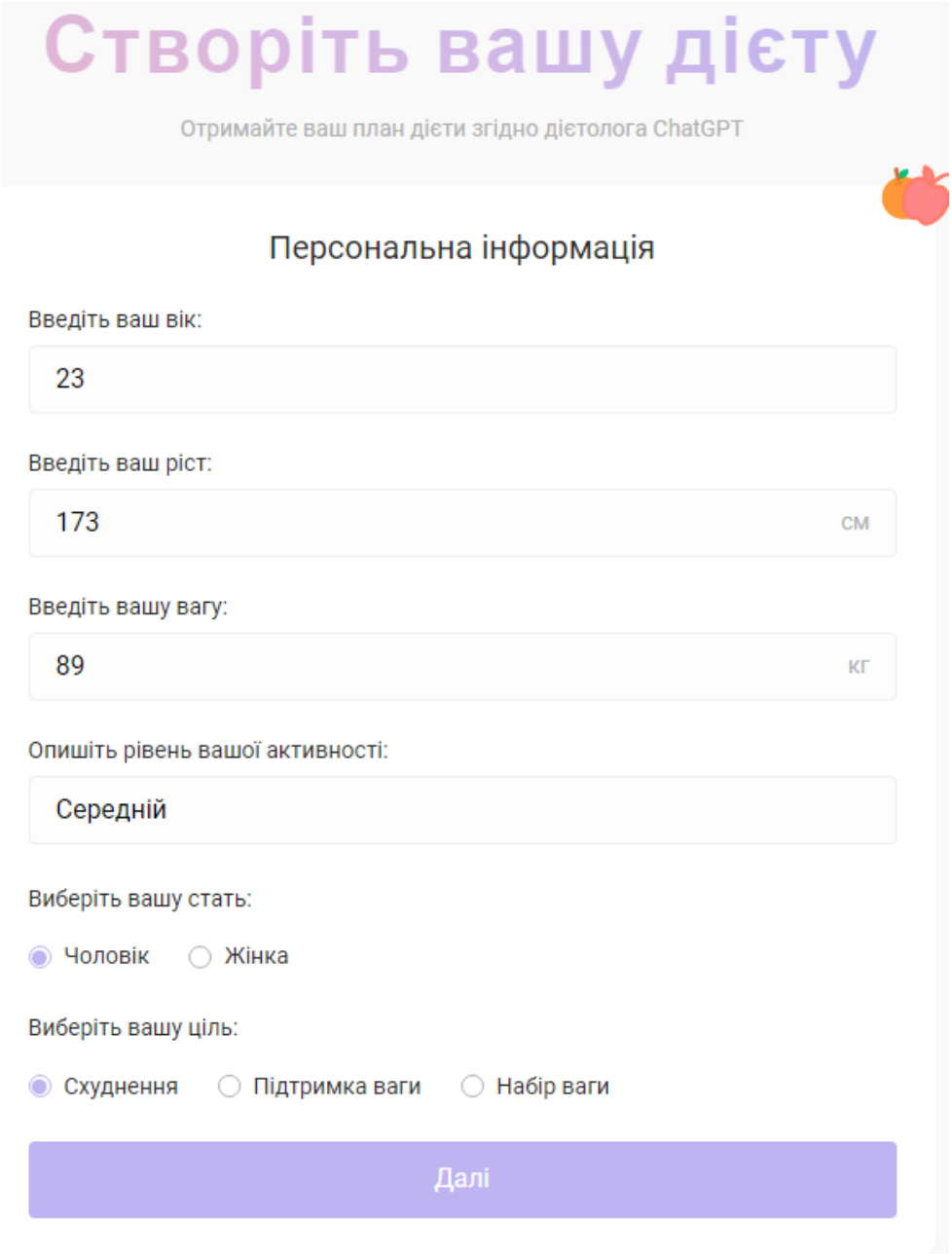

Рисунок 3.10 — Фрагмент форми на головній сторінці

Після введення основних фізіологічних даних та визначення цілей, користувач переходить на наступний етап, де заповнює додаткову інформацію про свій раціон. У цій формі користувач може вказати свої алергії, вподобання в їжі, продукти, які мають бути мінімум у його раціоні, а також ті, які повинні бути виключені (рисунок 3.11).

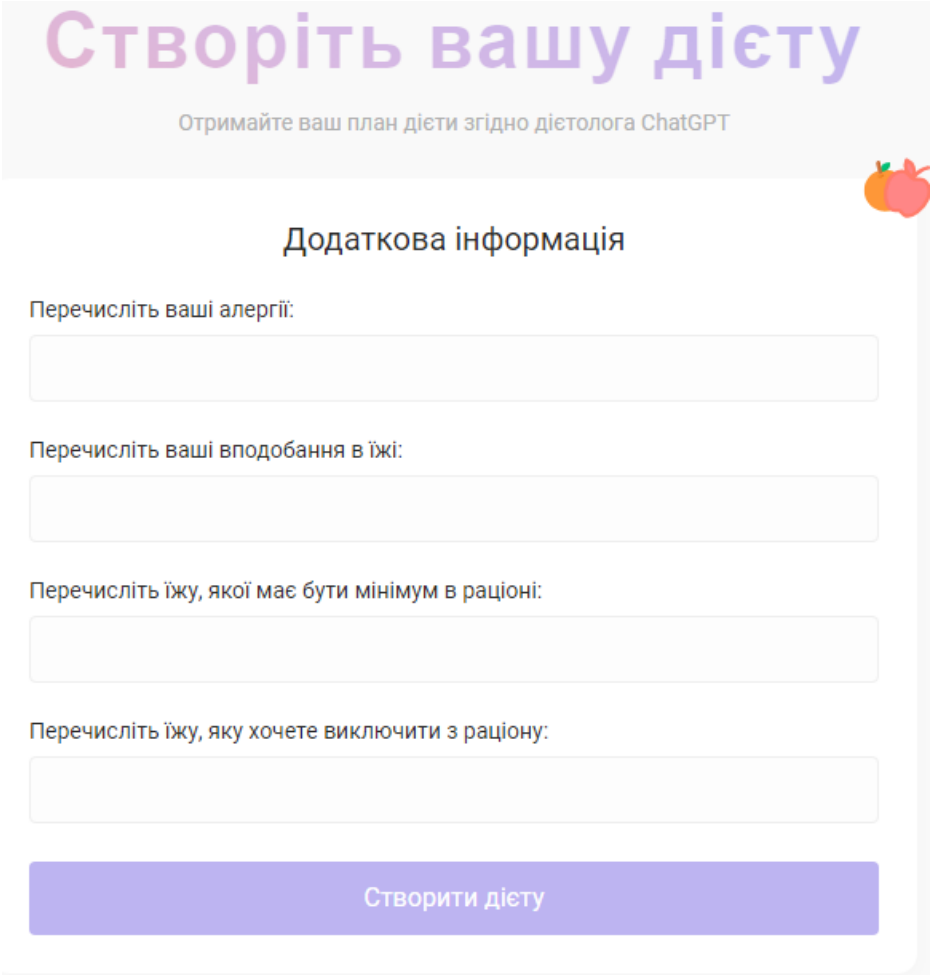

Рисунок 3.11 — Фрагмент форми з додатковою інформацією

Після того, як користувач успішно заповнив обидві форми та натиснув кнопку "Створити дієту", дані відправляються на сервер для обробки. На сервері інформація з форм зчитується, і відбувається побудова дієти відповідно до вказаних користувачем вимог та цілей.

Після успішної обробки сервером, користувач отримує відповідь, і його перенаправляє на сторінку, де відображається побудована дієта (рисунок 3.12).

Додаткова важлива особливість цього підходу полягає в тому, що друга форма, яка включає в себе інформацію про алергії, вподобання в їжі, переваги в раціоні та продукти, які варто виключити, є необов'язковою для заповнення. Користувач має можливість вибрати, які саме аспекти харчування він бажає враховувати, тим самим створюючи гнучкість та зручність у використанні платформи. Це дозволяє кожному користувачеві визначити обсяг та глибину наданої інформації, щоб забезпечити максимально комфортний та індивідуальний підхід до використання платформи.

