МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«___»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ **“БАКАЛАВР”**

Тема: «Вебзастосунок соціальної екосистеми для ентузіастів криптовалют з використанням фреймворку Next.js»

Виконавець: Ангеліна ЛЯПІН

Керівник: к.т.н., доцент кафедри КІТ Олена ТОЛСТІКОВА

Нормоконтролер: к.т.н., доцент Вікторія СИДОРЕНКО

КИЇВ 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)
«____»_____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Ляпін Ангеліни Вікторівни
(прізвище, ім'я, по батькові здобувача вищої освіти в родовому відмінку)

1. Тема кваліфікаційної роботи: «Вебзастосунок соціальної екосистеми для ентузіастів криптовалют з використанням фреймворку Next.js», затверджена наказом ректора від «05» квітня 2024 р. № 517/ст.
2. Термін виконання роботи: з 06 травня 2024 року по 16 червня 2024 року.
3. Вихідні дані до роботи: з редактор коду Visual Studio Code, мова програмування Typescript, фреймворк Next.js, бібліотека для управління станом додатку Zustand.
4. Зміст пояснювальної записки: з 1) Основи створення вебзастосунків та огляд блокчейн технологій. 2) Технології проектування вебзастосунків. 3) Розробка вебзастосунку.
5. Перелік графічного (ілюстративного) матеріалу: слайди презентації Power Point: 1) Актуальність та практична цінність. 2) Мета, об'єкт, предмет та наукова новизна кваліфікаційної роботи. 3) Задачі кваліфікаційної роботи. 4) Інструменти та технології розробки вебзастосунку. 5) Приклад роботи вебзастосунку.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Огляд теоретичних аспектів створення вебзастосунків та блокчейн-технологій.	06.05.2024–08.05.2024	
2	Вибір та аналіз технологій для розробки вебзастосунку, включаючи оцінку їх переваг та недоліків.	09.05.2024–11.05.2024	
3	Визначення основної функціональності проекту.	12.05.2024–17.05.2024	
4	Огляд файлової структури проекту. Розробка вебзастосунку.	18.05.2024–23.05.2024	
5	Перевірка на правильне функціонування програми	24.05.2024–25.05.2024	
6	Написання пояснювальної записки дипломного проекту	26.05.2024–30.05.2024	
7	Написання тексту доповіді. Оформлення додатків	31.05.2024–07.06.2024	

7. Дата видачі завдання «06» травня 2024 р.

Керівник кваліфікаційної роботи _____ Олена ТОЛСТІКОВА
(підпис керівника)

Завдання прийняв до виконання _____ Ангеліна ЛЯПІН
(підпис здобувача вищої освіти)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Вебзастосунок соціальної екосистеми для ентузіастів криптовалют з використанням фреймворку Next.js» містить: 55 сторінок, 23 рисунки, 16 інформаційних джерел.

Об’єкт дослідження: розробка вебзастосуноку соціальної екосистеми.

Предмет дослідження: вебзастосунок на базі фреймворку Next.js соціальної екосистеми для ентузіастів криптовалют.

Мета кваліфікаційної роботи: забезпечити зручний та ефективний інтерфейс для користувачів, дозволяючи їм взаємодіяти, обмінюватися інформацією та отримувати актуальні новини зв’язані з криптовалютою, а також надати можливості для знаходження партнерів та співпраці в криптовалютній галузі.

Методи дослідження: аналіз літературних джерел та предметної області, дослідження ринку, проектування та розробка застосунку.

Наукова новизна: запровадження нового підходу до розробки інтерфейсів користувача з використанням технологій SSR (серверного рендерингу) та SSG (статичного генерації сторінок) на основі сучасного фреймворку Next.js.

Результат кваліфікаційної роботи може бути використаним для вдосконалення процесів у сфері криптовалют.

Ключові слова: ВЕБЗАСТОСУНОК, HTML, CSS, NEXT.JS, TYPESCRIPT, YUP, FORMIK, VISUAL STUDIO CODE, SSR, SSG, ZUSTAND

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

HTML	–	HyperText Markup Language
ООП	–	Object-Oriented Programming
SSR	–	Server-Side Rendering
SSG	–	Static Site Generation
ISR	–	Incremental Static Regeneration
CSR	–	Client-Side Rendering
VS Code	–	Visual Studio Code
ES5	–	ECMAScript 5
ES6	–	ECMAScript 6
ES7	–	ECMAScript 7
CDN	–	Content Delivery Network
SEO	–	Search Engine Optimization

ВСТУП

У сучасному світі криптовалютні ринки набули великого значення. Тепер це не тільки об`єкт інвестицій, а місце активної спільноти ентузіастів для обміну інформацією. Наразі ця сфера з кожним днем розвивається все більше і більше.

Соціальні мережі стають необхідним інструментом для ентузіастів криптовалют. Спеціальна екосистема надає ентузіастам місце для обміну досвідом, та дозволяє отримувати актуальну інформацію щодо ринку, аналізувати ринкові тенденції та спілкуватись з однодумцями. Це може допомогти криптотрейдерам отримувати останні актуальні новини, події в цій сфері, тому, як наслідок, трейдери зможуть приймати обгрунтовані торгівельні рішення.

Метою розробки вебзастосунку є забезпечення покращення взаємодії між ентузіастами криптовалют через активний та актуальний інформаційний обмін, а також розширення можливостей для знаходження партнерів та співпраці в криптовалютній галузі.

Основними цілями розробки вебзастосунку є:

1. Покращення взаємодії між ентузіастами криптовалют з метою створення сприятливого середовища для обміну думками, досвідом та інформацією.

2. Забезпечення актуального інформаційного обміну щодо останніх подій та трендів у світі криптовалют.

3. Розширення мережевих можливостей для знаходження партнерів та співпраці у сфері криптовалютних проектів.

Для досягнення основних цілей потрібно вирішити наступні задачі:

− Розробити можливість створювати, редагувати та видаляти дописи у вебзастосунку.

− Інтегрувати з необхідними інструментами розробки, такими як редактор коду, система контролю версій тощо.

− Встановити та налаштувати Next.js.

− Вивчити очікувані функціональні можливості веб-застосунку.

− Створення основних функцій, таких як реєстрація користувачів, авторизація, створення профілю, перегляд новин про криптовалюту, перегляд економічного календарю (інструмент надає користувачам можливість відстежувати новини та події, які мають велике значення для ринків та можуть вплинути на їх рух), тощо.

− Забезпечити можливість користувачам слідкувати один за одним та взаємодіяти у мережі спільноти.

− Виконати оптимізацію коду для підвищення продуктивності та ефективності роботи веб-застосунку.

− Провести тестування для виявлення та виправлення помилок, а також перевірити забезпечення безпеки застосунку.

Актуальність теми індивідуального завдання полягає в розробці вебзастосунку, який сприятиме активній комунікації та обміну інформацією. Ентузіасти криптовалют потребують спеціалізованого ресурсу, за допомогою якого вони могли б отримувати найновішу інформацію та події в індустрії криптовалют, спілкуватися з однодумцями, та де вся основна інформація буде зібрана в одному місці.

Об'єктом дослідження є процес розробки та впровадження веб-додатку.

Наукова новизна полягає у створенні спеціалізованої платформи для обміну досвідом між однодумцями, з використанням таких технологій як SSR (серверного рендерингу) та SSG (статичної генерації сторінок), які забезпечують високу продуктивність та швидкість завантаження сторінок.

Предметом дослідження є функціональні можливості та ефективність використання фреймворку Next.js для створення вебзастосунку соціальної екосистеми в галузі криптовалют.

РОЗДІЛ 1
ОСНОВИ СТВОРЕННЯ ВЕБЗАСТОСУНКІВ ТА ОГЛЯД БЛОКЧЕЙН-ТЕХНОЛОГІЙ

1.1. Основні поняття вебзастосунків.

Веб-сторінка – це окрема сторінка або документ, зазвичай створений за допомогою HTML, яка розміщена на сервері, та відображається на екрані комп`ютера у браузері. Вебсторінка загалом це те, що ми бачимо, коли відвідуємо веб-сайт у браузері. Вона може мати купу різних матеріалів, таких як текст, медіа – зображення, відео, аудіо, посилання на інші сторінки, тощо.

Веб-сайт - це сукупність загальнодоступних, взаємопов'язаних веб-сторінок, які доступні через одне доменне ім'я. Веб-сайт розміщується на серверах, підключених до веб-мережі, і може редагувати або надсилати інформацію, до якої мають доступ користувачі з усього світу. Деякі сторінки веб-сайту мають посилання, які полегшують користувачеві перехід з однієї сторінки веб-сайту на іншу.

Доменне ім’я – це ім’я, за яким ідентифікується веб-сайт. Домен пов’язаний з певною ір-адресою. Також його можна придбати з будь-якою комбінацією літер, дефісів та цифр. Але важливий момент, те що домен не може починатися з дефіса. Залежно від розширення, як приклад можна взяти .com, .net, .org, доменне ім’я може містити до 63 символів, але не більше. Домен складається з доменів другого рівня (designtlc) і доменів верхнього рівня (com), розділених крапкою. Домени верхнього рівня відображають тип або де саме розташований веб-сайт: .com (комерційний), .gov (державний), .edu (освітній).

Веб-сервер - це комп`ютерні системи, які у себе розміщують та відправляють будь-який контент браузеру як HTML-розмітку, за певним запитом користувача. Таким чином користувач браузера матиме змогу бачити вміст сторінки сайту, до якої підключається. Веб-сервер зазвичай ще називають програмне забезпечення і сам сервер, на якому це програмне забезпечення працює.

Кожен веб-сервер складається з двох основних компонентів:

- апаратне забезпечення - це певний фізичний комп'ютер, оперативна пам'ять або жорсткі диски які зберігають програмне забезпечення веб-сервера та файли сайту, це можуть бути наприклад - HTML, CSS, JavaScript, зображення, файли шрифтів, тощо. Обмінюється данними з іншими пристроями через інтернет, тільки в тому випадку якщо обидва пристої підключені до інтернету.
- програмне забезпечення - складається з декількох елементів, що надають певну інформацію, як користувачі отримують доступ до розміщених файлів. Основним елементом є HTTP-сервер - це програмне забезпечення, яке використовується для обробки відповідей на запити користувача з браузера.

Головна задачу веб-сервера це отримувати HTTP-запити, обробляти їх і повертати HTTP-відповіді. Найголовніша програма, яка надає змогу працювати з HTTP-запитами та відповідями, - це звичайний браузер, який встановлений в операційній системі (Google Chrome, Opera, Safari, Firefox, Internet Explorer тощо).

Тобто запити до веб-сервера та відповіді, які опрацьовуються протоколом HTTP. HTTP - це спеціальний набір певних правил, які надають змогу спілкуватися браузеру та серверу між собою. Цей протокол дає можливість заходити на веб сторінки, скачувати медіа файли, проглядати інформацію.

Під час набору певної адреси в адресний рядок браузера, перед доменом сайту, насамперед вказується протокол, за яким працює веб-сайт. Це http, або https. Найголовніше, що потрібно розуміти, що після того, як ви ввели адресу сайту в адресний рядок браузера, натиснули клавішу Enter, починає виконуватися запит до веб-сервера. Саме веб-сервер обробляє ці запити. Він може бути встановлений на тому самому комп'ютері, де встановлено браузер.

Також цей веб-сервер може бути встановлений у якомусь іншому місці, іншому комп'ютері в мережі Інтернет.

Просто за допомогою браузера ми звертаємося до цього веб-сервера й отримуємо відповідь. І за отриманою http-відповіддю, браузер відображає вміст веб-сторінки.

Відповідь, яку надсилає веб-сервер, вона містить усю необхідну інформацію для того, щоб браузер зміг відобразити веб-сторінку. У тому вигляді, в якому це задумував розробник веб-сайту.

Основне завдання веб-сервера це отримати запит від HTTP-клієнта, зрозуміти до якого файлу стався запит, обробити цей файл і відправити назад відповідь клієнту.

HTTP-запити ми можемо робити не тільки з браузера. Це можна робити через командний рядок, через певні серверні додатки, тощо.

Також веб-сервер містить у собі налаштування про те, як і які файли потрібно обробляти на веб-сервері. Що потрібно зробити з якоюсь папкою. Загалом, ці всі правила містить веб-сервер. Оскільки веб-сервер це звичайна програма, то цю програму можуть випускати різні розробники. Різні компанії можуть за своїми алгоритмами розробляти веб-сервери, які тим чи іншим чином працюють. Мабуть, найпопулярнішими веб-серверами, які є на сьогоднішній день, є такі веб-сервери як: Apache, IIS, Nginx. [1, 2]

1.2. Інструменти для створення сайтів

В сучасному світі, наприклад для успішного ведення бізнесу важливо мати власний веб сайт. Завдяки розмаїттю доступних інструментів для створення веб-сайтів сучасності, створення власного сайту є простішим, ніж будь-коли. Безліч безкоштовних вебхостингів надають змогу створити сайт за допомогою готових шаблонів, але також є можливість створити сайт з нуля. Після того як сайт готовий та запущений, він може стати дуже потужним маркетинговим інструментом та постійно дозволяти тримати клієнтів і замовників в курсі останніх подій.

