Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

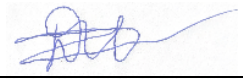
Кафедра Медіасистем та технологій
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка динамічних елементів та розгортання сайту кафедри МСТ
(тема)

Виконав:
студент 4 курсу, групи ВПВПС-18-2



Данг В.Л.
(прізвище, ініціали)

Спеціальність 186 Видавництво та поліграфія
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма
Видавничо-поліграфічна справа
(повна назва освітньої програми)

Керівник  проф. Єгорова І.М.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри МСТ _____ Дейнеко Ж.В.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Медіасистем та технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 186 Видавництво та поліграфія
Тип програми Освітньо-професійна
Освітня програма Видавничо-поліграфічна справа
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри МСТ_____
(підпис)
« 23 » травня 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Данг Вієт Лонг*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка динамічних елементів та розгортання сайту кафедри МСТ*

Затверджена наказом по університету від 21 травня 2022 р. № 558 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 21 червня 2022 р.

3. Вихідні дані до роботи
Мови розробки: React.js, CSS3; фреймворк – Gatsby; HTML-редактор – Visual Studio Code; середовище розповсюдження – Інтернет.

4. Перелік питань, що потрібно опрацювати в роботі
Вступ; Аналіз завдання на кваліфікаційну роботу, визначення цілей і задач проєктування; Аналітичний огляд досягнень у виробництві та застосуванні web-видань; Послідовність проєктування web-сайту; Вибір інструментальних засобів розробки; Проєктування іфнормаційної структури та навігації; Розробка модульної сітки; Створення динамічних елементів; наповнення контентом сторінок видання; Розгортання сайту; Економічне обґрунтування роботи; Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п. 5 включається до завдання за рішенням випускової кафедри)
Мета роботи; Вибір інструментальних засобів; React; Gatsby; GitHub; I18next; Gatsby Cloud; Створення динамічних елементів; Бургер меню; Пагінація; Інтернаціоналізація; Розгортання сайту; Економічна частина; Висновки.

6. Консультанти розділів роботи (п. 6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п. 1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Єгорова І.М.		21.06.2022
Економічна частина	проф. Полозова Т.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз завдання на кваліфікаційну роботу	23.05.2022	Вик.
2	Аналітичний огляд досягнень у виробництві та застосуванні web-видань	24.05.2022	Вик.
3	Послідовність проєктування web-сайту	24.05.2022	Вик.
4	Вибір інструментальних засобів розробки	25.05.2022	Вик.
5	Проєктування інформаційної структури та навігації	27.05.2022	Вик.
6	Розробка модульної сітки	28.05.2022	Вик.
7	Створення динамічних елементів	05.06.2022	Вик.
8	Розгортання сайту		
9	Економічна частиа	11.06.2022	Вик
9	Оформлення пояснювальної записки	12.06.2022	Вик.
10	Оформлення графічної частини	4.06	Вик.

Дата видачі завдання 23 травня 2022 р.

Студент _____ Данг В.Л.
(підпис)

Керівник роботи _____ проф. Єгорова І.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи містить: 47 с., 2 табл., 25 рис., 3 дод., 13 джерел.

REACT, GATSBY, ДИНАМІЧНІ ЕЛЕМЕНТИ, GITHUB.

Метою даної бакалаврської роботи є розробка динамічних елементів та розгортання сайту кафедри МСТ.

Об'єктом дослідження є процес розробки динамічних елементів для сайту .

У роботі розглянута технологія розробки динамічних елементів веб-сайту, розгортання веб-сайту для кафедри МСТ. Обрано та обгрунтовано необхідне програмне забезпечення; Також виконано економічне обґрунтування проекту.

ABSTRACT

Explanatory note of the qualification work: 47 p., 2 tabl., 25 pic., 3 app., 13 sources.

REACT, GATTSBY, DYNAMIC ELEMENTS, GITHUB.

The purpose of this bachelor's thesis is to develop dynamic elements and deploy the site of the Department of ITC.

The object of study is the process of developing dynamic elements for the site.

The technology of development of dynamic elements of the website, deployment of the website for the department of MST is considered in the work. Selected sound required software; The economic substantiation of the project was also performed.

ЗМІСТ

С.

ВСТУП

Зараз все більше речей переходить в онлайн, тому актуальність розробки сайту, що дозволяє студентам дізнаватись всю необхідну інформацію зростає. Це необхідно не тільки студентам, але в першу чергу викладачам.

На сайті завжди є елементи, які називають динамічними. Динамічними є рухомі чи анімовані елементи. Це можуть бути, наприклад, хмари, що рухаються на тлі сайту чи інші анімовані елементи, мета яких, як правило, декоративна.

Мета даної бакалаврської роботи – це створення динамічних елементів сайту кафедри МСТ не лише для декорації, а й для кращої роботи сайту та діяльності кафедри МСТ для залучення студентів.

Для виконання кваліфікаційної роботи нам необхідне:

– проаналізувати визначену предметну область;
– обрати найкращий варіант елементів для сайту,
– реалізувати елементи за допомогою коду;
– розробити функціонал роботи елементів.

1 АНАЛІЗ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Всі елементи, з яких створюються веб сайти, можна поділити на три типи: статичні, динамічні та інтерактивні:

– статичними є нерухомі елементи, з якими не передбачена взаємодія користувача. До них відносять, зокрема, тексти, зображення, декоративні елементи;

– динамічними є рухомі чи анімовані елементи. Це можуть бути, наприклад, хмари, що рухаються на тлі сайту чи інші анімовані елементи, мета яких, як правило, декоративна. Самі собою вони не становлять значного інтересу, тож в цьому матеріалі ми більшою мірою зосередимось на третьому типі елементів – інтерактивних;

– інтерактивними називають елементи, що так чи інакше реагують на дії користувача. Власне, слово «інтерактивні» буквально означає «взаємодіючі» (від англ. interactive). Базовими інтерактивними об’єктами є посилання, кнопки та елементи форм.

Мета данної кваліфікаційної роботи – створення динамічних елементів для оновленого сайту кафедри МСТ.

Усі сайти, відповідно до способу наповнення сторінок інформацією, умовно можна поділити на дві групи – статичні та динамічні. Між ними є відмінності.

Під статичним розуміється сайт, який складається із незмінних HTML-сторінок. Для прикладу – сайт-візитка. Тобто, користувач переглядає сторінку у тому вигляді, в якому вона зберігається на сервері.

Позитивними сторонами даного виду сайту є:

– економність використання;
– незначне навантаження на сервер;
– швидкість завантаження;
– легкість перенесення на інший сервер;
– простота створення HTML-сторінок.

Однак, є ряд недоліків:

– якщо Ви вирішили внести зміни на сайт (створення нового розділу, додання контенту тощо), то існує висока ймовірність, що доведеться правити усі сторінки, в результаті чого, це призведе до додаткових витрат;

– важкість у підтримці цілісності сайту;

– складність у забезпеченні розділення прав доступу до вмісту web-сайту.

Динамічний сайт**,** за аналогією, складається із динамічних, тобто, змінних сторінок. Варто зазначити, що такі сторінки формуються «на льоту» програмно, на основі запиту користувача:

– можливість самостійно вносити зміни на сайт, без допомоги фахівців;

– швидке відображення нововведених даних;

– простота в адмініструванні сайту та його верстки;

– широкий спектр функціональних можливостей.

Недоліки:

– високе навантаження на сервер;

– складність при перенесенні на новий хостинг;

– значні фінансові витрати.

Щоб краще зрозуміти в чому основна відмінність між цими видами сайту, узагальнимо основне правило: статична сторінка зберігається на сервері у незмінному вигляді, і в такому ж вигляді подається користувачеві, динамічна – генерується на основі запиту відвідувача.

Проаналізувавши усі переваги та недоліки, можна зробити висновок, що динамічний сайт надає більше можливостей для користувачів – простота змін в контенті, в елементах дизайну чи будь-яких інших. Статичний – дешевший у розробці, однак, дорожчий у відновленні ресурсу. Вибір було здійснено, враховуючи при цьому цілі розробки інтернет-ресурсу.

2 АНАЛІТИЧНИЙ ОГЛЯД ДОСЯГНЕНЬ У ВИРОБНИЦТВІ ТА ЗАСТОСУВАННІ WEB-ВИДАНЬ

Коли з'явилася бібліотека React – це на фундаментальному рівні змінило те, як працюють JavaScript-фреймворки та бібліотеки. У той час як інші подібні проекти просували ідеї MVC, MVVM , у React був обраний інший підхід. Зокрема, тут рендеринг візуальної складової програми був ізольований від представлення моделі. Завдяки React у фронтенд-екосистемі JavaScript з'явилася нова архітектура – Flux [1].

У 2013 році компанія Facebook щойно завершила серйозну роботу з інтеграції у свою платформу чату. Ця нова можливість була вбудована практично на кожну сторінку проекту, чат впливав на звичайні сценарії роботи з платформою. Це був складний додаток, вбудований в інший додаток, який і раніше не можна було назвати простим. Команді Facebook довелося зіткнутися з вирішенням нетривіальних завдань, справляючись з неконтрольованою мутацією DOM і необхідністю забезпечити паралельну асинхронну роботу користувачів у новому середовищі. [2]

Використовуючи популярні фронтенд-інструменти, що існували до React, нічого такого гарантовано забезпечити не можна. У ранніх веб-додатках «стан гонок» у DOM було однією з найпоширеніших проблем.

Головним завданням команди розробки React було вирішення цієї проблеми. Вони з нею впоралися, застосувавши два основні інноваційні підходи:

– однонаправлена прив'язка даних із використанням архітектури Flux;

– імутабельність стану компонента. Після того, як стан компонента встановлено, його вже не можна змінити. Зміни стану не торкаються візуалізованих компонентів. Натомість подібні зміни призводять до висновку нового уявлення, що має новий стан.