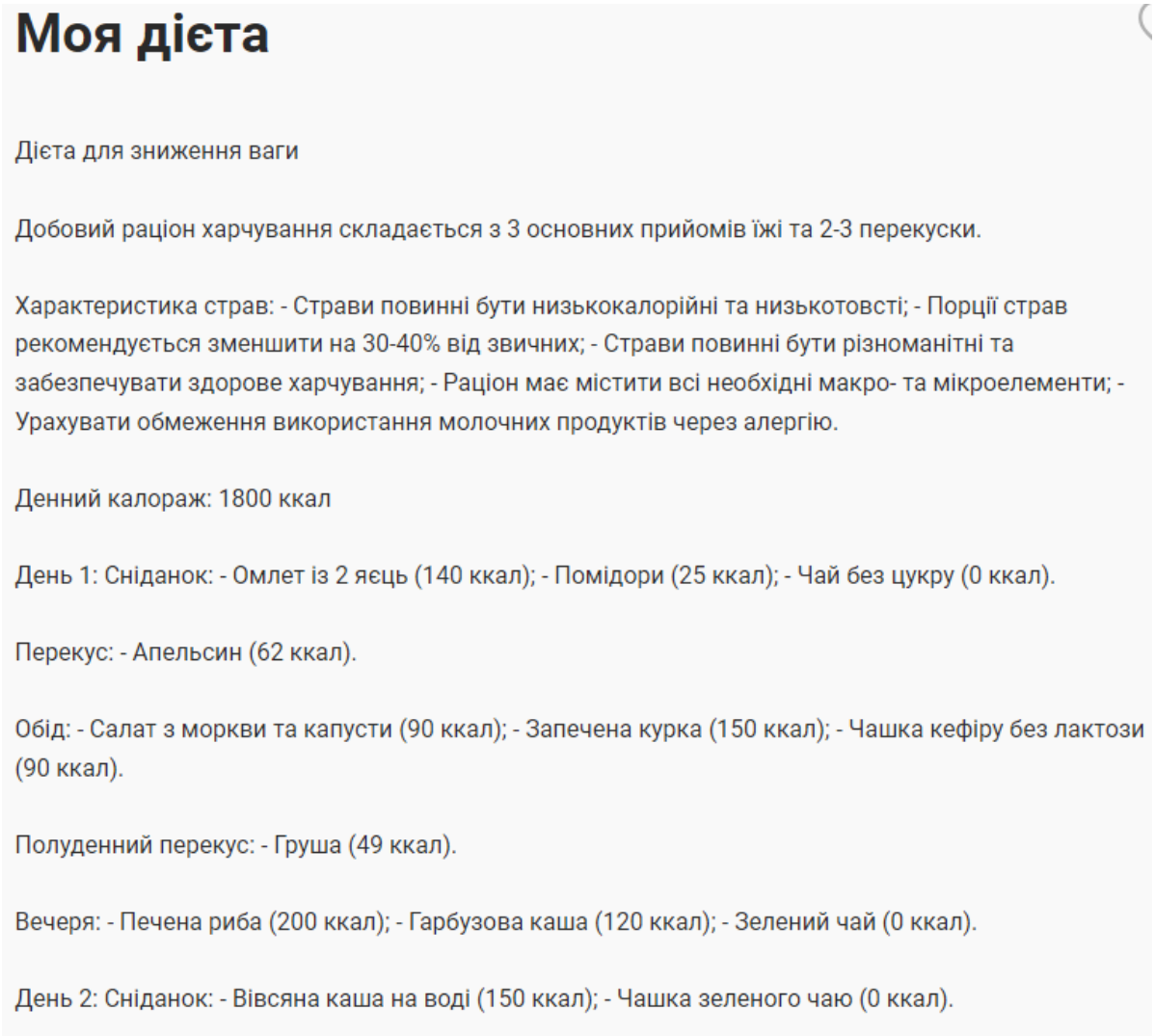

Рисунок 3.12 — Приклад відображення створеної дієти

На цій сторінці користувач може знайти деталізоване розподілення харчування на кожен день, а також інформацію про кількість споживаних калорій.

Також у користувача є доступ до сторінки зі статичною інформацією (рисунок 3.13). На цій сторінці користувач отримає важливі факти, що стосуються переваг здорового способу життя та збалансованого харчування. Ця інформація допомагатиме збудувати у користувача усвідомлене ставлення до власного здоров'я та підтримувати його у бажанні приділяти увагу правильному харчуванню та активному способу життя. Передача цих фактів може стати невід'ємною частиною місії платформи, оскільки вони надають користувачеві додатковий стимул та розуміння, чому важливо обирати здоровий спосіб життя.

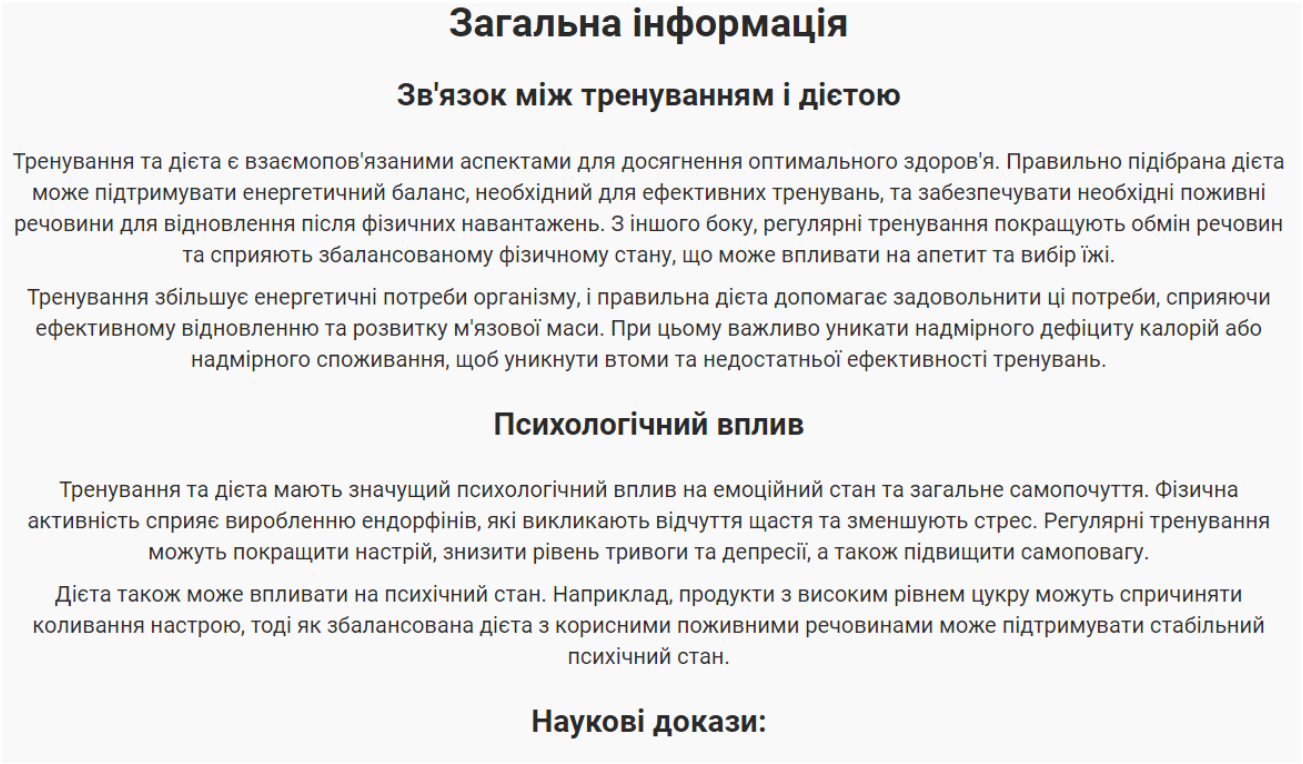

Рисунок 3.13 — Сторінка загальної інформації

Хедер та футер є сталими елементами на кожній сторінці, забезпечуючи єдність та зручність навігації для користувача.

У хедері розташовано навігаційне меню, логотип та іконка профілю користувача. Навігаційне меню містить посилання на ключові розділи веб-застосунку, такі як "Загальне", "Мої дієти", "Мої тренування" та "Обране". Логотип слугує візуальним ідентифікатором платформи, а іконка профілю дає змогу

користувачеві здійснювати дії, пов'язані з особистим обліковим записом (рисунок 3.14).

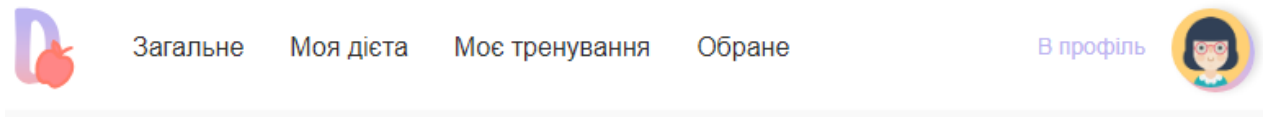

Рисунок 3.14 — Навігаційний елемент Header

Також на веб-платформі присутній компонент Footer (рисунок 3.15).

Рисунок 3.15 — Компонент Footer

Footer містить навігаційне меню, розташоване по центру. Він також містить загальні посилання на сторінки, які доступні з будь-якої частини платформи. Футер сприяє зручності переходу між ключовими розділами та забезпечує єдність дизайну на всіх сторінках.

4 ТЕСТУВАННЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

4.1 Характеристики програмного забезпечення для відображення веб-сторінки

Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту реалізований у форматі веб-сайту, забезпечуючи доступність для користувачів з комп’ютерами або смартфонами із базовими технічними характеристиками та можливістю встановлення веб-браузера.

Системні вимоги:

— операційна система: Windows XP/Vista/7/8/10, Mac OS X 10.6 або вище;

— процесор: Intel Pentium 4/Athlon 64 або пізнішої версії із підтримкою SSE2;

— вільне місце на диску: 350 Мб;

— оперативна пам'ять: 512 Мб;

— відеокарта: інтегрована.

До того ж, веб-застосунок встановлює певні вимоги до серверного хостингу. Мінімальні характеристики хостингу для цієї системи такі :

— SSD-диск: 30 ГБ;

— трафік: 100 ГБ;

— доступ до GIT;

— приблизно 10 000 відвідувань на місяць.

4.2 Аналіз та порівняння продуктивності Qwik та React

Для оцінки продуктивності застосунків, розроблених на Qwik і React, було використано інструмент PageSpeed Insights.

Порівняння веб-застосунків, розроблених з використанням Qwik та React, включає оцінку різноманітних показників, таких як швидкість рендерингу, реактивність взаємодії, відсутність блокування та загальна продуктивність. Ці параметри дозволяють здійснити комплексне порівняння ефективності обох фреймворків у реальних умовах використання.

Результати представлені на рисунку 4.1 та рисунку 4.2.