Наразі є три способи створення сайту:

- за допомогою конструктора сайтів – це одне з найпростіших та ефективних рішень, що дає змогу побудувати сайт за певним модульним принципом, це коли розробник збирає сайт за допомогою вже готових шаблонів, які надає сам конструктор сайтів. Такий підхід дає змогу створити сайт без серйозних знань про веброзробку та супутніх навичок. Підходить тим, хто має бажання зробити простий сайт наприклад про свою компанію, або для малих бізнесів. Загалом підхід описаний вище для створення сайтів використовується коли не важлива швидкість та якість роботи веб сайту. Також, з використанням конструкторів розробляють прості лендінги для, наприклад, невеликих рекламних кампаній.

- розробка за допомогою систем управління контентом (наприклад WordPress) – це певне рішення яке надає комплекс програмних засобів для управління веб-контентом. Простими словами - це вже готовий базовий каркас і набір певних додаткових інструментів, який дає змогу підтримувати роботу веб додатку, оновлювати контент і взаємодіяти з користувачами. Основна мова програмування для розробки на CMS це PHP.

- самостійна розробка, з використанням сучасних інструментів та фреймворків (Laravel, Django, Spring, React, Angular, Vue) – це найбільш творчий та вільний, але й самий трудомісткий процес. Дозволяє створити повністю унікальний функціонал. Написання сайту потребує впевнені знання мов програмування, розуміння архітектури, бізнес-процесів клієнта і багато іншого. При цьому, розробляючи сайт з повного нуля, клієнт отримає унікальний продукт, який вирішуватиме його поставлені завдання. Самостійна розробка дає змогу розробляти складні проекти, з будь яким дизайном або функціоналом. [3]

1.3. Можливості покращення продуктивності вебзастосунку

Продуктивність веб-застосунку – це здатність ефективно виконувати свої функції з мінімальними витратами ресурсів, такими як час завантаження,

використання пам'яті та швидкість відгуку. Для веб-застосунків, особливо в сучасному світі, де користувачі мають високі вимоги до продуктивності та швидкості роботи, оптимізація продуктивності стає важливим аспектом розробки.

Розробники постійно шукають різні способи оптимізації для покращення продуктивності вебзастосунків і забезпечення задоволення користувачів.

Серед основних можливостей покращення продуктивності вебзастосунків можна виділити такі:

1. SEO оптимізація: зосереджується саме на покращенні видимості та видимості веб-сайту в результатах пошукової видачі шляхом внесення змін на сторінці та підвищення популярності веб-сайту шляхом отримання посилань на нього з метою покращення рейтингу пошуку за ключовими пошуковими термінами. Якщо порівнювати з односторінковими додатками, то вони завантажують тільки одну HTML-сторінку. Щоразу, коли користувач запитує новий HTML-вміст, його динамічно завантажує JavaScript, маніпулюючи напряму з DOM-елементами. У результаті пошукові системи не можуть індексувати контент, який при кожній зміні створюється заново за допомогою JavaScript. Додатки з серверним рендерингом, особливо створені з використанням Next.js, можуть похвалитися і хорошою продуктивністю, і ефективністю SEO-оптимізації, оскільки сторінки відправляються до браузера вже з готовим контентом, це полегшує індексування пошуковими системами і покращує користувацький досвід завдяки швидкому відображенню сторінок. Одним із ефективних методів SEO оптимізації є використання HTML-тегів, таких як <nav>, <link>, <main>, <footer>, <header>, які несуть певний логічний зміст у своїх назвах. Ці теги допомагають структурувати контент, роблячи його більш зрозумілим як для користувачів, так і для пошукових систем, що в свою чергу сприяє покращенню рейтингу сайту у результатах пошуку.

2. Оптимізація швидкості завантаження сторінок: зменшення часу, для завантаження всіх ресурсів вебсайту. Це можна зробити зменшивши розмір файлів, видалення непотрібних скриптів та стилів, та використовувати сучасні технології для прискорення завантаження.

3. Кешування: зберігання часто використовуваних ресурсів на стороні клієнта, для зменшення часу завантаження при наступних запитах. Це можна зробити використавши кешування HTTP-заголовків, таким чином зменшивши навантаження на сервер.

4. Використання SSR: рендеринг певних сторінок на сервері, що дозволяє перед тим як користувач запитує сторінку, сгенерувати її на сервері, та повернути вже в готовому вигляді до клієнта. Також з використанням SSR сторінка повертається з готовим HTML, що саме дозволяє покращити SEO та легше індексувати сторінку пошуковими системами.

5. Розділення коду: поділ великого обсягу коду на менші частини, які будуть завантажуватись тільки при необхідності. За допомогою такого підходу можна зменшити час першочергового завантаження вебзастосунку.

6. Мемоізація: збереження результатів обчислень певних функцій при незмінних вхідних даних, щоб уникнути повторні розрахунки. За допомогою цього можна зменшити навантаження на клієнті, та оптимізувати вебсайт.

7. Використання Lazy Loading: підхід який дозволяє розбити код на певні логічні компоненти, та завантажувати їх тільки в тому разі, коли вони стають видимі для користувача. Цей підхід зменшує початковий час завантаження вебсторінки, та зменшує її вагу.[4, 5]

1.4. Види тестування вебзастосунків

Тестування - це процесом, який включає перевірку функціональності на відповідність певним вимогам, та виявлення помилок на етапі розробки.

Піраміда тестування є концептуальною моделлю, яка ілюструє співвідношення різних рівнів тестування для забезпечення якісного програмного продукту (рис 1.1). Ця модель допомагає розробникам зосередити свою увагу на створенні надійного коду, тестуючи його на різних рівнях: від найнижчого рівня модулів до найвищого рівня інтеграційного та енд-то-енд тестування. [6]

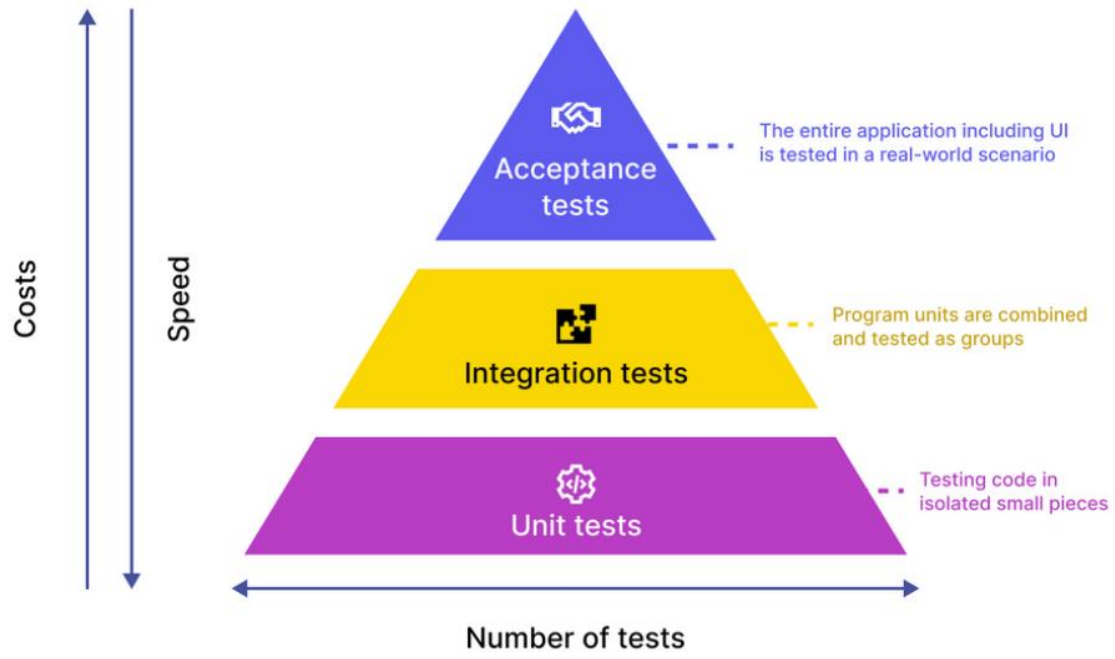

Рис. 1.1. Піраміда тестування

1.4.1. Модульне тестування

Одним з головних методів тестування є модульне тестування, яке займає важливе місце у піраміді тестування програмного забезпечення.

Модульне тестування — це тестування певних окремих компонентів програмного додатку.

Разом ці одиниці коду утворюють повну програму, і якщо вони погано працюють окремо, вони точно не працюватимуть добре разом. Модульне тестування гарантує, що кожен компонент програмного забезпечення працює так як потрібно сам по собі, перш ніж інтегрувати його в більшу систему.

Модульне тестування дозволяє виявити помилки під час розробки, що гарантує уникнення створення помилок при інтеграції нової функціональності в продукт.

За допомогою тестів ми зможемо знизити ризик виникнення помилок під час вдосконалення програмного коду та впровадження нової функціональності, що насамперед дозволить з легкістю вносити зміни, не порушуючи старий

функціонал. У цьому випадку одним із рішень є перевірити, як частини програми працюють разом як єдине ціле.

1.4.2. Інтеграційне тестування

Інтеграційне тестування – це певний підхід, який надає можливість перевірити як частини програми працюють як єдине ціле. Воно враховує побічні ефекти з самого початку.

Метою інтеграційного тестування є виявлення проблем на рівні взаємодії між модулями, які можуть виникнути після їх об'єднання. Воно допомагає знайти проблеми, які не є очевидними, досліджуючи реалізацію всієї програми або окремого блоку, що допомагає виявити дефекти у взаємодії кількох частин програми.

Переваги інтеграційного тестування:

- виявлення проблем на ранніх етапах: Інтеграційне тестування дозволяє виявляти проблеми на ранніх етапах об'єднання модулів.
- Підвищення якості системи: Забезпечення коректної взаємодії між модулями сприяє загальному підвищенню якості продукту та його надійності.
- Підготовка до системного тестування: Інтеграційне тестування є важливим етапом перед системним тестуванням, оскільки воно забезпечує базову функціональність та інтеграцію всіх частин програми.

1.4.3. Тести користувацького інтерфейсу

Тести користувацького інтерфейсу (UI-тести) спрямовані на перевірку поведінки програми з точки зору кінцевого користувача. Вони допомагають забезпечити відповідність програмного забезпечення вимогам та очікуванням користувачів, гарантуючи, що інтерфейс програми є зручним, функціональним та інтуїтивно зрозумілим. Розглянемо детальніше ключові аспекти UI-тестів.

Основна мета тестування інтерфейсу користувача - побачити, як кінцевий користувач взаємодіє з додатком. Вони охоплюють наступні аспекти:

- Функція: Перевірка того, чи всі елементи інтерфейсу працюють відповідно до вимог. Наприклад, чи правильно кнопка реагує на натискання і т.д.
- Зручність використання: Оцінка зручності користувальницького інтерфейсу. Проблеми з навігацією, виявлення неоднозначних або прихованих функцій, які можуть ускладнити використання програми.
- Відповідність дизайну: Перевірка, що інтерфейс відповідає макету та дизайну, затвердженим на етапі розробки. Це включає перевірку шрифту, кольору, розташування елементів та загального вигляду інтерфейсу.

1.5. Поняття Блокчейну та децентралізації

Технологія блокчейн — це механізм, який організовує безпечний обмін інформацією в рамках певних мереж. Дані зберігаються в блоках, в певних базах даних Blockchain, які з'єднані один з одним та криптографічно зашифровані. Кожен блок містить набір обмінів або інших даних і з'єднаний з попереднім блоком, утворюючи ланцюг. Цей ланцюг не можна видалити або змінити без згоди мережі, тому дані є послідовними з часом. Це надає можливість використовувати технологію блокчейн для незмінного, постійного відстеження та запису замовлень, платежів, та інших транзакцій. Система має захищені механізми баз даних для запобігання несанкціонованому проникненню і забезпечення безпеки.

Блокчейн найчастіше використовується для проведення та реєстрації певних транзакцій із залученням криптовалют, таких наприклад як Ethereum та Bitcoin, але технологія блокчейну може бути корисною також у багатьох інших контекстах. Кожен користувач блокчейну має власні криптографічні ключі — один відкритий і один закритий. Коли відбувається транзакція, одна сторона надсилає актив іншій стороні, використовуючи відкритий ключ останньої як свого роду адресу. Потім закритий ключ одержувача використовується для підтвердження своєї особи, щоб він міг «розблокувати» та прийняти актив.

Вузли в одноранговій мережі перевіряють дійсність цієї транзакції відповідно до протоколу, погодженого користувачами мережі. Після перевірки

всіх транзакцій у блоці та досягнення консенсусу щодо порядку, в якому вони відбувалися, блок закривається та пов’язується з попереднім блоком у ланцюжку, а копія блокчейну кожного вузла оновлюється.

