МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
_____
(підпис)
01 червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 – Комп'ютерні науки,
освітньо-професійної програми «Інформатика»
на тему: «Вебзастосунок ведення тематичного блогу»
здобувачки групи ІН – 02, Кривохижи Марини Дмитрівни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Марина КРИВОХИЖА
(підпис)

Керівник,
асистент кафедри комп'ютерних наук
кандидат фізико-математичних наук Ольга ШУТИЛЄВА _____
(підпис)

СУМИ – 2024

Сумський державний університет

Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 - Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувачки групи ІН-02, Кривохижи Марини Дмитрівни

1. Тема роботи: «Вебзастосунок ведення тематичного блогу» затверджена наказом по СумДУ від *«22» квітня 2024 р. № 0414-VI*
2. Термін здачі здобувачем кваліфікаційної роботи до *01 червня 2024 року.*
3. Вхідні дані до кваліфікаційної роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Проаналізувати проблеми та актуальності розробки блогу, конкурентів та постановка й формування завдань дослідження. 2) Обрати оптимальний підхід для створення блогу. 3) Розробити блог, використовуючи вибрані програмні засоби та дотримуючись завдань. 4) Аналіз отриманих результатів. 5) Оформлення пояснювальної записки до кваліфікаційної роботи.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «08» квітня 2024 р.

Завдання прийняв до виконання _____ Керівник _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Проаналізувати проблеми та актуальності розробки блогу, конкурентів та постановка й формування завдань дослідження*	06.05-10.05.2024	
2	*Обрати оптимальний підхід для створення блогу*	10.05-13.05.2024	
3	*Розробити блог, використовуючи вибрані програмні засоби та дотримуючись завдань*	13.05-20.05.2024	
4	*Аналіз отриманих результатів*	20.05-22.05.2024	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	22.05-28.05.2024	

Здобувач вищої освіти_____ Керівник _____

АНОТАЦІЯ

Записка: 84 стор., 20 рис., 3 табл., 23 додатки, 23 використаних джерел.

Обґрунтування актуальності теми роботи – створення сайту-блогу є важливим в сучасних умовах, коли інтернет стає основним джерелом інформації та комунікації. Блог дозволить користувачам ділитися своїми ідеями, досвідом, знаннями та точками зору на різноманітні теми, сприяючи вільному обміну інформацією та розвитку спільноти.

Об’єкт дослідження – розробка блогу, який забезпечить зручність використання для користувачами.

Мета роботи – основною метою роботи є створення інноваційного та зручного блогу, який дозволить користувачам вільно обмінюватися ідеями та знаннями.

Методи дослідження – у роботі використано методи аналізу конкурентів, розробки дизайну, вибору інструментів та технологій для створення інтерфейсу, зберігання даних та інтеграції системи розсилки.

Результати – результатом роботи є функціональний блог з інтуїтивно зрозумілим інтерфейсом, ефективною системою зберігання даних та можливістю інтеграції з системою розсилки. Проект показує, як сучасні вебтехнології можуть створювати зручні та корисні платформи для обміну інформацією та розвитку спільнот.

БЛОГ, ВЕБРОЗРОБКА, NEXT.JS, MONGODB, MAILCHIMP, GITHUB

ЗМІСТ

ВСТУП

У сучасному світі інформаційні технології стрімко розвиваються, забезпечуючи постійний обмін даними. Одним із ключових особливостей нашого часу є можливість для кожної людини висловити свою думку, поділитися знаннями та досвідом з іншими. Такий розширений доступ до інформації має величезне значення для суспільства, оскільки сприяє вільному обміну ідеями та розвитку нових концепцій.

Актуальність роботи визначається у тому, що в нашому світі існує величезний потік інформації, але не завжди є можливість почути голоси звичайних людей, які можуть мати цінний досвід та знання. Блог створює простір для таких людей, допомагаючи їм знайти свою аудиторію та сприяти більш відкритому і демократичному суспільству.

Об’єкт дослідження - процес розробки веб-платформи для обміну знаннями та досвідом користувачів.

Метою роботи є створення веб-платформи під назвою "OpenMind", яка дозволить користувачам висловлювати свої думки та ділитися знаннями з широкою аудиторією. Платформа має бути інтуїтивно зрозумілою та зручною у використанні.

Гіпотеза дослідження полягає у тому, що створення зручної та функціональної платформи для обміну знаннями сприятиме розвитку спільнот та взаєморозумінню між користувачами, а також покращить доступ до цінної інформації.

Новизна фінального результату полягає в створенні інтерактивної платформи, яка сприятиме обміну ідеями та знаннями, надаючи користувачам можливість висловити свої думки та взаємодіяти з іншими учасниками.

Структура роботи складається зі вступу, аналізу предметної області та існуючих аналогів, вибору фреймворку, бібліотек, розсилки, бази даних, репозиторію, також практичної реалізації та огляду використання програмного додатку , висновків, списку використаних джерел та додатків.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз предметної області

Поява сайтів-блогів знаменувала собою нову еру в розвитку інтернету, дозволивши будь-якій людині з доступом до мережі Інтернет створювати та публікувати власний текст. Перші блоги, що з'явилися наприкінці 1990-х років [1], мали форму онлайн-щоденників, де користувачі ділилися своїми думками, досвідом та подіями з особистого життя. Термін "блог" походить від слова "weblog" і буквально означає "веб-щоденник". Ці ранні блоги були простими, зосередженими переважно на текстовому контенті, але з часом вони еволюціонували у складніші та багатогранніші платформи.

Основна мета блогу – створення простору, де люди ділитимуться інформацією, обговорювати ідеї та висловлювати свої думки. Блоги можуть бути особистими, тематичними чи професійними. Вони можуть охоплювати широкий спектр тем.

В історії сайтів блогів важливим стало поширення серед різних професійних спільнот. Інструменти для блогінгу стали настільки зручними і доступними, що фахівці різних галузей, таких як медицина, право, бізнес та інші, почали використовувати їх для обміну знаннями, досвідом та найновішими тенденціями у своїх сферах. Це дало початок експертним блогам, де можна знайти якісну та достовірну інформацію, засновану на досвіді та дослідженнях.

Такі сайти також мають потенціал для монетизації. Блогери використовують різні методи, щоб отримувати дохід, такі як реклама, партнерські програми, спонсорство, продаж товарів або послуг, а також донати від читачів.

Ще одним важливим аспектом блогів є вплив на формування громадської думки. Блогери, які стали популярними завдяки своїм унікальним точкам зору, можуть впливати на погляди та поведінку своїх читачів. Деякі блогери стали

лідерами думок, здобули статус знаменитостей і отримують запрошення на заходи, інтерв'ю та співпрацю з великими брендами.

Сучасні блоги відрізняються різноманіттям контенту та підходів до його подання. Віртуальна реальність, штучний інтелект, інтерактивні елементи та нові формати контенту розширюють можливості для блогінгу. Це робить блоги все більш багатогранними та інноваційними, залишаючись при цьому важливим.

1.2 Аналіз існуючих аналогів

Перед розробкою сайту-блогу, який дозволить користувачам ділитися своїми думками та досвідом, було проведено аналіз існуючих веб-ресурсів з подібною тематикою. Завдяки цьому було визначено сильні і слабкі сторони цих сайтів. Це допомогло сформувати бачення для функціональності та інтерфейсу майбутнього сайту-блогу [2].

Перший аналог – “Medium” (рис. 1.1) [3], який є популярною платформою для блогів. Сайт привертає увагу своєю зручністю та стильним дизайном, дозволяє користувачам легко створювати та публікувати контент. Платформа пропонує широкий спектр тем, а також можливість спілкування та взаємодії з іншими авторами та читачами. Сильними сторонами “Medium” є зручна навігація, чистий дизайн та широкі можливості для поширення контенту. Однак, обмеження в безкоштовній версії та необхідність підписки на преміум-послуги можуть бути недоліками для деяких користувачів.

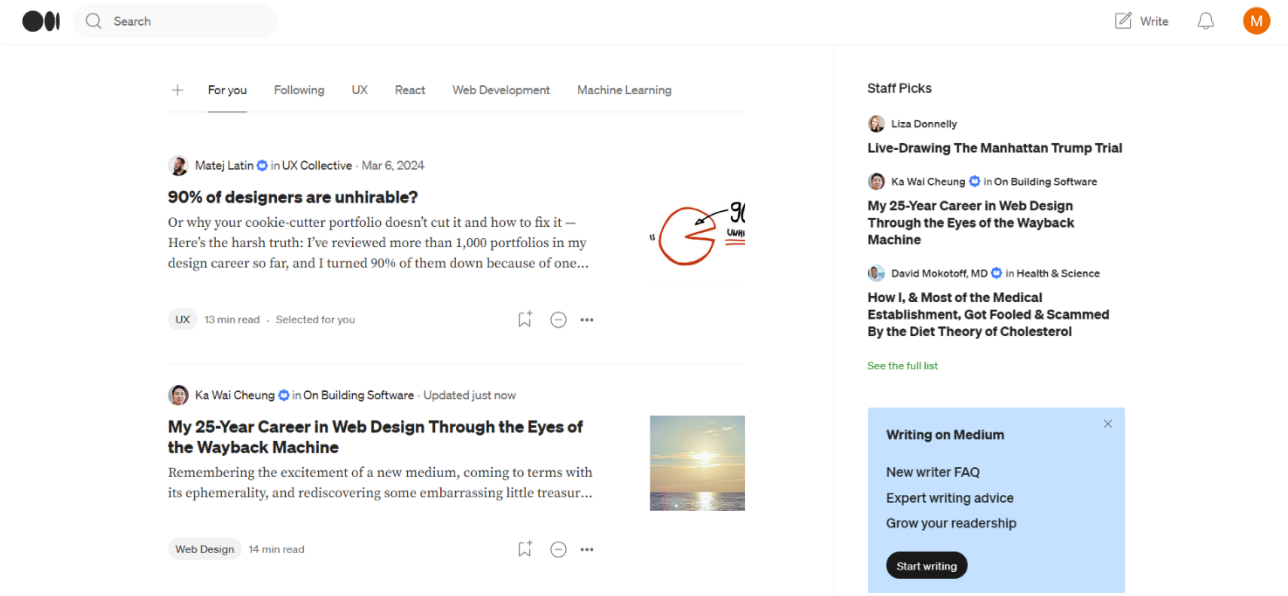

Рисунок 1.1 – Сайт-блог “Medium”

Другий аналог – “Dev.to” (рис.1.2) [4]– це спільнота для розробників, яка забезпечує платформу для обміну знаннями та досвідом у сфері технологій та програмування. Сайт пропонує зручну структуру, де користувачі можуть публікувати статті, писати коментарі та ділитися своїми проектами. Сильними сторонами “Dev.to” є його спеціалізація на технологічній тематиці, що робить його популярним серед розробників. Недоліком є вузька спрямованість контенту, яка обмежує залучення більш широкої аудиторії.

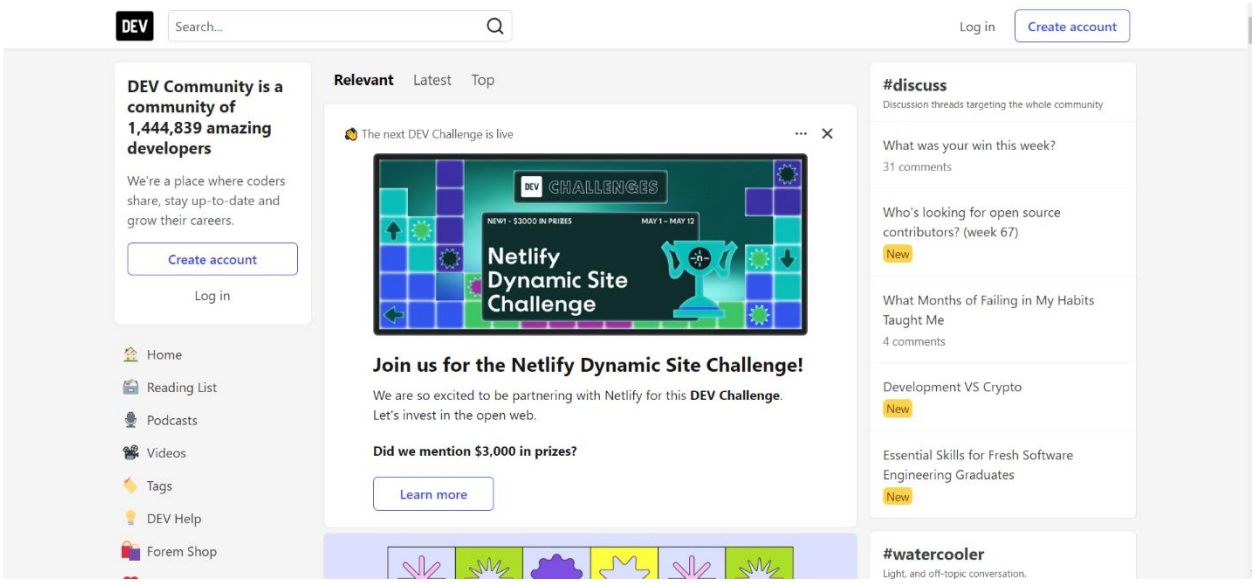

Рисунок 1.2 – Сайт-блог “Dev.to”

Третій аналог – “Steemit” (рис.1.3) [5]– це платформа для блогів, яка базується на технології блокчейн. Вона надає можливість користувачам отримувати винагороди у вигляді криптовалюти за створення та взаємодію з контентом. Сильними сторонами “Steemit” є інноваційна технологія та можливість монетизації контенту. Однак, сайт має непривабливий дизайн, складність використання для користувачів, які не знайомі з блокчейном, та нестабільність криптовалют можуть бути недоліками.

