Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

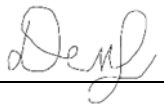
Кафедра Медіасистеми та технології
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Дослідження інструментальних засобів адаптивної верстки сайтів
(тема)

Виконав:
здобувач 2 курсу
групи КТСВПВм-23-1


Батраченко Д.А.

Спеціальності 186 Видавництво та поліграфія

Тип програми Освітньо-професійна

Освітня програма
Комп'ютерні технології та системи
видавничо-поліграфічних виробництв

Керівник 
проф.Ткаченко В.П.

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Медіасистеми та технології
Рівень вищої освіти другий (магістерський)
Спеціальність 186 Видавництво та поліграфія
Тип програми Освітньо-професійна
Освітня програма Комп'ютерні технології
та системи видавничо-поліграфічних виробництв
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри МСТ_____
(підпис)
« 18 » листопада 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Батраченко Денису Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження інструментальних засобів адаптивної верстки сайтів

затверджена наказом по університету від 8 листопада 2024 р. № 1188 Ст

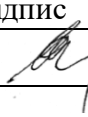
2. Термін подання студентом роботи до екзаменаційної комісії 20 січня 2025 р.

3. Вихідні дані до роботи
Національні та міжнародні стандарти оцінки якості веб-сайтів; ДСТУ Інженерія систем і програмних засобів. Розроблення та керування WEB-сайтами для систем, програмних засобів та інформаційних послуг

4. Перелік питань, що потрібно опрацювати в роботі
Постановка задачі дослідження; Огляд літератури за темою дослідження; Планування дослідження; Експериментальна частина; Оцінка якості; Економічна частина; Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій
Тема роботи; Актуальність теми; Мета роботи; Завдання кваліфікаційної роботи; Програмне забезпечення; Аналіз завдання і огляд літератури за темою; Практична частина; Економічна частина; Висновки.

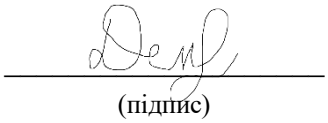
6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Ткаченко В. П.		19.01.25
Економічна частина	ас. Помогалова Н.В.		18.01.25

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі дослідження	01.12.2024	виконано
2	Огляд літератури за темою дослідження	05.12.2024	виконано
3	Планування дослідження	10.12.2024	виконано
4	Експериментальна частина	12.12.2024	виконано
5	Оцінка якості	18.12.2024	виконано
6	Економічна частина	24.12.2024	виконано
7	Оформлення пояснювальної записки	03.01.2025	виконано
8	Оформлення графічної частини	03.01.2025	виконано

Дата видачі завдання 18 листопада 2024 р.

Здобувач  Батраченко Д. А.

(підпис)

Керівник роботи  проф. Ткаченко В. П.

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 78 с., 5 табл., 14 рис., 1 дод., 13 джерел.

WEB-САЙТ, АДАПТИВНА ВЕРСТКА, МЕДІА-ЗАПИТИ, CSS ФРЕЙМВОРКИ, FLEXBOX, GRID LAYOUT, JAVASCRIPT, BOOTSTRAP, WEB-ДИЗАЙН, МОБІЛЬНА ОПТИМІЗАЦІЯ, ВЕБ-ТЕХНОЛОГІЇ, ВЕБ-СТАНДАРТИ, ДИЗАЙН.

Мета роботи полягає в дослідженні інструментальних засобів адаптивної верстки сайтів, а також у вивченні найбільш ефективних методів та технологій для створення веб-сайтів, які коректно відображаються на різних пристроях та екранах.

Для досягнення мети роботи були проведені кілька етапів дослідження. Спочатку здійснено аналіз теоретичних аспектів адаптивної верстки, визначено основні принципи та переваги використання адаптивних та респонсивних підходів до створення веб-сайтів. Далі вивчено сучасні інструменти та технології для реалізації адаптивного дизайну, серед яких CSS-фреймворки, такі як Bootstrap, а також методи використання медіа-запитів і CSS Grid.

Основним результатом роботи є розробка рекомендацій для вибору інструментальних засобів адаптивної верстки, а також створення набору практичних кроків для реалізації адаптивного дизайну сайту. Останній етап включає тестування сайту на різних пристроях, що дозволяє оцінити ефективність застосованих технологій і внести необхідні корективи.

Робота також містить економічну оцінку ефективності використання адаптивної верстки в контексті сучасних веб-проектів, що дозволяє визначити доцільність застосування таких технологій для конкретних завдань.

ABSTRACT

Explanatory note of the qualification work: 78 p., 10 tabl., 20 fig., 2 app., 20 sources.

WEBSITE, ADAPTIVE LAYOUT, MEDIA QUERIES, CSS FRAMEWORKS, FLEXBOX, GRID LAYOUT, JAVASCRIPT, BOOTSTRAP, WEB DESIGN, MOBILE OPTIMIZATION, WEB TECHNOLOGIES, WEB STANDARDS, DESIGN.

The aim of the work is to research the instrumental tools for adaptive layout of websites and to study the most effective methods and technologies for creating websites that display correctly on various devices and screens.

To achieve the aim of the work, several stages of research were conducted. Initially, a theoretical analysis of adaptive layout was performed, the main principles and advantages of using adaptive and responsive approaches to web design were identified. Next, modern tools and technologies for implementing adaptive design were studied, including CSS frameworks such as Bootstrap, as well as methods for using media queries and CSS Grid.

The main result of the work is the development of recommendations for selecting instrumental tools for adaptive layout, as well as the creation of a set of practical steps for implementing adaptive website design. The final stage includes testing the website on different devices, which allows evaluating the effectiveness of the applied technologies and making necessary adjustments.

The work also includes an economic assessment of the effectiveness of using adaptive layout in the context of modern web projects, which allows determining the feasibility of applying such technologies for specific tasks.

ЗМІСТ

С.

ВСТУП

Швидкий розвиток Інтернету та цифрових технологій суттєво вплинув на потреби користувачів у доступі до інформації на різних пристроях. Адаптивна верстка, що дозволяє сайту автоматично підлаштовуватися під розміри екрану пристрою, на якому він переглядається, стала необхідністю в умовах зростаючої різноманітності гаджетів, включаючи смартфони, планшети, ноутбуки та інші. Технологія адаптивної верстки базується на використанні таких підходів, як медіа-запити та відносні одиниці виміру, і відкриває нові можливості для підвищення зручності користування веб-сайтами.

Метою цього дослідження є аналіз та обґрунтування інструментів програмного забезпечення, які застосовуються для адаптивної верстки сайтів, а також розробка методики оцінки їх ефективності та зручності у застосуванні.

Актуальність теми дослідження полягає у зростаючій необхідності адаптивної верстки як основи для забезпечення якісного користувацького досвіду в умовах розширення спектру пристроїв для перегляду веб-контенту. Адаптивні сайти дозволяють компаніям та організаціям задовольняти очікування сучасних користувачів, підвищуючи їх залученість та доступність контенту на будь-якому пристрої.

Об'єктом дослідження є інструментальні технології розробки адаптивної верстки сайтів, що використовуються для забезпечення адаптивності та зручності користувацького інтерфейсу.

Предметом дослідження є процеси та технології, специфічні для адаптивної верстки, зокрема медіа-запити, фреймворки для адаптивного дизайну та інші інструменти, що допомагають оптимізувати взаємодію з сайтом.

Для досягнення поставленої мети слід вирішити такі завдання:

– провести аналіз існуючих інструментів для створення адаптивних інтерфейсів;

– дослідити особливості застосування цих інструментів при розробці адаптивних сайтів;

– визначити критерії оцінки ефективності інструментів адаптивної верстки;

– провести порівняння популярних технологій з точки зору зручності та адаптивності;

– розробити методику оцінки адаптивності сайту та її практичну реалізацію на рівні прототипу;

– виконати оцінку ефективності реалізованих рішень.

Теоретична значущість дослідження полягає у зборі та систематизації інформації, що сприяє більш якісному впровадженню адаптивної верстки в розробку веб-сайтів.

Практична значущість дослідження полягає у застосуванні отриманих результатів для покращення процесів розробки адаптивних сайтів, що можуть бути корисними як для фахівців у сфері ІТ, так і для зацікавлених осіб.

Для реалізації дослідження використовувались такі методи, як аналіз літературних джерел, систематизація матеріалу, порівняння та узагальнення, проектування, а також практична розробка і тестування. Інформаційну базу становлять літературні джерела, онлайн-ресурси та професійні довідники у сфері веб-розробки та дизайну.

Результати дослідження можуть бути представлені у вигляді публікацій та інтегровані в процес розробки адаптивних веб-сайтів.

1 ОГЛЯД ЛІТЕРАТУРИ ЗА ТЕМОЮ ДОСЛІДЖЕННЯ

1.1 Порівняльний огляд

Адаптивна верстка стала одним із ключових підходів у сучасній веб-розробці, оскільки вона дозволяє сайту автоматично підлаштовуватися під екран пристрою користувача, будь-то смартфон, планшет чи настільний комп’ютер. Завдяки цьому підходу підвищується зручність використання веб-ресурсів та покращується доступність інформації, що зробило адаптивну верстку важливим аспектом цифрової економіки.

Адаптивна верстка реалізується з використанням декількох ключових технологій: HTML5, CSS3 (включаючи медіа-запити), та JavaScript. HTML5 дозволяє структурувати контент, а CSS3 забезпечує налаштування стилів, включаючи адаптивність за допомогою медіа-запитів, що регулюють стилі в залежності від розмірів екрану. JavaScript використовується для додаткової інтерактивності, зокрема у випадках, коли необхідно змінювати поведінку елементів на сторінці в залежності від розміру пристрою.

Одним із фундаментальних інструментів адаптивної верстки є медіа-запити, які дозволяють налаштовувати CSS-стилі для різних розмірів екранів. Розширення можливостей CSS за допомогою медіа-запитів обумовлює підхід до адаптації контенту без створення окремих версій сайту, що є ефективним рішенням для досягнення адаптивності.

В останні роки активно розвиваються різні інструменти та фреймворки, які значно полегшують процес створення адаптивних сайтів. Найбільш популярними серед них є Bootstrap, Foundation, Bulma, та Tailwind CSS:

– Bootstrap є одним із найпопулярніших фреймворків для адаптивної верстки. Він пропонує готові CSS класи та компоненти, що значно прискорює розробку та робить її більш гнучкою;

– Foundation має схожий з Bootstrap підхід, але орієнтований на більш індивідуальні рішення, надаючи велику свободу налаштувань для дизайна;

– Bulma є сучасним CSS-фреймворком, що забезпечує простоту і швидкість розробки, надаючи широкий спектр адаптивних елементів;

– Tailwind CSS дозволяє створювати унікальні дизайни, не використовуючи заздалегідь визначених стилів компонентів, що робить його більш гнучким і персоналізованим.

Одним із найбільших викликів у процесі розробки адаптивного дизайну є підтримка сумісності з усіма пристроями та браузерами, а також оптимізація продуктивності. Крім того, вимагається правильна робота з медіа-контентом, як-то зображеннями та відео, які повинні динамічно підлаштовуватися під розмір екрану. Ще одним важливим аспектом є забезпечення доступності, яка передбачає можливість використання веб-сайту людьми з обмеженими можливостями.

Сучасні дослідження підкреслюють, що адаптивна верстка є важливим фактором для покращення SEO показників. Оптимізація сайту для мобільних пристроїв, яку забезпечує адаптивний дизайн, дозволяє покращити ранжування у пошукових системах, таких як Google, що робить сайт більш доступним для користувачів.

Цей огляд літератури підсумовує основні принципи адаптивної верстки та огляд популярних інструментів, які допомагають створювати сайти з адаптивним дизайном. Адаптивна верстка є важливим аспектом сучасного веб-дизайну, що дозволяє сайтам коректно відображатися на різних пристроях з урахуванням розміру екрану, особливостей браузера і платформи. Вона забезпечує зручність для користувачів, оскільки адаптує контент до будь-якого середовища, від мобільних телефонів до широкоформатних моніторів. Основними технологіями для реалізації адаптивної верстки є медіа-запити CSS, гнучкі сітки, а також відносні одиниці вимірювання, такі як відсотки та em. Кожен тип верстки має свої переваги та недоліки, що визначають зручність використання, адаптивність і продуктивність сайту. Зокрема, адаптивна верстка значно покращує користувацький досвід, але потребує більше зусиль для створення та тестування різних варіантів дизайну.

1.1.1 Фіксована верстка

Фіксована верстка – це підхід, при якому ширина сторінки встановлюється статично, без урахування розміру екрана пристрою. Контент в такій верстці прив'язаний до фіксованих значень пікселів, що дозволяє точніше контролювати вигляд і розташування елементів.

Розробник сайту встановлює певну ширину сторінки, яка не змінюється в залежності від пристрою. При цьому прокручування здійснюватиметься не тільки у вертикальній, а й у горизонтальній площині. Такий формат верстки морально застарів та практично не використовується на сучасних сайтах. Дуже незручний для мобільних пристроїв.

Переваги:

– простота в реалізації та налагодженні;

– ідеально підходить для сайтів, де важлива точність у розташуванні елементів.

Недоліки:

– погана адаптація під мобільні пристрої;

– викликає горизонтальну прокрутку, що створює дискомфорт для користувачів на пристроях з різною роздільною здатністю.

Фіксована верстка поступово вийшла з ужитку через зростання кількості мобільних користувачів. Вона використовується здебільшого для адміністративних панелей або систем, де адаптивність не є пріоритетною вимогою.

1.1.2 Резинова верстка

Резинова верстка передбачає встановлення мінімального і максимального значень ширини сторінки, що дозволяє контенту змінюватися пропорційно до розміру вікна браузера. Замість фіксованої ширини використовуються відсоткові значення, що дозволяє макету стягуватися або розтягуватися в певних межах. На відміну від фіксованої макет має більшу

гнучкість. Замість стандартного параметра ширини відбувається визначення допустимого максимального та мінімального значення ширини екрана з адаптацією контенту.

Переваги:

– більш гнучка, ніж фіксована верстка, і краще пристосовується до різних розмірів екранів;

– дає більше можливостей для створення адаптивного контенту без використання медіа-запитів.

Недоліки:

– недостатня адаптація для дуже великих або дуже маленьких екранів, що може призводити до втрати якості відображення;

– може потребувати додаткових налаштувань для забезпечення оптимального вигляду.

Резинову верстку використовують для сайтів, де потрібна базова адаптивність без складних медіа-запитів або складного коду, як-от блоги чи новинні ресурси з простими макетами.

1.1.3 Адаптивна (відзивчива) верстка

Адаптивна верстка, також відома як відзивчива (responsive), використовує медіа-запити CSS для зміни макету сторінки в залежності від розміру екрану пристрою. Це дозволяє керувати не тільки шириною, але й положенням елементів, розміром шрифтів, відступами тощо.

Тип верстки, що найбільш використовується, який дозволяє сайту відображати контент, виходячи з параметрів пристрою. З її допомогою можна змінювати не лише ширину сторінки, а й інші параметри розмір шрифту, відступи, розташування. В основі верстки – робота з медіа-запитами.

Переваги:

– оптимально підходить для мобільних пристроїв, оскільки дозволяє створити окремі стилі для кожного типу екрану;

– використання медіа-запитів забезпечує високу гнучкість і контроль над виглядом сторінки;

– підтримка різних роздільних здатностей дозволяє коректно відображати контент на пристроях з високою роздільною здатністю.

Недоліки:

– може бути складнішою у розробці, оскільки потребує продуманого налаштування медіа-запитів для кожного типу пристрою;

– вимагає ретельного тестування на різних екранах і пристроях, що може ускладнити процес розробки.