Коли люди здійснюють покупки за допомогою криптовалюти, їхні транзакції полегшуються саме через блокчейн. Покупець використовує відкритий ключ продавця, щоб надіслати йому криптовалюту, а закритий ключ продавця відкриває кошти. Ця транзакція перевіряється вузлами в мережі, а потім постійно вбудовується в блокчейн.

Децентралізація є керівним принципом організації системи, в якій адміністрація та контроль передаються між членами організації замість централізованих спеціалістів чи окремих підрозділів. Поєднання блокчейну та децентралізації робить можливим створення відкритих, зрозумілих, безпечних фреймворків, які відрізняються від централізованих моделей.

Блокчейн гарантує незмінну якість і непідкупність інформації, а децентралізація усуває централізований контроль, надаючи членам організації більше можливостей і незалежності. Такі інфраструктури широко використовуються в криптовалютних системах, децентралізованому бюджетному адмініструванні, електронному голосуванні, ланцюгах постачання та багатьох інших регіонах. Блокчейн і децентралізація забезпечують більш високу стійкість до маніпуляцій, покращуючи довіру до системи.

У криптовалютних мережах, наприклад, децентралізовані моделі дозволяють кожному учаснику брати участь у підтвердженні транзакцій, що зменшує ризики шахрайства та підвищує безпеку мережі. У системах електронного голосування децентралізація забезпечує анонімність і захист від фальсифікацій, а в ланцюгах постачання - прозорість та точне відстеження товарів.

Поєднання цих технологій відкриває нові можливості для створення надійних і ефективних систем у різних галузях, забезпечуючи вищий рівень безпеки та довіри. [7]

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі кваліфікаційної роботи було детально розглянуто основні поняття, інструменти та методи, що використовуються при створенні сучасних вебзастосунків, а також введено в основи блокчейн-технологій. Підсумовуючи ключові моменти, варто зазначити важливість кожного з них для розробників та користувачів вебзастосунків.

Вебзастосунки є інтерактивними програмами, що працюють на веб-серверах і взаємодіють з користувачами через веб-браузери. Вони містять як статичні, так і динамічні елементи, що забезпечують різноманітний функціонал для користувачів.

Для забезпечення високої продуктивності вебзастосунків необхідно оптимізувати фронтенд складові. Це включає мінімізацію завантажуваних ресурсів, використання кешування, оптимізацію запитів до баз даних та застосування асинхронних методів обробки даних. Також важливо використовувати CDN (Content Delivery Network) для зменшення затримок при завантаженні контенту.

Особливу увагу при створенні застосунків було приділено тестуванню, яке забезпечує якість та надійність. Модульне тестування спрямоване на перевірку окремих модулів або компонентів коду, інтеграційне тестування перевіряє взаємодію між різними компонентами системи, а тести користувацького інтерфейсу оцінюють зручність та правильність роботи інтерфейсу з точки зору кінцевого користувача.

В рамках першого розділу було також розглянуто блокчейн технології, що включає основні поняття, принципи роботи та можливості інтеграції з вебзастосунками.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ

2.1. Мова програмування Typescript

TypeScript - це підмножина JavaScript, яка має необов'язкову типізацію і компілюється у звичайний JavaScript. Насамперед ця мова додає додатковий синтаксис поверх основного JavaScript. Надає більш жорстку типізацію, додаткові структури такі як анотації типів, інтерфейси, дженерики, призначені для забезпечення узгодженості програми.

Свобода динамічної типізації часто призводить до помилок, які не тільки знижують ефективність роботи, але й можуть зупинити розробку через виникнення помилок на виявлення яких потрібно багато часу. Таким чином, відсутність у JavaScript таких речей, як типізація та перевірка помилок під час написання коду, робить його не самим найкращим вибором для написання серверного.

Причиною вибору TypeScript для розробки веб-застосунка було те, що TypeScript надає можливість визначення типів для змінних, параметрів функцій, об'єктів тощо, що покращує стабільність та надійність коду. Це також допоможе уникнути небажаних помилок, та надасть можливість індексувати помилки ще на етапі розробки веб-застосунку. [8]

TypeScript підтримує стандарт ECMAScript та може бути скомпільований в будь-яку версію JavaScript, таку як ES5, ES6, ES7 тощо. Це забезпечує гнучкість і функціональність для використання новітніх функцій і синтаксису JavaScript у проектах розробки.

Багато популярних інтегрованих середовищ розробки, таких як Visual Studio Code, підтримують TypeScript, і надають розширені можливості для розробки, такі як автозаповнення, перевірка типів та налагодження.

TypeScript допомагає покращити організацію та підтримку великих проектів, розділивши код на модулі, визначивши типи інтерфейсу та компонентів, а також автоматично генеруючи документацію. Саме через це він був обраний в якості провідної технології розробки даного веб-застосунку.

TypeScript автоматично визначає тип даних змінної на основі її початкового значення,тому не потрібно явно вказувати тип кожної змінної, що значно економить час. Анотації до примітивних типів значень включають числа, логічні значення та рядки.

Продуктивність є важливим аспектом мови програмування, оскільки вона безпосередньо впливає на ефективність та швидкодію програми. При порівнянні TypeScript та JavaScript часто виникають проблеми з продуктивністю через додаткові етапи компіляції TypeScript. Однак варто зазначити, що TypeScript компілюється в JavaScript і забезпечує однакову продуктивність на обох мовах. Хоча це може призвести до незначних витрат на етапі компіляції, найновіші компілятори TypeScript добре оптимізований, тому вплив на продуктивність під час виконання мінімальний.

Використання об'єктно-орієнтованої мови може зробити код надійним, зрозумілим та простим в підтримці. Крім того, TypeScript робить ООП більш елегантним завдяки простим у використанні класам, інтерфейсам та модулям порівняно з інтерфейсами JavaScript.

TypeScript надає ефективні функції, які включають класи, інтерфейси та модулі. Окрім розробки на стороні сервера, є можливість писати об'єктно-орієнтований код на стороні клієнта. Ця особливість робить його більш яскравим, ніж JavaScript. Функції ООП також роблять код на TypeScript більш зручним в тестуванні та більш організованим.

Читабельність коду має вирішальне значення для співпраці, підтримки коду та налагодження. Код на TypeScript легше читати порівняно з JavaScript, головним чином через його статичні можливості набору тексту. Анотації типів надають цінну документацію до бази коду та пояснюють типи змінних та підписи функцій. Крім того, підтримка інтерфейсу TypeScript сприяє чіткості та стисненню коду, покращуючи читабельність та видимість, особливо для великих проектів. [9]

Основні особливості TypeScript включають:

- Статична типізація: TypeScript дозволяє визначати типи змінних, типи параметрів функцій та значень які повертаються, що насамперед допомагає уникнути безліч помилок.
- Інтерфейси та типи: Можливість визначення власних типів даних та інтерфейсів для чіткішої структури коду. Загалом інтерфейси використовуються коли потрібно задати тип значенням які повертаються від серверу, а типи використовуються для того щоб типізувати значення які повертаються з функції.
- Декоратори: Синтаксичний цукор для додавання метаданих до класів та методів, що полегшує використання певних патернів проектування.
- Дженерики - це одна з найпотужніших функцій TypeScript, яка дозволяє створювати компоненти, що працюють з різними типами даних, зберігаючи при цьому безпеку за замовчуванням. Універсальні засоби дозволяють розробникам створювати функції, класи та інтерфейси, які працюють з будь-якими типами даних, але в той же час забезпечують типову структуру безпеки.
- Підтримка ES6: TypeScript підтримує всі можливості ES6 (ECMAScript 2015) і навіть пізніших стандартів. [10]
- TypeScript поєднує можливасті статичної типізації з останніми сучасними можливостями JavaScript, що робить його ідеальним вибором для великих і складних проектів.

2.2. Основні особливості фреймворку Next js

Next.js - це фреймворк, який надає змогу розробникам легко створювати додатки з використанням серверного рендерингу. Завдяки SSR, SSG, ISR вміст веб-сайту рендериться на сервері до того, як він потрапляє до браузера для відображення користувачу, що призводить до швидшого завантаження, та кращого індексування пошуковими системами.

Next.js також пропонує можливості попереднього рендерингу, що є ще однією важливою перевагою для SEO. Це означає, що HTML-код для кожної сторінки генерується заздалегідь, а не на стороні клієнта. В результаті, пошукові роботи можуть легко зрозуміти та проіндексувати вміст вебзастосунку, що призводить до кращого ранжування в пошукових системах.

Фреймворк Next.js став одним з найпопулярніших інструментів для розробки сучасних веб-застосунків. Його надійність, гнучкість та здатність до швидкої реалізації ідей зробили його привабливим вибором для розробників у всьому світі. Велика спільнота розробників активно співпрацює над розвитком та підтримкою цього фреймворку, що сприяє постійному вдосконаленню та забезпеченню актуальності його функціональності.

Розглянемо головні можливості даного фреймворку, за допомогою яких ми зможемо створити продуктивний, швидкий та інтуїтивно зрозумілий веб-застосунок. [11]

2.2.1. CSR (Client Side Rendering)

Клієнтський рендеринг — це техніка веб-розробки, при якій рендеринг веб-вмісту виконується в основному на стороні клієнта, а не на сервері(рис. 2.1.). Під час клієнтського рендерингу веб-сервер зазвичай надсилає необроблені дані або мінімальну оболонку HTML до клієнта. Потім JavaScript, запущений у браузері клієнта, отримує додаткові дані та динамічно відображає вже готову веб-сторінку. Таким чином використовуючи CSR призведе до того, що користувач буде чекати

на повну генерацію контенту, і тільки після цього він зможе взаємодіяти з вебзастосунком.

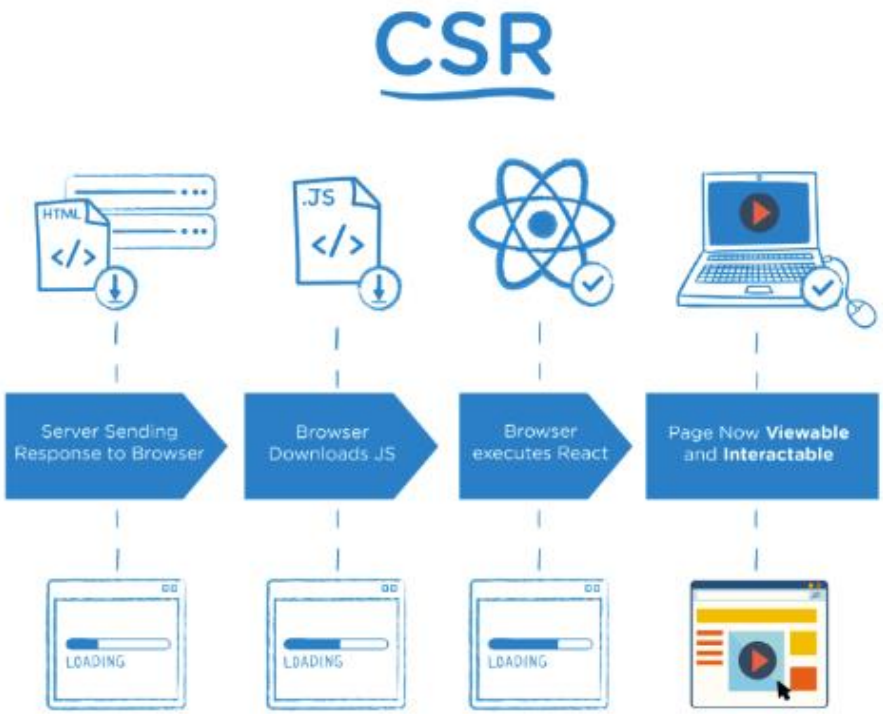

Рис. 2.1. Метод роботи CSR(Client Side Rendering)

2.2.2. SSR (Server Side Rendering)

Серверний рендеринг - це технологія, за якої веб-сервер генерує початковий HTML-вміст сторінки і надсилає його до браузеру. Це означає, що коли користувач заходить на сайт, він отримує повністю попередньо згенерований на сервері HTML-документ з готовим вмістом, який готовий до відображення.

За допомогою SSR є можливість значно скоротити першочергове завантаження веб-сторінки, оскільки користувач таким чином отримує вміст швидше, адже він генерується безпосередньо на сервері, і вже в готовому вигляді повертається користувачу (рис. 2.2.). Таким чином користувачі бачать контент швидше, що призводить до кращого сприйняття продуктивності та зручності для користувачів. Це корисно користувачам які мають повільне мережеве з'єднання або менш потужні пристрої.

Також за допомогою SSR є можливість позитивно вплинути на пошукову оптимізацію сайту. Пошукові системи можуть легше сканувати та індексувати вміст сторінок, оскільки HTML-код доступний негайно.