Бібліотека React спромоглася серйозно знизити гостроту проблеми неконтрольованих мутацій завдяки використанню архітектури Flux. Замість

приєднування до довільної кількості довільних об'єктів (моделей) обробники подій, що викликають оновлення DOM, бібліотека React дала розробникам єдиний спосіб управління станом компонента. Це диспетчеризація дій, що впливають на сховище даних. Коли змінюється стан сховища, система пропонує компоненту перерендеруватися. Архітектуру Flux показано на рисунку 2.1.

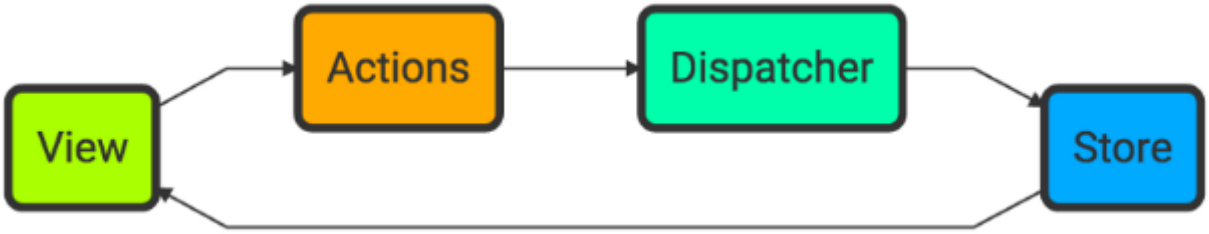


Рисунок 2.1 – Архітектура Flux

2.1 JSX

JSX - це розширення JavaScript, що дозволяє декларативно створювати компоненти інтерфейсу користувача. JSX має такі помітні можливості:

– застосування простої декларативної розмітки;

– код розмітки розташований там, де код компонента;

– реалізація принципу поділу відповідальності (наприклад - відокремлення опису інтерфейсу від логіки стану та від побічних ефектів). Причому реалізація, заснована не на використанні різних технологій (наприклад, HTML, CSS, JavaScript);

– абстрагування керування змінами DOM;

– абстрагування від особливостей різних платформ, для яких створюють програми React. Справа в тому, що завдяки використанню React можна створювати додатки, призначені для безлічі платформ (мова йде, наприклад, про розробку для мобільних пристроїв з використанням React Native, про програми для систем віртуальної реальності, про розробку для

Netflix Gibbon, про створення Canvas/WebGL -інтерфейсів, про проект (react-html-email) [3-5].

Якщо до появи JSX, потрібно було декларативно описувати інтерфейси, то не можна було обійтися без використання HTML-шаблонів. У ті часи не було загальновизнаного стандарту створення таких шаблонів. Кожен фреймворк використовував власний синтаксис. Цей синтаксис доводилося вивчати тому, кому, наприклад, потрібно було пройтися в циклі за деякими даними, вбудувати в текстовий шаблон значення змінних або прийняти рішення про те, який компонент інтерфейсу виводити, а який - ні.

У наші дні, якщо поглянути на різні фронтенд-інструменти, виявиться, що без спеціального синтаксису, як директива *ngFor з Angular, теж не обійтися. Але, оскільки JSX можна назвати надмножиною JavaScript, створюючи JSX-розмітку можна користуватися існуючими можливостями JS [3]. Приклад JSX-коду показано на рисунку 2.2.

```
const ItemList = ({ items }) => (
  <ul>
    {items.map((item) => (
      <li key={item.id}>
        <div>{item.name}</div>
      </li>
    ))}
  </ul>
);
```

Рисунок 2.2 – Приклад JSX-коду

При роботі з JSX необхідно враховувати деякі особливості, які, спочатку, можуть здатися незвичними.

Тут використовується підхід до іменування атрибутів елементів, який відрізняється від того, який прийнятий у HTML. Наприклад, class перетворюється на className. Йдеться застосування стилю іменування camelCase. Кожен елемент списку, який потрібно вивести, повинен мати

постійний унікальний ідентифікатор, призначений для використання в JSX-атрибуті key. Значення ідентифікатора має бути незмінним під час різних маніпуляцій з елементами списку. Насправді більшість елементів списків у моделях даних мають унікальні id, ці ідентифікатори зазвичай добре показують себе ролі значень для key [5-7].

React не нав'язує розробнику єдиний правильний спосіб роботи з CSS. Наприклад, компонент можна передати JavaScript-об'єкт зі стилями, записавши його у властивість style. За такого підходу більшість звичних імен стилів буде замінено на їх еквіваленти, записані за правилами camelCase. Але цим можливості роботи зі стилями не обмежуються. На практиці одночасно користуються різними підходами до стилізації React-додатків. Вибір конкретного підходу залежить від того, що потрібно стилізувати. Наприклад, глобальні стилі застосовуються для оформлення тем додатків та макетів сторінок, а локальні стилі для налаштування зовнішнього вигляду конкретного компонента.

2.2 Хуки React

У React 16.8 з'явилася нова концепція – хуки React. Це функції, які дозволяють підключатися до подій життєвого циклу компонентів, не користуючись синтаксисом класів і не звертаючись до методів життєвого циклу компонентів. Компоненти, в результаті, стало можливим створювати не у вигляді класів, а у вигляді функцій.

Виклик хука, в цілому, означає появу побічного ефекту - такого, що дозволяє компоненту працювати зі своїм станом і підсистемою введення-виведення. Побічний ефект – це будь-яка зміна стану, видима за межами функції, за винятком зміни значення, що повертається функцією.

Ось що дають нам хуки React:

– вони дозволяють створювати компоненти, представлені як функцій, а чи не як класів;

– вони допомагають краще організовувати код;

– завдяки їм спрощується спільне використання однієї й тієї ж логіки у різних компонентах;

– нові хуки можна створювати, виконуючи композицію існуючих хуків (викликаючи їх з інших хуків).

В цілому, рекомендується користуватися функціональними компонентами та хуками, а не компонентами, заснованими на класах. Функціональні компоненти зазвичай компактніші за компоненти, засновані на класах. Їхній код краще організований, відрізняється кращою читабельністю, краще підходить для багаторазового використання, його легше тестувати.

3 ПОСЛІДОВНІТЬ ВИГОТОВЛЕННЯ WEB-САЙТУ

3.1 Збір інформації: призначення, основні цілі

Етап попереднього дослідження та збору інформації визначає те, як будуть протікати всі наступні стадії розробки. Найважливіше на цьому етапі – отримати ясне та повне розуміння того, яким буде призначення вашого майбутнього сайту, яких цілей ви хочете досягти за його допомогою, а також якою є цільова аудиторія, яку ви хочете на нього залучити. Така своєрідна анкета веб-розробки дозволить визначити якнайкращу стратегію подальшого розвитку проекту. Новинки відрізняються від розважальних сайтів, а сайти для підлітків відрізняються від таких для дорослої аудиторії. Різні сайти надають відвідувачам різну функціональність, а отже, різні технології повинні використовуватися в тому чи іншому випадку. Детально складений план, створений на основі даних, отриманих на цьому етапі, може запобігти вам витрати додаткових ресурсів на вирішення непередбачених труднощів, таких як зміна дизайну або додавання функціоналу, не передбаченого спочатку.

3.2 Створення карти сайту

На цій стадії розробки можна отримати уявлення про те, яким буде майбутній сайт. На основі інформації, зібраної на попередній стадії, створюється мапа сайту (sitemap). Так, наприклад, виглядає карта сайту ІксБі Софтварі.

Карта сайту описує зв'язок між різними частинами вашого сайту. Це допомагає зрозуміти, наскільки зручним у використанні він буде. По карті сайту можна визначити «відстань» від головної сторінки до інших сторінок, що допомагає судити про те, наскільки просто користувачеві буде дістатися до інформації, що його цікавить. Основна мета створення карти сайту –

створити легкий з погляду навігації та дружній до користувача продукт. Це дозволяє зрозуміти внутрішню структуру майбутнього сайту, але не описує, як сайт виглядатиме. Іноді, перш ніж приступити до написання коду або розробки дизайну, може бути важливим отримати схвалення замовника. І тут створюється макет (wireframe чи mock-up). Макет є візуальним уявленням майбутнього інтерфейсу сайту. Але, на відміну, наприклад, від шаблону, про який ми поговоримо далі, він не містить елементів дизайну, таких як колір, логотипи, тощо. Він тільки описує, які елементи будуть розміщені на сторінці і як вони будуть розташовані. Макет є свого роду нарис майбутнього сайту. Ви можете використовувати один із доступних онлайн-сервісів для створення макетів. Зазвичай ми використовуємо Moqups.

Також на цьому етапі варто визначитися з тим, який стек технологій (мова програмування, фреймворки, CMS) буде використано.

3.3 Дизайн

На цьому етапі веб-сайт стає ще ближче до своєї остаточної форми. Весь візуальний контент, такий, як зображення, фото та відео, здається саме зараз. І опять-таки вся інформація, яка була зібрана на самій першій стадії проекту, крайне важливо на цьому етапі. Цікаві замовлення, а також цільова аудиторія повинні враховуватися в першу чергу під час роботи над дизайном. Дизайнером на цьому етапі створюється шаблон сторінки (макет сторінки). Основне призначення шаблону – візуальне зображення структури сторінок, її зміст, а також відображення основної функції. На цей раз, у відмінності від макета, використовуються елементи дизайну. Шаблон містить колір, логотипи та зображення. Він дає можливість судити про те, як в кінцевому результаті буде виглядати готовий сайт. Після створення шаблону може бути відправлений заказчик. Після огляду замовником виконаної роботи, він присилає свій відгук. Якщо його не враховують які-то аспекти дизайну, ви повинні змінити існуючий

шаблон і знову відправляти його заказчику. Цей цикл повторюється до тех пор, поки замовник не буде повністю задоволеним результатом.

3.4 Створення контенту

Процес створення контенту зазвичай відбувається паралельно з іншими стадіями розробки та її роль не можна недооцінювати. На цьому кроці необхідно описати суть того, що ви хочете донести до аудиторії свого веб-сайту, а також додати CTA (заклик до дії). Ця стадія включає також створення привабливих і помітних заголовків, написання і редагування тексту, компіляція існуючих текстів і т.д. Все це потребує витрати додаткового часу та зусиль. Як правило, замовник надає контент, вже готовий бути розміщеним на сайті. Важливо, щоб весь контент був підготовлений до стадії розробки.

