ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___» _____ 202_ р.

К В А Л І Ф І К А Ц І Й Н А Р О Б О Т А

на тему

«САЙТИ КАФЕДР НАВЧАЛЬНО-НАУКОВОГО ІНСТИТУТУ ДЕННОЇ ОСВІТИ: UA/UX ПРОЕКТУВАННЯ ТА ДИЗАЙН»

зі спеціальності 122 «Комп’ютерні науки»
освітня програма «Комп’ютерні науки»
ступеня магістра

Виконавець роботи Лазаренко Владислав Олександрович
_____ «___» _____ 202_ р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Ольховська Олена Володимирівна
_____ «___» _____ 202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2023

ЗАТВЕРДЖУЮ
Завідувач кафедри _____ Олена ОЛЬХОВСЬКА
«____» вересня 202__ р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Сайти кафедр навчально-наукового інституту денної освіти: UA/UX проектування та дизайн»
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістра
Прізвище, ім'я, по батькові Лазаренко Владислав Олександрович
Затверджена наказом ректора № ___-Н від «__» _____ 202__ р.
Термін подання студентом роботи «__» _____ 202__ р.
Вихідні дані до кваліфікаційної роботи: публікації з теми, веб-сайти сайтів кафедр.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)
ВСТУП
1. ПОСТАНОВКА ЗАВДАННЯ
2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1. Огляд конструкторів сайтів
2.2. Огляд способів самостійної розробки
2.3. Актуальність сайтів структурних підрозділів ВНЗ
3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Середовище розробки сайту
3.2. Технології розробки
3.3. Доступність веб-сайту
4. ПРАКТИЧНА ЧАСТИНА
4.1. Завантаження та встановлення необхідних інструментів
4.2. Налаштування проекту
4.3. Створення файлової структури
4.4. Підключення до CMS
4.5. Огляд виконаної роботи
4.6. Перевірка роботи сайту
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А
Перелік графічного матеріалу: 75 рисунків.

Консультанти розділів кваліфікаційної роботи

Розділ	ПІБ, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка завдання	Ольховська О. В.		
Інформаційний огляд	Ольховська О. В.		
Теоретична частина	Ольховська О. В.		
Практична реалізація	Ольховська О. В.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій і звітування викладачу		
3. Постановка завдання		
4. Інформаційний огляд джерел бібліотек та Інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доопрацювання (за необхідності), рецензування		

Дата видачі завдання «____»_____ 202__ р.

Здобувач вищої освіти Владислав Лазаренко

Науковий керівник к. ф.-м. н., Ольховська Олена Володимирівна

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____

(балів, оцінка за національною шкалою, оцінка за ЄКТС)

Протокол засідання ЕК № ____ від «____»_____ 202__ р.

Секретар ЕК _____ _____

(підпис) *(ініціали та прізвище)*

Затверджую
Зав. кафедри _____
к. ф.-м. н. Олена ОЛЬХОВСЬКА
«____» _____ 202__ р

Погоджено
Науковий керівник _____
к. ф.-м. н. Олена ОЛЬХОВСЬКА
«____» _____ 202__ р.

План

кваліфікаційної роботи ступеня магістра
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 «Комп'ютерні науки»
Лазаренко Владислав Олександрович
Прізвище, ім'я, по батькові

на тему «Сайти кафедр навчально-наукового інституту денної освіти: UA/UX проектування та дизайн»

ВСТУП
1. ПОСТАНОВКА ЗАВДАННЯ
2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1. Огляд конструкторів сайтів
2.2. Огляд способів самостійної розробки
2.3. Актуальність сайтів структурних підрозділів ВНЗ
3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Середовище розробки сайту
3.2. Технології розробки
3.3. Доступність веб-сайту
4. ПРАКТИЧНА ЧАСТИНА
4.1. Завантаження та встановлення необхідних інструментів
4.2. Налаштування проекту
4.3. Створення файлової структури
4.4. Підключення до бази даних
4.5. Огляд виконаної роботи
ВИСНОВОК
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Здобувач вищої освіти _____ Владислав ЛАЗАРЕНКО
«____» _____ 202__ р.

АНОТАЦІЯ

Записка: 81 стр., 75 рис., 1 додаток (на 13 стр.), 22 джерел.

Мета роботи – розробка сайту кафедри.

Об’єкт розробки – розробка сайту кафедри Навчально-наукового інституту денної освіти.

Методи дослідження та інформаційне забезпечення – середовище розробки Visual Studio Code, мова програмування TypeScript, JavaScript бібліотека React, React фреймворк Next.js, CSS препроцесор SASS, система управління контентом Contentful.

Розглянуто актуальні способи створення веб-сайтів та обрано найбільш оптимальний. Створено сайт кафедри Навчально-наукового інституту денної освіти.

Результати дослідження. Зроблено огляд методів розробки веб-сайтів, виявлено переваги та недоліки. Розроблено сайт кафедри Навчально-наукового інституту денної освіти.

Рекомендації щодо використання результатів дослідження. Рекомендується долучити програмний продукт до освітнього процесу.

Наукова апробація. За матеріалами роботи опубліковано тези (науково-практичний семінар «Комп’ютерні науки та інформаційні технології» (КНІТ-2023), Міжнародна молодіжна науково-практична інтернет-конференція «Наука і молодь в XXI сторіччі», 2023), написано статтю в журнал «Shipbuilding & marine infrastructure».

Ключові слова: веб-сайт, мова програмування TypeScript, Contentful, бібліотека React, SASS, фреймворк Next.js, сайт-конструктор, CMS.

ЗМІСТ

ВСТУП

Поява інтернету кардинально змінила способи отримання інформації та взаємодії з навколишнім світом [1]. Веб-сайти стали важливою частиною нашого життя, вони дозволяють нам отримувати доступ до великої кількості інформації, сервісу та ресурсів лише за декілька кліків. Веб-сайти стали настільки поширеними та впливовими, що змінюють правила взаємодії між людьми, проте не завжди на кращу сторону [2].

Актуальність цієї роботи визначається стрімким розвитком інформаційних технологій та їх впливом на освітні процеси. Створення цифрової платформи для кафедри є не лише зручним інструментом для комунікації між студентами та викладачами, але й важливим кроком для підвищення рівня доступності інформації та ефективності навчального процесу. Заклади вищої освіти покладаються на свої веб-сайти для поширення інформації про реєстрацію на курси, стажування, фінансову допомогу, програми обміну та студентські активності. Також веб-сайти є важливим засобом для реклами університету, їх використовують для презентації академічних програм, студентського життя.

Мета курсового проекту – розробка та впровадження сучасного прототипу веб-сайту кафедри Навчально-наукового інституту денної освіти, спрямованого на полегшення спілкування, обміну матеріалами та забезпечення доступу до актуальної інформації.

Об'єкт роботи - створення прототипу сайту кафедри Навчально-наукового інституту денної освіти, використовуючи сучасні методи розробки.

Предмет розробки — прототип сайту кафедри Навчально-наукового інституту денної освіти.

Методи розробки – для створення сайту було використано Next.js, React, TypeScript, Contentful , SASS.

Новизна роботи – при створенні проекту використовувалися новітні технології програмування та було оглянуто останні засоби розробки за допомогою штучного інтелекту та сайтів-конструкторів.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновку, списку використаної літератури та додатків.

Результати роботи представлено та апробовано на науково-практичному семінарі «Комп'ютерні науки та інформаційні технології» (КНІТ-2023), Міжнародній молодіжній науково-практичній інтернет-конференції «Наука і молодь в XXI сторіччі», 2023), студентській конференції ПУЕТ, в журналі «Shipbuilding & marine infrastructure».

Обсяг пояснювальної записки: 81 стор., в т.ч. основна частина - 62 стор., додатки - 13 стор., джерела - 22 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Відповідно до теми кваліфікаційної роботи поставлена задача створити прототип сайту Навчально-наукового інституту денної освіти, який можна буде використовувати як шаблон для всіх кафедр університету. Перед початком роботи, було отримано рекомендації щодо оптимального способу розробки.

Основні вимоги до сайту:

- **Респонсивний дизайн:** Сайт повинен коректно відображатися на різних пристроях (планшетах, смартфонах, комп'ютерах) та розмірах екранів.
- **Інтуїтивно зрозуміла навігація:** Логічне розміщення меню, простий пошук і швидкий доступ до головних розділів.
- **Безпека:** Захист від злому та збереження конфіденційності користувачів (наприклад, за допомогою SSL-шифрування для персональних даних).
- **Швидкість завантаження:** Оптимізація сайту для швидкого завантаження сторінок.
- **SEO-оптимізація:** Використання правильних ключових слів та описів, щоб забезпечити більшу видимість сайту у пошукових системах.
- **Якісний контент:** Інформація на сайті має бути цікавою, актуальною та корисною для цільової аудиторії.
- **Сумісність з браузерами:** Сайт повинен працювати на різних браузерах без відображення помилок.
- **Адаптивність:** Можливість швидко оновлювати та модернізувати сайт для відповідності потребам аудиторії, що постійно змінюються.
- **Соціальна інтеграція:** Наявність кнопок соціальних мереж для спільного використання контенту.
- **Відповідність правилам регулювання:** Врахування місцевих та міжнародних норм та законів, що стосуються інтернет-ресурсів.

Було поставлено завдання розмістити веб-сайт в інтернеті. Відповідно до дизайну сайт повинен мати 13 сторінок.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

Сайти можна створювати використовуючи засоби програмної розробки, сайти-конструктори та large language models (великі мовні моделі). Дані методи мають різні недоліки та переваги, розглянемо найголовніші з них.

Переваги конструкторів сайтів:

- Потребують нульових або мінімальних знань програмування.
- Простий та інтуїтивний інтерфейс розробки.
- Можна використовувати заздалегідь спроектовані шаблони.
- Відносно низька ціна, порівняно з послугами розробника.
- Полегшена робота з SEO оптимізацією.

Недоліки:

- Обмежена гнучкість розробки. Комплексний багатофункціональний сайт створити не вийде.
- Менший контроль технічних аспектів.
- Деякі конструктори можуть примусити розміщувати свої логотипи на сайті користувача.
- Швидкість роботи важче оптимізувати.

Переваги програмної розробки сайту:

- Максимальна гнучкість розробки. Креативність розробника нічим не обмежена.
- Повний контроль на технічними аспектами. Розробник сам обирає сервер, базу даних тощо.
- Просунута кастомізація. Чи буде дизайнерське рішення реалізоване залежить більше від навичок розробника ніж технологій.
- Більша кількість способі покращити швидкодію.

- Можна бути упевненим в захищеності.

Недоліки:

- Більша складність розробки та підтримки.
- Послуги розробника коштують більше ніж будь-який тарифний план конструкторів сайтів.
- Потрібно витрачати більше часу для внесення змін або розробки нових функції.
- Необхідні істотні знання програмування, які іноді втрачають актуальність. Потрібно регулярно вивчати щось нове.

Переваги великих мовних моделей:

- Мовні моделі можуть автоматизувати створення контенту.
- Зменшення трудовитрат.
- Збільшення ефективності та швидкості.

Недоліки:

- Мовна модель потребує навчання та налаштування.
- Складність коду, який генерує модель залежить від її навчання. Публічні моделі створюють тільки примітивні сайти.
- Модель має галюцинації. Може вигадувати інформацію, якої не існує.
- Відсутність самонавчання.
- Впевненість в неправильних рішеннях.

2.1 Огляд конструкторів сайтів

Розглянемо найпопулярніші конструктори сайтів.

Wix.com — міжнародна хмарна платформа для створення та розвитку інтернет-проектів, що дозволяє будувати професійні сайти і їхні мобільні версії на HTML5 з допомогою інструментів drag-and-drop. Пропонує широкий вибір шаблонів для різних типів сайту. Користувачі можуть обрати з-поміж 800 шаблонів, які можна кастомізувати за допомогою вбудованого редактору. Wix присутня можливість

покращення CEO. Дозволяє створювати безкоштовні сайти з обмеженими можливостями. Безкоштовні сайти міститимуть брендінг Wix. Платити потрібно тільки за поліпшення. Дехто критикує Wix за надлишкове покладання на шаблони там обмежену гнучкість проектування.

При створенні сайту можна обрати різноманітні компоненти (рис 2.1).

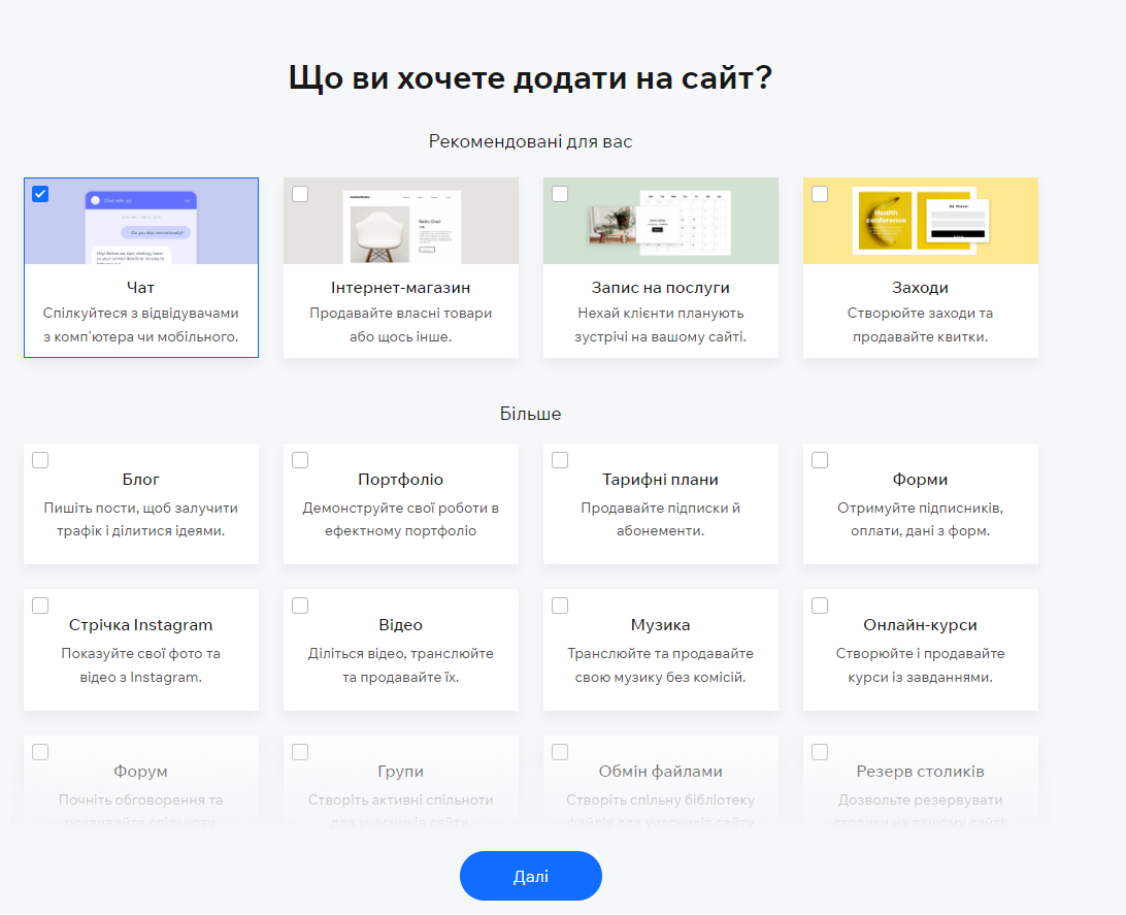

Рисунок 2.1 – Вибір компонентів

Можна створити сайт самотужки, використовуючи шаблон або Wix створить сайт самотужки за декілька хвилин (рис 2.2).

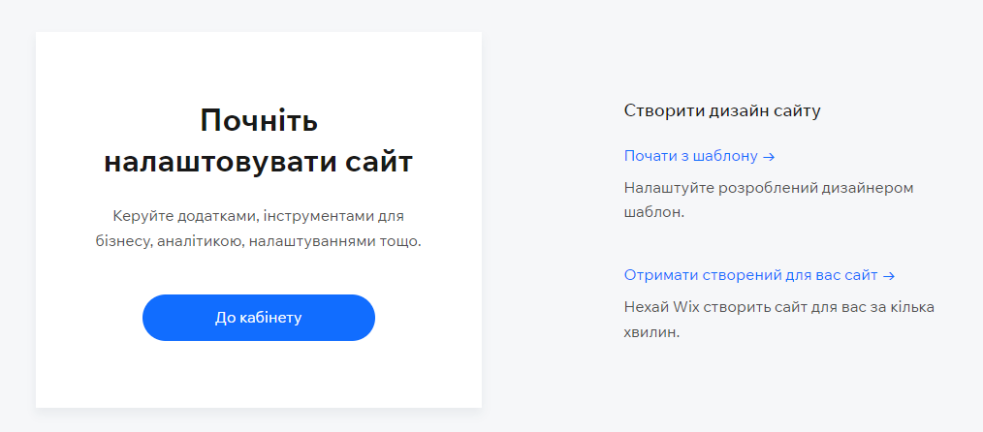

Рисунок 2.2 – Варіанті створення сайту

У наявності є якісні та деталізовані шаблони для шкіл, університетів та курсів (рис 2.3).

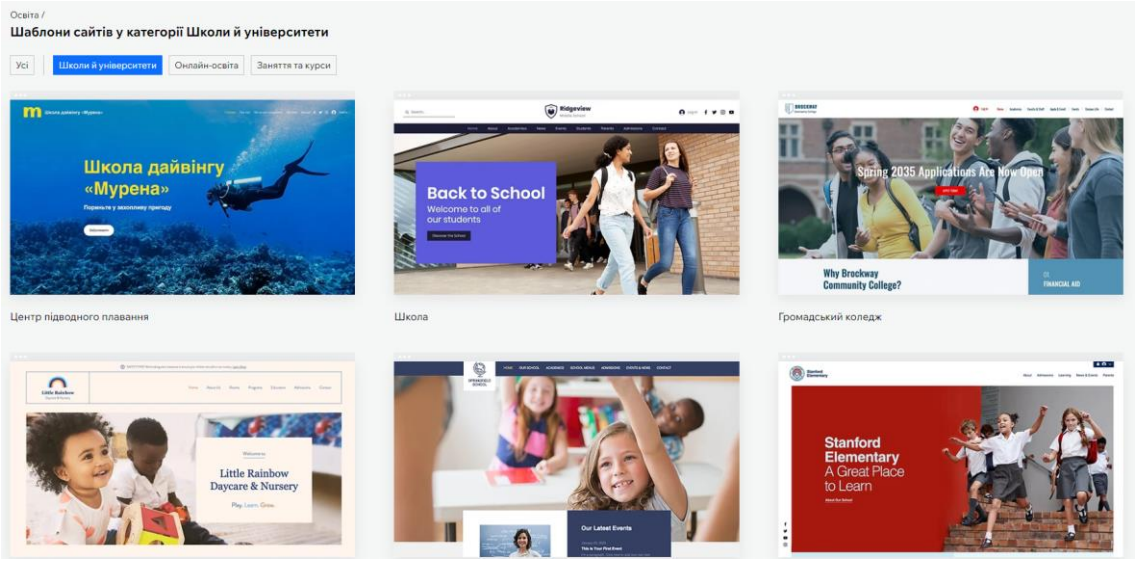

Рисунок 2.3 – Магазин шаблонів

Якщо була вибрана опція автоматичної генерації веб-сайту, то користувач отримає можливість обрати стиль (рис. 2.4).