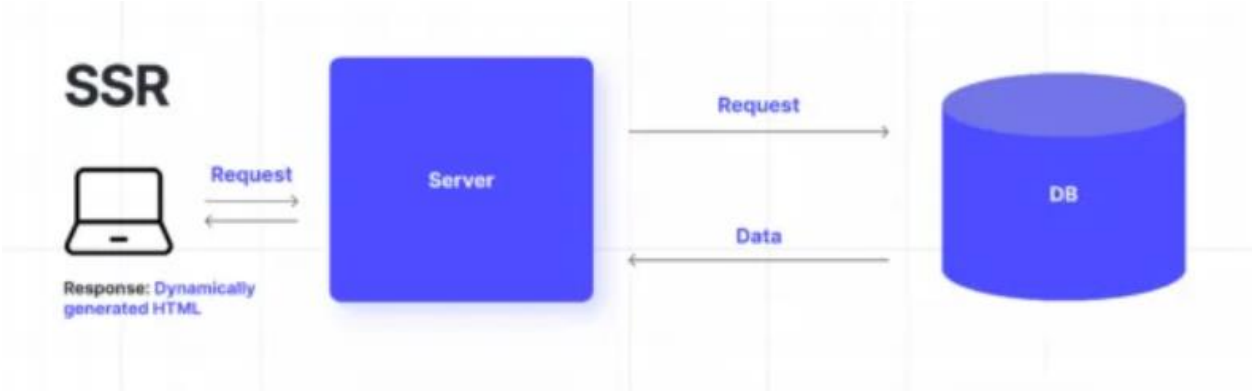

Рис. 2.2. Метод роботи SSR(Server Side Rendering)

2.2.3. SSG (Server Side Generating)

Статична генерація сайту рендерить сторінку під час створення. Це означає, що для SSG процес створення веб-сторінки відбувається до того, як сайт буде запущено і до нього отримають доступ користувачі.

HTML-код і вміст кожної веб-сторінки попередньо візуалізуються або попередньо генеруються, тому вони готові і статичні (незмінні), коли веб-сайт розгортається на веб-сервері (рис. 2.3.).

Рис. 2.3. Метод роботи SSG(Server-side Generating)

Коли користувач відвідує сторінку на сайті, попередньо відрендерений або попередньо згенерований статичний HTML негайно доставляється в його браузер,

без додаткової обробки на момент запиту користувача. Це дуже зручно, так як користувачу не потрібно чекати повної генерації веб сторінки на кожний запит.

Оскільки сторінки вже згенеровані і не вимагають додаткової обробки на сервері при кожному запиті, сервер може обслуговувати більше користувачів одночасно без втрати продуктивності.

2.2.4. ISR (Incremental Static Regeneration)

Інкрементна статична регенерація – це такий підхід, який надає можливість комбінувати SSR, і SSG. За допомогою ISR Next.js попередньо генерує статичну HTML-розмітку під час збірки застосунку та може повторно генерувати розмітку на стороні сервера під час виконання за потреби (рис. 2.4.).

За допомогою цієї функції є можливість створювати швидко оновлювані та оптимізовані для пошукових систем веб застосунки, зберігаючи при цьому актуальність даних без необхідності заново перегенеровувати вебзастосунок. Таким чином можна генерувати лише окремі сторінки або частини веб застосунку, які були змінені або термін дії їх минув, що тим часом може зменшити навантаження на сервер.

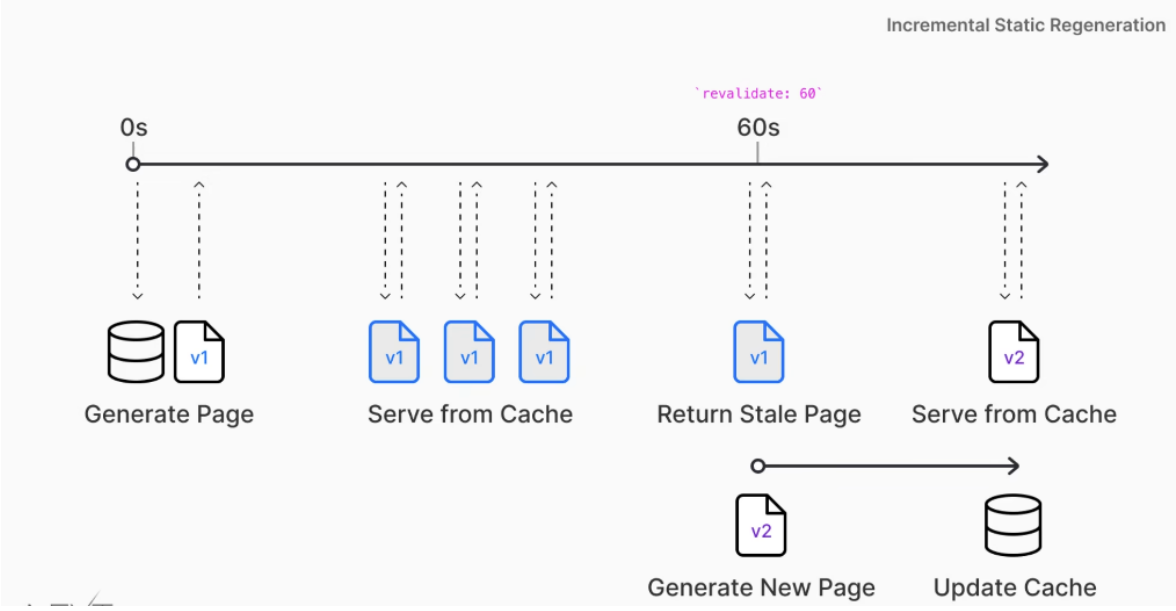

Рис. 2.4. Метод роботи ISR (Incremental Static Regeneration)

ISR може регенерувати сторінки у фоновому режимі. Коли користувач відвідує сторінку, якій потрібне оновлення, Next.js повертає закешовану стару версію, а у фоновому режимі генерує нову версію сторінки, яка стане доступною для наступного користувача. ISR надає можливість контролювати версії сторінок, як наслідок можна забезпечити більш передбачувану поведінку вебзастосунку. [12]

2.2.5. Маршрутизація

Маршрутизація це одна з головних особливостей фреймворку Next.js, яка дозволяє ефективно керувати переходами між різними сторінками та компонентами веб-застосунку. У Next.js маршрутизація здійснюється за допомогою файлу pages, де кожен файл відповідає конкретній сторінці або маршруту в застосунку(рис. 2.5.).

Даний веб-застосунок, має багато різних сторінок та компонентів, таких як профіль користувача, економічний календар, тощо. За допомогою маршрутизації користувачі зможуть легко переміщатися між цими розділами.

Рис. 2.5. Визначення маршрутів в Next.js

2.3. Переваги та недоліки Next js

Next.js є популярним фреймворком React, який надає потужні інструменти та функції для створення сучасних веб-додатків.

Переваги Next.js:

- Генерація контенту на стороні сервера (SSR): Next.js підтримує SSR із коробки, що дозволяє розробникам генерувати сторінки на сервері перед тим, як надсилати їх до клієнта. Це забезпечує гарну індексацію пошуковими системами, та робить застосунок більш оптимізованим.
- Автоматична маршрутизація: Next.js забезпечує автоматичну маршрутизацію за допомогою файлів та дозволяє розділяти код так, щоб для кожної сторінки завантажувати лише необхідні частини.
- Статична генерація сайту (SSG): Next.js пропонує SSG, який генерує статичні файли HTML під час збірки. Це особливо корисно для вебзастосунків, які змінюються часто, забезпечуючи кращу продуктивність і зменшене навантаження на сервер.
- Інерементальна статична генерація(ISR): Надає можливість статично генерувати контент під час завантаження, що дозволяє оновлювати певну частину контенту без необхідності перегенеровувати весь сайт.
- Турбопак: Заміняє Webpack для більш швидшого збирання, надає більш швидкий запуск застосунку та швидкість оновлення. Це сприяє підвищенню продуктивності розробників і швидшим ітераційним циклам, що важливо для великих проектів.
- Middleware: це інструмент, що дозволяє контролювати процес обробки запитів та відповідей на сервері. Вона виконується під час кожного запиту і може бути використана для виконання різних операцій, таких як аутентифікація, авторизація або обробка даних перед їх передачею на клієнтську сторону. Ця функція є важливою для контролю над взаємодією з користувачем та ефективної реалізації серверної логіки.

➢ API маршрути: Next.js надає можливість створювати бекенд логіку саме у тому ж самому застосунку, що спрощує архітектуру і зменшує потребу у створенні окремого серверу.

➢ Підтримка Typescript: Next.js підтримує Typescript з під коробки, що дозволяє насамперед виявляти помилки ще під час самої розробки, та покращувати якість коду.

Next.js це потужний фреймворк для розробки додатків React, і хоча він має багато переваг, він також має певні недоліки, які можуть вплинути на вибір інструментів розробки.

Недоліки Next.js:

➢ Складність для простих додатків: Для невеликих проектів наступне функції Next.js можуть бути надмірним. Платформа надає безліч функцій для рендеринга на стороні сервера, генерації статичних сторінок, маршрутизації та інших функцій, які не потрібні в простих додатках. В такому випадку, використовувати Next.js може ускладнити розробку.

➢ Надмірний склад: Next.js надає безліч можливостей для конфігурації, що може бути як перевагою, так і недоліком. Ці можливості можуть бути складними для новачків або розробників, які не мають досвіду роботи з фреймворками з великими можливостями конфігурації. Певні знання і досвід необхідні для правильного налаштування та використання всіх можливостей, що може збільшити час на початкову конфігурацію проекту.

➢ Обмежена гнучкість: Next.js має власні угоди та маршрутизацію на основі файлової системи, що може обмежити гнучкість для певних розширених випадків використання або існуючих структур проекту.[13]

2.4. Управління станом за допомогою Zustand

Zustand - це бібліотека управління станами для React, яка надає простий та гнучкий спосіб керувати станами додатку. Вона побудована на основі React Context API і використовує можливості сучасних функцій JavaScript, таких як

проксі та генератори. Він компактний, швидкий і масштабований. Це досить проста система з невеликим шаблонним кодом. Використання Zustand потребує набагато менше коду, ніж Redux та подібні бібліотеки для керування станами в React. Він не залежить від постачальника. Як наслідок, вам не потрібно кодувати так багато логіки React застосунках, що може зменшити повторюваність коду. Він працює на основі спрощених принципів потоку і в основному використовує хуки.

Причиною вибору Zustand для розробки веб-застосунку соціальної екосистеми для ентузіастів криптовалют, було те що ми отримуємо головне простий, ефективний та потужний інструмент для керування станом, який допомагає зробити додаток більш швидким, продуктивним та легким у розробці.

Він дозволяє легко створювати та оновлювати складні структури станів, обробляти асинхронні дії та обмінюватися станами між компонентами. (рис. 2.6.)

src > store > TS useMenuStore.ts > useMenuStore > create() callback

```ts
import { create } from 'zustand';

interface State {
  isMenuOpen: boolean;
  isSearchMenuOpen: boolean;
  selectedIconIndex: number;
}

interface Actions {
  toggleMenu: () => void;
  openSearchMenu: () => void;
  closeSearchMenu: () => void;
  setSelectedIcon: (index: number) => void;
}

const INITIAL_STATE: State = {
  isMenuOpen: true,
  isSearchMenuOpen: false,
  selectedIconIndex: 0,
};

export const useMenuStore = create<Actions & State>((set) => ({
  isMenuOpen: INITIAL_STATE.isMenuOpen,
  isSearchMenuOpen: INITIAL_STATE.isSearchMenuOpen,
  selectedIconIndex: INITIAL_STATE.selectedIconIndex,
  toggleMenu: () => set((state: State) => ({ isMenuOpen: !state.isMenuOpen })),
  openSearchMenu: () =>
    set(() => ({
      isSearchMenuOpen: true,
    })),
  closeSearchMenu: () =>
    set((state: State) => ({
      isSearchMenuOpen: false,
      selectedIconIndex: state.isSearchMenuOpen ? 0 : state.selectedIconIndex,
    })),
  setSelectedIcon: (idx: number) => set(() => ({ selectedIconIndex: idx })),
}));

```

Рис. 2.6. Приклад використання бібліотеки zustand у проекті

Zustand в першу чергу призначений для стану модуля, тобто визначаємо сховище в модулі і експортуємо його. Наприклад, деякі програми використовують react-запит для отримання стану сервера, і таким програмам потрібен лише невеликий глобальний стан. Zustand підходить як для маленьких так і для великих проектів.

У цьому фрагменті коду ми використовуємо бібліотеку zustand для створення стору для управління станом автентифікації в нашому веб-застосунку.

2.4.1. Порівняння Zustand і Redux:

Zustand і Redux – це дві популярні бібліотеки для керування станом в застосунках. Ці дві бібліотеки мають спільне призначення, але мають різні підходи до вирішення задач.

Ключова відмінність між Zustand і Redux це підхід до налаштування магазину для збереження стану додатка. Основна мета Zustand це надати просте налаштування для управління станом. Він доволі простий для розуміння та використання.