3.5 Верстка та створення динамічних елементів

Тепер можна перейти безпосередньо до верстки сайту. Усі графічні елементи, розроблені раніше, застосовуються на даній стадії. Зазвичай у першу чергу створюється домашня сторінка, а потім до неї додаються решта сторінок відповідно до ієрархії, розробленої на етапі створення карти сайту. Також на цьому етапі відбувається встановлення CMS. Усі статичні елементи веб-сайту, дизайн яких був розроблений раніше під час створення шаблону, перетворюються на реальні динамічні інтерактивні елементи веб-сторінки. Важливим завданням є проведення SEO-оптимізації (Search Engine Optimization), яка є оптимізацією елементів веб-сторінки (заголовків, опису, ключових слів) з метою підняття позицій сайту в результатах видачі пошукових систем. Валідність коду є дуже важливою в цьому випадку.

3.6 Тестування та запуск

Тестування є, мабуть, найрутиннішою частиною розробки. Кожне посилання має бути перевірено, кожна форма та кожен скрипт мають бути протестовані. Текст повинен бути перевірений програмою перевірки орфографії для виявлення можливих помилок та помилок. Валідатори коду використовуються для того, щоб бути впевненим, що створений на попередньому етапі код відповідає сучасним веб-стандартам. Це може бути дуже важливим, якщо вам критична, наприклад, кросбраузерна сумісність. Після того, як сайт було перевірено, він може бути завантажений на сервер. Зазвичай для цього використовується FTP-клієнт. Після завантаження сайту на сервер, необхідно провести ще один тест для того, щоб бути впевненим, що під час завантаження не відбулося непередбачених помилок і всі файли цілі та неушкоджені.

Потрібно постійно пам'ятати, що процес розробки веб-сайту не починається з написання коду та не закінчується після запуску сайту. Етап підготовки торкається всіх наступних етапів, визначаючи те, наскільки продуктивним виявиться процес роботи над проектом. Грунтовне та глибоке дослідження таких аспектів, як стать, вік та інтереси кінцевих користувачів може виявитися визначальним. Підтримка сайту вже після його запуску також дуже важлива. Потрібно бути достатньо оперативним, щоб мати можливість швидко виправляти помилки, що виникли, і вирішувати проблеми, що виникли в користувачів. Розуміння того, що серед етапів розробки веб-сайту немає таких, які можна було б вважати неважливими або необов'язковими, допоможе уникнути зайвого клопоту і дасть впевненість у тому, що робота над проектом рухається так, як було задумано і ви маєте повний контроль над процесом розробки.

4 ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РОЗРОБКИ

На кожному із етапів розробки сайту було використано багато програм, за допомогою яких було розроблено та оформлено web-додаток. Для кожного з аспектів створення Інтернет-сторінок використовують різні засоби: графічні редактори, текстові редактори, системи керування вмістом, локальні сервери, браузери та інші компоненти.

Під час проектування web-сайту кафедри МСТ було використано такі інструментальні засоби:

- webstorm;
- графічний редактор Figma;
- сайт для вибору шрифту Google Fonts;
- react;
- gatsby
- font awesome;
- gatsby cloud;
- фреймворк інтернаціоналізації I18next;
- github;
- NPM.

Розглянемо кожну програму детальніше.

4.1 WebStorm

JetBrains WebStorm – інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей.WebStorm постачається з перед-установленим плагінами JavaScript (такими як для Node.js), котрі доступні для PhpStorm безкоштовно.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) – в таких конструкціях підтримується коректний рефакторинг.

4.2 Figma

Figma - крос-платформний онлайн-сервіс для дизайнерів інтерфейсів і веб-розробників. Розробка інтерфейсів відбувається в онлайн-додатку. У Figma дві ключові особливості: доступ до макету прямо з вікна браузера і можливість спільної роботи над документами.

До появи Figma кільком дизайнерам складно було працювати над одним проектом і передавати макети розробникам. Photoshop відмовлявся коректно відкривати макет, поки ви не встановите потрібні шрифти. Або колега вносив зміни в свою копію проекту і забував сказати вам про це.

4.3 Google Fonts

Бібліотека понад 950 вільно розповсюджуваних шрифтів, інтерактивний каталог для їх перегляду і прикладні програмні інтерфейси для використання веб-шрифтів за допомогою CSS і на Андроїд.яКаталог Шрифтів Google призначений для того, щоб забезпечити пошук та дослідження шрифтів, а сервіс широко використовується із понад 17 трильйонами поданих шрифтів, а це означає, що в середньому всі 952 шрифти були завантажені понад 19 мільярдів разів, і що кожна людина на Землі в середньому завантажила кожен шрифт принаймні два-три рази.

Бібліотека підтримується через офіційний репозитарій GitHub github.com/google/fonts, де напряму доступні шрифтові файлі. Джерельні файли шрифтів доступні на git репозиторіях Github організації github.com/googlefonts, разом із вільно поширюваним інструментарієм, який використовується спільнотою Google Fonts.

4.4 React

React - відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS [7-9]. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови вебзастосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux. В даний час React використовують Khan Academy, Netflix, Yahoo, Airbnb, Sony, Atlassian та інші.

4.5 Gatsby

Gatsby – це платформа для створення програм та веб-сайтів з використанням React. Це один із інструментів, який дозволяє вам

використовувати набір технологій та практик під загальною назвою JAMstack. Гетсбі зараз є однією з найпопулярніших технологій у галузі розробки сайтів. Чому? Можливі причини:

– вибух підходу JAMstack до створення веб-додатків та веб-сайтів;

– швидке прийняття Прогресивні веб-додаткитехнології в галузі, що є однією з ключових особливостей Gatsby;

– це вбудоване в React і GraphQL, які є двома дуже популярними технологіями, що розвиваються;

– це дійсно потужно;

– це швидко;

– документація відмінна;

– мережевий ефект (люди використовують його, створюють сайти, роблять уроки, люди знають про нього більше, створюють цикл);

– все написано на JavaScript (не потрібно вивчати нову мову шаблонів);

– спочатку він приховує складність, але дає нам доступ до кожного кроку для налаштування.

4.6 Font Awesome

Font Awesome – набір інструментів для створення шрифтів та іконок на основі мов програмування CSS та LESS. Він був розроблений Дейвом Ганді для використання з Twitter Bootstrap, пізніше включеним до BootstrapCDN. На Font Awesome припадає 20% ринку тих веб-сайтів, які використовують скрипти сторонніх шрифтів на своїй платформі, займаючи друге місце після Google Fonts [2].

Версія 5 була випущена 7 грудня 2017 з 1278 значками. Він доступний у двох пакетах: Font Awesome Free та платний Font Awesome Pro. Безкоштовні версії (всі версії до версії 4 та безкоштовна версія 5) доступні за ліцензією SIL Open Font License 1.1, Creative Commons 4.0 та MIT.

4.7 Gatsby Cloud

Gatsby Cloud – це комерційна платформа, створена командою Gatsby, Inc. Він розроблений для фреймворка Gatsby з відкритим кодом. Cloud розпочався у 2019 році лише з купою функцій, включаючи інтеграцію з безголовою CMS та попередній перегляд у реальному часі. Це був великий крок вперед до вирішення болючої проблеми Гетсбі – довгих збірок. Вони надали редакторам вмісту можливість миттєвого попереднього перегляду нових змін без необхідності повністю перебудовувати (що іноді може зайняти десятки хвилин).

4.8 I18next

I18next – це фреймворк інтернаціоналізації, написаний на та для JavaScript. Але це набагато більше, ніж це.

i18next виходить за рамки просто надання стандартних функцій i18n, таких як (множина, контекст, інтерполяція, формат). Він надає повне рішення для локалізації вашого продукту з Інтернету на мобільний і настільний комп’ютер.

Спільнота i18next створила інтеграцію для зовнішніх фреймворків, таких як React, Angular, Vue.js та багато інших. Але на цьому все не закінчується. Також можна використовувати i18next з Node.js, Deno, PHP, iOS, Android та іншими платформами.

Більшість фреймворків вирішують, як завантажуються переклади. i18next несе відповідальність за визначення мови користувача, завантаження перекладів і введення їх у фреймворк. i18next надасть вам плагіни щоб:

- визначити мову користувача;
- завантажити переклади;
- за бажанням кешувати переклади.

4.9 Github

GitHub – один з найбільших вебсервісів для спільної розробки програмного забезпечення. Існують безкоштовні та платні тарифні плани користування сайтом. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc

Сервіс безкоштовний для проєктів з відкритим вихідним кодом, з наданням користувачам усіх своїх можливостей (включаючи SSL), а для окремих індивідуальних проєктів пропонуються різні платні тарифні плани. 21 вересня 2011 року кількість користувачів стала більшою за мільйон [3].

Розробники сайту називають GitHub «соціальною мережею для розробників».

Окрім розміщення коду, учасники можуть спілкуватись, коментувати редагування один одного, а також слідкувати за новинами знайомих. За допомогою широких можливостей Git програмісти можуть поєднувати свої репозиторії – GitHub дає зручний інтерфейс для цього і може показувати вклад кожного учасника в вигляді дерева.

Для проєктів є особисті сторінки, невеликі Вікі та система відстеження помилок. Прямо на сайті можна дивитись файли проєктів з підсвічуванням синтаксису для більшості мов програмування.

– на платних тарифних планах можна створювати приватні репозиторії, доступні обмеженому колу користувачів;

– є можливість прямого додавання нових файлів в свій репозиторій через вебінтерфейс сервісу;

– код проєктів можна не лише скопіювати через Git, але й завантажити у вигляді архіву. (Для цього достатньо додати /zipball/master/ в кінець адресного рядка.);

– окрім Git, сервіс підтримує отримання і редагування коду через SVN і Mercurial;

– на сайті є pastebin-сервіс[13] для швидкої публікації фрагментів коду;

– зберігання документації, включаючи автоматично відтворювані файли README у різних форматах файлів типу Markdown;

– вкладені списки завдань у файлах;

– візуалізація геопросторових даних;

– 3D-рендеринг файлів, які можна попередньо переглянути, використовуючи новий інтегрований переглядач файлів STL, який відображає файли на «3D canvas». Переглядач підтримує WebGL і Three.js;

– внутрішній формат PSD для Photoshop з можливістю попередньо перегляду та порівняння з попередніми версіями того самого файлу.