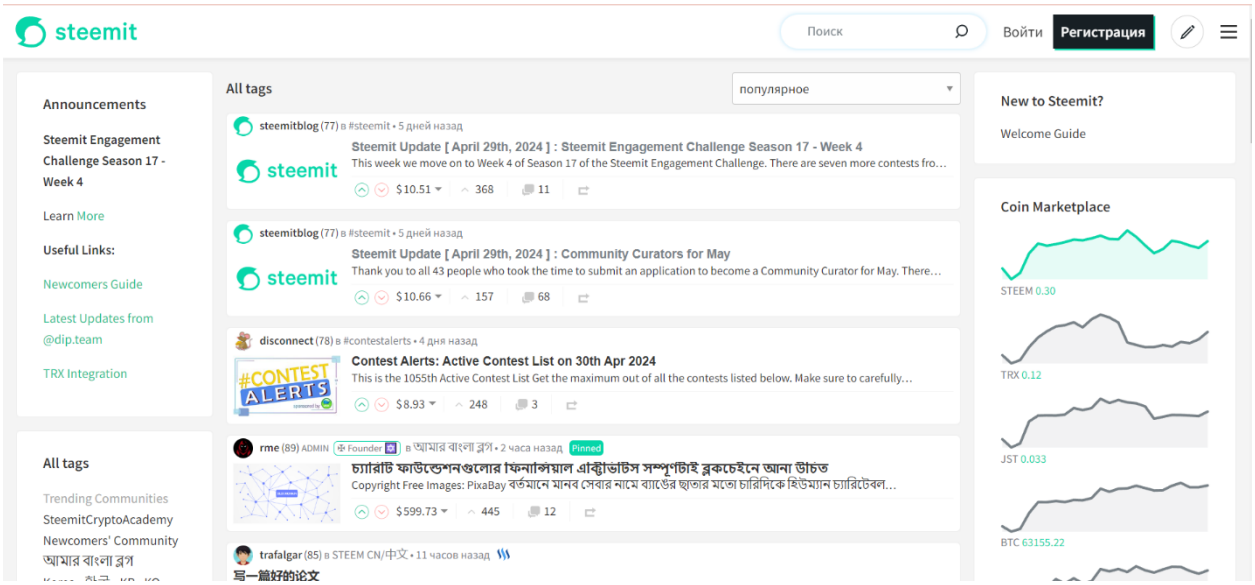

Рисунок 1.3 – Сайт-блог “Steemit”

Четвертий аналог – “HubPages” (рис.1.4) [6]– пропонує платформу для створення контенту на різні теми. Він надає можливість користувачам заробляти на рекламі, спонсорських матеріалах та партнерських програмах. Сильними сторонами “HubPages” є різноманітність контенту та можливості для монетизації. Однак, складна структура та наявність великої кількості реклами є недоліками.

Рисунок 1.4 – Сайт-блог “HubPages”

П'ятий аналог – “Vocal.Media” (рис.1.5) [7]– платформа, яка орієнтується на творців контенту в різних жанрах, включаючи статті, відео та подкасти. Вона пропонує можливості для монетизації та створення спільнот. Сильними сторонами “Vocal.Media” є його різноманітність та підтримка креативного контенту. Однак, конкуренція серед авторів і складність виходу на стабільний дохід можуть бути викликами для новачків.

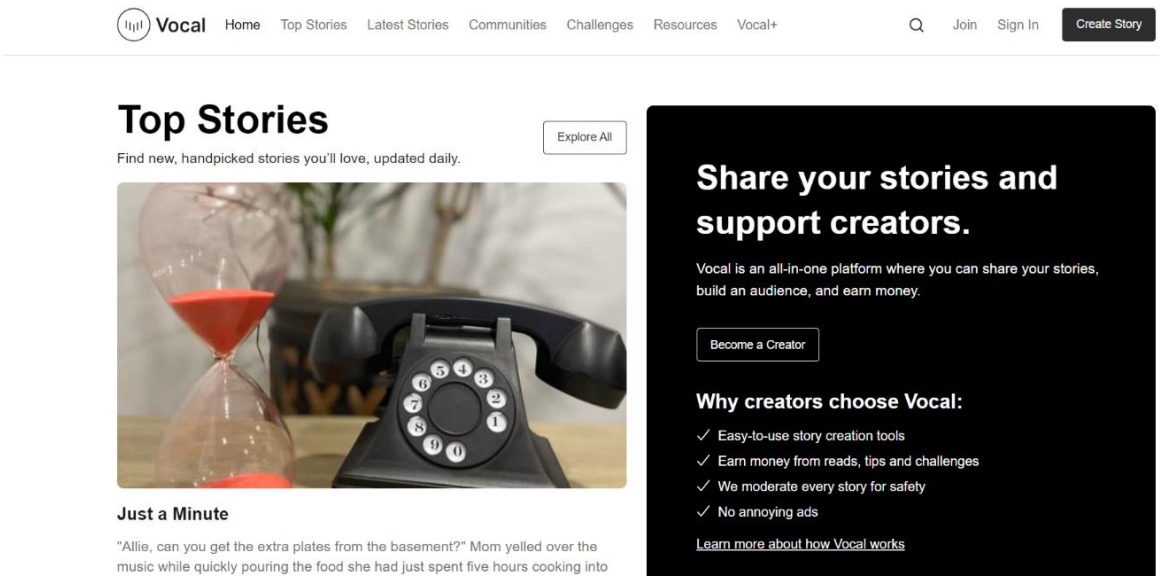

Рисунок 1.5 – Сайт-блог “ Vocal.Media ”

Згідно аналізу аналогів, була сформована порівняльна таблиця розглянутих прикладів та майбутнього інтернет-магазину. Результати аналізу представлено у таблиці 1.1.

Таблиця 1.2.1 – Порівняльний аналіз аналогів

Назва критерію	Назва ресурсу				
	Medium	Dev.to	Steemit	HubPages	Vocal.Media
Зручність навігації	+	+	-	-	+
Адаптивність	+	+	+	+	+
Монетизація	-	-	+	+	+
Дизайн	+	+	-	+	+
Різноманітність контенту	+	-	-	+	+
Відсутність реклами	+	+	+	-	+

1.3 Постановка задачі

У межах виконання кваліфікаційної роботи необхідно розробити сайт-блог під назвою "OpenMind". Основна мета – створити онлайн-платформу, де користувачі можуть ділитися своїми ідеями, досвідом, знаннями та точками зору на різні теми.

Для успішної реалізації проекту необхідно на основі аналізу існуючих аналогічних рішень – визначити, які саме елементи дизайну, функціональності та інтерфейсу є популярними та ефективними. Крім того, аналіз конкурентів дозволить виявити можливі слабкі сторони, які варто уникнути при розробці власного сайту-блогу.

Далі слід розробити дизайн сайту, враховуючи сучасні тенденції та потреби користувачів. Дизайн має бути естетично привабливим, але водночас функціональним і зручним. "OpenMind" має бути інтуїтивно зрозумілим та зручним у використанні для користувача.

Наступним кроком є вибір інструментів для розробки інтерфейсу. Потрібно обрати фреймворк, який гарантуватиме стабільність та швидке завантаження. Крім того, слід передбачити можливість інтеграції системи розсилки для підтримки зв'язку з користувачами та інформування їх про новий контент або важливі оновлення. Для зберігання та управління даними про користувачів та постів буде використана база даних.

2. ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Вибір фреймворку

Фреймворк – це набір інструментів та бібліотек, які надають структуру та базові функції для розробки програмного забезпечення. Використання фреймворку спрощує процес розробки, оскільки він вже має готові компоненти та шаблони, дозволяючи розробникам зосередитися на функціональності програми, а не на дрібних деталях.

Для розробки сайту-блогу обрано фреймворк Next.js [8]– це популярний фреймворк для створення сучасних веб-додатків на базі React [9] та JavaScipt [10] [11], який дозволяє створювати високопродуктивні та SEO-оптимізовані веб-сайти. Його переваги включають серверний рендеринг, статичне генерування сторінок, автоматичну оптимізацію та підтримку різних підходів до створення маршрутів.

Вибір Next.js обумовлений його універсальністю та популярністю серед розробників. Цей фреймворк надає широкі можливості для оптимізації веб-сайтів, що є важливим для SEO та швидкого завантаження сторінок. До того ж Next.js забезпечує зручність інтеграції з різними сервісами та інструментами, що робить його ідеальним вибором для сайту-блогу, який прагне залучити широку аудиторію.

Популярність Next.js постійно зростає. За статистикою, у 2022 [12] році частка використання цього фреймворку становила 13,93% (рис.2.1), тоді як у 2023 [13] році вона зросла до 16,68% (рис.2.2). Такий зріст свідчить про надійність та ефективність Next.js, а також про його активне впровадження в проекти різного масштабу та складності.

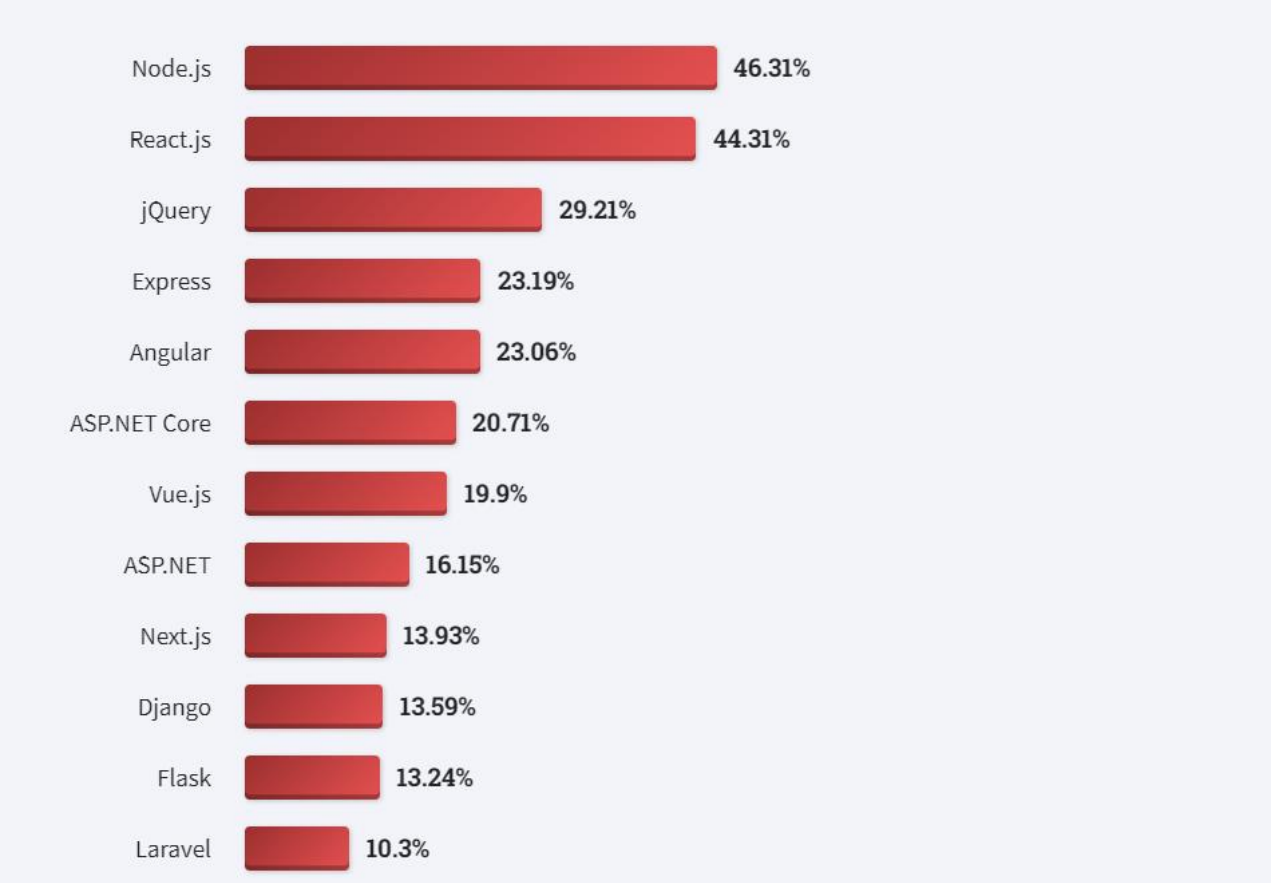

Рисунок 2.1 – Найпопулярніші фреймворки 2022

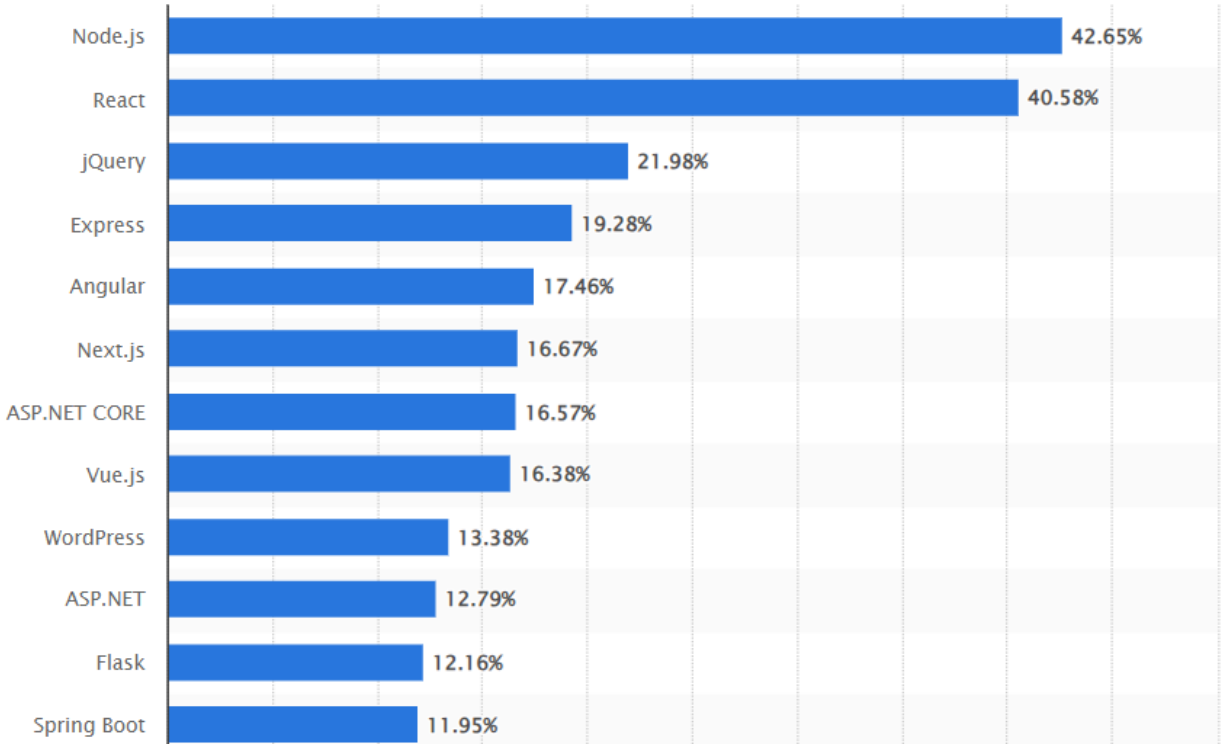

Рисунок 2. 2 – Найпопулярніші фреймворки 2023

Отже, вибір Next.js як фреймворку для розробки сайту-блогу "OpenMind" є обґрунтованим рішенням, яке забезпечує ефективність, продуктивність та гнучкість у розробці. Цей вибір дозволяє створити сучасний вебсайт, який буде відповідати вимогам ринку та потребам користувачів.

2.2 Вибір розсилки

Для ефективного зв'язку з користувачами сайту-блогу "OpenMind" необхідно використовувати сервіс розсилки. Такі сервіси дозволяють надсилати електронні листи великій кількості підписників, тримати їх в курсі нових публікацій, а також відстежувати взаємодію з листами. Сервіси розсилки є важливими інструментами для підтримки зворотного зв'язку, залучення аудиторії та підтримки взаємодії.

