МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА СИСТЕМИ КЕРУВАННЯМ ВМІСТОМ З ВИКОРИСТАННЯМ КАРКАСІВ ВЕБДОДАТКІВ»**

Виконав: студент 4 курсу, групи 6.1219-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
О.О. Петрушенко
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Лимаренко Ю.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент директор ТОВ «Голден-Тім» Калін М.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Петрушенку Олексію Олексійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка системи керуванням вмістом з використанням каркасів вебдодатків

керівник роботи Лимаренко Юлія Олексіївна, к.т.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2022 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
1. Основні теоретичні відомості.
3. Розробка системи керуванням вмістом з використанням каркасів вебдодатків.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.02.2023	
2.	Збір вихідних даних.	15.02.2023	
3.	Обробка методичних та теоретичних	22.02.2023	
	джерел.		
4.	Розробка першого та другого розділу.	05.04.2023	
5.	Розробка третього розділу.	10.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	21.06.2023	

Студент _____ О.О. Петрушенко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Ю.О. Лимаренко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка системи керуванням вмістом з використанням каркасів вебдодатків»: 52 с., 40 рис., 8 джерел.

АВТЕНТИФІКАЦІЯ, АВТОРИЗАЦІЯ, БАЗА ДАНИХ, БЕКЕНД, ВЕБДОДАТОК, ТОКЕН.

Об’єкт дослідження – основні техніки та інструменти, що використовуються для створення вебдодатків.

Мета цієї роботи – розробити функціональний вебсайт та форум для рок-групи, що б дозволили фанатам взаємодіяти, обговорювати новини, музичні релізи та концерти, а також отримувати актуальну інформацію про групу.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання та проєктування програмного забезпечення.

У кваліфікаційній роботі було проведено аналіз області музичних груп, розглянуто існуючі вебресурси в цій сфері, а також описано інструменти для створення вебдодатків. На основі зібраних теоретичних знань було розроблено вебсайт та форум для рок-групи. Результатом виконання цієї роботи є вебдодаток з клієнт-серверною архітектурою, створений з використанням технологій Next.js, MongoDBта Node.js. Веб-сайт розроблений таким чином, щоб надати користувачам можливість активної взаємодії, обговорення новин, релізів та інших тем, пов’язаних з рок-групою.

SUMMARY

Bachelor’s qualifying paper «Development of a Content Management System using Web Frameworks»: 52 pages, 40 figures, 8 references.

AUTHENTICATION, AUTHORIZATION, DATABASE, BACKEND, WEB APPLICATION, TOKEN.

Object of study – basic techniques and tools used to create web applications.

The aim of the study is to develop a functional website and forum for a rock band that would allow fans to interact, discuss news, music releases and concerts, and receive up-to-date information about the band.

Research methods are methods of collecting and analyzing software requirements, software modeling and design methods.

The qualification work analyzed the field of music bands, reviewed existing web resources in this area, and described tools for creating web applications. Based on the collected theoretical knowledge, a website and a forum for a rock band were developed. The result of this work is a web application with a client-server architecture created using Next.js, MongoDB, and Node.js technologies. The website is designed to allow users to actively interact, discuss news, releases, and other topics related to the rock band.

ЗМІСТ

ВСТУП

В сучасному світі вебсайти та форуми відіграють важливу роль у побудові спільноти та взаємодії зі споживачами, особливо в контексті музичної індустрії. Вони дозволяють виконавцям підтримувати зв'язок з фанатами, поширювати новини та оновлення, а також ділитися контентом. Відповідно, розробка системи управління контентом для сайту рок-групи та пов'язаного з ним форуму є важливим завданням, оскільки це дозволяє спрощувати процеси комунікації та взаємодії.

Така система може включати функції, такі як редагування та публікація новин, подій, музики, відео, фотографій, управління форумом та іншими формами взаємодії з користувачами. Розробка такої системи підвищує ефективність управління контентом, забезпечує своєчасне та організоване оновлення вебресурсів, та сприяє збільшенню взаємодії з фанатами та користувачами.

Предмет дослідження – процес та фактори розробки системи управління контентом за допомогою каркасів вебзастосунків.

Мета: розробити та протестувати систему управління контентом для сайту рок-групи та форуму.

Завдання дослідження:

– проаналізувати існуючі системи управління контентом для сайтів музичних груп та форумів;
– спроєктувати систему управління контентом, яка відповідає вимогам та потребам рок-групи та її спільноти.

Методи дослідження: теоретико-методологічний аналіз, проєктування, моделювання, програмування.

Інструменти які використані у розробці системи: Visual Studio Code, Next.js, Node.js,Yarn [3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметного середовища

Предметне середовище для розробки системи керування вмістом з використанням каркасів вебдодатків охоплює ряд різноманітних областей. Основний фокус роботи полягає в створенні вебсайту для рок-групи з приєднаним форумом, який вимагає інтеграції різних технологій та інструментів.

У технічному плані предметне середовище включає в себе ряд технологій та інструментів, які потрібні для розробки вебсайту та форуму. Це включає, але не обмежується, вибором підходящого вебфреймворка, CMS, бази даних, технологій фронтенду (HTML, CSS, JavaScript), технологій бекенду (наприклад, PHP, Python, Ruby, Node.js) [6].

Предметне середовище також включає в себе важливі нормативно-правові аспекти, такі як захист даних користувачів, цифрові права, авторські права, та вимоги до доступності.

Важливим аспектом є взаємодія між різними користувачами сайту. Веб-сайт та форум повинні бути розроблені таким чином, щоб вони були зручними та привабливими для всіх користувачів. Всі вони повинні мати зручний доступ до контенту, незалежно від свого віку, досвіду, пристрою, який вони використовують, чи інших особливостей.

Розробка вебсайту та форуму в такому предметному середовищі вимагає обміркованого підходу, гнучкості та здатності враховувати різноманітність та складність завдань. Використання каркасів веб-додатків є однією з можливостей, яка може допомогти у вирішенні цих завдань, пропонуючи рішення, які вже були перевірені та оптимізовані для різних сценаріїв використання.

Також важливо врахувати, що такий сайт може вимагати інтеграції з

різними зовнішніми сервісами, такими як соціальні мережі, платформи для прослуховування музики, сервіси продажу квитків на концерти тощо.

Загалом, предметне середовище розробки системи керування вмістом з використанням каркасів вебдодатків включає в себе широкий спектр різноманітних технічних, соціальних, культурних та законодавчих факторів.

1.2 Опис можливих інструментів розробки

Існує багато можливих інструментів та технологій для розробки вебсайтів та систем управління контентом, що можуть використовуватися для проекту, наприклад, вебфреймворки. Вони допомагають в автоматизації багатьох загальних задач розробки вебсайтів. Є декілька відомих вебфреймворків, таких як Django (Python), Ruby on Rails (Ruby), Laravel (PHP), Express.js (Node.js), Angular, React.js та Vue.js.

Django – це високорівневий веб-фреймворк, написаний на мові програмування Python, який надає розробникам потужні інструменти для швидкої та ефективної розробки веб-додатків. Його унікальність полягає в комбінації простоти в використанні та гнучкості, що дозволяє розробникам швидко розгортати веб-сайти будь-якої складності. Django пропонує модульну структуру, що дозволяє легко організувати проєкт, і вбудовану адміністративну панель, яка спрощує управління даними. Завдяки вбудованій ORM, Django дозволяє розробникам працювати з базою даних в об'єктно-орієнтованому стилі, зменшуючи необхідність у прямому написанні SQL-запитів. Крім того, Django надає потужні засоби для обробки URL-запитів, керування сеансами, аутентифікації користувачів і багато інших функцій, які допомагають забезпечити безпеку і продуктивність додатків.

Загалом, Django є надійним і стабільним вибором для розробки веб-додатків, який поєднує широкий функціонал з простотою використання і забезпечує розробникам зручні інструменти для ефективної роботи.

Ruby on Rails, також відомий просто як Rails, є веб-фреймворком, розробленим на мові програмування Ruby. Він пропонує розробникам потужні та зручні інструменти для швидкої та ефективної розробки веб-додатків. Унікальність Rails полягає у концепції "Convention over Configuration" (конвенція над конфігурацією), що спрощує розробку, оскільки багато рішень та конфігурацій вже встановлені за замовчуванням. Використовуючи модель-контролер-представлення (MVC) для організації логіки додатка та його представлення, Rails надає зручні інструменти для роботи з базами даних через ActiveRecord. Фреймворк також має потужну систему маршрутизації, вбудовану підтримку тестування і широкий спектр розширень та пакетів, які допомагають розробникам прискорити роботу та створити високоякісні веб-додатки. Rails має велику спільноту розробників, що забезпечує доступ до багатьох ресурсів, документації та підтримки.

В цілому, Ruby on Rails є потужним та популярним вибором для розробки веб-додатків, який пропонує зручну та елегантну розробку за допомогою Ruby.

Laravel – це веб-фреймворк, розроблений на мові програмування PHP, який надає розробникам потужні та елегантні інструменти для швидкої та ефективної розробки веб-додатків. Його унікальність полягає в простоті використання, зрозумілому синтаксисі та широких можливостях розширення. Laravel пропонує зручну модель MVC (Model-View-Controller) для організації логіки додатка та його представлення, що дозволяє розробникам ефективно розділити різні компоненти додатку. Фреймворк має багатий набір вбудованих функцій, таких як маршрутизація, обробка запитів та відповідей, робота з базою даних, аутентифікація, кешування та багато інших. Laravel також зосереджується на безпеці, надаючи захист від потенційних атак та шкідливих дій.

Завдяки розширюваній архітектурі та великій спільноті розробників, Laravel є потужним інструментом для створення різноманітних веб-додатків, від простих блогів до складних корпоративних систем.

Express.js – це веб-фреймворк для розробки веб-додатків з використанням мови програмування Node.js. Його головна перевага полягає в простоті та гнучкості, що дозволяє розробникам швидко створювати масштабовані та ефективні додатки. Express.js пропонує потужні інструменти для маршрутизації, обробки запитів та відправлення відповідей, а також розширюваність за допомогою плагінів. Він також підтримує middleware, що дозволяє розробникам легко додавати проміжні шари для обробки запитів та виконання спеціалізованих функцій.

Express.js дозволяє розробникам швидко побудувати надійні та розширювані веб-додатки, зосереджуючись на основних функціях та забезпечуючи гнучкість для втілення унікальних вимог проєкту.

