ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти Бакалавр

на тему: «Розроблення вебсайту для бізнесу з використанням фреймворків JavaScript»

Виконав: здобувач вищої освіти
за освітньо-професійною
програмою Інформаційні
управляючі системи спеціальності
126 Інформаційні системи та
технології
ступеня вищої освіти Бакалавр
групи 126ІСТбд41
Сосновський А.С.
Керівник: Копішинська О.П.
Рецензент: Брикун О.М.

Полтава – 2024 року

ВСТУП

З появою сучасного інтернету вже більше 30 років вебсайти дають змогу підприємцям будувати власний бізнес, вирішувати безліч комплексних задач майже у всіх сферах комерції, від інформаційного пошуку й навчання до різнопланового бізнесу, фінансової діяльності та багато іншого.

Актуальність теми кваліфікаційної роботи пов'язана з великим попитом на послуги інтернет-комерції у сфері продажу й обслуговування автотранспорту. Створення інтернет-магазину для бізнесу з продажу автомобілів із використанням новітніх вебтехнологій, які також невпинно розвиваються, покращує користувацький досвід та оптимізує пошук необхідних товарів. Це актуально в контексті зростання попиту на зручні та безпечні онлайн-платформи для покупки автомобілів. Великий сегмент вебсайтів в інтернеті мають обмежені можливості та не в повній мірі відповідають новим вимогам щодо вебсайтів.

Метою кваліфікаційної роботи є розроблення вебсайту, придатного для ведення з обраної предметної області, із використанням сучасних вебтехнологій.

Завдання кваліфікаційної роботи:

– дослідження потреб ринку бізнесу з продажів автомобілів та забезпеченості комерційними вебсайтами;

– розроблення інтерфейсу майбутнього комерційного вебсайту;

– вибір стеку вебтехнологій та розроблення функціонального вебсайту зі зручним інтерфейсом для користувачів, а також забезпечення безпеки та захисту персональних даних клієнтів.

Об'єкт дослідження – процес розроблення вебсайту для бізнесу на основі поєднання сучасних вебтехнологій.

Предмет дослідження – методологія використання різних вебтехнологій, програмних бібліотек при створенні комерційного вебсайту, зокрема React, NodeJS та Styled Component.

Методи дослідження: інформаційно-пошуковий, аналітичний, емпіричний, порівняльний, прикладного програмування, тестування, графічний.

Інформаційна база кваліфікаційної роботи : ресурси мережі інтернет, наукові публікації та популярна література, а також сегмент інтернет-магазинів з продажу автомобілів.

Апробація результатів роботи здійснювалася на студентських наукових конференціях у вигляді доповідей, презентації та публікації тез доповідей. були апробовані шляхом тестування вебсайту та зворотнього зв'язку від користувачів,

Практична значущість роботи: розроблено готовий вебсайт, який був розміщений на безкоштовному хостингу протягом пільгового періоду на сервісі https://carsaleo.000webhostapp.com/. Відбулося тестування та реалізовані заходи вебоптимізації. Вебсайт готовий до використання в реальному бізнесі.

Робота складається зі вступу, трьох розділів, висновків та списку літератури. Обсяг роботи становить 52 сторінки, 14 рисунків, 3 таблиць.

РОЗДІЛ 1
АНАЛІЗ ТА ОБРАННЯ ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБСАЙТУ

1.1 Відмінності та взаємодія складових Front-End і Back-End в розробці сучасних вебсайтів

Первинна технологія розроблення вебсайтів, яка починалася з гіпертекстової розмітки тексту HTML (від англ. Hypertext Markup Language), була розрахована по-перше, на створення інформаційних вебсайтів та реалізацію ідеї прочитання сторінок у браузері [1]. З часом ситуація кардинально змінилася, вебсайти стали повноцінними додатками й на сьогодні маємо індустрію технолгій розробки як зовнішньої частини сайтів – фронтенду (в перекладі з англ. Front-End) так і їх прихованої, серверної, частини – бекенду (переклад з англ. Back-End), яка відповідає за обробку даних і зв'язок із базами даних та іншими додатками [2]. Отже, розробка вебсайту включає два ключові напрямки: фронтенд і бекенд. Фронтенд-розробка зосереджена на створенні інтерфейсу, який користувачі бачать і з яким взаємодіють, включаючи елементи дизайну, анімації та поведінку форм. Бекенд-розробка, у свою чергу, відповідає за серверну частину, логіку обробки даних, бази даних та інтеграцію з іншими сервісами. Обидва напрямки вимагають різних навичок та технологій, але разом вони створюють повноцінний продукт. Важливою частиною процесу є їх взаємодія: фронтенд повинен ефективно комунікувати з бекендом для забезпечення злагодженої роботи сайту [3].

Вебсайт, за визначенням, – це колекція вебсторінок та пов'язаних з ними файлів, які розташовані на вебсервері та доступні через інтернет [4]. Вебсайт може містити різноманітний контент, такий як текст, зображення, відео, аудіо, графіку та інші елементи, які відображаються в браузері користувача. Вебсайти, в залежності від призначення, можуть мати різну структуру та функціональність, включаючи статичні сторінки, динамічний контент, форми взаємодії з користувачем, бази даних, інтерактивні елементи та інше.