Для сайту-блогу обрано Mailchimp [14] як сервіс для розсилки. Mailchimp – це один із найпопулярніших сервісів електронної пошти, який надає широкий спектр можливостей для створення та управління розсилками. Він зручний для користувачів завдяки інтуїтивно зрозумілому інтерфейсу та різноманітним шаблонам для електронних листів.

Один із головних плюсів цього сервісу – це його гнучкість. Він дозволяє налаштовувати розсилки відповідно до потреб сайту, включаючи дизайн, вміст, сегментацію та автоматизацію. Наприклад, за допомогою Mailchimp можна створити автоматичну розсилку для нових підписників або налаштувати індивідуальні розсилки для різних сегментів аудиторії.

Mailchimp також пропонує аналітику та звіти, що дозволяє відстежувати ефективність розсилок. Це допомагає зрозуміти, які листи працюють краще, які мають вищий рівень відкриттів, і відповідно коригувати стратегію комунікації.

Вибір Mailchimp як сервісу для розсилки для сайту-блогу "OpenMind" базується на його зручності, гнучкості, інтеграції з іншими сервісами та можливостях аналітики. Цей сервіс допоможе ефективно підтримувати зв'язок із користувачами та сприяти зростанню аудиторії блогу.

2.3 Вибір репозиторію

Репозиторій — це місце, де зберігається вихідний код програмного забезпечення, разом із історією змін, доповненнями та гілками розробки. Репозиторії дозволяють командам розробників працювати разом над одним проектом, відстежувати зміни коду, відновлювати попередні версії, а також мати резервну копію всіх змін у безпечному середовищі.

Для зберігання коду та контролю версій у розробці сайту-блогу "OpenMind" обрано GitHub [15]. Він є однією з найпопулярніших платформ для репозиторіїв, яка використовує систему керування версіями Git. Вона дозволяє розробникам працювати спільно над кодом, обмінюватися змінами та легко відстежувати історію розробки.

GitHub Desktop – це графічний клієнт для взаємодії з репозиторіями GitHub, який надає користувачам зручний спосіб працювати з кодом. Завдяки інтуїтивно зрозумілому інтерфейсу GitHub Desktop спрощує такі завдання, як створення гілок, додавання змін до репозиторію, створення коммітів і відстеження змін у проєкті.

Крім того, GitHub дозволяє зберігати код у хмарі, що забезпечує доступ до нього з будь-якого місця та з різних пристроїв.

Таким чином, використання GitHub та GitHub Desktop для зберігання коду забезпечить ефективну роботу, зручне відстеження змін та високу надійність зберігання вихідного коду проєкту "OpenMind".

2.4 Вибір бази даних

База даних (БД) є важливим компонентом будь-якого вебдодатку, що зберігає та управляє даними. Вибір відповідної БД має вирішальне значення для забезпечення ефективного зберігання даних, швидкого доступу до них та надійності роботи додатку.

Для поставлених до розробки задач у роботі було обрано використання MongoDB [16] як БД. MongoDB – це документно-орієнтована БД, яка зберігає дані у форматі JSON-подібних документів. Це забезпечує високу гнучкість і масштабованість, що є важливим для сучасних вебзастосунків. MongoDB дозволяє зберігати складні, вкладені структури даних без необхідності визначати схему заздалегідь, що полегшує розробку та підтримку додатка. Однією з головних переваг MongoDB є її здатність обробляти великі обсяги даних і підтримувати горизонтальне масштабування, що робить її ідеальним вибором для високонавантажених вебзастосунків.

У MongoDB присутня можливість використовувати API (Application Programming Interface) шляхи. API шляхи дозволяють додатку спілкуватися з БД за допомогою HTTP-запитів [17]. Це забезпечує зручний і стандартний спосіб доступу до даних, який може бути легко інтегрований з фронтенд-частиною сайту. За допомогою API шляхів можна виконувати різні операції з даними, такі як створення нових записів, читання існуючих даних, оновлення та видалення записів. Це дозволяє динамічно керувати контентом на сайті та забезпечувати швидкий доступ до актуальної інформації.

Вибір MongoDB як БД для проєкту "OpenMind" був обґрунтований її гнучкістю, масштабованістю та зручністю використання у поєднанні з API шляхами.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Проєктування сайту

Проєктування сайту-блогу починається з визначення його структури, яка задає організацію сторінок і навігацію між ними. Структура повинна бути зрозумілою, зручною та інтуїтивною, щоб користувачі могли легко знаходити потрібну їм інформацію [18]. Сайт включає кілька основних сторінок, кожна з яких виконує конкретну роль.

Сторінки сайту для користувача:

- Головна сторінка
- Категорії
- Задана категорія
- Пост
- Про нас
- Додати пост
- Всі пости
- Вхід/Вихід

Розроблена структура сайту-блогу представлена нижче (рис 3.1).

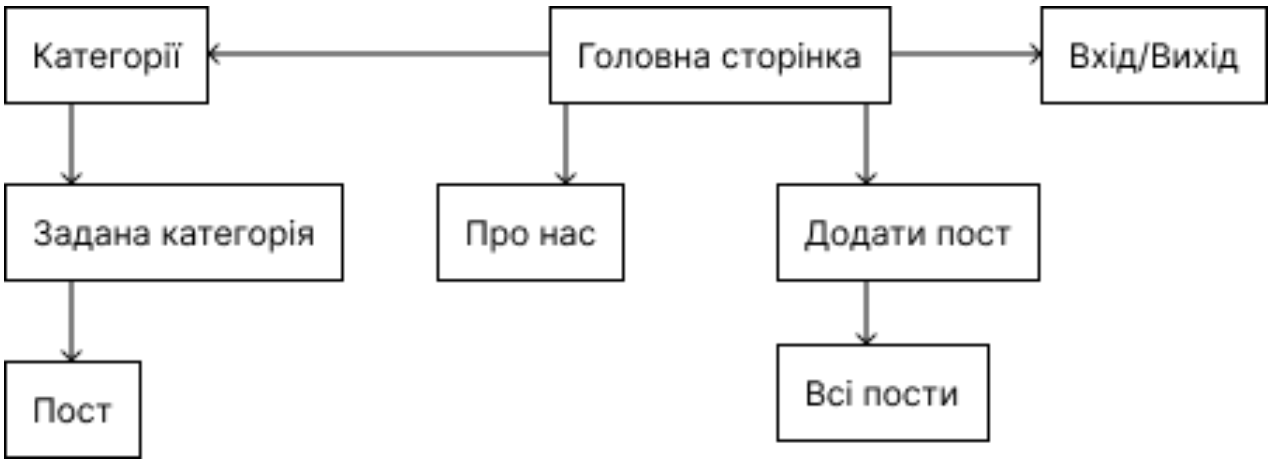

Рисунок 3.1 – Структура сайту-блогу

3.2 Інформаційна модель

Інформаційна модель є важливим кроком у процесі розробки. Вона допомагає візуалізувати, як користувачі будуть взаємодіяти з сайтом, і дозволяє розробникам продумати навігацію, функціональність та розміщення контенту [18].

Однією з основних складових інформаційної моделі є прототип. Прототип — це попередня версія сайту, яка створюється для демонстрації і тестування дизайну та функціональності. Прототип не обов'язково має бути повністю функціональним, але він повинен давати уявлення про те, як сайт виглядатиме і працюватиме в реальних умовах.

Інформаційна модель з використанням прототипу є важливою частиною для створення успішного сайту. Це допомагає розробникам і дизайнерам працювати злагоджено, також зменшує ризики, пов'язані з внесенням змін у кінцевому продукті. Прототип дає можливість побачити загальну картину проекту та забезпечує ефективне планування і реалізацію сайту.

Результати прототипів деяких сторінок зображено на рисунках 3.2-3.4.

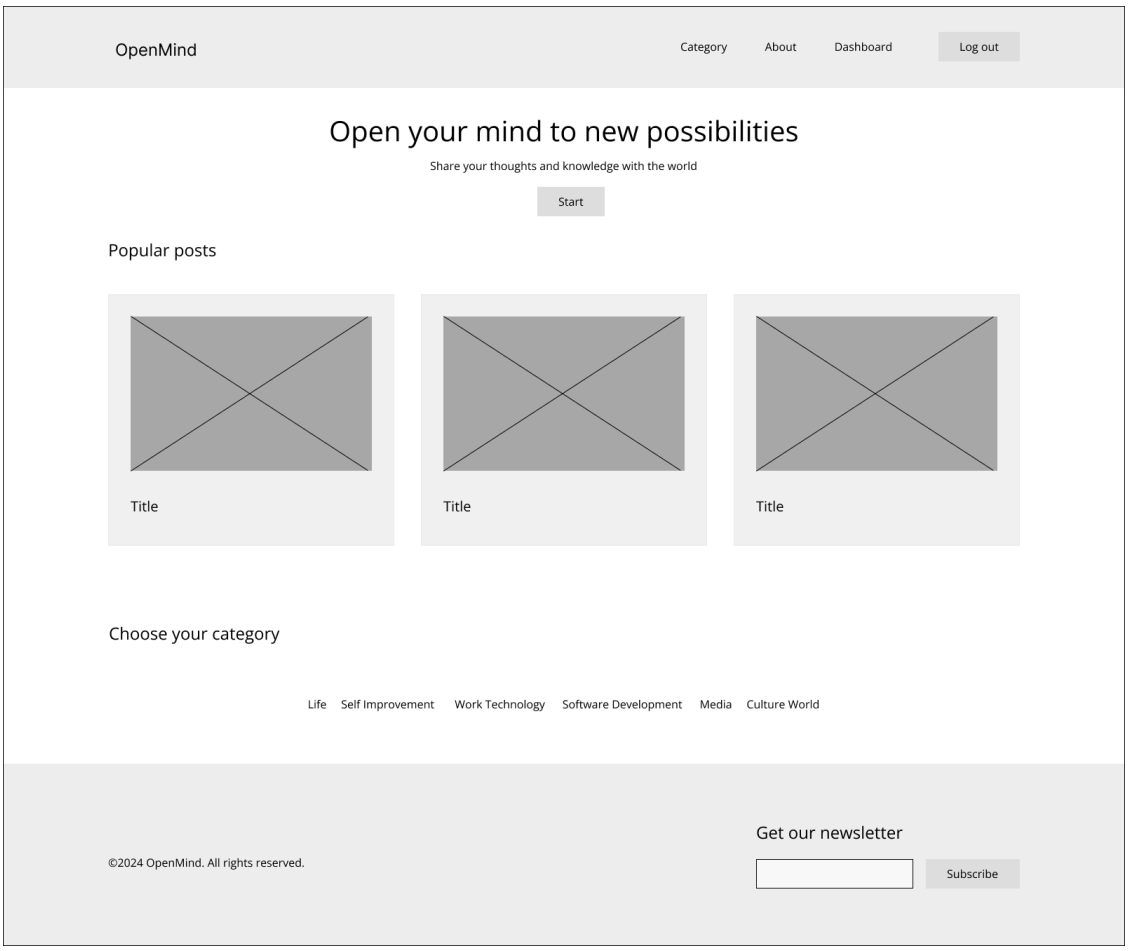

Рисунок 3.2 – Прототип сторінки “Головна сторінка”

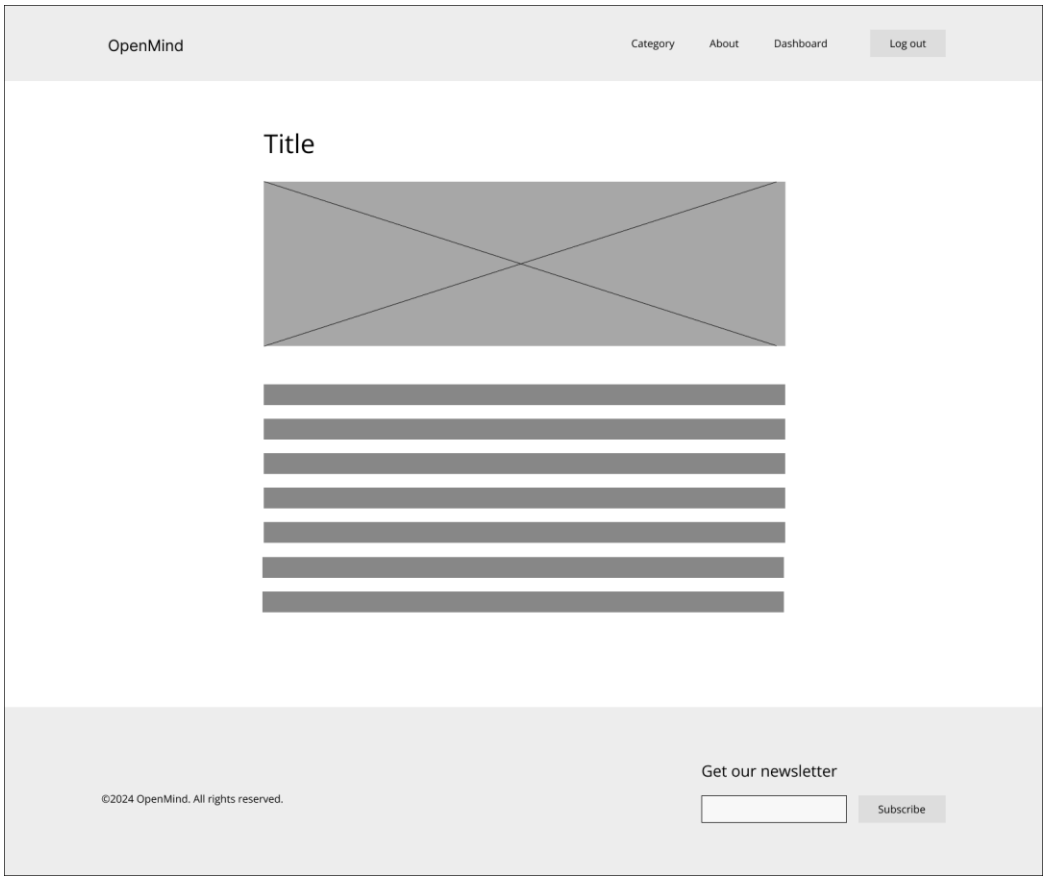

Рисунок 3.3 – Прототип сторінки “Пост”