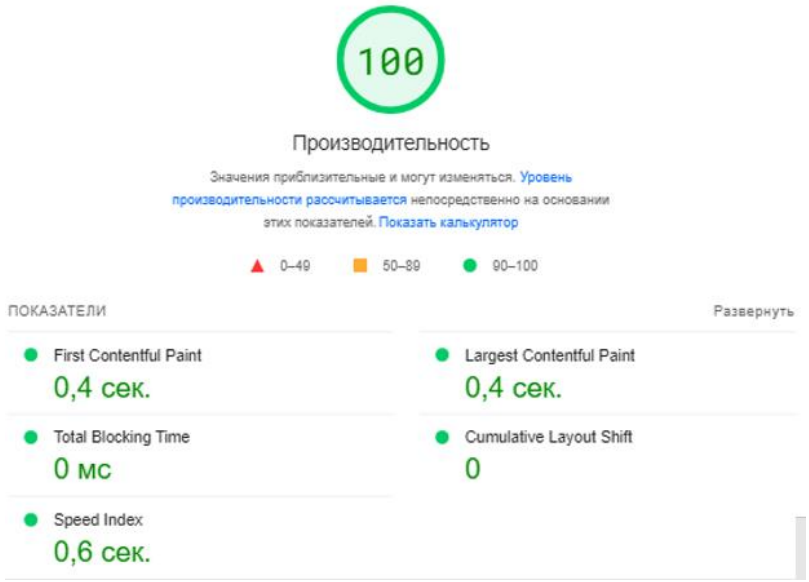

Рисунок 4.1 — Показники продуктивності веб-застосунку на Qwik

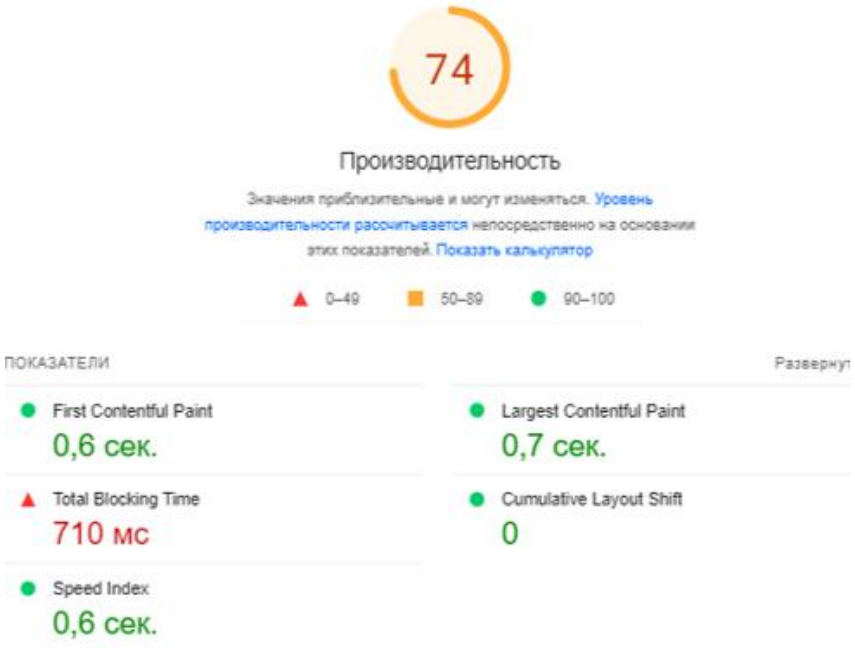

Рисунок 4.2 — Показники продуктивності веб-застосунку на React

Перше відображення контенту First Contentful Paint Індекс — показник, який відображає час між початком завантаження сторінки та появою першого зображення чи блоку тексту. Qwik виявився на 0,2 секунди швидшим,

підкреслюючи прискорений час рендерингу та відображення контенту на клієнтському боці.

Загальний час блокування Total Blocking Time — сума (у мілісекундах) всіх періодів від першого відмальовування контенту до завантаження контенту для взаємодії, коли швидкість виконання завдань перевищувала 50 мс. Qwik має загальний час блокування 0 мс, у порівнянні з React, де цей показник може досягати 710 мс. Це свідчить про те, що Qwik забезпечує миттєвий доступ до вмісту, підвищуючи зручність використання.

Відображення найбільшого контенту Largest Contentful Paint — показник, який відображає час, потрібний на повне відображення найбільшого зображення або текстового блоку. Qwik показав кращі результати будучи швидшим на 0.3 секунди.

Загальний показник продуктивності Qwik значно вищий, визначаючи його як більш ефективний та продуктивний фреймворк, перевершуючи React на 26 балів.

Результати цього порівняння вказують на переваги Qwik у різних аспектах тестування, привертаючи особливу увагу до високої продуктивності фреймворку.

4.3 Тестування клієнтської частини

Після завершення розробки клієнтської частини веб-застосунку виникає необхідність провести етап тестування, спрямований на перевірку її працездатності та відповідність визначеним вимогам та специфікаціям.

Перш за все особлива увага приділяється перевірці функціональності реєстрації користувача. Під час реєстрації, введені користувачем дані, такі як ім'я, електронна пошта, пароль та його підтвердження, є критичними для безпеки та коректності роботи веб-застосунку. Особлива увага приділяється перевірці коректності обробки введених паролів.

Напрямок тестування включає в себе сценарій, при якому користувач вводить різні паролі та їх підтвердження. Веб-застосунок повинен автоматично перевіряти їхню відповідність та виявляти невідповідності. У випадку, коли паролі не

співпадають, система має повідомити користувача про цю помилку та надати відповідне повідомлення (рисунок 4.3).

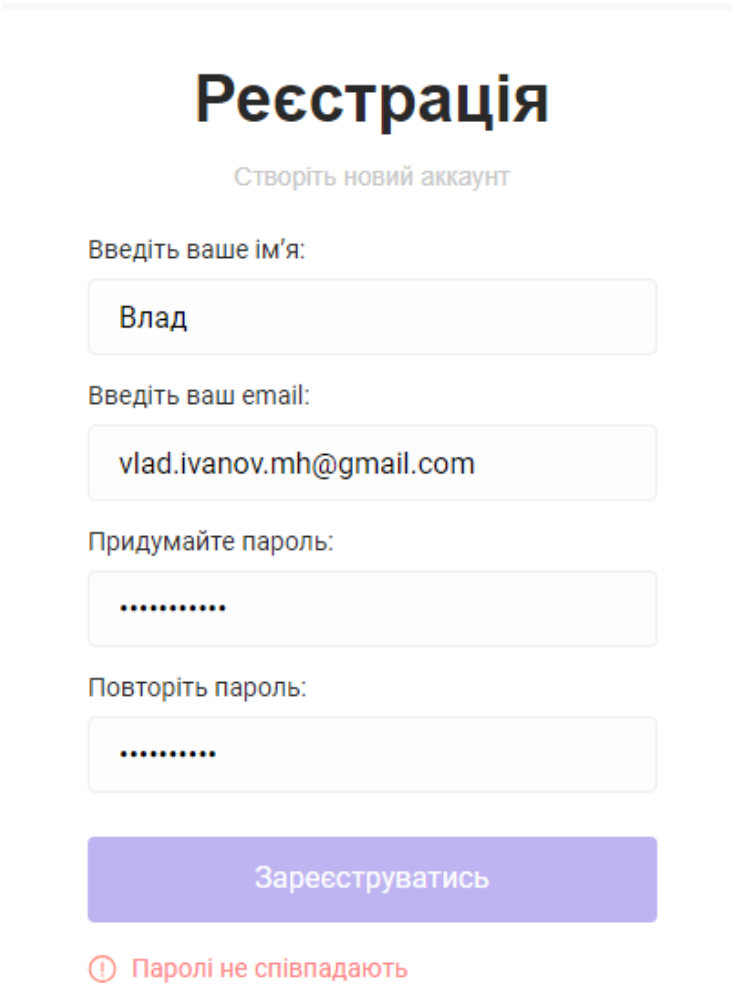

Рисунок 4.3 — Приклад виведення повідомлення про помилку

Також врахована перевірка коректності введення електронної пошти користувачем. При некоректному введенні електронної пошти на сервері відбувається перевірка, яка визначає, чи відповідає введений формат правилам коректної адреси електронної пошти.

Якщо введені дані не відповідають критеріям коректної електронної пошти, сервер повертає відповідне повідомлення про помилку. На клієнтській частині веб-застосунку реалізовано ефективне виведення цього повідомлення користувачеві на екран (рисунок 4.4).

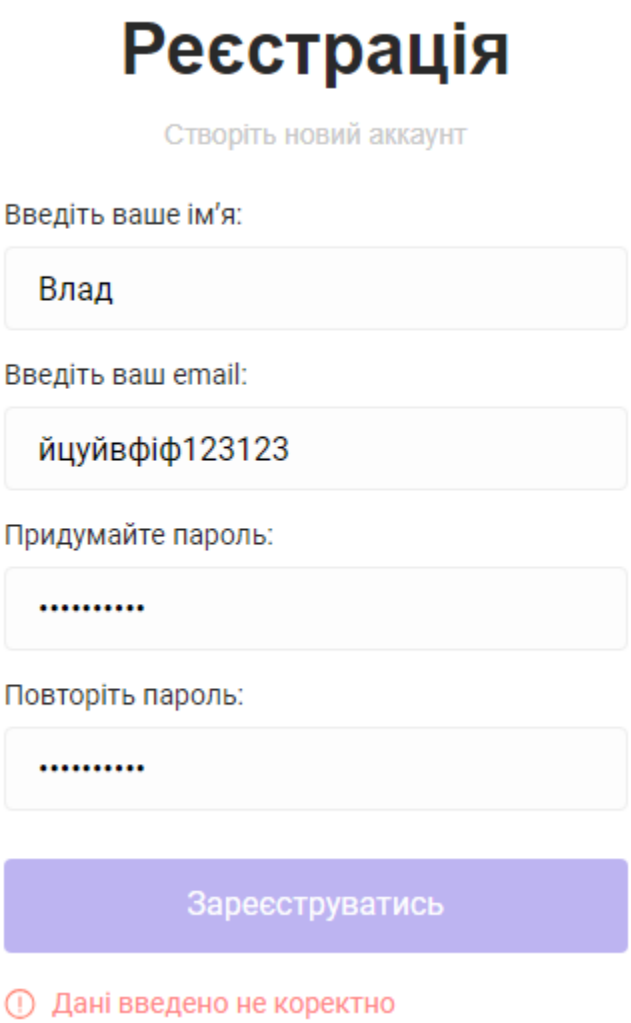

Рисунок 4.4 — Приклад виведення повідомлення про некоректні дані

Важливим аспектом тестування реєстрації є перевірка механізму унікальності реєстрації для кожного користувача. На сервері реалізовано додаткову перевірку, яка визначає, чи існує вже користувач зареєстрований з введеною електронною поштою. У випадку виявлення дубліката, сервер повертає відповідне повідомлення про помилку, попереджаючи про неможливість реєстрації з вже існуючою електронною поштою.

На клієнтській частині веб-застосунку реалізовано виведення цього повідомлення про помилку користувачеві на екран (рисунок 4.5). Такий механізм гарантує, що користувач отримає чітке та зрозуміле повідомлення, що забороняє реєстрацію з вже зареєстрованою електронною поштою. Це сприяє уникненню непорозумінь та покращує користувацький досвід.