Поява розділених технологій для інтерфейсу та функціоналу виникла від потреби в розподілі ролей та вдосконаленні процесу розробки вебдодатків. Перші вебсайти були простими HTML-сторінками, де весь код для відображення і функціональності знаходився в одному файлі. З плином часу вебсайти стали складнішими та більш вимогливими щодо функціональності та інтерактивності. Почали використовуватися скрипти на JavaScript для валідації форм та реалізації простих ефектів, таких як розкриття і приховування блоків ті ін.

Розділений підхід виник як відповідь на цю складність, щоб розділити завдання та обов'язки між розробниками і дизайнерами.

Розділена технологія передбачає розбиття Front-End (інтерфейсу) та Back-End (функціоналу) на окремі складові частини. Front-End відповідає за відображення та інтерактивність користувача, тоді як Back-End займається обробкою даних, виконанням бізнес-логіки та взаємодіє з базою даних. Відповідно, застосовуються пізні технології.

Front-End частина складається з HTML, CSS та JavaScript, і ці технології використовуються для відображення та взаємодії з користувачем. HTML відповідає за структуру сторінки, CSS – за її вигляд, стилізований дизайн, а JavaScript – за динамічні ефекти та взаємодію з користувачем [5]. Крім того, для покращення роботи з CSS можуть використовуватися різні препроцесори, такі як Sass, Less або Stylus.

Back-End частина складається з мов програмування, баз даних та серверного програмного забезпечення. Найпоширеніші мови програмування для Back-End - це Java, Python, Ruby, PHP та JavaScript (з використанням Node.js) [6]. Бази даних використовуються для збереження даних та їхньої організації. Серверне програмне забезпечення, таке як Apache, Nginx або IIS, відповідає за обробку запитів від клієнтів та відправку відповідей [7].

Загалом, структура сучасних вебсайтів може бути досить складною і вимагати від розробників знань та досвіду у різних технологіях. Проте, використання правильного стеку технологій та добре спроєктованої архітектури може допомогти створити потужну та ефективну вебплатформу.

У 2000-х роках з'явилися технології, такі як CSS та AJAX, які дозволили створювати більш складні інтерфейси користувача та динамічно змінювати вміст сторінки без перезавантаження. Також з'явилася можливість створювати вебдодатки з використанням серверної логіки та баз даних, що дозволяє зберігати та обробляти інформацію з більшою ефективністю та безпекою.

Сучасні вебсайти складаються з Front-End та Back-End, які розробляються окремо та з'єднуються за допомогою API. Це дозволяє забезпечити більшу модульність та масштабованість вебдодатків, а також розподіляти завдання між розробниками та забезпечувати більш високу швидкість розробки.

1.2 Базові технології розроблення клієнтської частини вебсайту

Розглянемо більш детально технології клієнтська частини – це частина веброзробки, яка займається розробкою клієнтської (браузерної) частини вебдодатку або сайту. Front-end розробники відповідають за створення інтерфейсу користувача, або UI-дизайну (від англ. Usability Interface) з використанням HTML, CSS та JavaScript [8].

HTML - це мова розмітки, що використовується для створення вебсторінок та їх структурування. Використовуючи HTML, можна визначати різні елементи вебсторінки, такі як заголовки, абзаци, списки, таблиці, зображення та інші, а також встановлювати зв'язки між вебсторінками за допомогою гіперпосилань.

HTML є основним компонентом вебсторінок та є стандартом в інтернеті. Більшість браузерів підтримують HTML, що дозволяє відображати вебсторінки з мінімальними відмінностями на різних пристроях та операційних системах. HTML є однією з базових мов веброзробки та часто використовується в поєднанні з CSS та JavaScript для створення більш складних та інтерактивних вебсайтів.

CSS (Cascading Style Sheets) – це мова опису стилів, яка використовується для задання зовнішнього вигляду вебсторінок. CSS дозволяє відділити структуру

та зміст вебсторінок від їхнього вигляду та оформлення. Завдяки CSS можна відокремити дизайн вебсторінок від коду, що дозволяє зберігати код сторінок більш чистим і легким для редагування [9]. CSS дозволяє задавати такі елементи стилю, як кольори, фон, шрифти, відступи, рамки, анімації та багато іншого. Використовуючи CSS, можна значно поліпшити зовнішній вигляд вебсторінок та зробити їх більш привабливими та зрозумілими для користувачів. На сьогодні все більш популярними є фреймворки CSS – готові комбінації стилів, які значно економлять час та роблять роботу дизайнера більш якісною та еефктивною. Рейтинг популярності топ-5 відомих фреймворків наведено на рис. 1.1 за даними [10].