Раніше Ruby-проєкти могли бути автоматично опубліковані в RubyGems-репозиторії сервісу, але в жовтні 2009 GitHub відмовився від цього сервісу.

4.10 NPM

NPM (Node Package Manager) - це менеджер пакунків для мови програмування JavaScript. Для середовища виконання Node.js це менеджер пакунків за замовчуванням. Включає в себе клієнт командного рядка, який також називається npm, а також онлайн-базу даних публічних та приватних пакунків, яка називається реєстром npm. Реєстр доступний через клієнт, а доступні пакунки можна переглядати та шукати через вебсайт npm. Менеджер пакунків та реєстр керуються npm, Inc.

NPM включено як рекомендовану функцію в Node.js інсталяторі. npm складається з клієнта командного рядка, який взаємодіє з віддаленим реєстром. Це дозволяє користувачам користуватися модулями JavaScript та розповсюджувати їх. Пакунки в реєстрі знаходяться у форматі CommonJS і включають в себе файли метаданих у форматі JSON В головному реєстрі npm доступно понад 477 000 пакунків. Реєстр не має процедури перевірки, а це

означає, що знайдені там пакунки можуть бути низькоякісними або небезпечними. Натомість npm спирається на звіти користувачів, щоб видаляти пакунки, якщо вони порушують політику безпеки (є незахищеними, зловмисними або низькоякісними). npm показує статистику, включаючи кількість завантажень та кількість пакунків, щоб допомогти розробникам оцінювати якість пакетів.

npm може управляти пакунками, які є локальними залежностями певного проекту, а також глобально інстальованими інструментами JavaScript. При використанні npm як менеджера залежності для локального проекту, можна встановити одною командою всі залежності проекту через файл package.json. У файлі package.json кожна залежність може визначати діапазон дійсних версій, використовуючи схему семантичної версії, що дозволяє розробникам автоматично оновлювати свої пакети, одночасно уникаючи небажаних змін.

5 РОЗРОБКА МОДУЛЬНОЇ СІТКИ

Модульна сітка – це сукупність невидимих прямих ліній що взаємоперпендикулярні між собою та формують в місцях перетину вершини можливих модулів. Об'єднання декілька модулів між собою дозволяють сформувати майбутні колонки у багатоколонкових виданнях. Модульна сітка дозволяє розміщувати елементи на макеті, забезпечувати візуальний зв'язок між окремими блоками та надає друкованому виданню цілісності та єдності стилю.

Її створення стає актуальним тоді, коли готовий приблизний макет сайту. На цій стадії вона допоможе структурувати всю інформацію та вирівняти кожну частину контент. Зазвичай використовується сітка, що має лише вертикальні колонки, кількість яких варіюється в залежності від сайту, який розробляється [7-13].

Сітка також має бути адаптивною. Вона повинна підлаштовуватись під параметри пристрою, на якому переглядається web-сторінка. Разом з нею, під певний розмір екрану підлаштовуються усі елементи, розміщені в ній.

Для проєктування сайту кафедри було створено модульну сітку, що складається з дванадцяти колонок. Ширина кожної з них дорівнює 60 пікселів, а ширина проміжків між ними – 40 пікселів. Такі параметри має сітка, що відповідає розташуванню елементівсторінки на моніторі з шириною екрану від 1440 пікселів (рис. 5.1).

При ще меншому розміру екрану сітка має ще менше колонок. Так, при роздільній здатності 576 пікселів в ширину (еквівалент деяких мобільних пристроїв), вона складається з шести колонок по 60 пікселів з шириною проміжків 20 пікселів [9-12] (рис. 5.2).

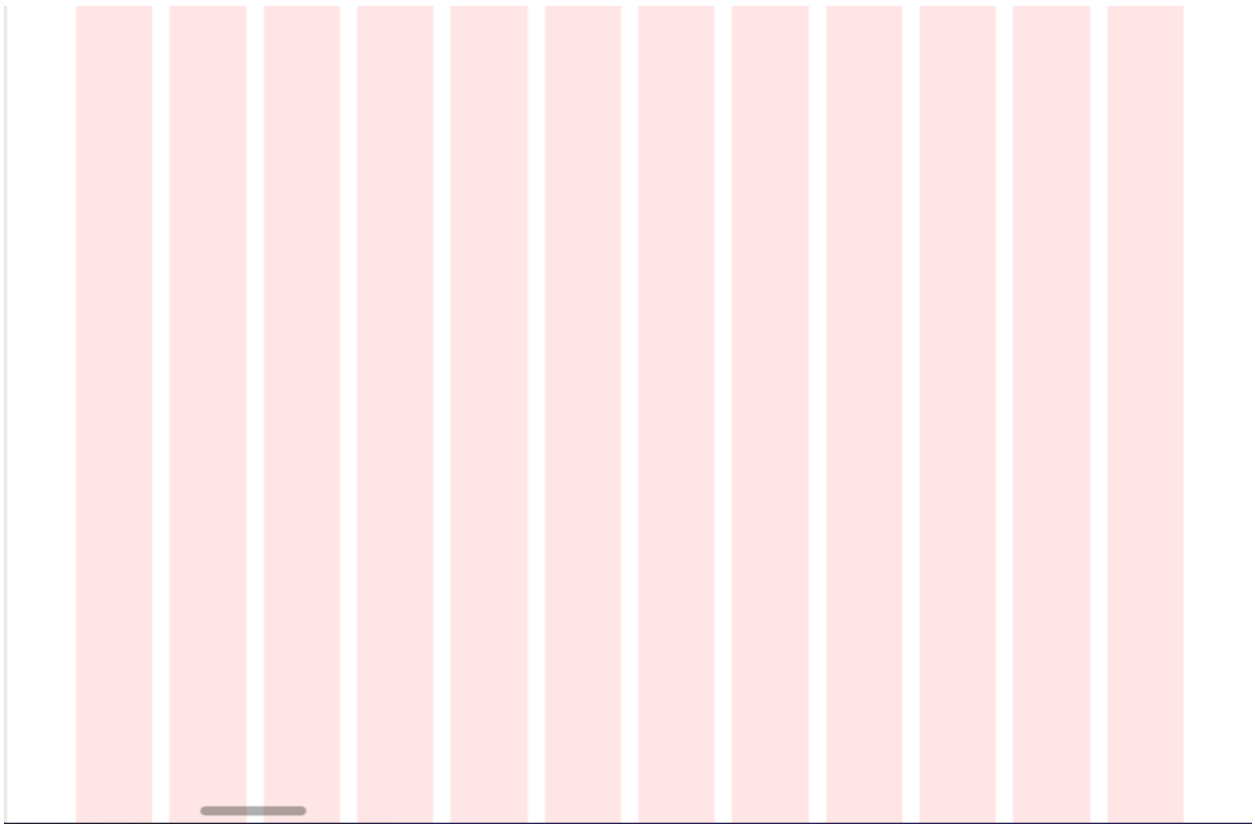


Рисунок 5.1 – Модульна сітка для десктопної версії

Рисунок 5.2 – Модульна сітка для мобільної версії

6 СТВОРЕННЯ ДИНАМІЧНИХ ЕЛЕМЕНТІВ

Розробку веб-сайту проведено в інтегрованому середовищі розробки Webstorm. До праці були встановлені та підключені усі необхідні технології, за допомогою пакетного менеджера NPM. За допогою команди “npm start” запускали проект та починали розробку [13]. Далі ми починаємо писати код для створення динамічних елементів. Список встановлених плагінів для створення елементів показано на рисунку 6.1

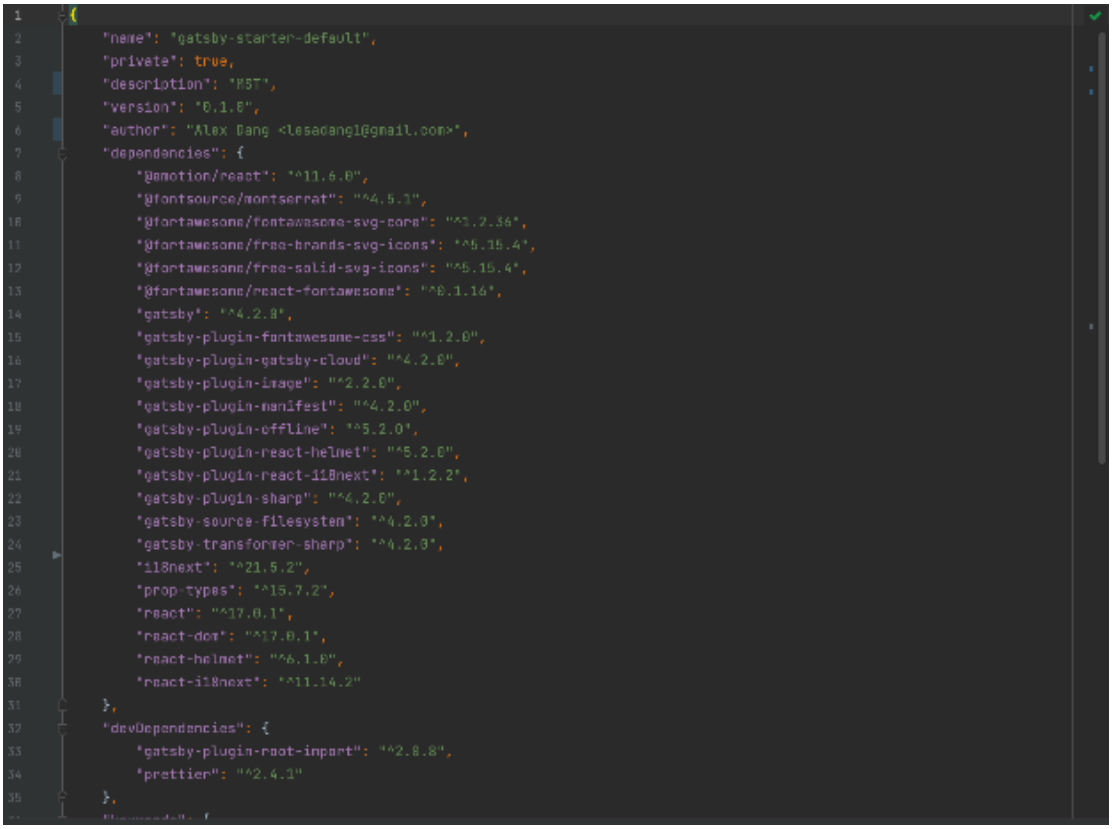


Рисунок 6.1 – Список встановлених плагінів

6.1 Бургер меню

Для створення бургер меню був використаний Font Awesome для відображення іконок. Бургер меню з'являється для пристроїв які мають ширину менш ніж 576px. Бургер меню показано на рисунку 6.2.