Zustand може бути більш продуктивним, ніж Redux, у деяких випадках. Це зумовлено тим, що Zustand використовує Context API для надання стану компонентам, що усуває потребу в підписках і зменшує кількість повторних рендерингів. Redux, натомість, використовує підписки для оновлення компонентів при зміні стану, що може призводити до непотрібних повторних рендерингів. Зрештою, вибір між Redux і Zustand залежить від певних вимог та потреб програми.

Складність Redux проявляється в дотриманні певних жорстких принципів та вказівок, що забезпечує масштабованість застосунку та зручність використання у великих програмах. Redux буде кращим вибором коли потрібно працювати на великих проектах.

Вибір відповідної бібліотеки для керування станом може значно вплинути на процес розробки, масштабованість та легкість обслуговування вашого застосунку.

Redux є складнішою бібліотекою порівняно з Zustand. Завдяки архітектурі односпрямованого потоку даних він може бути важким для розуміння початківцями, оскільки потребує вивчення багатьох понять і термінології. Zustand, навпаки, набагато простіший і зрозуміліший. Він використовує функціональний підхід до управління станом, який легше освоїти.

2.4.2. Переваги використання Zustand:

- Простота налаштування: Zustand відомий своєю легкістю налаштування, оскільки не потребує інтеграції програми з компонентами постачальника або використання будь-яких складних шаблонних схем як наприклад за допомогою Redux.
- Інтеграція з DevTools: Zustand забезпечує можливість відстеження та доступу до стану додатка через інструменти розробника, а також додаткове підключення Zustand DevTools. Ці інструменти значно спрощують налагодження, перевірку стану та розуміння потоку даних у додатку. Для цього необхідно підключити відповідне проміжне програмне забезпечення для журналювання дій.
- Оснований на хуках: Zustand оснований на React-хуках, що саме надає можливість більш простого та передбачуваного управління станом. Цей підхід відповідає сучасним практикам розробки в React, та Next і забезпечує зручний зрозумілий інтерфейс.
- Незмінність (іммутабельність): Zustand підтримує принцип незмінності, що є гарною практикою для розробки React або Next застосунків. Під час оновлення стану, Zustand створює новий об'єкт стану, що гарантує повторне змінення компонентів лише тоді, коли це буде необхідно. Такий підхід гарно впливає на продуктивність вебзастосунку.

- Підтримка TypeScript: Zustand підтримує використання TypeScript, що надає можливість створювати більш якісні вебзастосунки та спрощує підтримку безпеки типів.
- Використання сховища: Zustand надає можливость інтегрувати застосунок з локальним або сеансовим сховищем. Ця функція є безцінною для збереження стану під час перезавантаження сторінки, забезпечуючи послідовну взаємодію з користувачем.
- Підтримка SSR (Server-Side Rendering): Сумісний із такими фреймворками такими як Next.js. Незалежно від того, це клієнтський, серверний або статично згенерований проект, Zustand буде працювати будь де.
- Ефективний повторний рендеринг: Zustand оптимізує продуктивність вебзастосунку, забезпечуючи оновлення компонентів лише за необхідності. Це сприяє швидкому реагуванню інтерфейсу та покращує користувацький досвід завдяки більш плавній та швидкій роботі застосунку.[14]

2.5. Редактор коду Visual Studio Code

Visual Studio Code (також відомий як VS Code) - це широко використовуваний редактор коду з відкритим вихідним кодом. Він був розроблений корпорацією Microsoft (рис. 2.7.). VS Code відомий своєю універсальністю, швидкістю, простотою використання, зручним інтерфейсом, розумним завершенням коду, налагодженням, інтеграцією Git, а також довгим переліком фу5нкцій, такі як сотні тисяч розширень та багато іншого.

Процес розробки буде відбуватися саме в ньому, адже VS Code дозволяє розробникам отримати доступ до потужного середовища розробки без необхідності оплати за ліцензію. Він забезпечує розширену підтримку для JavaScript та TypeScript, що робить зручним роботу з кодом на базі фреймворку Next.js.

Його функціональність дозволяє користувачам змінювати редактор відповідно до їх потреб. Це означає, що користувачі можуть завантажувати бібліотеку з Інтернету та інтегрувати її з кодом відповідно до своїх вимог.

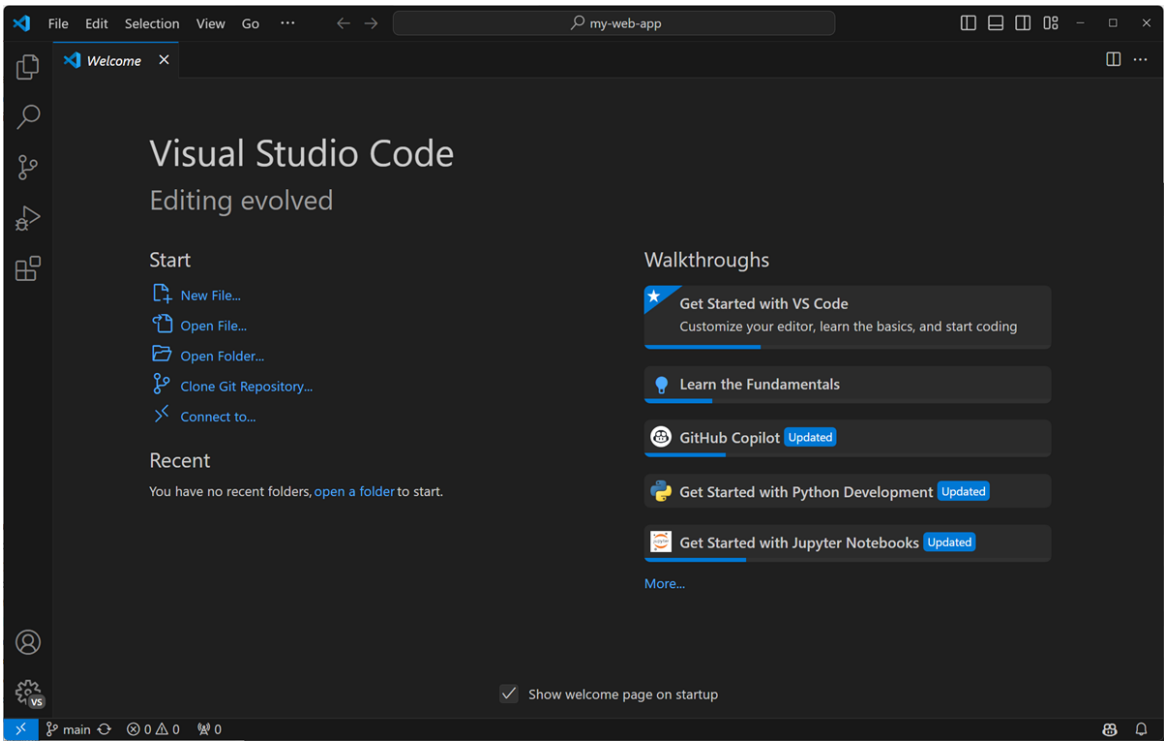

Рис. 2.7. Головне вікно програми Visual Studio Code

VS Code також має ряд розширень і додатків, які допомагають розробникам писати код, включаючи розширення для роботи з JavaScript, React, Next.js та іншими технологіями.

Але головною причиною вибору VS Code як редактору коду для розробки даного вебзастосунку полягає в тому, що він має вбудовану підтримку Git, що дозволяє зручно керувати версіями коду та співпрацювати з іншими розробниками.

Таким чином, вибір Visual Studio Code для розробки вебзастосунку є безкоштовним, з підтримкою JavaScript і TypeScript, розширюваністю, простотою використання та інтеграцією з іншими інструментами розробки.[15]

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі кваліфікаційної роботи було розглянуто технології, що використовуються для розробки сучасних веб-додатків, та основна увага приділялася мові програмування TypeScript та фреймворку Next.

Детально було розглянуто популярний фреймворк, Next.js який забезпечує такі функції, як серверне завантаження (SSR), статичне генерування (SSG) та інкрементальна статична регенерація (ISR). Фреймворк також підтримує клієнтське завантаження (CSR), що дозволяє створювати швидко завантажені та оптимізовані для пошукових систем веб-застосунки.

Огляд переваг Next.js у поєднанні з TypeScript, є важливим етапом, що надає можливість створювати потужні, оптимізовані та продуктивні вебзастосунки. TypeScript додає підтримку статичної типізації та сучасні можливості, що робить розробку більш ефективною.

Після порівняння Zustand та Redux для вибору менеджера стану програми було обрано Zustand. Використання Zustand оптимізує продуктивність і ефективність повторної візуалізації компонентів, що робить його відмінним вибором для управління станом в цьому вебзастосунку.

Підсумувавши все варто відзначити що об'єднання Next.js і TypeScript - це сучасний та потужний підхід. Ці технології забезпечують високий рівень безпеки додатків за рахунок статичної типізації і надають широкі можливості для створення і генерації сторінок. Це робить Next.js та Typescript ідеальним вибором для створення даного вебзастосунку.

РОЗДІЛ 3
РОЗРОБКА ВЕБЗАСТОСУНКУ

3.1. Головна структура вебзастосунку

Структура вебзастосунку в основному подібна у більшості проектів. Її можна змінювати в залежності від вимог конкретного проекту. Для вебзастосунку соціальної екосистеми для ентузіастів криптовалют, створеного за допомогою фреймворку Next.js, основна файлова структура виглядає наступним чином (рис 3.1.):

pages/ - містить файли проекту, де кожен файл відповідає певному маршруту. Кожен файл в цій папці, це окремі сторонки застосунку. Наприклад, папка profile з файлом index.tsx буде відповідати окремій сторінці профілю користувача.

components/ - папка, яка містить всі компоненти застосунку. Наприклад, Footer.tsx, Post.tsx, LeftSideBar.tsx - кожен файл є окремим компонентом, який може багаторазово використовуватися в різних частинах проекту. Це зменшує дублювання коду та покращує його читабельність.

hooks/ - у цій папці зберігаються користувацькі хуки, які дозволяють повторно використовувати логіку стану у різних компонентах. Наприклад, у даному проекті створено хук use-outside-click.tsx, який надає можливість відслідковувати натискання миші поза межами певних компонентів, що покращує взаємодію користувача з інтерфейсом.

public/ - Ця папка містить статичні ресурси, такі як зображення та шрифти, які будуть доступні безпосередньо з кореня URL. Загалом в цій папці зберігаються картинки та svg іконки.

store/ - У цій папці розміщуються файли, що стосуються управління станом.

styles/ - У цій папці зберігаються стилі, які використовуються в застосунку. Вони можуть бути у форматі CSS, Sass або будь-якому іншому підтримуваному форматі.

utils/ - У цій папці зберігаються допоміжні функції та утиліти, які можуть використовуватися в різних частинах застосунку.

constants/ - папка де зберігаються всі константи.

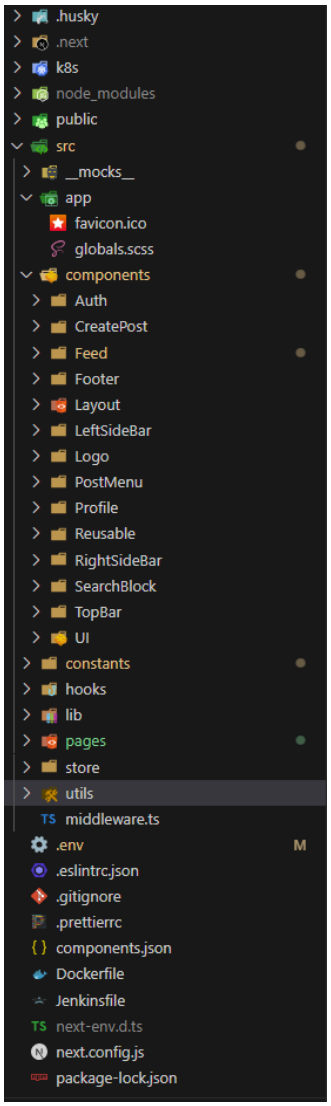

Рис. 3.1. Файлова структура проекту

3.2. Реалізація ключової функціональності вебзастосунку

Реалізація ключової функціональності вебзастосунку включає в себе розробку основних компонентів і модулів, які забезпечують його роботу та взаємодію з користувачем.