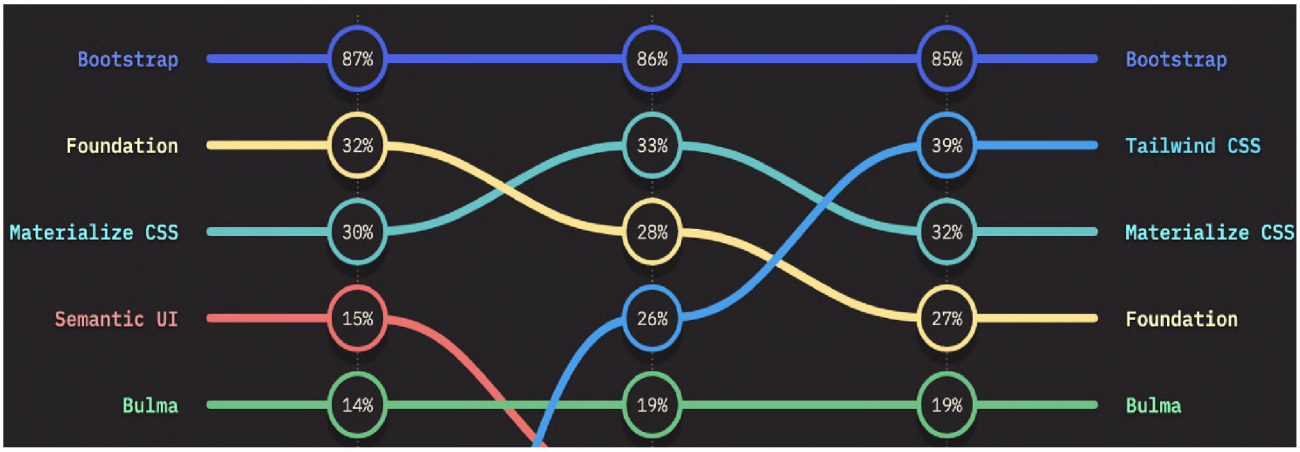

Рисунок 1 1 – Рейтинг популярності відомих CSS-фреймврків за 2020-2022 рр.

JavaScript – мова програмування, яка використовується веброзробниками для створення динамічних і інтерактивних елементів сайту. Існування JavaScript зробило можливим існування вебдодатків – застосунків, в яких оновлення інформації відбувається без оновлення всієї сторінки [К16,11]. У JavaScript фактично немає конкурентів. Вона може бути виконана на зовнішньому (клієнтському) або на серверному боці за допомогою різних фреймворків та бібліотек.

Мова програмування JavaScript використовується програмістами для створення динамічних інтерактивних вебсторінок. JS настільки популярна, що її використання на сайтах сягає майже 98% [12] (рис. 1.2).

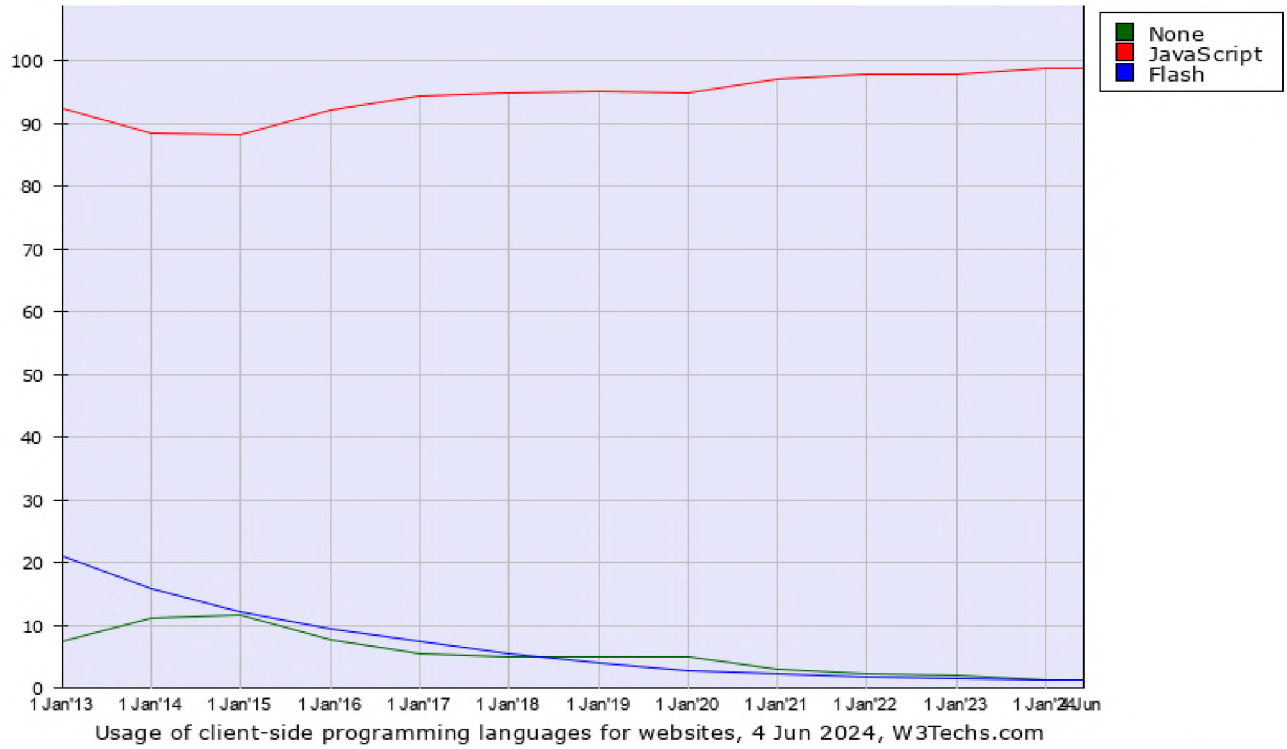

Рисунок 1.2 – Звіт W3Techs про використання JavaScript 2013-2023 рр. [12]

Графік популярності (див. рис. 1.2, верхня лінія) залишається стабільним більше 10 років. Для JavaScript розробки також є багато корисних технологій та інструментів, наприклад фреймворки, такі як React, Angular, Vue.js, які дозволяють створювати складні вебдодатки швидко та ефективно [13].