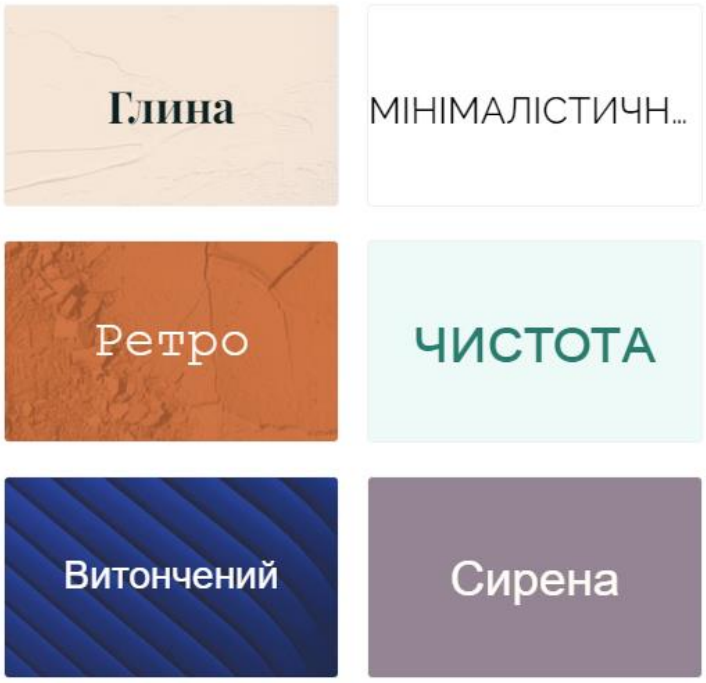

Рисунок 2.4 – Вибір стилю

Було обрано “Витончений” стиль. Wix згенерував декілька варіантів (рис 2.5).

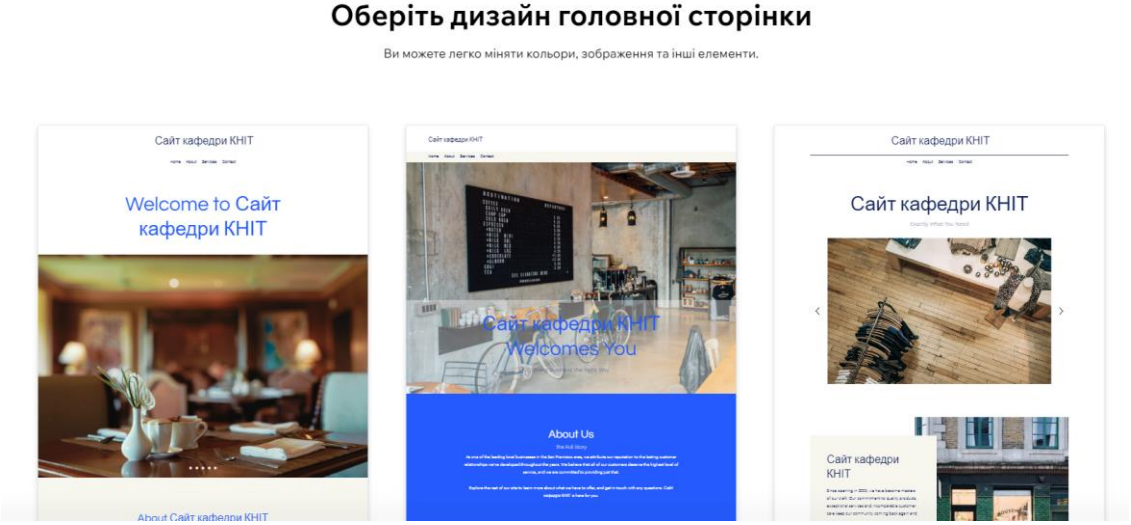

Рисунок 2.5 – Згенеровані варіанти

Було обрано третій стиль. Далі конструктор пропонує додати сторінки (рис 2.6).

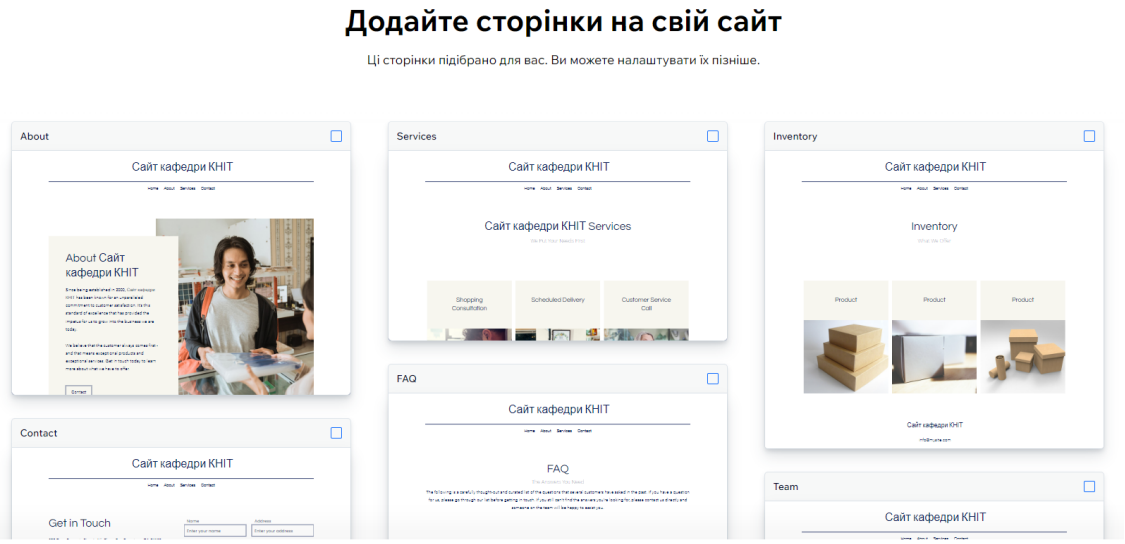

Рисунок 2.6 – Додавання сторінок

Після цього було згенеровано веб-сайт, який вже можна редагувати на власний розсуд (рис 2.7).

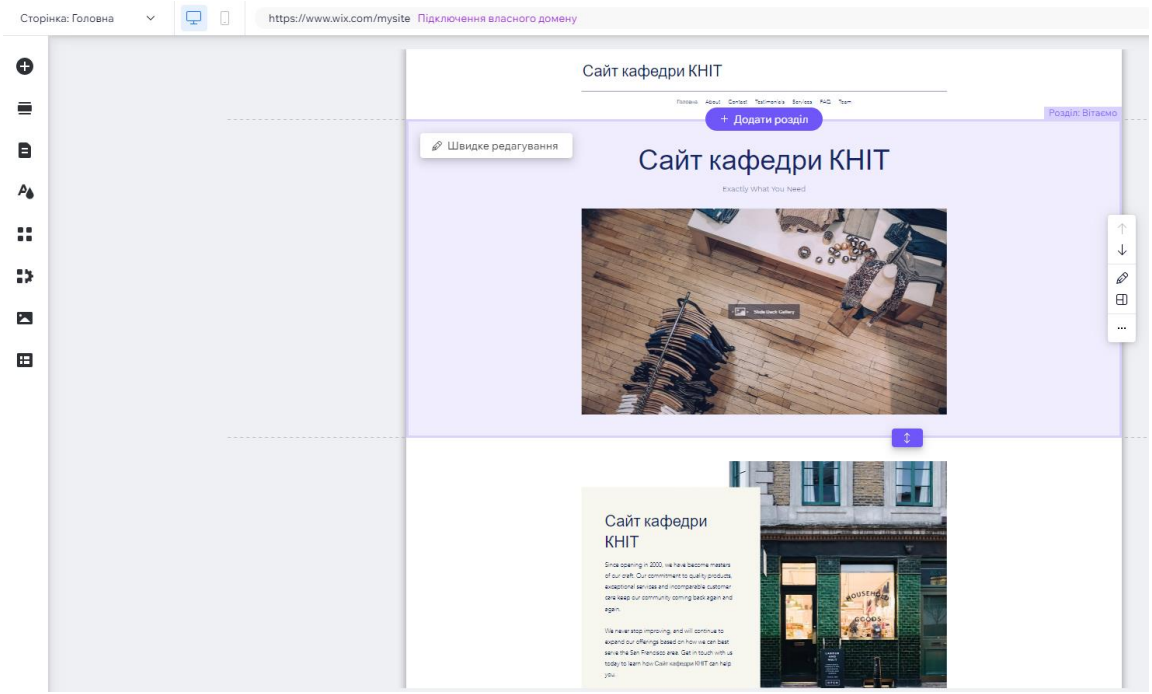

Рисунок 2.7 – Редактор сайту

Через редактор можна додавати загальнодоступні та бекенд файли (рис 2.8).

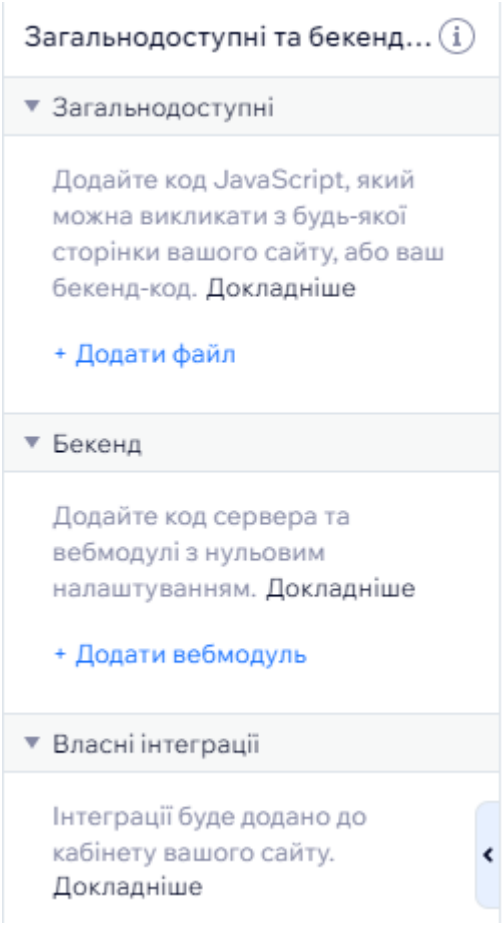

Рисунок 2.8 – Додавання загальнодоступних та бекенд файлів

Можна встановлювати пакунки коду (рис. 2.9).

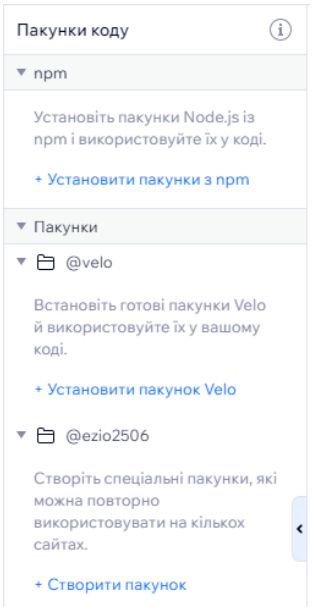

Рисунок 2.9 – Встановлення пакунків коду

Можна писати власний JavaScript код (рис 2.10).

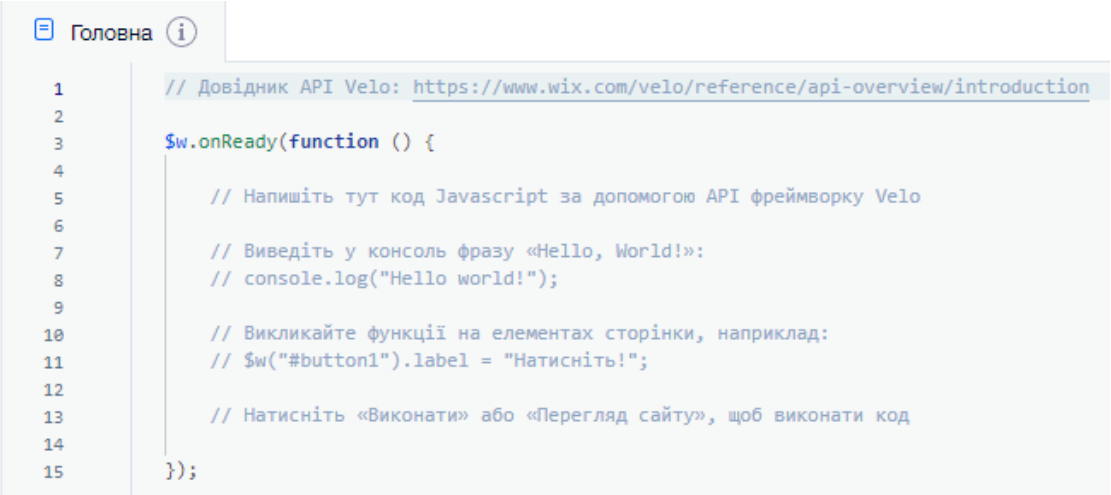

Рисунок 2.10 – Додавання JavaScript коду

Розглянемо наступний конструктор сайтів.

Framer – це просунута CMS, яка дозволяє і проектувати дизайн, і розробляти веб-сайти. За допомогою відповідного плагіну в магазині середовища проектування Figma, в якому був спроектований веб-сайт, можна створити сайт на його основі в Framer. Присутній безкоштовний план.

Так виглядає сторінка редагування сайту (рис 2.11).

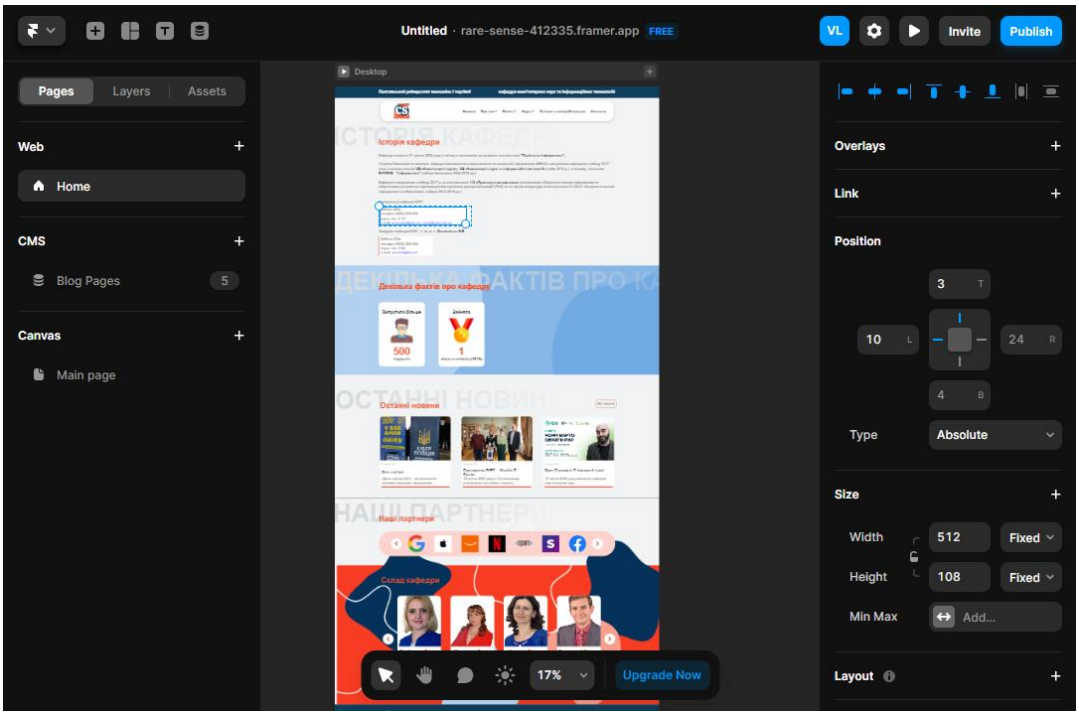

Рисунок 2.11 – Редагування сайту

На даній сторінці можна займатися проектуванням сайту, можливості якого не поступаються таким відомим середовищам проектування як Figma, Adobe XD, Sketch. Можна змінювати власне веб-сайт або вносити зміни до CMS.

Загалом, веб-сайт, який був створений автоматично на основі дизайн макету Figma, виглядає майже як на макеті, за винятком логотипу кафедри, який був деформований (рис. 2.12).

Рисунок 2.12 – Логотип кафедри

Та блоку наші партнери, на якому з’явилася заява тінь (рис. 2.13).

Рисунок 2.13 – Блок “Наші партнери”

2.2 Огляд мовних моделей

Розглянемо сервіс Durable. Durable використовує штучний інтелект для побудови веб-сайту без використання коду.

Створення веб-сайту починається з вибору типу бізнесу (рис. 2.14).

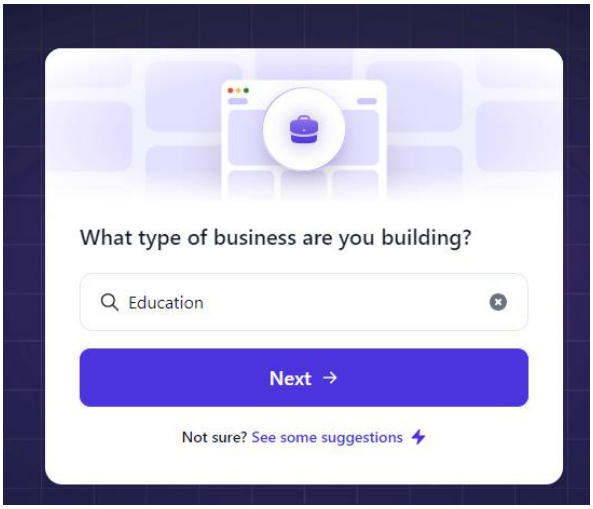

Рисунок 2.14 – Вибір типу бізнесу

Після цього потрібно обрати локацію (рис. 2.15).