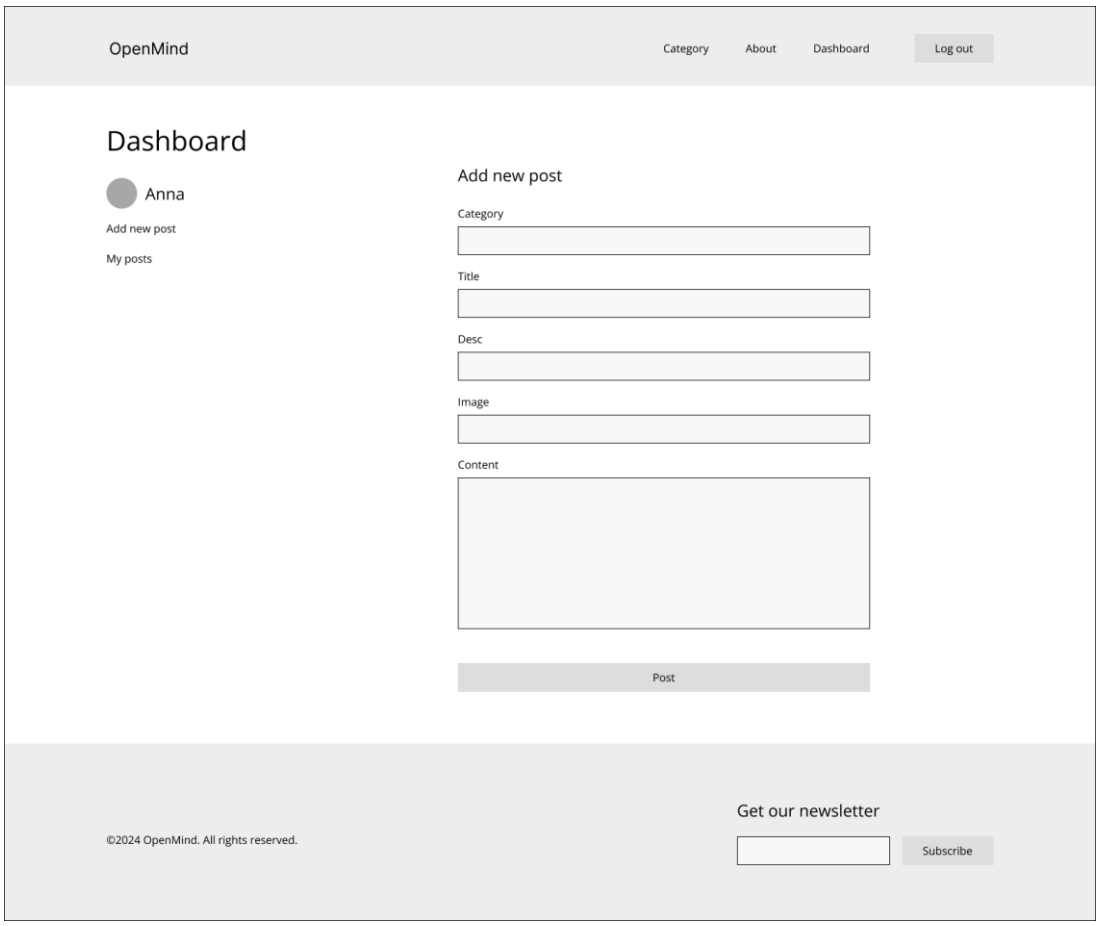

Рисунок 3.4 – Прототип сторінки “Додати пост”

3.3 Підготовка робочого середовища

Підготовка до проєкту передбачає налаштування робочого середовища, організацію структури папок та інсталяцію необхідних додатків. Ці кроки важливі для того, щоб забезпечити ефективну розробку.

Для початку роботи потрібно завантажити Node.js [19] з офіційного сайту. Node.js є необхідним компонентом для роботи з Next.js, оскільки Next.js використовує Node.js як середовище виконання.

Тепер переходимо до створення проекту в Visual Studio Code. У терміналі потрібно ввести команду для встановлення Next.js [8].

```
npx create-next-app@latest
```

Після завантаження фреймворку, ми отримали структуру папок, далі добавимо папки для бажаної структури сайту (рис. 3.5).

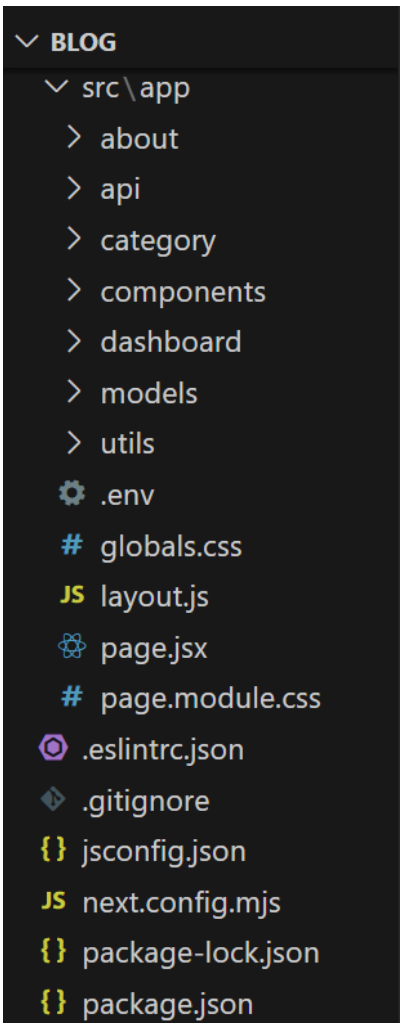

Рисунок 3.5 – Структура папок про’кту

Розглянемо, що містить кожна папка:

- src/app – містить код головної сторінки та інші папки – сторінки сайту, налаштування/підключення БД;
- about – папка, яка містить код з інформацією про сайт;
- api – папка, яка містить папки з серверною логікою для обробки HTTP-запитів
- auth – папка, яка містить папки для автентифікації користувачів.
- [...nextauth] - папка, яка містить код для налаштування та обробники автентифікації користувачів.
- signup - папка, яка містить код для реєстрації нових користувачів.
- posts - папка, яка містить код для роботи з постами

- [category] - папка, яка містить код для роботи з постами за певними категоріями.
- [id] - папка, яка містить код для обробки запитів до конкретного посту.
- category – містить код сторінки з певною категорією та папками.
- [topic] - папка, яка містить код для роботи з постами за певною темою.
- [id] - папка, яка містить код для обробки запитів до конкретного посту.
- components – папка, яка містить React-компоненти.
- authProvider - папка, яка містить код, який відповідає за забезпечення автентифікації користувачів.
- button - папка, яка містить код-компонент кнопок.
- footer - папка, яка містить код-компонент для нижньої частини сторінок.
- input - папка, яка містить код-компонент полів вводу.
- menu - папка, яка містить код-компонент меню.
- navbar - папка, яка містить код-компонент навігаційної панелі.
- dashboard - папка, яка містить папки, які відповідають за інтерфейс панелі керування
- add - папка, яка містить код для додавання нових даних.
- signin - папка, яка містить код для входу користувачів.
- signout - папка, яка містить код для виходу користувачів.
- usetposts - папка, яка містить код для управління постами користувачів.
- models - папка, яка містить код для визначення моделей для роботи з базою даних.
- utils - папка, яка містить код для підключення до бази даних.

На ступний крок – занесення коду до репозиторію. Спочатку потрібно зареєструватися в GitHub [15], завантажити GitHub Desktop та під’єднати їх. Після цього можна заносити код (рис 3.6).

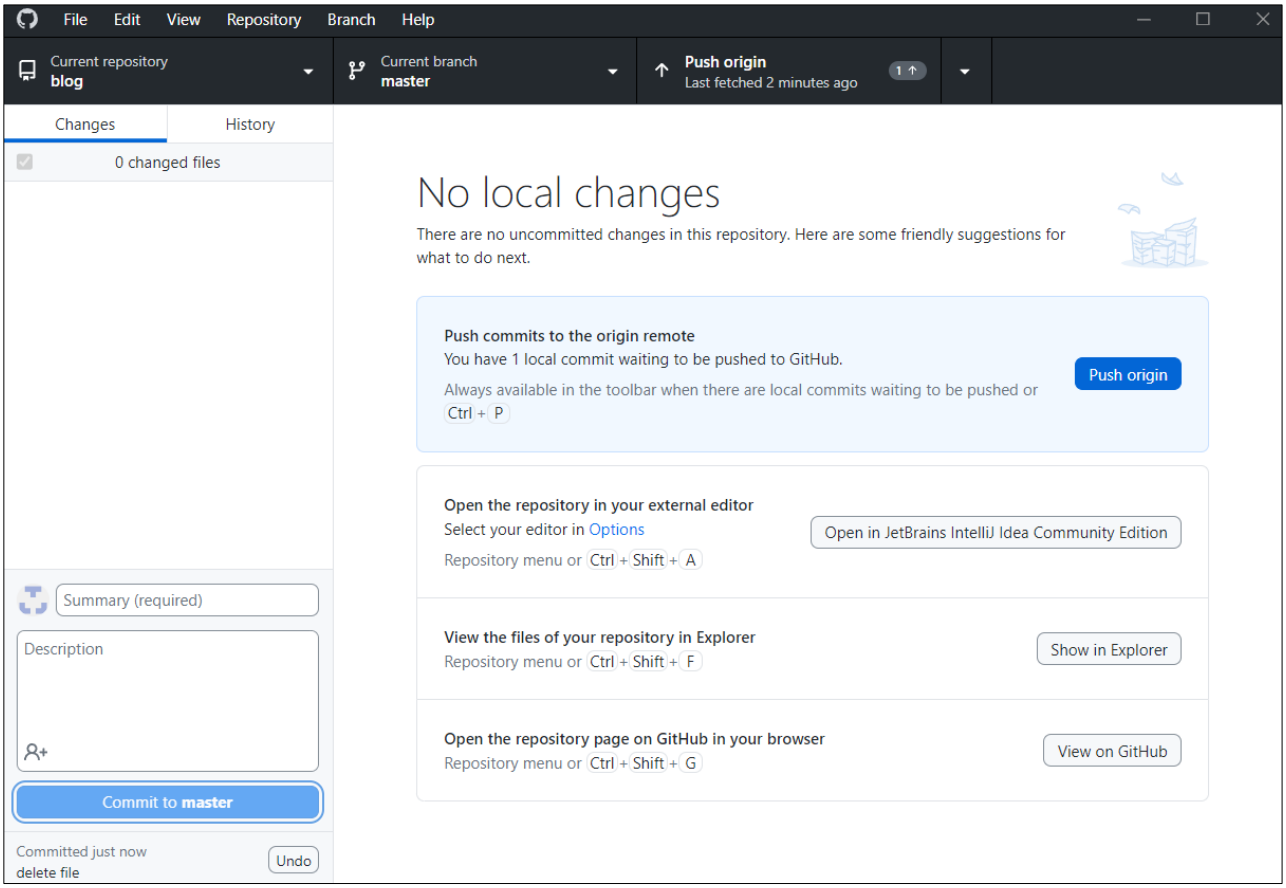

Рисунок 3.6 – GitHub Desktop

Тепер в GitHub з’явився код проекту, де можна переглянути його, подивитися історію змін та інше (рис. 3.7).

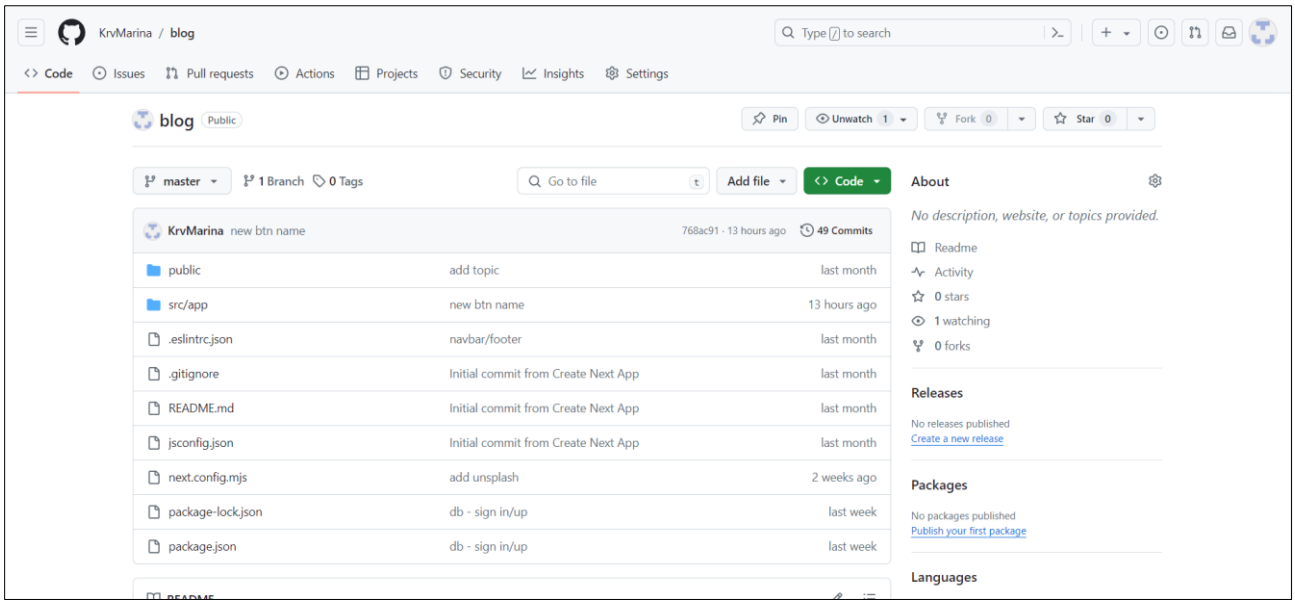

Рисунок 3.7 – GitHub Desktop

3.4 Проєктування бази даних

У розробці будь-якого вебдодатку потрібно проектувати БД, оскільки від цього залежить ефективність зберігання та обробки даних. У випадку сайту-блогу "OpenMind" база даних повинна забезпечувати надійне зберігання інформації про користувачів та публікації.

Схеми зображені в таблицях 3.1-3.2.

Таблиця 3. 1 – Схема користувача

Користувач	
Назва поля	**Тип даних**
Id	Number
Ім’я	String
Адреса	String
Пароль	String

Таблиця 3.2 – Схема публікації

Публікація	
Назва поля	**Тип даних**
Id	Number
Назва публікації	String
Ім’я користувача	String
Зображення	String
Категорія	String
Опис	String

3.5 Розробка клієнтської частини

Розробка клієнтської частини відповідає потребам користувачів. На цьому етапі здійснюється розробка інтерфейсу користувача та його взаємодії з

системою. Цей етап включає в себе відображення графічного інтерфейсу, реалізацію функціональності, взаємодію з сервером та обробку даних.

Переглянути можна в Додаток А – Додаток М.

Щоб запустити проект, в терміналі потрібно написати команду [8]:

```
npm run dev
```

Натиснути на http://localhost:3000 (рис. 3.8).

Рисунок 3.8 – Термінал

Вигляд головної сторінки сайту зображено на рисунку 3.9.