React – дозволяє розробникам створювати вебсайти та вебдодатки, які можуть швидко та ефективно оновлюватися без перезавантаження сторінки. React був розроблений компанією Facebook і випущений в 2013 р.

React базується на концепції компонентів, що дозволяє розробникам створювати окремі блоки коду, які можуть бути використані повторно в різних частинах вебдодатку. Це робить код більш організованим та легким для зберігання та розуміння.

React також використовує віртуальну DOM (Document Object Model), що дозволяє зменшити кількість змін, які необхідно робити безпосередньо в реальному DOM, що забезпечує більш швидку роботу вебдодатку. Вона має кілька переваг порівняно з іншими бібліотеками, які розглянуто далі [14].

1. Єдина «парасолька»: React є лише бібліотекою для відображення інтерфейсу, а не повноцінним фреймворком. Це означає, що можливо використовувати React разом з будь-яким іншим інструментом або бібліотекою, що надає більшу гнучкість та контроль над вашим проектом.

2. Висока продуктивність: React використовує віртуальну DOM (Document Object Model), що дозволяє реалізувати швидкі та ефективні переробки елементів візуального інтерфейсу. Це дозволяє досягнути високої продуктивності вебдодатків, навіть при великій кількості даних.

3. Модульність: React дозволяє розбити ваш інтерфейс на незалежні компоненти, що підвищує його модульність. Це означає, що можливо повторно використовувати код компонентів, що дозволяє зменшити час розробки та покращити підтримку вашого коду.

4. Легкість навчання: React має просту та логічну структуру, що дозволяє легко навчитися розробці з його використанням. Більшість розробників, які знайомі з JavaScript, можуть швидко вивчити React.

5. Велике співтовариство: React має велику та активну спільноту розробників, яка забезпечує підтримку, допомогу та постійне вдосконалення бібліотеки. Це означає, що можливо швидко знайти відповіді на свої запитання та знайти відповідні рішення для вашого проекту.

6. Гнучкість: React дозволяє змінювати та розширювати його функціонал за допомогою сторонніх бібліотек та плагінів. Це дозволяє збільшити функціональність вашого проекту та забезпечити більшу гнучкість в розробці.

7. Розробка мобільних додатків: React Native - це фреймворк для розробки мобільних додатків з використанням React. Він дозволяє розробляти нативні додатки для iOS та Android, використовуючи звичний для розробників React підхід до розробки інтерфейсу. Це забезпечує зменшення часу розробки та спільний код для різних платформ.

Також варто відзначити, що React є дуже популярною технологією, тому використання його в проекті може забезпечити більшу кількість вакансій та більшу кількість розробників, знайомих з цією технологією.

Vue.js – це прогресивний фреймворк для створення користувацьких інтерфейсів вебдодатків. Він дозволяє розробляти високоякісні вебдодатки з високою продуктивністю та ефективністю. Основні переваги Vue.js порівняно з іншими фреймворками:

1. Легкість вивчення та використання: Vue.js має просту та зрозумілу структуру, що дозволяє легко вивчити його використання. Також він має документацію, яка дозволяє швидко знайти рішення для більшості проблем.

2. Простота інтеграції: Vue.js можна легко інтегрувати з будь-яким іншим інструментом або бібліотекою, що забезпечує більшу гнучкість та контроль над вашим проектом.

3. Модульність: Vue.js дозволяє розбити ваш інтерфейс на незалежні компоненти, що підвищує його модульність. Це означає, що можливо повторно використовувати код компонентів, що дозволяє зменшити час розробки та покращити підтримку вашого коду.

4. Висока продуктивність: Vue.js використовує віртуальну DOM, що дозволяє забезпечити високу продуктивність та ефективність роботи вебдодатків.

5. Активна спільнота розробників: Vue.js має велику та активну спільноту розробників, яка забезпечує підтримку, допомогу та постійне вдосконалення фреймворку. Це означає, що можливо швидко знайти відповіді на свої запитання та знайти відповідні рішення для вашого проекту.

Angular – це повноцінний фреймворк для розробки вебдодатків, розроблений і підтримуваний компанією Google. Він базується на мові програмування TypeScript і має вбудовану підтримку компонентної архітектури.

Переваги Angular порівняно з іншими фреймворками:

1. Широкі можливості: Angular має вбудовану підтримку багатьох функцій, таких як маршрутизація, форми, аутентифікація, перевірка стану. Це дозволяє створювати повноцінні вебдодатки без додаткових зусиль.

2. Строга типізація: TypeScript забезпечує строгу типізацію, що допомагає зменшити кількість помилок в ході розробки та полегшує роботу з кодом.

3. Компонентна архітектура: Angular побудований на компонентній архітектурі, що дозволяє розбити додаток на незалежні компоненти, що підвищує модульність та забезпечує можливість повторного використання коду.

4. Оптимізація продуктивності: Angular використовує віртуальну DOM, що дозволяє досягнути високої продуктивності вебдодатків, навіть при великій кількості даних.