Рисунок 6.2 – Бургер меню

Якщо натиснути на бургер меню ми побачимо головні розділи. Це було реалізовано за допомогою реакту. При натиску на меню додавався клас на блок, який відображає меню. Як працює бургер меню показано на рисунку 6.3.

Щоб відобразити розділи був створений масив, який потім ітерувався в jsx. Це дозволяло не дублювати код та робити його легшим для написання та подальшого підтримки функціоналу. Приклади масиву показано на рис. 6.4.

Якщо головний розділ мав свої підрозділи, то натиснувши на нього він буде виділений, а під ним з'являться його підрозділи. Щоб перейти знову на головні підрозділи потрібно натиснути на виділений розділ. Як працюють підрозділи у бургер меню показано на рисунку 6.5.

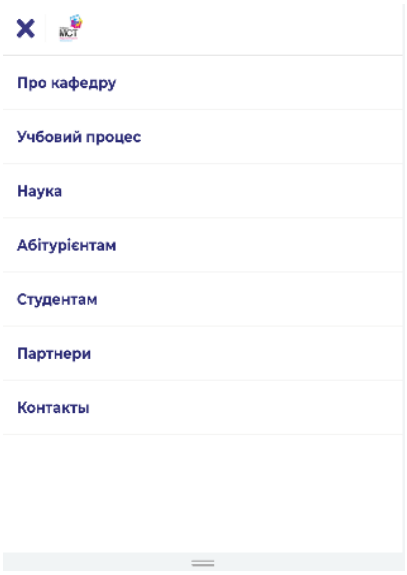
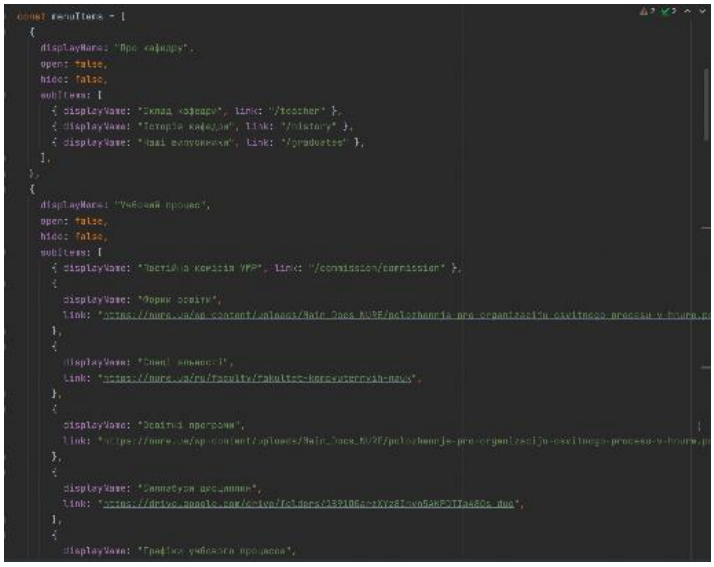


Рисунок 6.3 – Відкрите бургер меню



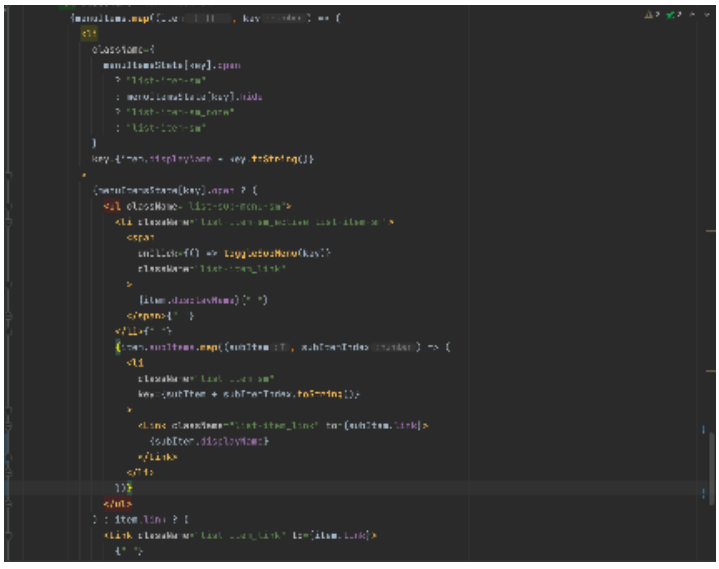


Рисунок 6.4 – Приклад використання масиву

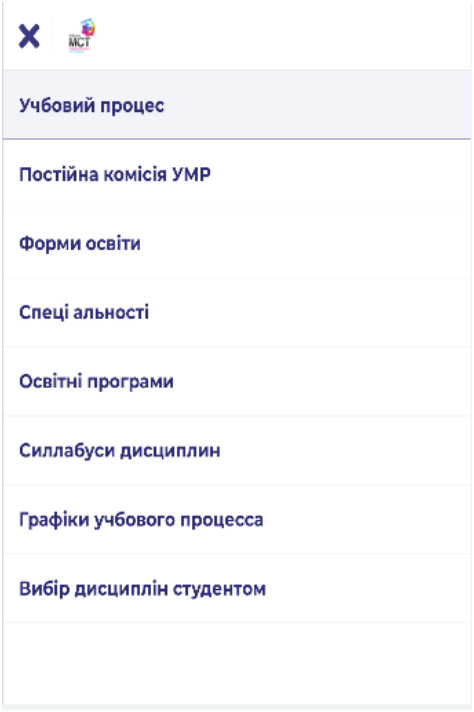


Рисунок 6.5 – Підрозділи у бургер меню

Для написання цього була створена функція “toggleSubMenu”, який ховав та показував підрозділи. Код функції toggleSubMenu показано на рис. 6.6.

```
const [burgerMenu, setBurgerMenu] = useState( initialState: false)
const [menuItemsState, setMenuItemsState] = useState(menuItems)

const toggleSubMenu = key => {
  setMenuItemsState(
    menuItemsState.map((item :{...}| ... , index :number ) => {
      if (index === key) {
        item.open = !item.open
      } else {
        item.hide = !item.hide
      }

      return item
    })
  )
}
```

Рисунок 6.6 – Функція toggleSubMenu

Повний код бургер меню показано у додатку А.

6.2 Пагінація

Приклад елементу пагінації показано на рисунку 6.7

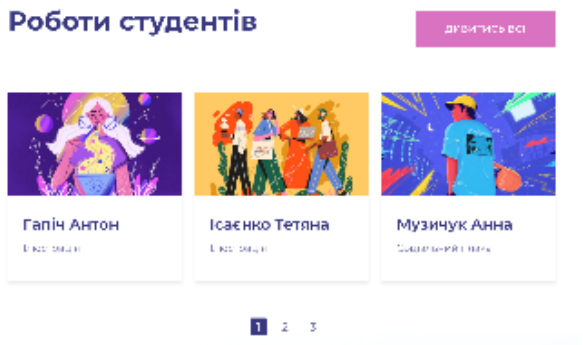


Рисунок 6.7 – приклад пагінації на сайті

Для створення пагінації було створено окремий компонент, щоб мати можливість його використовувати в декількох сторінках одразу. Цей компонент отримує параметром кількість елементів, ліміт елементів на сторінку та метод, який змінює сторінку. Приклад вигляду пагінації в коду показано на рисунку 6.8.

```
import React, { useState } from "react"
import "./pagination.css"

const Pagination = ({ items, limit = 3, handlePageChange }) => {
  const [currentPage, setPage] = useState(initialState: 1)
  const pages = Math.ceil(x: items / limit)

  return (
    <div className="works__pagination">
      {Array(pages)
        .fill(value: "")
        .map((_, index: number) => (
          <span
            key={index.toString()}
            className={
              index + 1 === currentPage
                ? "works__pagination__item active"
                : "works__pagination__item"
            }
            onClick={() => {
              setPage(value: index + 1)
              handlePageChange(index + 1)
            }}
          >
            {index + 1}
          </span>
        ))}
    </div>
  )
}
export default Pagination
```

Рисунок 6.8 – Приклад коду пагінації

Повний код пагінації показано у додатку Б.

6.3 Інтернаціолізація

Щоб підключити інтернаціолізацію в Gatsby був встановлений плагін “gatsby-plugin-react-i18next”. Після встановлення плагіну потрібно налаштувати конфігурацію, щоб сайт перекладав необхідні мови у файлі “gatsby-config.js” (рис. 6.9).