Цей підхід став стандартом для сучасних сайтів, оскільки дозволяє оптимально адаптувати сайт під будь-який пристрій. Адаптивну верстку використовують як для персональних, так і для комерційних сайтів, інтернет-магазинів та соціальних мереж, де важливий позитивний досвід користувача на різних пристроях.

1.1.4 Кросбраузерна верстка

Кросбраузерна верстка спрямована на забезпечення коректного відображення контенту незалежно від того, який браузер використовує користувач. Це досягається за допомогою оптимізації коду, щоб різні браузери (Chrome, Safari, Firefox, Edge тощо) однаково інтерпретували HTML, CSS та JavaScript.

Різновид верстки, який дозволяє сайту коректно відображатися в різних браузерах. Іншими словами, незалежно від вибраного браузера сторінки збережуть свою функціональність та зручність використання.

Переваги:

– забезпечує зручний користувацький досвід незалежно від браузера;

– дозволяє уникнути помилок відображення, спричинених відмінностями в обробці коду різними браузерами.

Недоліки:

– може вимагати додаткових налаштувань і налаштування спеціальних стилів для старих браузерів;

– висока складність підтримки на різних версіях браузерів, особливо старих або малопоширених.

Кросбраузерна верстка є важливою для будь-якого сайту, що націлений на широку аудиторію, оскільки користувачі можуть відвідувати його з різних браузерів. Вона є стандартом і часто поєднується з іншими видами верстки, такими як адаптивна або резинова, для створення повністю оптимізованого та кросплатформного ресурсу.

1.1.5 Чуйно-адаптивна верстка

Чуйно-адаптивна верстка (Responsive-Adaptive Layout) поєднує принципи чуйного (responsive) та адаптивного (adaptive) підходів до створення сайтів, щоб забезпечити максимальну гнучкість у відображенні контенту на різних пристроях. Цей підхід будується на використанні CSS медіа-запитів, але також адаптує контент та функціональні елементи сайту, ґрунтуючись на детальному аналізі пристрою його роздільній здатності, розмірі екрану, щільності пікселів і навіть поведінкових характеристик.

Чуйно-адаптивна верстка використовує одночасно адаптивні макети та чуйні елементи. Зокрема:

– медіа-запити та відносні одиниці дозволяють змінювати стиль контенту в залежності від екрану. Наприклад, на мобільних пристроях меню може стати гамбургером, щоб не захаращувати простір, тоді як на великому екрані воно буде розгорнуто горизонтально;

– гнучкість сітки та елементів, контент розтягується або стискається, підтримуючи пропорції, щоб завжди залишатися зручним для читання та функціональним на різних пристроях;

– адаптація зображень та медіа, зображення та відео підлаштовуються під екран, наприклад, за рахунок зміни роздільної здатності та щільності пікселів, щоб мінімізувати завантаження на мобільних пристроях;

– користувацькі переваги, чуйно-адаптивна верстка може включати функції, такі як збереження даних або зменшення графічних елементів для заощадження трафіку.

Переваги:

– зручність для користувача, контент сайту відображається коректно та зручно на всіх типах пристроїв, від мобільних телефонів до великих моніторів;

– SEO-оптимізація, пошукові системи, такі як Google, надають пріоритет сайтам, оптимізованим під мобільні пристрої, що покращує ранжування;

– економія ресурсів на розробці, створюється один сайт, який коректно працюватиме на всіх пристроях, без необхідності розробляти окремі версії;

– підвищення продуктивності та адаптивності, чуйно-адаптивна верстка дозволяє скоротити час завантаження, адаптуючи якість медіа та мінімізуючи обсяг даних для мобільних пристроїв.

Недоліки:

– складність розробки та тестування, вимагає професійного підходу та детального налаштування для різних дозволів та пристроїв;

– необхідність враховувати старі версії браузерів - деякі браузери (особливо старі версії) можуть некоректно відображати складні адаптивні елементи;

– може збільшувати розмір коду через велику кількість медіа-запитів і додаткових стилів код може стати об'ємним, що позначається на швидкості завантаження сторінок.

Чуйно-адаптивна верстка є стандартом для сучасних веб-сайтів, особливо для тих, де важлива взаємодія з користувачем та постійний доступ з різних пристроїв. Вона ідеально підходить для інтернет-магазинів, сайтів новин, блогів і корпоративних порталів, де потрібно забезпечити комфортний доступ для широкої аудиторії.

1.2 Порівняння підходів до адаптивної верстки

Розглянемо порівняння різних підходів до адаптивної верстки, враховуючи їх сильні сторони, обмеження та сфери застосування.

Фіксована верстка – це найпростіший, але водночас застарілий метод. Вона характеризується статичною шириною сторінок, яка не адаптується під розмір екрана пристрою, через що сайти з фіксованою версткою виглядають зручно лише на тих екранах, для яких вони створювалися. На великих екранах контент може виглядати обмеженим, а на маленьких – викликати горизонтальну прокрутку.

Основна перевага фіксованої верстки – це простота реалізації, яка робить її вигідною для простих сайтів або внутрішніх систем, де адаптація під різні пристрої не є пріоритетом. Проте в умовах сучасних вимог до адаптивності фіксована верстка майже повністю вийшла з ужитку.

Резинова верстка є більш гнучкою, оскільки дозволяє контенту розтягуватися або стягуватися в межах певного діапазону ширини екрана, використовуючи відсоткові значення замість фіксованих пікселів. Це робить резинову верстку більш універсальною, ніж фіксовану, і зручнішою для перегляду на екранах різних розмірів. Проте вона теж має певні обмеження, надто великі екрани можуть розтягувати контент надмірно, а надто маленькі – робити його незручним для перегляду. Резинова верстка підходить для сайтів із простим контентом або блогів, де не потрібна точна адаптація під кожен пристрій.

Адаптивна верстка значно більш універсальна і є стандартом у сучасному веб-дизайні. Вона базується на використанні медіа-запитів, що дозволяють змінювати стилі в залежності від ширини або інших параметрів екрана. Це дозволяє не тільки змінювати ширину, а й налаштовувати розміри шрифтів, відступів і навіть порядок елементів, підлаштовуючи сторінку під різні пристрої.

На відміну від резинової верстки, адаптивна верстка забезпечує більше контролю над виглядом сторінки на різних екранах, оскільки може точно адаптуватися до кожного з них.

Основні недоліки адаптивної верстки полягають у більшій складності розробки, а також потребі ретельного тестування, проте ці витрати виправдовуються високою зручністю користувацького інтерфейсу.

Кросбраузерна верстка, на відміну від інших підходів, орієнтована на забезпечення коректного відображення сторінки в різних браузерах, що особливо актуально для підтримки широкої аудиторії. Кросбраузерна верстка не замінює адаптивні підходи, а є важливою складовою, яку комбінують з ними. Вона забезпечує, щоб адаптивні сайти зберігали свою функціональність та зручність, незалежно від браузера, на якому вони переглядаються. Для цього потрібно враховувати відмінності в обробці HTML, CSS і JavaScript у різних браузерах.

Це може збільшити складність розробки та вимагає додаткового тестування, проте є необхідним для сайтів, орієнтованих на широку аудиторію.

Відзивчиво-адаптивна верстка об’єднує принципи адаптивної та відзивчої верстки, щоб забезпечити максимальну гнучкість і зручність на всіх типах пристроїв. Використовуючи медіа-запити, вона налаштовує не тільки ширину та розташування елементів, але також змінює функціональність в залежності від типу пристрою (наприклад, спрощує навігацію на мобільних пристроях чи забезпечує швидку завантаження на повільних з’єднаннях).

Такий підхід дає більше можливостей для оптимізації користувацького досвіду, дозволяючи сайту пристосовуватися не тільки до розміру екрану, але й до технічних характеристик пристрою, наприклад, до швидкості з'єднання або енергозбереження.

Недоліком відзивчиво-адаптивної верстки є її складність, адже вона вимагає ретельного підходу до дизайну і глибоких знань медіа-запитів, CSS і JavaScript.

1.3 Висновки з розділу

Фіксована та резинова верстка сьогодні поступаються адаптивній та відзивчо-адаптивній верстці, які забезпечують повноцінний користувацький досвід. Адаптивна верстка є базовим стандартом, а відзивчиво-адаптивна надає більше можливостей для налаштування під специфічні вимоги користувача та пристрою. Кросбраузерна сумісність залишається важливим компонентом будь-якого сучасного веб-проекту, адже вона дозволяє коректно працювати сайту незалежно від середовища браузера.

2 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

2.1 Обґрунтування доцільності дослідження та визначення ключових завдань

Адаптивна верстка сайтів стала невід’ємною частиною сучасного веб-дизайну, оскільки користувачі все частіше звертаються до Інтернету через різні пристрої, такі як смартфони, планшети та настільні комп'ютери. Зростання використання мобільних пристроїв і різноманіття розмірів екранів вимагає від веб-розробників впровадження технологій, які забезпечують оптимальне відображення контенту на різних платформах.

Актуальність дослідження полягає в необхідності розвитку та вдосконалення інструментального програмного забезпечення, що підтримує адаптивну верстку. Сьогодні існує безліч фреймворків і бібліотек, таких як Bootstrap, Foundation, а також CSS Grid і Flexbox, проте вибір оптимального рішення та його адаптація під специфічні вимоги проекту залишаються складними завданнями для розробників.

Ключові завдання дослідження включають:

– аналіз сучасних інструментів для адаптивної верстки, їх переваг та недоліків. Це дозволить визначити найбільш ефективні рішення для різних сценаріїв використання;

– вивчення принципів адаптивного дизайну та його впливу на користувацький досвід. Необхідно виявити, як різні підходи до адаптивної верстки можуть впливати на задоволеність користувачів і їхню взаємодію з сайтом;

– розробка рекомендацій щодо вибору інструментів для адаптивної верстки на основі проведеного аналізу. Це допоможе веб-розробникам приймати обґрунтовані рішення при виборі технологій;

– оцінка ефективності застосування різних методів адаптивної верстки в реальних проектах. Вивчення випадків успішних реалізацій дозволить виділити найкращі практики та визначити можливі труднощі в процесі впровадження.

Таким чином, дослідження спрямоване на комплексний підхід до аналізу інструментального програмного забезпечення адаптивної верстки, що забезпечить підвищення якості веб-розробки та покращення користувацького досвіду.

2.2 Визначення критеріїв для оцінювання

Для ефективної оцінки інструментального програмного забезпечення адаптивної верстки сайтів важливо визначити ключові критерії, які допоможуть у виборі найкращого рішення. Нижче наведені основні критерії, що будуть враховані у дослідженні.

1. Функціональність та можливості.

Гнучкість, цей критерій оцінює, наскільки легко інструмент дозволяє налаштовувати елементи верстки. Гнучкість є важливою, оскільки веб-розробники часто стикаються з унікальними вимогами проектів. Інструмент повинен надавати можливість створювати складні макети, адаптувати стилі та компоненти під специфічні потреби без значних обмежень. Це включає підтримку модульного підходу, що дозволяє використовувати реюзабельні компоненти та блоки коду для підвищення ефективності розробки.

2. Адаптивність і кросбраузерність.

Адаптивний дизайн. Цей критерій визначає, чи генерує інструмент адаптивну верстку для різних пристроїв, включаючи мобільні телефони, планшети та настільні комп'ютери. Важливість адаптивного дизайну полягає в тому, що сучасні користувачі мають доступ до Інтернету через різні платформи, і сайти повинні виглядати і функціонувати належним чином на всіх з них. Інструмент повинен автоматично підлаштовувати елементи верстки під різні розміри екрану.

Кросбраузерність. Цей аспект оцінює, чи підтримує інструмент коректне відображення сайту в усіх сучасних браузерах (Chrome, Firefox, Safari тощо). Неправильне відображення в різних браузерах може негативно вплинути на користувацький досвід і призвести до втрати відвідувачів.

3. Швидкість та продуктивність.

Оптимізація швидкості. Цей критерій важливий для оцінки, наскільки швидко завантажуються сторінки, створені за допомогою інструмента. Швидкість завантаження безпосередньо впливає на користувацький досвід: чим швидше завантажується сайт, тим більше шансів, що користувач залишиться на ньому. Інструмент має генерувати оптимізовані файли, зменшувати обсяг даних, які потрібно завантажити, і використовувати техніки, такі як кешування та стиснення. Швидкість та продуктивність є економічним показником. Він безпосередньо впливає на ефективність витрат на хостинг та обслуговування сайту, оскільки швидкість завантаження сторінок може впливати на користувацький досвід та конверсію, а також на витрати на ресурси. Оптимізація швидкості може знизити витрати на обслуговування та покращити загальні показники проекту.

Ці три критерії забезпечують комплексний підхід до оцінки інструментального програмного забезпечення для адаптивної верстки, допомагаючи визначити його відповідність сучасним вимогам веб-розробки.

2.3 Визначення основної гіпотези дослідження

Отже, основна гіпотеза дослідження полягає в тому, інструментальне програмне забезпечення адаптивної верстки сайтів, яке має високу гнучкість, підтримує адаптивний дизайн, оптимізує швидкість завантаження, суттєво підвищує ефективність веб-розробки і покращує користувацький досвід у порівнянні з менш ефективними інструментами.

Ця гіпотеза передбачає, що ключові характеристики інструментів для адаптивної верстки впливають на загальну продуктивність веб-сайтів і їхню сприйнятливість з боку користувачів. Дослідження прагне підтвердити або спростувати цю гіпотезу шляхом порівняння різних інструментів за визначеними критеріями, що дозволить виявити оптимальні рішення для сучасних веб-проектів.

У даному розділі було визначено основні критерії для оцінювання інструментального програмного забезпечення адаптивної верстки сайтів, що є важливими для сучасної веб-розробки. Зокрема, було обрано три ключових критерія:

– функціональність та можливості, забезпечує гнучкість налаштувань та модульність компонентів, що дозволяє ефективно реалізовувати унікальні вимоги проектів;

– адаптивність і кросбраузерність гарантує, що сайти будуть коректно відображатися на різних пристроях та в усіх сучасних браузерах, що критично важливо для користувачів;

– швидкість та продуктивність, оптимізація швидкості завантаження сторінок впливає на користувацький досвід та може знизити витрати на хостинг, що є важливим економічним аспектом.

На основі цих критеріїв була сформульована основна гіпотеза, яка передбачає, що високоякісні інструменти адаптивної верстки суттєво підвищують ефективність веб-розробки і покращують користувацький досвід. Подальше дослідження має на меті перевірити цю гіпотезу шляхом порівняння різних інструментів та оцінки їхніх характеристик.

Таким чином, визначені критерії та сформульована гіпотеза створюють основу для глибшого аналізу інструментального програмного забезпечення в контексті адаптивної верстки, що дозволить зробити обґрунтовані висновки про їхню ефективність і доцільність використання.

3 ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ АДАПТИВНОЇ ВЕРСТКИ

3.1 CSS медіа-запити: можливості та обмеження

Медіа-запити є одним із ключових інструментів у CSS для створення адаптивного дизайну веб-сторінок. Вони дозволяють змінювати стилі залежно від характеристик пристрою, на якому відкривається сайт. Це включає розміри екрану, орієнтацію, роздільну здатність, кольорову схему та інші параметри.

1. Можливості CSS медіа-запитів.

Медіа-запити дають змогу налаштовувати дизайн, орієнтуючись на особливості пристрою, що використовує користувач. Основні можливості:

– гнучке налаштування стилів. Використання медіа-запитів дозволяє застосовувати специфічні стилі для різних розмірів екранів. Це забезпечує належний вигляд сайту як на мобільних пристроях, так і на настільних комп’ютерах;

– підтримка різних орієнтацій екрана. Медійні запити можуть адаптувати дизайн залежно від портретної або ландшафтної орієнтації;

– оптимізація ресурсів. Можна змінювати не лише стилі, а й завантаження певних ресурсів, наприклад, зображень низької якості для мобільних пристроїв;

– підтримка адаптивних шрифтів та інтерфейсів. Завдяки використанню відносних одиниць (em, rem) у поєднанні з медіа-запитами можна адаптувати текст до різних пристроїв.