Angular – це потужна платформа для розробки веб-додатків, яка забезпечує компонентну архітектуру та пропонує повний набір інструментів для швидкої та ефективної розробки. Використовуючи мову програмування TypeScript, Angular надає розробникам уніфікований підхід до створення високоякісних, модульних та легко підтримуваних додатків. Завдяки вбудованим функціям, таким як маршрутизація, обробка форм, двостороннє зв'язування даних та багато інших, Angular дозволяє швидко створювати багатофункціональні додатки з високою продуктивністю та відзивчивим інтерфейсом.

React.js – це бібліотека JavaScript, яка дозволяє розробникам побудувати ефективні та модульні веб-додатки. Вона використовує компонентний підхід для розбиття додатка на невеликі, самодостатні компоненти. React.js також використовує віртуальний DOM для оптимізації оновлення інтерфейсу, що забезпечує швидке та ефективне відображення змін. Завдяки декларативному підходу React.js спрощує розробку, позбавляючи розробника необхідності безпосередньо маніпулювати DOM та стежити за станом додатка.

В цілому, React.js дозволяє розробникам будувати сучасні та інтерактивні веб-додатки.

1.3 Етапи створення веб-сайту

Створення вебсайту починається з детального планування і визначення основних вимог. На цьому етапі визначаються ключові функції, вимоги до користувачів, інтерфейс, вимоги до пристроїв та платформ, а також інші важливі аспекти. Завдяки цьому можна визначити обсяг роботи, тривалість проєкту та ресурси, необхідні для його виконання.

Після визначення вимог йде процес дизайну. Дизайн – це більше, ніж просто вигляд за стосунку – він також визначає, як користувачі взаємодіють з додатком, і впливає на загальний досвід користувачів. Дизайн працює над створенням візуального враження користувача, включаючи елементи інтерфейсу користувача, кольорову схему, шрифти, кнопки, відступи та інше. Закінчивши з дизайном, перейшов до написання коду.

Кодування – це процес, який залежить від складності та розміру проєкту, і він може зайняти від кількох тижнів до кількох місяців. На цьому етапі створюється робоча версія вебсайту, використовуючи різноманітні технології та мови програмування.

Після того, як основний код було написано, настає час тестування. Це може включати ручне тестування, автоматизоване тестування, тестування навантаження, тестування безпеки та інше. Тестування вебсайту – це критично важливий етап, який допомагає виявити і виправити помилки та проблеми.

Після успішного тестування вебсайт готовий до його розміщення на хостингу. Це означає, що він розміщується на сервері, де користувачі можуть отримати до нього доступ. Розгортання може включати налаштування сервера, налаштування бази даних, конфігурацію безпеки та інші технічні аспекти.

1.4 Вебсайти, що вирішують подібну задачу

В контексті розробки системи управління контентом, існує великий спектр програмних рішень, що пропонують різні функціональності та рівень

налаштування. В процесі пошуку, виявлено форум, що виконує функції аналогічні до основних завдань проєкту.

Reddit є одним з найпопулярніших в інтернеті місць для обміну думками, ідеями, обговоренням та контентом. Це велика спільнота, яка складається з безлічі «субреддітів» – тематичних форумів, присвячених певним темам, від музики до наукових дискусій.

Система Reddit організована довкола користувацького контенту. Користувачі публікують посилання, відео, картинки, текстові дописи, після чого інші користувачі можуть оцінювати цей контент, залишати коментарі та вести дискусії. На основі рейтингу (кількість «upvotes» та «downvotes») контенту формується його позиція на сторінці субреддіту або головній сторінці.

Кожен субреддіт має свої правила та модераторів, які контролюють порушників і видаляють неприйнятний контент. Це дозволяє створювати здорові та конструктивні дискусії на різні теми.

Що стосується інтерфейсу, Reddit відрізняється своєю простотою та зручністю. Головна сторінка відображає найпопулярніші дописи з усіх субреддітів, до яких користувач підписався. На сторінці субреддіта відображаються дописи, пов'язані з певною темою.

Також Reddit має систему повідомлень, що дозволяє користувачам вести приватні бесіди, а також отримувати повідомлення про нові коментарі та відповіді на їх дописи. Це робить Reddit зручним інструментом для обміну ідеями, обговорення та спілкування на різні теми.

2 ПРОЄКТУВАННЯ

2.1 Технічне завдання

Перед тим, як перейти до програмної реалізації вебсайту та форуму для рок-групи, необхідно сформулювати технічне завдання, визначити технічні вимоги до додатку та етапи його створення.

2.1.1 Найменування та область застосування

Програмний продукт, що розробляється, отримує найменування: «Вебсайт і форум для рок-групи».

Додаток призначений для автоматизації взаємодії між фанатами групи, а також для надання їм платформи для обговорення творчості групи, новин, концертів та інших тем, пов'язаних з групою.

2.1.2 Підстава для розробки

Застосунок розробляється на підставі робочої програми по курсу «Дипломна робота», затвердженої наказом від 26 січня 2023р. №102-с.

2.1.3 Призначення розробки

Призначенням розробки в рамках цієї дипломної роботи є створення інтегрованої вебплатформи для рок-групи та її шанувальників, яка буде включати в себе вебсайт групи та пов'язаний з ним форум.

2.1.4 Вимоги до функціональних характеристик

Авторизація та Реєстрація: платформа має мати систему реєстрації та авторизації для користувачів, що забезпечує можливість взаємодії на форумі.

Форум: платформа має мати форум для обговорень, де користувачі можуть створювати теми, ділитися думками та обговорювати різноманітні аспекти, пов'язані з групою.

Новини: платформа повинна мати секцію для публікації новин та оновлень від групи.

Тури: секція з інформацією про тури необхідна для користувачів, які хочуть відслідковувати гастролі групи.

Історія групи: Інформація про історію групи є важливою для фанів, які хочуть знати більше про їх улюблену групу.

2.1.5 Вимоги до надійності

Надійність – це критичний аспект будь-якої системи або програмного засобу.

Стабільність: вебсайт повинен працювати стабільно без збоїв або зависань. Всі функції та можливості повинні працювати без помилок, забезпечуючи безперебійну роботу вебсайт.

Відмовостійкість: у випадку внутрішніх помилок або помилок, спричинених зовнішніми факторами (наприклад, втрата мережевого з'єднання), вебсайт повинен відновитися автоматично і продовжувати нормальну роботу

Резервне копіювання та відновлення даних: якщо в додатку використовуються користувацькі дані, необхідно регулярно робити їх резервні копії та мати можливість відновлення в разі втрати.

2.1.6 Вимоги до технічних засобів та програмної сумісності

При створенні будь-якого програмного продукту важливо враховувати вимоги до апаратного забезпечення та програмної сумісності, щоб забезпечити правильну роботу та оптимальну продуктивність.

Апаратні вимоги для користувача:

- комп’ютер, ноутбук або мобільний пристрій з підключенням до інтернету;
- браузер останньої версії для оптимальної роботи;
- інтернет-з’єднання.

2.2 Проєктування системи

Проєктування системи для вебсайту розділене на декілька ключових етапів, що включають розробку архітектури сайту, структури даних, дизайну.

Архітектура сайту повинна бути простою та інтуїтивно зрозумілою, щоб фанати легко могли знайти потрібну інформацію. Архітектура включає наступні розділи:

- головна сторінка;
- біографія;
- учасники групи;
- дискографія;
- тури;
- форум.

Система повинна мати базу даних для зберігання логіну(пошти), паролів, тем, коментарів та іншого контенту сайту. Для форуму найбільше підходить не реляційна база даних MongoDB.

MongoDB – це високопродуктивна, гнучка система управління базами даних типу NoSQL, яка створена для забезпечення масштабування і зберігання

даних в дуже великих обсягах. З наступних причин MongoDB – ідеальний вибір для форуму.

MongoDB дозволяє використовувати гнучкі схеми даних. Це означає, що ви можете зберігати різні типи даних в одному місці, а також легко модифікувати структуру даних без необхідності модифікувати всю базу даних. Форуми часто містять велику кількість різноманітних даних, таких як пости користувачів, теми, профілі користувачів тощо, які зручно зберігати в гнучких схемах.

База даних була розроблена з метою легкого масштабування, що дозволяє з легкістю додавати нові сервери та обробляти великі обсяги даних. Форуми, особливо популярні, можуть швидко рости і обробляти великі обсяги даних, тому масштабованість є критично важливою.

Технологія використовує систему зберігання даних у форматі BSON, що дозволяє швидко зчитувати та записувати дані. Це особливо важливо для форумів, де користувачі очікують швидкого відгуку на свої запити.

Також, MongoDB має багато вбудованих API для різних мов програмування, що спрощує розробку та інтеграцію з різними системами.

Дизайн інтерфейсу користувача повинен бути розроблений з урахуванням бренду групи та вподобань її фанатів. Важливо забезпечити зручність користування сайтом, чіткість навігації та оптимальний вигляд на різних пристроях та розмірах екрану (адаптивний дизайн).

2.2.1 Use Case Diagram

Резервне копіювання та відновлення даних: якщо в додатку використовуються користувацькі дані, необхідно регулярно робити їх резервні копії та мати можливість відновлення в разі втрати.

Use Case Diagram, або діаграма використання, це графічний інструмент, що використовується в моделюванні вимог до програмного забезпечення. Вона

часто використовується в рамках UML (Unified Modeling Language), що є стандартом у сфері проєктування та моделювання програмного забезпечення.

У контексті веб-сайтів, Use Case Diagram допомагає визначити основні актори, їхні дії (використання) та взаємодію з системою. Акторами можуть бути різні користувачі веб-сайту: звичайні відвідувачі, адміністратори, модератори форуму тощо.

Наприклад, на веб-сайті рок-групи акторами можуть бути відвідувачі сайту, користувачі, що зареєстровані на форумі, модератори та адміністратори сайту. Всі вони мають свої власні дії: перегляд новин групи, коментування на форумі, модерація коментарів, управління контентом сайту відповідно.

Використання Use Case Diagram допомагає команді розробників отримати ясне розуміння потреб користувачів від сайту, які функції потрібно реалізувати, і як ці функції мають взаємодіяти між собою (див. рис. 2.1).