3.2.1. Реалізація головної сторінки вебзастосунку

Головна сторінка має такий вигляд (рис 3.2.):

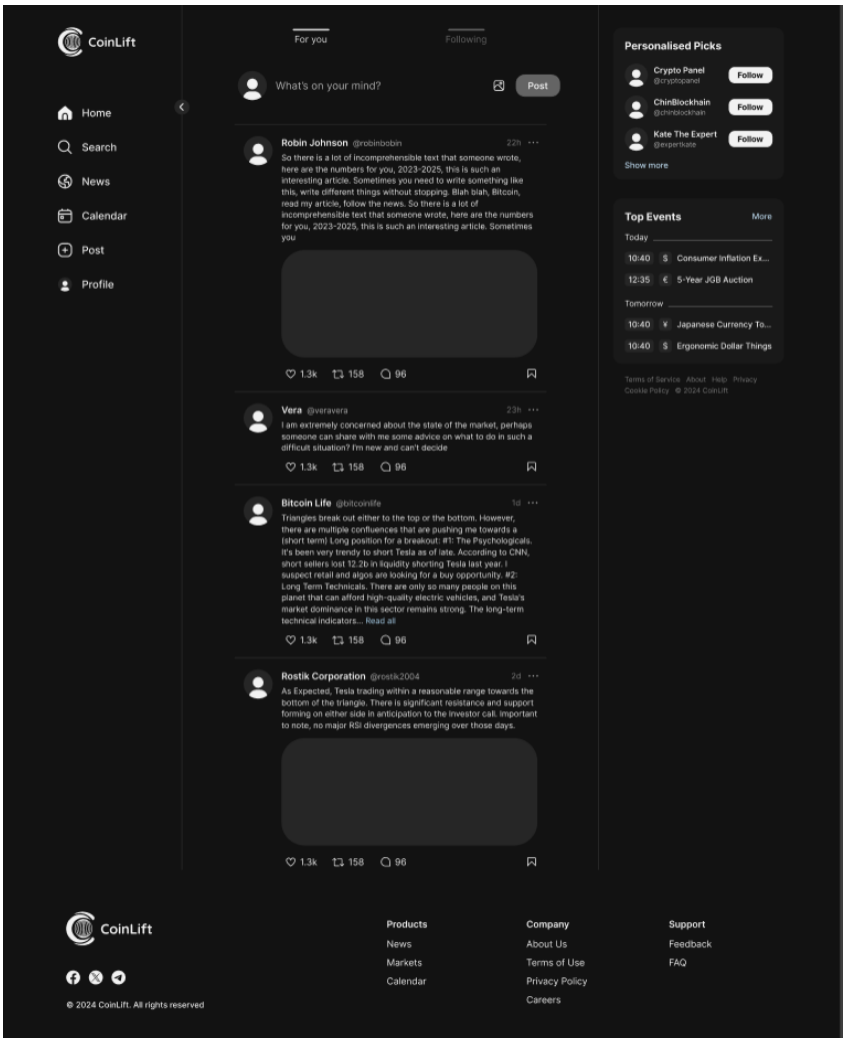

Рис. 3.2. Зовнішній вигляд головної сторінки

Реалізація головної сторінки вебзастосунку складається з кількох основних етапів. Першим кроком є створення файлу index.tsx в папці pages, який буде розташований у кореневій директорії та відповідатиме за головну сторінку сайту (рис 3.3.).

Цей файл стане основною точкою входу для користувачів, тому важливо належним чином структурувати та організувати його вміст, забезпечуючи зручну навігацію та привабливий інтерфейс для користувачів.

```
function PostPage({ following, recommendations, users }: IPostsProps) {
  const selectedAction = useTopBar((state) => state.selectedAction);
  const { setUsers } = useFollowsStore();
  const router = useRouter();

  useEffect(() => {
   setUsers(users);
  }, [setUsers, users]);

  const refreshData = () => {
    router.replace(router.asPath);
  };

  return (
    <main>
      <TopBar />
      <CreatePost refreshData={refreshData} />
      <MainContainer>
        {selectedAction === 'For you' ? (
          recommendations.posts.length ? (
            <PostsWithScroll
              key={Date.now()}
              content={recommendations}
              type={PostFetchTypes.RECOMMENDATIONS}
            />
          ) : (
            <div className="text-center text-gray-400">No Posts Yet</div>
          )
        ) : following.posts.length ? (
          <PostsWithScroll
            content={following}
            type={PostFetchTypes.FOLLOWING}
          />
        ) : (
          <div className="text-center text-gray-400">No Posts Yet</div>
        )}
      </MainContainer>
    </main>
  );
}
```

Рис. 3.3. Програмний код сторінки index.tsx

В файлі index.tsx імпортуємо всі головні компоненти вебсторінки. Головний контент знаходиться в html тезі <main>.

TopBar – це компонент, що складається з двох кнопок: For you (усі пости) та Following (пости користувачів, на яких підписаний користувач). В залежності від вибору користувача, будуть відображатися відповідні пости (рис 3.4.).

Рис. 3.4. Компонент TopBar

CreatePost - компонент призначений для створення нових публікацій користувачів (рис. 3.4.). Він надає користувачу інтерфейс для введення тексту, обробки подій введення та відправки даних на сервер. Після успішного створення публікації, компонент оновлює дані для відображення нової інформації.

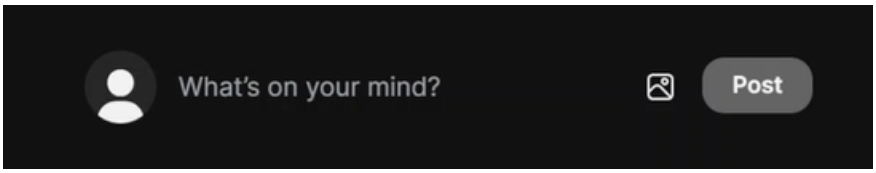

Рис. 3.4. Компонент CreatePost

PostsWithScroll – це клієнтський компонент, який реалізує функціонал нескінченного скролінгу для відображення постів (рис. 3.5.). Компонент отримує початковий вміст постів та інформацію про те, чи є ще пости для завантаження, під час прокрутки сторінки автоматично підвантажує нові пости. Користувачам не потрібно натискати на кнопки для завантаження нових постів, що робить взаємодію з вебзастосунком більш плавною і безперервною. Вони можуть продовжувати переглядати контент без додаткових зусиль. Також, завантаження постів у міру потреби, а не все одразу, допомагає зменшити і оптимізувати використання ресурсів.

Цей компонент включає в себе додатковий компонент Post, який надає користувачу можливість взаємодіяти з публікаціями інших користувачів. Він включає наступні функції:

- Вподобати пост: Користувач може натиснути кнопку "Вподобати", щоб виразити свою симпатію до поста. Ця дія може бути використана для позначення публікацій, які сподобалися користувачеві.

- Зберегти пост: Кнопка "Зберегти" дозволяє користувачеві зберегти публікацію для подальшого перегляду. Це корисна функція для збереження цікавих або корисних постів, які користувач може бажати переглянути знову в майбутньому.

- Зробити репост: Опція "Репост" дозволяє користувачеві поділитися постом у своєму власному профілі або в інших мережах. Це дає змогу розповсюджувати цікаві матеріали серед своїх підписників або друзів.

- Час від публікації поста: Показує час, що пройшов з моменту публікації поста. Ця інформація може бути корисною для користувачів, які хочуть знати, наскільки "свіжим" є контент, або для оцінки актуальності інформації.

- Зробити репорт: При натисканні на кнопку "Більше" у будь-якому пості з’являється випадаюче меню з опціями для репорту, такими як "Spam", "Violence", "Personal Information" та інші.

-

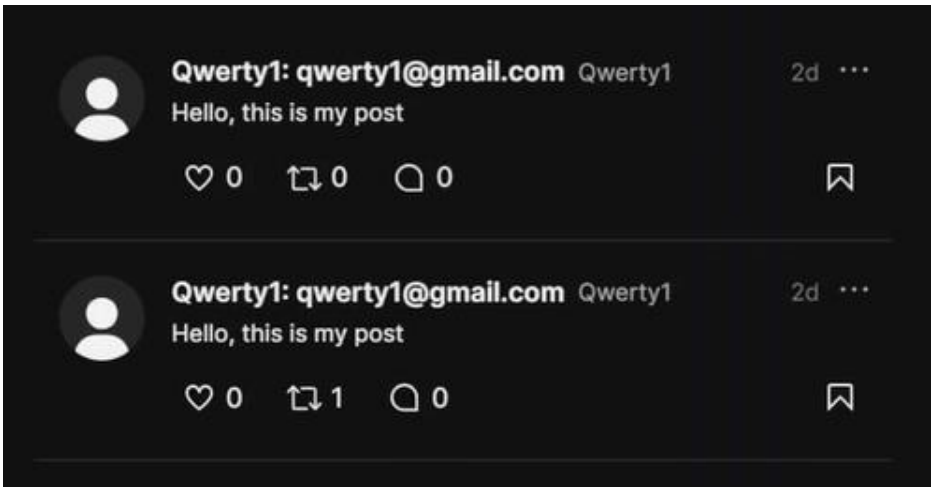

Рис. 3.5. Компонент PostsWithScroll

Після створення основного контенту головної сторінки, наступним кроком є реалізація бічних панелей. Ці компоненти розташовані збоку від основного контенту сторінки та мають додаткові можливості для користувача. Їх функціональність включає навігаційне меню, рекомендації, та останні події в галузі криптовалюти. Реалізація бічних панелей створить більш зручний та ефективний користувацький інтерфейс.

Бічна панель навігації забезпечує зручний доступ користувачам до різних сторінок застосунку (рис. 3.6.). Вона надає можливість швидко переходити між різними секціями, такими як "Головна", "Профіль", "Новини", "Календар". Це робить навігацію по застосунку простою та зручною для користувачів.

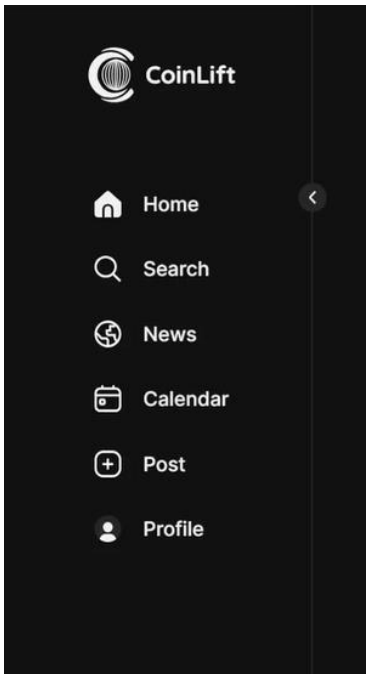

Рис. 3.6. Навігаційний компонент

З правого боку бічна панель складається з персоналізованого компонента, який містить унікальну вибірку користувачів, на яких можна швидко підписатись.

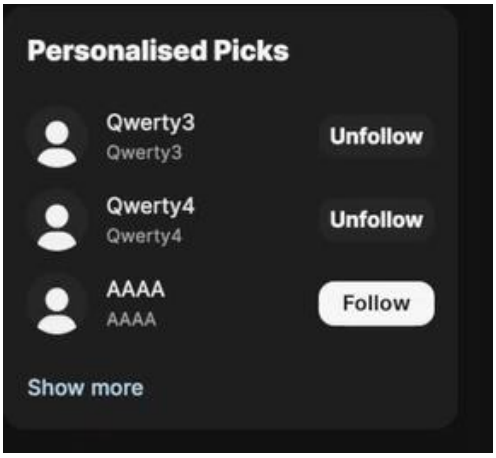

Рис. 3.7. Компонент рекомендацій

3.2.2. Реалізація авторизації користувача

Авторизація користувача є ключовою складовою будь-якого вебзастосунку, особливо в контексті соціальної екосистеми для ентузіастів криптовалют. Вона забезпечує безпеку даних та дозволяє користувачам отримати доступ до основного функціоналу. Авторизація гарантує, що доступ до конфіденційної інформації, такої як персональні дані, отримують лише авторизовані користувачі. Це захищає дані від несанкціонованого доступу та потенційних загроз.

При вході до застосунку першою сторінкою, яку бачить користувач, є Sign In (рис. 3.7.). Тут він має можливість ввійти до свого облікового запису, якщо він вже зареєстрований. Це дозволяє з легкістю повернутися до вже існуючого акаунту.