Попри свої численні переваги, медіа-запити мають певні обмеження, які впливають на їх ефективність:

– складність у підтримці великих проєктів. У великих веб-додатках кількість медіа-запитів може значно збільшуватися, ускладнюючи їхню підтримку та оптимізацію;

– залежність від устарілих рішень. Деякі застарілі браузери можуть частково або повністю не підтримувати сучасні специфікації медіа-запитів;

– жорсткість підходу. Медійні запити орієнтовані на статичні умови (наприклад, розмір екрану), що не враховують динамічні параметри, такі як продуктивність пристрою або доступність ресурсів;

– відсутність логіки на рівні контенту. Медійні запити не дозволяють змінювати контент залежно від пристрою, наприклад, адаптувати текст чи вибіркові блоки.

2. Практичне використання медіа-запитів.

Для ефективного використання медіа-запитів доцільно дотримуватися таких рекомендацій:

– мобільний перший підхід (Mobile First): починати з базових стилів для мобільних пристроїв та поступово додавати стилі для більших екранів;

– оптимізація продуктивності: уникати надмірної кількості запитів, групуючи їх за схожими умовами;

– тестування на реальних пристроях: перевіряти роботу медіа-запитів на різних платформах та браузерах для забезпечення сумісності.

CSS медіа-запити є потужним інструментом для створення адаптивного дизайну, забезпечуючи гнучкість та зручність користування сайтом на різних пристроях. Однак їх ефективне використання вимагає ретельного планування, тестування та оптимізації. У наступних розділах буде розглянуто додаткові технології, які можуть доповнити можливості медіа-запитів та спростити процес розробки адаптивного дизайну.

3.2 Фреймворки для адаптивного дизайну

Фреймворки для адаптивного дизайну є невід'ємною частиною сучасної розробки веб-додатків. Вони забезпечують готові рішення для створення адаптивних макетів, що дозволяє значно скоротити час розробки та підвищити ефективність роботи. Найбільш популярними серед них є Bootstrap, Foundation, а також Tailwind CSS, Bulma та Materialize. У цьому розділі розглядаються особливості, переваги та недоліки зазначених інструментів.

Bootstrap є найпопулярнішим фреймворком для адаптивної верстки, розробленим командою Twitter.

Особливості**:**

– 12-колонна сітка, яка дозволяє створювати макети будь-якої складності;

– готові компоненти: кнопки, навігаційні панелі, модальні вікна тощо;

– підтримка мобільного першого підходу (Mobile First);

– вбудована адаптивність через медіа-запити.

Переваги**:**

– велика документація та підтримка спільноти;

– легке впровадження та інтеграція у проєкти;

– сумісність із більшістю сучасних браузерів.

Недоліки**:**

– можливе перевантаження коду через використання стандартних стилів;

– одноманітність дизайнів через часте використання базових компонентів.

Foundation – фреймворк, створений компанією Zurb, що орієнтований на професійних розробників.

Особливості**:**

– розширена система сіток із можливістю створення складних макетів;

– інструменти для створення доступних інтерфейсів (accessibility);

– підтримка Sass для гнучкої кастомізації.

Переваги**:**

– широкий набір інструментів для кастомізації;

– більша гнучкість у порівнянні з Bootstrap.

Недоліки**:**

– менш популярний, що ускладнює пошук спільноти та навчальних матеріалів;

– більш складний у використанні для новачків.

Tailwind CSS відрізняється від традиційних фреймворків тим, що пропонує утилітарний підхід до стилізації.

Особливості:

– використання класів для кожної стилістичної властивості;

– висока кастомізація через конфігураційні файли.

Переваги**:**

– зменшення використання зайвого CSS;

– гнучкість у створенні унікальних дизайнів.

Недолік: більше часу на вивчення та налаштування.

Для вибору відповідного фреймворку важливо враховувати специфіку проєкту. У таблиці 3.1 наведено основні характеристики.

Таблиця 3.1 – Порівняльний аналіз фреймворків

Фрейм-ворк	Система сіток	Компо-ненти	Докумен-тація	Кастомізація	Складність використання
Bootstrap	12 колонок	Велика	Відмінна	Середня	Низька
Foundation	Гнучка	Велика	Хороша	Висока	Середня
Tailwind	Відсутня	Відсутні	Відмінна	Дуже висока	Висока
Bulma	Flexbox	Середня	Хороша	Середня	Низька
Materialize	12 колонок	Середня	Хороша	Середня	Низька

Фреймворки для адаптивного дизайну є потужним інструментом для розробників, що дозволяє спростити та пришвидшити створення адаптивних веб-додатків. Bootstrap залишається найпопулярнішим завдяки простоті використання та великій спільноті, тоді як Foundation і Tailwind CSS пропонують більше можливостей для кастомізації. Вибір інструменту залежить від вимог конкретного проєкту та рівня підготовки розробника.

3.3 Використання гнучких сіток і відносних одиниць

Гнучкі сітки та відносні одиниці є основою адаптивного веб-дизайну, який дозволяє створювати макети, що автоматично підлаштовуються під різні розміри екранів і пристрої. Вони дозволяють підтримувати зручність користувацького інтерфейсу на будь-якому пристрої, від мобільних телефонів до великих моніторів, без необхідності створення окремих версій сайту для кожного розміру екрана. Використання таких технологій вимагає розуміння їхніх основних принципів і можливостей.

Гнучкі сітки базуються на використанні відносних величин для визначення ширини елементів, що дозволяє їм змінювати свої розміри в залежності від доступного простору. Однією з основних переваг гнучких сіток є можливість забезпечити адекватне відображення контенту на різних пристроях без необхідності переписувати код чи застосовувати медіа-запити для кожної конкретної ситуації. Це особливо корисно для створення сайтів, які повинні добре працювати як на мобільних пристроях, так і на десктопах.

Для реалізації гнучких сіток найчастіше використовують CSS властивості, такі як display: flex та display: grid. Flexbox дозволяє створювати гнучкі контейнерні елементи, де блоки автоматично адаптуються до доступного простору, розподіляючи його між собою. Використовуючи Flexbox, можна легко створювати лінійні макети, де елементи вирівнюються по горизонталі чи вертикалі, зберігаючи пропорції та зручність для користувача.

Grid Layout, з іншого боку, дозволяє створювати більш складні двовимірні сітки, які забезпечують точне вирівнювання елементів як по вертикалі, так і по горизонталі. Це робить Grid дуже потужним інструментом для створення складних макетів, таких як багатоетапні форми чи карточки товарів на e-commerce сайтах.

Окрім гнучких сіток, для забезпечення адаптивності також використовуються відносні одиниці вимірювання, такі як проценти (%), em, rem, vw (відсоток ширини вікна перегляду) та vh (відсоток висоти вікна перегляду). Використання цих одиниць дозволяє елементам масштабуватися пропорційно до розміру екрану або контейнера. Наприклад, встановлення ширини елемента у відсотках дозволяє йому автоматично змінювати розмір залежно від ширини батьківського елемента. Це дає змогу створювати інтерфейси, які не "ламаться" при зміні розміру екрану.

Відносні одиниці, такі як em та rem, дозволяють задавати розміри елементів, шрифтів і відступів, що залежать від розміру шрифтів на попередньому рівні або кореневого елемента відповідно. Цей підхід

забезпечує більш точне керування розмірами елементів на різних пристроях, особливо для адаптації шрифтів до різних роздільних здатностей екранів.

Застосування гнучких сіток у поєднанні з відносними одиницями дозволяє створювати веб-сторінки, які здатні автоматично підлаштовуватися під екран будь-якого пристрою, зберігаючи при цьому функціональність і естетичний вигляд. Це робить такі макети ідеальними для сучасних веб-додатків, де важлива зручність користувача та універсальність дизайну. Гнучкі сітки і відносні одиниці не тільки полегшують процес розробки, але й забезпечують високу продуктивність веб-сайтів, оскільки дозволяють зменшити кількість необхідних стилів і медіа-запитів для адаптації до різних пристроїв.

Незважаючи на численні переваги, використання гнучких сіток і відносних одиниць має свої обмеження. Інколи досягнення бажаного макета може вимагати складних налаштувань, особливо якщо потрібно підтримувати підтримку старих браузерів або специфічні сценарії для різних пристроїв. Проте з розвитком веб-технологій ці проблеми стають менш актуальними, і сучасні браузери вже підтримують більшість функцій, що дозволяють використовувати ці методи для створення складних, але ефективних адаптивних інтерфейсів.

В результаті, гнучкі сітки та відносні одиниці є важливими інструментами для створення адаптивних веб-дизайнів. Вони дозволяють забезпечити максимальну гнучкість і адаптацію інтерфейсу до різноманітних умов роботи, що значно підвищує зручність користувачів і ефективність розробки веб-сайтів.

3.4 Інтеграція сучасних технологій

Сучасні технології CSS, такі як Flexbox та Grid Layout, значно змінили підхід до верстки веб-сторінок, надаючи розробникам потужні інструменти для створення складних, адаптивних макетів з мінімальними зусиллями. Ці технології забезпечують гнучкість, дозволяючи створювати макети, які

адаптуються до різних розмірів екранів і платформ, що є важливим для досягнення високої продуктивності та зручності користування.

Flexbox (Flexible Box Layout) є модулем CSS, що дозволяє створювати макети, де елементи можуть автоматично підлаштовуватись під доступний простір. Flexbox є особливо корисним для створення лінійних макетів, де елементи вирівнюються в одному напрямку – по горизонталі або вертикалі. Однією з ключових особливостей Flexbox є його здатність автоматично розподіляти простір між елементами, дозволяючи їм рости або стисатися залежно від доступного місця. Це важливо для адаптивного дизайну, оскільки дозволяє елементам змінювати свої розміри без втрати пропорцій або порушення структури макета.

Flexbox особливо ефективний при створенні простих макетів, таких як навігаційні меню, форми, картки товарів або групи кнопок. Завдяки властивостям, таким як justify-content, align-items та flex-wrap, розробники можуть легко керувати вирівнюванням елементів, їхнім розподілом по осі та перенесенням елементів на новий рядок при необхідності. Ці можливості дозволяють створювати гнучкі інтерфейси, що автоматично адаптуються до різних розмірів вікна перегляду, забезпечуючи зручність користувача.

З іншого боку, Grid Layout є більш потужним і гнучким інструментом для створення двовимірних макетів, де елементи можуть бути розташовані не тільки по горизонталі, а й по вертикалі. CSS Grid дозволяє розробникам створювати складні макети з численними рядами і стовпцями, що дає змогу точно керувати розташуванням кожного елемента на сторінці. Однією з основних переваг Grid Layout є можливість створювати макети без необхідності використання додаткових контейнерів, що зменшує складність і кількість коду. Grid Layout значно покращує процес розробки, оскільки дозволяє створювати візуально складні макети за допомогою простих CSS-властивостей, таких як grid-template-columns, grid-template-rows, grid-gap і align-items. Це дозволяє розташовувати елементи за допомогою сітки, де можна вказати ширину, висоту і відстань між рядами та стовпцями. CSS Grid

особливо корисний для створення таких складних макетів, як сітки з картками, розширені форми, багатоетапні таблиці та інші елементи, де точне вирівнювання важливе для зручності користувача.

Важливим аспектом є можливість поєднувати Flexbox і Grid Layout в одному проєкті. Це дає змогу вибирати найбільш підходящий інструмент для кожної частини макета, поєднуючи гнучкість Flexbox для лінійних елементів та потужність Grid Layout для більш складних і багатовимірних структур. Таке поєднання дозволяє розробникам створювати адаптивні веб-сайти, які мають не лише зручний вигляд на різних пристроях, але й високу функціональність.

Використання Flexbox і Grid Layout у поєднанні з відносними одиницями вимірювання, такими як проценти, em або rem, дає ще більшу гнучкість у створенні адаптивних макетів. Завдяки цим технологіям можна легко керувати розмірами елементів, їхнім розташуванням і відстанями на сторінці, забезпечуючи гармонійний вигляд інтерфейсу на різних пристроях і роздільних здатностях екранів.

Ці технології також мають важливе значення для покращення продуктивності веб-сторінок. Використовуючи Flexbox і Grid Layout, розробники можуть зменшити кількість необхідних медіа-запитів і зменшити загальний обсяг CSS-коду, оскільки ці інструменти дозволяють створювати адаптивні макети без великих змін у стилях на різних екранах. Це сприяє зменшенню часу завантаження сторінок і підвищенню швидкості роботи сайту, що є важливим аспектом для забезпечення хорошого користувацького досвіду.

Висновок, який можна зробити на основі інтеграції Flexbox і Grid Layout, полягає в тому, що ці технології значно спрощують процес створення адаптивних веб-сторінок. Вони дозволяють розробникам легко та ефективно створювати гнучкі, масштабовані та функціональні макети, що автоматично підлаштовуються під різні пристрої та розміри екранів. В поєднанні з іншими сучасними технологіями CSS, такими як медіа-запити та відносні одиниці, Flexbox і Grid Layout забезпечують потужний інструментарій для створення високоякісних веб-дизайнів.

4 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ІНСТРУМЕНТІВ АДАПТИВНОЇ ВЕРСТКИ

4.1 Критерії оцінки адаптивності та продуктивності сайтів

Оцінка адаптивності та продуктивності сайтів є важливим етапом у розробці та тестуванні веб-додатків. Вона дозволяє визначити, наскільки ефективно сайт працює на різних пристроях та у різних браузерах, а також які ресурси витрачаються під час його завантаження і взаємодії з користувачем. У цьому контексті адаптивність та продуктивність є двома основними факторами, що визначають якість користувацького досвіду.

Адаптивність сайту вимірюється здатністю коректно відображати контент на різних екранах – від мобільних телефонів до великих моніторів. Сайт, що має високу адаптивність, повинен забезпечувати зручне читання тексту, правильне розташування елементів і збереження функціональності незалежно від того, на якому пристрої він переглядається. Одним із основних критеріїв є адаптивний дизайн, який дозволяє елементам сторінки автоматично змінювати свій вигляд, розміри та позицію залежно від ширини екрана. Важливим аспектом є також використання медіа-запитів, які дозволяють на різних етапах завантаження сайту адаптувати його вміст до специфікацій пристрою користувача.

Продуктивність, з іншого боку, стосується швидкості завантаження сайту та ефективності використання ресурсів. Оцінка продуктивності включає в себе вимірювання часу завантаження сторінки, часу відгуку сервера, а також рівня використання пам’яті та процесора. Важливим аспектом є оптимізація зображень, скриптів та інших ресурсів, щоб мінімізувати час завантаження. Продуктивність також можна оцінювати за допомогою різноманітних інструментів, таких як Google PageSpeed Insights або Lighthouse, які надають детальні рекомендації для покращення швидкості завантаження.

Для всебічної оцінки адаптивності та продуктивності необхідно враховувати також такі критерії, як кросбраузерність, доступність (accessibility) і безперебійна робота на різних платформах. Сучасні інструменти для тестування адаптивності дозволяють симулювати різні умовні налаштування екранів, що робить цей процес ефективним та чітким.

4.2 Проведення порівняльного аналізу фреймворків

Аналіз фреймворків для адаптивної верстки є важливим етапом для вибору оптимальних інструментів для розробки веб-додатків. Існує кілька популярних фреймворків, таких як Bootstrap, Foundation, Bulma та інші, кожен з яких має свої сильні сторони та обмеження.