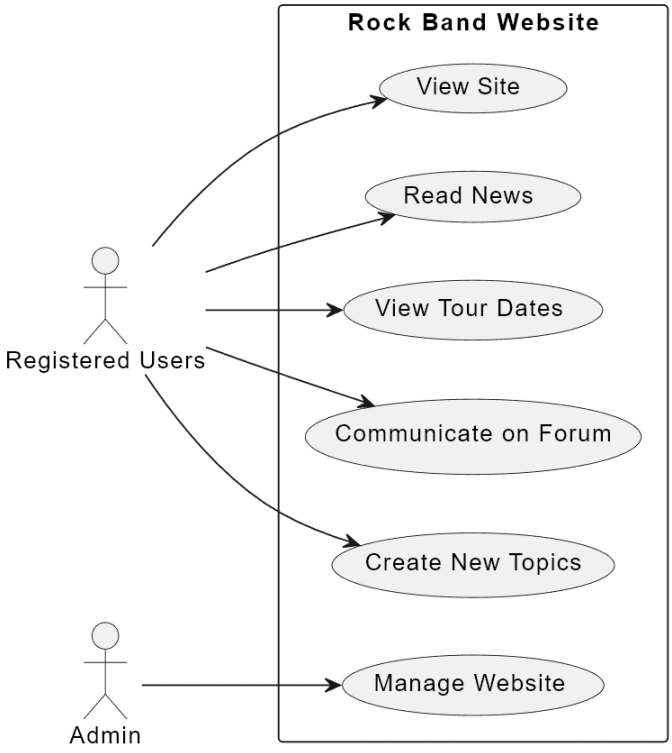

Рисунок 2.1 – Use Case Diagram

На цій діаграмі представлені актори «Зареєстровані користувачі» та «Адміністратор». Зареєстровані користувачі можуть переглядати сайт, читати

новини, переглядати дати турів, спілкуватись на форумі та створювати нові теми. Адміністратор займається менеджментом сайту.

2.2.2 Діаграма розгортання

Діаграма розгортання є одним з типів UML (Unified Modeling Language) діаграм і використовується для візуального зображення фізичного розташування компонентів системи або програмного забезпечення на різних обчислювальних вузлах або серверах. Вона надає зрозумілу та зручну модель для відображення взаємодії між компонентами системи та їх фізичним розташуванням.

У розробці вебсайтів діаграма розгортання допомагає описати архітектуру системи та розподілених компонентів, які використовуються для функціонування вебсайту. Вона вказує, які компоненти розташовані на сервері, клієнтському комп'ютері або інших обчислювальних вузлах. Діаграма розгортання відображає такі елементи:

- вказує, на яких серверах розташовані різні компоненти системи;
- вказує пристрої, які використовуються для доступу до вебсайту;
- вказує, як дані передаються між серверами та клієнтами;
- демонструє з'єднання між компонентами системи;
- надає інформацію про комунікаційні протоколи.

Діаграма розгортання допомагає зрозуміти фізичну архітектуру системи та її компонентів. Вона допомагає виявити потенційні проблеми з розгортанням, такі як недостатні обчислювальні ресурси або недостатнє з'єднання мережі. Діаграма розгортання також служить основою для планування розгортання системи, визначення вимог до обладнання та налаштування серверів та інфраструктури (див. рис. 2.2).

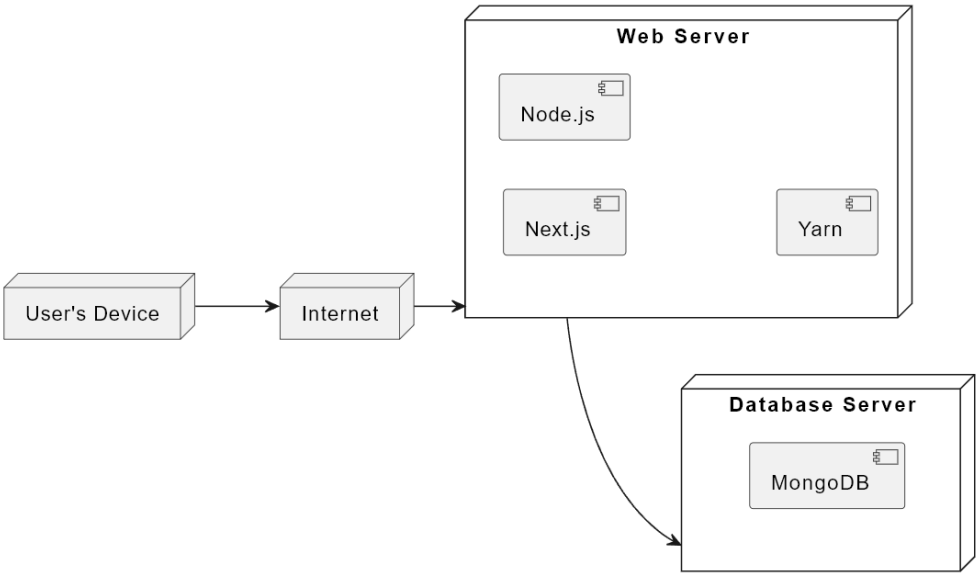

Рисунок 2.2 – Діаграма розгортання

На цій діаграмі пристрій користувача (User’s Device) з’єднується з інтернетом, який в свою чергу з’єднується з веб-сервером. Веб-сервер включає в себе Next.js, Node.js. Вебсервер також з'єднаний з сервером бази даних, який використовує MongoDB.

2.2.3 Діаграма послідовності

Діаграма послідовності є одним із типів діаграм UML (Unified Modeling Language) і використовується для візуального зображення взаємодії між об'єктами або компонентами системи відповідно до послідовності дій. Вона надає зрозумілу модель для відображення того, як об'єкти взаємодіють між собою та обмінюються повідомленнями в певному порядку.

У розробці вебсайтів діаграма послідовності допомагає описати взаємодію між різними компонентами веб-додатку, включаючи клієнтську сторону (браузер або інший клієнтський програмний інтерфейс) та серверну сторону (веб-сервер, база даних тощо). Вона дозволяє зрозуміти, які об'єкти взаємодіють між собою та в якому порядку вони обмінюються повідомленнями.

Діаграма послідовності відображає такі елементи:

– об’єкти, які беруть участь у взаємодії (браузери, сервери, бази даних тощо);
– який об’єкт ініціює взаємодію та які повідомлення він надсилає іншим об'єктам;
– часові лінії, які показують хронологію взаємодії.

Діаграма послідовності допомагає зрозуміти логіку взаємодії компонентів системи та послідовність дій, які відбуваються під час виконання певної функціональності. Також вона може бути використана для виявлення можливих проблем або неузгодженостей у взаємодії компонентів та допомагає покращити розуміння системи в цілому (див. рис. 2.3).

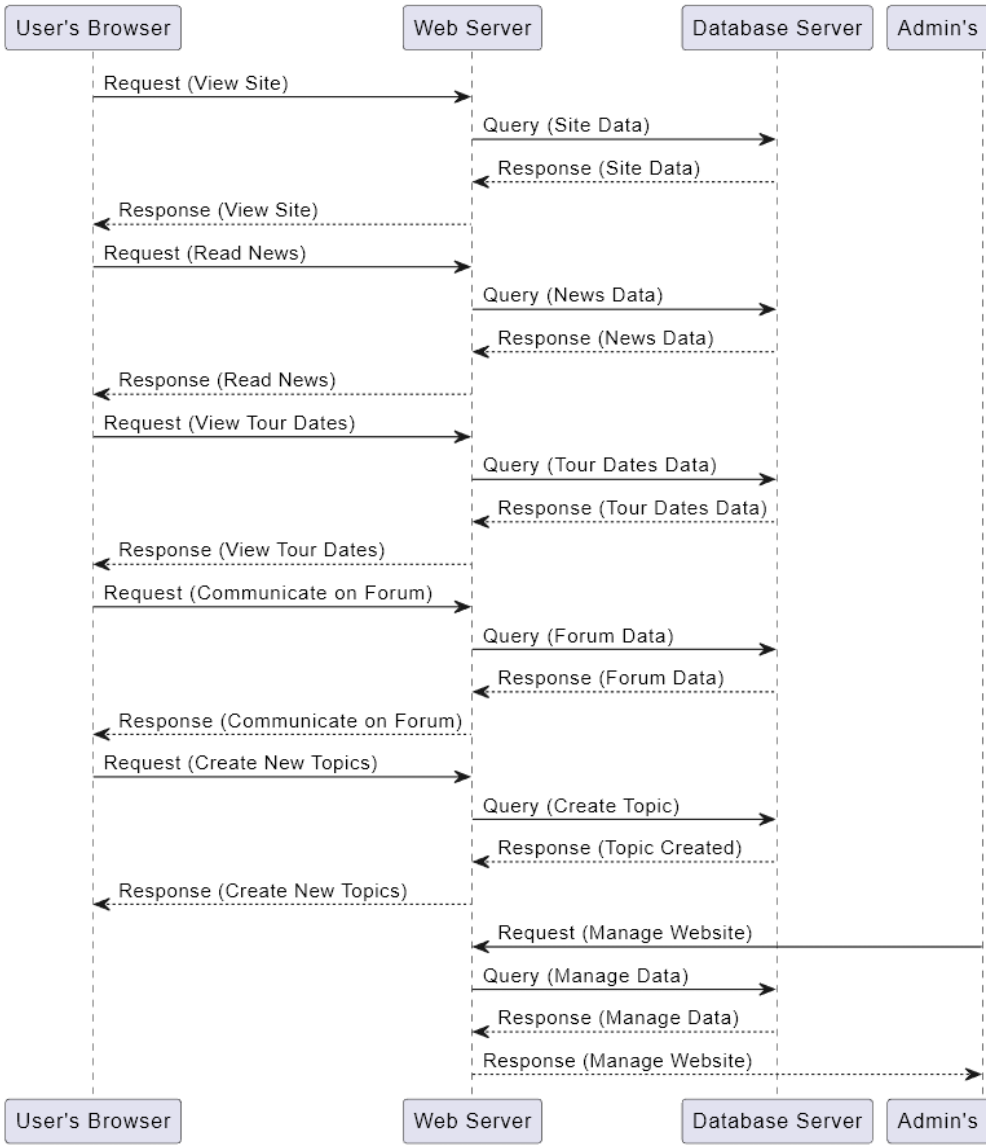

Рисунок 2.3 – Діаграма послідовності

2.2.4 Діаграма компонентів

Діаграма компонентів є важливим інструментом в розробці вебсайтів. Вона використовується для візуального зображення структури системи та взаємозв'язків між компонентами. Діаграма компонентів відображає компоненти системи, їх залежності та інтерфейси.

Компоненти – це функціональні блоки або модулі системи. Кожен компонент виконує певні функції і може мати внутрішню реалізацію. Залежності між компонентами вказують, як компоненти взаємодіють між собою. Це можуть бути залежності використання, коли один компонент використовує інший для виконання певних дій або отримання інформації. Інтерфейси визначають спосіб взаємодії з компонентами, включаючи методи, параметри та повернені значення.

Діаграма компонентів допомагає в розробці вебсайтів зрозуміти структуру системи, функціональні блоки та взаємозв'язки між компонентами, допомагає планувати архітектуру системи, виявляти проблеми та залежності, а також полегшує відлагодження та підтримку системи.