5. Підтримка компанією Google: Angular є фреймворком, який розробляється і підтримується компанією Google, що забезпечує його стабільність, підтримку та постійний розвиток.

6. Можливості тестування: Angular має вбудовану підтримку для тестування, що дозволяє легко виявляти та виправляти помилки в ході розробки вебдодатку.

7. Широке співтовариство: Angular має велике та активне співтовариство, що забезпечує наявність безлічі різноманітних додатків, бібліотек, пакетів та рішень для різних задач.

Графік популярності базований на кількості питань, що стосуються відповідного фреймворку на Stack Overflow наведено на рис. 1.3.

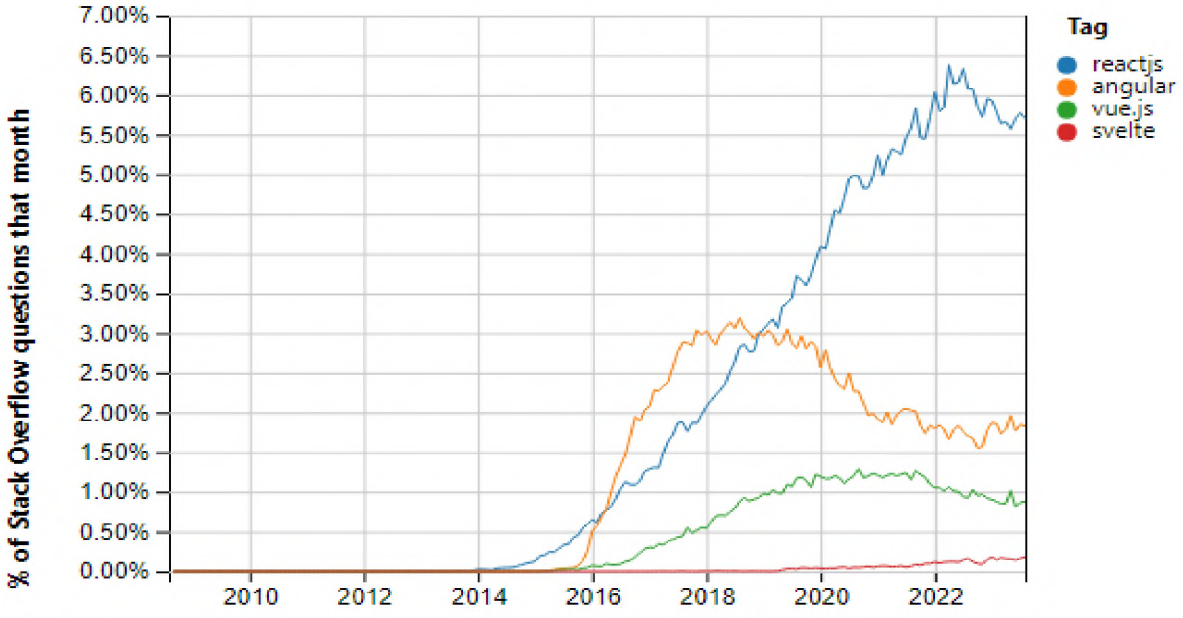

Рисунок 1.3 – Графік популярності фреймворків на Stack Overflow станом на 14.09.2023 (за матеріалами [16])

На графіку видно, що найпопулярнішим фреймворком з великим відривом від інших є React, хоча за останній рік він дещо погіршив свої показники. Angular, який був неймовірно популярним в 2016-2018р. з тих пір втратив значну частину користувачів, але залишається стабільна база розробників, які надають перевагу саме цьому фреймворку. Vue зберігає свою відносно велику аудиторію. Новий Svelte ще тільки набирає популярності, але деякі розробники вже вважають його «вбивцею» інших фреймворків.

Для покращення розробки JavaScript можна використовувати бібліотеки, такі як jQuery, Lodash, Moment.js та інші, які дозволяють працювати з DOM, маніпулювати даними та працювати зі строками та датами.

Для організації та керування структурою вебдодатку можна використовувати різні інструменти, такі як Webpack, Gulp, Grunt, Parcel, які дозволяють автоматизувати процес збірки та оптимізації вебдодатків.

Окрім того, для розробки вебдодатків можна використовувати різноманітні інструменти тестування, які дозволяють перевіряти працездатність та функціональність додатку, такі як Jest, Mocha, Chai та інші.

UX/UI (User Experience/User Interface) – це скорочення, яке поєднує два поняття, що відносяться до процесу розробки програмного забезпечення та вебдизайну. UX зосереджений на взаємодії користувача з продуктом і його відчуттями під час цієї взаємодії, тоді як UI стосується зовнішнього вигляду продукту та його елементів.

UX/UI дизайнер – це креативний фахівець, який проектує призначені для користувача інтерфейси. UX і UI - це два різних сторони дизайну, але частіше за все завдання по обох напрямках тісно пов'язані між собою, а тому їх часто робить один універсальний фахівець [17].

Що потрібно знати UX/UI дизайнеру і чим володіти:

1. Figma (дизайн інтерфейсу з можливістю колективної роботи)
2. Adobe Photoshop і Illustrator (додатковий функціонал)
3. Adobe XD (створення прототипів, дизайн інтерфейсу)
4. Invision App (створення прототипів з можливістю колективної роботи)

5. Balsamiq (створення макетів)
6. RedPen (колективний задачник)
7. Notion (менеджер завдань)
8. Framer (дизайн інтерактивних мобільних інтерфейсів)

Основні правила UX/UI дизайну є керівними принципами, які сприяють створенню високоякісного дизайну та покращенню взаємодії користувача з продуктом [18]. Розглянемо кожне з правил детальніше.