Першим кроком у порівняльному аналізі є вивчення основних можливостей фреймворків, таких як наявність вбудованих компонентів для створення адаптивних елементів, систем сіток, стилів для кнопок, форм, карток та інших елементів інтерфейсу. Наприклад, Bootstrap пропонує розвинену систему сіток, що дозволяє легко створювати адаптивні макети, включаючи компоненти для створення меню, модальних вікон, карток і багатьох інших елементів. Водночас, Foundation надає більш гнучкий підхід до налаштувань, надаючи розробникам більшу свободу у проектуванні макетів і стилів.

Під час порівняння важливим аспектом є також документація та підтримка спільноти. Фреймворк з хорошою документацією полегшує роботу з ним, оскільки розробник може швидко знайти відповіді на свої питання та знайти оптимальні рішення для своєї задачі. Підтримка спільноти є важливою, оскільки дозволяє отримати допомогу в разі виникнення проблем.

Ще одним важливим фактором є ефективність коду, який генерується фреймворками. Різні фреймворки можуть мати різний розмір коду, що впливає на швидкість завантаження сайту та продуктивність. Наприклад, хоча Bootstrap є потужним інструментом, він може бути більш важким для завантаження через велику кількість вбудованих компонентів, тоді як легші

фреймворки, такі як Bulma, займають менше місця в коді та можуть бути кращими для швидких і простих проектів.

Нарешті, важливо враховувати можливості кастомізації та розширюваності. Деякі фреймворки надають вбудовані теми та стилі, що дозволяє швидко налаштувати зовнішній вигляд сайту, інші ж дають більшу свободу у налаштуванні, але можуть потребувати більше часу на розробку.

4.3 Дослідження впливу адаптивної верстки на швидкість завантаження сторінок

Одним із важливих аспектів адаптивної верстки є її вплив на швидкість завантаження сторінок. Адаптивні верстки зазвичай передбачають використання медіа-запитів, різних стилів для мобільних та десктопних версій сайтів, а також оптимізацію контенту для різних пристроїв. Це може вплинути на час завантаження сторінки, оскільки на мобільних пристроях завантажуються лише необхідні елементи, тоді як на десктопах завантажуються більше ресурси.

Адаптивна верстка дозволяє зменшити кількість запитів до сервера, оскільки медіа-запити застосовуються до вже завантажених стилів, що дозволяє уникнути дублювання контенту для різних платформ. Однак, для оптимізації швидкості завантаження важливо правильно налаштовувати зображення, шрифти та інші елементи, щоб вони не займали занадто багато місця і швидко завантажувались. Вплив адаптивної верстки на швидкість завантаження залежить також від того, як реалізовані медіа-запити та як вони оптимізують завантаження елементів для кожного пристрою.

Крім того, важливим фактором є також використання кешування, стиснення ресурсів і асинхронне завантаження скриптів. Це дозволяє значно зменшити час завантаження сторінки, навіть якщо на сайті використовується адаптивна верстка з великою кількістю елементів. Оцінка впливу адаптивної верстки на швидкість завантаження може бути проведена за допомогою

інструментів, таких як Google PageSpeed Insights або Lighthouse, які дозволяють виміряти час завантаження сторінки, а також надати рекомендації щодо її оптимізації.

Загалом, ефективна адаптивна верстка повинна враховувати баланс між функціональністю та швидкістю завантаження. Під час дослідження впливу адаптивної верстки на швидкість важливо враховувати не лише використання медіа-запитів і оптимізацію контенту, але й загальні принципи оптимізації веб-сторінок, що дозволяють досягти високої продуктивності при будь-яких умовах.

5 ПЛАНУВАННЯ ДОСЛІДЖЕННЯ

5.1 План проведення експерименту

Проведення експерименту з дослідження інструментальних засобів адаптивної верстки вебсайтів вимагає чіткого та структурованого підходу. План експерименту передбачає такі етапи:

1. Визначення критеріїв оцінки інструментів.

Розробити список критеріїв, які дозволяють об’єктивно оцінити ефективність інструментів. Основними критеріями можуть бути:

– простота використання;
– гнучкість налаштувань;
– продуктивність (швидкість завантаження сторінок);
– сумісність з різними браузерами та пристроями;
– підтримка сучасних стандартів веброзробки;
– наявність документації та спільноти підтримки.

2. Вибір інструментів для тестування.

Включити до експерименту популярні інструменти адаптивної верстки, такі як:

– CSS-фреймворки (Bootstrap, Foundation, Bulma);
– утилітарні бібліотеки (Tailwind CSS);
– препроцесори (Sass, LESS);
– інструменти автоматизації (Gulp, Webpack).

Створення тестових макетів.

Розробити макети вебсторінок з однаковою структурою та вмістом. Забезпечити наявність елементів, що вимагають адаптації, таких як:

– навігаційні панелі;
– графічні зображення;
– форматування тексту;
– таблиці даних.

Реалізація верстки з використанням обраних інструментів:

– застосувати кожен інструмент до тестових макетів для реалізації адаптивного дизайну.

– забезпечити створення адаптивної верстки, що відповідає різним розмірам екранів: смартфони, планшети, ноутбуки, настільні ПК.

– Тестування реалізованих макетів:

– виконати тестування на різних пристроях та в різних браузерах (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge).

– перевірити коректність відображення елементів, швидкість завантаження сторінок та загальну якість адаптації.

Аналіз результатів тестування:

– порівняти отримані результати для кожного інструмента відповідно до визначених критеріїв.

– виділити переваги та недоліки кожного рішення.

Узагальнення висновків та формування рекомендацій:

– сформулювати рекомендації щодо вибору інструментів для різних типів проєктів.

– розробити узагальнений підхід до вибору інструментальних засобів адаптивної верстки залежно від потреб користувачів.

Цей план дозволяє забезпечити послідовне та обґрунтоване проведення експерименту, результати якого будуть надійною основою для висновків та рекомендацій.

5.2 Вибір і обґрунтування методів і критеріїв оцінки якості інтерфейсу сайту з елементами адаптивної верстки

Для ефективного аналізу якості інтерфейсу вебсайту з елементами адаптивної верстки необхідно чітко визначити критерії оцінки та методи їх застосування. У даному дослідженні обрано три ключові критерії: функціональність та можливості, адаптивність і кросбраузерність, а також

швидкість та продуктивність. Ці критерії дозволяють всебічно оцінити якість реалізації адаптивного дизайну та зручність користування сайтом.

5.2.1 Функціональність та можливості.

Це ключовий критерій оцінки якості інтерфейсу вебсайту, що характеризує його здатність виконувати заявлені завдання, забезпечувати інтуїтивно зрозумілу взаємодію користувача з вебсайтом і надавати повний спектр необхідних функцій. Цей критерій особливо важливий у контексті адаптивної верстки, оскільки функціональність повинна залишатися незмінною незалежно від пристрою, що використовується.

Функціональність інтерфейсу визначає, наскільки сайт відповідає своїм основним завданням і чи забезпечує він користувачам необхідні інструменти для взаємодії. Основні аспекти функціональності та можливостей.

1. Коректність роботи всіх функціональних елементів:

– кнопки та інтерактивні елементи. Кнопки повинні виконувати заявлені дії (наприклад, перехід на іншу сторінку, запуск форм або інших процесів);

– навігація. Меню має забезпечувати зручний доступ до всіх розділів сайту, включаючи випадаючі меню або інші інтерактивні компоненти;

– форми зворотного зв’язку. Поля вводу, кнопки надсилання, випадаючі списки та інші елементи форм повинні працювати без помилок.

2. Відповідність функцій потребам користувачів:

– сайт має відповідати очікуванням аудиторії. Наприклад, інтернет-магазин повинен мати функціонал пошуку товарів, фільтрації, корзини та оплати;

– необхідно уникати "зайвих" функцій, які ускладнюють інтерфейс і не додають цінності для користувача.

3. Інтуїтивність і зручність використання:

– простота доступу до основних функцій. Наприклад, кнопка "Купити" або "Замовити" має бути розташована на видному місці;

– логічна структура. Інтерфейс повинен мати зрозумілу ієрархію, де основні функції розташовані у відповідних розділах;

– візуальні підказки. Анімації, підсвічування, повідомлення про помилки мають допомагати користувачу зрозуміти, як взаємодіяти з сайтом.

4. Доступність для користувачів з особливими потребами:

– підтримка стандартів доступності (WCAG). Сайт повинен бути доступним для людей із порушеннями зору, слуху чи моторики. Наприклад, використання семантичного HTML для полегшення роботи з екранними рідерами;

– доступність клавіатурної навігації. Всі функціональні елементи мають бути доступними без використання миші.

5. Стабільність роботи функцій:

– елементи інтерфейсу повинні коректно працювати незалежно від розміру екрану та версії браузера;

– важливо уникати ситуацій, коли певна функція не працює на мобільному пристрої, але працює на десктопі.

Методи оцінки функціональності.

Для оцінки функціональності вебсайту застосовуються різні методи.

1. Функціональне тестування:

– ручне або автоматизоване тестування всіх функцій сайту, включаючи кнопки, меню, форми та інші інтерактивні елементи;

– використання інструментів (Selenium), для автоматизації тестування.

2. Юзабіліті-тестування:

– залучення реальних користувачів для оцінки інтуїтивності та зручності використання функціоналу;

– проведення тестових сценаріїв, наприклад: "Знайти товар і додати його в кошик", "Зареєструватися на сайті".

Тестування на реальних пристроях: перевірка роботи функцій сайту на смартфонах, планшетах та комп’ютерах. Це дозволяє виявити проблеми адаптивності функцій.

3. Перевірка відповідності стандартам:

– використання валідаторів HTML і CSS для перевірки коду. Наприклад, інструмент W3C Validator допомагає виявити помилки у структурі сторінки;

– оцінка доступності за допомогою таких інструментів, як Axe Accessibility або Lighthouse.

Функціональність є базовою характеристикою якісного інтерфейсу, оскільки навіть найпривабливіший дизайн не буде ефективним без правильно реалізованих функцій. У контексті адаптивної верстки забезпечення функціональності означає, що:

– усі функції сайту доступні незалежно від типу пристрою;

– інтерактивні елементи (кнопки, форми, меню) працюють однаково добре на екранах різних розмірів;

– дизайн адаптується так, щоб ключові функції залишалися доступними навіть на найменших екранах.

Функціональність та можливості вебсайту, це основа його ефективності. Чітка реалізація всіх функцій, орієнтованість на користувача та стабільність роботи є критичними факторами для успішного використання адаптивної верстки. Високоякісний функціонал дозволяє забезпечити позитивний досвід користувачів і підвищити ефективність сайту в досягненні його бізнес-цілей.

Адаптивність і кросбраузерність як основні принципи сучасного веб-дизайну для забезпечення універсальності та зручності користування веб-ресурсами на різних платформах.

Адаптивність – це здатність вебсайту підлаштовуватися під різні розміри та характеристики екранів пристроїв, зберігаючи коректне відображення контенту та повну функціональність. Сьогодні адаптивність є ключовою вимогою для вебсайтів, оскільки користувачі заходять на сайти з різноманітних пристроїв: смартфонів, планшетів, ноутбуків, настільних ПК, а також пристроїв із нестандартними розмірами екранів, як-от смарт-телевізори.

Основні аспекти адаптивності:

– коректне масштабування елементів інтерфейсу, елементи дизайну (текст, кнопки, зображення, таблиці) повинні автоматично змінювати свої розміри залежно від розміру екрана. Наприклад, шрифт має збільшуватися або зменшуватися таким чином, щоб залишатися читабельним;

– графічні елементи повинні пропорційно змінюватися, зберігаючи співвідношення сторін, і не створювати горизонтальної прокрутки;

– гнучке компонування сторінок використання технологій, таких як CSS Flexbox і CSS Grid, дозволяє розташовувати елементи на сторінці у зручний спосіб незалежно від розмірів вікна браузера;

– наприклад, на мобільних пристроях елементи, розташовані в ряд на великих екранах, можуть змінювати компонування на вертикальне для зручнішої навігації;

– підтримка медіазапитів (CSS Media Queries) адаптивність забезпечується за допомогою медіазапитів, які дозволяють змінювати стиль залежно від параметрів пристрою, таких як ширина екрану, орієнтація (пейзажна чи портретна), роздільна здатність;

– можна встановити різні стилі для пристроїв із шириною екрана менше 768 пікселів (мобільні телефони) та більше 1024 пікселів (настільні комп’ютери);

– оптимізація для сенсорних пристроїв елементи інтерфейсу повинні бути зручними для взаємодії за допомогою сенсорного екрану. Наприклад, кнопки повинні бути достатньо великими, щоб їх можна було легко натиснути пальцем, а посилання – розташованими на відстані, щоб уникнути випадкових натискань.

Методи оцінки адаптивності:

– тестування на різних пристроях, використання реальних пристроїв (смартфони, планшети, ноутбуки) для перевірки коректності відображення;

– інструменти розробника в браузерах, наприклад, Google Chrome DevTools дозволяє емулювати різні розміри екранів і орієнтації пристроїв;

– онлайн-платформи для тестування, BrowserStack, Responsinator, LambdaTest забезпечують перевірку адаптивності без фізичних пристроїв.

Кросбраузерність – це здатність вебсайту однаково добре функціонувати та виглядати у всіх популярних браузерах. Це включає як коректне відображення стилів, так і стабільну роботу функціональних елементів, незалежно від використовуваного програмного забезпечення.

Основні аспекти кросбраузерності.

Сумісність із популярними браузерами, вебсайт повинен підтримувати всі поширені браузери, включаючи Google Chrome, Mozilla Firefox, Safari, Microsoft Edge та Opera. У деяких випадках важливо забезпечити підтримку старих версій браузерів, таких як Internet Explorer (якщо це необхідно для цільової аудиторії).

Стабільність відображення стилів,у різних браузерах можуть бути відмінності у реалізації CSS-стилів. Для забезпечення кросбраузерності розробники використовують:

– нейтральні CSS-резети (normalize.css), які приводять стилі до однакового вигляду в усіх браузерах;

– перевірку підтримки властивостей, наприклад, за допомогою Can I Use.

Сумісність JavaScript-функціоналу: JavaScript-код може працювати по-різному залежно від браузера. Для вирішення цих проблем використовують поліфіли (polyfills) і бібліотеки, які забезпечують однакову роботу функцій у різних середовищах (наприклад, Babel).

Використання стандартів веброзробки: дотримання стандартів W3C забезпечує більш високий рівень кросбраузерної сумісності.

Методи оцінки кросбраузерності.

1. Ручне тестування у всіх основних браузерах це дозволяє виявити візуальні та функціональні відмінності.

2. Використання автоматизованих тестувань, наприклад, інструменти Selenium або Cypress дозволяють перевіряти функціональність сайту у різних браузерах.

3. Онлайн-інструменти для перевірки кросбраузерності, BrowserStack, CrossBrowserTesting допомагають швидко оцінити сумісність на різних пристроях та у різних браузерах.

Спільне значення адаптивності та кросбраузерності. Поєднання адаптивності та кросбраузерності забезпечує максимально широкий доступ до вебсайту для користувачів незалежно від пристроїв чи браузерів, які вони використовують. Це підвищує якість користувацького досвіду, задоволеність аудиторії та конкурентоспроможність вебресурсу.

Адаптивність і кросбраузерність – це два взаємопов’язані аспекти, які забезпечують універсальність, зручність та ефективність вебсайту. Висока якість реалізації цих параметрів дозволяє гарантувати, що вебсайт буде доступним і зручним для максимальної кількості користувачів, незалежно від їхніх технічних умов і переваг.