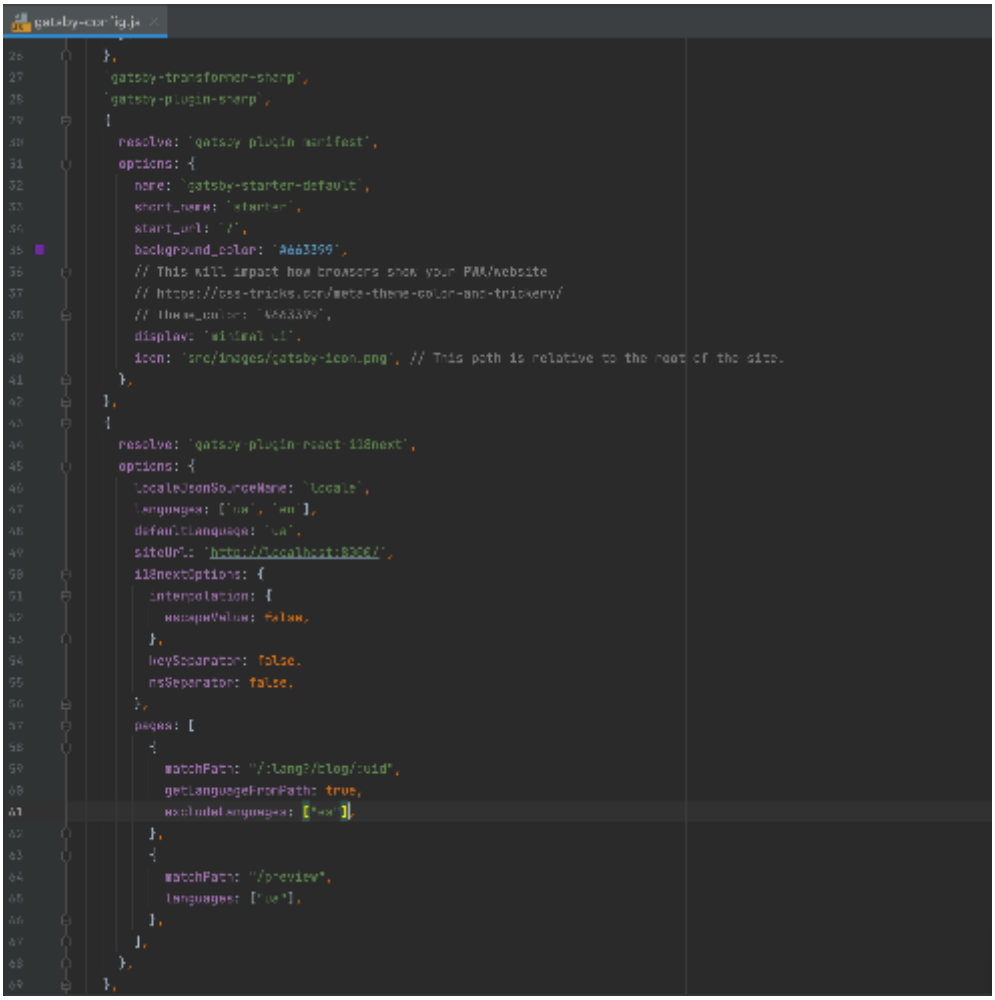


Рисунок 6.9 – Файл gatsby-config.js

У файлі конфігурації вказуємо, що використовуємо українську та англійську мову на сайті, за замовчуванням - українську. Після цього потрібно створити папку locales, в якій ще створюємо папку з необхідними для нас мовами з файлом “translation.json” (рис. 6.10).

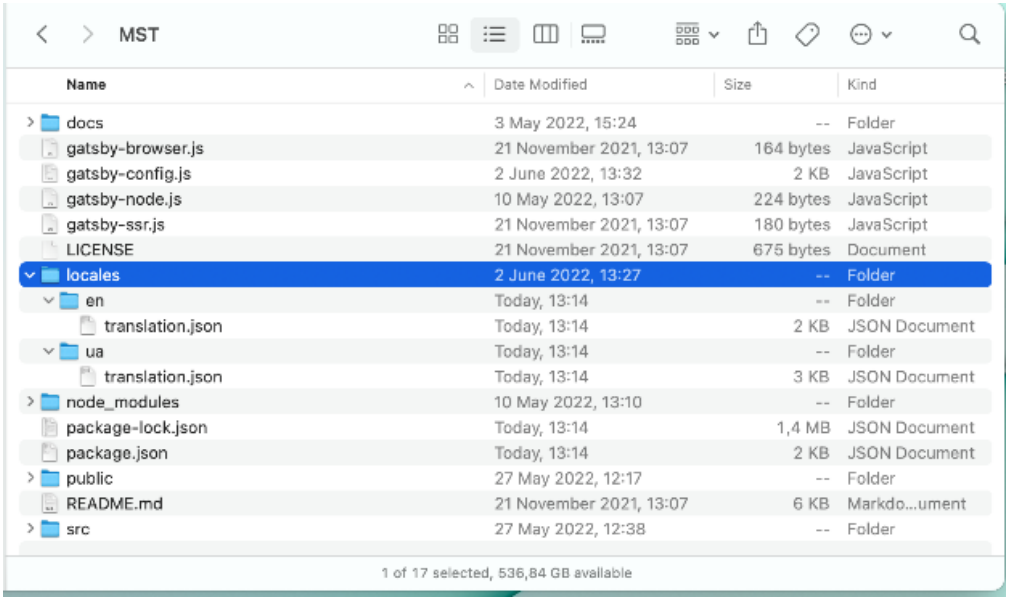


Рисунок 6.10 – Файли translation.json

Тепер маємо можливість перекладати слова, де необхідно. Для цього використовуємо тег “<Trans></Trans>” (рис 6.11) та всередину нього пишемо змінну, яка береться з файлів “translation.json”. Приклад написання тексту у translation.json показано на рисунку 6.12.

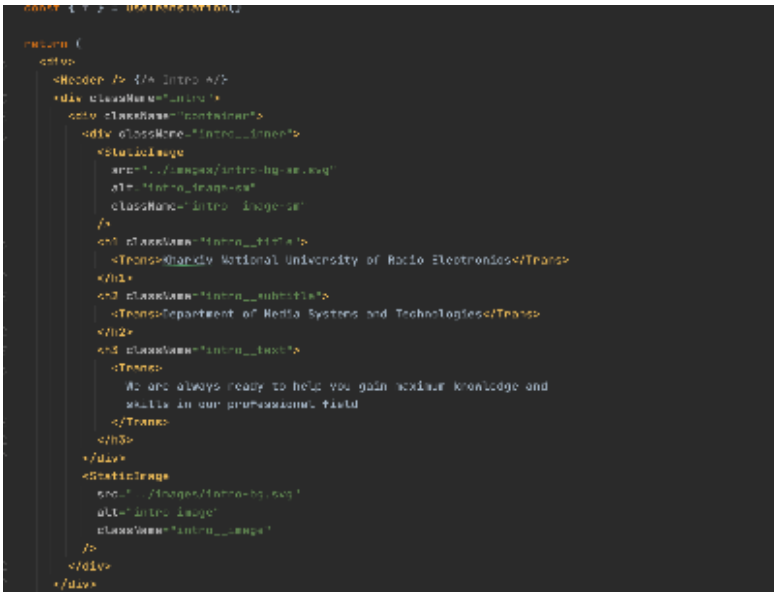


Рисунок 6.11 – Використання тегу <Trans>

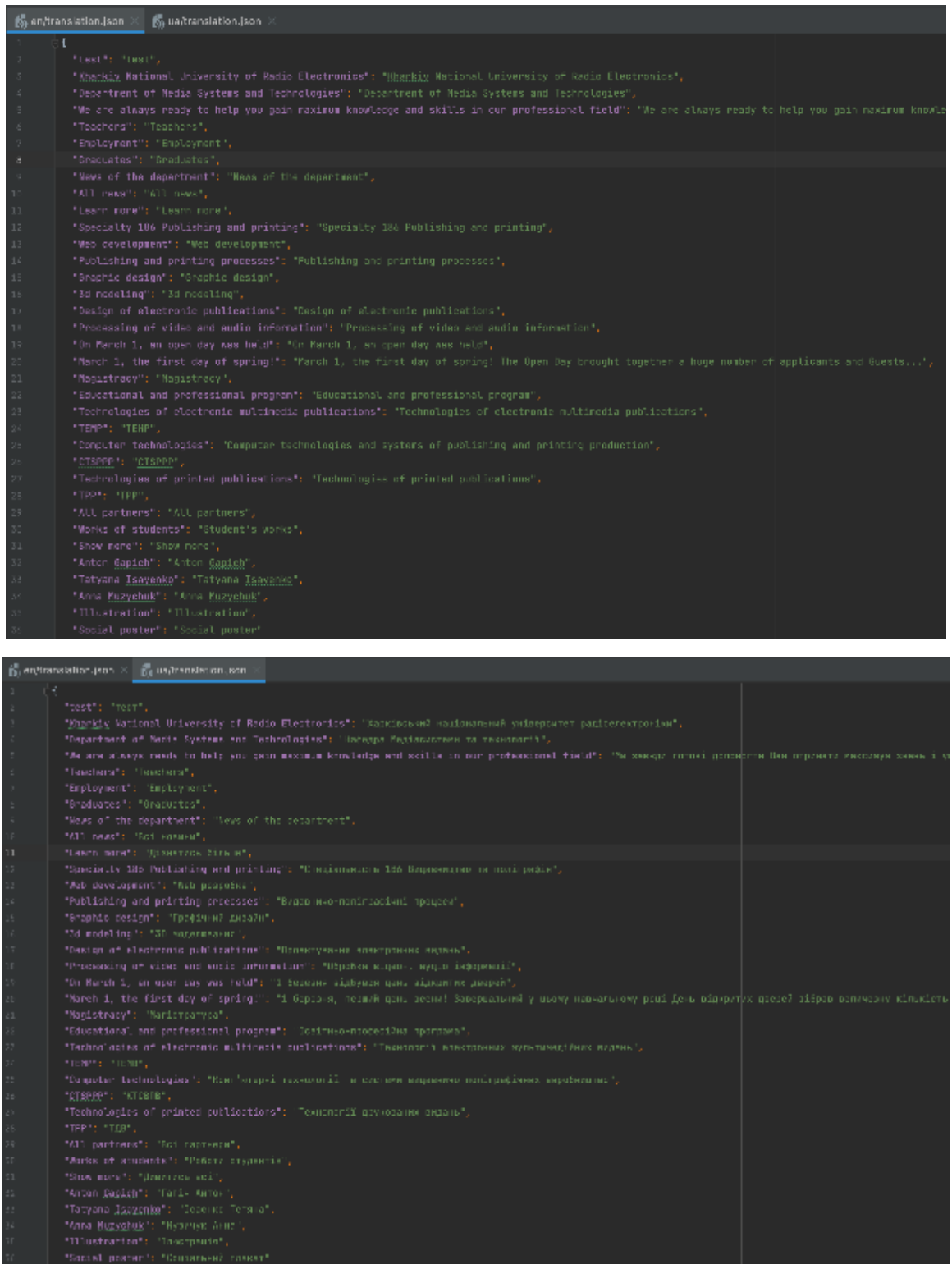


Рисунок 6.12 – Написання тексту у translation.json

Тепер можемо перейти на нашу сторінку та подивитись, як сайт працює українською та англійською мовами (рис. 6.14) та (рис. 6.15).