1. Врахування потреб та очікувань користувачів: дизайн має бути спрямований на задоволення потреб і вимог цільової аудиторії, задовольняти цілі, бажання і очікування від продукту.

2. Створення зручного та інтуїтивно зрозумілого інтерфейсу: інтерфейс повинен бути логічним, простим у використанні і відповідати ментальним моделям користувачів. Елементи управління, функції та контент мають бути розташовані таким чином, щоб користувачі могли легко розуміти, як вони працюють і як з ними взаємодіяти.

3. Забезпечення ефективного взаємодії користувача з продуктом: дизайн має сприяти безпроблемній та ефективній взаємодії між користувачем і продуктом. Це може бути досягнуто шляхом розумного розташування елементів, використання зрозумілих символів та мікроінтеракцій, які надають зворотний зв'язок користувачу під час його взаємодії з інтерфейсом.

4. Зменшення кількості кроків, необхідних для виконання завдання: мінімізація кількості кроків та обмеження зусиль, які необхідно зробити користувачеві для досягнення своєї мети, є важливим аспектом UX/UI дизайну. Це може включати скорочення кількості полів для заповнення у формах, використання прогресивної дисклозії для поступового відкриття інформації та інші методи спрощення взаємодії.

5. Використання чітких та зрозумілих елементів управління: елементи управління, такі як кнопки, посилання, меню, повинні бути легкими для сприйняття та натискання користувачем. Їхні функції мають бути очевидними та консистентними на всіх сторінках продукту.

6. Забезпечення однорідного дизайну та навігації на всіх сторінках продукту: чітка структура і навігація допомагають користувачам легко орієнтуватися в продукті та знаходити потрібну інформацію. Використання однакових стилів, кольорів, шрифтів та інших елементів дизайну на всіх сторінках сприяє єдності і забезпечує зручну навігацію.

7. Тестування та збір фідбеку від користувачів для постійного вдосконалення: тестування продукту з реальними користувачами та збір їхнього фідбеку дозволяє виявити слабкі місця і поліпшити дизайн. Важливо залучати користувачів у процес розробки, щоб забезпечити, що їхні потреби та вимоги враховані.

Загалом, UX/UI дизайн має на меті створити продукт, який є зручним, привабливим та легким у використанні для користувачів [2]. Дотримання цих правил допомагає покращити якість взаємодії та задоволення користувачів, що в свою чергу сприяє успіху продукту на ринку. Співвідношення кольорів в UX/UI дизайні має велике значення для створення настрою та передачі інформації. Основні принципи вибору кольорів включають:

Колірна гармонія включає в себе використання кольорових комбінацій, які взаємодіють між собою гармонійно. Наприклад, використання аналогічних кольорів (кольори, що знаходяться поряд один з одним на колірному колесі) створює спокійну і збалансовану атмосферу. Комплементарні кольори (кольори, які знаходяться протилежні один одному на колірному колесі) можуть створювати контраст і виразності. Такі відповідність та контрастність між кольорами допомагають покращити сприйняття та розуміння інформації.

Контекст також важливий при виборі кольорової схеми. Наприклад, якщо продукт пов'язаний з певним брендом, варто враховувати його кольори та стиль для створення єдності і розпізнаваності. Важливо враховувати цільову аудиторію та її вподобання щодо кольорів: деякі культурні аспекти також можуть впливати на сприйняття та значення кольорів, тому важливо бути свідомим цих нюансів. Узгоджений вибір кольорів у UX/UI дизайні сприяє створенню гармонійного, привабливого та зрозумілого інтерфейсу.

Крім того, UX/UI дизайн включає такі елементи, як інформаційна архітектура, принципи навігації, типографіка, анімація, розташування елементів на сторінці та відступи. Всі ці елементи повинні працювати разом для створення приємного, функціонального та заохочуючого досвіду для користувачів.

Навчання користувачів, збір та аналіз даних, тестування продукту та вдосконалення на основі фідбеку користувачів також важливі складові процесу UX/UI дизайну. Дизайнери UX/UI постійно працюють над удосконаленням продуктів, забезпечуючи максимальну задоволеність користувачів та досягаючи бізнес-цілей компанії.

1.3 Огляд технологій розроблення серверної частини вебсайту

Серверна частина сайту – це програмне забезпечення, яке виконується на сервері та забезпечує обробку запитів користувачів, доступ до баз даних, зберігання та надсилання даних до клієнтської частини сайту [7].Технології, що використовуються для розробки серверної частини, можуть бути різними? залежно від мов програмування та фреймворків. Деякі з найпопулярніших технологій для розробки серверної частини сайту:

1. Node.js: платформа, яка дозволяє використовувати JavaScript для розробки серверної частини сайту. Вона базується на двигуні JavaScript V8 від Google та надає розробникам можливість розробляти серверні додатки, API [19].

2. PHP: мова програмування, яка використовується для розробки вебдодатків та серверних додатків. Вона має багато фреймворків та інструментів для розробки серверної частини сайту [20].