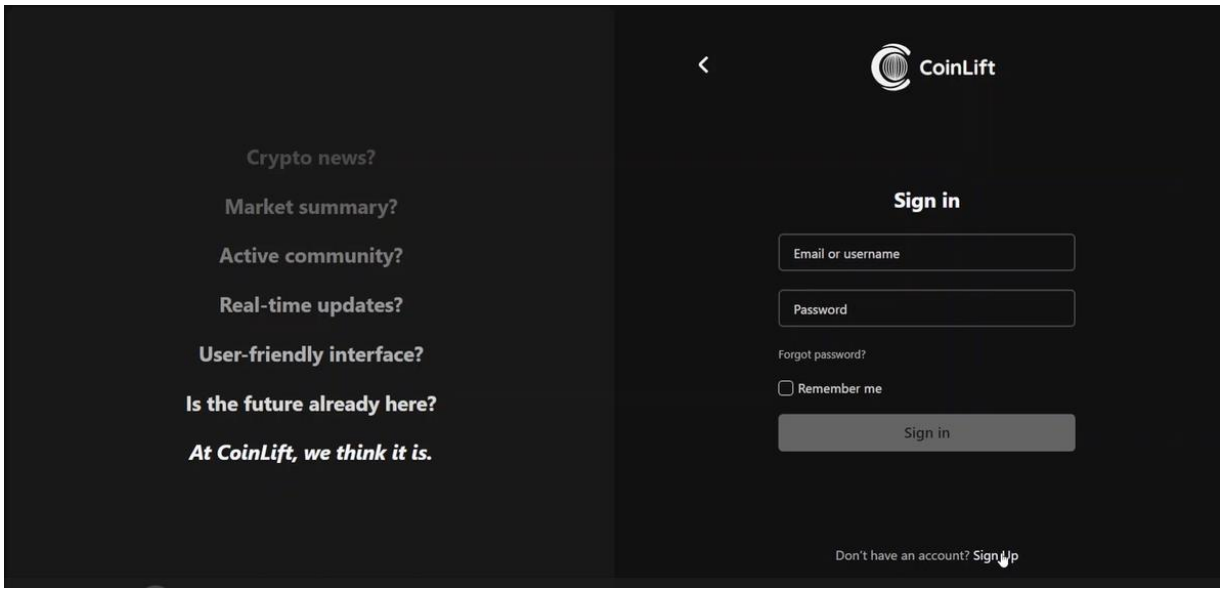

Рис. 3.7. Форма входу до застосунку

Ця сторінка має посилання на реєстрацію, у разі якщо користувач не має облікового запису (рис. 3.7.). Для цього потрібно ввести деякі персональні дані, такі як ім'я, адресу електронної пошти, нікнейм та пароль.

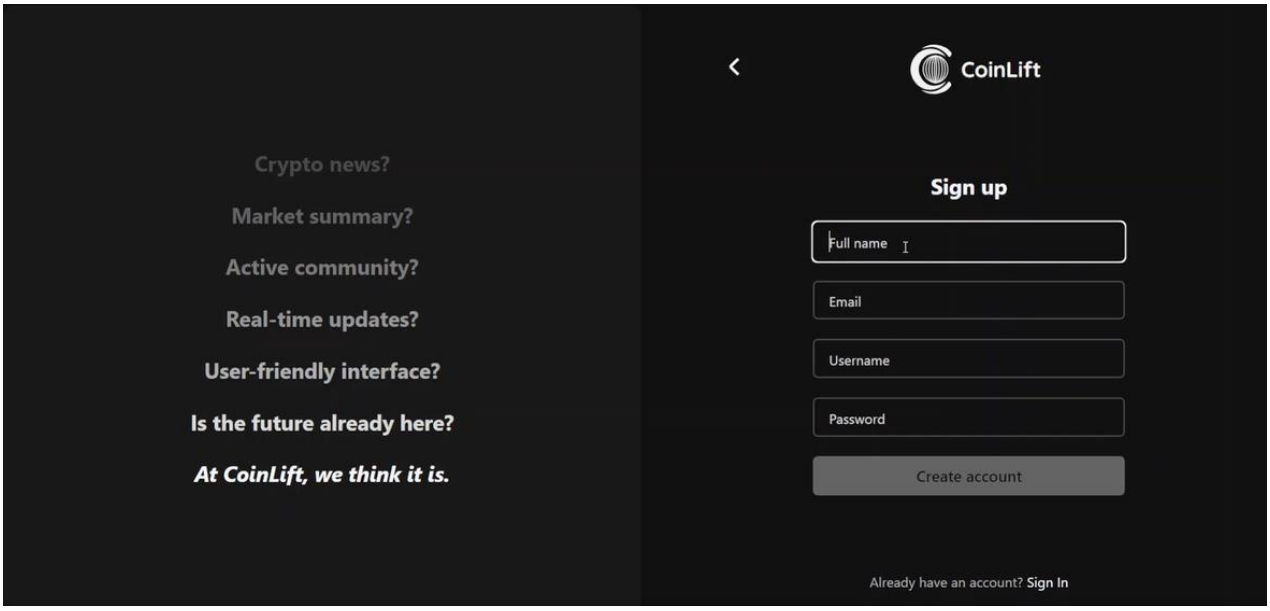

Рис. 3.7. Форма реєстрації

Перед входом в систему система перевірить точність введених даних. У цьому процесі використовується пакет перевірки форми, такий як Formik та Yup. Formik - це бібліотека управління формами, яка спрощує процес створення та обробки форм, підтримуючи управління станом форми, автентифікацію та обробку подій. Yup діє як схема перевірки і дозволяє визначати правила перевірки для кожного поля форми.

Якщо при вході в систему введені дані не відповідають встановленим вимогам (наприклад, неправильний формат адреси електронної пошти або пароля), створити обліковий запис або увійти в нього буде неможливо до тих пір, поки ці помилки не будуть виправлені. За допомогою Formik і Yup є можливість ефективно перевіряти і обробляти дані, що вводяться, забезпечуючи високу точність і надійність при вході в систему. Ці пакети були обрані через їх популярність, гнучкість та зручний інтерфейс.

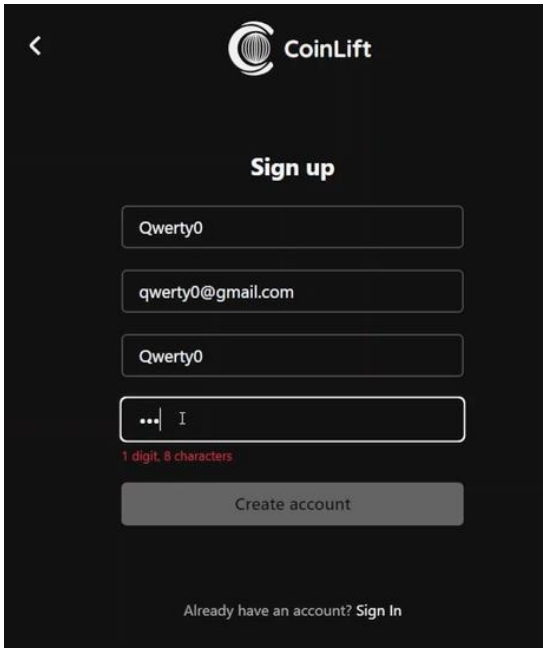

Рис. 3.7. Виявлення помилок валідації

Введені дані було оброблено схемою валідації від yup (рис. 3.8.)

```
export const SignupSchema = Yup.object().shape({
  fullName: Yup.string().required(''),
  username: Yup.string()
    .min(3, 'Username is too short')
    .max(50, 'Username is too long!')
    .required('Username is required'),
  email: Yup.string()
    .email('Please enter a valid email address')
    .required('Email is required'),
  password: Yup.string()
    .required('Password is required')
    .test(
      'password-checks',
      'Password must meet the following criteria:',
      function (value) {
        const errors = [];
        if (!/(?=.*[0-9])/.test(value)) {
          errors.push('1 digit');
        }
        if (!/(?=.*[A-Z])/.test(value)) {
          errors.push('1 capital letter');
        }
        if (!/(?=.{8,})/.test(value)) {
          errors.push('8 characters');
        }
        return errors.length === 0
          ? true
          : this.createError({ message: errors.join(', ') });
      },
    ),
});
```

Рис. 3.8. Схема валідації yup

Під час реєстрації користувача йому автоматично присвоюється унікальний токен доступу. Для збереження цього токена та забезпечення безпеки та автентифікації користувача, він зберігається у формі cookies у браузері

користувача. Зберігання токена доступу у cookies дозволяє вебзастосунку автоматично ідентифікувати користувача при кожному його відвідуванні. Він буде використовуватись для перевірки користувача при подальших запитах.

Після успішної реєстрації потрібно підтвердити адресу електронної пошти за допомогою шестизначного коду, який буде відправлено за цією адресою (рис 3.9.).

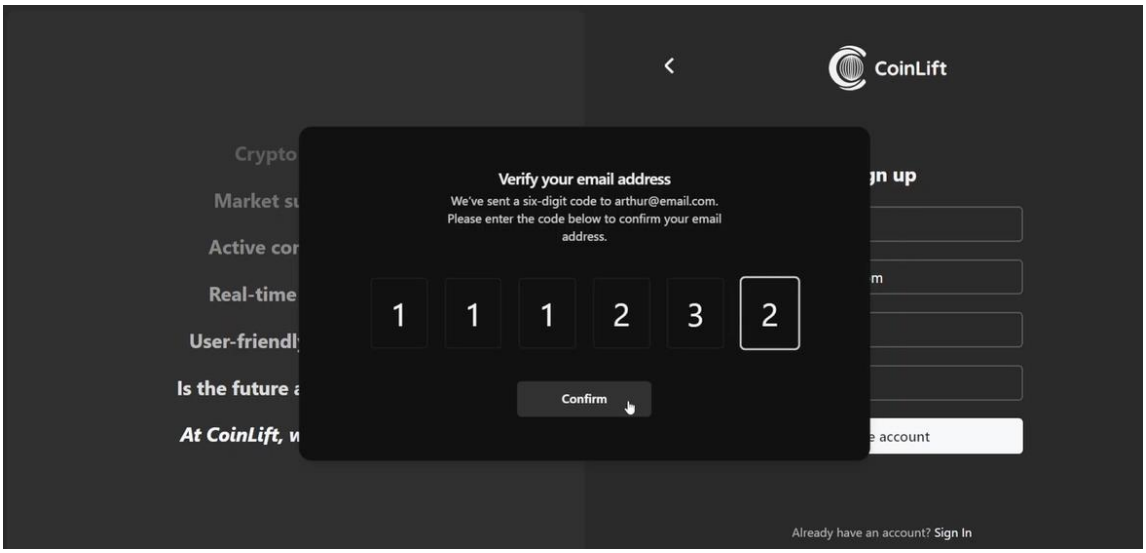

Рис. 3.9. Підтвердження електронної пошти

3.2.3. Реалізація профілю користувача

Створення функціонального та ефективного профілю користувача вебзастосунку є ключовим аспектом для багатьох веб-додатків. Профіль користувача дозволяє зберігати та відстежувати особисті дані користувача, налаштовувати їхні уподобання та надавати персоналізований досвід.

При переході на сторінку "Профіль", користувач отримує можливість докладніше ознайомитися з власним профілем та взаємодіяти з особистими даними (рис. 3.10.).

Основні функціональність цієї сторінки включає:

1. Перегляд особистих постів: Користувач може переглянути всі свої створені пости, що надає можливість відстежувати власну активність в соціальній екосистемі.

2. Перегляд уподобаних та репостнутих постів: Крім того, на сторінці профілю доступний список постів, які користувач відзначив як уподобані або зробив репост, що дозволяє повертатися до цікавого контенту.

3. Детальна інформація про користувача: Сторінка профілю також містить детальну інформацію про користувача, таку як ім'я, фотографія профілю, кількість підписників.

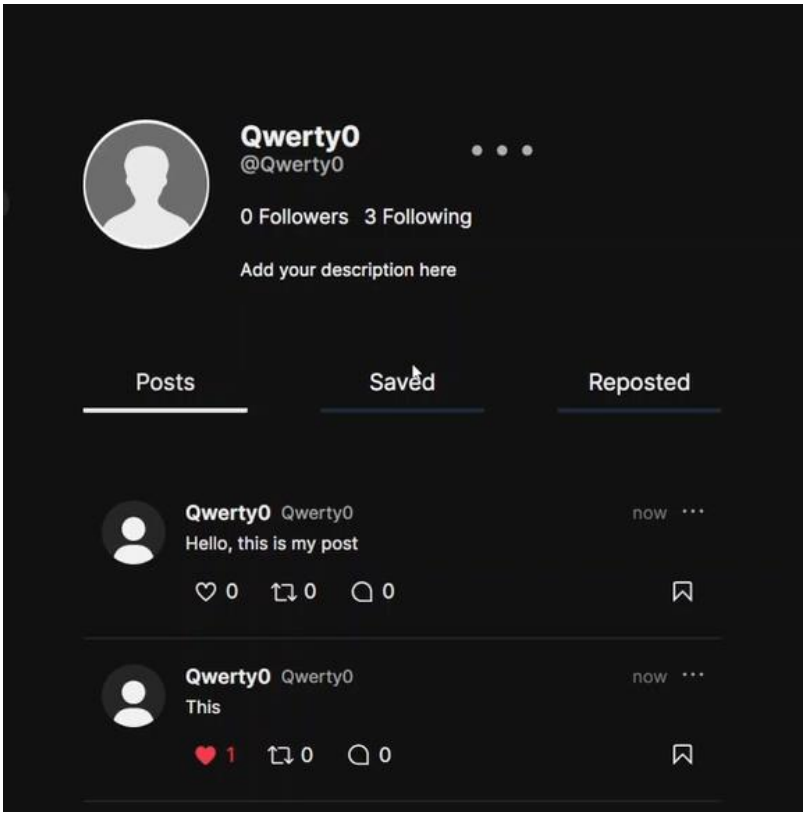

Рис. 3.10. Профіль користувача

3.2.4. Реалізація економічного календарю

Економічний календар - це важливий інструмент для трейдерів, інвесторів та аналітиків, який надає інформацію про майбутні економічні події, що можуть вплинути на криптовалютні ринки. Реалізація економічного календаря у

вебзастосунку може бути дуже корисною. Користувачі зможуть отримувати актуальну інформацію про важливі економічні події, що допоможе їм приймати більш обґрунтовані рішення щодо інвестицій та торгівлі криптовалютами. Для створення інтерактивного інтерфейсу економічного календаря було використано Next.js, Html, Css.