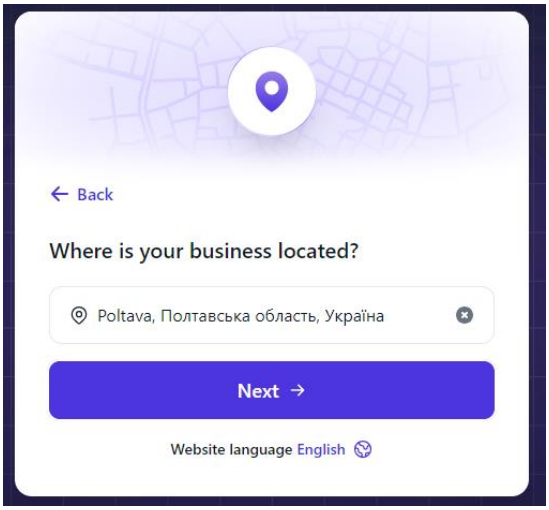

Рисунок 2.15 – Вибір локації

Після обирання всіх параметрів, було згенеровано веб-сайт (рис. 2.16).

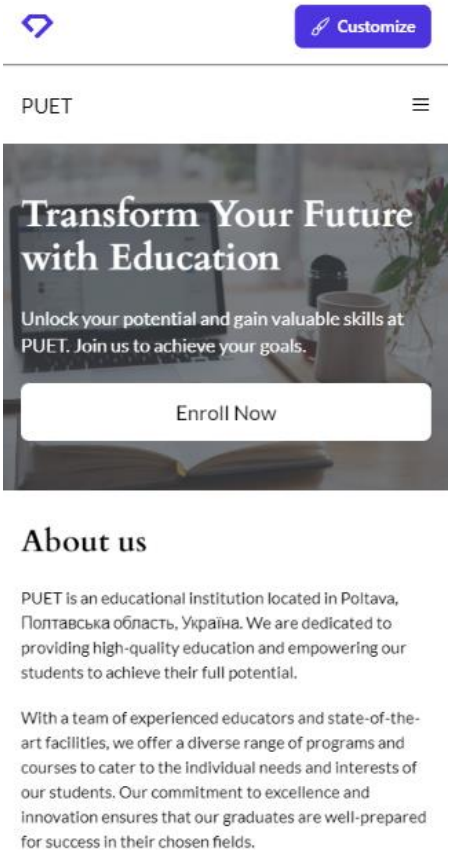

Рисунок 2.16 – Згенерований веб-сайт

Редактор дозволяє регенерувати та редагувати кожну секцію сайту (рис. 2.17).

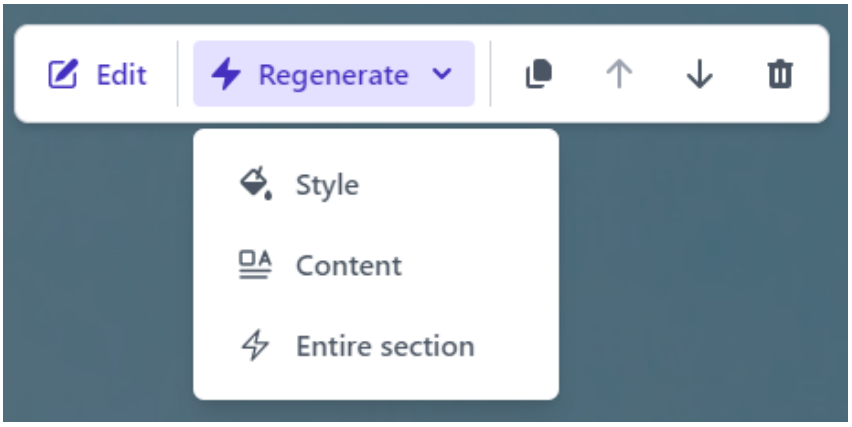

Рисунок 2.17 – Опції генерації секції

Загалом, сайт виглядає простим та поверховим, хоча він був згенерований швидко та просто. Для створення простих односторінкових веб-сайтів цього вистачить, але для комплексного сайту кафедри цього замало.

Перейдемо до більш різносторонньої мовної моделі, яка може не тільки писати програмний код веб-сайту, а й грати в шахи, писати статті, допомагати вчити мови та багато іншого. Але хоч він і вміє багато, він ні в чому не є майстром [3]. ChatGPT або Chat Generative Pre-trained Transformer – це чат-бот на базі великої мовної моделі, який розробляється компанією OpenAI. В останній версії було додано функцію розпізнавання вмісту зображення. ChatGPT може розуміти функціонал та дизайн веб-сайту, який намальований вручну в зошиті (рис. 2.18).

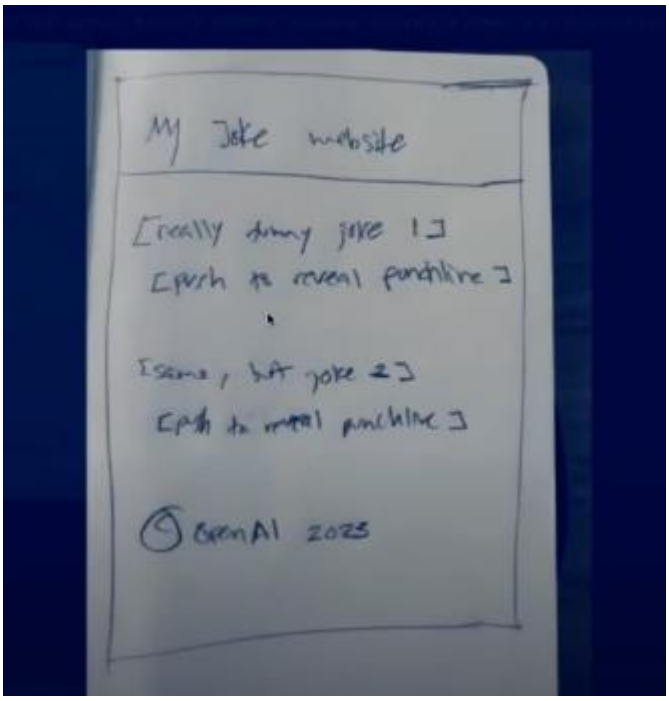

Рисунок 2.18 – Скетч веб-сайту

На папері намальований прототип веб-сайту, що показує жарти. Він повинен складатися з тексту різного типу та стилізованих кнопок, що будуть відображати текст жарту. ChatGPT впорався з цією задачею (рис. 2.19).

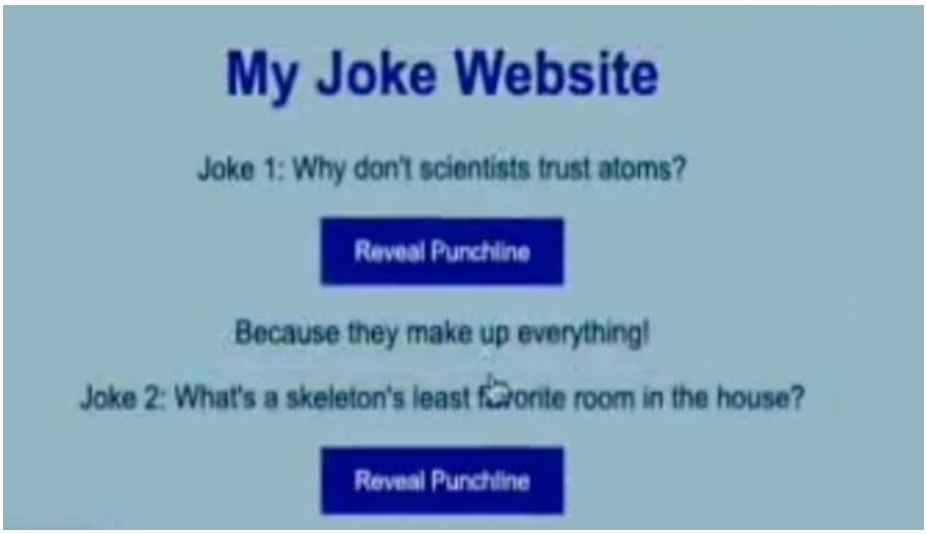

Рисунок 2.19 – Сайт створений ChatGPT

2.3 Огляд способів самостійної розробки

Головними будівельними блоками кожного багатофункціонального веб-сайту є HTML, CSS, JavaScript.

HTML - мова гіпертекстової розмітки, яку використовують для створення документів, що становлять фундамент кожного сайту. Першочергово, він був створений для розмітки наукових статей, але згодом набув більш широкого поширення. У ньому описується структура сайту, метаінформація про сайт та підключаються зовнішні файли, такі як CSS та JavaScript. Дозволяє робити базову стилізацію контенту, але для більш складних стилів слід використовувати CSS.

CSS – каскадна таблиця стилів, що використовується для стилізації HTML документу. Для зміни стилю елементу використовуються спеціальні селектори.

JavaScript – мова програмування, яка дозволяє отримати доступ до об’єктної моделі документа для зміни сторінки.

Будь-який веб-сайт в підсумку є просто набором HTML, CSS, JavaScript файлів. Тому для розробки веб-сайту можна використовувати тільки їх. Проте JavaScript код є досить комплексним та об’ємним. При написанні коду потрібно слідкувати за його ефективністю, не робити зайвих змін DOM. При розробці будь-

якого веб-сайту розробник повинен буде робити одні і ті ж речі, що є не дуже ефективним.

Для вирішення всіх цих проблем були створені фреймворки та бібліотеки. Фреймворк – це програмне середовище, в якому запускається написаний код. Він є більш складним, ніж бібліотека, бо має більший функціонал та сталі патерни розробки. Бібліотека – заздалегідь написаний код, який полегшує розробку. Зазвичай, бібліотеку додають до проекту з репозиторію npm.

Найпопулярнішими бібліотеками або фреймворками JS є Angular, React, Vue, Svelte, Next.js.

Angular – це відкритий JavaScript фреймворк, який розроблений та підтримується компанією Google. Створений в 2010 році, швидко набув популярності серед розробників для будівництва динамічних та односторінкових застосунків. Прийнято використовувати в парі з мовою програмування TypeScript.

Основні особливості Angular:

- Двостороннє зв'язування даних: Однією з ключових функцій AngularJS є можливість встановити двостороннє зв'язування даних між моделлю та представленням. Це означає, що будь-які зміни, внесені до моделі, автоматично відображаються в представленні, і навпаки. Це спрощує процес підтримки взаємодії даних та користувацького інтерфейсу і зменшує кількість шаблонного коду.
- Директиви: AngularJS вводить поняття директив, які є маркерами на елементах DOM і розширюють функціональність HTML. Директиви дозволяють створювати перевикористовувані компоненти та надавати власні поведінки елементам, що спрощує маніпулювання DOM і побудову інтерактивних інтерфейсів.
- Впровадження залежностей: AngularJS має вбудовану підтримку впровадження залежностей, що допомагає керувати залежностями між різними компонентами додатку. Впровадження залежностей сприяє модульності та тестовості, дозволяючи впроваджувати залежності, а не

включати їх як жорсткі залежності у коді. Це спрощує написання модульних тестів і відокремлює компоненти, зробивши їх більш перевикористовуваними.

- Шаблонування: AngularJS використовує HTML як мову шаблонів, що дозволяє розробникам писати декларативні шаблони, що описують структуру та поведінку користувацького інтерфейсу. Шаблони можуть містити вирази та директиви для динамічної генерації вмісту на основі даних. Цей підхід спрощує процес розробки і сприяє перевикористовуванню коду.
- Маршрутизація: AngularJS надає потужний механізм маршрутизації, який дозволяє розробникам створювати односторінкові додатки. Він дозволяє визначати різні маршрути та пов'язувати їх з шаблонами та контролерами, що спрощує навігацію між різними розділами додатка без перезавантаження всієї сторінки.
- Тестування: AngularJS наголошує на тестовості і надає засоби для написання модульних тестів та тестів "з кінця в кінець". Він має в своєму складі фреймворк для тестування під назвою "Karma" для запуску тестів та фреймворк під назвою "Protractor" для тестування AngularJS-додатків.
- Розширюваність: AngularJS дуже розширюваний, що дозволяє розробникам створювати власні директиви, фільтри, сервіси та модулі для виконання конкретних завдань. У нього є велика екосистема сторонніх бібліотек і модулів, які можна легко інтегрувати в AngularJS-додатки.

Переваги Angular:

- Найстаріший з популярних фреймворків, тому найбільш стабільний;
- Підтримується Google.

– Найкраще управління станом. На відміну від інших, не потрібно використовувати сторонні бібліотеки для ефективного управління станом всього додатку;
– Найбільша підтримка TypeScript. Такі фреймворки як Svelte та Vue мають проблеми з його підтримкою.

Недоліки:

– Кожен компонент розділений на три файли: CSS, HTML, TypeScript;
– Новий компонент потрібно реєструвати в файлі app.module.ts.
– Роздутий та важкий. Для реалізації навіть примітивних речей потрібно писати багато коду.
– Потребує багато часу для вивчення, значно більше, ніж будь-який інший фреймворк.

React - це відкрита JavaScript бібліотека, розроблена Facebook. Вона призначена для побудови користувацьких інтерфейсів (UI) веб-додатків. За допомогою React можна створювати компоненти, які оновлюються динамічно, коли змінюються дані.

Основні особливості:

– Компонентна структура: React базується на компонентній архітектурі, де користувацький інтерфейс розбивається на невеликі незалежні компоненти. Кожен компонент має свій власний стан (state) та властивості (props), які визначають його поведінку та вигляд. Це сприяє перевикористовуванню коду та зручності у розробці та підтримці додатків.
– Віртуальний DOM: React використовує віртуальний DOM (Document Object Model) для ефективного оновлення та маніпуляції вмісту сторінки. Замість безпосередньо взаємодіяти з реальним DOM, React працює з його копією у пам'яті, що дозволяє оптимізувати процеси оновлення та рендерингу, що призводить до покращення продуктивності додатків.

- JSX: React використовує JSX (розширений синтаксис JavaScript), який дозволяє писати шаблони UI безпосередньо в коді JavaScript. JSX поєднує можливості HTML та JavaScript, що спрощує створення компонентів та вбудовування динамічних даних у розмітку.
- Управління станом: React пропонує різні підходи до управління станом додатку. Найпоширенішим є використання концепції "однієї джерела правди" (Single Source of Truth) за допомогою збереження стану у верхніх компонентах і передачі його дочірнім компонентам через властивості.
- Розширюваність: React є дуже розширюваним, і його функціональність можна розширити за допомогою сторонніх бібліотек і розширень. Наприклад, бібліотека Redux широко використовується для управління станом у великих React-додатках.
- Реактивність: React дозволяє створювати реактивні інтерфейси, які миттєво реагують на зміни даних. При зміні стану або властивостей компонента React автоматично перерендерює тільки необхідні частини інтерфейсу, що полегшує розробку швидких та ефективних додатків.

Переваги:

- Має найбільшу підтримку та популярність серед розробників.
- Підтримується Facebook.
- Дозволяє писати більш лаконічний код, ніж будь-який інший фреймворк.
- Найкраща інтеграція JavaScript коду та розмітки.
- Найбільша гнучкість. Так як React це бібліотека, а не фреймворк, він надає більшу свободу використання патернів та допоміжних бібліотек.
- Можливість написання компоненту в одному файлі, використовуючи JSX розмітку, яка об’єднує в собі JavaScript та HTML та бібліотеку Styled components, яка дозволяє створювати та використовувати стилі в JavaScript файлі.

Недоліки:

- Неможливість використання директиви в HTML. Потрібно використовувати логічні оператори та методи масиву замість “if”, “for”, “while”.
- Складніші методи життєвого циклу.
- Єдиний з чотирьох фреймворків, який немає двостороннього зв’язку.

Vue - це прогресивний фреймворк JavaScript для розробки користувацьких інтерфейсів, що був створений працівником Google, який хотів мати простішу альтернативу Angular. Він часто порівнюється з Angular та React, але відрізняється своїм підходом до розробки та легкістю в освоєнні.

Переваги:

- Дозволяє писати весь код для компоненту в єдиному файлі.
- Хоча Vue є досить мінімалістичним, використовуючи middleware патерн можна додавати додаткові модулі.
- Двосторонній зв’язок.
- Функціональне та потужне управління станом.

Недоліки:

- Менша спільнота, ніж у Angular та React.
- Більший розмір кінцевого бандла.
- Немає корпоративної підтримки, тримається тільки завдяки ентузіастам.

Svelte - це фреймворк для розробки веб-інтерфейсів. Основною концепцією Svelte є те, що він перетворює код на чистий, оптимізований JavaScript в процесі компіляції, а не працює з віртуальним DOM під час виконання.

Основні особливості Svelte:

- Компіляція: Svelte використовує компіляцію для перетворення коду, написаного на мові Svelte, в оптимізований JavaScript під час збірки проекту. Це дозволяє генерувати дуже ефективний код без необхідності використання віртуального DOM.

- Реактивність: У Svelte є вбудована підтримка реактивності. Ви можете використовувати директиви, щоб зв'язувати дані зі шаблоном і автоматично оновлювати його при зміні даних.
- Компонентна архітектура: Svelte сприяє побудові додатків на основі компонентів. Ви можете створювати компоненти, які містять свій внутрішній стан, логіку та представлення, і повторно використовувати їх у своїй програмі.
- Мінімальний розмір бандла: Оскільки Svelte генерує оптимізований JavaScript під час компіляції, розмір кінцевого бандла може бути значно меншим порівняно з іншими фреймворками. Це дозволяє створювати швидкі та легкі додатки, що особливо важливо для мобільних пристроїв та обмежених мережевих умов.
- Швидкість виконання: Завдяки відсутності віртуального DOM і компіляції в процесі збірки, Svelte може забезпечити дуже швидку реакцію на зміни стану додатку та ефективну роботу зі шаблонами.

Переваги:

- За результатами опитування був визнаний найулюбленішими фреймворком.
- Підтримка анімацій.
- Працює швидше за основних конкурентів.
- Найпростіший для вивчення.
- Підтримується Vercel.

Недоліки:

- Незначна підтримка спільноти.
- Незручна та застаріла документація.
- Слабка підтримка TypeScript.

У загальному, всі популярні фреймворки є функціональними та потужними. Хоча вони відрізняються за швидкістю: Svelte є найшвидшим, а React - найповільнішим, все одно їхньої швидкості достатньо.

Для виконання даного проекту було обрано бібліотеку React, тому що вона має найкращу документацію, найбільшу спільноту, найзручніший синтаксис та використовується більшістю відомих сайтів.

2.4 Актуальність сайтів структурних підрозділів ВНЗ

У нинішній цифровий час присутність структурного підрозділу вищого навчального закладу онлайн є дуже важливою для комунікації зі студентами.