3. Java: мова програмування, яка часто використовується для розробки вебдодатків. Java має різноманітні фреймворки, такі як Spring, Hibernate та Struts, що дозволяють швидко та ефективно розробляти серверну частину.

4. Ruby on Rails: фреймворк, який дозволяє швидко створювати вебдодатки.

5. Django: фреймворк, який використовує мову програмування Python. Django має вбудовану адміністративну панель та багато інших функцій, що спрощують розробку вебдодатків.

Це лише кілька з технологій, які використовуються для розробки серверної частини сайту. Кожна з них має свої переваги та недоліки, тому вибір технології залежить від конкретного проекту та вимог до нього. Наприклад, Node.js зазвичай використовують для розробки додатків, які мають велику кількість взаємодій з користувачами та потребують високої швидкодії обробки запитів, PHP використовують для розробки більш традиційних вебсайтів з більш простими функціями, а Java та Ruby on Rails використовують для розробки великих та складних вебдодатків.

Розглянемо ближче Node.js, тому що вона є однією з найпопулярніших технологій для розробки серверної частини сайту.

Ось декілька переваг Node.js:

1. Використання JavaScript: Node.js використовує JavaScript для розробки серверної частини, що дозволяє розробникам використовувати одну мову програмування для клієнтської та серверної частини.

2. Висока швидкість: Node.js використовує двигун JavaScript V8 від Google, що забезпечує високу швидкість виконання коду.

3. Асинхронний ввід/вивід: Node.js забезпечує асинхронну обробку ввідно-вивідних операцій, що дозволяє зменшити час очікування та покращити продуктивність додатків.

4. Масштабованість: Node.js забезпечує легку масштабованість додатків за рахунок можливості розподіленої обробки запитів.

5. Велика кількість модулів: Node.js має велику кількість модулів та бібліотек, які дозволяють розробникам швидко та ефективно створювати різноманітні додатки.

6. Open-source: Node.js є відкритим програмним забезпеченням, що дозволяє розробникам використовувати його безкоштовно та долучатися до розвитку платформи.

7. Компанії, які використовують Node.js: Node.js використовують багато відомих компаній, таких як Netflix, LinkedIn, PayPal та багато інших, що свідчить про популярність та ефективність цієї технології.

8. Екосистема NPM: Node.js має потужну екосистему пакетів та модулів, яка називається NPM (Node Package Manager). Це дозволяє розробникам швидко знаходити, встановлювати та використовувати сторонні модулі.

Також однією з популярних технологій для розробки Front-End є TypeScript. TypeScript є розширенням мови JavaScript, яке дозволяє додавати типи до звичайного JavaScript коду, що допомагає виявляти помилки в процесі компіляції, забезпечує більшу безпеку та надійність в коді та полегшує розробку більших та складних проектів.

TypeScript - це мова програмування, яка розширює JavaScript, дозволяючи розробникам використовувати строгу типізацію, класи, інтерфейси та інші функції ООП. Приклад типізації в TypeScript показано на рис. 1.4.

```
1   type Status = 'CDD' | 'CDI' | 'Stagiaire';
2
3   interface Employee {
4       firstName: string;
5       lastName: string;
6       birthdate: Date;
7       status: Status;
8   }
9
10  interface Ubidreams {
11      employees: Employee[];
12      turnover: number;
13  }
```

Рисунок 1.4 – Приклад типізації TypeScript

Переваги TypeScript:

1. Строга типізація: TypeScript дозволяє використовувати строгу типізацію, що забезпечує більшу безпеку в ході розробки та дозволяє виявляти помилки в коді на ранніх етапах.

2. Підтримка ООП: TypeScript підтримує ООП, що дозволяє розробникам використовувати класи, інтерфейси, наслідування та інші ООП-концепції для покращення організації коду.

3. Розширені функції JavaScript: TypeScript дозволяє використовувати нові функції, які не підтримуються в чистому JavaScript, такі як перевантаження функцій, декоратори та інші.

4. Підтримка від IDE: TypeScript дозволяє підтримку від різних редакторів коду, таких як Visual Studio, Visual Studio Code, Sublime Text, Atom, WebStorm та інші.

5. Краща читабельність коду: TypeScript дозволяє додавати анотації типів, що підвищує читабельність коду та дозволяє іншим розробникам краще зрозуміти код, який вони пишуть.

6. Підтримка різних фреймворків: TypeScript може бути використаний з різними фреймворками, такими як Angular, React, Vue та інші, що забезпечує більшу гнучкість при розробці вебдодатків.

7. Багатий набір інструментів та бібліотек: TypeScript має широкий вибір інструментів та бібліотек, які допомагають розробникам полегшити роботу та покращити продуктивність, такі як TypeScript Compiler (tsc), ts-node, TypeORM, NestJS, і багато інших.

TypeScript покращує читабельність коду і може бути використаний з різними фреймворками, роблячи його гнучким варіантом для розробки вебдодатків. Використання TypeScript може покращити безпеку та ефективність розробки проектів, забезпечуючи більшу надійність та зручність у процесі програмування.