Діаграма компонентів є важливим інструментом для комунікації між розробниками. Вона дозволяє краще розуміти структуру системи та сприяє ефективній розробці вебсайту (див. рис. 2.4).

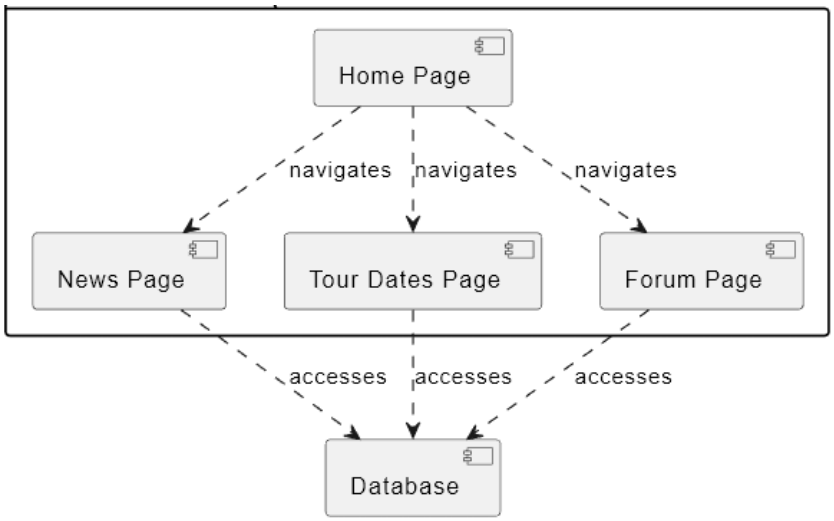

Рисунок 2.4 – Діаграма компонентів

3 РОЗРОБКА

3.1 Характеристика потенційної аудиторії проєкту

Потенційна аудиторія проєкту, присвяченого рок-групі, може бути досить широкою і різноманітною, оскільки рок-музика відома своєю універсальністю та масовою популярністю серед людей різного віку, культури та інтересів.

3.2 Вибір засобів розробки

Серед можливих засобів розробки були обрані наступні програмні продукти: інтегроване середовище розробки Visual Studio Code від компанії Microsoft, потужний фреймоврк Next.js, а для серверної частини був обраний Node.js [9].

Visual Studio Code (VS Code) – це відкритий редактор коду, розроблений компанією Microsoft, який включає в себе такі особливості, як підсвічування синтаксису, інтелектуальне автозавершення коду, перегляд коду, Git-інтеграція, дебаггінг та вмонтовану термінал, що дозволяє виконувати команди без виходу з редактора.

В моїй дипломній роботі VS Code стане незамінним інструментом розробки, особливо при роботі з такими технологіями, як Node.js, Yarn та Next.js. Він має вбудовану підтримку JavaScript та TypeScript, а також розширення для підтримки React та Node.js.

VS Code також включає в себе можливість встановлення розширень, що серйозно розширюють його функціональність. Деякі розширення, які можуть бути корисними, включають Prettier для автоматичного форматування коду, ESLint для виявлення та виправлення проблем із кодом, а також різні розширення для підсвічування синтаксису, автозавершення коду та навігації.

Ще однією корисною особливістю VS Code є його інтеграція з Git. Можна легко перевіряти статус свого репозиторію, створювати коміти, виконувати pull та push, а також переглядати історію комітів без виходу з редактора.

Останнім, але не менш важливим, є термінал VS Code. Він дозволяє виконувати команди для Yarn та Node.js без виходу з редактора. Це особливо корисно, адже потрібно буде запустити сервер розробки, виконати тести або встановити нові пакети.

Next.js – це JavaScript фреймворк, розроблений на базі React, що підтримує як клієнтський, так і серверний рендеринг. Він був створений компанією Vercel і є відкритим програмним забезпеченням.

Next.js – це одна з передових технологій у світі веб-розробки, яка стала популярною через свої унікальні характеристики, такі як гібридний статичний і серверний рендеринг, автоматичний роутинг на основі файлової системи, вбудована оптимізація зображень та підтримка API-роутів. Завдяки Next.js, веб-розробники здатні створювати складні, добре структуровані веб-застосунки на React з підтримкою серверного рендерингу і генерації статичних сайтів.

Важливою властивістю Next.js є його система роутингу на основі файлової системи, яка автоматично визначає маршрути на основі імен файлів у папці проєкту. Це дозволяє з легкістю створювати нові сторінки та шляхи до них, просто додаючи нові файли до цієї папки.

Вбудована оптимізація зображень забезпечує автоматичну оптимізацію зображень на сервері, що дозволяє поліпшити продуктивність веб-сайту, оскільки зображення є найбільш великими ресурсами, які завантажуються на веб-сторінках.

Yarn – це потужний менеджер пакетів, який допомагає автоматизувати і контролювати процес встановлення, оновлення, конфігурації та видалення JavaScript пакетів (бібліотек) в проєкті.

Як відомо, сучасні вебсайти і вебдодатки часто використовують різноманітні JavaScript-бібліотеки і фреймворки, щоб підтримувати різні функції, такі як анімація, AJAX-запити, маршрутизація сторінок, використання

компонентів і так далі. Yarn дозволяє розробникам встановлювати, оновлювати, налаштовувати і видаляти ці бібліотеки ефективно. Він також гарантує, що всі розробники, які працюють над проєктом, використовують одну й ту ж версію кожного пакету, що допомагає уникнути проблем, пов'язаних з несумісністю версій.

3.3 Розробка програмного застосунку

Для того, щоб безпосередньо перейти до розробки треба підготувати все необхідне. Розглянемо цей процес покроково.

Спочатку потрібно завантажити та встановити Visual Studio Code з офіційного сайту (див. рис. 3.1).

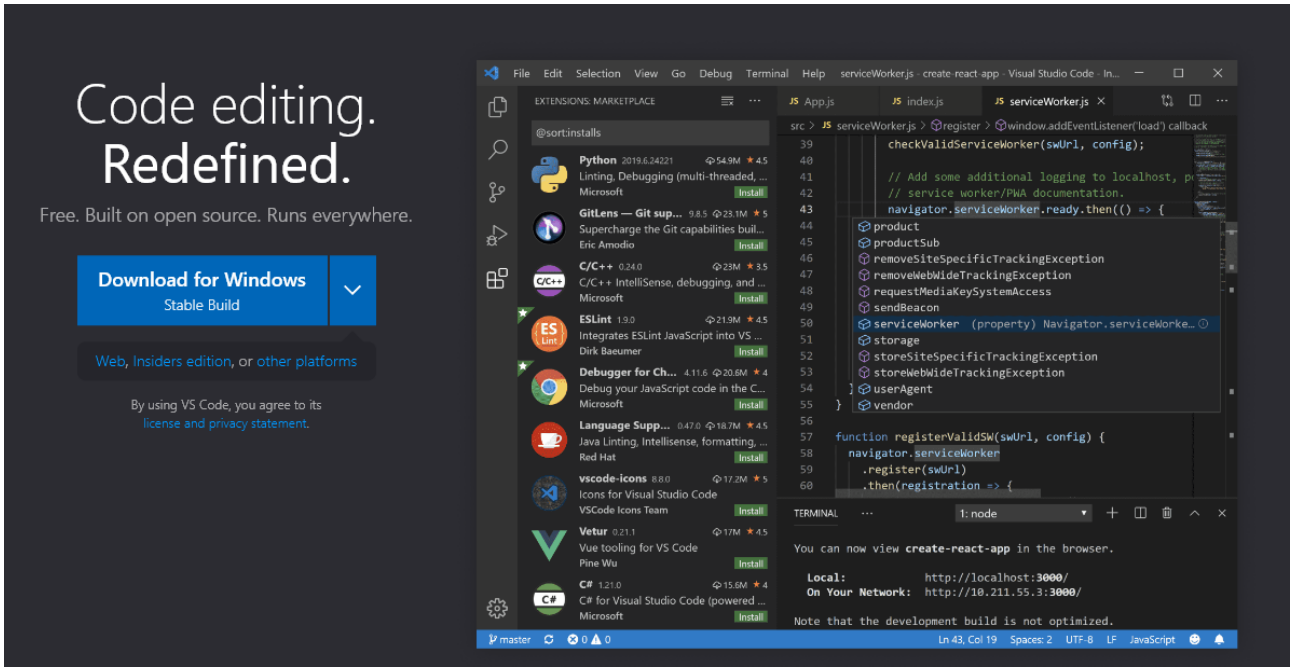

Рисунок 3.1 – Сторінка завантаження Visual Studio Code

Далі необхідно завантажити та встановити Node.js (див. рис. 3.2).

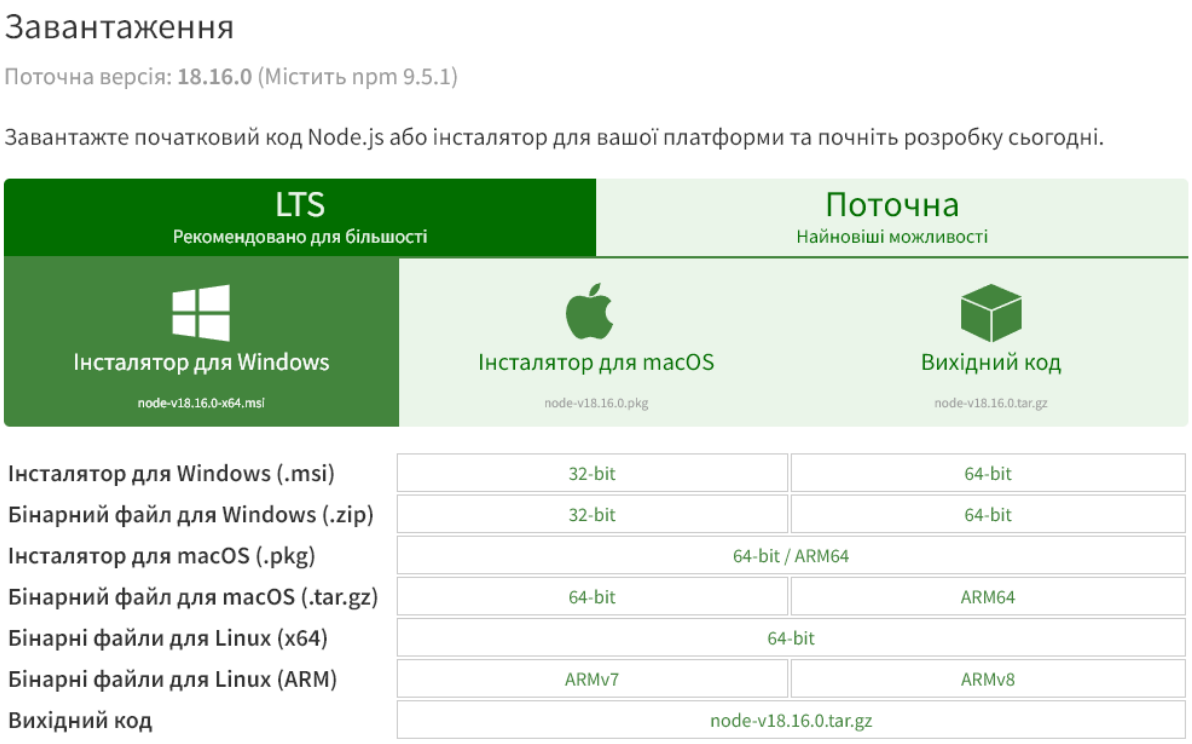

Рисунок 3.2 – Сторінка завантаження Node.js

Встановити менеджер пакетів Yarn (див. рис. 3.3).