В умовах воєнного стану можливість дистанційного зв’язку є ще більш критичною, бо студенти можуть знаходитися не тільки в різних куточках України, а й світу. Студенти повинні мати змогу доєднатися до освітнього процесу з будь-якого місця, не ставлячи своє життя та безпеку під загрозу [4]. Веб-сайт може повідомити про заходи безпеки під час навчання, зміни до розкладу. Під час війни доступ до інформації може бути обмеженим, зв’язок з викладачами не стійкий. На сайті можуть бути розміщені онлайн ресурси для самонавчання або наданий доступ до бібліотеки чи лекційних матеріалів для самоосвіти.

Університети є динамічними інституціями, які вимагають регулярного оновлення інформації про курси, події, стажування, новини тощо. Наявність сайту полегшує донесення цієї інформації.

Пристрої, які мають доступ до інтернету з кожним роком стають все більш доступними, тому потреба в наявності структурного підрозділу університету в мережі буде тільки збільшуватися.

Більшість абітурієнтів починають ознайомлення з університетом використовуючи інтернет. Сайт закладу вищої освіти впливає на перші враження майбутніх студентів. Наявність сайту важлива, тому що тільки після його огляду, більшість абітурієнтів відвідують університет особисто [5].

Цим обумовлена актуальність даної роботи.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Середовище розробки сайту

У якості середовища розробки було використано текстовий редактор Visual Studio Code.

Visual Studio Code – це популярний редактор, який розроблений компанією Microsoft. Він є легким, багатоплатформним з можливістю широкої кастомізації, використовуючи плагіни з вбудованого магазину.

3.2 Технології розробки

Для розробки веб-сайту було обрано дані технології:

- Next.js;
- TypeScript;
- SASS;
- React;
- Contentful;
- MUI.

Оглянемо кожну з них детальніше.

Next.js - React-фреймворк, який дозволяє створювати веб-застосунки з покращеною швидкодією, рендером на стороні серверу (SSR) або статичною генерацією (SSG). Дозволяє покращити CEO сторінки. Має вбудовану маршрутизацію сторінок.

Зазвичай React додатки використовують JavaScript компілятор Babel. Наприклад, веб-сайти та сам JavaScript не розуміють JSX розмітку, яку використовують в React для створення елементів. Вона хоч і подібна до HTML розмітки, але є звичайним JavaScript кодом (рис. 3.1).

```jsx
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

Рисунок 3.1 – JSX розмітка

За лаштунками, за допомогою Babel вона замінюється на виклик функції React.createElement(), яка приймає всі властивості передані JSX елементу (рис. 3.2).

```jsx
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

Рисунок 3.2 – Виклик React.createElement()

Але, хоч без Babel і важко уявити React додаток, він повільний. Тому в Next.js додатках за замочуванням використовується Speedy Web Compiler (SWC) - компілятор на базі мови програмування Rust, який швидший у 17 разів.

TypeScript — це мова, що конвертується в JavaScript. TypeScript додає додаткові типи до JavaScript, які підтримують інструменти для великомасштабних програм JavaScript для будь-якого браузера, для будь-якого хоста, у будь-якій ОС. TypeScript компілюється у читабельний, заснований на стандартах JavaScript код.

Однією з основних проблем JavaScript є динамічна нестрога типізація. При створенні змінної її тип буде визначатися під час компіляції і буде залежати від типу даних, які вона містить. Якщо змінна містить рядок, то її типом буде String, якщо число – Number. Коли змінній з типом String буде присвоєне нове значення типу Boolean, то сама змінна теж змінить свій тип. Іноді це призводить до помилок, які важко знайти.

TypeScript є мовою зі строгою типізації, хоча вона дозволяє динамічну. Можна попередньо вказати, який тип повинна приймати функція або змінна, і якщо передані дані не будуть йому відповідати, то TypeScript покаже помилку. Те, настільки строгим повинен бути TypeScript, що він повинен вважати за помилку, а що просто ігнорувати, залежить від його конфігу. Так як TypeScript дозволяє вказати вигляд об’єкту чи функції перед компіляцією, середовище розробки зможе надавати більш детальні та корисні підказки, в яких буде вказуватися, які методи містить клас або об’єкт та які параметри приймає функція.

SASS – розшифровується як Синтаксично Вражаюча Таблиця Стилів. Пре-процесор, який розширює можливості CSS.

SASS стилі, як і CSS, прийнято розбивати по різним файлам, де кожен файл виконує тільки одне завдання. Тобто на кожен компонент або сторінку, повинен бути окремий файл стилів, який потрібно імпортувати у відповідний Typescript файл. Це робиться з метою полегшення та пришвидшення роботи, бо файл, який має 10 рядків підтримувати легше, ніж файл, який має 200. Але коли проект збирається у готовий сайт, збирач модулів webpack створює один CSS файл, який містить всі файли стилів, написані розробником. Тому хоча при розробці стилі між собою не перетинаються, в кінцевому варіанті вони знаходяться в одному файлі. Якщо є конфліктуючі CSS селектори, які обирають один і то й же елемент, то пріоритетним буде той, який або був написаний останнім, або той, який має вищий пріоритет (рис. 3.3).

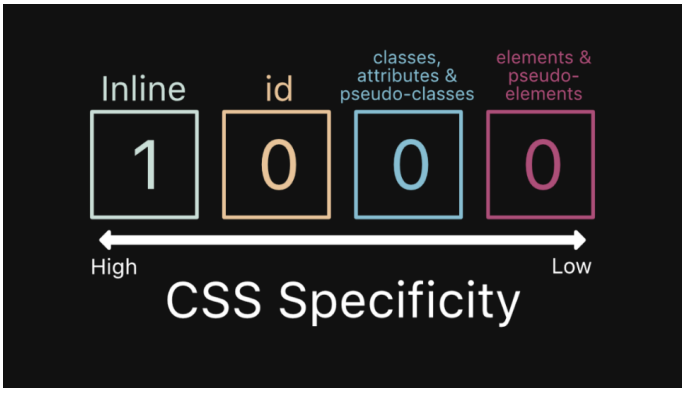

Рисунок 3.3 – Шкала специфічності CSS селектора

Якщо селектори мають однакову специфічність, то обраний буде останній. Враховуючи, що всі файли стилів об’єднаються в один, одні і ті ж самі класи або інші типи селекторів можуть перетинатися, що зробить вигляд сайту некоректним. Вирішити цю проблему можна використовуючи методологію іменування CSS класів BEM (Block Element Modifier), за допомогою якої імена класів будуть більш унікальними. Також можна використати CSS модулі – це файли, які мають розширення module. Для того, щоб використати класи з цього файлу, потрібно імпортувати його в компонент, а потім посилатися на нього як на об’єкт (рис 3.4).

```
<p className={styles.description}>{description}</p>
```

Рисунок 3.4 – Використання класу CSS модуля

Кожна властивість цього об’єкту буде відповідати класу в CSS файлі. Далі при збірці проекту webpack створить нове ім’я класу, яке буде складатися назви файлу, назви класу та випадкового набору символів (рис 3.5).

```
<p class="Staff_description__1Uasi">
```

Рисунок 3.5 – Згенерований унікальний клас

Використовуючи CSS модулі при розробці можна використовувати одне і теж ім’я класу в різних файлах, не хвилюючись про те чи будуть вони конфліктувати, бо кожного разу вони будуть унікальними.

При роботі над проектом, було надано перевагу CSS модулям.

React - JavaScript-бібліотека для створення користувацьких інтерфейсів. Має декларативну форму – потрібно описати те, що повинно бути зроблено, а не думати про реалізацію. Використовує JSX розмітку, яка імітує HTML код, що полегшує розробку. Так як браузер не розуміє JSX код, під час збірки проекту він перетворюється на HTML.

Будівельними блоками React є компоненти. Компонент – функція або клас, що повертає JSX. Компоненти поділяються на примітивні та складні. Примітивні компоненти майже ніколи не змінюють свій стан, не мають сторонніх ефектів, тільки отримують параметри та на основі них створюють елемент інтерфейсу. Складні компоненти мають стан, який змінюється, мають сторонні ефекти – вони можуть отримувати дані з API, використовують сторонні бібліотеки, втручаються в методи життєвого циклу компоненту. До 16 версії React, єдиним способом створення складних компонентів були класові компоненти. Примітивні компоненти можна було створювати за допомогою функцій. Після оновлення, у якому було додано React хуки, у звичайних функціях стало можливим зміна стану, використання методів життєвого циклу. Більшість нових проектів використовують функціональні компоненти, у новій документації React важко знайти згадку класових. Функціональні компоненти є більш лаконічними, три методи життєвого циклу componentDidMount, componentDidUpdate, componentWillUnmount заміняє одна функція useEffect. Тому в даному проекті будуть використовуватися функціональні компоненти.

Завдяки React Virtual DOM зміни в структуру сайту не вносяться без потреби, бо зміна DOM браузера є затратним процесом, тому краще не втручатися в нього без потреби. Virtual DOM є імітацією реального DOM, яку React використовує для слідкування за змінами. Кожного разу, коли оновлюється компонент, React порівнює новий Virtual DOM та попередній. Якщо між ними є відмінності, то DOM в браузері оновлюється в максимально ефективний спосіб, використовуючи передові алгоритми порівняння. До уваги береться тип елементу та його атрибути. Якщо в елементі був змінений тільки клас, то буде оновлений не весь елементи, а тільки його клас. Оновлення компоненту викликає зміна властивостей, які йому передаються, якщо компонент обгорнутий в React.memo, оновлення батьківського компоненту або виклик функції зміни стану, такої як useState, useReducer та useContext. У класових компонентів, які не використовують вище згадані функції, компонент оновлюється за допомогою методу setState.

Contentful - це платформа для управління контентом, яка дозволяє компаніям створювати, керувати та поширювати контент на різних цифрових платформах. Вона відома своїм headless підходом CMS (Content Management System), що відокремлює контент від його представлення, дозволяючи розробникам доставляти контент на різні канали (веб-сайти, додатки, пристрої Інтернету речей) за допомогою API-інтерфейсів. Contentful є зручною альтернативою створення бекенду для веб-сайтів, що потребують частого створення та оновлення контенту. Використовуючи систему управління контенту, можна відразу отримати API, за допомогою якого отримуються, оновлюються, видаляються данні та зручний графічний інтерфейс для управління ними. У випадку створення альтернативи до CMS, потрібно було б створювати базу даних, звертатися до неї через сервер, оформлювати валідацію даних на сервері, створювати зручний візуальний інтерфейс на клієнтській стороні з перевіркою даних.

Contentful дозволяє створювати новини, за допомогою зручного графічного інтерфейсу, використовуючи створену модель контенту (рис. 3.6).

Рисунок 3.6 – Модель новини

Створена новина з’явиться в списку новин, де можна буде її видаляти та редагувати (рис. 3.7).

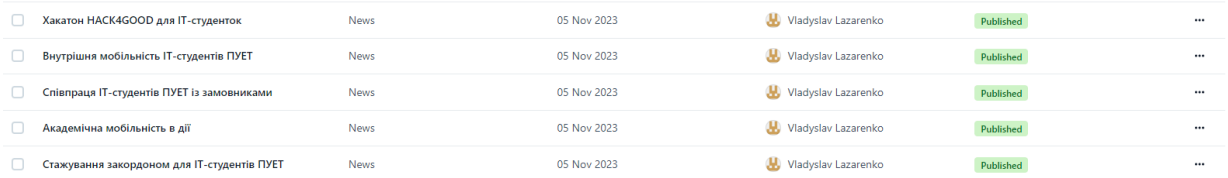

Рисунок 3.7 – Список новин

3.3 Доступність веб-сайту

Кожна людина має право на здобуття вищої освіти. За статистикою, у світі проживає близько 15% людей з інвалідністю, в Україні – 2 мільйони 703 тисячі. З 2,5 мільйонів студентів вітчизняних навчальних закладів 8 тисяч складають люди з особливими потребами [6]. Тому покращення доступності є важливим з соціальної точки зору. Проте висока доступність ще допомагає покращити індексацію сторінки в пошуковій системі.

Люди з проблемами зору використовують засоби, які озвучують зміст сторінки. Для того, щоб це було більш ефективним, розмітка сторінки повинна містити семантичні теги. Семантичні теги можуть в собі містити відтінок, емоцію, що дозволяє користувачу краще розуміти сенс інформації. Наприклад, тег strong або em означає, що текст є важливим і на нього потрібно звернути увагу, тому він буде прочитаний з особливою інтонацією.

Теги h1 – h6 позначають заголовки, де число означає пріоритетність. Прийнято використовувати тільки один h1 тег в рамках сторінки, який повинен описувати контент всієї сторінки. Не рекомендується пропускати заголовки, після h1 завжди першим повинен бути h2, а не h3 чи будь-який інший, тому що це збиває з пантелику користувача.

Теги header, footer, main, section, article, aside позначають смислові частини сторінки. У header розміщена верхня частина сторінки, яка зазвичай містить навігацію, або секції, footer містить нижню частину сторінки, main містить основний контент, section містить частину сторінки, яка виділяється з-поміж інших за смислом, article містить самодостатній контент, aside містить контент, який слабо відноситься до сторінки, зазвичай це рекламні банери.

Спеціальні атрибути aria, які розшифровуються як Accessible Rich Internet Applications, допомагають ще більше розширити доступність. З їх використанням можна додавати тип для кожного елементу, приховувати інформацію для читача екрану та багато іншого.

Кожна ілюстрація повинна мати alt атрибут, у якому описується контент. Це допомагає не тільки людям з проблемами зору, яким засоби читання екрану зможуть повідомити про зміст фотографії, а й людям у яких проблеми з інтернетом. Якщо ілюстрація не завантажиться, то вони хоча б зможуть побачити її опис.

Важливим для гарного досвіду користування є достатня контрастність кольорів. Текст повинен бути добре видимим на фоні, для того щоб переконатися в цьому, можна використовувати онлайн засоби визначення контрастності.

4. ПРАКТИЧНА ЧАСТИНА

4.1 Завантаження та встановлення необхідних інструментів

Для та запуску проекту використовується середовище запуску Node.js. За замочуванням він має менеджер пакунків npm, який використовується для встановлення сторонніх бібліотек, та npx, який використовується для запуску npm пакетів.

За допомогою команди npx create-next-app@latest –typescript встановимо Next.js та TypeScript (рис 4.1).

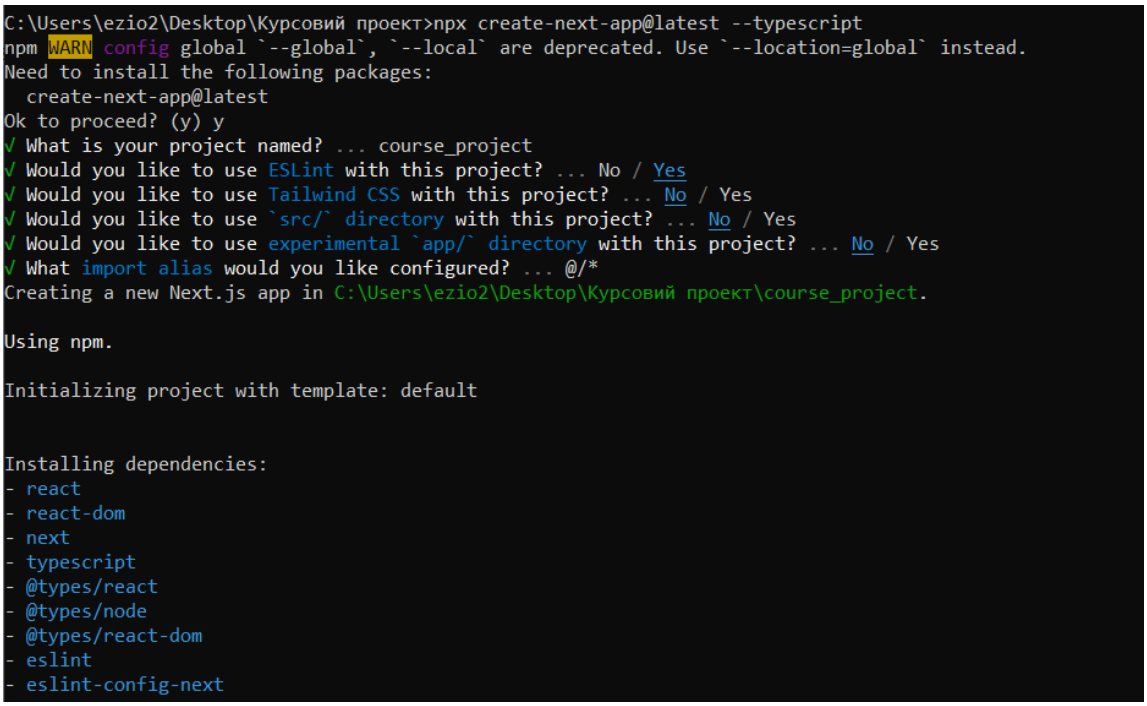

Рисунок 4.1 – Встановлення Next.js та TypeScript

Встановлено залежності необхідні для роботи (рис 4.2).

Рисунок 4.2 – Список залежностей в package.json

4.2 Налаштування проекту

Так як на сайті використовується шрифт Arial, підключимо його за допомогою відповідної властивості CSS (рис. 4.3).

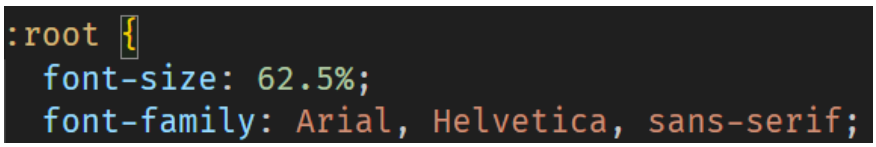

Рисунок 4.3 – Підключення необхідного шрифту

Для покращення адаптивності веб-сайту, псевдокласу :root, який у випадку html документу відповідає елементу html, присвоєний розмір шрифту 62.5%. Це означає, що яким би не був розмір шрифту обраний користувачем, сайт буде використовувати 62.5% від нього. У випадку стандартного шрифту, що дорівнює 16 пікселів, корінний розмір шрифту буде становити 10 піксель. Надалі в CSS буде

використовуватися одиниця виміру rem, яка розшифровується як root em. Її користь в тому, що змінюючи тільки розмір корінного шрифту, будь-які властивості, які використовують rem, будуть змінені також. Без його використання, довелося б змінювати кожну властивість окремо.

Для збільшення ефективності роботи з TypeScript, додаймо власні правила до tsconfig.json (рис. 4.4).

```
"noUnusedLocals": true,
"noFallthroughCasesInSwitch": true,
"noImplicitAny": true,
"noImplicitThis": true,
"noUnusedParameters": true,
```

Рисунок 4.4 – Правила tsconfig.json

4.3 Створення файлової структури

React є достатньо вільною бібліотекою у якої немає чітких вимог до вигляду файлової структури та архітектури додатку, що є одночасно і плюсом, і мінусом. Але є декілька загальних та популярних правил оформлення проекту [7].