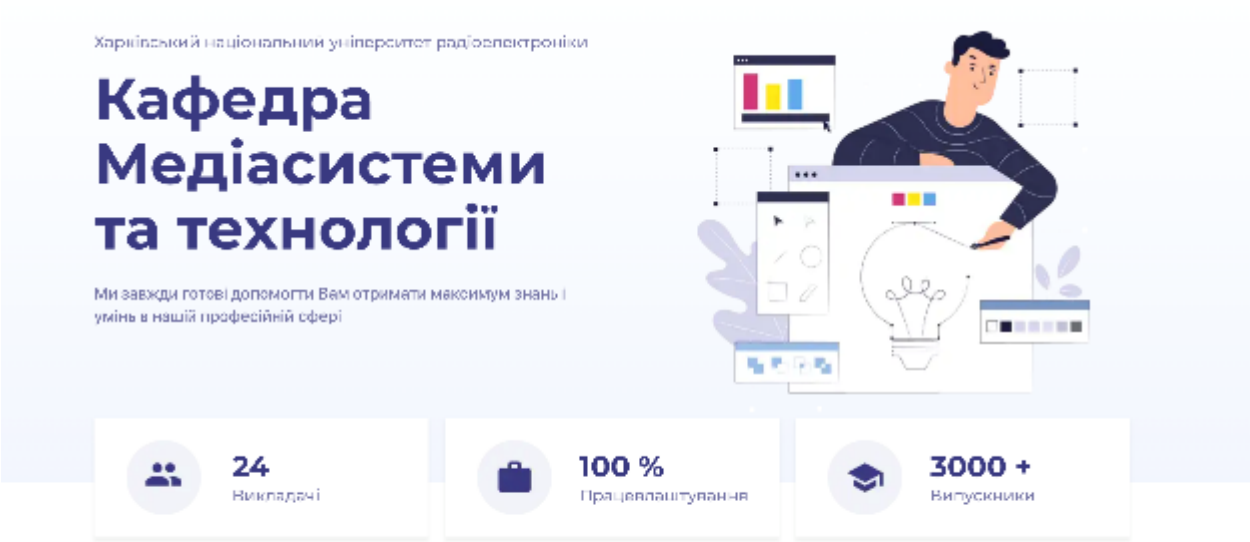


Рисунок 6.14 – Сайт українською мовою

Рисунок 6.15 – Сайт англійською мовою

7 РОЗГОРТАННЯ САЙТУ

Щоб розвернути сайт для початку ми створили репозиторій і розмістили наш код на сторінці “Github” (рис. 7.1).

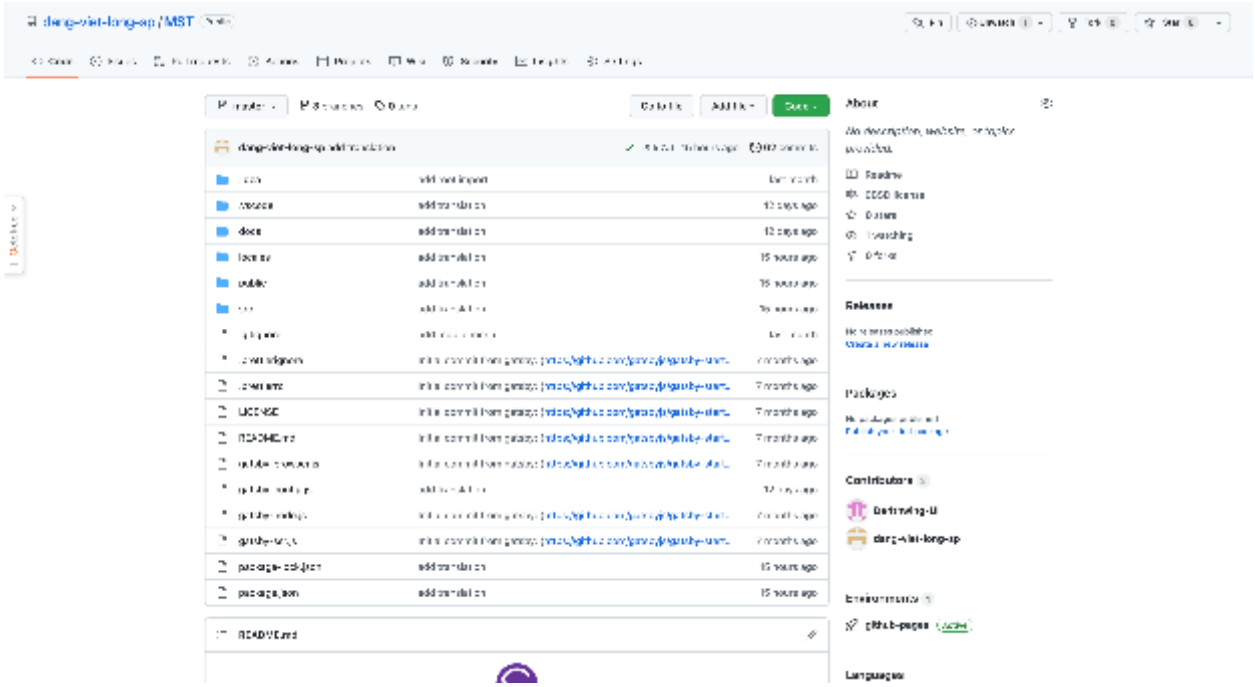


Рисунок 7.1 – Репозиторій Github

За допомогою команд:

– git add --all – добавляли всі змінені файли в комміт;

– git commit -m – добавляли назву комміту;

– git push origin – відправляли комміт у необхідну гілку.

Далі використовуємо “Gatsby Cloud”, щоб розгорнути сайт. Для цього ми за допомогою github авторицції на офіційній сторінці Gatsby Cloud, потім обираємо необхідний репозиторій та гілку для розгортування (рис 7.2).

Тепер Gatsby Cloud на кожний комміт ініціалізує та запускає наш сайт і створює для нього окреме посилання, переходячи на який ми побачимо зміни комміта. (рис. 7.3). Як Зміни коміта перенеслись до ро розгоратного сайту показано на рисунку 7.4.