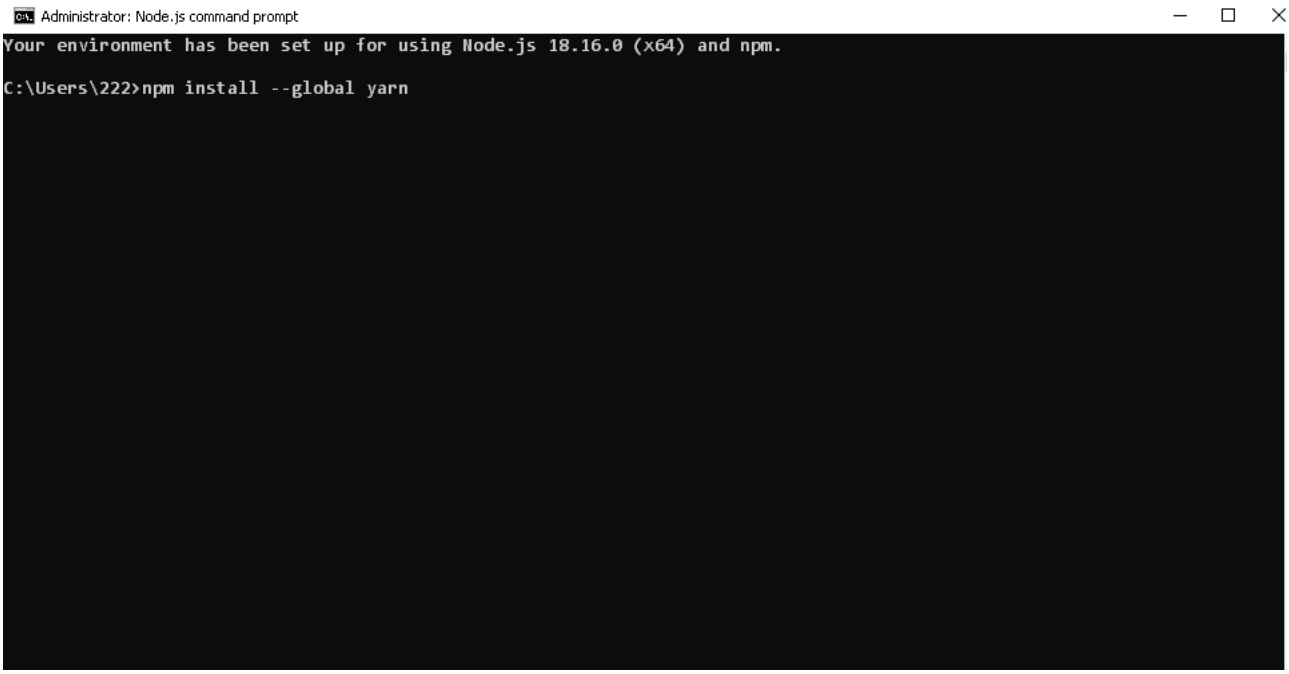

Рисунок 3.3 – Встановлення Yarn в системі через npm

Встановлення Next.js. Для цього потрібно мати встановленим на комп'ютері Node.js, відкрити командний рядок і ввести наступну команду: «yarn create next-app» та вказати назву проєкту. Ця команда встановить та налаштує всі необхідні залежності для створення базового проєкту Next.js (див. рис. 3.4).

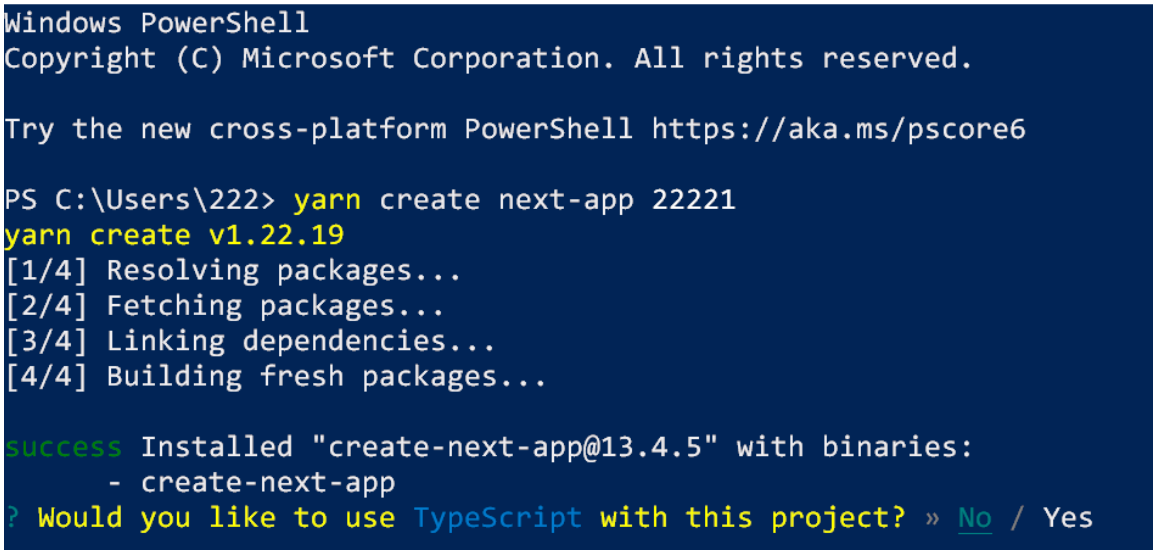

Рисунок 3.4 – Створення проєкту через командний рядок

Команда «yarn create next-app» використовує пакет «create-next-app» для створення нового проєкту на Next.js. Це спеціальний пакет, який розробники додали в Next.js, щоб полегшити і автоматизувати процес створення нового проєкту. Спочатку команда перевіряє чи встановлений пакет create-next-app чи ні. Якщо пакет вже встановлено, команда переходить до наступного етапу, в іншому випадку вона встановлює create-next-app в системі.

На наступному етапі «yarn create next-app» використовує «create-next-app» для генерації нового проєкту. Вона створює нову директорію з назвою, яку користувач вказав, і наповнює її шаблоном проєкту Next.js.

Шаблон проєкту містить ряд конфігураційних файлів, які необхідні для роботи проєкту на Next.js. Вони включають файли package.json і next.config.js, які визначають, які пакети та налаштування будуть використовуватися в проєкті. Також в шаблоні проєкту присутні декілька базових сторінок, які

демонструють, як створювати маршрутизацію в Next.js.

Встановлення Next.js завершилось. Переходимо в директорію проєкту, щоб оцінити його структуру (див. рис. 3.5). Для цього ввів наступну команду: «code .».

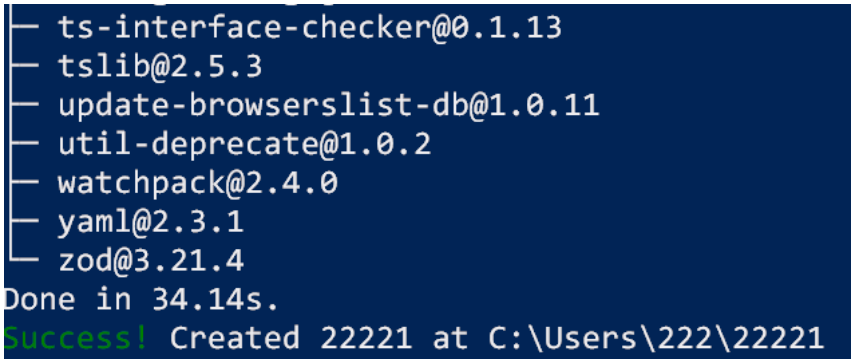

Рисунок 3.5 – Повідомлення про успішну збірку проєкта

Верстка в Next.js починається зі створення компонентів. Компоненти представляють собою незалежні блоки коду, які мають свою розмітку (JSX), стилі (CSS) та функціонал (JavaScript). Компоненти можна використовувати для побудови сторінок, макетів і різних частин веб-додатка.

JSX – це розширений синтаксис JavaScript, який дозволяє змішувати HTML-подібний код з JavaScript у компонентах. JSX дозволяє описувати структуру компонента, включаючи елементи, атрибути, вкладені елементи і текстовий контент.

Створення компонентів відбувається шляхом створення файлів з розширенням .js, .jsx, .ts або .tsx у папці pages або у власних папках, які можуть знаходитися в інших директоріях проєкту. Кожен файл компонента містить JSX-розмітку, в якій описується його структура та також логіка його роботи.

Після створення компонентів їх можна використовувати для побудови сторінок за допомогою маршрутизації. У Next.js маршрутизацію можна налаштувати за допомогою файлів з розширенням .js, .jsx, .ts або .tsx у папці pages. Кожен файл у папці pages представляє собою окрему сторінку, і його URL-шлях визначається за його розташуванням у структурі папок.

Після створення компонентів і налаштування маршрутизації можна розпочати верстку сторінок, додавати стилі до компонентів і використовувати їх для відображення вмісту на сторінках.

Важливо також зазначити, що в Next.js для верстки часто використовуються бібліотеки стилів, такі як CSS-модулі, styled-components, Tailwind CSS і багато інших, які дозволяють зручно організувати стилі компонентів та їх повторне використання.

Починаю з компонентів: меню, футер, хедер та додатково напишу компонент Button.