5.2.2 Швидкість та продуктивність вебсайту

Швидкість та продуктивність вебсайту є критичними характеристиками, що безпосередньо впливають на користувацький досвід, поведінку відвідувачів, рейтинг у пошукових системах та конверсію. Швидкий вебсайт забезпечує миттєву взаємодію з контентом, тоді як продуктивність визначає, наскільки ефективно сайт виконує свої завдання навіть за умов високого навантаження.

Основні аспекти швидкості та продуктивності.

Час завантаження сторінки, оптимальний час завантаження вебсторінки повинен становити не більше 2-3 секунд, оскільки більшість користувачів залишають сайт, якщо завантаження займає більше часу.

Ключовими етапами завантаження є:

– початкове завантаження контенту (DOMContentLoaded);

– повне завантаження сторінки (Load Event).

Оптимізація графічних елементів:

– стиснення зображень. Великі файли зображень є головною причиною повільного завантаження. Стиснення без втрати якості (за допомогою інструментів, таких як TinyPNG, ImageOptim) зменшує обсяг файлів;

– використання сучасних форматів. Формати зображень, як-от WebP, забезпечують менший розмір файлів при збереженні якості;

– ліниве завантаження (Lazy Loading). Зображення завантажуються лише тоді, коли користувач скролить до відповідного розділу сторінки.

Мінімізація CSS, JavaScript та HTML:

– видалення зайвих пробілів, коментарів і непотрібного коду з CSS, JavaScript і HTML файлів. Це зменшує їх розмір і прискорює завантаження;

– використання інструментів для мінімізації, таких як UglifyJS або CSSNano.

Кешування:

– використання кешування зменшує час завантаження, оскільки браузер зберігає статичні ресурси (зображення, CSS, JavaScript) локально;

– HTTP-кешування. Встановлення заголовків кешу (наприклад, Cache-Control, Expires) дозволяє браузеру зберігати файли локально для повторного використання.

Зменшення кількості HTTP-запитів:

– злиття файлів CSS і JavaScript в один або кілька більших файлів;

– використання CSS Sprite для об’єднання кількох іконок в один файл.

Підключення Content Delivery Network (CDN): CDN розподіляє статичні ресурси сайту (зображення, скрипти, стилі) через мережу серверів по всьому світу, що забезпечує їх швидке завантаження для користувачів з різних регіонів.

Асинхронне завантаження ресурсів: завантаження JavaScript-файлів у фоновому режимі за допомогою атрибутів async або defer. Це дозволяє браузеру не блокувати рендеринг сторінки під час завантаження скриптів.

Використання легких і оптимізованих бібліотек: вибір менш важких бібліотек JavaScript і CSS для підвищення продуктивності. Наприклад, використання Vanilla JS замість великих бібліотек, таких як jQuery.

Виявлення та усунення «важких» запитів:

– моніторинг і оптимізація запитів до бази даних. Запити повинні бути швидкими та використовувати індексацію;

– впровадження кешування запитів до бази даних.

Стабільність роботи під навантаженням:

– тестування сайту за умов високого навантаження дозволяє оцінити його здатність обслуговувати велику кількість користувачів одночасно;

– використання інструментів для навантажувального тестування, таких як Apache JMeter або k6, дозволяє змоделювати реальні сценарії навантаження.

Методи оцінки швидкості та продуктивності.

Інструменти аналізу швидкості:

– Google PageSpeed Insights: Оцінює швидкість завантаження сайту на мобільних пристроях і ПК, надає рекомендації щодо оптимізації;

– GTmetrix: Вимірює час завантаження, кількість HTTP-запитів, продуктивність браузера і пропонує покращення;

– Lighthouse: Інтегрований у браузер Chrome, забезпечує оцінку продуктивності, зручності використання та SEO.

Тестування під навантаженням: симуляція роботи сайту з великою кількістю одночасних користувачів допомагає визначити точки відмови та оптимізувати продуктивність.

Моніторинг у реальному часі: інструменти, як-от New Relic, дозволяють відстежувати продуктивність сайту в реальному часі, виявляти проблеми та вузькі місця.

Значення швидкості та продуктивності:

– позитивний користувацький досвід. Час завантаження сайту є критично важливим фактором для користувачів. Дослідження показують, що

40% користувачів залишають сайт, якщо його завантаження триває більше 3 секунд;

– SEO-рейтинг. Швидкість завантаження впливає на ранжування сайту у пошукових системах, таких як Google. Швидший сайт отримує вищі позиції в пошуковій видачі;

– ефективність бізнесу. Швидший сайт сприяє збільшенню конверсій та зниженню показника відмов.

Швидкість та продуктивність є критично важливими для успіху будь-якого вебсайту. Високопродуктивний сайт забезпечує позитивний досвід для користувачів, високе місце в пошукових системах та підтримує стабільну роботу навіть під час високого навантаження. Впровадження сучасних методів оптимізації дозволяє досягти цих цілей, роблячи вебресурс зручним, швидким і надійним.

Обґрунтування вибору критеріїв. Вибір зазначених критеріїв обумовлений сучасними вимогами до вебсайтів:

Функціональність визначає, чи може сайт виконувати свої завдання та чи забезпечує він належну взаємодію з користувачами.

Адаптивність та кросбраузерність гарантують доступність сайту для широкої аудиторії незалежно від пристрою чи браузера.

Швидкість та продуктивність впливають на задоволеність користувачів і рейтинг сайту в пошукових системах.

Таким чином, ці критерії забезпечують всебічну оцінку якості сайту з елементами адаптивної верстки, а застосовані методи дозволяють об’єктивно визначити його переваги та недоліки.

5.3 Визначення набору функцій сайту, що використовують інструменти адаптивного дизайну

При розробці сучасних вебсайтів із використанням інструментів адаптивного дизайну важливо визначити набір функцій, які повинні

забезпечити не лише коректне відображення на різних пристроях, але й високу зручність для користувачів. Адаптивний дизайн має на меті зробити сайт доступним та функціональним незалежно від розміру екрану, браузера або типу пристрою.

Основні функції, що реалізуються за допомогою адаптивного дизайну. Гнучке компонування та адаптація макету: розподіл елементів на сторінці. Забезпечується коректне розташування контенту, включаючи текст, зображення, кнопки та інші елементи, відповідно до ширини та орієнтації екрана. Ввикористання технологій, таких як CSS Flexbox і CSS Grid, для динамічної зміни компонування сторінок. Медіазапити CSS Media Queries дозволяють адаптувати стилі під різні екрани (наприклад, смартфони, планшети, настільні ПК).

Адаптивна типографіка. Динамічне масштабування шрифтів залежно від розміру екрана для збереження читабельності тексту. Використання відносних одиниць вимірювання шрифтів, таких як em, rem, або vw/vh, замість фіксованих значень у пікселях.

Респонсивні зображення та медіа. Автоматичне масштабування зображень залежно від роздільної здатності та розміру екрана. Використання тегу <picture> та атрибуту srcset, які дозволяють завантажувати зображення оптимального розміру для конкретного пристрою. Впровадження лінивого завантаження (Lazy Loading) для підвищення швидкості завантаження сторінки.

Адаптивне меню навігації:

– зміна навігаційного меню на компактний вигляд (наприклад, "гамбургер-меню") на мобільних пристроях;

– забезпечення доступності меню за допомогою сенсорних жестів та клавіатури.

Форми зворотного зв’язку та вводу даних. Адаптація форм під мобільні пристрої, включаючи:

– великі поля вводу для зручності використання пальцями;

– автоматичний вибір відповідного типу клавіатури (наприклад, числової для введення номерів телефону);

– забезпечення адаптивності повідомлень про помилки та валідацію даних.

Модальні вікна та спливаючі елементи: реалізація модальних вікон, що підлаштовуються під розміри екрана. Наприклад, на великих екранах модальні вікна можуть бути в центрі, а на мобільних у повноекранному режимі.

Оптимізація для сенсорних пристроїв:

– реалізація елементів, чутливих до жестів, таких як свайпи, збільшення чи зменшення масштабу;

– підвищення розміру інтерактивних елементів (кнопок, чекбоксів) для зручності використання пальцем.

Динамічна робота мультимедіа:

– масштабування відео та інтерактивних елементів (наприклад, інтегрованих карт) для забезпечення коректного відображення на різних пристроях;

– використання адаптивних плеєрів, таких як Plyr.js, які змінюють свої розміри залежно від екрана.

Підтримка багатомовності та регіональних налаштувань: забезпечення коректного відображення текстів та інтерфейсу різними мовами. Наприклад, зміна напряму тексту для мов із письмом справа наліво (RTL), таких як арабська чи іврит.

Інтеграція анімацій: створення легких адаптивних анімацій, які не перевантажують систему і коректно працюють на мобільних пристроях. Наприклад, анімації появи елементів при скролі або кліку.

Вибір інструментів для реалізації функцій адаптивного дизайну.

CSS-фреймворки:

– Bootstrap: забезпечує готові респонсивні компоненти, такі як грід-система, адаптивне меню, модальні вікна тощо;

– Foundation: пропонує гнучкіші налаштування для створення

адаптивного дизайну;

– Tailwind CSS: утилітарний підхід для створення кастомного адаптивного дизайну.

Бібліотеки JavaScript:

– Modernizr: дозволяє визначити, які функції браузера підтримуються, і адаптувати сайт відповідно;

– GSAP: для створення плавних адаптивних анімацій.

Препроцесори CSS: використання Sass чи LESS для організації коду та налаштування адаптивності за допомогою змінних і міксинів.

Визначення набору функцій сайту, що використовують інструменти адаптивного дизайну, є ключовим етапом у створенні сучасного вебресурсу. Такі функції, як гнучке компонування, адаптивна типографіка, респонсивні медіа, оптимізоване меню навігації, форми та мультимедіа, забезпечують комфортне використання сайту на будь-якому пристрої. Вибір відповідних інструментів дозволяє реалізувати ці функції ефективно, зберігаючи високу якість і продуктивність ресурсу.

6 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

6.1 Проектування структури сайту

Сайт призначений для ознайомлення користувачів з традиційними українськими стравами, їх історією, рецептами та культурною спадщиною. Основна мета проєкту – популяризація української кухні серед широкої аудиторії та надання зручного інструменту для вивчення рецептів та замовлення страв.

Головні вимоги до сайту:

– інтуїтивна структура, забезпечення логічної організації контенту, щоб користувачі могли легко знаходити рецепти та додаткову інформацію;

– адаптивний дизайн, підтримка коректного відображення на різних пристроях (смартфонах, планшетах, комп’ютерах);

– простота навігації, розробка зрозумілої системи меню та посилань;

– модульність, розділення сайту на функціональні частини (наприклад, розділи з рецептами, інформація про історію страв, контактна сторінка).

Інформаційна структура сайту визначає логіку організації контенту та його подання користувачам. Сайт складається з таких основних розділів.

Головна сторінка:

– вітальний блок із загальною інформацією про українську кухню;

– прев’ю ключових розділів сайту, таких як "Рецепти", "Про нас", "Контакти";

– блок із закликом до дії (наприклад, "Досліджуйте рецепти зараз!").

Рецепти:

– список категорій (наприклад, перші страви, основні страви, десерти);

– сторінки з детальним описом кожного рецепту, включаючи: зображення готової страви, перелік інгредієнтів, покрокову інструкцію приготування.

– фільтри для пошуку (за категоріями, складністю, часом приготування).

Про нас:

– інформація про мету створення сайту та його авторів;

– розділ про культурне значення української кухні.

Контакти:

– контактна форма для зворотного зв’язку;

– інформація про соціальні мережі та можливість підписки на оновлення сайту.

Логічна структура сайту базується на використанні сучасних вебтехнологій та модульного підходу. Структура реалізована у вигляді багаторівневої системи:

– верхній рівень – головна сторінка, яка виконує роль навігаційного центру;

– другий рівень – основні розділи ("Рецепти", "Про нас", "Контакти";

– третій рівень – сторінки кожного конкретного рецепту або статті.

Сайт побудований з використанням таких технологій:

– HTML – для створення структури сторінок. Використано семантичні елементи (наприклад, <header>, <nav>, <main>, <footer>), що забезпечують зручність для користувачів та SEO-оптимізацію;

– CSS – для стилізації сторінок. Реалізовано адаптивний дизайн за допомогою медіазапитів (CSS Media Queries) для забезпечення коректного відображення на різних пристроях;

– JavaScript – для інтерактивності (наприклад, реалізація фільтрів, відкриття модальних вікон);

– GitHub Pages – для хостингу сайту. Це дозволяє забезпечити зручне розгортання сайту та доступ до його вихідного коду.

Принципи адаптивності та доступності.

Адаптивний дизайн:

– сайт спроєктовано для коректного відображення на екранах з різною роздільною здатністю (від 320px до 1920px);

– впроваджено "mobile-first" підхід: стилі для мобільних пристроїв визначені як базові, а стилі для більших екранів додаються через медіазапити.

Доступність

– забезпечено підтримку стандартів WCAG (Web Content Accessibility Guidelines);

– використано атрибути доступності (aria-*) для полегшення взаємодії з сайтом для користувачів із особливими потребами.

Проектування структури сайту базується на принципах простоти, зручності та адаптивності. Логічна організація контенту, використання сучасних вебтехнологій та орієнтація на потреби користувачів дозволяють створити сайт, який відповідає сучасним вимогам якості та функціональності.

6.2 Створення графічного дизайну

Основною ідеєю при розробці дизайну стало використання палітри, що складається з кольорів, які гармонійно доповнюють один одного. У цьому випадку вибір припав на яскраве і контрастне поєднання трьох класичних кольорів: чорного, білого та жовтого. Для акценту, коли необхідно залучити увагу користувача до конкретного елемента, додано малиновий відтінок, який проявляється при наведенні курсору. На рис. 6.1 зображено обрані кольори.

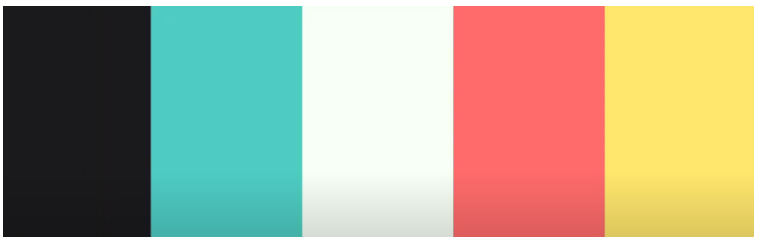


Рисунок 6.1 – Палітра

При створенні дизайну сайту було визначено головний контейнер з шириною 1312 пікселів, який залишається незмінним на всіх сторінках, забезпечуючи зручність сприйняття інформації користувачами. Для структурування контенту була використана універсальна 12-колонкова модульна сітка, що дозволяє ефективно розміщувати блоки та їх елементи.

На головній сторінці розміщено великий заголовок, який одразу інформує користувача про основну тему сайту, а також текстовий блок з коротким описом. Праворуч від тексту розташовано яскраве та привабливе зображення. Крім того, на цій сторінці є дві кнопки, які дозволяють взаємодіяти з сайтом: при натисканні на кнопку користувач перейде до наступного розділу, а іншу кнопка замінить зображення на відео. Оформлення головної в сторінки представлено на рисунку 6.2.