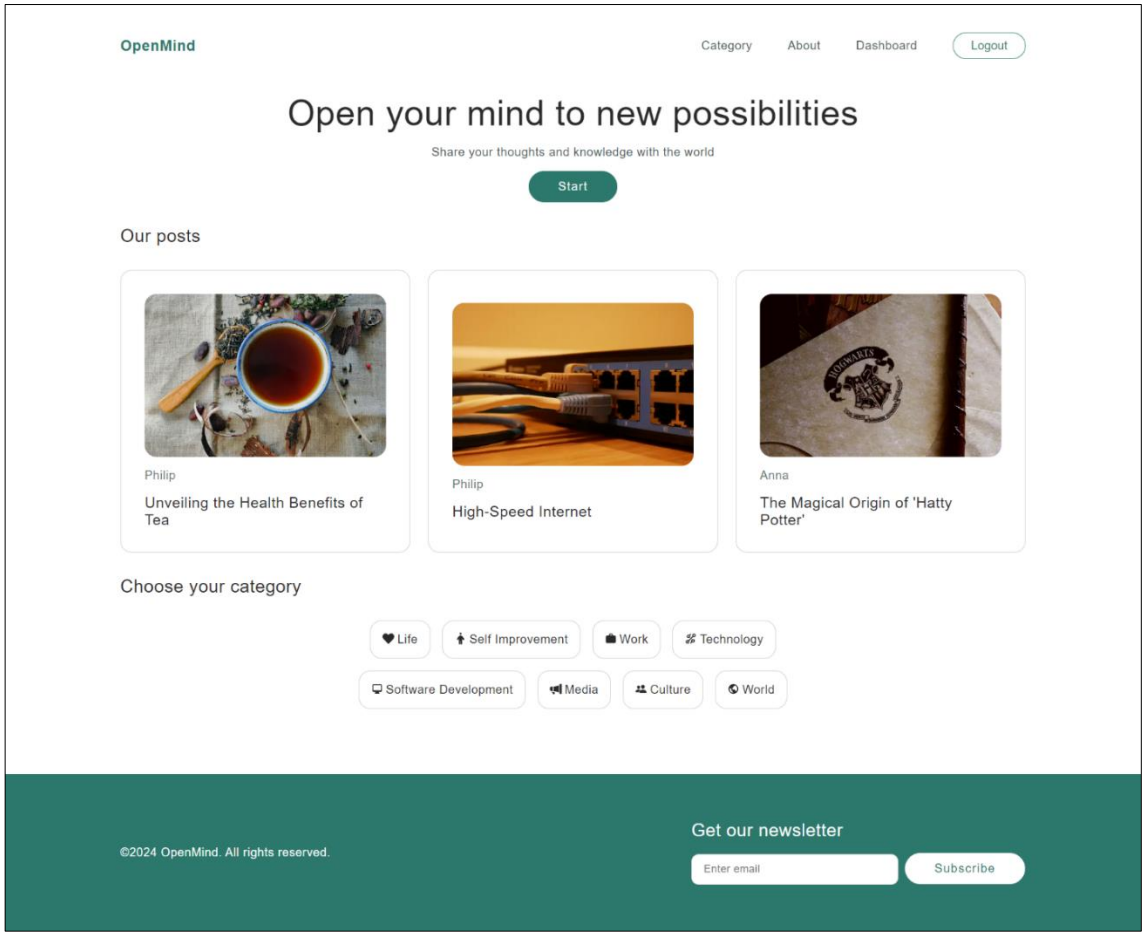

Рисунок 3.9 – Запуск проєкту

3.6 Розробка серверної частини

Спочатку слід перейти на офіційний сайт MongoDB [16] та зареєструватися. Після цього створюємо новий проект, обираємо безкоштовну версію та створюємо БД – пишемо назву та пароль. Відкривається вікно, де знаходиться унікальний рядок, який будемо використовувати для з'єднання з нашою БД (рис.3.10).

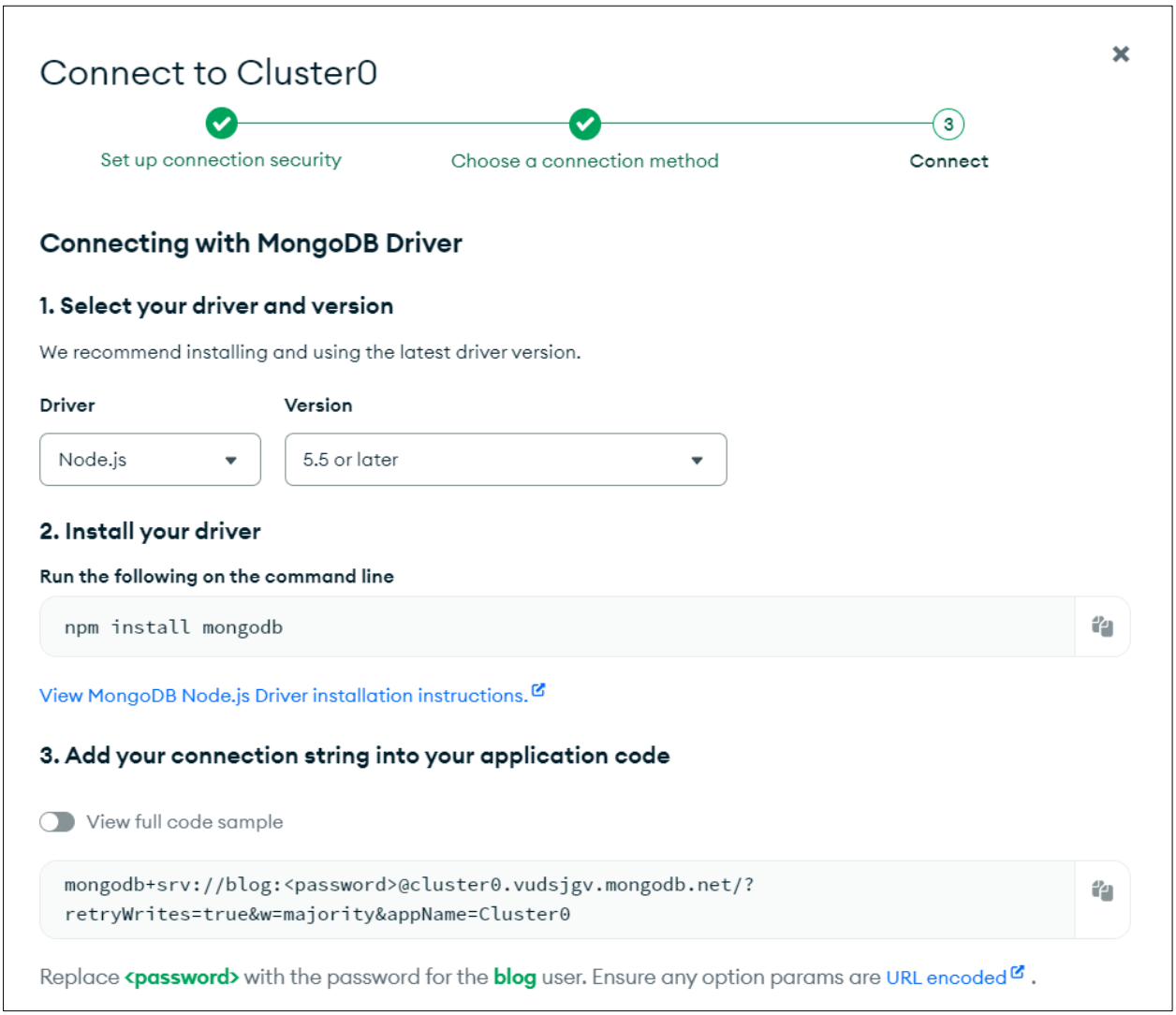

Рисунок 3.10 – Унікальний рядок

Також, щоб все працювало, потрібно завантажити mongoose [20]– це бібліотека для роботи з MongoDB, у терміналі пишемо команду:

```
npm i mongoose
```

Підключення до БД можна переглянути в Додаток Н.

Для забезпечення безпеки та захисту конфіденційності користувачів використаємо бібліотеки auth.js та bcrypt.js.

Auth.js [21] надає набір інструментів для реалізації аутентифікації та авторизації користувачів. Бібліотека допомагає реалізувати безпечну систему ідентифікації користувачів у додатку. Команда для завантаження:

```
npm i next-auth
```

Bcrypt.js [22] дозволяє забезпечити безпечне зберігання та обробку паролів користувачів. Бібліотека використовує алгоритм хешування паролів. Команда для завантаження:

```
npm i bcryptjs
```

Виконувавши дії, такі як зареєструватись та опублікувати пост, в MongoDB бачимо дані (рис. 3.11).

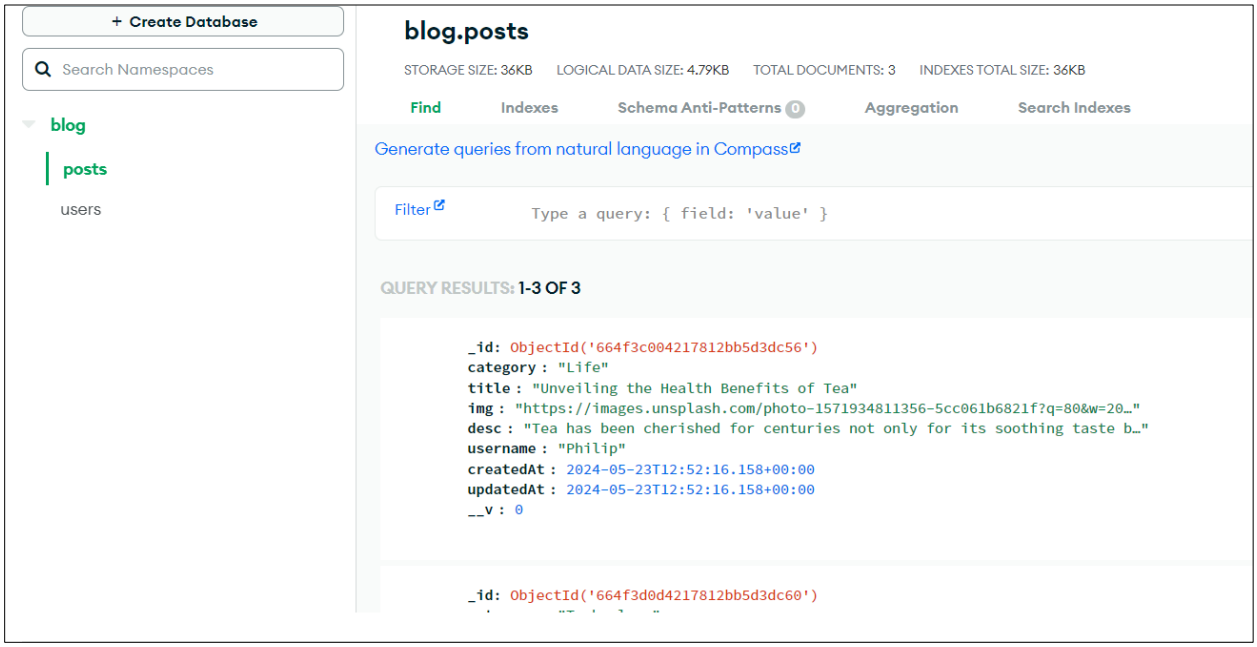

Рисунок 3.11 – Дані в MongoDB

Код серверної частини можна побачити Додаток Н – Додаток Т.

3.7 Розробка розсилки

Для початку потрібно завантажити react-mailchimp-subscribe [23]. У терміналі пишемо команду:

```
npm i react-mailchimp-subscribe
```

Потрібно отримати унікальне посилання, яке можна отримати після реєстрації в Mailchimp [14], переходам по сторінкам “Audience” – “Signup forms” – “Embedded form”, та натиском кнопки “Continue”, після цього відкривається вікно, де копіюємо посилання (рис 3.12).

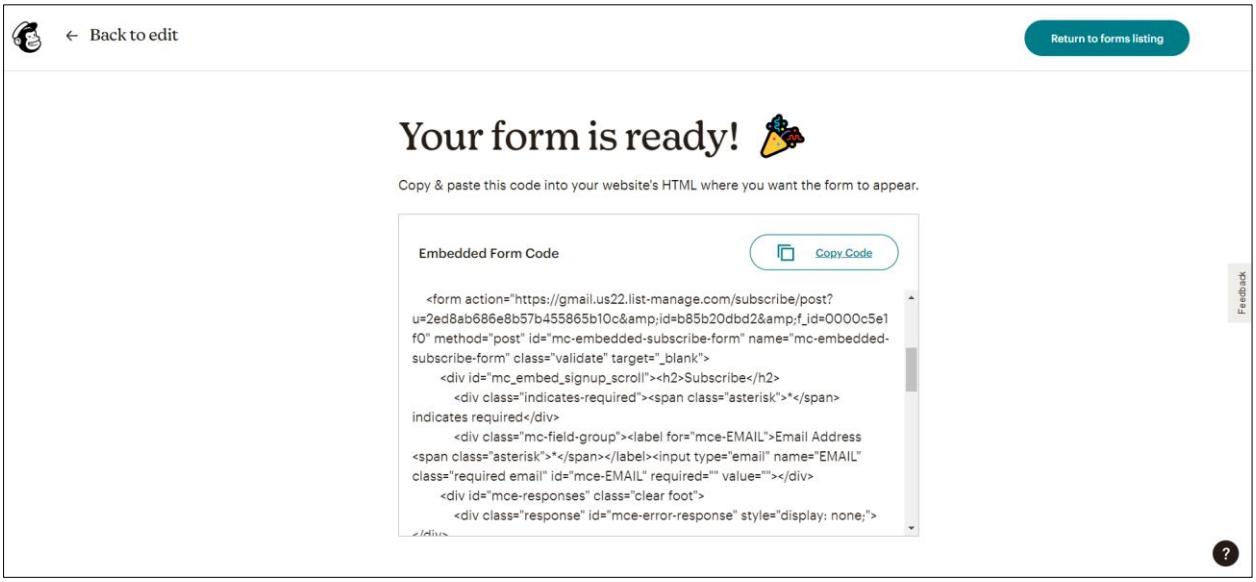

Рисунок 3.12 – Унікальне посилання

Код для створення розсилки в Додаток Ж.

Наступним кроком – створення листа, в Mailchimp переходимо на сторінку “ Campaigns” натискаємо на кнопку “Create” потім на “ Regular email”, далі налаштовуємо самостійно дизайн та натискаємо “Send”.

На сторінці сайту, коли користувач підпишиться на розсилку, на пошту прийде лист (рис 3.13).