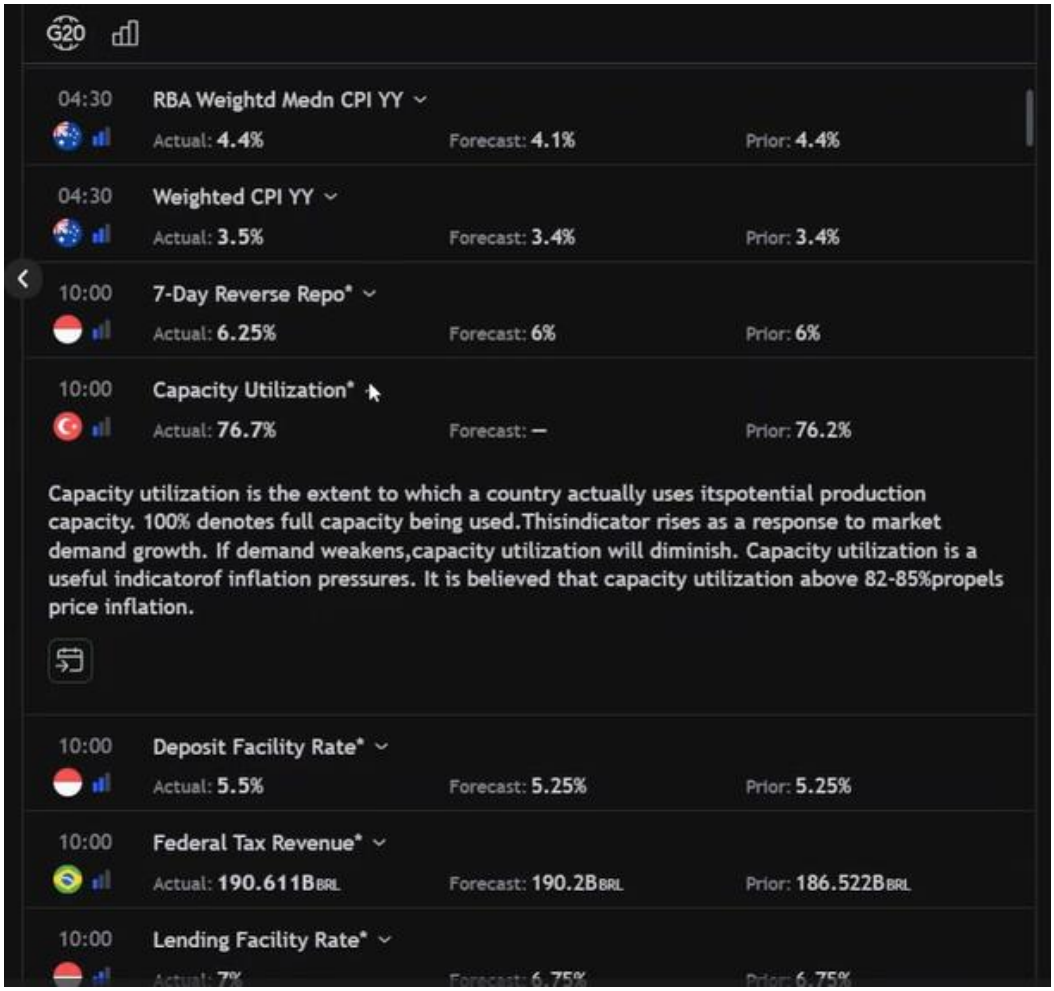

Рис. 3.10. Економічний календар

Цей календар надає змогу користувачам переглядати та відслідковувати останні події, які можуть впливати на криптовалютний ринок. Кожна подія має детальний опис, для цього потрібно просто натиснути на неї.

Загальною метою реалізації такого економічного календаря є забезпечення доступу до актуальної та корисної інформації про економічні події, для певного аналізу ринків.

3.3. Тестування вебзастосунку

Тестування вебзастосунку є критично важливим етапом у процесі розробки програмного забезпечення. Воно дозволяє виявити помилки, підвищити якість та забезпечити стабільну роботу системи.

Для тестування даного застосунку було обрано модульне тестування, яке спрямоване на перевірку окремих модулів або компонентів вебзастосунку. Кожен модуль тестується ізольовано для забезпечення правильності його функціонування (рис. 3.11). У проекті модульне тестування проводилось з використанням таких інструментів:

- Jest: Основний фреймворк для тестування JavaScript, який дозволяє писати тести та перевіряти їх виконання.
- Testing Library: Використовується разом з Jest для тестування компонентів React, дозволяючи фокусуватися на тестуванні користувацького інтерфейсу.
- Mock Service Worker: Використовується для фейкових запитів до серверу.

```
it('renders loader while fetching data', () => {
  render(
    <Provider store={store}>
      <SchemasArgs />
    </Provider>
  );

  expect(screen.getByTestId('loader')).toBeInTheDocument();
});
```

Рис. 3.11. Приклад модульного тесту

Тестування вебзастосунку є необхідним етапом для забезпечення його надійності та функціональності. Використання різних методів тестування дозволяє виявити та виправити помилки на різних рівнях розробки. У даному проекті застосування інструментів, таких як Vitest, Testing Library, Mock Service Worker забезпечило високу якість і стабільність вебзастосунку, що сприяло досягненню поставлених цілей та вимог.

ВИСНОВОК ДО РОЗДІЛУ 3

У третьому розділі було розглянуто кожен етап розробки, від налаштування проекту до впровадження основної функціональності, яка має критичне значення для досягнення високої якості кінцевого продукту.

Перед початком розробки було організувано файлову структуру, адже це є важливим етапом, який визначає організацію коду, розподіл обов'язків між компонентами та загальну архітектуру проекту.

Розглянуто реалізацію таких компонентів, як PostsWithScroll для нескінченного скролінгу, економічного календаря та інших важливих елементів, забезпечує функціональність, зручність і залученість користувачів.

Описано процес реалізації профілю користувача, який дозволяє зберігати та відстежувати особисті дані, налаштовувати уподобання та надавати персоналізований досвід. Це включає можливість перегляду власних постів, збережених або вподобаних публікацій інших користувачів, а також детальну інформацію про активність користувача.

Було показано головні етапи розробки авторизації користувача, яка є ключовою складовою будь-якого вебзастосунку, особливо в контексті соціальної екосистеми для ентузіастів криптовалют. Вона забезпечує безпеку даних та дозволяє користувачам отримати доступ до основного функціоналу.

Загалом, успішна реалізація вебзастосунку вимагає чіткої організації, координації роботи, глибокі знання таких технологій як Next.js та Typescript, та постійного вдосконалення. Інтеграція таких сучасних технологій, допомагає створити продуктивний, масштабований та зручний для користувачів продукт, який відповідає вимогам сучасного ринку криптовалют.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було реалізовано вебзастосунок соціальної екосистеми для ентузіастів криптовалют з використанням фреймворку Next.js, спрямований на покращення взаємодії між ентузіастами через активний та актуальний інформаційний простір.

Метою цього проекту є розширення можливостей для знаходження партнерів та співпраці у криптовалютній галузі. Основні завдання кваліфікаційної роботи включають покращення взаємодії між ентузіастами криптовалют, створення сприятливого середовища для обміну думками, досвідом та інформацією, забезпечення актуального інформаційного обміну щодо останніх подій та трендів у світі криптовалют, а також розширення мережевих можливостей для знаходження партнерів та співпраці у сфері криптовалютних проектів.

В ході роботи було проведено детальний аналіз теоретичних понять, пов'язаних з вебзастосунками та блокчейн-технологіями. Це включало дослідження принципів роботи вебзастосунків, їх архітектури, а також особливостей та переваг блокчейну. Особливу увагу в кваліфікаційній роботі було приділено обґрунтуванню вибору фреймворку Next.js для розробки вебзастосунку. Розглянуто його переваги, зокрема підтримка серверного та клієнтського рендерингу, а також можливість інтеграції з іншими сучасними технологіями, такими як TypeScript, Zustand та інші.

Під час розробки були реалізовані ключові компоненти, такі як PostsWithScroll для нескінченного скролінгу, профіль користувача та економічний календар. Детально описано процес реалізації профілю користувача, який дозволяє зберігати та відстежувати особисті дані, налаштовувати уподобання та надавати персоналізований досвід. Це включає можливість перегляду власних постів, збережених або вподобаних публікацій інших користувачів, а також детальну інформацію про активність користувача.

Також у кваліфікаційній роботі враховано вимоги до зручності та зрозумілості інтерфейсу користувача. Було створено інтуїтивно зрозумілий інтерфейс, який забезпечує користувачам легкий доступ до всіх функцій вебзастосунку, включаючи перегляд та створення постів, використання економічного календаря та управління особистими профілями. Завдяки цьому вебзастосунок стає зручним інструментом для взаємодії криптоентузіастів та сприяє розвитку криптовалютної спільноти.

Для ефективної роботи застосунку було проведено модульне тестування яке забезпечило правильність роботи основних функцій вебзастосунку, його надійність, продуктивність та зручність використання. У процесі тестування було забезпечено перевірку окремих компонентів вебзастосунку, гарантуючи їх правильну роботу. Використання Vitest дозволило швидко виявляти та виправляти помилки в коді. Використання Mock Service Worker забезпечило імітацію API-запитів, що дозволило тестувати взаємодію з сервером без необхідності підключення до реального серверу.

Результати розробки вебзастосунку соціальної екосистеми для ентузіастів криптовалют з використанням фреймворку Next.js можуть значно покращити взаємодію та комунікацію між учасниками криптовалютної спільноти. Вебзастосунок надає платформу для обміну думками, досвідом та інформацією, що сприяє зміцненню зв’язків та співпраці між користувачами. Завдяки інтуїтивно зрозумілому інтерфейсу та широкому спектру функціональних можливостей, таких як перегляд та створення постів, використання економічного календаря та управління особистими профілями, вебзастосунок стає незамінним інструментом для криптоентузіастів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a web server? [Electronic resource]. Access mode: https://developer.mozilla.org/enUS/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server (last access 10.05.2024). – Title from the screen.

2. What is the difference between web page, website, web server, and search engine? [Electronic resource]. Access mode: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/Pages_sites_servers_and_search_engines (last access 09.05.2024). – Title from the screen.

3. Best Ways to Make a Website: Free & Paid Options [Electronic resource]. Access mode: https://tech.co/website-builders/3-best-ways-make-website (last access 10.05.2024). – Title from the screen.

4. React Optimization Techniques to Help You Write More Performant Code [Electronic resource]. Access mode: https://www.freecodecamp.org/news/react-performance-optimization-techniques/ (last access 11.05.2024). – Title from the screen.

5. Website optimization techniques you need to be using [Electronic resource]. Access mode: https://www.hotjar.com/website-optimization/techniques/ (last access 11.05.2024). – Title from the screen.

6. The Testing Pyramid: Simplified for One and All [Electronic resource]. Access mode: https://www.headspin.io/blog/the-testing-pyramid-simplified-for-one-and-all (last access 11.05.2024). – Title from the screen.

7. What is Blockchain Technology? [Electronic resource]. Access mode: https://aws.amazon.com/what-is/blockchain/?nc1=h_ls&aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc (last access 11.05.2024). – Title from the screen.

8. What is TypeScript? [Electronic resource]. Access mode: https://www.typescriptlang.org/ (last access 12.05.2024). – Title from the screen.

9. Why should I use TypeScript? [Electronic resource]. Access mode: https://www.w3schools.com/typescript/typescript_intro.php/ (last access 18.05.2024). – Title from the screen.

10. Learning TypeScript. Enhance Your Web Development Skills Using Type-Safe JavaScript; 2022. 318 с.

11. Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production; 1nd edition. 2022. 366 с.

12. Understanding CSR, SSR, SSG, and ISR: A Next.js Perspective [Electronic resource]. Access mode: https://bootcamp.uxdesign.cc/understanding-csr-ssr-ssg-and-isr-a-next-js-perspective-fcaf36686de6 (last access 14.05.2024). – Title from the screen.

13. Advantages and disadvantages of next.js – 2024 updated version [Electronic resource]. Access mode: https://pagepro.co/blog/pros-and-cons-of-nextjs/ (last access 14.05.2024). – Title from the screen.

14. React State Management — using Zustand [Electronic resource]. Access mode: https://medium.com/globant/react-state-management-b0c81e0cbbf3 (last access 14.05.2024). – Title from the screen.

15. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers; 1th edition. 2019. 192 с.

16. Положення про кваліфікаційні роботи (проєкти) здобувачів вищої освіти національного авіаційного університету; СМЯ НАУ П 03.01(10) – 03 – 2024. Київ 2024, 62с.