Компонент Menu буде відображати меню сайту з елементами, які створюються на основі даних в data.js (див. рис. 3.6).

```
import React from "react";
import { Item, ItemsWrapper, Logo, Wrapper } from "./views";
import { data } from "../Footer/data";
import Link from "next/link";
const Menu = ({ setMenuVisible }) => {
return (
<Wrapper>
<ItemsWrapper>
{data.map((el) => (
<>
<Link href={el.href} onClick={() => setMenuVisible(false)}>
<Item>{el.text}</Item>
</Link>
</>
))}
</ItemsWrapper>
<Logo src="/images/Menu_Logo.svg" />
</Wrapper>
);
};
export default Menu;
```

Рисунок 3.6 – Код компонента Menu

Масив data містить в собі об’єкти з інформацією про пункти меню. Кожен об'єкт представляє один пункт меню і містить текст пункту і href – посилання,

на яке здійснюватиметься перехід при кліці на пункт меню (див. рис. 3.7).

```
export const data = [
{ text: "Головна", href: "/main" },
{ text: "Історія группи", href: "/history" },
{ text: "Новини", href: "/news" },
{ text: "Дискографія", href: "/discography" },
{ text: "Тури та концерти", href: "/concerts" },
];
```

Рисунок 3.7 – Код масиву data

Підвал (footer) – це розділ вебсторінки, який зазвичай знаходиться внизу сторінки. Він містить додаткову інформацію, посилання, контактні дані, копірайт та іншу допоміжну інформацію, пов'язану з вебсайтом. За моїм дизайном там має буде логотип, невеличке меню та кнопка, яка виконує функцію прокручування сторінки до верхньої позначки вебсайту для зручності користувачів (див. рис. 3.8).

Рисунок 3.8 – Приклад підвалу

Цей компонент представляє підвал (footer) вебсторінки і відображає список елементів, логотип і кнопку в підвалі.

Також, в підвалі можуть бути розміщені посилання на соціальні мережі та контактні дані (див. рис. 3.9).

Компонент Button зберігає в собі розмітку JSX для створення кнопки і надає різні варіанти відображення на основі переданих параметрів. Таких як: текст, посилання та варіанти відображення.

Крім передачі параметрів, які визначають текст кнопки та посилання, варіант відображення у компоненті Button дає можливість налаштувати стиль або зовнішній вигляд кнопки залежно від переданого значення.

```
import React from "react";
import { Button, Item, Items, Logo, Wrapper } from "./views";
import { data } from "./data";
import Link from "next/link";
const Footer = () => {
return (
<Wrapper>
<Items>
{data.map((el) => (
<>
<Link href={el.href}>
<Item>{el.text}</Item>
</Link>
</>
))}
</Items>
<Link href="/">
<Logo src="/images/Logo.png" />
</Link>
<Button src="/images/Btn_To_Top.svg" />
</Wrapper>
);
};
export default Footer;
```

Рисунок 3.9 – Код компонент підвалу

Варіант відображення у компоненті Button являє собою додатковий параметр, який дає змогу налаштувати стиль або зовнішній вигляд кнопки залежно від переданого значення (див. рис. 3.10).

```
import React from "react";
import { Text, Wrapper } from "./views";
import Link from "next/link";
const Button = ({ text, href, variant2 }) => {
return (
<>
{href ? (
<Link href={href}>
<Wrapper>
<Text>{text}</Text>
</Wrapper>
</Link>
) : (
<Wrapper variant2={variant2}>
<Text>{text}</Text>
</Wrapper>
)}
</>
);
};
export default Button;
```

Рисунок 3.10 – Код компоненту кнопки

У підсумку, компонент Button надає гнучкість у створенні кнопок із різними текстами, посиланнями та варіантами відображення.

Від створення компонентів я перейшов до їх правильного розташування на сторінці використовуючи бібліотеку «styled-components» для створення стилізованих компонентів (див. рис. 3.11).

```
import styled from "styled-components";
export const Wrapper = styled.div`
  margin-top: 121px;
  height: 300px;
  padding: 0 240px;
  display: flex;
  justify-content: space-between;
  align-items: center;
`;
export const Logo = styled.img`
  width: 680px;
  height: 140px;
  margin-right: 110px;
  cursor: pointer;
`;
export const Button = styled.img``;
export const Item = styled.div`
  font-size: 22px;
  line-height: 27px;
  color: white;
  transition: 0.3s;
  cursor: pointer;
  &:hover {
    color: #c70b19;
`;
export const Items = styled.div`
  display: flex;
  flex-direction: column;
  align-items: start;
`;
```

Рисунок 3.11 – Стилізований компонент підвалу

Для коректного відображення стилів для компонентів в папці під назвою «styles» створив файл з назвою «global.css».

Ці стилі будуть застосовані до відповідних елементів у проєкті, щоб змінити їхній зовнішній вигляд і поведінку відповідно до заданих значень змінних і правил стилів (див. рис. 3.12).

```
:root {
--max-width: 1100px;
--border-radius: 12px;
--font-mono: ui-monospace, Menlo, Monaco, "Cascadia Mono", "Segoe UI Mono","Roboto Mono", "Oxygen Mono", "Ubuntu Monospace", "Source Code Pro", "Fira Mono", "Droid Sans Mono", "Courier New", monospace;
}
@font-face {
font-family: "centurygothic";
src: url("../../public/fonts/centurygothic.ttf") format("truetype");
}
html {
font-size: 62.5%;
}
body {
max-width: 100vw;
overflow-x: hidden;
background-color: black;
margin: 0;
color: white;
font-family: "centurygothic";
color: inherit;
text-decoration: none;
}
@media (prefers-color-scheme: dark) {
html {
color-scheme: dark;
```

Рисунок 3.12 – Глобальні стилі

У деяких користувачів є проблема, пов’язана з мерехтінням стилів при використанні бібліотеки «styled-components». Вона полягає в тому, що інколи при серверному рендерингу сторінок стилі компонентів не включаються до HTML-коду, що призводить до того, що стилі просто не застосовуються або до їх некоректного відображення при першому завантаженні сторінки. Це негативно впливає на користувацький досвід та робить сайт зовсім не

привабливим.

Для подолання цієї проблеми потрібно використати клас ServerStyleSheet, який надається styled-components і використовується для отримання стилів компонентів під час серверного рендерингу. Наступний код дозволяє збирати стилі компонентів styled-components під час серверного рендерингу і додати їх до серверного HTML-коду (див. рис. 3.13).

```
try {
ctx.renderPage = () =>
originalRenderPage({
enhanceApp: (App) => (props) =>
sheet.collectStyles(<App {...props} />),
});
const initialProps = await Document.getInitialProps(ctx);
return {
...initialProps,
styles: (
<>
{initialProps.styles}
{sheet.getStyleElement()}
</>
),
};
} finally {
sheet.seal();
}
```

Рисунок 3.13 – Частина коду для налаштування серверного рендерингу

Таким чином, навіть якщо JavaScript у браузері буде вимкненим користувач все одно зможете побачити стилі, застосовані до заголовків і контейнерів.

Далі була створена окрема папка для зберігання інформації про артистів, концерти, дискографію та новини.

Створив компонент MainPage, який буде головною сторінкою сайту. Компонент MainPage – це функціональний компонент, який відображає вміст головної сторінки. Він містить розмітку сторінки, яка містить різні секції та

компоненти: банер, учасники гурту, історія гурту, новини, дискографія та тури і концерти. Кожна секція використовує відповідні компоненти та дані для відображення інформації. Для кожної секції використовується цикл map для відображення елементів на основі даних із відповідного масиву. Для деяких секцій, додав кнопки за допомогою компонента Button, який я описував вище. Кнопки дають змогу перейти на відповідні сторінки.

Також хук «useSession» був створений для перевірки наявності сеансу користувача. Якщо сеанс відсутній – компонент MainPage повертає компонент Login за допомогою оператора return – замість відображення решти коду компонента MainPage, буде відображено лише компонент Login (див. рис. 3.14).

```
import {
  ArtistsContainer, Banner, ButtonStyle, ConcertsContainer, DiscoContainer Left,
MainContainer, NewsContainer, Picture, Text, TextWImageContainer, Title, Wrapper,
} from "./views";
import Artist from "./Artist/Artist";
import { concertsData, discoData, mainData, newsData } from "./data";
import Button from "@/components/Button";
import NewsCard from "./News";
import DiscoCard from "./Dyscography";
import ConcertItem from "./Concerts";
import { ReactMarkdown } from "react-markdown/lib/react-markdown";
import { useSession } from "next-auth/react";
import { useRouter } from "next/router";
import Login from "../login";
const MainPage = () => {
  const { data: session } = useSession();
  if (!session) {
    return <Login />;
```

Рисунок 3.14 – Частина коду головної сторінки

Авторизація є важливою складовою будь-якого вебсайту, що дозволяє користувачам увійти до свого облікового запису та отримати доступ до обмеженого функціоналу.

Щоб реєстрація та авторизація на сайті працювали, необхідно виконати наступні кроки:

– встановити пакет NextAuth.js;

– встановити MongoDB, а саме драйвер для нього.

Для цього в термінал vsCode вводимо команду «yarn add next-auth» для пакету NextAuth.js та «yarn add mongodb» для MongoDB. Для реалізації авторизації були використані наступні компоненти.

FormWrapper – контейнер, який охоплює всю форму входу і реєстрації. TabsContainer – контейнер, що містить вкладки для переключення між формою входу та реєстрації. Tab – вкладка, що представляє форму входу або реєстрації, в залежності від активної вкладки (див. рис. 3.15).

```
<FormWrapper variant={activeTab === 2}>
<TabsContainer>
<Tab
active={activeTab === 0 || activeTab === 2}
onClick={() => setActiveTab(0)}
Увійти на сайт
</Tab>
<Tab active={activeTab === 1} onClick={() => setActiveTab(1)}>
Реєстрація
</Tab>
</TabsContainer>
```

Рисунок 3.15 – Приклад коду активної вкладки

Title – заголовок, який відображає поточний режим вхіду, реєстрації або відновлення пароля (див. рис. 3.16).

```
<Title>
{activeTab === 0
? "Вкажіть дані для входу на сайт"
: activeTab === 1
? "Вкажіть дані для реєстрації на сайті"
: "Відновлення пароля"}
</Title>
```

Рисунок 3.16 – Title

InputsWrapper – контейнер, що містить поля вводу для електронної пошти та пароля. InputLog – поле вводу для електронної пошти або пароля. Button –

кнопка для виконання дії (увійти, зареєструватися або відновити). Text – текстовий блок з додатковою інформацією або поясненнями. Імпорт інших додаткових компонентів, таких як axios, loginUser і useRouter, для взаємодії з сервером та маршрутизації.

Спільно ці компоненти створюють форму входу та реєстрації з відповідною валідацією даних та обробкою подій для виконання відповідних дій (увійти, зареєструватися або відновити пароль). Авторизація здійснюється через відправку запитів на сервер. При натисканні на кнопку «Увійти», дані форми входу відправляються на сервер, де проводиться перевірка валідності користувача.