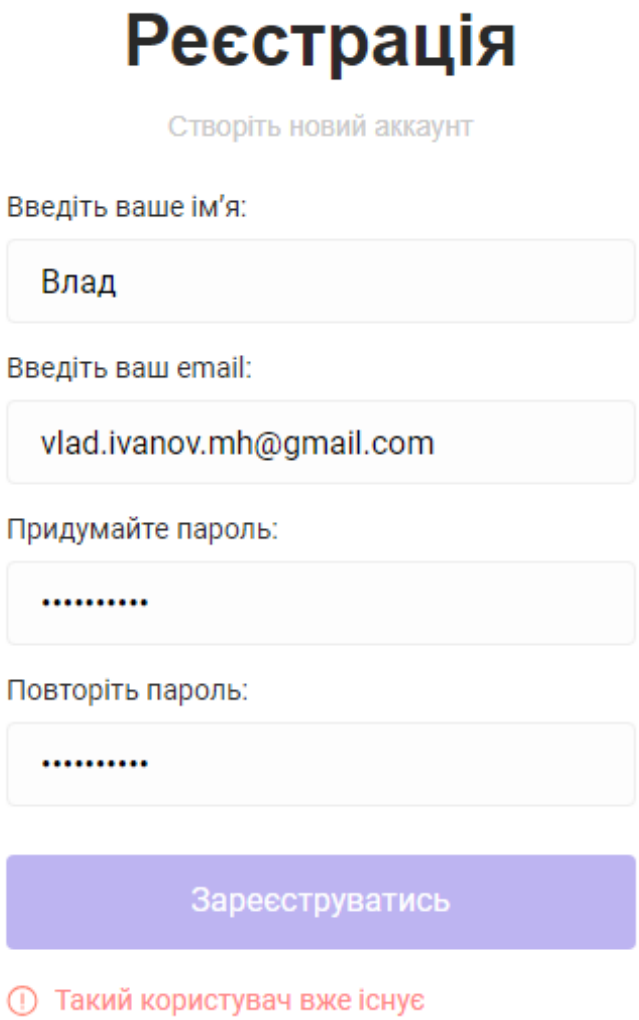

Рисунок 4.5 — Приклад відображення повідомлення про існуючого користувача

Тестування автентифікації включає в себе перевірку низки важливих аспектів, щоб гарантувати безпечність та правильність входу користувача. Одним елементів є перевірка коректності введеної електронної пошти, аналогічно реєстрації.

Додатково, тестування повинно охоплювати випадки, коли користувач вводить невірний пароль. В такому випадку система повинна адекватно реагувати та надавати зрозуміле повідомлення про помилку, щоб користувач мав можливість виправити невірні дані.

Для перевірки того, як система реагує на невірний пароль, введемо коректну електронну пошту, яка раніше була використана при реєстрації та невірний пароль для цього облікового запису (рисунок 4.6).

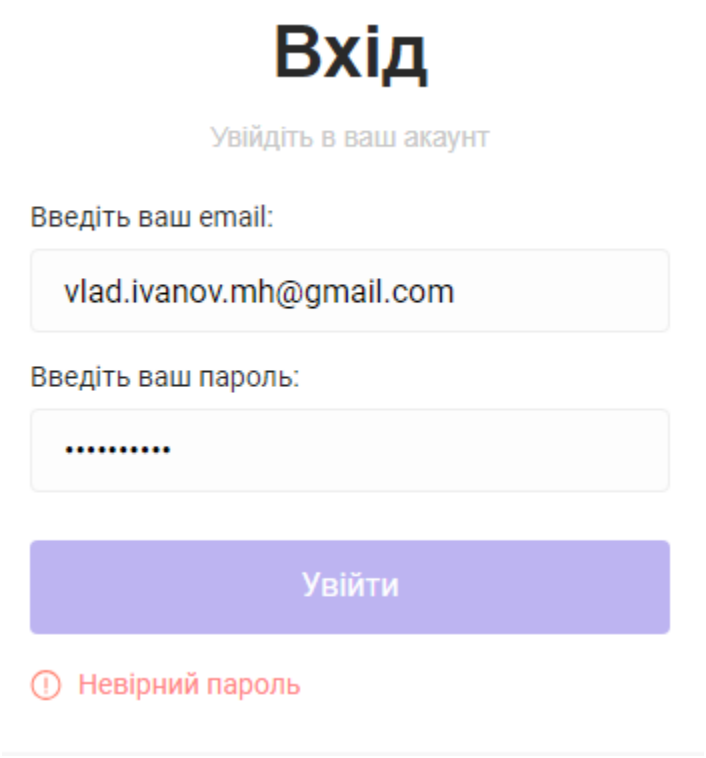

Рисунок 4.6 — Приклад відображення повідомлення про введення невірного паролю

Результати тестування реєстрації та автентифікації вказують на те, що обидві функціональності працюють коректно та ефективно. Клієнтська частина веб-застосунку відповідно обробляє основні помилки, які можуть виникати під час введення даних користувачем.

Під час тестування форми введення фізіологічних даних та образу життя було враховано, що усі дані є обов'язковими для заповнення. В разі некоректного або неповного введення інформації, користувач отримує повідомлення про необхідність заповнення всіх полів форми перед відправленням (рисунок 4.7).

Форма введення фізіологічних даних та образу життя грає ключову роль у зборі важливих даних для подальшого створення дієти або тренувань. Ці дані, такі як вік, вага, зріст, рівень активності, стать та обрана ціль (схуднення, підтримка ваги або набір ваги), визначають індивідуальні потреби користувача. Тому правильне заповнення цієї форми дозволяє системі надати персоналізовані рекомендації з питань харчування та фізичної активності.

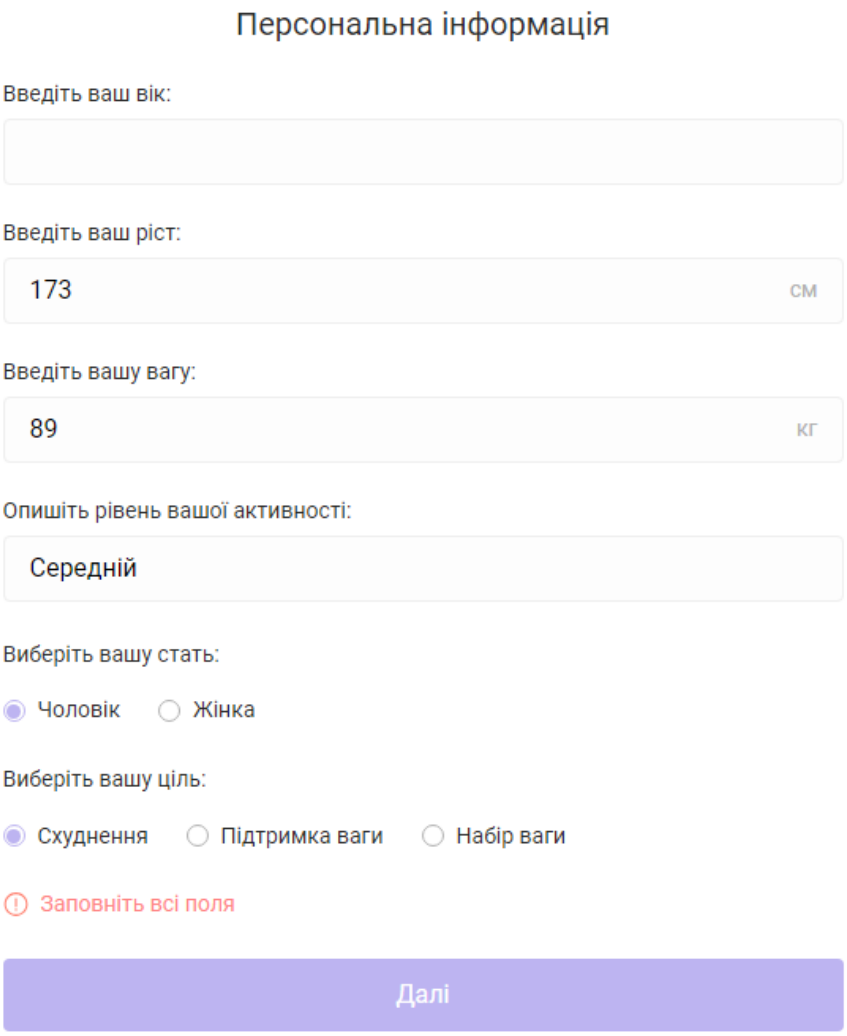

Рисунок 4.7 — Відображення повідомлення про необхідність заповнення всіх полів

Тестові результати свідчать про правильну роботу системи у визначенні коректності введених даних, а також висвітлення відповідних повідомлень про стан форми. Це сприяє надійності та коректності збору фізіологічних даних та образу життя користувачів для подальшого використання в персоналізованих планах харчування та фізичної активності.

Також під час тестування було проведено перевірку коректності виведення відповіді після відправки даних з форми. Система вірно обробляє введені дані та виводить інформативні повідомлення, що дозволяє користувачам отримати зрозумілу і зручну інформацію щодо їхніх тренувань та дієт (рисунок 4.8).