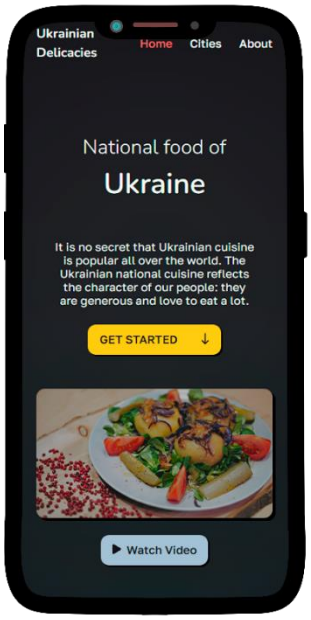


Рисунок 6.2 – Головна сторінка

На головній сторінці, якщо прокрутити її вниз, користувач зможе побачити блок, який пропонує вибір серед трьох можливих варіантів міст України. Дещо нижче розташований блок з основними перевагами цього сайту. На слайдері відображаються великі міста, однак, якщо користувач бажає побачити більше варіантів, він може натиснути кнопку, яка перенаправить його до наступного розділу з інтерактивною картою. При наведенні на карту мишкою з'являється підказка з назвою міста, а при натисканні відкривається вікно, що містить короткий опис страви та фотографію. Користувач може натискати на кнопку в цьому вікні, щоб перейти на сторінку, присвячену цьому місту.

Детальний дизайн сторінки можна побачити на рис. 6.3.

Рисунок 6.3 – Карта міст

Для переходу на другу сторінку достатньо натиснути на відповідний розділ в верхній частині сайту. На цій сторінці розташовано заголовок, а також плаваючі зображення, які демонструють визначні пам'ятки великих міст України. Прокручуючи сторінку вниз, користувач побачить картки, що містять інформацію про всі міста України, з можливістю вибору того, чия пам'ятка привернула найбільшу увагу. При натисканні на будь-яке місто, користувач буде перенаправлений на відповідну сторінку з деталями. Деталі розробленого дизайну цієї сторінки можна на рис. 6.4.

На третій, останній сторінці користувач знайде інформацію про авторів сайту та мету його створення. Для переходу на цю сторінку також можна натискати на відповідний розділ у верхній частині сайту.

На рис. 6.5 зображено оформлення сторінки про автора.

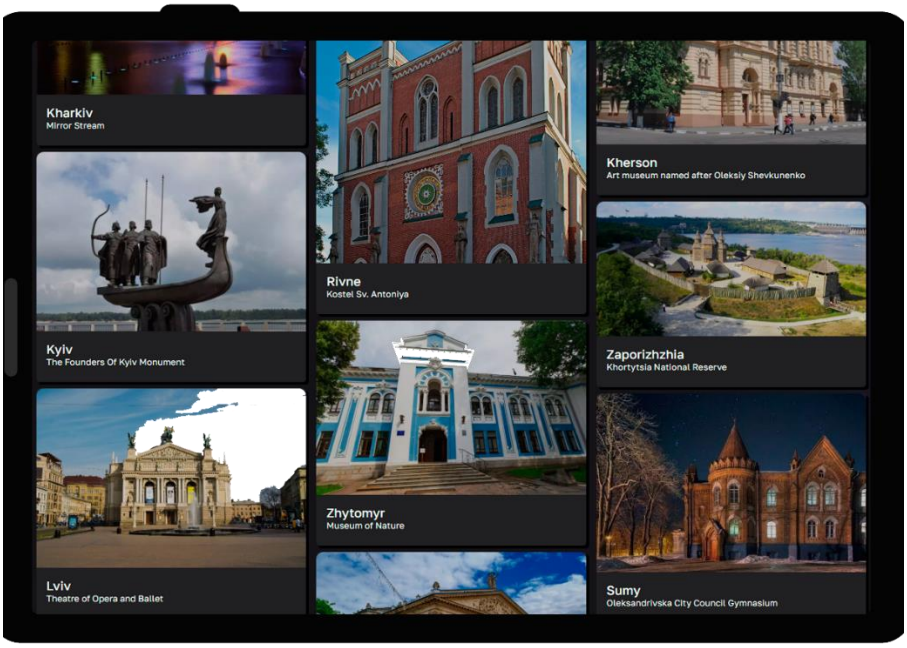


Рисунок 6.4 – Друга сторінка сайту

Рисунок 6.5 – Третя сторінка

Сайт також містить двадцять п'ять окремих сторінок, присвячених кожному місту України. Дизайн цих сторінок є єдиним і складається з двох основних блоків. Перший блок надає користувачеві інформацію про страву, яка є національним надбанням цього міста. У цьому ж блоці розміщено зображення страви, що дозволяє користувачеві побачити, як вона виглядає.

На рис. 6.6 зображено оформлення сторінки про місто.

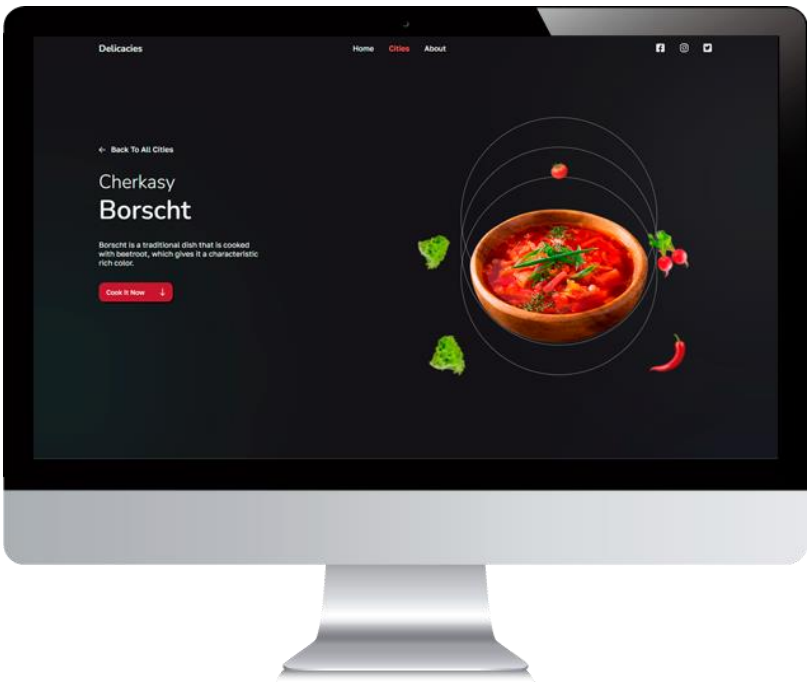


Рисунок 6.6 – Сторінка сайту

При натисканні на кнопку, користувач переходить до другого блоку. У ньому розміщено рецепт страви, що включає перелік усіх необхідних інгредієнтів та покрокову інструкцію з приготування, а також вказано час, необхідний для приготування.

Деталі розробленого дизайну цієї сторінки можна знайти на рис. 6.7.

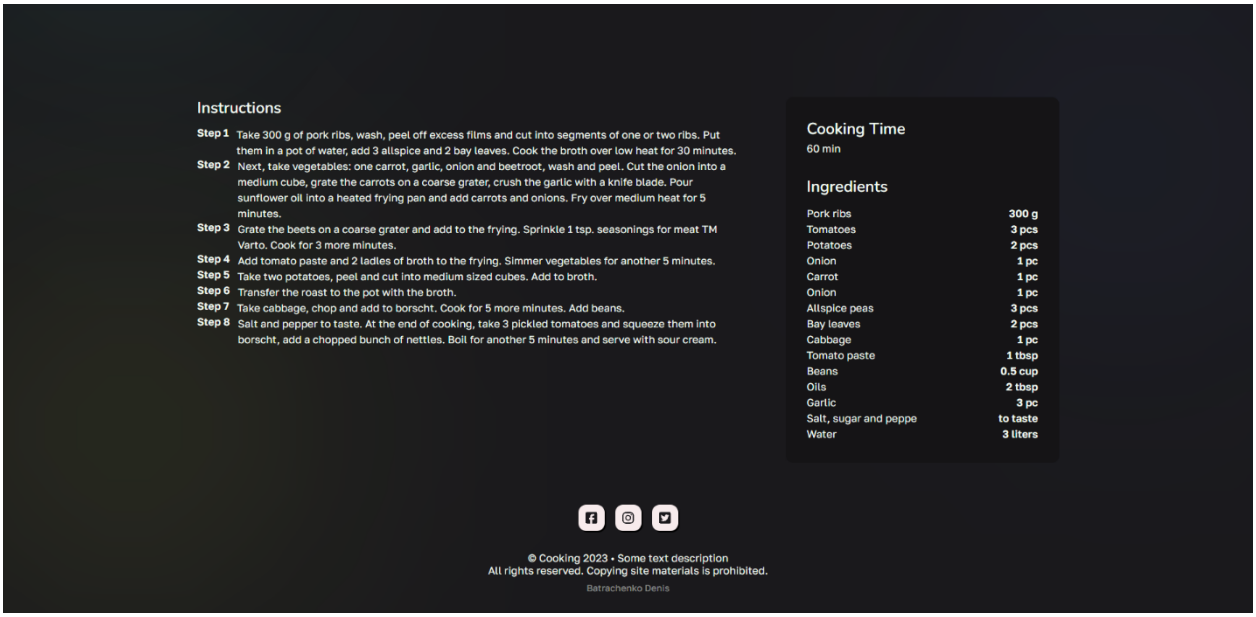


Рисунок 6.7 – Блок сайту

6.3 Наповнення контентом сторінок сайту

Усі блоки на сайті побудовані з використанням семантичних тегів, таких як header, nav, main, section, footer.

Header – це елемент, що зазвичай використовується на початку сторінки та містить вступні або навігаційні елементи. На цьому сайті в ньому знаходяться назва сайту, навігаційна панель, яка дозволяє переходити між сторінками, а також список соціальних мереж.

Nav – тег, що використовується для визначення окремої секції документа, яка призначена для навігаційних посилань. Ці посилання направляють користувача на всі сторінки сайту.

На рис. 6.8 зображено оформлення навігаційного блоку.

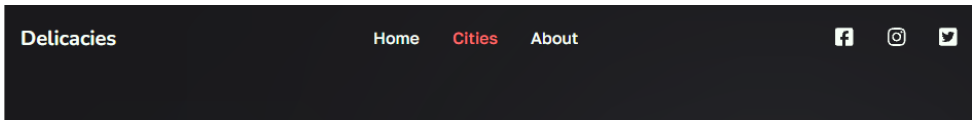


Рисунок 6.8 – Навігаційних блок

Main – елемент, який відповідає за основний контент документа або сторінки. Він містить вміст, що безпосередньо стосується головної теми документа або розвиває її.

Section – тег, призначений для створення автономних розділів у HTML-документі. Це самостійний блок контенту, який не може бути більш чітко визначений іншими семантичними елементами. Зазвичай кожен розділ містить заголовок, що допомагає ідентифікувати та описати його вміст. На цьому сайті більшість блоків є секціями, які включають заголовок, короткий текстовий опис і зображення.

На рис. 6.8 зображено оформлення блоку Section.

Footer – елемент, що використовується для створення нижнього колонтитулу (футера або підвалу) для найближчого секційного контенту або кореневого елемента секції. Футер зазвичай містить інформацію про авторські

права, посилання на соціальні мережі та контактні дані або інформацію про автора сайту.

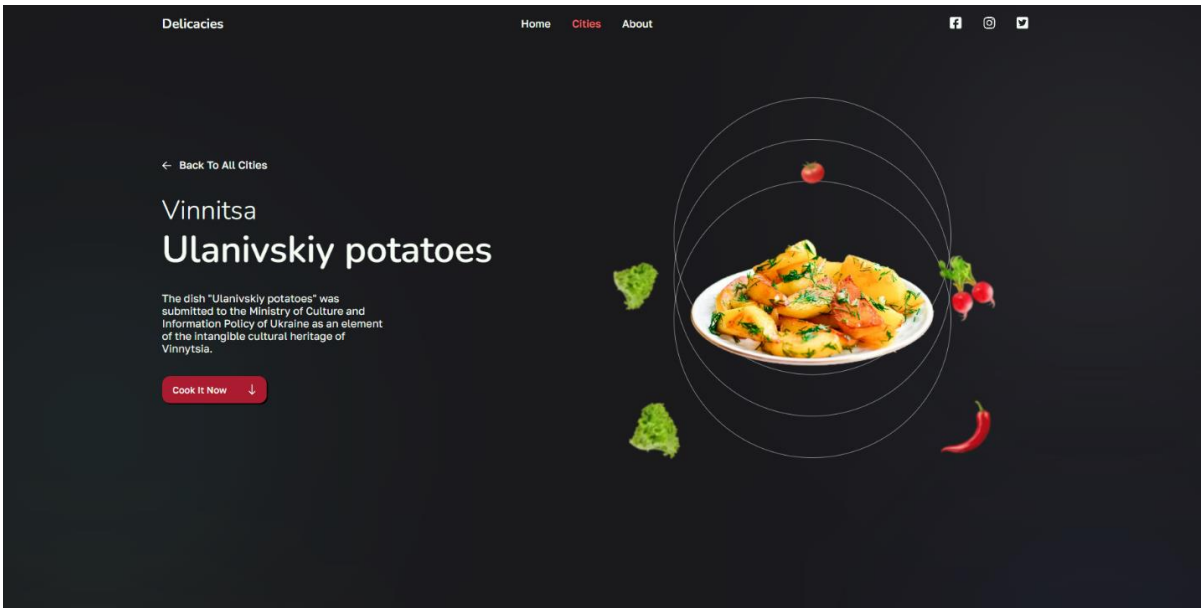


Рисунок 6.9 – Блок Section

На рис. 6.10 зображено оформлення підвалу сайту.

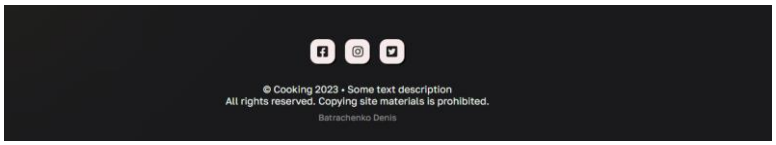


Рисунок 6.10 – Footer

6.4 Реалізація спеціальних функцій

Для дослідження та реалізації адаптивної верстки сайту обрано дві популярні CSS-бібліотеки: Tailwind CSS і Bootstrap. Ці бібліотеки широко використовуються у веброзробці завдяки своїм можливостям швидкого створення адаптивних інтерфейсів, але вони суттєво відрізняються за підходами до роботи зі стилями та дизайном.

Tailwind CSS – це утилітарна CSS-бібліотека, яка пропонує набір низькорівневих класів для безпосередньої стилізації елементів. Tailwind

дозволяє створювати унікальні дизайни без потреби в написанні власного CSS або використанні готових компонентів. Основні особливості:

– утилітарний підхід: стилі задаються безпосередньо в HTML через набір класів (наприклад, bg-blue-500, text-lg, flex);

– висока кастомізація: змінні, які використовуються в Tailwind, можна гнучко налаштовувати через файл конфігурації tailwind.config.js;

– механізм Tree Shaking: видаляє всі невикористані класи з фінального CSS-файлу, що значно зменшує його розмір;

– адаптивність: підтримка мобільного першого підходу завдяки використанню медіакласів (sm:, md:, lg:, xl:).

Bootstrap – це популярний CSS-фреймворк, який надає готові компоненти для створення адаптивних інтерфейсів. Bootstrap використовує модульну структуру та дозволяє швидко створювати функціональні інтерфейси завдяки попередньо визначеним стилям і скриптам. Основні особливості:

– готові компоненти: включає грід-систему, кнопки, форми, модальні вікна та інші елементи;

– мобільний перший підхід: стилі оптимізовані для мобільних пристроїв і масштабуються для більших екранів;

– простота у використанні: підходить для швидкого створення стандартних веб інтерфейсів завдяки базовим стилям і налаштуванням;

– додатковий функціонал: містить інтегровані компоненти JavaScript для додавання інтерактивності (наприклад, випадаючі меню або каруселі).