Функція validateData, яка перевіряє дані форми на валідність, включаючи перевірку довжини електронної пошти, правильності формату та довжини пароля. Після успішної перевірки валідності користувача, здійснюється виконання відповідних дій, таких як авторизація, реєстрація або відновлення пароля, залежно від обраної опції. Функція також оновлює стан помилок валідації (див. рис. 3.17).

```
A const validateData = () => {
const err = [];
if (data.email?.length < 4) {
err.push("e-mail Повинен бути довшим за 4 символи");
} else if (data.email?.length > 30) {
err.push("e-mail має бути коротшим за 30 символів");
} else if (!regEmail.test(data.email)) {
err.push("Некоректный e-mail");
} else if (data.password?.length < 6) {
err.push("Пароль повинен містити більше 6 символів");
}
setValidatorsError(err);
if (err.length > 0) {
return false;
} else {
return true;
}
};
```

Рисунок 3.17 – Функція validateData

Функція handleOnchange, яка оновлює стан даних форми при зміні введених значень (див. рис. 3.18).

```
const handleOnchange = (e) => {
setData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
```

Рисунок 3.18 – Функція handleOnchange

Функція handleSignup, яка виконує реєстрацію користувача. Вона викликає API-запит до сервера з введеними даними, перевіряє результат та зберігає дані сеансу використовуючи функцію loginUser (див. рис. 3.19).

```
const handleSignup = async () => {
if (validateData()) {
setLoading(true);
try {
const apiRes = await axios.post(
"http://localhost:3000/api/auth/signup",
data);
if (apiRes?.data?.success) {
const loginRes = await loginUser({
email: data.email,
password: data.password,
});
if (loginRes && !loginRes.ok) {
setSubmitError("");
} else {
setActiveTab(0);
}}} catch (error) {}
setLoading(false);
};
```

Рисунок 3.19 – Функція handleSignup

Функція handleLogin, яка здійснює вхід користувача. Вона викликає loginUser з введеними даними, перевіряє результат та перенаправляє користувача на головну сторінку (router.push("/main")), якщо вхід успішний (див. рис. 3.20).

```
const [activeTab, setActiveTab] = useState(0);
const [submitError, setSubmitError] = useState([]);
const [loading, setLoading] = useState(false);
const [validatorsError, setValidatorsError] = useState([]);
const [data, setData] = useState({
email: "",
password: "",
const { data: session } = useSession();
const router = useRouter();
const regEmail = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
```

Рисунок 3.20 – Функція handleLogin

На основі дій наведених вище отримав наступний результат (див. рис. 3.21).

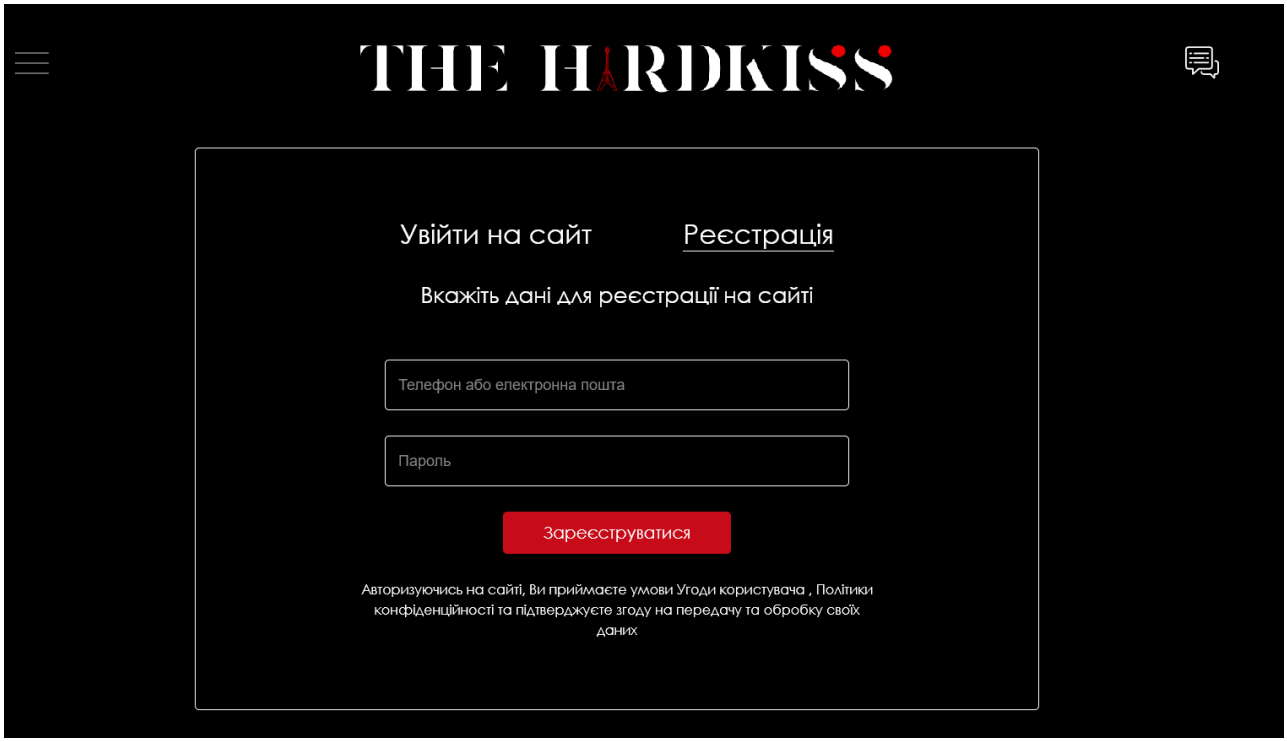

Рисунок 3.21 – Сторінка авторизації та реєстрації

Наступним кроком було зареєструватись на MongoDB та налаштувати з'єднання з базою даних. З'єднання з базою даних необхідне для збереження користувачів та виконання операцій з даними.

MongoDB – це документоорієнтована база даних. Після реєстрації створив новий проєкт, вказав його назву, вона ж буде назвою бази даних. В

MongoDB пропонували можливість додати деяких користувачів до проєкту, але я один, тому цей пункт було пропущено. Далі натиснув на кнопку «Bild Cluster» та обрав безкоштовний. Вибрав максимально близько розташований до України регіон та, нарешті, створив кластер. Для остаточного створення та доступу до нього потрібен певний час. Додав до проєкту в vsCode залежності:

- axios – клієнт для виконання HTTP-запитів;
- bcryptjs – бібліотека для хешування та перевірки паролів;
- mongoose – спеціальна бібліотека для спрощення роботи з базою даних Mongo, так званий, Object-Document Mapper – аналог Object-Relational Mapping для документоорієнтованої бази даних [7].

Повернувся в package.json та створив скрипт, який буде відповідати за старт серверу, використовував команду «start» , всередині прописав «next start» – це потрібно для того, щоб кожен раз при зміні файлів сервер перезапускався, простіше кажучи, створив dev server – сервер для розробки.

Для підключення до MongoDB створив файл .env у кореневому каталозі проєкту та додав URL-адресу підключення до MongoDB Atlas – повноцінного хмарного сервісу бази даних MongoDB. Загалом, цей URL-адрес містить інформацію про сервер, ім'я користувача, пароль та параметри підключення, необхідні для з'єднання з базою даних MongoDB.

Також додав згенерований секретний ключ для підпису токенів та збереження сесій в NextAuth.js. Секретний ключ використовується для забезпечення безпеки та ідентифікації токенів, що передаються між сервером та клієнтом. Це важливий елемент в процесі аутентифікації та авторизації, оскільки забезпечує конфіденційність і цілісність даних.

Цей ключ важливо тримати в секретності і не розголошувати його публічно. Він служить для захисту від зловживання та несанкціонованого доступу. У коді він буде використовуватись для генерації та перевірки підпису токенів, що передаються між клієнтом та сервером для забезпечення безпеки та цілісності даних.

Після підключення бази даних потрібно забезпечити механізм

аутентифікації на основі облікових даних користувача, переданих через поля «Email» та «Password» у формі входу. Вирішив додати хешування паролів для користувачів. Для цього використав функцію hash з завчасно встановленого пакету bcryptjs.

Хешування паролю – це процес перетворення введеного паролю на незворотнє значення (хеш). Хеш – це фіксована довжина символів, яка представляє унікальне представлення вихідного паролю. В моєму випадку встановив кількість «солі» на значення 12. Встановлювати вищі значення не рекомендується, так як хешування буде тривати занадто довго.

Сіль (salt) – це випадкове значення, яке додається до пароля перед хешуванням. Використовується для збільшення унікальності та складності хеша пароля. Застосування солі в хешуванні паролів вносить додатковий рівень безпеки (див. рис. 3.22).

```
const hashedPassword = await hash(password, 12);
```

Рисунок 3.22 – Хешування паролю

Також використав JWT (JSON Web Token) для забезпечення механізму аутентифікації та авторизації [8]. Після успішної аутентифікації користувача буде створюєтись JWT-токен, який містить інформацію про користувача і підписаний секретним ключем. Цей токен передається між клієнтом і сервером для підтвердження ідентичності користувача при кожному запиті. Код має мати функцію зворотного виклику jwt, яка відповідає за маніпуляцію JWT-токенами. У цьому коді вона отримує токен та користувача як аргументи. Функція перевіряє, чи присутній користувач, і якщо так, то додає його до токену. Після цього вона повертає оновлений токен (див. рис. 3.23).

```
session: {
strategy: "jwt",
callbacks: {
jwt: async ({ token, user }) => {
user && (token.user = user);
return token;
session: async ({ session, token }) => {
const user = token.user;
session.user = user;
return session;
},
```

Рисунок 3.23 – Автентифікація за допомогою JSON Web Tokens

На основі коду наведеного вище в базу даних при реєстрації нового користувача записуються наступні параметри:

– _id – унікальний ідентифікатор користувача;
– email – електронна пошта користувача;
– password – захешований пароль користувача;
– __v – версія документа. Це внутрішнє поле, яке автоматично додається до кожного документа в MongoDB і використовується для керування версіями даних (див. рис. 3.24).

Рисунок 3.24 – Приклад запису про нового користувача

Перейшовши до розробки форуму зосередився на створенні модальних вікон для додавання нових тем та відображенні коментарів користувачів.