Слід групувати файли за функціями або маршрутами (рис 4.5).

```
common/
  Avatar.js
  Avatar.css
  APIUtils.js
  APIUtils.test.js
feed/
  index.js
  Feed.js
  Feed.css
  FeedStory.js
  FeedStory.test.js
  FeedAPI.js
profile/
  index.js
  Profile.js
  ProfileHeader.js
  ProfileHeader.css
  ProfileAPI.js
```

Рисунок 4.5 – Групування файлів за функціями або маршрутами

Також можна структурувати файли за схожістю. Наприклад, всі компоненти будуть розміщені в папці компонентів, а те, що стосується API – в папці API (рис. 4.6).

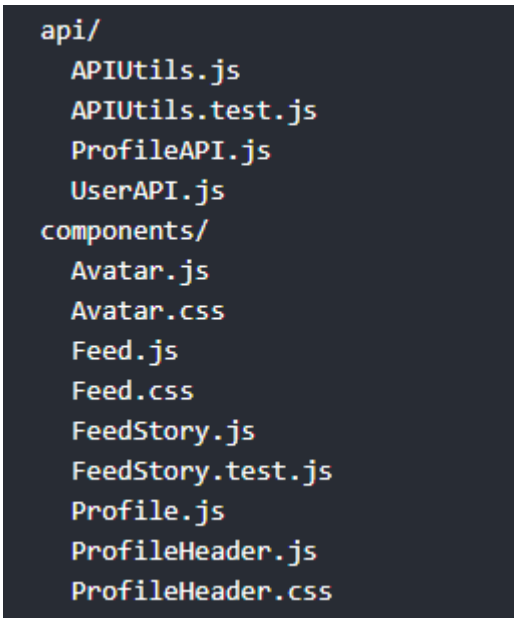

Рисунок 4.6. – Структурування за схожістю

Потрібно уникати надлишкового вкладення. Глибина вкладення впливає на простоту імпортування файлу – чим більше вкладення, тим складніший шлях імпорту.

Хоча файлова структура проекту є дуже важливою та помітно впливає на швидкість та ефективність роботи розробника над проектом, не обов’язково витрачати на неї багато часу. Можна розпочати з будь-якої структури, а потім у ході роботи оптимізувати її за потреби.

Звернувши увагу на дані рекомендації, створимо необхідну файлову структуру (рис. 4.7).

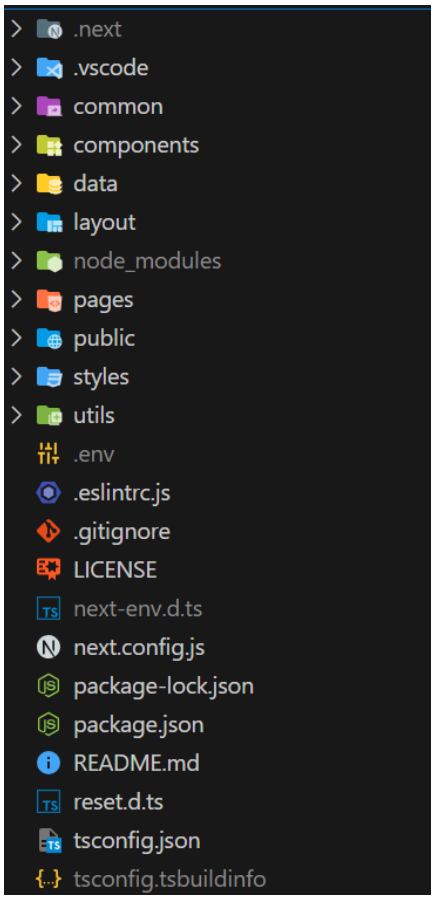

Рисунок 4.7 – Файлова структура проекту

Оглянемо найважливіші папки та файли проекту.

Папка .next містить побудований сайт на основі файлів проекту (рис 4.8).

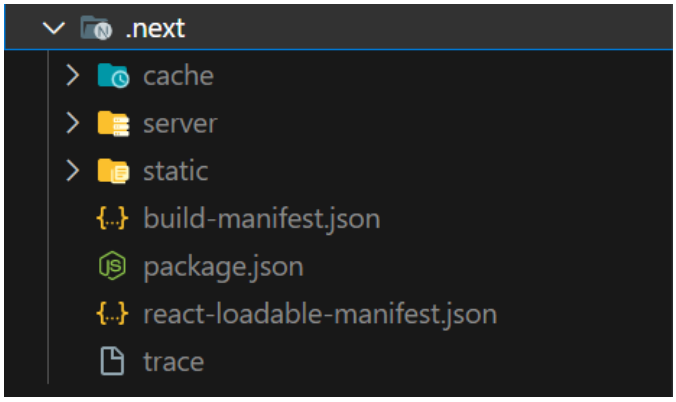

Рисунок 4.8 – Структура папки .next

Папка components містить React-компоненти (рис. 4.9). React-компоненти – це будівельні блоки React додатку. Будь-яка функція або клас, який повертає JSX розмітку являється компонентом.

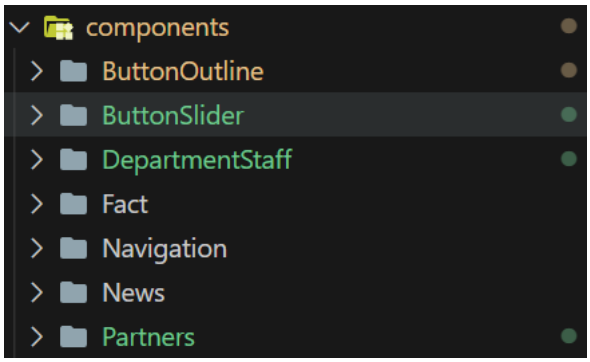

Рисунок 4.9 – Структура папки компонентів

Папка layout має універсальні React-компоненти, які використовуються для побудови макету на різних сторінках (рис. 4.10).

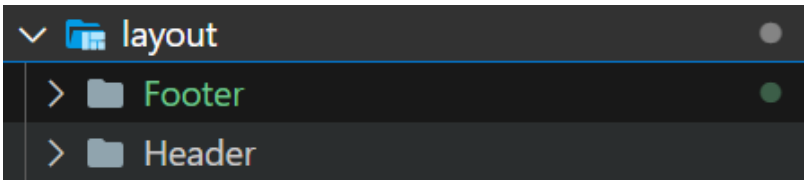

Рисунок 4.10 – Структура папки layout

У папці node_modules знаходяться всі JavaScript бібліотеки, які використовуються в проекті.

Папка pages містить в собі сторінки проекту та application programming interface (API). API зазвичай використовують для звернення до баз даних. (рис. 4.11).

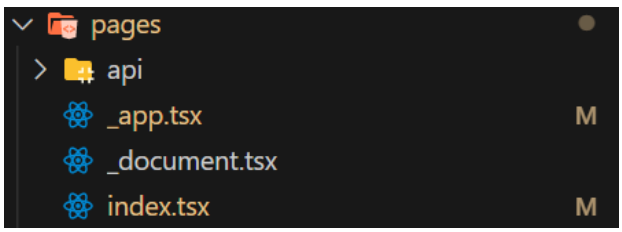

Рисунок 4.11 – Структура папки pages

Styles містить файли SASS. Кожен файл має назву, яка відповідає його функції (рис. 4.12).

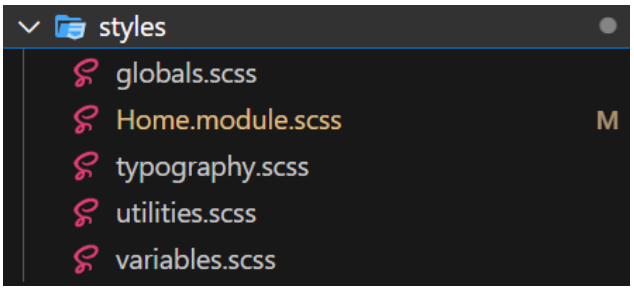

Рисунок 4.12 – Структура styles

У файлі next.config.json розміщені налаштування next проекту (рис. 4.13).

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  reactStrictMode: true,
  images: {
    remotePatterns: [
      {
        protocol: "http",
        hostname: "www.matmodel.puet.edu.ua",
        port: "",
        pathname: "/photo/**",
      },
    ],
  },
};

module.exports = nextConfig;
```

Рисунок 4.13 – Налаштування Next проекту

Для кращої оптимізації завантаження фото, фреймворк Next надає компонент Image. Він за лаштунками використовує численні засоби оптимізації фотографій, що дозволяє завантажувати їх поступово та швидко. Але якщо компонент Image використовує фотографії зі стороннього ресурсу, як у нашому випадку, в next.config потрібно вказати веб-сайт, з якого вони будуть завантажуватися.

4.4 Підключення до CMS

Система управління контентом Contentful має власну бібліотеку для TypeScript. Для роботи з нею потрібно імпортувати функцію createClient та передати їй токен доступу та ідентифікатор місця, в якому розміщений контент (рис. 4.14).

```
1  import { createClient } from "contentful";
2
3  export const client = createClient({
4    accessToken: process.env.CONTENTFUL_ACCESS_TOKEN as string,
5    space: process.env.CONTENTFUL_SPACE_ID as string,
6  });
```

Рисунок 4.14 – З’єднання з CMS

Ці дані генеруються в особистому кабінеті. Так як вони є конфіденційними й не повинні відправлятися до репозиторію, їх було додано в файл .env, який використовується для збереження чутливих даних, що не повинні бути розміщені в інтернеті. Сам файл .env не відправляється до репозиторію, а залишається тільки на персональному комп’ютері.

Для звернення до CMS потрібно використати метод клієнту getEntries. Якщо він викликається без аргументів, то він просто поверне об’єкт з результатом запиту. Також йому можна передавати аргументи, такі як тип контенту, не

За замочуванням, кожна сторінка Next.js використовує Static Site Generation, але за потреби, якщо потрібно, щоб при кожному в ході на сторінку отримувалася актуальна інформація, можна використати Server-Side Rendering. Для цього сторінка повинна експортувати функцію getServerSideProps, яка повинна повернути об'єкт props, який можна використовувати на сторінці. Ця функція запускається на сервері, тому звернення до CMS є безпечним.

4.5 Огляд виконаної роботи

У ході роботи було створено головну сторінку сайту кафедри та сторінку помилки. Сайт повністю відповідає дизайн макету, відображається коректно навіть на малих дисплеях, які мають 320 пікселів.

Було створено Layout сторінок. Він визначає те, що буде розміщено на кожній сторінці (рис. 4.15).

```
export default function App({ Component, pageProps }: AppProps) {
  return (
    <>
      <Header />
      <Component {...pageProps} />
      <Footer />
    </>
  );
}
```

Рисунок 4.15 – Layout сторінки

На кожній сторінці розміщено компонент Header (рис. 4.16) та Footer (рис. 4.17), які містять навігацію.

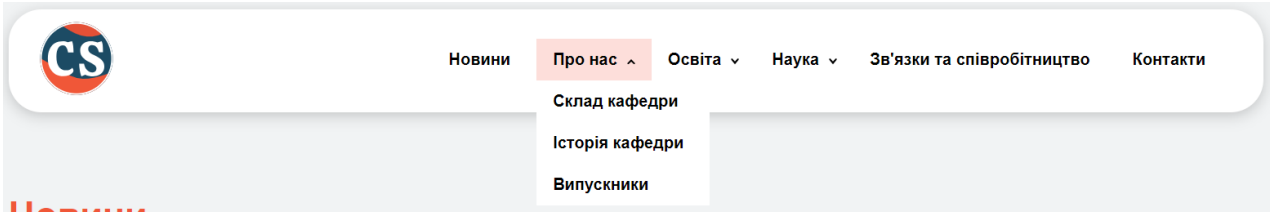

Рисунок 4.16 – Компонент Header

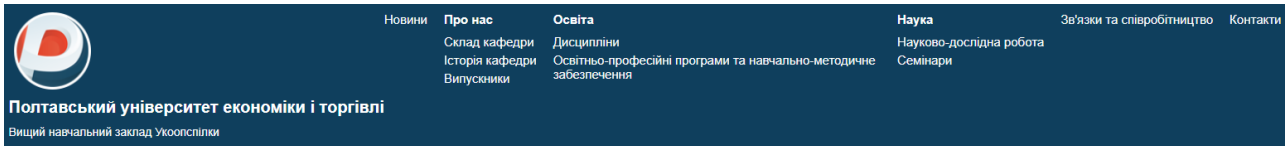

Рисунок 4.17 – Компонент Footer

Header є гнучким, правильно відображається на різних дисплеях. Починаючи з розміру екрану в 768 пікселів, перетворюється на “бургер” меню (рис. 4.18).

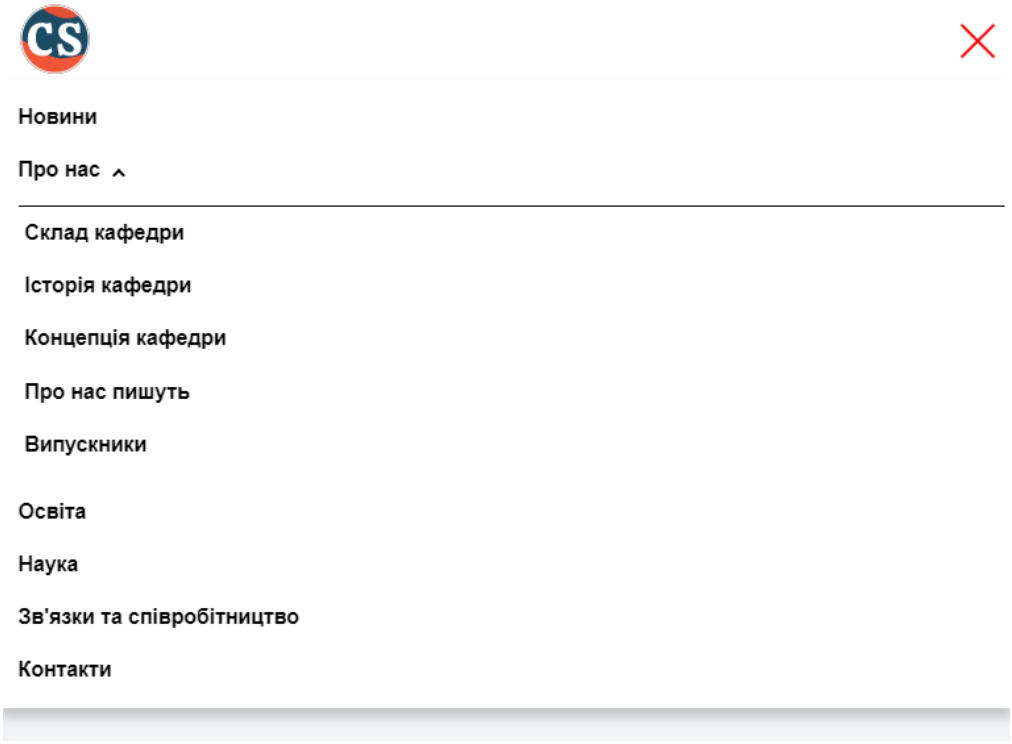

Рисунок 4.18 – Бургер меню

Розглянемо головну сторінку.

На сторінці розміщений блок з історією кафедри (рис. 4.19).

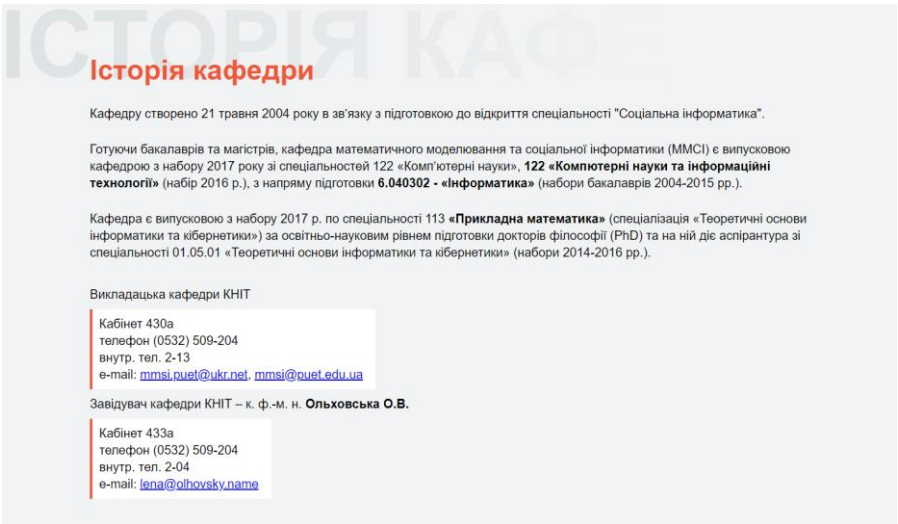

Рисунок 4.19 – Історія кафедри

Блок з фактами про кафедру (рис. 4.20).

Рисунок 4.20 – Факти про кафедру

Блок з останніми новинами (рис. 4.21).

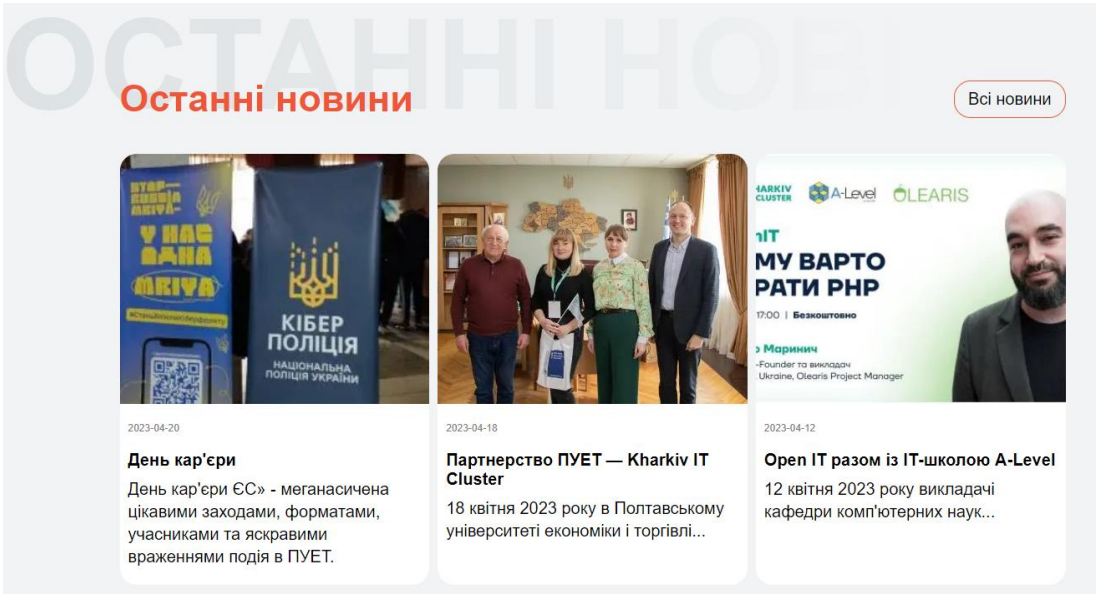

Рисунок 4.21 – Останні новини

У цьому блоці розміщені три останні новини, отримані з системи управління контентом Contentful. Так як фреймворк Next.js підтримує рендер на стороні серверу, кожного разу коли користувач заходить на головну сторінку, відбувається запит до CMS зі сторони серверу, на основі якого формується сторінка (рис. 4.22).