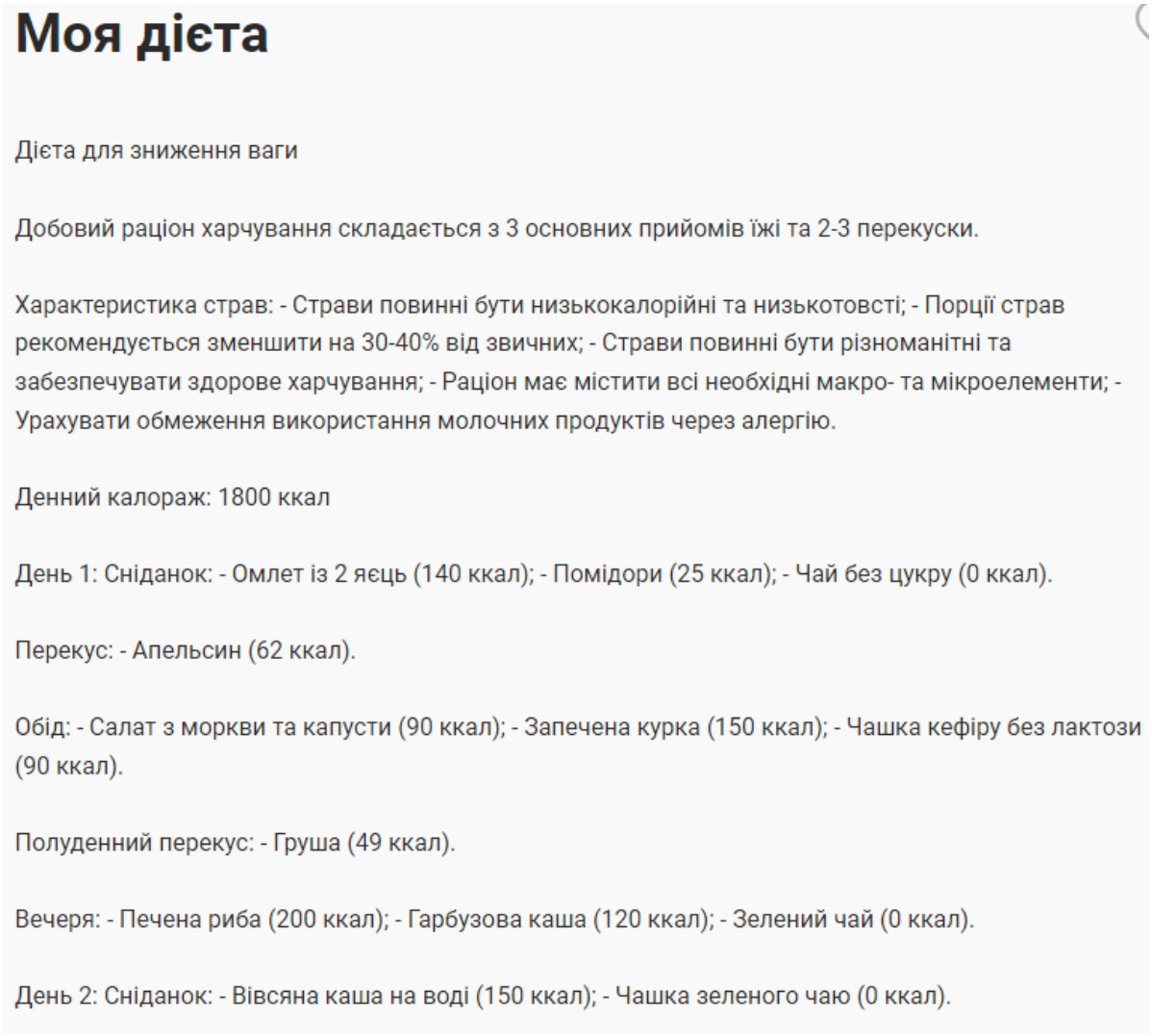

Рисунок 4.8 — Відображення інформації про дієту користувача

В результаті проведеного тестування було виявлено, що веб-застосунок ефективно взаємодіє з користувачем на різних етапах, таких як реєстрація, автентифікація та введення фізіологічних даних. Форми взаємодії із системою продемонстрували високий рівень коректності та зручності для користувача.

Система вірно обробляє різні сценарії, пов'язані з реєстрацією та автентифікацією, і ефективно взаємодіє з користувачем у важливих аспектах. Користувачі можуть отримати інформативні повідомлення про стан форм, а також чітко розуміють результати своїх дій.

За результатами тестування клієнтської частини можна зробити висновок, що клієнтська частина веб-застосунку протестована, працює коректно та може бути використана для організації тренувань та підбору дієти.

5 ЕКОНОМІЧНА ЧАСТИНА

Ефективне впровадження науково-технічної розробки стає можливим, якщо вона відповідає поточним вимогам науково-технічного прогресу та враховує економічні аспекти. Надання оцінки економічної ефективності отриманих результатів науково-дослідної роботи є важливою частиною цього процесу.

Магістерська робота, що присвячена розробці та дослідженню “Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik”, віднесена до науково-технічних робіт, спрямованих на введення на ринок. Рішення про комерціалізацію розробки може бути прийняте протягом самої роботи, дозволяючи реалізувати можливість виведення її на ринок. Цей напрямок розглядається як пріоритетний, оскільки розроблені результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Однак для успішного втілення цього процесу важливо знайти зацікавленого інвестора, який був би зацікавлений у реалізації цього проекту, і переконати його в обґрунтованості таких інвестицій.

Для цього визначені наступні етапи виконання робіт:

— проведено комерційний аудит науково-технічної розробки, що включає в себе визначення науково-технічного рівня та комерційного потенціалу;

— розраховані витрати на реалізацію науково-технічної розробки;

— проведено розрахунок економічної ефективності науково-технічної розробки в разі її впровадження та комерціалізації потенційним інвестором, а також обґрунтовано економічну доцільність комерціалізації для інвестора.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою “Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1

Таблиця 5.1 — Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

Продовження таблиці 5.1

	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти	Необхідне незначне навчання фахівців та збільшення їх	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної
	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси. Джерела	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела	Не потребує додаткового фінансування
0	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуют
1	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій
2	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь— які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно зведені до таблиці 5.2. Для опитування було залучені експерти: к.пед.н. доц. кафедри Войцеховська О.В., к.пед.н., доц. Добровольська Н.В., к.т.н, доц. Тарновський М.Г.

Таблиця 5.2 — Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Войцеховська О.В. к.т.н, доц.	Добровольська Н.В. к.т.н, доц.	Тарновський М.Г. к.т.н, доц.
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	4	4	4
5. Ринкові переваги (експлуатаційні витрати)	4	3	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	$СБ_1$=43	$СБ_2$=42	$СБ_3$=44
Середньоарифметична сума балів $СБ_с$	(43+42+44)/3 = 43		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3

Таблиця 5.3 — Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik» становить 43 бали, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

Результатом магістерської кваліфікаційної роботи Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik» — веб-застосунок для надання користувачам дієт та програм тренувань під їх індивідуальні потреби.

5.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою:

$$q_i = \frac{P_i}{P_{базі}}. \qquad (5.1)$$

де q_i — одиничний параметричний індекс, розрахований за *i*-м параметром;

P_i — значення і-го параметра виробу;

$P_{базі}$ — аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.4.

Таблиця 5.4 — Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Швидкодія, %	90	94	1,02	20
SEO-оптимізація, %	92	95	1,03	30
SSL-шифрування, %	96	93	1,03	30
Адаптивність, %	90	95	1,06	20

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 — пристрій відповідає нормам і стандартам; 0 — не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою**:**

$$I_{НП} = \prod_{i=1}^{n} q_i \,, \tag{5.2}$$

де $I_{нп}$ — загальний показник конкурентоспроможності за нормативними параметрами;

q_i — одиничний (частинний) показник за *i*-м нормативним параметром;

n — кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \,, \tag{5.3}$$

де $I_{тп}$ — груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i — одиничний параметричний показник і-го параметра;

α_i — вагомість і-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n — кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{тп} = 1{,}02 \cdot 0{,}2 + 1{,}03 \cdot 0{,}3 + 1{,}03 \cdot 0{,}3 + 1{,}06 \cdot 0{,}2 = 0{,}93.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \,, \tag{5.4}$$

де $I_{ЕП}$ — груповий параметричний індекс за економічними показниками;

q_i — економічний параметр і-го виду;

β_i — частка і-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m — кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці.

$$I_{ЕП}=0,76 \cdot 0,5 + 0,84 \cdot 0,5 = 0,80.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \quad (5.5)$$

$$K_{ІНТ} = 1 \cdot 0,93 / 0,80 = 1,16.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему “Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik”, під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями..

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці [24].

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{М_{пі} \cdot t_i}{Т_р}, \quad (5.6)$$

де k — кількість посад дослідників залучених до процесу досліджень;

$М_{пі}$ — місячний посадовий оклад конкретного дослідника, грн;

$t_і$ — число днів роботи конкретного дослідника, дн.;

$Т_р$ — середнє число робочих днів в місяці, $Т_р$=21 дні.

$$З_о = 42000 \cdot 10 / 21 = 19091 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.5.

Таблиця 5.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	42000	1909,1	10	19091
Інженер-програміст	39000	1772,7	55	97500
Всього				116591

Витрати на основну заробітну плату робітників (Зр) за відповідними найменуваннями робіт НДР на тему “Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.7)$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.8)$$

де M_M — розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо Мм=6500 грн;

K_i — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2);

K_c — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p — середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ — тривалість зміни, год.

$$C_1 = 6500{,}00 \cdot 1 \cdot 1{,}65 / (21 \cdot 8) = 65{,}8 \text{ грн.}$$

$$З_{р1} = 65{,}8 \cdot 2 = 131{,}6 \text{ грн.}$$

Результати приведено в таблиці 5.6

Таблиця 5.6 — Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1.Підготовчі	2	1	65,8	131,6
2.Налагоджувальні	10	2	72,4	723,8
3.Випробувальні	2	4	98,7	197,4

Продовження таблиці 5.6

Всього	1052,9

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.9)$$

де $Н_{дод}$ — норма нарахування додаткової заробітної плати. Приймемо 11%.

$$З_{дод} = (116591 + 1052,9) \cdot 11 / 100\% = 12940,81 \text{ грн.}$$

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.10)$$

де $Н_{зп}$ — норма нарахування на заробітну плату. Приймаємо 22%.

$$Зн = (116591 + 1052,9 + 12940,81) \cdot 22 / 100\% = 28728,61 \text{ грн.}$$

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення

досліджень за темою “Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik».

Витрати на матеріали (*М*) (таблиця 5.7), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (5.11)$$

де H_j — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

$Ц_j$ — вартість матеріалу j-го найменування, грн/кг;

K_j — коефіцієнт транспортних витрат, (K_j = 1,1 … 1,15);

B_j — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів *j*-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір А 4	146	1	146
Ручка	14	1	14
Диск оптичний OPTIMA CD	15	1	15
Flesh-пам'ять GOODRAM 64 C10A	410	1	410
Всього			585
З врахуванням коефіцієнта транспортування			643,5

5.3.4 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.12)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = (45000 \cdot 1) / (2 \cdot 12) = 1875 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.8.