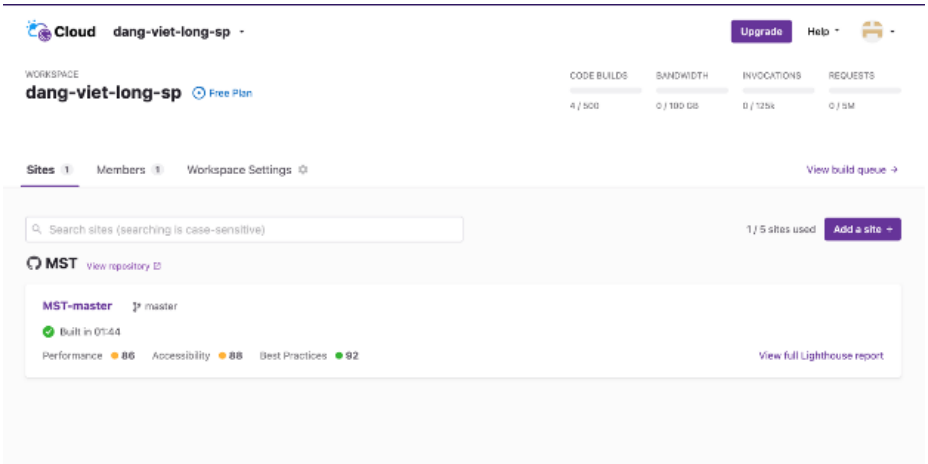


Рисунок 7.2 – Репозиторій Gatsby Cloud

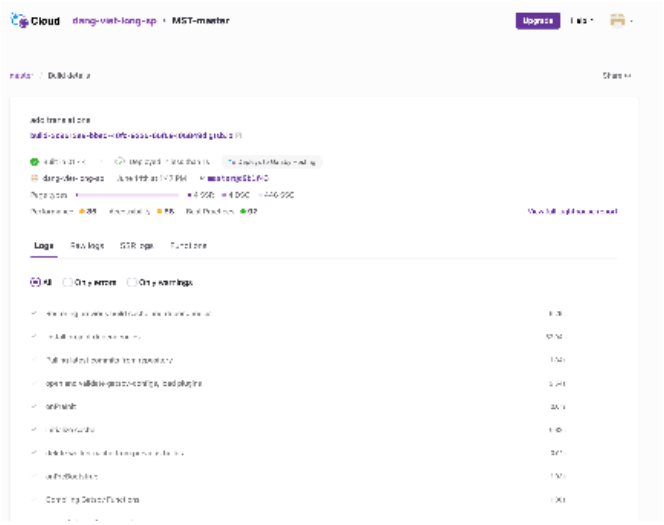


Рисунок 7.3 – Зміни коміта

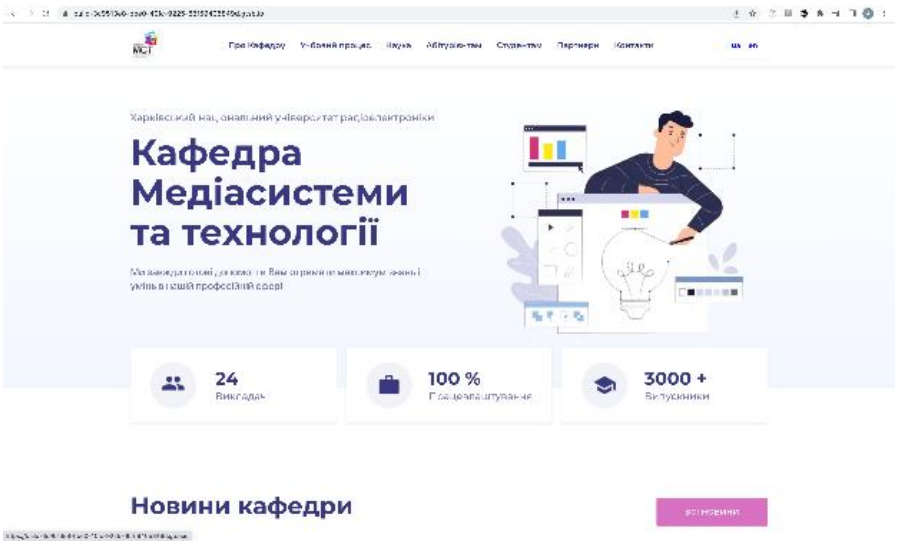


Рисунок 7.4 – Зміни коміта

8 ЕКОНОМІЧНА ЧАСТИНА

У результаті виконання кваліфікаційної роботи було розроблено динамічні елементи для сайту кафедри МСТ. Впровадження до сайту динамічних елементів є закономірним і діючим кроком для покращення роботи сайту.

Економічна ефективність проєкту розраховується перед проєктуванням і розробкою дизайну елементів, у результаті чого можливо спрогнозувати потенційний ефект і доцільність впровадження даного сайту. Спочатку розраховується собівартість розробки, потім визначається ціна.

Розглянемо джерела економії, доходу, джерела фінансування.

Для розробника джерелом доходу є навчання, установка й вдосконалення даного веб-сайту. Витрати фірми містять витрати на розробку дизайну для динамічних елементів. Джерелом фінансування є власні кошти фірми-розробника.

Здійснимо розрахунок собівартості і ціни розробки веб-сайту.

У собівартість розробки веб-сайту входять наступні статті витрат:

– основна заробітна плата;
– додаткова заробітна плата;
– єдиний соціальний внесок;
– інші витрати.

Розробку проводять два фахівці: дизайнер інтерфейсу, фронтенд-розробник, який розробляє динамічні елементі. Зарплата дизайнера інтерфейса становить 80 грн./год, фронтенд-розробника – 100 грн./год,. При цьому тривалість робочого дня кожного з них становить 8 годин. Робота над створенням динамічних елементів тривало 10 днів.

Розрахунок основної заробітної плати наведено в таблиці 9.1.

Додаткова заробітна плата – це винагорода за працю понад установлені норми, за трудові успіхи та винахідливість і за особливі умови праці.

Додаткова заробітна плата становить 20 % від основної:

$$5850 * 0,2 = 1170 \text{ грн.}$$

Таблиця 9.1– Розрахунок витрат на заробітну плату

Етап	Вид робіт	Виконавець		Годинна ставка	Тривалість виконання, дні	Заробітна плата, грн.
		Кількість	Посада			
1. Початковий	Формулювання вимог до динамічних елементів	1	Фронтенд-розробник	100	1	800
2. Графічна частина	Розробка дизайну інтерфейса для елементів	1	Дизайнер інтерфейсу	80	3	1050
3. Розробка й кодування компонентів	Розробка кожного компонента інтерфейсу	1	Фронтенд-розробник	100	2	1600
4. Основний етап	Тестування компонентів	1	Фронтенд-розробник	100	1	800
	Комплексне тестування взаємодії користувача з елементами	1	Фронтенд-розробник	100	1	800
	Оформлення програмної документації	1	Фронтенд-розробник	100	1	800
5. Заключний етап	Корекція програмної документації	1	Фронтенд-розробник	100	1	800
Разом					10	5850
Додаткова заробітна плата (20 %)						1170
Усього						7020

Ставка єдиного соціального внеску становить 22 % від величини основної і додаткової заробітної плати:

$$7020 * 0,22 = 1544,4 \text{ грн.}$$

До інших витрат слід віднести витрати на обслуговування ЕОМ і плату за електроенергію.

Витрати на електроенергію розраховуються виходячи зі споживаної потужності устрою і тарифу на електроенергію. У даному випадку передбачається використання 1 комп'ютеру з потужністю 0,8 кВт/год. Вартість однієї кВт/год електроенергії прийнято у розмірі 1,68 грн. Час використання електроенергії в процесі розробки:

$$8 * 8 = 64 \text{ години.}$$

Отже, плата за електроенергію при використані комп'ютера складе:

$$0,8 * 1,68 * 64 * 2 = 172,032 \text{ грн.}$$

Витрати на обслуговування ЕОМ визначаються з вартості ЕОМ і часу її експлуатації, після закінчення якого, вона підлягає заміні (звичайно цей час не перевищує 3-х років), протягом року ЕОМ використовує 254 робочих дні.

$$(4200 / (3 * 8 * 254)) * 64 = 44,09 \text{ грн.}$$

Проєкт впроваджується на 1 сервер, тому собівартість розробки:

$$8780,48 /1=8780,48 \text{ грн.}$$

Розрахуємо суму прибутку від реалізації розробки (виходячи з рівня рентабельності 30 %):

$$8780,48 * 0,3 = 2634,14 \text{ грн.}$$

Розрахуємо ціну розробки сайту без податку на додану вартість (ПДВ):

8780,48+ 2634,14 = 11 414,62 грн.

Розрахуємо суму ПДВ, вона рівна 20% від ціни:

11 414,62 * 0,2 = 2282,92 грн.

Розрахуємо ціну сайту з урахуванням ПДВ:

11 414,62 + 2282,92 = 13 697,54 грн.

Результати розрахунків наведено у таблиці 9.2.

Таблиця 9.2 – Розрахунок витрат на розробку та ціни веб-сайту

Стаття витрат	Сума, грн.
Основна заробітна плата	5850
Додаткова заробітна плата	1170
Єдиний соціальний внесок	1544,4
Витрати на обслуговування ЕОМ	44,09
Витрати на електроенергію	172,42
Собівартість розробки сайту	8780,48
Прибуток (рівень рентабельності 20 %)	2634,14
Ціна без ПДВ	11 414,62
Податок на додану вартість (ПДВ)	2282,92
Ціна з урахуванням ПДВ	13 697,54

Таким чином, виходячи з виконаних розрахунків повна вартість розробки сайту складе 13 697,54 грн. Термін виконання всіх етапів розробки становить 10 днів для команди з одного дизайнера інтерфейсу і фронт-енд-розробника. Очікувана сума прибутку складе 2634,14 грн.

ВИСНОВКИ

У роботі розглянута технологія розробки динамічних елементів веб-сайту, розгортання веб-сайту для кафедри МСТ. Обрано та обгрунтовано необхідне програмне забезпечення; Також виконано економічне обґрунтування проекту.

У ході виконання кваліфікаційної роботи було виконано наступні поставлені задачі:

– проаналізували визначену предметну область;
– обрали найкращий варіант елементів для сайту,
– реалізували елементи за допомогою коду;
– розробили функціонал роботи елементів;

Було зроблено економічне обгрунтування роботи. Таким чином виходячи з виконаних розрахунків повна вартість розробки сайту складе 13 697,54 грн. Термін виконання всіх етапів розробки становить 10 днів для команди з одного дизайнера інтерфейсу і фронт-енд-розробника. Очікувана сума прибутку складе 2634,14 грн.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тиленс Т.М. React в дії. Київ, 2018. 368с.

2. Єгорова І.М. Проектування та розробка Web-документів: навч. посібник. Харків: ХНУРЄ, 2018. 264с.

3. Iegorova I. Trends for modern WEB- development // Поліграфічні, мультимедійні та web-технології: тези доп. V Міжнар. наук.-техн. конф., Україна, м.Київ, 3-6 листоп. 2020. Т. 1. С. 75-77.

4. Єгорова І.М., Коміна М.М. Розробка методики ефективного застосування анімації у WEB // Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. 2020. № 4 (6). С. 60-64.

5. 9. Єгорова І.М., Антипенко К.Д. Про застосування кривих Без'є для покращення CSS анімації // Системи обробки інформації. 2019. Випуск 2 (157). С. 40-44.

6. 14. Егорова И.Н., Кочура Л.А. Розробка навчальної гри «Web-технології» // Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. 2019. № 2. С. 49-53.

7. 11. Єгорова І.М., Самокіш В.В. Про використання спрайтової анімації у веб-виданнях // Системи обробки інформації. 2018. Випуск 3 (154). С. 100-104.

8. 12. Єгорова І.М., Гладка А.А. Дослідження можливостей резервного копіювання веб-сайтів, створених на основі WordPress // Вісник Національного технічного університету «ХПІ». Серія: Нові рішення у сучасних технологіях. 2017. № 23 (1245). С. 95-99.

9. Єгорова І.М., Худолій А.Ю. Дослідження можливостей компонентного підходу розробки веб-сайтів // Системи обробки інформації. 2017. Випуск 4 (150). С. 76-78.

10. Єгорова І.М., Горєлова Р.А. Розробка методики пошукової оптимізації веб-сайтів // Системи обробки інформації. 2017. Випуск 4 (150). С. 73-75.

11. Єгорова І.М., Кадушкевич О.М. Методика ефективного використання інструментів Google Analytics // ScienceRise. 2016. №1/2 (18). С. 40-44.

12. Єгорова І.М., Філіпенко О.В. Розробка методики створення графічного інтерфейсу веб-сайтів // ScienceRise. 2016. № 1/2 (18). С. 58-61.

13. Єгорова І.М. Web-технології: методичні вказівки до курсового проекту. Харків: ХНУРЕ, 2019. 24 с.