Сайт був реалізований із використанням обох бібліотек для оцінки їх ефективності в реальних умовах. Зокрема, функціонал сайту, його адаптивність, продуктивність і можливість кастомізації перевірялися на основі трьох основних сторінок:

– головна сторінка;

– сторінка рецептів;

– контактна сторінка.

Впровадження Tailwind CSS на сайт.

Функціональність та можливості:

– використано утилітарні класи для швидкої стилізації елементів. Наприклад, кнопки оформлені за допомогою класів px-6 py-3 bg-blue-500 text-white font-semibold rounded-lg hover:bg-blue-600;

– вся стилізація визначена безпосередньо в HTML-коді, що дозволило уникнути написання окремих CSS-файлів;

– за допомогою конфігураційного файлу tailwind.config.js додано кастомні кольори та шрифти, щоб відобразити традиційний стиль української культури.

Адаптивність та кросбраузерність:

– завдяки медіакласам (sm, md, lg, xl) сторінки адаптовані для коректного відображення на екранах різного розміру. Наприклад, секції із рецептами автоматично змінюють кількість колонок залежно від ширини екрана (grid-cols-1 md:grid-cols-2 lg:grid-cols-3);

– кросбраузерність забезпечується стандартами CSS, що підтримуються Tailwind CSS, і перевірена на браузерах Google Chrome, Mozilla Firefox, Safari та Microsoft Edge.

Швидкість та продуктивність:

– завдяки механізму Tree Shaking фінальний CSS-файл містить лише ті класи, які використовуються на сайті. Це зменшило розмір файлу до 30 кілобайтів, що значно покращило продуктивність сайту;

– всі стилі були автоматично оптимізовані при складанні проєкту.

Впровадження Bootstrap на сайт.

Функціональність та можливості:

– стилізація елементів виконана через готові компоненти Bootstrap. Наприклад, для кнопок використано клас btn btn-primary, а для навігаційного меню – компонент navbar;

– для додавання модальних вікон і каруселей використано стандартні JavaScript-компоненти Bootstrap.

Адаптивність та кросбраузерність:

– грід-система Bootstrap (row, col-md-4) забезпечила автоматичне адаптування контенту до різних розмірів екранів;

– адаптивність реалізована через класи, наприклад, d-flex, flex-column, flex-md-row;

– сумісність із браузерами перевірена та забезпечена завдяки інтеграції стандартів Bootstrap.

Швидкість та продуктивність:

– Bootstrap містить багато стилів і компонентів, які не використовувалися на сайті, через що фінальний CSS-файл мав розмір 140 кілобайтів;

– хоча компоненти JavaScript спрощують реалізацію інтерактивності, вони додали вагу до кінцевого проєкту.

Tailwind CSS дозволив створити легший, більш кастомізований дизайн із мінімальним розміром файлу стилів. Утилітарний підхід до стилізації прискорив процес розробки, але потребував глибшого розуміння системи.

Bootstrap забезпечив швидке створення стандартних елементів інтерфейсу завдяки готовим компонентам. Однак обмеженість кастомізації вимагала додаткових зусиль для створення унікального дизайну, а більший обсяг стилів негативно вплинув на продуктивність.

Обидві бібліотеки успішно впроваджені на сайт і продемонстрували свої переваги та недоліки. Tailwind CSS виявився ефективнішим у створенні унікального дизайну з оптимізованими стилями, тоді як Bootstrap забезпечив швидшу розробку типових компонентів, але створив додаткове навантаження на сайт через більший розмір стилів.

Цей розділ аналізує, як обидві бібліотеки можуть використовуватися залежно від завдань і вимог до вебпроєкту, забезпечуючи надійну базу для подальшого порівняння їхніх характеристик.

6.5 Тестування сайту

Тестування є ключовим етапом у розробці вебсайту, який дозволяє оцінити його відповідність вимогам, стабільність роботи та відповідність поставленим критеріям. У рамках роботи було проведено тестування сайту, реалізованого з використанням бібліотек Bootstrap і Tailwind CSS, за трьома основними критеріями: функціональність та можливості, адаптивність і кросбраузерність, швидкість та продуктивність.

Основною метою тестування було визначення, наскільки сайт відповідає сучасним стандартам веброзробки, а також порівняння ефективності реалізації з використанням бібліотек Bootstrap і Tailwind CSS. Тестування проводилося з урахуванням реальних умов експлуатації: різні пристрої (десктопи, планшети, смартфони, популярні браузери (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge), повільне інтернет-з'єднання для оцінки продуктивності.

Тестування функціональності передбачало перевірку роботи інтерактивних елементів сайту, таких як меню, кнопки, форми та пошук.

Bootstrap:

– навігаційне меню працювало стабільно, однаково коректно реагуючи на кліки та переходи між сторінками;

– готові компоненти (форми, кнопки) функціонували без помилок, але їхня кастомізація була обмеженою;

– для забезпечення унікальності дизайну довелося додати власні стилі CSS.

Tailwind CSS:

– усі інтерактивні елементи були повністю кастомізовані завдяки утилітарним класам. Наприклад, стилі кнопок і форм створювалися безпосередньо в HTML;

– використання Tailwind дозволило створити більш унікальний інтерфейс, однак це вимагало більше часу на стилізацію елементів.

Результати тестування підтвердили, що обидві бібліотеки забезпечують коректну функціональність

Але Tailwind CSS надає більше можливостей для індивідуалізації інтерфейсу.

Тестування адаптивності проводилося для перевірки коректного відображення сайту на різних екранах (мобільні пристрої, планшети, десктопи).

Bootstrap:

– завдяки вбудованій грід-системі сайт автоматично підлаштовувався під різні розміри екранів;

– всі медіазапити (col-md-4, col-lg-6) працювали коректно, але налаштування складних макетів вимагало додаткового написання стилів;

– кросбраузерність була забезпечена, сайт стабільно працював у всіх популярних браузерах.

Tailwind CSS:

– використання медіакласів (sm:, md:, lg:, xl:) дозволило створити повністю адаптивний дизайн із більшою гнучкістю;

– складні макети були реалізовані швидше завдяки грід- або флексбокс-класам (grid-cols-3, flex, justify-center);

– кросбраузерність також підтверджена: сайт коректно працював у Google Chrome, Mozilla Firefox, Safari та Microsoft Edge.

Результати тестування показали, що обидві бібліотеки забезпечують високий рівень адаптивності, але Tailwind CSS пропонує більше можливостей для налаштування структури сторінки.

Для тестування продуктивності використовувалися інструменти Google PageSpeed Insights та GTmetrix, а також проводилося ручне тестування на повільних інтернет-з'єднаннях.

Bootstrap. Розмір фінального CSS-файлу склав близько 140 кілобайтів, оскільки він включає всі стилі фреймворку, навіть ті, що не використовуються на сайті. Час завантаження сторінки на мобільних пристроях склав у середньому 3,2 секунди. Високий рівень продуктивності досягався завдяки інтеграції з готовими компонентами, але надлишковий CSS збільшував час завантаження.

Tailwind CSS. Завдяки механізму Tree Shaking фінальний CSS-файл мав розмір лише 30 кілобайтів, оскільки в ньому містилися лише стилі, які реально використовувалися на сторінках. Час завантаження сторінки на мобільних пристроях був 2,1 секунди, що на 34% швидше порівняно з Bootstrap. Оптимізація стилів забезпечила кращу швидкість і вищий рейтинг продуктивності в Google PageSpeed Insights.

Результати тестування продемонстрували, що Tailwind CSS значно перевершує Bootstrap у продуктивності завдяки оптимізації використаних стилів.

На основі тестування було зроблено такі висновки:

– Bootstrap забезпечує швидку реалізацію стандартних елементів, але обмежує кастомізацію;

– Tailwind CSS надає більше гнучкості для створення унікальних інтерфейсів;

– обидві бібліотеки забезпечують високий рівень адаптивності та стабільну роботу у різних браузерах. Tailwind CSS надає більше можливостей для налаштування макетів;

– Tailwind CSS забезпечує значно менший розмір файлу стилів і швидше завантаження сторінок, що робить його кращим вибором для оптимізації продуктивності.

Результати тестування підтвердили, що обидві бібліотеки мають свої сильні сторони. Bootstrap ідеально підходить для швидкої розробки стандартних сайтів, а Tailwind CSS – для створення унікальних дизайнів із високою продуктивністю.

7 ОЦІНКА ЯКОСТІ

Оцінка якості бібліотек Bootstrap і Tailwind CSS базується на їх функціональності, можливостях кастомізації, адаптивності, кросбраузерності та швидкості роботи. Обидві бібліотеки є лідерами у сфері веброзробки, але їх філософія, підхід до реалізації стилів та сфера застосування суттєво відрізняються, що впливає на кінцеву якість сайту залежно від потреб розробників.

Функціональність і можливості Bootstrap та Tailwind CSS демонструють різні підходи до стилізації та побудови інтерфейсу. Bootstrap забезпечує розробників готовими компонентами, такими як кнопки, форми, модальні вікна та навігаційні панелі. Ці елементи допомагають швидко створювати типові інтерфейси без потреби у значній кількості додаткових стилів. Це є великою перевагою для новачків і тих, хто прагне заощадити час на розробці. Водночас недоліком Bootstrap є обмеженість кастомізації. Для зміни стандартних стилів часто потрібно писати додатковий CSS-код або редагувати змінні через препроцесори, що створює додаткові труднощі, особливо у випадках, коли необхідно створити унікальний дизайн.

Tailwind CSS, навпаки, використовує утилітарний підхід, який дозволяє задавати стилі безпосередньо у HTML-коді за допомогою спеціальних класів. Це дає розробникам майже повний контроль над виглядом елементів, що робить бібліотеку ідеальним вибором для проєктів, де важливим є унікальний дизайн. Tailwind дозволяє зосередитися на кастомізації з самого початку, уникаючи необхідності перевизначати готові стилі, як це часто трапляється у Bootstrap. Проте такий підхід може виявитися складнішим для новачків, адже для ефективного використання бібліотеки потрібно добре розуміти структуру та принципи роботи класів.

Адаптивність і кросбраузерність у обох бібліотек реалізовані на високому рівні. Bootstrap використовує готову грід-систему, яка автоматично підлаштовує компонування сторінок під різні розміри екранів. Це забезпечує

швидку адаптацію сайту, але знову ж таки обмежує кастомізацію макетів. Tailwind CSS, завдяки медіакласам, таким як sm:, md:, lg:, дозволяє реалізовувати гнучкі макети з точним налаштуванням під кожен розмір екрана. Цей підхід дає більше можливостей для розробників, які прагнуть створити оригінальне компонування сторінок. У плані кросбраузерності обидві бібліотеки забезпечують стабільну роботу у популярних браузерах, таких як Google Chrome, Mozilla Firefox, Safari та Microsoft Edge, і тут суттєвої різниці між ними немає.

Швидкість та продуктивність є однією з найважливіших складових оцінки якості. Bootstrap має більший розмір CSS-файлів, оскільки включає всі стилі та компоненти, навіть ті, що не використовуються в конкретному проєкті. Це може призвести до збільшення часу завантаження сайту, особливо на мобільних пристроях із повільним інтернетом. Tailwind CSS, завдяки механізму Tree Shaking, генерує фінальний CSS-файл, що містить лише ті класи, які фактично використовуються на сторінках. Це значно зменшує розмір стилів і прискорює завантаження сайту. У тестах Tailwind CSS продемонстрував кращу продуктивність, особливо на сайтах з великою кількістю сторінок, де оптимізація ресурсів є критично важливою.

Щодо підсумкової оцінки, слід зазначити, що обидві бібліотеки мають свої сильні сторони, але їх застосування залежить від типу проєкту. Bootstrap краще підходить для швидкої розробки стандартних сайтів, де важлива швидкість реалізації, але не потрібен унікальний дизайн. Це робить його ідеальним для створення корпоративних вебсайтів або внутрішніх панелей керування, де універсальність і стабільність важливіші за кастомізацію. Tailwind CSS, своєю чергою, виявляється більш потужним інструментом для створення індивідуальних рішень з високим рівнем кастомізації. Його переваги особливо проявляються у проєктах, де важлива продуктивність, унікальність дизайну та адаптивність під потреби клієнта.

На основі проведеного аналізу можна зробити висновок, що Tailwind CSS є кращим вибором для розробників, які прагнуть створити сучасні,

унікальні, продуктивні та легко масштабовані вебпроєкти. Однак у випадках, коли необхідно швидко реалізувати типові інтерфейси, Bootstrap залишається незамінним завдяки своїй простоті та готовим рішенням. Таким чином, вибір бібліотеки залежить від цілей і специфіки проєкту, але з точки зору продуктивності, гнучкості та адаптивності Tailwind CSS демонструє більш високий рівень якості.

Таблиця 7.1 – Таблиця результатов оцінки якості

Критерій	Bootstrap	Tailwind CSS
Швидкість завантаження сторінки	2.8 с (±0.1 с)	2.4 с (±0.1 с)
Функціональність	4.6	4.8
Адаптивність	4.7	4.9
Кросбраузерність	4.5	4.8
Продуктивність	4.2	4.6

Bootstrap продемонстрував хороші результати у всіх критеріях, проте Tailwind CSS виявився трохи швидшим у завантаженні сторінок, забезпечуючи середній час у 2.4 секунди проти 2.8 секунди у Bootstrap. Tailwind CSS також перевершив Bootstrap у функціональності, адаптивності та кросбраузерності завдяки своїй модульній структурі, що дозволяє більш точний контроль над стилізацією. У категорії продуктивності Tailwind CSS також випереджає, оскільки його підхід до роботи з класами знижує складність коду.

8 ЕКОНОМІЧНА ЧАСТИНА

8.1 Характеристика науково-дослідної роботи

Метою даного розділу є економічне обґрунтування витрат на проведення науково-дослідної роботи (НДР), в межах якої передбачається дослідження використання інструментальних засобів для адаптивної верстки сайтів, оцінка їх впливу на зменшення витрат часу і коштів при розробці веб-продуктів.

У роботі досліджено використання систем контролю версій при розробці мультимедійних проєктів, розглянуто проблеми які виникають при використання загальних рішень та розроблено і реалізовано альтернативну методику збереження версій, проведено розрахунок економічної ефективності. Під час такого обґрунтування буде здійснено: розрахунок трудовитрат та заробітної плати працівникам, розрахунок одноразових витрат і прибутку, оцінку результатів НДР.

Етапи виконання НДР:

– аналіз ринку інструментальних засобів адаптивної верстки;

– визначення алгоритму реалізації проєкту;

– дослідження використання інструментальних засобів для адаптивної верстки сайтів;

– тестування інструментів;

– розробка прототипу сайту;

– аналіз та оцінка результатів НДР.

8.2 Етапи виконання НДР, їх трудомісткість та заробітна плата

Умовно НДР можна розділити на такі етапи: підготовчий, основний і заключний.

Для підготовки було сформульовано основні цілі дослідження, а також визначено перелік задач, які необхідно вирішити для досягнення результату.

Проведено порівняльний аналіз популярних інструментів, таких як Bootstrap, Figma тощо, з метою виявлення їх переваг і недоліків. Сформовано критерії для оцінки інструментів, які включають продуктивність, зручність використання, економічну ефективність, підтримку спільноти та сумісність з різними платформами.

В основній частині було проведено тестування інструментів на основі визначених критеріїв. Це дозволило оцінити їх ефективність у реальних умовах розробки. На основі обраного інструменту створено адаптивний прототип сайту, що відповідає сучасним вимогам зручності та функціональності.

У заключній частині проведено аналіз отриманих даних, складено фінальний звіт та зроблено висновки щодо ефективності використання обраного інструменту для адаптивної верстки. Виконано розрахунок витрат на виконання НДР, а також оцінено економічну ефективність впровадження результатів роботи у практичну діяльність.