```
export async function getServerSideProps() {
  const newsList = await client.getEntries({
    content_type: "news",
    order: ["-fields.date"],
  });

  const firstThreeNews = newsList.items.slice(0, 3);

  return {
    props: {
      newsList: JSON.parse(JSON.stringify(firstThreeNews)),
    },
  };
}
```

Рисунок 4.22 – Звернення до CMS

У запиті зазначається тип контенту та параметр сортування за датою створення новини. Тому користувач завжди бачить актуальну інформацію.

Якщо натиснути на кнопку “Всі новини”, то відкриється окрема сторінка новин. Кожна новина є “клікабельною”, вона обгорнута в посилання, яке має вигляд https://puet-csit.vercel.app/news/[id], де id – ідентифікатор новини в CMS. Сторінка новини отримує дане id, яке використовується для звернення до CMS на стороні серверу (рис. 4.23).

```
export async function getServerSideProps(context: any) {
  const { items } = await client.getEntries({
    content_type: "news",
    "sys.id": context.params.id,
  });

  const news = items[0]?.fields;
  return { props: { news: JSON.parse(JSON.stringify(news)) } };
}
```

Рисунок 4.23 – Отримання новини з CMS

На основі отриманих даних буде сформована сторінка (рис. 4.24).

Рисунок 4.24 – Детальний перегляд новини

Блок з партнерами кафедри (рис. 4.25).

Рисунок 4.25 – Партнери кафедри

Цей блок має вигляді каруселі, в якій ілюстрації постійно гортаються в ліву сторону. При наведені, карусель стає на паузу.

Склад кафедри (рис. 4.26).

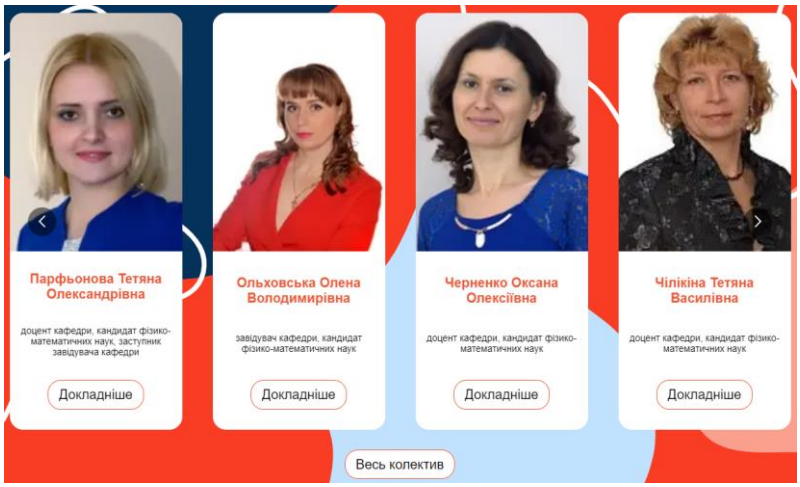

Рисунок 4.26 – Склад кафедри

Після натискання на кнопку “Докладніше” відкриється модульне вікно з детальною інформацію про викладача (рис. 4.27).

Рисунок 4.27 – Детальний перегляд викладача

Для перегляду всього колективу кафедри, потрібно натиснути на кнопку “Весь колектив”.

При переході на неіснуючу сторінку або іншій клієнтській помилці, користувач побачить сторінку помилки в стилі bsod (рис. 4.28).

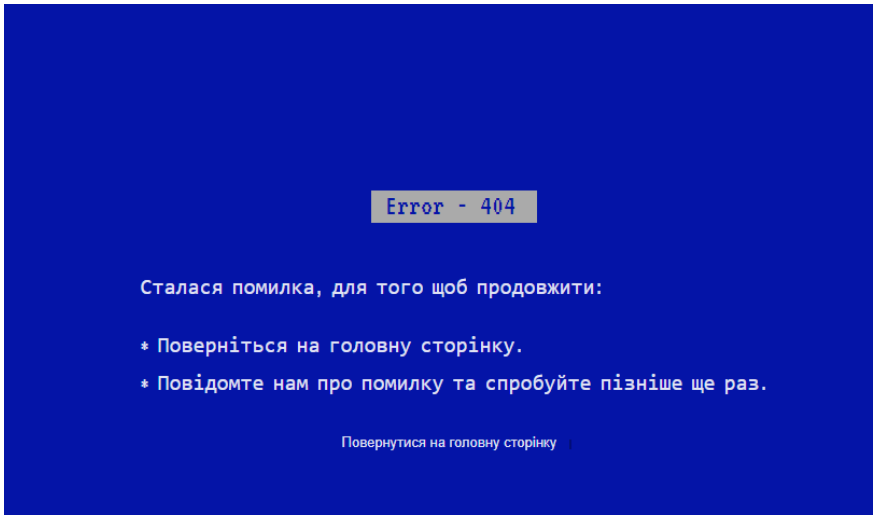

Рисунок 4.28 - Сторінка помилки

Створено сторінку новин. Новини створюються з використанням вбудованого редактору контенту в CMS. Їх можна отримати з CMS Contentful за допомогою офіційного SDK. При заході на сторінку, отримуються останні 10 новин та закріплені новини, формується загальна кількість сторінок (рис. 4.29).

```
export async function getServerSideProps(context: any) {
  const NEWS_PER_PAGE = 10;

  const category = context.query.category;
  const search = context.query.search;
  const page: number = context.query.page || 1;

  const { items: newsList, total } = await client.getEntries({
    content_type: "news",
    "fields.category": category,
    query: search,
    limit: NEWS_PER_PAGE,
    skip: (page - 1) * NEWS_PER_PAGE,
    order: ["-fields.date"],
  });

  const totalPages = Math.ceil(total / NEWS_PER_PAGE);

  const pinnedNewsEntries = await client.getEntries({
    content_type: "news",
    "fields.pinned": "true",
  });

  const pinnedList = pinnedNewsEntries.items.slice(0, 3);

  return {
    props: {
      pinnedList: JSON.parse(JSON.stringify(pinnedList)),
      newsList: JSON.parse(JSON.stringify(newsList)),
      total: totalPages,
    },
  };
}
```

Рисунок 4.29 - Отримання новин

На сторінці розміщений блок з закріплені новинами (4.30).

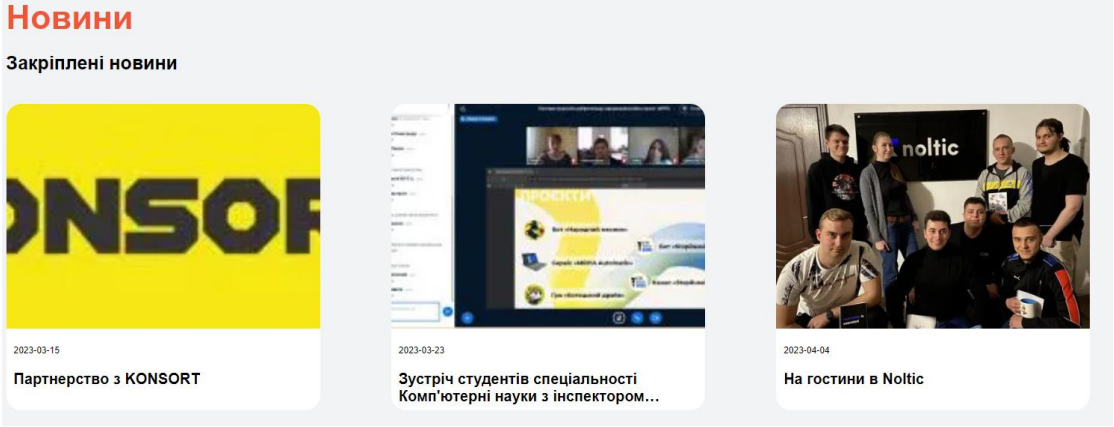

Рисунок 4.30 – Закріплені новини

Закріплювати або відкріплювати новини можна в системі управління контентом.

Створено фільтр, сортування та пошук новин за назвою (рис. 4.31).

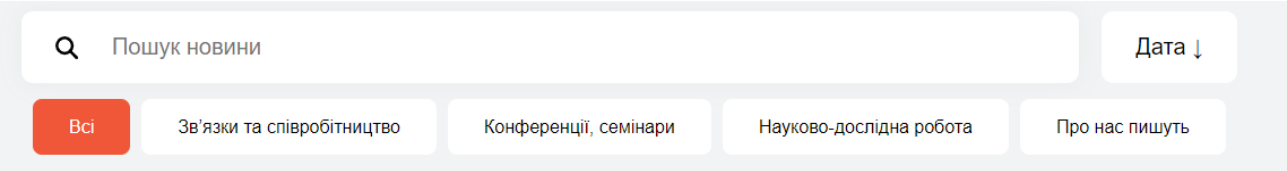

Рисунок 4.31 - Фільтр новин

Компонент пошуку новини змінює посилання сторінки, що призводить до її оновлення з новини параметрами пошуку (рис. 4.32).

```
export default function Search({ placeholder, className = "" }: SearchType) {
  const router = useRouter();

  const searchParams = useSearchParams();
  const category = searchParams.get("category")
    ? `category=${searchParams.get("category")}&`
    : "";

  const pathname = usePathname();

  const searchChangeHandler = (event: React.ChangeEvent<HTMLInputElement>) => {
    router.push(`${pathname}/?${category}search=${event.target.value}`, "", {
      scroll: false,
    });
  };

  return (
    <div className={`${styles.container} ${className}`}>
      <FontAwesomeIcon icon={faMagnifyingGlass} width={"20px"} />
      <input
        type="text"
        placeholder={placeholder}
        onChange={searchChangeHandler}
      />
    </div>
  );
}
```

Рисунок 4.32 – Компонент пошуку

Під час пошуку зберігається поточна категорія, тому пошук відбувається не серед всіх новин, а тільки серед обраної категорії.

Компонент категорії змінює нинішнє посилання в браузері, додаючи нову категорію (рис. 4.33).

```
<Category
  selected={"conferences" === selectedCategory}
  href={`/news/?category=conferences&${search}`}
>
  Конференції, семінари
</Category>
```

Рисунок 4.33 – Компонент категорії

Список новин містить набір компонентів новини. Кожен елемент відображає назву та дату створення (рис. 4.34).

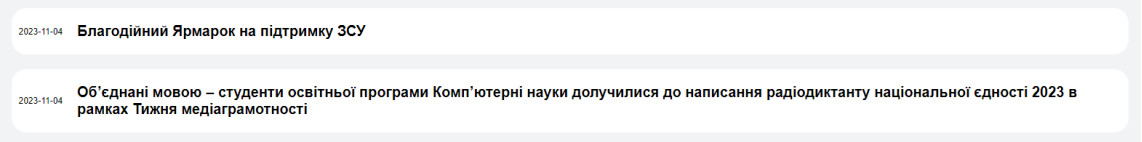

Рисунок 4.34 – Список новин

Новини переключаються компонентом пагінації (рис. 4.35).

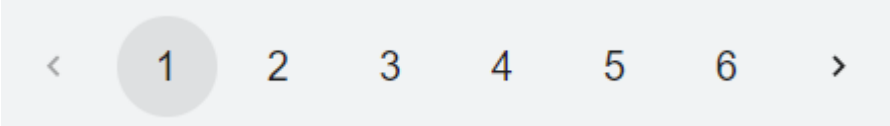

Рисунок 4.35 – Пагінація

Для пагінації використовується компонент бібліотеки для побудови візуальних інтерфейсів MUI (рис. 4.36).

```tsx
const handlePageChange = (
  _event: React.ChangeEvent<unknown>,
  value: number
) => {
  const url = createPageURL(value);
  router.push(url, url, { scroll: false });
};

return (
  <MUIPagination
    className={`${styles.pagination} ${className}`}
    size="large"
    count={totalPages}
    defaultPage={currentPage}
    onChange={handlePageChange}
  />
);
```

Рисунок 4.36 – Компонент пагінації

Новина складається з заголовку, списку соціальних мереж для поширення та власне самого контенту (рис. 4.37).

Рисунок 4.37 – Сторінка новини

Якщо до новини прикріплено 2 і більше фотографій підряд, то вони додаються до слайдера (рис. 4.38).

Рисунок 4.38 – Слайдер фотографій

Створена сторінка випускників кафедри (рис. 4.39).

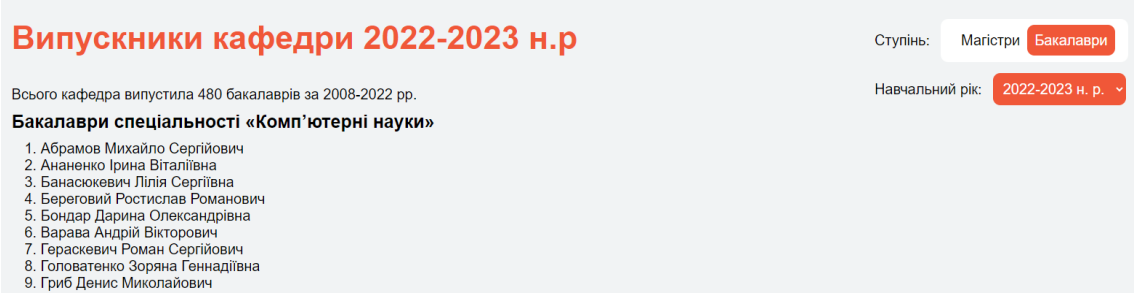

Рисунок 4.39 – Сторінка випускників

Кожен рік випуску розміщений на окремій сторінці, для кращого досвіду користування. Навігація між сторінками випускників відбувається за допомогою компоненту фільтру (рис. 4.40).

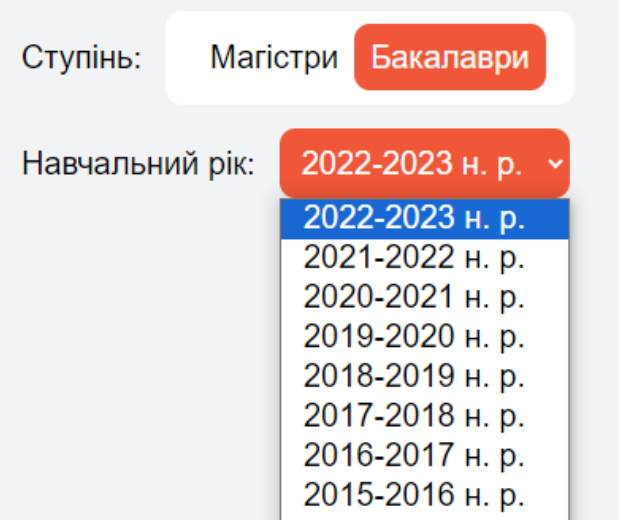

Рисунок 4.40 – Фільтр випускників

Фільтр змінює параметри посилання сторінки і перенаправляє на відповідний випуск.

Створена сторінка зі списком дисциплін для бакалаврів та магістрів (рис. 4.41).

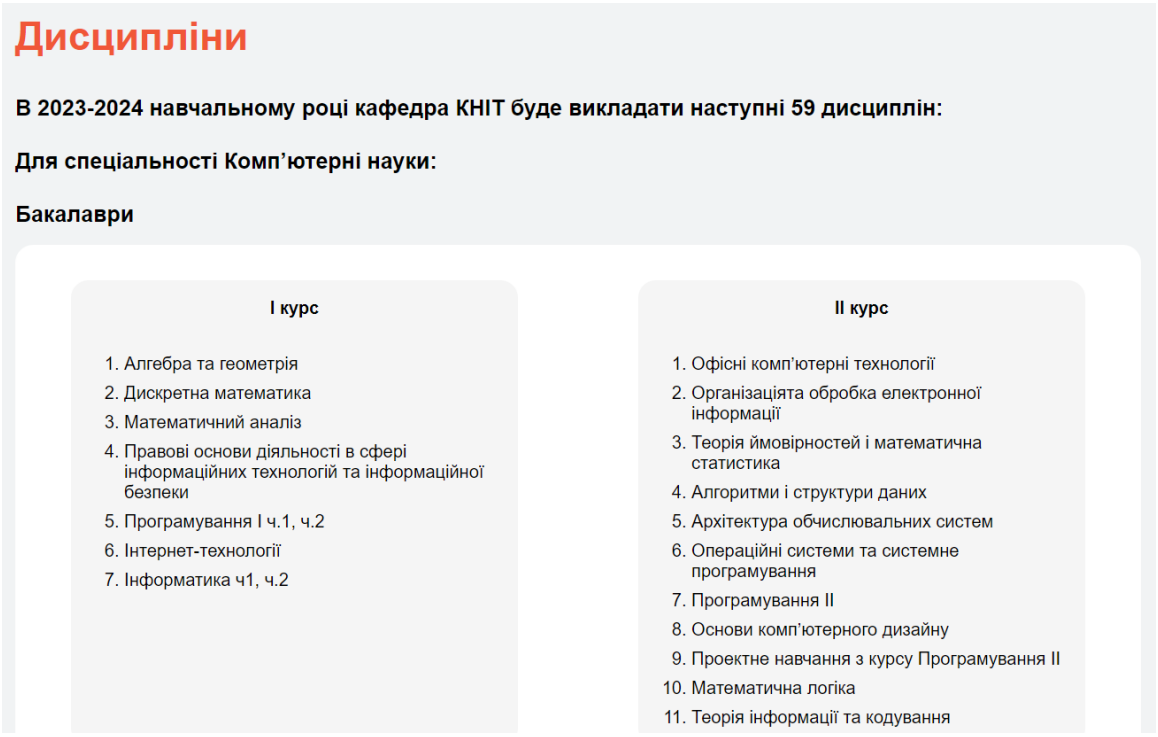

Рисунок 4.41 – Список дисциплін

Створена сторінка ‘Освітньо-професійні програми та навчально-методичне забезпечення’ (рис. 4.42).

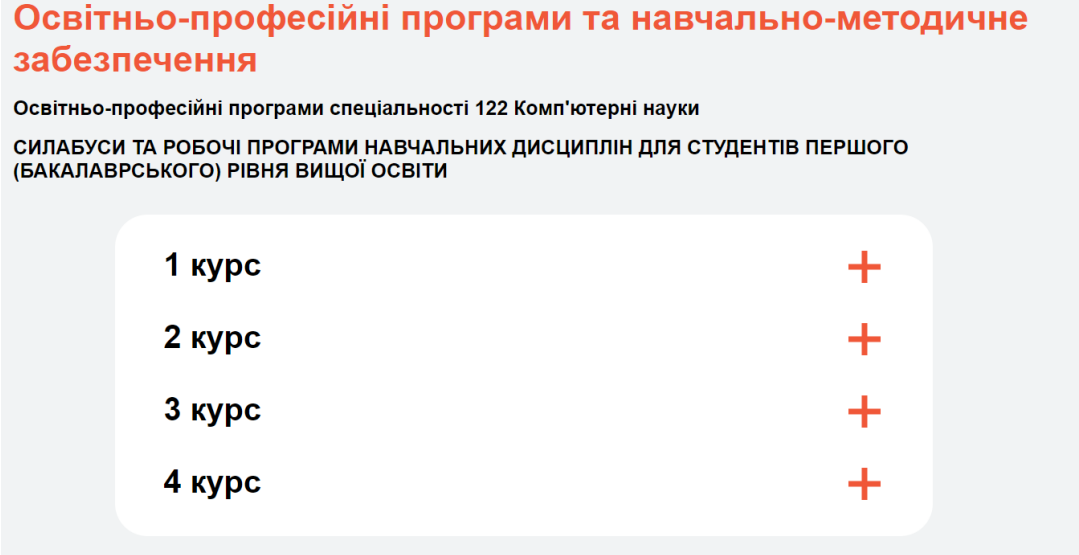

Рисунок 4.42 – Освітньо-професійні програми та навчально-методичне забезпечення

Силабуси та робочі програми розміщені списком, окремо для кожного курсу (рис. 4.43).

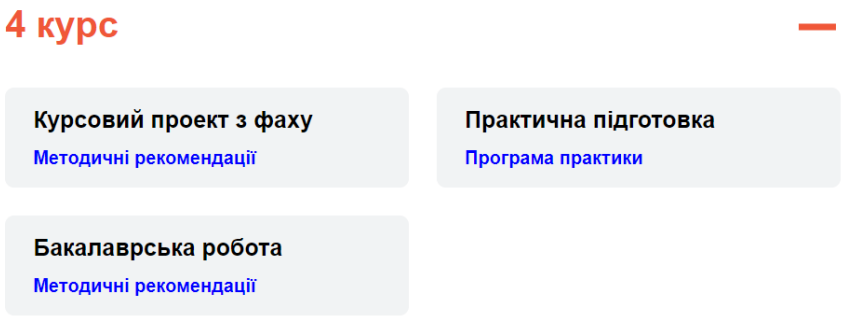

Рисунок 4.43 – Силабуси та робочі програми для обраного курсу

Створено компонент для кожної робочої програми та силабусу, що уніфікує та прискорює розробку (рис. 4.44).