Таблиця 5.8— Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп’ютер	45000	2	1	1875,00
Приміщення лабораторії	190000	20	1	791,67
Всього				2666,67

5.3.5 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_i \cdot Ц_е \cdot K_{впі}}{\eta_i}, \quad (5.13)$$

де $W_{уі}$ — встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 7,5 грн;

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі}$ <1;

$\eta_і$ — коефіцієнт корисної дії обладнання, $\eta_і$<1.

$$В_е = 0,25 \cdot 250,0 \cdot 7,5 \cdot 0,5 / 0,8 = 292,97 \text{ грн.}$$

5.3.6 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему “Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik” належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (5.14)$$

де $H_{св}$ — норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$$В_{св} = (116591 + 1052,9) \cdot 20 / 100\% = 23528,75 \text{ грн.}$$

5.3.7 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.15)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$$І_в = (116591 + 1052{,}9) \cdot 50 / 100\% = 58821{,}88 \text{ грн.}$$

5.3.8 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.16)$$

де $H_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$$В_{нзв} = (116591 + 1052,9) \cdot 100 / 100\% = 117643,77 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи на тему “Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik” розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.17)$$

$В_{заг}$= 116591+1052,9+12940,81+28728,61+643,5+2666,67+*292,97*+23528,75+ 58821,88+117643,77= 363004,48 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.19)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 363004,48 / 0,9 = 403338,31 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою “Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik” передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN — збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N — кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ — вартість послуги у році до впровадження інформаційної системи, приймемо 2000,00 грн;

$\pm\Delta Ц_о$ — зміна вартості послуги від впровадження результатів, приймемо зростання на 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.21)$$

де λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 40\%$;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (1 \cdot 500 + 2000 \cdot 1500) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18/100\%) = 640684,79 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (1 \cdot 500 + 2000 \cdot (1500 + 1200)) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18/100\%) = 1153578,9 \text{грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (1 \cdot 500 + 2000 \cdot (1500 + 1200 + 850)) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18/100\%) = 1516585,2 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.22)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=18%;

t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 640684{,}79/(1+0{,}18)^1 + 1153578{,}9/(1+0{,}18)^2 + 1516585{,}2/(1+0{,}18)^3 =$$
$$= 2216736{,}01 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (5.23)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 403338,31 грн.

$$PV = k_{інв} \cdot ЗВ = 2 \cdot 403338{,}31 = 806676{,}61 \text{ грн.}$$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \qquad (5.24)$$

де $ПП$ — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 2216736,01 грн;

PV — теперішня вартість початкових інвестицій, 806676,61 грн.

$$E_{абс} = ПП - PV = 2216736{,}01 — 806676{,}61 = 1410059{,}39 \text{ грн.}$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.25)$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн;

PV — теперішня вартість початкових інвестицій, грн;

$T_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+1410059{,}39\ /806676{,}61)^{1/3} — 1 = 0{,}65.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.26)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f — показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,65 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.27}$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 0,65 = 1,5 \text{ р.}$$

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

ВИСНОВОК

У комплексній магістерській кваліфікаційній роботі спроектовано та розроблено клієнстьку частину веб-застосунку для організації тренувань та дієти з інтеграцією штучного інтелекту фреймворком Qwik.

Проведено аналітичний огляд технологій для розробки клієнтської частини веб-застосунку. Розглянуто базові технології, на яких базуються сучасні фреймворки та препроцесори для стилізації веб-сторінок. Проведено огляд та порівняння фреймворків Qwik та React, обгрунтовано вибір технології Qwik, оскільки вона орієнтована на досягнення максимальної продуктивності та оптимізації роботи веб-застосунку та зосереджується на уникненні надмірного завантаження JavaScript, що призводить до швидкого завантаження сторінок та поліпшення інтерактивності для користувача.

Вдосконалено метод відображення веб-сторінки, в якому відсутня гідратація та використовується припинення рендерингу на сервері та його відновлення на стороні клієнта без необхідності завантаження всієї логіки веб-застосунку, що дає можливість підвищити швидкодію завантаження веб-сторінки. Проаналізовано функціональні та нефункціональні вимоги до клієнтської частини веб-застосунку для організації тренувань та дієти. Також спроектовано структуру клієнтської частини веб-застосунку, визначено основні елементи та зв’язки між ними.

Розроблено клієнтську частину веб-застосунку для організації тренувань та дієти. При цьому використано компонентний підхід, що сприяє поділу функціоналу на окремі компоненти, що полегшує їх перевикористання та підтримку. Кожен компонент відповідає за конкретний аспект функціоналу або інтерфейсу, що зробило структуру більш зрозумілою та легко розширюваною.

Під час розробки веб-застосунку були створені ключові компоненти, такі як хедер, футер, форми реєстрації та автентифікації, а також форма для введення фізіологічних даних користувача. Реалізовано можливість реєстрації та автентифікації, а також забезпечено легку розширюваність та масштабованість системи.

Проведено тестування клієнтської частини веб-застосунку, яке підтвердило коректну працездатність та ефективну обробку помилок під час використання веб-застосунку користувачем. Проаналізовано та порівняно веб-застосунок розроблений на React та Qwik. Результати цього порівняння вказують на переваги Qwik у різних аспектах тестування, привертаючи особливу увагу до високої продуктивності фреймворку.

В роботі проведений економічний аналіз доцільності розробки. Абсолютний економічний ефект складає 1410059,39 грн, що свідчить про комерційну привабливість науково-технічної розробки. Термін окупності складає 1,5 роки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Аналіз технологій створення веб-клієнту за допомогою реактивного фреймворка React / В.М. Іванов, О.В. Войцеховська. Матеріали LI наукової-технічної конференції підрозділів Вінницького національного технічного університету (Вінниця, 2022 р.) [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/author/submission/15736.

2. Аналіз технологій розробки клієнтської частини веб-додатків / В.М. Іванов, О.В. Войцеховська, О.С. Городецька. Матеріали Молодь в науці: дослідження, проблеми, перспективи (МН-2024), 2023 р. [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/ viewFile/19089/15839

3. Методичні вказівки до виконання магістерських кваліфікаційних робіт студентами спеціальності 123 «Комп'ютерна інженерія». / Укладачі
О. Д. Азаров, О. В. Дудник, С. І. Швець − Вінниця : ВНТУ, 2023.

4. Гіпертекстова розмітка HTML. [Електронний ресурс]. Режим доступу: http://www.znannya.org/?view=html_teach

5. Каскадні таблиці стилів. [Електронний ресурс]. — Режим доступу: https://naurok.com.ua/urok-kaskadni-tablici-stiliv-css-90218.html

6. Препроцесори CSS. [Електронний ресурс]. — Режим доступу: https://vyspiansky.gitbook.io/introduction-to-web-development/css/preprocessors.

7. Мова JavaScript та її можливості. [Електронний ресурс]. Режим доступу: https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti

8. Мова TipeScript. [Електронний ресурс]. — Режим доступу: https://www.typescriptlang.org/docs/

9. Фреймворк Qwik. [Електронний ресурс]. — Режим доступу: https://qwik.builder.io/docs/concepts/think-qwik/

10. Маршрутизація Qwik. [Електронний ресурс]. — Режим доступу: https://qwik.builder.io/docs/routing/

11. Стан у Qwik [Електронний ресурс]. — Режим доступу: https://qwik.builder.io/docs/components/state/

12. Фреймворк React. [Електронний ресурс]. — Режим доступу: https://uk.reactjs.org/docs/getting-started.html

13. Virtual DOM. [Електронний ресурс]. — Режим доступу: https://www.codecademy.com/article/react-virtual-dom

14. React Router. [Електронний ресурс]. — Режим доступу: https://o7planning.org/12137/undertanding-react-router-with-example-on-the-client-side

15. Основи Webpack [Електронний ресурс]. — Режим доступу: https://codeguida.com/post/454

16. Основи Vite [Електронний ресурс]. — Режим доступу: https://vitejs.dev/guide/why.html.

17. Pagespeed Insights [Електронний ресурс]. — Режим доступу: https://developers.google.com/speed/docs/insights/v5/about?hl=ua.

18. CSR та SSR rendering [Електронний ресурс] — Режим доступу до ресурсу: https://dou.ua/forums/topic/31720/

19. Resumability [Електронний ресурс] — Режим доступу до ресурсу: https://qwik.builder.io/docs/concepts/resumable/

20. Reactivity [Електронний ресурс] — Режим доступу до ресурсу: https://qwik.builder.io/docs/concepts/reactivity/

21. Components [Електронний ресурс] — Режим доступу до ресурсу: https://qwik.builder.io/docs/components/overview/

22. State [Електронний ресурс] — Режим доступу до ресурсу: https://qwik.builder.io/docs/components/state/

23. Tasks and Lifecycle [Електронний ресурс] — Режим доступу до ресурсу: https://qwik.builder.io/docs/components/tasks/

24. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. — 42 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. О. Д. Азаров
“ ” 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання комплексної магістерської кваліфікаційної роботи
«Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. Клієнтська частина з використанням фреймворку Qwik»
08-54.КМКР.050.00.000 ПЗ

Науковий керівник к.т.н., доц. каф. ОТ
Городецька О. С.
Студент групи 1КІ-22м
Іванов В. М.

Вінниця 2023

1 Підстава для використання КМКР

1.1 На сьогоднішній день існує чимало онлайн та оффлайн ресурсів, які надають розписане харчування та програми тренувань. Проте деякі застосунки не надають достатньо персоналізованих рекомендацій, що зменшує їх ефективність. Також якщо веб-застосунок не оптимізований, він завантажується повільно, що погіршує користувацький досвід. Тому розробка веб-застосунку для організації тренувань та дієти з підвищеною швидкодією є актуальною задачею.

1.2 Наказ про затвердження теми кваліфікаційної роботи.

2 Мета і призначення КМКР

2.1 Метою комплексної магістерської роботи є підвищення швидкодії завантаження клієнтської частини веб-застосунку для організації тренувань та дієти шляхом вдосконалення методу відображення веб-сторінок;

2.2 Призначення розробки — реалізована клієнтська частина веб-застосунку, який надає можливість створення дієт та програм тренування для індивідуальних потреб користувачів.

3 Вихідні данні для виконання КМКР

3.1 Проведення аналізу технологій для реалізації клієнстької частини;

3.2 Вдосконалення методу відображення веб-сторінок;

3.3 Розробка клієнтської частини веб-застосунку;

3.4 Тестування клієнтської частини веб-застосунку;

3.5 Виконання розрахунків для доведення доцільності нової розробки.