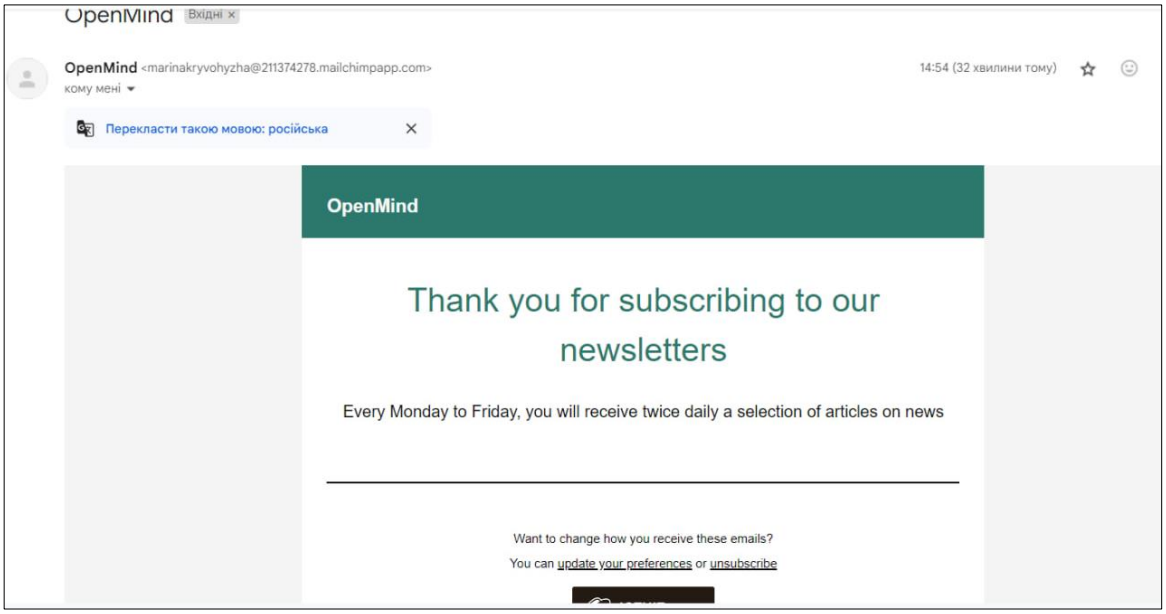

Рисунок 3.13 – Лист

На цьому розробка сайту-блогу "OpenMind", який задовольняє потреби користувачів у вільному обміні ідеями та знаннями є завершеною.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено сайт-блог "OpenMind", який дозволяє користувачам ділитися своїми думками та ідеями.

На початковому етапі було проведено аналіз аналогічних рішень, щоб зрозуміти їхні сильні та слабкі сторони, це допомогло визначити основні вимоги та оптимальний дизайн сайту-блогу. Далі було розроблено дизайн, який враховував сучасні тенденції та потреби користувачів. Для створення інтерфейсу було обрано фреймворк Next.js.

Для зберігання даних про користувачів та постів була обрана БД MongoDB, що забезпечила ефективну роботу та швидкий доступ до інформації. Крім того, для забезпечення безпеки та конфіденційності даних були використані бібліотеки Auth.js та Bcrypt.js для реалізації аутентифікації та шифрування паролів.

Для забезпечення зручного способу підписки був використаний сервіс Mailchimp.

Увесь код був збережений на репозиторії GitHub Desktop, що дозволило ефективно керувати версіями.

Отже, у кінці процесу був отриманий готовий сайт-блог "OpenMind", який задовольняє потреби користувачів у вільному обміні ідеями та знаннями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. From online filter to web format: Articulating materiality and meaning in the early history of blogs. URL: https://journals.sagepub.com/doi/full/10.1177/0306312711420190 (дата звернення: 07.05.2024).

2. Shuai Qinghong, Li Zhongjun. E-Commerce Industry Chain: Theory and Practice. - Springer, 2023. — 384 p.

3. Сайт-блог Medium. URL: https://medium.com/ (дата звернення: 9.05.2024).

4. Сайт-блог Dev.to. URL: https://dev.to/ (дата звернення: 08.05.2024).

5. Сайт-блог Steemit. URL: https://steemit.com/ (дата звернення: 10.05.2024).

6. Сайт-блог Hubpages. URL: https://discover.hubpages.com/ (дата звернення: 08.05.2024).

7. Сайт-блог Vocal. URL: https://vocal.media/ (дата звернення: 09.05.2024).

8. Сайт Next.js. URL: https://nextjs.org/ (дата звернення: 10.05.2024).

9. Wieruch Robin. The Road to React: The React.js in JavaScript Book. - Lean Publishing, 2024. — 416 p.

10. Di Francesco H. JavaScript Design Patterns: Deliver fast and efficient production-grade JavaScript applications at scale. - Birmingham: Packt Publishing, 2024. — 300 p.

11. Gagniuc P.A. Coding Examples from Simple to Complex: Applications in JavaScript. - Springer, 2024. - 242 p.

12. Stack Overflow Developer Survey 2022. URL: https://survey.stackoverflow.co/2022#most-popular-technologies-webframe-prof/ (дата звернення: 10.05.2024).

13. Most used web frameworks among developers worldwide, as of 2023. URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (дата звернення: 10.05.2024).

14. Сайт розсилки Mailchimp. - https://mailchimp.com/ (дата звернення: 11.05.2024).

15. Сайт GitHub. URL: https://github.com/ (дата звернення: 11.05.2024).

16. Сайт MongoDB. URL: https://www.mongodb.com/ (дата звернення: 12.05.2024).

17. Done Paul. Practical MongoDB Aggregations: The official guide to developing optimal aggregation pipelines with MongoDB 7.0. - Packt Publishing, 2023. — 313 p.

18. Miller Brian D., Ackerman Jason. Principles of Web Design. - 3rd Edition. — Allworth, 2022. — 280 p.

19. Сайт Node.js. URL: https://nodejs.org/en (дата звернення: 13.04.2024).

20. Сайт Mongoose. URL: https://mongoosejs.com/ (дата звернення: 14.05.2024).

21. Сайт Authjs. URL: https://authjs.dev/ (дата звернення: 15.05.2024).

22. Bcrypt.js – npm URL: https://www.npmjs.com/package/bcryptjs (дата звернення: 17.05.2024).

23. React-mailchimp-subscribe. URL: https://www.npmjs.com/package/react-mailchimp-subscribe (дата звернення: 19.05.2024).

Додаток А

Layout.js:

```
import "./globals.css";
import Navbar from "./components/navbar/navbar";
import Footer from "./components/footer/footer";
import AuthProvider from "./components/authProvider/authProvider";

export const metadata = {
  title: "OpenMind",
  description: "page",
};

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body >
        <AuthProvider>
          <div className="container center-container">
            <Navbar />
            {children}
          </div>
          <div className="footer-back">
            <div className="container">
              <Footer />
            </div>
          </div>
        </AuthProvider>
      </body>
    </html>
  );
}
```

Додаток Б

Global.css:

```
* {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
}

:root {
  --main-colour: #2C786C;
  --title-colour: #292929;
  --text-colour: #485656;
  --light-colour: #ffffff;
  --small_text-colour: #5D7373;
  --background-card: #fff;
  --border-card: solid #D0D0D0 1px;
  --border-btn: solid #2C786C 1px;
  --border-main-colour: solid #2C786C 1px;
}

html,
body {
  max-width: 100vw;
  overflow-x: hidden;
  background-color: var(--light-colour);
  color: var(--text-colour);
  font-family: "Open Sans", sans-serif;
  font-weight: 400;
  letter-spacing: 0.5px;
}

a {
  text-decoration: none;
  color: inherit;
}

.container {
  max-width: 1224px;
  margin: 0 auto;
  padding: 24px 0;
  display: flex;
  flex-direction: column;
}

.center-container {
  min-height: 100vh;
}

.footer-back {
```

```
  color: var(--light-colour);
  background-color: var(--main-colour);
}
```

Додаток В

Папка app.

Page.jsx:

```
"use client";
import Image from "next/image";
import styles from "./page.module.css";
import Button from "./components/button/button";
import Link from "next/link";
import { IoEyeSharp } from "react-icons/io5";
import { FaHeart } from "react-icons/fa";
import { RxAvatar } from "react-icons/rx";
import { FaPerson } from "react-icons/fa6";
import { MdOutlineWork } from "react-icons/md";
import { GrTechnology } from "react-icons/gr";
import { HiMiniComputerDesktop } from "react-icons/hi2";
import { IoPeople } from "react-icons/io5";
import { BiWorld } from "react-icons/bi";
import { BsFillMegaphoneFill } from "react-icons/bs";

const categories = [
    {
        href: "/category/Life",
        icon: <FaHeart />,
        title: "Life"
    },
    {
        href: "/category/Self Improvement",
        icon: <FaPerson />,
        title: "Self Improvement"
    },
    {
        href: "/category/Work",
        icon: <MdOutlineWork />,
        title: "Work"
    },
    {
        href: "/category/Technology",
        icon: <GrTechnology />,
        title: "Technology"
    },
    {
        href: "/category/Software Development",
        icon: <HiMiniComputerDesktop />,
        title: "Software Development"
    },
    {
        href: "/category/Media",
        icon: <BsFillMegaphoneFill />,
```

```
        title: "Media"
    },
    {
        href: "/category/Culture",
        icon: <IoPeople />,
        title: "Culture"
    },
    {
        href: "/category/World",
        icon: <BiWorld />,
        title: "World"
    }
];

async function getData() {
    const res = await fetch("http://localhost:3000/api/posts", {
        cache: "no-store",
    });

    if (!res.ok) {
        throw new Error("Failed to fetch data");
    }
    return res.json();
}

export default async function Home() {
    const data = await getData();
    const popularPosts = data.slice(0, 3);
    return (
        <div className={styles.container}>
            <div className={styles.text_block}>
                <h1 className={styles.titel}>Open your mind to new
possibilities</h1>
                <p className={styles.subtitle}>Share your thoughts and knowledge
with the world</p>
                <Button url="/dashboard/add" text="Start"></Button>
            </div>

            <h3 className={styles.titel_h3}>Our posts</h3>
            <div className={styles.cards_container}>
                {popularPosts.map((item) => (
                    <Link href={`/category/topic/${item._id}`} key={item._id}>
                        <div className={styles.card}>
                            <Image
                                src={item.img}
                                width={500}
                                height={500}
                                alt="photo"
                                className={styles.img}>
                            </Image>
                            <p className={styles.author}>
```

```
                                {item.username}
                            </p>
                            <h4 className={styles.title_card}>{item.title}</h4>
                        </div>
                    </Link>
                ))}
            </div>

            <h3 className={styles.titel_h3}>Choose your category</h3>
            <div className={styles.category_content}>
                {categories.map((category, index) => (
                    <Link key={index} href={category.href} >
                        <div className={styles.category_card}>
                            <h4
className={styles.title_card_category}>{category.icon}{category.title}</h4>
                        </div>
                    </Link>
                ))}
            </div>
        </div>
    );
}
```

Page.module.css:

```
.container {
    margin: 32px 0;
}

.titel {
    font-size: 48px;
    color: var(--title-colour);
    font-weight: 400;
}

.subtitle {
    font-size: 16px;
}

.text_block {
    display: flex;
    flex-direction: column;
    gap: 16px;
    text-align: center;
}

.titel_h3 {
    font-size: 24px;
    color: var(--title-colour);
    margin: 32px 0 24px 0;
    font-weight: 400;
```

```
}

.cards_container {
    margin-top: 32px;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 24px;
}

.img {
    width: 100%;
    height: 220px;
    object-fit: cover;
    border-radius: 16px;
}

.card {
    border: var(--border-card);
    background: var(--background-card);
    padding: 32px;
    border-radius: 16px;
    height: 100%;

    display: flex;
    flex-direction: column;
    justify-content: center;
    gap: 16px;
}

.title_card {
    font-size: 20px;
    font-weight: 400;
    color: var(--title-colour);
}

.author {
    display: flex;
    align-items: center;
    gap: 4px;
}

.author {
    color: var(--small_text-colour);
}


.category_content {
    margin: 32px auto;
    display: flex;
    width: 720px;
    flex-wrap: wrap;
```

```
    justify-content: center;
    gap: 16px;
}

.category_card {
    border: var(--border-card);
    background: var(--background-card);
    padding: 16px;
    border-radius: 16px;
}

.title_card_category {
    display: flex;
    gap: 4px;
    font-size: 16px;
    font-weight: 400;
    color: var(--title-colour);
}
```

Додаток Г

Папка about.

Page.jsx:

```
import React from "react";
import styles from './page.module.css'
import Button from "../components/button/button";

const About = () => {
    return (
        <div className={styles.container}>

            <h2 className={styles.titel}>About</h2>
            <p>At OpenMind, we believe in the power of diverse perspectives and
                the value of shared experiences. Our platform is designed to be a
                welcoming space where everyone can find something that resonates
                with them. Whether you're interested in life, self improvement,
                work, technology, media, or any other topic, our blog is a place
                where you can explore a wide range of subjects and engage with
                thoughtful content.
            </p>
            <p>
                Our mission is simple: to create a community where ideas can flow
freely,
                and where every voice matters. <span
className={styles.accent}>OpenMind is more than just a blog—it's a
                    platform for connection, inspiration, and discovery.</span>
Here, you can read
                articles and opinions from people of all backgrounds and
experiences,
                each bringing their unique insights to the table. And if you're
interested
                in sharing your own thoughts, we encourage you to contribute!
Whether
                you're a seasoned writer or just starting, <span span
className={styles.accent}>it is the perfect place
                    to express yourself.</span>
            </p>
            <p>
                <span span className={styles.accent}>We value open-mindedness and
inclusivity.</span> We believe that great conversations
                happen when people are open to different viewpoints, and that's
exactly what
                we aim to foster. Our community is built on respect, understanding,
and a
                shared curiosity about the world around us.
            </p>
            <p span className={styles.accent}>Let's explore the world of ideas
together!</p>
```

```
            <Button url="/about" text="Write"></Button>
        </div>
    )
}

export default About;
```

Page.module.css:

```
.accent {
    color: var(--main-colour);
    font-weight: 600;
}

.container {
    margin: 32px 0;
}

.titel {
    font-size: 48px;
    color: var(--title-colour);
    font-weight: 400;
}

.container p {
    margin: 24px 0px;
    font-size: 18px;
    line-height: 200%;
}
```

Додаток Ґ

Папка category.