Використовуючи бібліотеку «next-auth», створив безпечну сесію, яка використовується для ідентифікації автора при створенні нових тем на форумі. Це важливий аспект безпеки, оскільки він допомагає запобігти зловживанням та дозволяє користувачам відслідковувати свої власні публікації (див. рис. 3.25).

```
const { data: session } = useSession();
const [data, setData] = useState({
   themename: "",
   text: "contentState",
   author: session.user.email,
   comments: [],
});
```

Рисунок 3.25 – Приклад коду ідентифікації автора

Використовуючи бібліотеку «react», а конкретно такі хуки як useState та useSession створив інтерактивний користувацький інтерфейс, що реагує на дії користувачів в реальному часі. Крім вищезгаданих хуків, також можна використовувати інші хуки з бібліотеки React для реалізації різноманітних функціональностей. Наприклад, хук useEffect дозволяє виконувати певний код при зміні певних змінних або при завантаженні компоненту. Це дозволяє реагувати на зміни стану та взаємодіяти з сервером або зовнішніми даними. Таким чином, з використанням хуків React можна створювати динамічні та інтерактивні інтерфейси, які змінюються в реальному часі залежно від дій користувачів (див. рис. 3.26).

```
const [editorState, setEditorState] = useState(EditorState.createEmpty());
const [contentState, setContentState] = useState(null);
const [loading, setLoading] = useState(false);
const [data, setData] = useState({
themename: "",
text: "contentState",
author: session.user.email,
comments: [],
});
```

Рисунок 3.26 – Приклад реалізації хуків

Продовжую використовувати бібліотеку «styled-components» для створення візуально привабливого дизайну з урахуванням UX-принципів. Раніше написані модулі стилів, такі як Wrapper, AuthorContainer, AuthorInfoContainer та інші допомогли створити чіткі та інтуїтивно зрозумілі

компоненти інтерфейсу (див. рис. 3.27).

```
import styled from "styled-components";
export const Wrapper = styled.div
export const AuthorContainer = styled.div
export const AuthorInfoContainer = styled.div
```

Рисунок 3.27 – Компоненти інтерфейсу

Використовуючи бібліотеку «react-draft-wysiwyg» забезпечив потужний текстовий редактор у формі для додавання тем, що дозволить користувачам форматувати свої повідомлення. Бібліотека «react-draft-wysiwyg» надає розширені можливості для роботи з текстом, що дозволяє користувачам використовувати різноманітні форматування, такі як жирний шрифт, курсив, підкреслення, списки, вирівнювання тексту та багато інших. Також цей текстовий редактор підтримує можливість вставки зображень, таблиць та посилань, що дозволяє користувачам створювати багатофункціональні та естетично привабливі повідомлення. Використання «react-draft-wysiwyg» спрощує процес створення та редагування тексту, роблячи його зручним (див. рис. 3.28).

```
import { EditorState } from "draft-js";
import dynamic from "next/dynamic";
const Editor = dynamic(
() => import("react-draft-wysiwyg").then((mod) => mod.Editor),
{ ssr: false }
import "react-draft-wysiwyg/dist/react-draft-wysiwyg.css";
const [editorState, setEditorState] = useState(EditorState.createEmpty());
const handleEditorStateChange = (editorState) => {
setEditorState(editorState);
};
```

Рисунок 3.28 – Частина коду текстового редактору

Використовуючи «next/dynamic» створив динамічний імпорт редактора «react-draft-wysiwyg», який забезпечує оптимізацію продуктивності шляхом відкладеного завантаження коду редактора. Редактор завантажується тільки

тоді, коли він дійсно потрібний (див. рис. 3.29).

```
import dynamic from "next/dynamic";
const Editor = dynamic(
() => import("react-draft-wysiwyg").then((mod) => mod.Editor),
{ ssr: false }
);
```

Рисунок 3.29 – Частина коду текстового редактору

Впровадив використання «axios» для виконання запитів до нашого сервера, що дозволяє створювати, відображати та керувати темами на форумі в ефективний та безпечний спосіб. Пакет «axios» використовується для виконання HTTP-запитів, що дозволяє зберігати та отримувати дані теми. Це важливий інструмент для забезпечення відповідного взаємодії з сервером (див. рис. 3.30).

```
const addTheme = () => {
setLoading(true);
try {
axios.post("http://localhost:3000/api/query/set_themes", data);
setModal(false);
} catch (error) {}
setLoading(false);
update();
};
```

Рисунок 3.30 – Виконання запитів

Розробив можливість відповіді на кожен коментар за допомогою «CommentItem», який відображає аватар користувача, ім'я автора, час публікації та текст коментаря (див. рис. 3.31).

Можливість відповідати на коментар сприяє покращенню взаємодії між користувачами, створюючи зручну платформу для обговорень та обміну думками.

Завдяки цим новим можливостям, користувачі зможуть більш ефективно комунікувати один з одним, що сприятиме покращенню якості взаємодії та

залучення більшої аудиторії. Люди можуть легко обмінюватися думками, ідеями, встановлювати контакти з новими людьми, знаходити спільні інтереси і формувати групи з подібними інтересами.

```
const CommentItem = ({ author, text }) => {
return (
<Wrapper>
<AuthorContainer>
<AvatarXS src="/images/Avatar.png" />
<AuthorInfoContainer>
<AuthorName>{author}</AuthorName>
<AuthorTime>нещодавно</AuthorTime>
</AuthorInfoContainer>
</AuthorContainer>
<TextArea dangerouslySetInnerHTML={{ __html: text }}></TextArea>
</Wrapper>
);
};
```

Рисунок 3.31 – Частина коду, що використовується для відповідей

Використав «dangerouslySetInnerHTML» в компоненті «CommentItem» для вставки HTML-коду коментаря безпосередньо в DOM.

DOM (Document Object Model) - це спосіб представлення вебсторінки у вигляді структурованого дерева об'єктів, де кожен об'єкт відповідає своїй частині сторінки.

Коли вебсторінка відкривається, браузер отримує HTML-документ і перетворює його в DOM. Ним можна маніпулювати за допомогою JavaScript, що дозволяє динамічно змінювати вміст і структуру веб-сторінки (див. рис. 3.32).

```
<TextArea dangerouslySetInnerHTML={{ __html: text }}></TextArea>
```

Рисунок 3.32 – Виконання вставки коментаря

Розробив обробник серверного запиту в Next.js, який використовується для встановлення з'єднання з базою даних MongoDB і виконання операцій з

даними. У випадку отримання POST-запиту він бере id з тіла запиту, встановлює з'єднання з MongoDB, знаходить теми в базі даних з цим id і повертає їх у відповідь на запит. Якщо стається якась помилка, він повертає статус 400 із повідомленням про помилку (див. рис. 3.33).

```
connectToMongoDB().catch((err) => res.json(err));
if (req.method === "POST") {
const { id } = req.body;
try {
await connectToMongoDB();
const themes = await Theme.find({ _id: id });
res.status(200).send(themes);
} catch (error) {
res.status(400).send({ error, msg: "something wrong" });
};
```

Рисунок 3.33 – Функція connectToMongoDB

Додав код, що обробляє GET-запити на сервері в Next.js, витягуючи всі теми з MongoDB. Коли GET-запити починають відправлятись код спробує знайти всі записи в колекції Theme в MongoDB та відправити ці записи як відповідь на запит. Якщо під час виконання цих операцій виникає помилка, він відправить помилку з кодом 400 (див. рис. 3.34).

```
if (req.method === "GET") {
  try {
    await connectToMongoDB();
    const themes = await Theme.find();
    res.status(200).send(themes);
  } catch (error) {
    res.status(400).send({ error, msg: "something wrong" });
```

Рисунок 3.34 – Метод GET

Написав код обробки POST-запитів на сервері в Next.js, що дозволяє користувачам створювати нові теми та зберігати їх в базі даних MongoDB. У разі успіху, обробник відправляє відповідь з кодом статусу 201 та об'єктом

теми. Якщо створення нової теми завершується з помилкою, обробник відправляє відповідь з кодом статусу 405 та повідомленням про помилку (див. рис. 3.35).

```
if (req.method === "POST") {
if (!req.body) {
return res.status(400).json({ error: "Data is missing" });
}
const { author, text, themename } = req.body;
const themeobj = {
author: author,
themename: themename,
text: text,
comments: [],
};
Theme.create(themeobj)
.then((result) => {
return res.status(201).json({
success: true,
themeobj,
});
.catch((err) => {
res.status(405).json({ error: err });
```

Рисунок 3.35 – Метод обробки POST-запитів

Наступний код було розроблено для POST-запитів на сервері, що дозволяє користувачам створювати нові коментарі до існуючих тем в базі даних MongoDB. Він використовує метод Theme.find() для пошуку в базі даних теми з відповідним ID. Для цього об'єкта він отримує масив з коментарями

У випадку успішного додавання коментаря сервер відправляє відповідь з кодом статусу 200 та повідомленнями про успіх. У випадку помилки при додаванні коментаря, сервер відправляє відповідь з кодом статусу 400 та повідомленням про помилку (див. рис. 3.36).

```
const obj = await Theme.find({ _id: id }).distinct("comments");
const aue = await Theme.updateOne(
{ _id: id },
$set: {
comments: [...obj, { text: text, author: author }],
},
},
{ strict: false }
).then((result, err) => {
return res.status(200).json({
data: result,
message: "Value Updated",
message2: obj,
});
} catch (error) {
res.status(400).send({ error, msg: "something wrong" });
```

Рисунок 3.36 – Код створення створювати нових коментарів

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було створено веб-сайт і форум для рок-групи.

Для розробки використовувався наступний стек технологій:

– платформа Node.js, мова JavaScript;
– база даних MongoDB, інструмент Yarn;
– фреймворк Next.js.

У ході роботи були виконані наступні завдання:

– аналіз сфери музичних груп, визначення функціональності аналогічних веб-сайтів і форумів;
– формулювання мети розробки, складання технічного завдання та вимог до сайту і форуму;
– проєктування та створення бази даних, використовуючи MongoDB;
– розробка клієнтської частини сайту, використовуючи фреймворк Next.js;
– тестування сайту і форуму.

Розроблений вебсайт та форум відповідають сформованому технічному завданню та вимогам.

ПЕРЕЛІК ПОСИЛАНЬ

1. Node.js Documentation. URL: https://nodejs.org/dist/latest-v18.x/docs/api (дата звернення: 18.03.2023).
2. Next.js Documentation. URL: https://nextjs.org/docs (дата звернення: 18.03.2023).
3. Yarn Documentation. URL: https://classic.yarnpkg.com/lang/en/docs/cli/add/ (дата звернення: 22.03.2023).
4. MongoDB Documentation. URL: https://docs.mongodb.com/manual (дата звернення: 19.03.2023).
5. JavaScript Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 18.03.2023).
6. Mongoose Documentation. URL: https://mongoosejs.com/docs/guide.html (дата звернення: 18.03.2023).
7. JSON Web Tokens Documentation. URL: https://jwt.io/introduction/ (дата звернення: 22.03.2023).
8. Visual Studio Code Documentation. URL: https://code.visualstudio.com/docs (дата звернення: 22.03.2023).