```
<SyllabusItem
  title={"Програмування I"}
  programList={[
    "http://www.matmodel.puet.edu.ua/files/lic2020/rp-p1-22.pdf",
  ]}
  syllabusList={[
    "http://www.matmodel.puet.edu.ua/files/lic2020/syl-p1-22.pdf",
  ]}
/>
```

Рисунок 4.44 – Компонент силабусу та робочої дисципліни

Компонент отримує назву дисципліни, список програм та силабусів. На основі цього створюється JSX розмітка, що зменшує кількість коду.

4.6 Перевірка роботи сайту

Сайт адаптивний, інформація коректно відображається на екранах різних розмірів. Працює правильно в різних браузерах.

Перевірено роботу в браузері Chrome на операційній системі Windows (рис. 4.45).

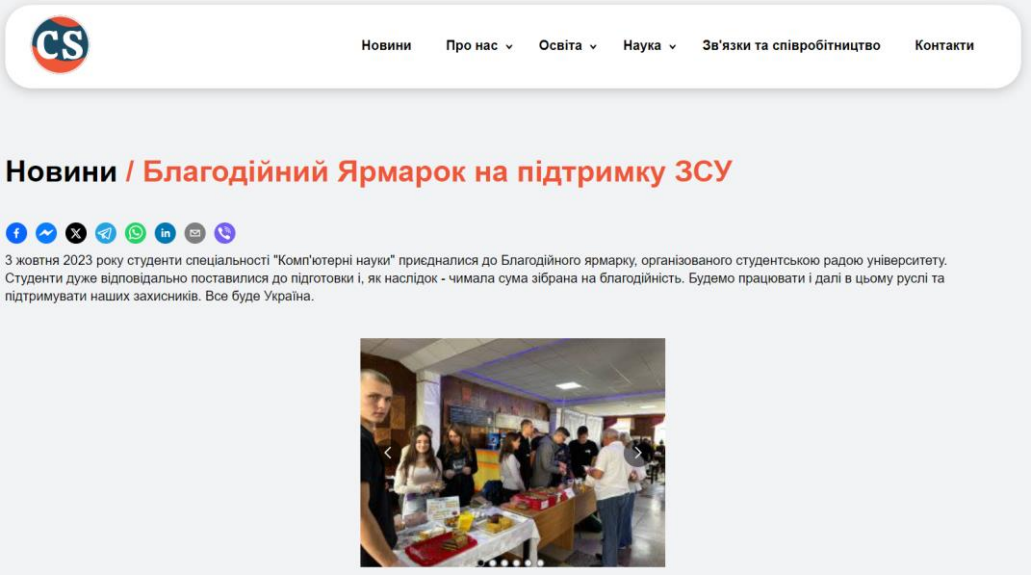

Рисунок 4.45 – Робота сайту на Windows

Перевірено працездатність сайту на мобільному телефоні з системою Android (рис. 4.46).

Рисунок 4.46 – Робота сайту на Android

Перевірено роботу на операційній системі iOS (рисунок 4.47).

Рисунок 4.47 – Робота сайту на iOS

Сайт працює коректно, елементи сторінок адаптуються під невеликий розмір дисплею. Працює швидко.

Проведено перевірку основних показників сайту використовуючи вбудовану в Google Chrome утиліту LightHouse. Вона показала, що сайт працює досить швидко та якісно. Було отримано максимальну оцінку за швидкість, CEO та найкращі практики (рис. 4.48).

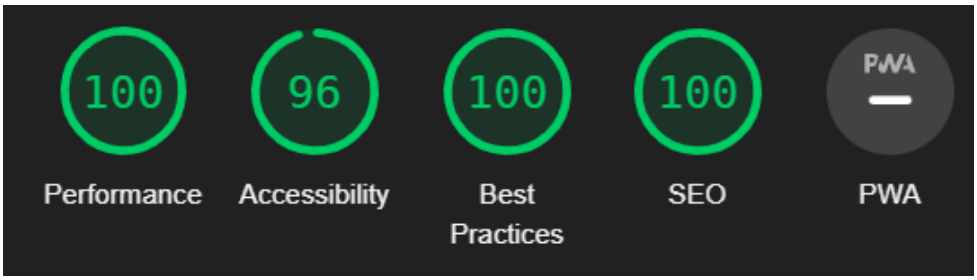

Рисунок 4.48 – Звіт LightHouse

Утиліта оцінки доступності сайту WAVE знайшла тільки один незначний недолік, який вирішувати необов’язково (рис. 4.49).

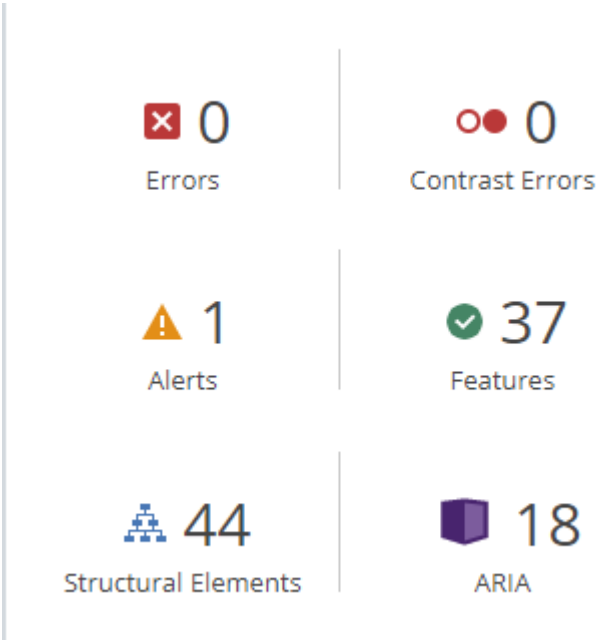

Рисунок 4.49 – Оцінка WAVE

Веб-сайт немає ніяких помилок, кольори є достатньо контрастними для зручного користування. Широко використовуються атрибути aria, які додають семантики елементам.

ВИСНОВКИ

Наявність сайту структурного підрозділу вищого навчального закладу, допомагає підвищити якість освіти, покращити комунікацію між студентами та викладачами, полегшує знайомство для абітурієнтів та дозволяє долучатися до навчання в незалежності від місцезнаходження.

У результаті виконання проекту було виконано поставлену задачу, а саме розроблено сайт кафедри Комп’ютерних наук та інформаційних технологій, як прототип, який може бути використаний і для інших структурних підрозділів.

При створені сайту було використано актуальні технології так як: Next.js, TypeScript, React, Contentful, SASS. Сайт отримав високі оцінки за доступність, продуктивність, SEO та використання найкращих практик.

Веб-сайт був протестований на різних пристроях, які мають відмінні операційні системи та розширення екрану. Застосунок працював коректно, відображення елементів залишалося стабільним.

Огляд методів розробки показав, що існує чимало інструментів розробки сайту, які дозволяють людям без досвіду створити власний застосунок. Для більшості можливостей, які надають сайти конструктори, буде достатньо.

У ході роботи було отримано навички роботи з новими технології.

Запровадження нових технологій, розширення функціональності для взаємодії користувачів можуть стати наступними кроками у вдосконаленні сайту. Корисним буде впровадження великих мовних моделей, для покращення пошукової системи сайту, для надання персоналізованого зворотнього зв’язку з унікальними відповідями, які враховують контекст. Також можна впровадити систему сповіщень, яка буде відправляти повідомлення про актуальні новини на обраний засіб комунікації. Сайт розміщено в інтернеті у вільному доступі.

Результати роботи представлено та апробовано на науково-практичному семінарі «Комп’ютерні науки та інформаційні технології» (КНІТ-2023),

Міжнародній молодіжній науково-практичній інтернет-конференції «Наука і молодь в XXI сторіччі», 2023), студентській конференції ПУЕТ, в журналі «Shipbuilding & marine infrastructure».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Impact of the Internet on Society: A Global Perspective [Електронний ресурс] // MIT Technology Review is a bimonthly magazine wholly owned by the Massachusetts Institute of Technology, and editorially independent of the university. – Режим доступу: https://www.technologyreview.com/2014/09/08/171458/the-impact-of-the-internet-on-society-a-global-perspective/
2. Turkle S. Alone Together: Why We Expect More from Technology and Less from Each Other / S. Turkle – 2012 - Basic Books, 384 c.
3. Kocoń, J., Cichecki, I., Kaszyca, O., Kochanek, M., Szydło, D., Baran, J., ... & Kazienko, P. (2023). ChatGPT: Jack of all trades, master of none. Information Fusion, 101861.
4. Освітній процес в умовах воєнного стану в Україні : матеріали всеукраїнського науково-педагогічного підвищення кваліфікації, 3 травня – 13 червня 2022 року. – Одеса : Видавничий дім «Гельветика», 2022. – 504 с.
5. Microsoft Word – 10560 [Електронний ресурс] // The Academic and Business Research Institute is a privately-owned, for-profit publishing corporation incorporated and operating in the state of Florida, United States of America. - Режим доступу: http://www.aabri.com/manuscripts/10560.pdf
6. Студенти з особливими потребами | КПІ ім. Ігоря Сікорського [Електронний ресурс] // Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського” – Режим доступу: https://kpi.ua/616-4
7. File Structure – React [Електронний ресурс] // Офіційна документація React – Режим доступу: https://legacy.reactjs.org/docs/faq-structure.html
8. Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale / B. Cherny – 2019 – O’Reilly Media, 322 c.
9. Banks A. Learning React: Modern Patterns for Developing React Apps / A. Banks – 2020 – O’Reilly Media, 307 c.

10. TypeScript: The starting point for learning TypeScript [Електронний ресурс] // Офіційна документація React – Режим доступу: https://www.typescriptlang.org/docs/
11. Flanagan D. The Definitive Guide: Master the World's Most-Used Programming Language / D. Flanagan – 2020 – O'Reilly Media, 704 с.
12. MDN Web Docs [Електронний ресурс] // Documenting web technologies, including CSS, HTML, and JavaScript, since 2005. – Режим доступу: https://developer.mozilla.org/en-US/
13. Saichaie, K., & Morphew, C. C. (2014). What college and university websites reveal about the purposes of higher education. The Journal of higher education, 85(4), 499-530.
14. Gordon, J., & Berhow, S. (2009). University websites and dialogic features for building relationships with potential students. Public relations review, 35(2), 150-152.
15. Almahamid, S. M., Tweiqat, A. F., & Almanaseer, M. S. (2016). University website quality characteristics and success: lecturers' perspective. International Journal of Business Information Systems, 22(1), 41-61.
16. Hassouna, M. S., Sahari, N., & Ismail, A. (2017). University website accessibility for totally blind users. Journal of Information and Communication Technology, 16(1), 63-80.
17. Ott, H., Wang, R., & Bortree, D. (2016). Communicating sustainability online: An examination of corporate, nonprofit, and university websites. Mass Communication and Society, 19(5), 671-687.
18. Sukmasetya, P., Setiawan, A., & Arumi, E. R. (2020, April). Usability evaluation of university website: a case study. In Journal of Physics: Conference Series (Vol. 1517, No. 1, p. 012071). IOP Publishing.
19. VandeCreek, L. M. (2005). Usability analysis of Northern Illinois University Libraries' website: a case study. OCLC Systems & Services: International digital library perspectives, 21(3), 181-192.

20. Ishak, Z., Alexander, O., Al-Sanjary, O. I., & Yusuf, E. (2020, February). Potential students preferences towards university website interface design: the methodology. In 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA) (pp. 115-119). IEEE.
21. Undu, A. and Akuma, S., 2018. Investigating the usability of a university Website from the users' perspective: An empirical study of Benue State University Website. International Journal of Computer and Information Engineering, 12(10), pp.922-929.
22. Гаркуша С. В., Методичні рекомендації щодо виконання кваліфікаційної роботи студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня магістра / С. В. Гаркуша, О. В. Ольховська, О. О. Черненко – Полтава : ПУЕТ, 2023. – 68 с.

ДОДАТОК А. ПРОГРАМНИЙ КОД

КОД ГОЛОВНОЇ СТОРІНКИ

```
import Image from "next/image";
import { INews } from "@/common/types";
import { client } from "@/utils/contentful";
import Head from "next/head";
import Fact from "@/components/Fact/Fact";
import ButtonOutline from "@/components/ButtonOutline/ButtonOutline";
import Partners from "@/components/Partners/Partners";
import StaffList from "@/components/DepartmentStaff/StaffList";
import NewsList from "@/components/News/NewsList";
import styles from "@/styles/Home.module.scss";
type indexProps = {
  newsList: Array<INews>;
};
export default function Home({ newsList }: indexProps) {
  return (
    <>
      <Head>
        <title>
          Кафедра комп&amp;apos;ютерних наук та інформаційних технологій
        </title>
        <meta
          name="description"
          content="Офіційний сайт кафедри комп'ютерний наук та інформаційних технологій Полтавського університету економіки та торгівлі"
        />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <meta name="robots" content="index, follow" />
        <link rel="icon" href="/Logo.svg" />
      </Head>
      <main>
        <section className={`${styles.section} ${styles.history}`}>
          <div className="container">
            <h2 className="h-1 background" data-background="Історія кафедри">
              Історія кафедри
            </h2>
            <div className={styles.description}>
              <p className="text-1">
                Кафедру створено 21 травня 2004 року в зв’язку з підготовкою до
                відкриття спеціальності {""}Соціальна інформатика{""}.
              </p>
              <p className="text-1">
                Готуючи бакалаврів та магістрів, кафедра математичного
                моделювання та соціальної інформатики (ММСІ) є випусковою
                кафедрою з набору 2017 року зі спеціальностей 122 «Комп’ютерні
```

```
        науки», <b>122 «Компютерні науки та інформаційні технології»</b>{" "}
        (набір 2016 р.), з напряму підготовки{" "}
        <b>6.040302 - «Інформатика»</b> (набори бакалаврів 2004-2015
        рр.).
      </p>
      <p className="text-1">
        Кафедра є випусковою з набору 2017 р. по спеціальності 113{" "}
        <b>«Прикладна математика»</b> (спеціалізація «Теоретичні основи
        інформатики та кібернетики») за освітньо-науковим рівнем
        підготовки докторів філософії (PhD) та на ній діє аспірантура зі
        спеціальності 01.05.01 «Теоретичні основи інформатики та
        кібернетики» (набори 2014-2016 рр.).
      </p>
      <div className={styles.contacts}>
        <div>
          <p className="text-1">Викладацька кафедри КНІТ</p>
          <div className={styles.contact}>
            <p className="text-1">Кабінет 430а</p>
            <p className="text-1">телефон (0532) 509-204</p>
            <p className="text-1">внутр. тел. 2-13</p>
            <p className="text-1">
              e-mail:{" "}
              <a href="mailto:mmsi.puet@ukr.net">mmsi.puet@ukr.net</a>,{" "}
              <a href="mailto:mmsi@puet.edu.ua">mmsi@puet.edu.ua</a>
            </p>
          </div>
        </div>
        <div>
          <p className="text-1">
            Завідувач кафедри КНІТ – к. ф.-м. н. <b>Ольховська О.В.</b>
          </p>
          <div className={styles.contact}>
            <p className="text-1">Кабінет 433а</p>
            <p className="text-1">телефон (0532) 509-204</p>
            <p className="text-1">внутр. тел. 2-04</p>
            <p className="text-1">
              e-mail:{" "}
              <a href="mailto:lena@olhovsky.name">lena@olhovsky.name</a>
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
<section className={`${styles.facts} ${styles.section}`}>
  <div className="container">
    <h2
      className="h-1 background"
      data-background="Декілька фактів про кафедру"
    >
```