Page.jsx:

```
import React from "react";
import Link from "next/link";
import styles from './page.module.css'
import { FaHeart } from "react-icons/fa";
import { FaPerson } from "react-icons/fa6";
import { MdOutlineWork } from "react-icons/md";
import { GrTechnology } from "react-icons/gr";
import { HiMiniComputerDesktop } from "react-icons/hi2";
import { IoPeople } from "react-icons/io5";
import { BiWorld } from "react-icons/bi";
import { BsFillMegaphoneFill } from "react-icons/bs";

const categories = [
    {
        href: "/category/Life",
        icon: <FaHeart />,
        title: "Life",
        description: "Food, Home, Health, Family, Relationships, Pets"
    },
    {
        href: "/category/Self Improvement",
        icon: <FaPerson />,
        title: "Self Improvement",
        description: "Mental Health, Productivity, Mindfulness"
    },
    {
        href: "/category/Work",
        icon: <MdOutlineWork />,
        title: "Work",
        description: "Marketing, Business, Leadership, Remote Work"
    },
    {
        href: "/category/Technology",
        icon: <GrTechnology />,
        title: "Technology",
        description: "Blockchain, Data Science, Gadgets, Security, Artificial
Intelligence"
    },
    {
        href: "/category/Software Development",
        icon: <HiMiniComputerDesktop />,
        title: "Software Development",
        description: "Programming, Programming Languages, Dev Ops, Operating
Systems"
    },
```

```
    {
        href: "/category/Media",
        icon: <BsFillMegaphoneFill />,
        title: "Media",
        description: "Writing, Art, Gaming, Humor, Movies, Music, Photography,
Television"
    },
    {
        href: "/category/Culture",
        icon: <IoPeople />,
        title: "Culture",
        description: "Philosophy, Religion, Spirituality, Fashion, Beauty,
Language, Sports"
    },
    {
        href: "/category/World",
        icon: <BiWorld />,
        title: "World",
        description: "Cities, Nature, Travel"
    }
];

const Category = () => {
    return (
        <div className={styles.container}>
            <h2 className={styles.titel}>Category</h2>
            <div className={styles.cards_container}>
                {categories.map((category, index) => (
                    <Link key={index} href={category.href} >
                        <div className={styles.card}>
                            <h4
className={styles.title_card}>{category.icon}{category.title}</h4>
                            <p>{category.description}</p>
                        </div>
                    </Link>
                ))}
            </div>
        </div>
    );
};

export default Category;
```

Page.module.css:

```
.container {
    margin: 32px 0;
}
```

```css
.titel {
    font-size: 48px;
    color: var(--title-colour);
    font-weight: 400;
}

.cards_container {
    margin-top: 32px;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 24px;
}

.card {
    border: var(--border-card);
    background: var(--background-card);
    padding: 32px;
    border-radius: 16px;
    height: 100%;

    display: flex;
    flex-direction: column;
    justify-content: center;
    gap: 16px;
}

.title_card {
    display: flex;
    gap: 4px;
    font-size: 20px;
    font-weight: 400;
    color: var(--title-colour);
}
```

Додаток Д

Папка category/[topic].

Page.jsx:

```
import React from "react";
import styles from './page.module.css'
import Image from "next/image";
import Link from "next/link";
import { notFound } from "next/navigation";

async function getData(topic) {
    const res = await fetch(`http://localhost:3000/api/posts/${topic}`, {
        cache: "no-store",
    });
    if (!res.ok) {
        return notFound()
    }
    return res.json();
}

const Topic = async ({ params }) => {
    const data = await getData(params.topic);
    return (
        <div className={styles.container}>
            <h2 className={styles.titel}>{decodeURIComponent(params.topic)}</h2>
            <div className={styles.cards_container}>
                {data.map((item) => (
                    <Link href={`/category/${params.topic}/${item._id}`}
key={item._id}>
                        <div className={styles.card}>
                            <Image
                                src={item.img}
                                width={500}
                                height={500}
                                alt="photo"
                                className={styles.img}>
                            </Image>
                            <p className={styles.author}>
                                {item.username}
                            </p>
                            <h4 className={styles.title_card}>{item.title}</h4>

                        </div>
                    </Link>
                ))}
            </div>
        </div>
    );
}
```

```
export default Topic;
```

Page.module.css:

```
.titel {
    font-size: 48px;
    color: var(--title-colour);
    font-weight: 400;
}

.cards_container {
    margin-top: 32px;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 24px;
}

.img {
    width: 100%;
    height: 220px;
    object-fit: cover;
    border-radius: 16px;
}

.card {
    border: var(--border-card);
    background: var(--background-card);
    padding: 32px;
    border-radius: 16px;
    height: 100%;

    display: flex;
    flex-direction: column;
    justify-content: center;
    gap: 16px;
}

.title_card {
    font-size: 20px;
    font-weight: 400;
    color: var(--title-colour);
}

.author {
    display: flex;
    align-items: center;
}

.author {
```

```
    color: var(--small_text-colour);
}
```

Додаток Е

Папка category/[topic]/[id].

Page.jsx:

```
import React from "react";
import styles from './page.module.css';
import Image from "next/image";
import { notFound } from "next/navigation";

async function getData(_id) {
    const res = await fetch(`http://localhost:3000/api/posts/category/${_id}`, {
        cache: "no-store",
    });

    if (!res.ok) {
        return notFound()
    }

    return res.json();
}

const Post = async ({ params }) => {
    const data = await getData(params.id);

    return (
        <div className={styles.container}>
            <h2 className={styles.titel}>{data.title}</h2>
            <div className={styles.author}>
                <p className={styles.avatar}>
                    {data.username}
                </p>
            </div>
            <Image
                src={data.img}
                width={500}
                height={500}
                alt="photo"
                className={styles.img}>
            </Image>
            <p className={styles.post_text}>
                {data.desc}
            </p>
        </div>
    );
};


export default Post;
```

Page.module.css:

```
.titel {
    font-size: 48px;
    color: var(--title-colour);
    font-weight: 400;
}

.container {
    margin: 32px auto;
    max-width: 720px;
    display: flex;
    flex-direction: column;
    gap: 24px;
}

.avatar {
    display: flex;
    align-items: center;
}

.author {
    color: var(--small_text-colour);
    font-size: 18px;
    display: flex;
    justify-content: space-between;
}

.img {
    width: 100%;
    height: 400px;
    object-fit: cover;
    border-radius: 16px;
}

.post_text {
    font-size: 18px;
    line-height: 200%;
}
```

Додаток Є

Папка components/button.

Page.jsx:

```jsx
import React from "react";
import styles from './button.module.css'
import Link from "next/link";

const Button = ({ text, url, width, inverted }) => {
    const buttonClass = inverted ? styles.inverted : styles.container;

    return (
        <Link href={url}>
            <button className={buttonClass} style={{ width }}>{text}</button>
        </Link>
    )
}

export default Button;
```

Page.module.css:

```css
.container {
    font-size: 16px;
    color: var(--light-colour);
    background-color: var(--main-colour);
    font-weight: 400;
    padding: 12px 40px;
    letter-spacing: 1.25px;
    border: none;
    border-radius: 50px;
    cursor: pointer;
}

.inverted {
    font-size: 16px;
    color: var(--main-colour);
    background-color: var(--light-colour);
    font-weight: 400;
    padding: 12px 40px;
    letter-spacing: 1.25px;
    border: var(--border-btn);
    border-radius: 50px;
    cursor: pointer;
}
```

Додаток Ж

Папка components/footer.

Page.jsx:

```
"use client"

import React from 'react';
import styles from './page.module.css';
import MailchimpSubscribe from 'react-mailchimp-subscribe';

const mailchimpURL = "https://gmail.us22.list-
manage.com/subscribe/post?u=2ed8ab686e8b57b455865b10c&id=b85b20dbd2";

const SimpleForm = ({ onSubmitted }) => {
    const [email, setEmail] = React.useState('');

    const handleSubmit = (e) => {
        e.preventDefault();
        onSubmitted({ EMAIL: email });
    };

    return (
        <form onSubmit={handleSubmit} className={styles.form}>
            <h3 className={styles.titel_h3} >Get our newsletter</h3>
            <div className={styles.subscribe}>
                <input
                    className={styles.input}
                    type="email"
                    placeholder="Enter email"
                    value={email}
                    onChange={(e) => setEmail(e.target.value)}
                    required
                />
                <button type="submit"
className={styles.inverted}>Subscribe</button>
            </div>
        </form>
    );
};

const Footer = () => (
    <div className={styles.container}>
        <p>©2023 OpenMind. All rights reserved.</p>
        <MailchimpSubscribe
            url={mailchimpURL}
            render={({ subscribe, status, message }) => (
                <div>
                    <SimpleForm onSubmitted={(formData) => subscribe(formData)} />
```

```
                    {status === 'sending' && <div
classNameName={styles.message_status}>Sending...</div>}
                    {status === 'error' && (
                        <div classNameName={styles.message_status}
dangerouslySetInnerHTML={{ __html: message }} />
                    )}
                    {status === 'success' && <div
classNameName={styles.message_status}>Subscribed!</div>}
                </div>
            )}
        />
    </div>
);

export default Footer;
```

Page.module.css:

```
.container {
    height: 164px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.form {
    display: flex;
    flex-direction: column;
    gap: 16px;
}

.inverted {
    font-size: 16px;
    color: var(--main-colour);
    background-color: var(--light-colour);
    font-weight: 400;
    padding: 12px 40px;
    letter-spacing: 1.25px;
    border: var(--border-btn);
    border-radius: 50px;
    cursor: pointer;
}

.input {
    padding: 12px 16px;
    background-color: var(--background-card);
    border: var(--border-card);
    border-radius: 10px;
```

```
    outline: none;
    width: 280px;
    margin-right: 8px;
}

.titel_h3 {
    font-size: 24px;
    font-weight: 400;
}

.input::placeholder {
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
    letter-spacing: 0.5px;
    font-size: 14px;
}

.input[type="text"] {
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
    font-size: 14px;
    color: var(--text-colour);
}

.message_status {
    color: var(--light-colour);
}
```

Додаток 3

Папка components/input.

Page.jsx:

```
import React from "react";
import styles from './input.module.css'


const Input = ({ type, id, placeholder }) => {

    return (
        <div className={styles.container}>
            <label htmlFor={id} className={styles.label}>
                {String(id).charAt(0).toUpperCase() +
String(id).slice(1).toLowerCase()}
            </label>
            <input type={type} id={id} placeholder={placeholder}
className={styles.input} />
        </div>
    )
}

export default Input;
```

Page.module.css:

```
.container {
    display: flex;
    flex-direction: column;
    gap: 8px;
}

.input {
    margin-bottom: 24px;
    padding: 12px 16px;
    background-color: var(--background-card);
    border: var(--border-card);
    border-radius: 10px;
    outline: none;
}

.input::placeholder {
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
    letter-spacing: 0.5px;
    font-size: 14px;
```

```
}

.input[type="text"] {
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
    font-size: 14px;
    color: var(--text-colour);
}
```

Додаток И

Папка components/menu.

Page.jsx:

```
import React from "react";
import styles from "./page.module.css";
import Link from "next/link";

const Menu = ({ username }) => {
    return (
        <div className={styles.menu}>
            <h3 className={styles.titel_h3}>
                {username}
            </h3>
            <Link href="./add">Add new post</Link>
            <Link href="./userposts">My posts</Link>
        </div>
    )
}

export default Menu;
```

Page.module.css:

```
.menu {
    display: flex;
    flex-direction: column;
    gap: 16px;
}

.titel_h3 {
    font-size: 24px;
    color: var(--title-colour);
    font-weight: 400;
    display: flex;
    align-items: center;
    gap: 4px;
    margin-bottom: 8px;
}
```

Додаток І

Папка components/navbar.

Page.jsx:

```
"use client";
import Link from "next/link";
import React from "react";
import styles from "./page.module.css";
import { signOut, useSession } from "next-auth/react";

const Navbar = () => {
    const session = useSession();
    return (
        <div className={styles.container}>
            <Link href="/." className={styles.logo}>OpenMind</Link>
            <div className={styles.links}>
                <Link href="/category">Category</Link>
                <Link href="/about">About</Link>
                <Link href="/dashboard/add">Dashboard</Link>
                {session.status === "authenticated" && (
                    <button className={styles.logout} onClick={signOut}>
                        Logout
                    </button>
                )}
            </div>
        </div>
    )
}

export default Navbar;
```

Page.module.css:

```
.container {
    height: 64px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.logo {
    color: var(--main-colour);
    font-weight: 600;
    font-size: 20px;
}

.links {
    display: flex;
    align-items: center;
    gap: 48px;
}

.logout {
```

```
    border: var(--border-main-colour);
    background-color: transparent;
    color: var(--main-colour);
    cursor: pointer;
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
    font-size: 16px;
    border-radius: 50px;
    padding: 8px 24px;
}
```

Додаток Ї

Папка components/authProvider.

authProvider.jsx:

```jsx
"use client"

import { SessionProvider } from "next-auth/react";
import React from "react";

const AuthProvider = ({ children }) => {
  return <SessionProvider>{children}</SessionProvider>;
};

export default AuthProvider;
```

Додаток Й

Папка dashboard/add.

Page.jsx:

```
"use client";
import React, { useState } from "react";
import useSWR from "swr";
import styles from "./page.module.css";
import Input from "@/app/components/input/input";
import Menu from "../../components/menu/page";
import { useSession } from "next-auth/react";
import { useRouter } from "next/navigation";

const categories = [
    "Life",
    "Self Improvement",
    "Work",
    "Technology",
    "Software Development",
    "Media",
    "Culture",
    "World"
];

const Add = () => {
    const session = useSession();
    const router = useRouter();
    const [errorMessage, setErrorMessage] = useState('');

    const fetcher = (...args) => fetch(...args).then((res) => res.json());

    const { data, mutate, error, isLoading } = useSWR(
        `/api/posts?username=${session?.data?.user.username}`,
        fetcher
    );
    console.log(session?.data?.user.username);

    if (session.status === "loading") {
        return <p>Loading...</p>;
    }

    if (session.status === "unauthenticated") {
        router?.push("/dashboard/signin");
    }

    const handleSubmit = async (e) => {
        e.preventDefault();
        const category = e.target[0].value;
        const title = e.target[1].value;
```

```
        const img = e.target[2].value;
        const desc = e.target[3].value;

        if (!category) {
            setErrorMessage('Category is required.');
            return;
        }

        if (title.length < 5) {
            setErrorMessage('Title must be at least 5 characters long.');
            return;
        }

        if (title.length > 50) {
            setErrorMessage('Title cannot be more than 50 characters long.');
            return;
        }

        if (!img.startsWith("https://images.unsplash.com/photo")) {
            setErrorMessage('Image URL must start with
"https://images.unsplash.com/photo".');
            return;
        }

        if (desc.length < 100) {
            setErrorMessage('Description must be at least 100 characters long.');
            return;
        }

        try {
            await fetch("/api/posts", {
                method: "POST",
                body: JSON.stringify({
                    category,
                    title,
                    desc,
                    img,
                    username: session.data.user.username,
                }),
            });
            mutate();
            e.target.reset();
            setErrorMessage('');
        } catch (err) {
            console.log(err);
        }
    };

    if (session.status === "authenticated") {
        return (
            <div className={styles.container}>
```

```
                <h2 className={styles.titel}>Dashboard</h2>
                <div className={styles.content}>
                    <Menu username={session.data.user.username} />
                    <form className={styles.form} onSubmit={handleSubmit}>
                        <h3 className={styles.titel_h3}>Add new post</h3>
                        <label className={styles.label}>Category</label>
                        <select defaultValue="" className={styles.category_select}>
                            <option value="" disabled>
                                Choose your category
                            </option>
                            {categories.map((category, index) => (
                                <option key={index} value={category}>
                                    {category}
                                </option>
                            ))}
                        </select>
                        <Input type="text" id="title" placeholder="Enter title" />
                        <Input type="text" id="image" placeholder="Enter image from
unsplash" />
                        <label className={styles.label}>Content</label>
                        <textarea
                            placeholder="Enter content"
                            className={styles.text_area}
                            rows="10"
                        ></textarea>
                        <button className={styles.btn}>Post</button>
                        {errorMessage && <p
className={styles.error}>{errorMessage}</p>}
                    </form>
                </div>
            </div>
        )
    }
}

export default Add;
```

Page.module.css:

```
.container {
    margin: 32px 0;
}

.content {
    margin: 32px 0;
    display: grid;
    grid-template-columns: 1fr 4fr;
}
```

```css
.titel {
    font-size: 48px;
    color: var(--title-colour);
    font-weight: 400;
}

.titel_h3 {
    font-size: 24px;
    color: var(--title-colour);
    margin-bottom: 24px;
    font-weight: 400;
}

.form {
    display: flex;
    flex-direction: column;
    width: 640px;
    margin: 0 auto;
}

.label {
    margin-bottom: 8px;
}

.text_area,
.category_select {
    margin-bottom: 24px;
    padding: 10px 16px;
    background-color: var(--background-card);
    border: var(--border-card);
    border-radius: 10px;
    outline: none;
}

.select-container option {
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
    letter-spacing: 0.5px;
    font-size: 14px;
}

.text_area::placeholder {
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
    letter-spacing: 0.5px;
    font-size: 14px;
}

.text_area {
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
```

```
    font-size: 14px;
    color: var(--text-colour);
}

.category_card {
    border: var(--border-card);
    background: var(--background-card);
    padding: 16px;
    border-radius: 16px;
}

.title_card_category {
    display: flex;
    gap: 4px;
    font-size: 16px;
    font-weight: 400;
    color: var(--title-colour);
}

.error {
    margin-top: 8px;
    color: #d60606;
    text-align: center;
}

.btn {
    font-size: 16px;
    color: var(--light-colour);
    background-color: var(--main-colour);
    font-weight: 400;
    padding: 12px 40px;
    letter-spacing: 1.25px;
    border: none;
    border-radius: 50px;
    cursor: pointer;
}
```

Додаток К

Папка dashboard/signin.