Дану роботу виконували 3 фахівці: ілюстратор, інженер-дослідник, програміст. Середня заробітна плата ілюстратора за версією сайту dou.ua становить 30 000,00 грн, інженер-дослідник – 110 000,00 грн, програміст – 40 000,00 грн.

Проведемо розрахунок трудовитрат і заробітної плати виконавців робіт. Середньоденна заробітна плата виконавця робіт ($З_{ср.дн.}$):

$$З_{ср.дн.} = \frac{З_{ср.міс.}}{n}, \tag{8.1}$$

де $З_{ср.дн.}$ – середньомісячна зарплата виконавця роботи;

n – число робочих днів у місяці, (n = 22).

Підставивши дані до (8.1), отримаємо середньоденну заробітну плату ілюстратора у розмірі 1363,63 грн, інженера-дослідника – 5000,00 грн, програміста – 1818,18 грн.

Етапи виконання НДР, перелік і зміст робіт, трудомісткість їх виконання, заробітна плата виконавців робіт представлені у таблиці 8.1.

Таблиця 8.1 – Розрахунок трудовитрат і заробітної плати виконавців робіт

	Перелік робіт	Кіль-кість вико-навців	Посада виконавця	Трудо-місткість робіт, люд.-днів	Середньо-денна заро-бітна плата, грн	Сума заро-бітної плати, грн
1	Аналіз ринку інструментальних засобів	1	Інженер-дослідник	5	5000,00	25000,00
2	Розробка критеріїв оцінки інструментів	1	Інженер-дослідник	3	5000,00	15000,00
3	Тестування та вибір інструменту	1	Програміст	6	1818,18	10909,08
4	Розробка прототипу сайту	1	Програміст	5	1818,18	9090,90
5	Створення графічних елементів для прототипу	1	Ілюстратор	3	1363,63	4090,89
6	Аналіз результатів проведення роботи	1	Інженер-дослідник	2	5000,00	10000,00
7	Підготовка фінального звіту	1	Інженер-дослідник	2	5000,00	10000,00
	Усього:			26		84090,87

Таким чином, сума витрат на заробітну плату в межах виконання НДР складе 84090,87 грн.

8.3 Розрахунок одноразових витрат на розробку НДР

Калькуляція собівартості розраховується відповідно до існуючих нормативних актів України. До складу калькуляції входять такі статті витрат:

- матеріальні витрати;
- витрати на оплату праці;
- єдиний соціальний внесок;
- амортизація основних засобів (вартість машинного часу);
- витрати на спожиту електроенергію;
- інші витрати.

Витрати на оплату праці розраховуються, виходячи з необхідного для виконання робіт складу й кількості працівників, а також із середньомісячної заробітної плати. Відповідно до проведених розрахунків витрати на оплату праці виконавців роботи дорівнюють 84090,87 грн.

Єдиний соціальний внесок (ЄСВ) є об'єднаним внеском, який регулярно і обов'язково сплачується до системи загальнообов'язкового державного соціального страхування. Цей внесок має на меті забезпечити соціальний захист у випадках, визначених законодавством, та гарантувати право на страхові виплати для застрахованих осіб та членів їхніх сімей у рамках різних видів державного соціального страхування.

Ставка єдиного соціального внеску складає 22 % від витрат на оплату праці, тобто розмір ЄСВ дорівнює 18499,99 грн.

Витрати на електроенергію розраховуються, виходячи зі споживаної потужності пристрою і тарифу на електроенергію. У даному випадку передбачається використання 3-ох комп'ютерів потужністю 0,6 кВт/год. Вартість однієї кВт/год електроенергії прийнято у розмірі 4,32 грн. Витрати на використану обладнанням електроенергію (Ве) розраховуються за формулою:

$$B_e = M \cdot t \cdot T_{кВт}, \tag{8.2}$$

B_e = (0,6×96×4,32) + (0,6×88×4,32) + (0,6×24×4,32) = 539,14 грн.

де М – потужність устаткування, тобто кількість енергії, споживаної за одиницю часу (кВт/година);

t – кількість годин використання устаткування за період проведення науково-дослідницької роботи;

$T_{кВт}$ – тариф, тобто вартість використання 1 кВт електроенергії.

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 використовується 254 робочих дні. Отже амортизація основних засобів розраховується за формулою:

$$AB = \sum_{k=1}^{L} \frac{BO_k}{TE_k} \times T \,, \tag{8.3}$$

$$AB = \frac{90000{,}00 \times 26}{762} = 3070{,}87 \text{ грн.}$$

де АВ – сума амортизаційних відрахувань, нарахованих під час проведення НДР;

BO_k – вартість основних засобів k-го виду;

TE_k – термін експлуатації основних засобів k-го виду, днів;

Т – термін НДР, днів;

L – кількість видів обладнання.

При виконанні НДР застосовувалися 3 комп'ютери вартістю по 30000,00 грн кожен.

До інших статей витрат відносяться такі:

– адміністративні витрати: (водопостачання, водовідведення, освітлення, опалення), які прийнято у розмірі 20 % від витрат на оплату праці;

– вартість оплати послуг зв'язку.

Адміністративні витрати складатимуть 20 % від витрат на оплату праці, тобто дорівнювати 16818,17 грн.

Вартість оплати послуг зв'язку, а саме Інтернет – 220,00 грн за 26 днів виконання НДР.

За період виконання НДР витрати на відрядження, аутсорсинг, інформаційні послуги та маркетингові заходи не мали місця. Протягом розробки матеріальні витрати також не мали місця.

Для виконання НДР використовувалося ряд програмного забезпечення та онлайн платформ. Для створення графічних елементів для проєкту використовувалась програма Figma, для розробки проєкту використовувався Bootstrap Studio. Для перевірки роботи НДР був куплений хостинг для тестування. Всі розрахунки на використані сервіси та програми наведено в таблиці 8.2.

Результати розрахунку кошторису витрат, тобто одноразових витрат, на виконання НДР, наведені у таблиці 8.2.

Таблиця 8.2 − Кошторис витрат на розробку НДР

№ з/п	Стаття витрат	Сума, грн
1	Заробітна плата	84090,87
2	Єдиний соціальний внесок (22 % від п.1)	18499,99
3	Матеріальні витрати	-
4	Амортизація основних засобів	3070,87
5	Витрати на спожиту електроенергію	539,14
6	Інші витрати, у тому числі:	
6.1	Ліцензія на Figma (3 міс.)	2100,00
6.2	Ліцензія Bootstrap Studio	1500,00
6.3	Хостинг для тестування(3 міс.)	600,00
6.4	адміністративні витрати (20 % від п.1)	16818,17
6.5	вартість послуг інтернету	220,00
6.6	вартість послуг сервісу ChatGPT	700,00
	Усього витрати на розробку (*Вр*)	125238,12

Таким чином, кошторис витрат на виконання даної НДР складає 128139,04 грн.

8.4 Оцінка результатів науково-дослідної роботи

Результат науково-дослідної роботи (НДР) – це наслідок послідовності дій, виконаних під час дослідження, що дозволяє оцінити ефективність використання інструментів адаптивної верстки (Bootstrap і Tailwind CSS). У рамках дослідження результат оцінюється за трьома ключовими характеристиками: швидкістю розробки, розміром CSS-файлу та можливостями кастомізації. Швидкість розробки показує, як швидко можна створити адаптивний макет за допомогою кожного інструменту. Використання Tailwind CSS дозволяє розробникам зменшити час створення адаптивного макету завдяки гнучкості утилітарного підходу. Це на 15 хвилин швидше, ніж при використанні Bootstrap, де значна частина часу витрачається на кастомізацію готових компонентів.

Розмір CSS-файлу впливає на швидкість завантаження сторінки. У випадку Tailwind CSS розмір стилів значно менший завдяки застосуванню механізму Tree Shaking, який видаляє невикористані класи з кінцевого файлу. Зменшення обсягу файлу на 110 кілобайт позитивно впливає на продуктивність сайту. Bootstrap генерує більший обсяг CSS через включення багатьох заздалегідь визначених стилів, які не завжди використовуються.

Кастомізація дизайну є важливим показником для створення унікального вигляду вебсайту. Tailwind CSS надає більше можливостей для гнучкого налаштування стилів завдяки використанню змінних у файлі конфігурації, тоді як Bootstrap передбачає обмежені варіанти змін у межах своїх готових компонентів.

Вплив на продуктивність у реальних умовах також враховувався. Використання Tailwind CSS забезпечує швидше завантаження сторінки, оскільки сайт генерує мінімальний обсяг CSS, необхідний для відображення конкретного інтерфейсу. У реальних тестах Tailwind CSS показав стабільно кращі результати на мобільних пристроях зі слабкими технічними характеристиками. Bootstrap, хоча і має високу стабільність, створює більші файли, що може сповільнити завантаження сторінки при повільному інтернет-з'єднанні.

Порівняння інструментів адаптивної верстки Bootstrap та Tailwind CSS за ключовими характеристиками наведено у таблиці 8.3.

Таблиця 8.3 – Порівняння характеристик Bootstrap і Tailwind CSS

Показник	Bootstrap	Tailwind CSS
Час розробки (хв)	60	45
Розмір CSS-файлу (кб)	140	30
Кількість кастомізацій (од.)	5	10

Результат від впровадження НДР визначається за формулою:

$$\Delta P_j = \left| Хб_j - Хн_j \right|, \quad (8.4)$$

де ΔP_j – покращення характеристики;

$Хб_j$ – базове значення j-ої характеристики;

$Хн_j$ – нове значення j-ої характеристики.

Результати розрахунків:

$$\Delta P_{\text{Час розробки}} = |60\text{-}45| = 15 \text{ хв},$$
$$\Delta P_{\text{Розмір CSS–файлу}} = |140\text{-}30| = 110 \text{ кБ},$$
$$\Delta P_{\text{Кількість кастомізацій}} = |5\text{-}10| = 5 \text{ од.}$$

На основі цих розрахунків можна стверджувати, що використання Tailwind CSS дозволяє зменшити час розробки на 15 хвилин, зменшити розмір кінцевого CSS-файлу на 110 кб, а також збільшити кількість кастомізацій на 5.

Отже, зменшення часу на 15 хвилин дозволяє розробникам виконувати більше проєктів у той самий період, зменшення на 110 кілобайт сприяє кращій швидкості завантаження сторінки та позитивно впливає на SEO-рейтинги, а зростання можливостей кастомізації на 5 пунктів дає змогу створювати більш унікальні рішення, що позитивно впливає на конкурентоспроможність вебсайту.

5.5 Визначення економічної ефективності результатів науково-дослідної роботи

Економічна ефективність НДР визначається шляхом порівняння витрат на реалізацію розробки з отриманими результатами. Основним показником економічної ефективності є коефіцієнт економічної ефективності, який обчислюється за формулою:

$$K_{ев} = \frac{\Delta P_j}{B_p}, \tag{8.5}$$

де B_p – витрати (кошторисна вартість) на виконання НДР, грн;

$K_{ев}$ – коефіцієнт «ефект-витрати», який відбиває, наскільки кожна гривня витрат НДР змінює j-ту характеристику досліджуваного процесу.

Розрахунки коефіцієнта «ефект-витрати» для кожного показника:

$$\text{K}_{\text{ев (Час розробки)}} = \frac{15}{128139{,}04} \times 100\ \% = 0{,}012\ \%,$$

$$\text{K}_{\text{ев (Розмір CSS–файлу)}} = \frac{110}{128139{,}04} \times 100\ \% = 0{,}086\ \%,$$

$$\text{K}_{\text{ев (Кількість кастомізацій)}} = \frac{5}{128139{,}04} \times 100\ \% = 0{,}004\ \%.$$

Таким чином, Tailwind CSS демонструє вищу економічну ефективність порівняно з Bootstrap завдяки меншим витратам часу на розробку, зменшенню обсягу коду та розширеним можливостям налаштування, а кожна гривня витрат на розробку забезпечує поліпшення даних характеристик на 0,012 %, 0,086 % та 0,004 % відповідно.

Впровадження результатів НДР дозволяє розробникам економити ресурси, покращувати продуктивність та створювати більш конкурентоспроможні рішення. Це підтверджує доцільність використання Tailwind CSS для реалізації адаптивних вебсайтів у реальних умовах. У цілому роботу можна вважати ефективною та такою, що має науковий і технічний рівень.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено дослідження інструментальних засобів адаптивної верстки сайтів, зокрема розглянуто методи, інструменти та технології, які забезпечують коректне відображення веб-сайтів на різних пристроях і екранах. Метою роботи було вивчити найбільш ефективні підходи до створення адаптивних і респонсивних сайтів, що гарантують зручність користування для відвідувачів, незалежно від типу їх пристрою.

В результаті дослідження були вивчені основні принципи адаптивного дизайну, а також проведений аналіз інструментальних засобів для реалізації таких підходів. Особлива увага приділялася популярним CSS-фреймворкам, таким як Bootstrap, а також методам використання медіа-запитів і CSS Grid для забезпечення гнучкості верстки.

Робота також включала розробку рекомендацій щодо вибору оптимальних інструментів для адаптивної верстки та створення практичних кроків для реалізації таких рішень на сайті. Оцінка ефективності застосованих технологій та тестування на різних пристроях підтвердили їх коректну роботу і високу якість реалізації адаптивного дизайну.

Завдяки цим результатам було досягнуто мети створення універсальних рекомендацій, які допоможуть веб-розробникам ефективно застосовувати інструменти адаптивної верстки для створення зручних і функціональних веб-сайтів. Крім того, проведена економічна оцінка показала, що використання адаптивного дизайну може позитивно вплинути на ефективність веб-проектів, зокрема, у контексті оптимізації витрат і покращення взаємодії з користувачами на різних пристроях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Honcharenko D. Як створити сайт з нуля? URL: https://hostpro.ua/blog/ua/how-to-build-a-website (дата звернення: 20.12.2024).

2. Documenting web technologies, including CSS, HTML, and JavaScript URL: https://developer.mozilla.org/ru/ (дата звернення: 20.12.2024).

3. Розробка веб-додатків: основні етапи URL: https://www.purrweb.com/blog/kak-sozdat-veb-prilozhenie/ (дата звернення: 20.09.2024).

4. Етапи створення веб-сайтів URL: https://webtune.com.ua/statti/web-rozrobka/etapy-stvorennya-veb-sajtiv/ (дата звернення: 20.12.2024).

5. Морган Н. JavaScript для дітей: навч. посіб. Львів: Вид-во Старого Лева, 2021. 408 с.

6. Фрімен Е. Head First. Вивчаємо HTML, XHTML та CSS: навч. посіб. Київ: Фабула, 2019. 720 с.

7. Фрімен Е., Робсон Е. Head First. Програмування на JavaScript: навч. посіб. Київ: Фабула, 2022. 672 с.

8. Ткаченко В.П., Силантьєв В.С. Аналіз життєвого циклу розробки проекту WEB системи // Поліграфічні, мультимедійні та web-технології. 2023. Т. 1. С. 126-129.

9. Овсянникова Н.М. Про задачу розробки Internet-ресурсу здорового харчування // Радіоелектроніка та молодь у XXI столітті. 2023. Т. 6. Ч. 2. С. 206-207.

10. Ткаченко В.П., Тимченко Є.М. Вибір і обґрунтування критеріїв оцінки якості інтерфейсу сайту // Поліграфічні, мультимедійні та web-технології. 2019. Т. 1. С.136-137.

11. Tailwind Labs // Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 20.12.2024).

12. Bootstrap Team // Bootstrap Documentation. URL: https://getbootstrap.com/docs (дата звернення: 20.12.2024).

13. Utility-First CSS: What Makes Tailwind CSS Different. URL: https://css-tricks.com (дата звернення: 20.12.2024).