```
      Декілька фактів про кафедру
    </h2>
    <ul className={styles.facts_list}>
      <li>
        <Fact
          img="/Person.svg"
          title="Випустила більше"
          number="500"
          description="студентів"
        />
      </li>
      <li>
        <Fact
          img="/Gold_medal.svg"
          title="Зайняла"
          number="1"
          description="місце на олімпіаді 2019р."
        />
      </li>
    </ul>
  </div>
</section>
<section className={`${styles.news} ${styles.section}`}>
  <div className="container">
    <header>
      <h2 className="h-1 background" data-background="Останні новини">
        Останні новини
      </h2>
      <ButtonOutline type="link" href="/news"> Всі новини</ButtonOutline>
    </header>
    <NewsList newsList={newsList} />
  </div>
</section>
<div className={styles.background_wrapper}>
  <section className={`${styles.partners} ${styles.section}`}>
    <div className="container">
      <h2 className="h-1 background" data-background="Наші партнери">
        Наші партнери
      </h2>
      <Partners>
        <Image
          src={"/Partners/Beetroot_Logo.png"}
          width={100}
          height={100}
          alt="Beetroot logo"
        />
        <Image
          src={"/Partners/Kharkiv_Cluster.svg"}
          width={100}
          height={100}
          alt="Kharkiv Cluster logo"
```

```
            />
            <Image
              src={"/Partners/logoQaTestLab.svg"}
              width={100}
              height={100}
              alt="QaTestLab logo"
            />
            <Image
              src={"/Partners/NIX_Logo.png"}
              width={100}
              height={100}
              alt="NIX logo"
            />
            <Image
              src={"/Partners/Noltic_Logo.png"}
              width={100}
              height={100}
              alt="Noltic logo"
            />
          </Partners>
        </div>
      </section>
      <section className={`${styles.department} ${styles.section}`}>
        <div className="container">
          <StaffList />
        </div>
      </section>
    </div>
  </main>
 </>
 );
}
export async function getServerSideProps() {
 const newsList = await client.getEntries({
  content_type: "news",
  order: ["-fields.date"],
 });
 const firstThreeNews = newsList.items.slice(0, 3);
 return {
  props: {
   newsList: JSON.parse(JSON.stringify(firstThreeNews)),
  },
 };
}
```

КОД СТОРІНКИ НОВИНИ

```
import { client } from "@/utils/contentful";
import Image from "next/image";
import { usePathname } from "next/navigation";
import { BLOCKS, INLINES } from "@contentful/rich-text-types";
import { documentToReactComponents as renderRichText } from "@contentful/rich-text-react-renderer";
```

```
import Link from "next/link";
import { ImageLoader } from "next/image";
import styles from "@/styles/NewsIndividual.module.scss";
import ContentWrapper from "@/components/News/ContentWrapper";
import Share from "@/components/Share/Share";
const contentfulImageLoader: ImageLoader = ({ src, width }) => {
  return `https:${src}?w=${width}`;
};
export default function NewsIndividual({ news }: any) {
  const renderOptions = {
    renderNode: {
      [INLINES.EMBEDDED_ENTRY]: (node: any) => {
        if (node.data.target.sys.contentType.sys.id === "blogPost") {
          return (
            <a href={`/blog/${node.data.target.fields.slug}`}>
              {" "}
              {node.data.target.fields.title}
            </a>
          );
        }
      },
      [BLOCKS.EMBEDDED_ENTRY]: (node: any) => {
        if (node.data.target.sys.contentType.sys.id === "codeBlock") {
          return (
            <pre>
              <code>{node.data.target.fields.code}</code>
            </pre>
          );
        }
        if (node.data.target.sys.contentType.sys.id === "videoEmbed") {
          return (
            <iframe
              src={node.data.target.fields.embedUrl}
              height="100%"
              width="100%"
              frameBorder="0"
              scrolling="no"
              title={node.data.target.fields.title}
              allowFullScreen={true}
            />
          );
        }
      },
      [BLOCKS.EMBEDDED_ASSET]: (node: any) => {
        return (
          <Image
            className={"image"}
            loader={contentfulImageLoader}
            src={node.data.target.fields.file.url}
            height={node.data.target.fields.file.details.image.height}
            width={node.data.target.fields.file.details.image.width}
```

```
          alt={node.data.target.fields.description}
        />
      );
    },
  },
};
return (
  <main className="main">
    <div className="container">
      <h1 className="h-1 page-heading">
        <Link className={styles.main_page_link} href={"/news"}>
          Новини
        </Link>{" "}
        / {news.title}
      </h1>
      <Share shareUrl={`https://${host}${pathname}`} title={news.title} />
      <article className={`${styles.content} text-1`}>
        <ContentWrapper>
          {renderRichText(news.content, renderOptions)}
        </ContentWrapper>
      </article>
    </div>
  </main>
);
}
export async function getServerSideProps(context: any) {
  const { items } = await client.getEntries({
    content_type: "news",
    "sys.id": context.params.id,
  });
  const news = items[0]?.fields;
  return { props: { news: JSON.parse(JSON.stringify(news)) } };
}
```

КОД СТОРІНКИ НОВИН

```
import Pagination from "@/components/Pagination/Pagination";
import { client } from "@/utils/contentful";
import Search from "@/components/Search/Search";
import ButtonSort from "@/components/ButtonSort/ButtonSort";
import Categories from "@/components/Categories/Categories";
import NewsList from "@/components/News/NewsList";
import styles from "@/styles/News.module.scss";
export default function News({ newsList, pinnedList, total }: any) {
  return (
    <main className="main">
      <div className="container">
        <header className={styles.header}>
          <h1 className="h-1">Новини</h1>
        </header>
        {pinnedList.length > 0 && (
          <section className={styles.pinned}>
```

```
          <h1 className="h-2">Закріплені новини</h1>
          <NewsList className={styles.pinned_list} newsList={pinnedList} />
        </section>
      )}
      <div className={styles.filter}>
        <div>
          <Search className={styles.search} placeholder="Пошук новини" />
          <ButtonSort />
        </div>
        <Categories />
      </div>
      {newsList.length > 0 ? (
        <section>
          <NewsList newsList={newsList} compact />
          <Pagination className={styles.pagination} totalPages={total} />
        </section>
      ) : (
        <p className={styles.news_empty}>Новини відсутні</p>
      )}
    </div>
  </main>
 );
}
export async function getServerSideProps(context: any) {
 const NEWS_PER_PAGE = 10;
 const category = context.query.category;
 const search = context.query.search;
 const page: number = context.query.page || 1;
 const { items: newsList, total } = await client.getEntries({
  content_type: "news",
  "fields.category": category,
  query: search,
  limit: NEWS_PER_PAGE,
  skip: (page - 1) * NEWS_PER_PAGE,
  order: ["-fields.date"],
 });
 const totalPages = Math.ceil(total / NEWS_PER_PAGE);
 const pinnedNewsEntries = await client.getEntries({
  content_type: "news",
  "fields.pinned": "true",
 });
 const pinnedList = pinnedNewsEntries.items.slice(0, 3);
 return {
  props: {
   pinnedList: JSON.parse(JSON.stringify(pinnedList)),
   newsList: JSON.parse(JSON.stringify(newsList)),
   total: totalPages,
  },
 };
}
```

КОД СТОРІНКИ ПОМИЛКИ

```
import Link from "next/link";
import styles from "@/styles/404.module.scss";
export default function Custom404() {
  return (
    <main className={`${styles.bsod} ${styles.container}`}>
      <div>
        <h1 className={`${styles.neg} ${styles.title}`}>
          <span className={styles.bg}>Error - 404</span>
        </h1>
        <p className={styles.sub}>Сталася помилка, для того щоб продовжити:</p>
        <p>* Поверніться на головну сторінку.</p>
        <br />
        <p>* Повідомте нам про помилку та спробуйте пізніше ще раз.</p>
        <nav className={styles.nav}>
          <Link href="/" className={styles.link}>
            Повернутися на головну сторінку
          </Link>
           | 
        </nav>
      </div>
    </main>
  );
}
```

КОД СТОРІНКИ ЗВ’ЯЗКИ ТА СПІВРОБІТНИЦТВО

```
import Table from "@/components/Table/Table";
export default function Connection() {
  const headers = [
    "№ з/п",
    "Підприємство, установа (організація)-партнер",
    "Тема співробітництва",
    "Документ, в рамках якого здійснюється співробітництво",
    "Практичні результати від співробітництва",
  ];
  return (
    <main className="main">
      <div className="container">
        <h1 className="h-1">Зв'язки та співробітництво</h1>
        <Table headers={headers} rows={rows} />
      </div>
    </main>
  );
}
```

КОД СТОРІНКИ З КОНТАКТАМИ

```
import Table from "@/components/Table/Table";
import styles from "@/styles/Contacts.module.scss";
export default function Contact() {
  const contacts = [
    { id: 1, name: "Ольховська О.В., к.ф.-м.н.", email: "lena@olhovsky.name" },
```

```
    { id: 2, name: "Ольховський Д.М., доц.", email: "dmitriy@olhovsky.name" },
    { id: 3, name: "Парфьонова Т.О., доц.", email: "tapa.poltava@gmail.com" },
    { id: 4, name: "Чілікіна Т.В., доц.", email: "tv.0502@ukr.net" },
    { id: 5, name: "Черненко О.О., доц.", email: "oksanachernenko7@gmail.com" },
    { id: 6, name: "Олексійчук Ю.Ф., доц.", email: "olexijchuk@gmail.com" },
    { id: 7, name: "Кошова О. П., доц.", email: "koshova.o111@gmail.com" },
    { id: 8, name: "Карнаухова Г.В., ст. викл.", email: "ta.annet@gmail.com" },
    { id: 9, name: "Оріхівська О.Г., ст. викл.", email: "orikhivska@ukr.net" },
  ];
  return (
    <main className="main">
      <div className={`container`}>
        <h1 className="h-1 page-heading">Контакти</h1>
        <p className="text-1">
          Завідувач кафедри комп’ютерних наук та інформаційних технологій (КНІТ)
          – к.ф.-м.н.,{" "}
          <b>
            <i>Ольховська Олена Володимирівна.</i>
          </b>
        </p>
        <div className={styles.contact}>
          <p className="text-1">Кабінет 433а</p>
          <p className="text-1">телефон (0532) 509-204</p>
          <p className="text-1">внутр. тел. 2-04</p>
          <p className="text-1">
            e-mail: <a href="mailto:lena@olhovsky.name">lena@olhovsky.name</a>
          </p>
          <p className="text-1">моб. тел. 066 506 0968</p>
        </div>
        <br />
        <p className="text-1">Викладацька кафедри КНІТ:</p>
        <div className={styles.contact}>
          <p className="text-1">Кабінет 430а</p>
          <p className="text-1">телефон (0532) 509-204</p>
          <p className="text-1">внутр. тел. 2-13</p>
          <p className="text-1">
            e-mail: <a href="mmsi.puet@ukr.net">mmsi.puet@ukr.net</a>
            <a href="mailto:mmsi@puet.edu.ua">mmsi@puet.edu.ua</a>
          </p>
        </div>
        <br />
        <h2 className="h-2">Викладачі кафедри</h2>
        <Table
          headers={["№", "П.І.Б. викладача", "Ел. пошта"]}
          rows={contacts}
        />
      </div>
    </main>
  );
}
```

КОД СТОРІНКИ СКЛАД КАФЕДРИ

```
import staffData from "@/data/staff.json";
import StaffLargeList from "@/components/DepartmentStaff/StaffLargeList";
import styles from "@/styles/Department_staff.module.scss";
export default function DepartmentStaff() {
  return (
    <main className="main">
      <div className={`container`}>
        <h1 className="h-1 page-heading">Склад кафедри</h1>
        <StaffLargeList data={staffData.current} />
        <h2 className={`${styles.honorable_heading} h-1`}>
          Почесні працівники кафедри
        </h2>
        <StaffLargeList data={staffData.honorable} />
      </div>
    </main>
  );
}
```

КОД СТОРІНКИ НАУКОВО-ДОСЛІДНА РОБОТА

```
import ScienceTable from "@/components/Table/ScienceTable";
export default function ScienceAndResearch() {
  return (
    <main className="main">
      <div className={`container`}>
        <h1 className="h-1 page-heading">Науково-дослідна робота</h1>
        <h2 className="h-2 page-subheading">
          Наукові студентські гуртки кафедри КНІТ на 2023-24 н.р.
        </h2>
        <ScienceTable />
      </div>
    </main>
  );
}
```

КОД СТОРІНКИ СЕМІНАР

```
import Link from "next/link";
import SeminarLayout from "@/layout/SeminarLayout/SeminarLayout";
export default function Seminars() {
  return (
    <SeminarLayout>
      <div>
        <p className="text-1">
          В наступних файлах наведені правила оформлення матеріалів семінару
          "КНІТ"
        </p>
        <ul className="text-1">
          <li>
            <a href="files/knit1.doc">КНІТ Шаблон тез для pdf</a>{" "}
          </li>
          <li>
```

```
          <a href="files/knit2.doc">КНІТ Шаблон тез для rtf</a>{" "}
        </li>
        <li>
          <a href="files/knit3.doc">КНІТ правила оформлення</a>{" "}
        </li>
        <li>
          <a href="files/knit4.doc">КНІТ Формули</a>
        </li>
      </ul>
      <Link
        className="text-1"
        href={"http://www.matmodel.puet.edu.ua/files/knit-zb2022.pdf"}
      >
        Збірник праць науково-практичного семінару "Комп’ютерні науки та
        інформаційні технології" (КНІТ-2022). Випуск 1.
      </Link>
      <Link
        className="text-1"
        href={"http://www.matmodel.puet.edu.ua/files/knit-zb2022-2.pdf"}
      >
        Збірник праць науково-практичного семінару "Комп’ютерні науки та
        інформаційні технології" (КНІТ-2023). Випуск 2.
      </Link>
    </div>
  </SeminarLayout>
 );
}
```

КОД СТОРІНКИ ОСВІТНЬО-ПРОФЕСІЙНІ ПРОГРАМИ ТА НАВЧАЛЬНО-МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ

```
import ChoosenDisciplinesBachelor from "@/components/Syllabus/ChoosenDisciplinesBachelor";
import ChoosenDisciplinesMasters from "@/components/Syllabus/ChoosenDisciplinesMasters";
import List from "@/components/Syllabus/List";
import ListItem from "@/components/Syllabus/ListItem";
import SyllabusListBachelor from "@/components/Syllabus/SyllabusListBachelor";
import SyllabusListMasters from "@/components/Syllabus/SyllabusListMasters";
import styles from "@/styles/Methodology.module.scss";
import Link from "next/link";

export default function Methodology() {
 return (
  <main>
   <div className="container">
    <h1 className={`${styles.main_heading} h-1`}>
     Освітньо-професійні програми та навчально-методичне забезпечення
    </h1>
    <h2 className={`${styles.section_heading}`}>Бакалавр</h2>
    <Link
     className={styles.link_program}
     href={"http://www.matmodel.puet.edu.ua/files/122-np-b-2023.pdf"}
    >
```

```
  Освітня програма 122 "Комп'ютерні науки" бакалавра 2023
</Link>
<List>
  <ListItem heading="Рецензії">
    <ul className={styles.list_reviews}>
      <Link
        className="link"
        href={"http://www.matmodel.puet.edu.ua/files/rec122b23-1.pdf"}
      >
        Рецензія 1
      </Link>
      <Link
        className="link"
        href={"http://www.matmodel.puet.edu.ua/files/rec122b23-2.pdf"}
      >
        Рецензія 2
      </Link>
      <Link
        className="link"
        href={"http://www.matmodel.puet.edu.ua/files/rec122b23-3.pdf"}
      >
        Рецензія 3
      </Link>
      <Link
        className="link"
        href={"http://www.matmodel.puet.edu.ua/files/rec122b23-3.pdf"}
      >
        Рецензія 4
      </Link>
    </ul>
  </ListItem>
</List>
<Link
  className={styles.link_program}
  href={"http://www.matmodel.puet.edu.ua/files/compare-122-b.pdf"}
>
  Порівняльний аналіз освітньої програми Комп’ютерні науки ступеня
  бакалавра з провідними ЗВО
</Link>
<h2 className={`${styles.section_heading}`}>Магістр</h2>
<Link
  className={styles.link_program}
  href={"http://www.matmodel.puet.edu.ua/files/122-op-m-2023.pdf"}
>
  Освітня програма 122 "Комп'ютерні науки" магістра 2023
</Link>
<List>
  <ListItem heading="Рецензії">
    <ul className={styles.list_reviews}>
      <Link
        className="link"
```

```
            href={"http://www.matmodel.puet.edu.ua/files/rec122m-1.pdf"}
          >
            Рецензія 1
          </Link>
          <Link
            className="link"
            href={"http://www.matmodel.puet.edu.ua/files/rec122m-2.pdf"}
          >
            Рецензія 2
          </Link>
          <Link
            className="link"
            href={"http://www.matmodel.puet.edu.ua/files/rec122m-3.pdf"}
          >
            Рецензія 3
          </Link>
          <Link
            className="link"
            href={"http://www.matmodel.puet.edu.ua/files/rec122m-4.pdf"}
          >
            Рецензія 4
          </Link>
          <Link
            className="link"
            href={"http://www.matmodel.puet.edu.ua/files/rec122m-5.pdf"}
          >
            Рецензія 5
          </Link>
        </ul>
      </ListItem>
    </List>
    <Link
      className={styles.link_program}
      href={"http://www.matmodel.puet.edu.ua/files/compare-122-m.pdf"}
    >
      Порівняльний аналіз освітньої програми Комп’ютерні науки ступеня
      бакалавра з провідними ЗВО
    </Link>
    <h2 className={`${styles.section_heading}`}>
      Освітньо-професійні програми спеціальності 122 Комп'ютерні науки
    </h2>
    <h2 className={`${styles.section_heading}`}>
      СИЛАБУСИ ТА РОБОЧІ ПРОГРАМИ НАВЧАЛЬНИХ ДИСЦИПЛІН ДЛЯ СТУДЕНТІВ
ПЕРШОГО
      (БАКАЛАВРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ
    </h2>
    <SyllabusListBachelor />
    <h2 className={`${styles.section_heading}`}>
      Вибіркові освітні компоненти
    </h2>
    <ChoosenDisciplinesBachelor />
```

```
        <h2 className={`${styles.section_heading} `}>
          СИЛАБУСИ ТА РОБОЧІ ПРОГРАМИ НАВЧАЛЬНИХ ДИСЦИПЛІН ДЛЯ СТУДЕНТІВ
ДРУГОГО
          (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ
        </h2>
        <SyllabusListMasters />
        <h2 className={`${styles.section_heading} `}>
          Вибіркові освітні компоненти
        </h2>
        <ChoosenDisciplinesMasters />
      </div>
    </main>
  );
}
```