Page.jsx:

```
"use client"
import React from "react";
import styles from './page.module.css';
import Input from "@/app/components/input/input";
import Link from "next/link";
import { signIn, useSession } from "next-auth/react";
import { useRouter, useSearchParams } from "next/navigation";


const Signin = () => {
    const session = useSession();
    const router = useRouter();
    const params = useSearchParams();

    if (session.status === "loading") {
        return <p>Loading...</p>;
    }

    if (session.status === "authenticated") {
        router?.push("/dashboard/add");
    }

    const handleSubmit = (e) => {
        e.preventDefault();
        const email = e.target[0].value;
        const password = e.target[1].value;

        signIn("credentials", {
            email,
            password,
        });
    };

    return (
        <div className={styles.container}>
            <h3 className={styles.titel_h3}>Welcome back</h3>
            <form className={styles.form} onSubmit={handleSubmit}>
                <Input type="e-mail" id="email" placeholder="Enter email" />
                <Input type="password" id="password" placeholder="Enter password"
/>
                <button className={styles.btn}>Login</button>
                <p className={styles.under_form}>No account? <Link
href="/dashboard/signup">Create</Link></p>
            </form>
        </div>
```

```
    )
}

export default Signin;
```

Page.module.css:

```
.container {
    margin: 32px auto;
    display: flex;
    justify-content: center;
    flex-direction: column;
}

.titel_h3 {
    font-size: 24px;
    color: var(--title-colour);
    text-align: center;
    margin: 24px 0;
    font-weight: 400;
}

.form {
    display: flex;
    flex-direction: column;
    width: 500px;
}

.under_form {
    margin-top: 8px;
    text-align: center;
}

.btn {
    font-size: 16px;
    color: var(--light-colour);
    background-color: var(--main-colour);
    font-weight: 400;
    padding: 12px 40px;
    letter-spacing: 1.25px;
    border: none;
    border-radius: 50px;
    cursor: pointer;
}
```

Додаток Л

Папка dashboard/signup.

Page.jsx:

```
"use client"
import React, { useState } from "react";
import styles from './page.module.css';
import Input from "@/app/components/input/input";
import Link from "next/link";
import { useRouter } from "next/navigation";


const Signup = () => {
    const [error, setError] = useState(false);
    const router = useRouter();

    const handleSubmit = async (e) => {
        e.preventDefault();
        const username = e.target[0].value;
        const email = e.target[1].value;
        const password = e.target[2].value;

        if (username.length < 2) {
            setError("Username must be at least 2 characters long");
            return;
        }

        if (!email.includes("@") || !email.includes(".") || email.length < 7) {
            setError("Please enter a valid email");
            return;
        }

        if (password.length < 8) {
            setError("Password must be at least 8 characters long");
            return;
        }

        if (!/[A-Z]/.test(password) || !/\d/.test(password)) {
            setError("Password must contain at least one uppercase letter and one
digit");
            return;
        }

        try {
            const res = await fetch("/api/auth/signup", {
                method: "POST",
                headers: {
                    "Content-Type": "application/json",
                },
```

```
                body: JSON.stringify({
                    username,
                    email,
                    password,
                }),
            });
            res.status === 201 && router.push("/dashboard/signin?success=Account
has been created");
        } catch (err) {
            setError(err);
            console.log(err);
        }
    };
    return (
        <div className={styles.container}>
            <h3 className={styles.titel_h3}>Create account</h3>
            <form className={styles.form} onSubmit={handleSubmit}>
                <Input type="text" id="username" placeholder="Enter username" />
                <Input type="e-mail" id="email" placeholder="Enter email" />
                <Input type="password" id="password" placeholder="Enter password"
/>
                <button className={styles.btn}>Register</button>
                <p className={styles.under_form}>Already have an account? <Link
href="/dashboard/signin">Sign in</Link></p>
                {error && <p className={styles.error}>{error}</p>}
            </form>
        </div>
    )
}

export default Signup;
```

Page.module.css:

```
.container {
    margin: 32px auto;
    display: flex;
    justify-content: center;
    flex-direction: column;
}

.titel_h3 {
    font-size: 24px;
    color: var(--title-colour);
    text-align: center;
    margin: 24px 0;
    font-weight: 400;
}
```

```
.form {
    display: flex;
    flex-direction: column;
    width: 500px;
}

.under_form {
    margin-top: 8px;
    text-align: center;
}

.error {
    margin-top: 8px;
    color: #d60606;
    text-align: center;
}

.btn {
    font-size: 16px;
    color: var(--light-colour);
    background-color: var(--main-colour);
    font-weight: 400;
    padding: 12px 40px;
    letter-spacing: 1.25px;
    border: none;
    border-radius: 50px;
    cursor: pointer;
}
```

Додаток М

Папка dashboard/userposts.

Page.jsx:

```
"use client";
import React from "react";
import styles from "./page.module.css";
import Image from "next/image";
import Menu from "../../components/menu/page";
import useSWR from "swr";
import { useSession } from "next-auth/react";
import { useRouter } from "next/navigation";


const Userposts = () => {
    const session = useSession();
    const router = useRouter();
    const fetcher = (...args) => fetch(...args).then((res) => res.json());

    const { data, mutate, error, isLoading } = useSWR(
        `/api/posts?username=${session?.data?.user.username}`,
        fetcher
    );

    if (session.status === "loading") {
        return <p>Loading...</p>;
    }

    if (session.status === "unauthenticated") {
        router?.push("/dashboard/signin");
    }

    const handleDelete = async (id) => {
        try {
            await fetch(`/api/posts/categoty/${id}`, {
                method: "DELETE",
            });
            mutate();
        } catch (err) {
            console.log(err);
        }
    };

    if (session.status === "authenticated") {
        return (
            <div className={styles.container}>
                <h2 className={styles.titel}>Dashboard</h2>
                <div className={styles.content}>
                    <Menu username={session.data.user.username} />
```

```
                    <div> <h3 className={styles.titel_h3}>My posts</h3>
                        <div className={styles.cards_container}>
                            {isLoading
                                ? "loading"
                                : data?.map((post) => (
                                    <div className={styles.card} key={post._id}>
                                        <Image
                                            src={post.img}
                                            width={500}
                                            height={500}
                                            alt="photo"
                                            className={styles.img}>
                                        </Image>

                                        <h4
className={styles.title_card}>{post.title}</h4>
                                        <button
                                            className={styles.delete}
                                            onClick={() => handleDelete(post._id)}
                                        >
                                            Delete
                                        </button>
                                    </div>
                                ))}
                        </div>
                    </div>
                </div>
            </div>
        )
    }
}

export default Userposts;
```

Page.module.css:

```
.container {
    margin: 32px 0;
}

.content {
    margin: 32px 0;
    display: grid;
    grid-template-columns: 1fr 4fr;
}

.titel {
    font-size: 48px;
    color: var(--title-colour);
```

```
    font-weight: 400;
}

.titel_h3 {
    font-size: 24px;
    color: var(--title-colour);
    margin-bottom: 24px;
    font-weight: 400;
}

.cards_container {
    margin-top: 32px;
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 24px;
}

.img {
    width: 100%;
    height: 220px;
    object-fit: cover;
    border-radius: 16px;
}

.card {
    border: var(--border-card);
    background: var(--background-card);
    padding: 32px;
    border-radius: 16px;
    height: 100%;

    display: flex;
    flex-direction: column;
    justify-content: center;
    gap: 16px;
}

.title_card {
    font-size: 20px;
    font-weight: 400;
    color: var(--title-colour);
}

.delete {
    border: var(--border-main-colour);
    background-color: transparent;
    color: var(--main-colour);
    cursor: pointer;
    font-family: "Open Sans", sans-serif;
    font-weight: 400;
    font-size: 16px;
```

```
    border-radius: 50px;
    padding: 8px 24px;
}
```

Додаток Н

Папка utils.

db.js:

```
import mongoose from "mongoose";

const connectDB = async () => {
    if (mongoose.connections[0].readyState) {
        return true;
    }

    try {
        await
mongoose.connect('mongodb+srv://blog:blog@cluster0.vudsjgv.mongodb.net/blog');
    } catch (error) {
        throw new Error("Connection failed!");
    }
}

export default connectDB;
```

Додаток О

.env:

```
NEXTAUTH_SECRET = "1234567890"
NEXTAUTH_URL = "http://localhost:3000/"
```

Додаток П

Папка models.

postModel.js:

```
import { Schema, model, models } from "mongoose";

const postSchema = new Schema({
    category: {
        type: String,
        required: true,
    },
    title: {
        type: String,
        required: true,
    },
    img: {
        type: String,
        required: true,
    },
    desc: {
        type: String,
        required: true,
    },
    username: {
        type: String,
        required: true,
    }
}, { timestamps: true })

const PostModel = models.post || model('post', postSchema);

export default PostModel;
```

userModel.js:

```
import { Schema, model, models } from "mongoose";

const userSchema = new Schema(
    {
        username: {
            type: String,
            unique: true,
            required: true,
        },
        email: {
            type: String,
            unique: true,
            required: true,
```

```
        },
        password: {
            type: String,
            required: true,
        },
    }, { timestamps: true }
);

const UserModel = models.user || model('user', userSchema);

export default UserModel;
```

Додаток Р

Папка api/posts.

route.js:

```
import { NextResponse } from "next/server";
import connectDB from "@/app/utils/db";
import PostModel from "@/app/models/postModel";


export const GET = async (request) => {
    const url = new URL(request.url);
    const username = url.searchParams.get("username");

    try {
        await connectDB();
        const data = await PostModel.find(username && { username });
        console.log(data);
        return new NextResponse(JSON.stringify(data), { status: 200 });
    } catch (err) {
        return new NextResponse("Database Error", { status: 500 });
    }
};

export const POST = async (request) => {
    const body = await request.json();
    const newPost = new PostModel(body);

    try {
        await connectDB();
        await newPost.save();
        return new NextResponse("Post has been created", { status: 200 });
    } catch (err) {
        return new NextResponse("Database Error", { status: 500 });
    }
};
```

Додаток С

Папка api/posts/[category].

route.js:

```
import { NextResponse } from "next/server";
import connectDB from "@/app/utils/db";
import PostModel from "@/app/models/postModel";

export const GET = async (request, { params }) => {
    const { category } = params;

    try {
        await connectDB();
        const data = await PostModel.find({ category: category });

        return new NextResponse(JSON.stringify(data), { status: 200 });
    } catch (err) {
        console.error(err);
        return new NextResponse("Database Error", { status: 500 });
    }
};
```

Додаток У

Папка api/posts/[category]/[id].

route.js:

```
import { NextResponse } from "next/server";
import connectDB from "@/app/utils/db";
import PostModel from "@/app/models/postModel";

export const GET = async (request, { params }) => {
    const { id } = params;

    try {
        await connectDB();

        const data = await PostModel.findById(id);

        return new NextResponse(JSON.stringify(data), { status: 200 });
    } catch (err) {
        return new NextResponse("Database Error", { status: 500 });
    }
};

export const DELETE = async (request, { params }) => {
    const { id } = params;

    try {
        await connectDB();

        await PostModel.findByIdAndDelete(id);

        return new NextResponse("Post has been deleted", { status: 200 });
    } catch (err) {
        return new NextResponse("Database Error", { status: 500 });
    }
};
```

Додаток Ф

Папка api/auth/ [...nextauth].

route.js:

```
import NextAuth from "next-auth";
import connectDB from "@/app/utils/db";
import UserModel from "@/app/models/userModel";
import bcrypt from "bcryptjs";
import CredentialsProvider from "next-auth/providers/credentials";

const handler = NextAuth({
    providers: [
        CredentialsProvider({
            id: "credentials",
            name: "Credentials",
            async authorize(credentials) {
                await connectDB();

                try {
                    const user = await UserModel.findOne({
                        email: credentials.email,
                    });

                    if (user) {
                        const isPasswordCorrect = await bcrypt.compare(
                            credentials.password,
                            user.password
                        );

                        if (isPasswordCorrect) {
                            return user;
                        } else {
                            throw new Error("Wrong Credentials!");
                        }
                    } else {
                        throw new Error("User not found!");
                    }
                } catch (err) {
                    throw new Error(err);
                }
            },
        })
    ],
    pages: {
        error: "/dashboard/signup",
    },
    callbacks: {
        async jwt({ token, user }) {
            if (user) {
```

```
                token.username = user.username;

            }
            return token;
        },
        async session({ session, token }) {
            if (token) {

                session.user.username = token.username;

            }
            return session;
        },
    },
});

export { handler as GET, handler as POST };
```

Додаток Т

Папка api/auth/ signup.

route.js:

```
import UserModel from "@/app/models/userModel";
import connectDB from "@/app/utils/db";
import bcrypt from "bcryptjs";
import { NextResponse } from "next/server";

export const POST = async (request) => {
    const { username, email, password } = await request.json();

    await connectDB();

    const hashedPassword = await bcrypt.hash(password, 8);

    const newUser = new UserModel({
        username,
        email,
        password: hashedPassword,
    });

    try {
        await newUser.save();
        return new NextResponse("User has been created", {
            status: 201,
        });
    } catch (err) {
        return new NextResponse(err.message, {
            status: 500,
        });
    }
};
```