4 Технічні вимоги до виконання КМКР

Основними вимогами до виконання КМКР є:

— наявність клієнтської частини, яка буде адаптивна під різні пристрої;

— наявність функціоналу для коректної передачі даних на сервер.

5 Етапи КМКР та очікувані результати

Робота виконується за п’ять етапів, таблиця А.1.

Таблиця А.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз технологій для розробки клієнстької частини веб-застосунку	04.10.2023	15.10.2023	Аналітичний огляд літературних джерел
2	Вдосконалення методу відображення веб-сторінки та розробка клієнстької частини веб-застосунку	16.10.2023	04.11.2023	2,3 розділ
3	Тестування та перевірка працездатності клієнтської частини веб-застосунку	05.11.2023	30.11.2023	4 розділ
4	Підготовка економічної частини	30.11.2023	03.12.2023	5 розділ
5	Оформлення пояснювальної записки, графічного матеріалу і/або презентації	04.12.2023	18.12.2023	пояснювальна записка, графічний матеріал і/або презентація

6 Матеріали, що подаються до захисту КМКР

До захисту КМКР подаються: пояснювальна записка КМКР, графічні і ілюстративні матеріали, протокол попереднього захисту КМКР на кафедрі, відзив наукового керівника, відзив опонента, протоколи проходження перевірки на плагіат, анотації до КМКР українською та іноземною мовами, нормоконтроль про відповідність оформлення КМКР діючим вимогам.

7 Порядок контролю виконання та захисту КМКР

Виконання етапів графічної та розрахункової документації КМКР контролюється науковим керівником згідно зі встановленими термінами. Захист

КМКР відбувається на засіданні екзаменаційної комісії, затверджено наказом ректора.

8 Вимоги до оформлення та порядок виконання КМКР

8.1 При оформлюванні КМКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп'ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання КМКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02—П.001.01:21»

ДОДАТОК Б

Схема процесу рендеренгу веб-сторінки різними методами

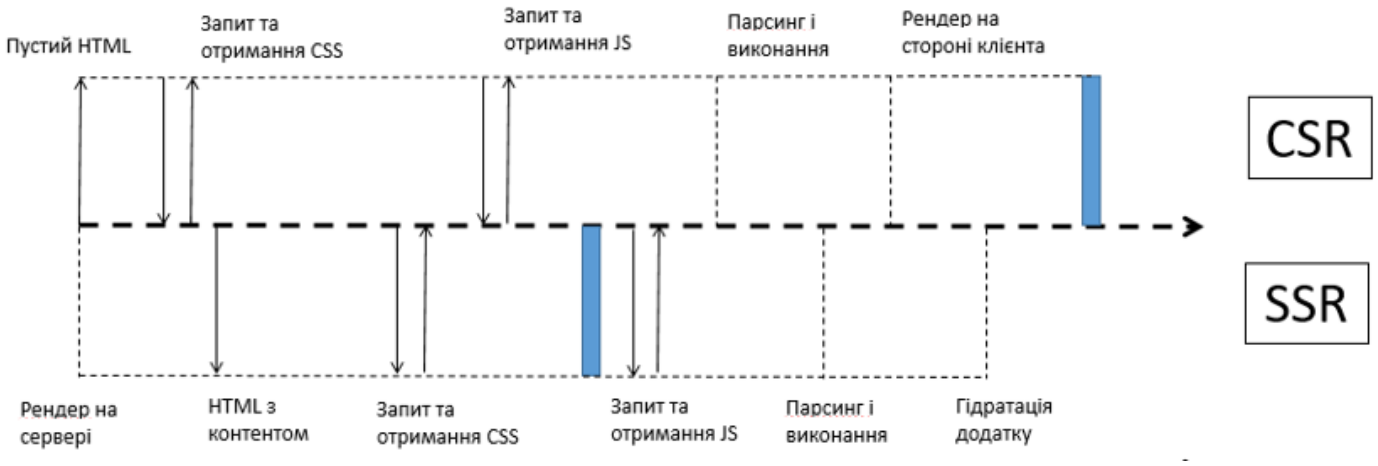

Рисунок Б.1 — Схема процесу рендеренгу сторінки різними методами

ДОДАТОК В

Структурна схема процесу відображення веб-сторінки з гідратацією та її відсутністю

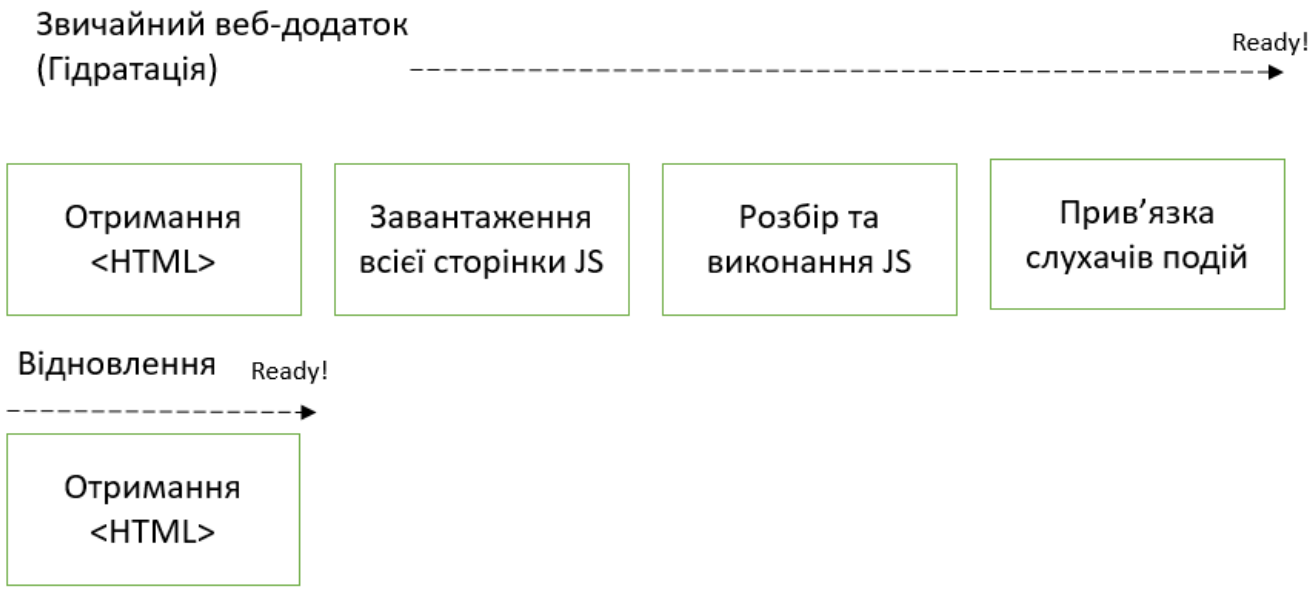

Рисунок В.1 — Структурна схема процесу відображення веб-сторінки з гідратацією та її відсутністю

ДОДАТОК Г

Структурна схема вдосконаленого методу відображення веб-сторінки

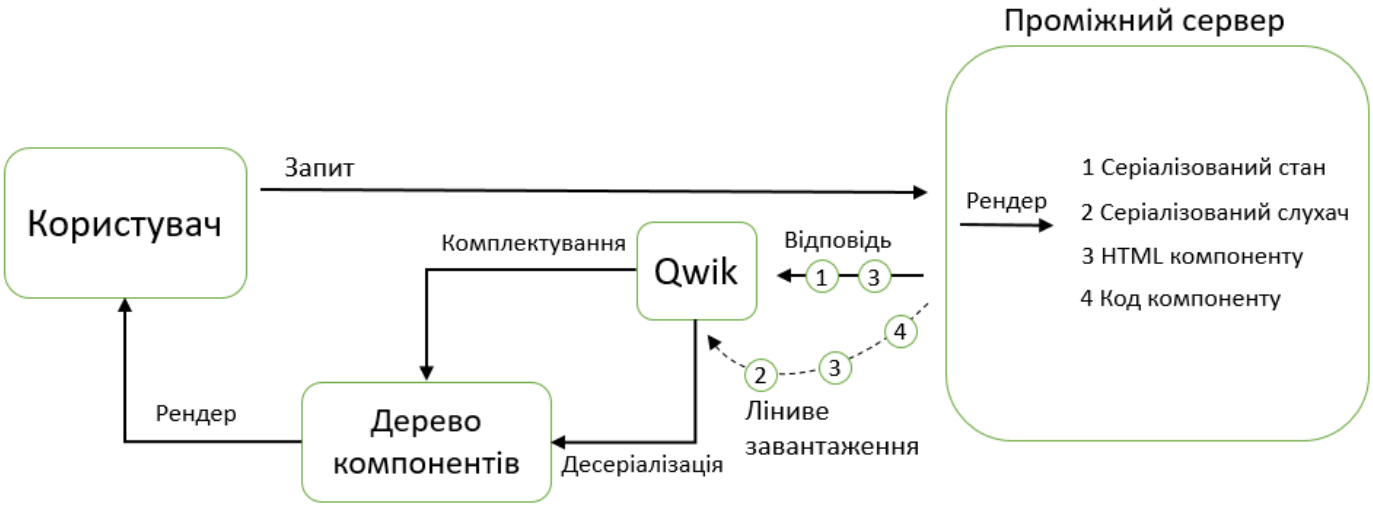

Рисунок Г.1 — Структурна схема вдосконаленого методу відображення веб-сторінки

ДОДАТОК Д

Структурна схема клієнтської частини веб-застосунку

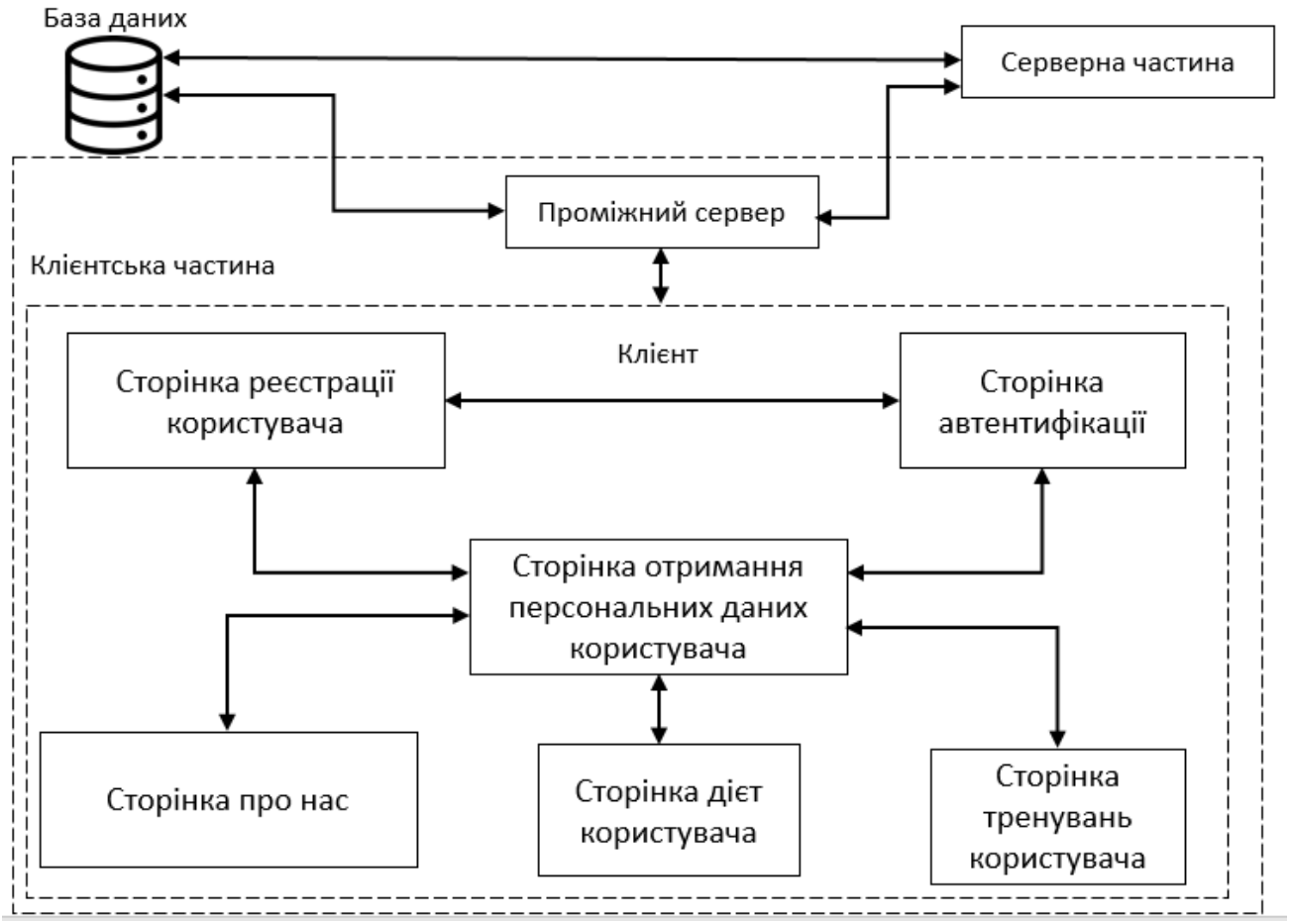

Рисунок Д.1 — Структурна схема клієнтської частини веб-застосунку

ДОДАТОК Е

Лістинг форми реєстрації користувача

```
import { $, component$, JSXNode, useSignal } from '@builder.io/qwik';
import { SubmitHandler, useForm } from '@modular-forms/qwik';
import { RegisterForm, useFormLoader } from '~/routes/auth/register';
import { useAuth } from '~/hooks/useAuth';
import { api } from '~/api';
import styles from '~/components/LoginForm/LoginForm.module.scss';
import Button from '~/components/Button';
import Title from '~/components/Title';
import Loader from '~/components/Loader';
import FormInput from '~/components/FormInput';
import { Link } from '@builder.io/qwik-city';
import EmailIcon from '~/assets/icons/email.svg?jsx';
import PassWordIcon from '~/assets/icons/password.svg?jsx';
import PassWordRepeatIcon from '~/assets/icons/password-repeat.svg?jsx';
import NameIcon from '~/assets/icons/name.svg?jsx';
type RegisterFormKeys = keyof RegisterForm;
interface RegisterFormFields {
  icon: JSXNode;
  label: string;
  fieldName: RegisterFormKeys;
  type: 'text' | 'email' | 'password';
}
const formFields: RegisterFormFields[] = [
  {
    icon: <NameIcon />,
    label: 'Введіть ваше ім’я:',
    fieldName: 'name',
    type: 'text',
  },
  {
    icon: <EmailIcon />,
    label: 'Введіть ваш email:',
```

```
        fieldName: 'email',
        type: 'email',
    },
    {
        icon: <PassWordIcon />,
        label: 'Придумайте пароль:',
        fieldName: 'password',
        type: 'password',
    },
    {
        icon: <PassWordRepeatIcon />,
        label: 'Повторіть пароль:',
        fieldName: 'repeatPassword',
        type: 'password',
    },
];
export default component$(() => {
    const isLoading = useSignal(false);
    const error = useSignal('');
    const auth = useAuth();
    const [, { Form, Field }] = useForm<RegisterForm>({
        loader: useFormLoader(),
    });
    const handleSubmit = $<SubmitHandler<RegisterForm>>(async (values) => {
        if (values.password === values.repeatPassword) {
            isLoading.value = true;
            const response = await api.registerUser(values);
            isLoading.value = false;
            if (response.isError) {
                // @ts-ignore
                error.value = response.error?.message ?? 'Error';
                return;
            }
            const loginResponse = await auth.loginUser({
                email: values.email,
```

```
            password: values.password,
        });
        if (loginResponse.isError) {
            // @ts-ignore
            error.value = loginResponse.error?.message ?? 'Error';
            return;
        }
    } else {
        error.value = 'Паролі не співпадають';
    }
});
return (
    <>
        <div class={styles.form}>
            <div class={styles.blockTitle}>
                <Title>Реєстрація</Title>
                <p class={styles.titleText}>Створіть новий аккаунт</p>
            </div>
            {isLoading.value && <Loader class={styles.loader} size={36} />}
            <Form class={styles.formWrapper} onSubmit$={handleSubmit}>
                {formFields.map((fields) => (
                    <Field name={fields.fieldName} key={fields.fieldName}>
                        {(field, props) => {
                            return (
                                <FormInput
                                    {...props}
                                    name={fields.fieldName}
                                    label={fields.label}
                                    type={fields.type}
                                    value={field.value}
                                >
                                    {fields.icon}
                                </FormInput>
                            );
                        }}
```

```
                </Field>
              ))}
              <Button type="submit" class={styles.formButton}>
                Зареєструватись
              </Button>
              {error.value && (
                <div class={styles.formError}>{error.value}</div>
              )}
            </Form>
          </div>
          <p class={styles.lastText}>
            Маєте аккаунт?{' '}
            <Link href="/auth/login" class={styles.link}>
              Увійти
            </Link>
          </p>
        </>
    );
});
```

ДОДАТОК Ж

Лістинг форми автентифікації користувача

```
import { $, component$, JSXNode, useSignal } from '@builder.io/qwik';
import { Link } from '@builder.io/qwik-city';
import { SubmitHandler, useForm } from '@modular-forms/qwik';
import { LoginForm, useFormLoader } from '~/routes/auth/login';
import styles from './LoginForm.module.scss';
import FormInput from '../FormInput';
import EmailIcon from '~/assets/icons/email.svg?jsx';
import PassWordIcon from '~/assets/icons/password.svg?jsx';
import { useAuth } from '~/hooks/useAuth';
import Title from '../Title';
import Button from '~/components/Button';
import Loader from '~/components/Loader';
type LoginFormKeys = keyof LoginForm;
interface LoginFormFields {
   icon: JSXNode;
   label: string;
   fieldName: LoginFormKeys;
   type: 'text' | 'email' | 'password';
}
const formFields: LoginFormFields[] = [
   {
      icon: <EmailIcon />,
      label: 'Введіть ваш email:',
      fieldName: 'email',
      type: 'email',
   },
   {
      icon: <PassWordIcon />,
      label: 'Введіть ваш пароль:',
      fieldName: 'password',
      type: 'password',
   },
```

```
];
export default component$(() => {
  const isLoading = useSignal(false);
  const error = useSignal('');
  const auth = useAuth();
  const [, { Form, Field }] = useForm<LoginForm>({
    loader: useFormLoader(),
  });
  const handleSubmit = $<SubmitHandler<LoginForm>>(async (values) => {
    isLoading.value = true;
    const response = await auth.loginUser(values);
    isLoading.value = false;
    if (response.isError) {
      error.value = response.error ?? 'Error';
    }
  });
  return (
    <>
      <div class={styles.form}>
        <div class={styles.blockTitle}>
          <Title>Вхід</Title>
          <p class={styles.titleText}>Увійдіть в ваш акаунт</p>
        </div>
        {isLoading.value && <Loader class={styles.loader} size={36} />}
        <Form class={styles.formWrapper} onSubmit$={handleSubmit}>
          {formFields.map((fields) => (
            <Field name={fields.fieldName} key={fields.fieldName}>
              {(field, props) => {
                return (
                  <FormInput
                    {...props}
                    name={fields.fieldName}
                    label={fields.label}
                    type={fields.type}
                    value={field.value}
```

```
                        >
                            {fields.icon}
                        </FormInput>
                    );
                }}
            </Field>
        ))}
        <Button type="submit" class={styles.formButton}>
            Увійти
        </Button>
        {error.value && (
            <div class={styles.formError}>{error.value}</div>
        )}
    </Form>
</div>
<p class={styles.lastText}>
    Ще не маєте акаунту?{' '}
    <Link href="/auth/register" class={styles.link}>
        Зареєструватись
    </Link>
</p>
</>
);
});
```

ДОДАТОК И

Протокол перевірки кваліфікаційної робити

Назва роботи: Веб-застосунок для організації тренувань та дієти з інтеграцією штучного інтелекту. Частина 2. «Клієнтська частина з використанням фреймворку Qwik»

Тип роботи: комплексна магістерська кваліфікаційна робота
(БДР, КМКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95,4% Схожість 4,6%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Іванов В.М.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Городецька О. С.
(підпис) (прізвище, ініціали)