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ ІНТЕРФЕЙСУ ТА ФУНКЦІОНАЛЬНОЇ ЧАСТИНИ ТА ВЕБСАЙТУ

2.1 Формування вимог до фукнкціоналу вебсайту та оцінка складності розробки

В результаті огляду ринку комерційних вебсайтів прийнято рішення про розробку вебсайту для пошуку та розміщення оголошень про продаж автомобілів на вторинному ринку. Серед варіантів побудови було обрано розроблення сайту в форматі SPA (Single Page Application). Це є оптимальним вибором з урахуванням таких аргументів, як більш ефективна навігація, покращена продуктивність, кешування даних, більш плавна інтерактивність [21]. Вебсайт повинен надавати користувачам зручний інтерфейс для швидкого та ефективного пошуку автомобілів з урахуванням різних параметрів, а також можливість розміщення оголошень для продажу авто. Основні функціональні вимоги до вебсайту:

1. Реєстрація та авторизація користувачів: можливість створити обліковий запис та увійти в систему для доступу до додаткових функцій, таких як розміщення оголошень та налаштування профілю.

2. Пошук автомобілів: можливість здійснювати пошук автомобілів за різними параметрами, такими як марка, модель, рік випуску, тип кузова, об’єм двигуна тощо. Пошук може бути здійснений за допомогою фільтрів.

3. Відображення результатів пошуку у зручному форматі, показуючи коротку інформацію про автомобілі та зображення. Користувачі повинні мати можливість переглянути докладнішу інформацію про певний автомобіль.

4. Розміщення оголошень: надати можливість розміщувати оголошення про продаж своїх автомобілів. Для цього необхідно надати форму для заповнення деталей автомобіля, включаючи марку, модель, рік випуску, фотографії та контактну інформацію.

5. Користувацький профіль: доступ до особистого профілю, можливість налаштовувати свої персональні дані, переглядати статистику та виконувати інші дії, пов'язані з їх обліковим записом.

Використовуючи інструменти та технології, такі як Figma для дизайну та прототипування, необхідно розробити інтерфейс вебсайту, який відповідає вимогам і забезпечує зручне та ефективне користування для користувачів.

При виборі технологій для розробки вебсайту слід звернути увагу на наступні аспекти:

1. Вимоги проекту: слід ретельно проаналізувати всі вимоги проекту, включаючи функціональні та нефункціональні вимоги, обсяг даних, кількість користувачів, очікувану навантаженість, вимоги до безпеки тощо.

2. Рівень складності: слід оцінити рівень складності проекту, включаючи функціональність, логіку, розмір проекту, кількість розробників, які будуть працювати над проектом.

3. Екосистема: слід вивчити екосистему технологій та фреймворків, щоб оцінити наявність готових рішень та плагінів, які можуть допомогти розробникам під час роботи.

4. Швидкість розробки: слід враховувати швидкість розробки з використанням конкретних технологій та фреймворків.

5. Масштабованість: слід враховувати можливість масштабування проекту, за необхідності додавати нові функції або розширювати функціональність в майбутньому.

6. Сумісність: враховувати сумісність технологій та фреймворків з різними браузерами та операційними системами, які можуть використовувати на сайті.

7. Безпека: слід враховувати безпекові аспекти, такі як вразливості, захист від атак, можливість шифрування даних, захист від витоків інформації тощо.

8. Підтримка: слід враховувати рівень підтримки технологій та фреймворків, так як підтримка може бути важливою для забезпечення безпеки та забезпечення розширення функціональності в майбутньому.

9. Вартість: слід враховувати вартість використання конкретних технологій та фреймворків, так як це може бути важливим фактором для бізнесу.

Таким чином, вибір технологій для розробки вебсайту повинен бути зроблений на основі ретельного аналізу всіх вищезгаданих аспектів, а також інших факторів, які можуть бути важливими для конкретного проекту.

2.2 Визначення стеку технологій для розробки вебсайту

Figma – це інтерактивний інструмент для дизайну та прототипування, який дозволяє дизайнерам створювати високоякісні вебдизайни, мобільні додатки та інші цифрові продукти. Figma дозволяє створювати, редагувати та співпрацювати над проектами в режимі реального часу, що дозволяє командам працювати швидше та ефективніше [22]. Розглянемо деякі переваги використання Figma, узагальнюючи відгуки розробників [23].

1. Колаборація в реальному часі: Figma дозволяє дизайнерам та іншим членам команди працювати разом над одним проектом в режимі реального часу, що сприяє ефективності та швидкості роботи.

2. Простота використання: Figma має інтуїтивно зрозумілий інтерфейс, що дозволяє швидко навчитися користуватися програмою навіть початківцям.

3. Наявність великої кількості готових елементів та компонентів: містить велику бібліотеку готових компонентів та елементів, що дозволяє дизайнерам прискорювати роботу та створювати проекти відразу з готовими елементами.

4. Хмарне зберігання проєктів у хмарі, що дозволяє дизайнерам доступатися до своїх проектів з будь-якого місця з підключенням до інтернету.

5. Широкі можливості експорту: Figma дозволяє експортувати проекти у різних форматах, таких як PNG, JPG, SVG, PDF та ін.

Можливості прототипування: Figma дозволяє створювати інтерактивні прототипи, що дозволяють тестувати дизайн перед тим, як він буде реалізований в життя.

Створення логотипу вебсайту в сервісі для розробки інтерфейсів Figma представлено на рис. 2.1.

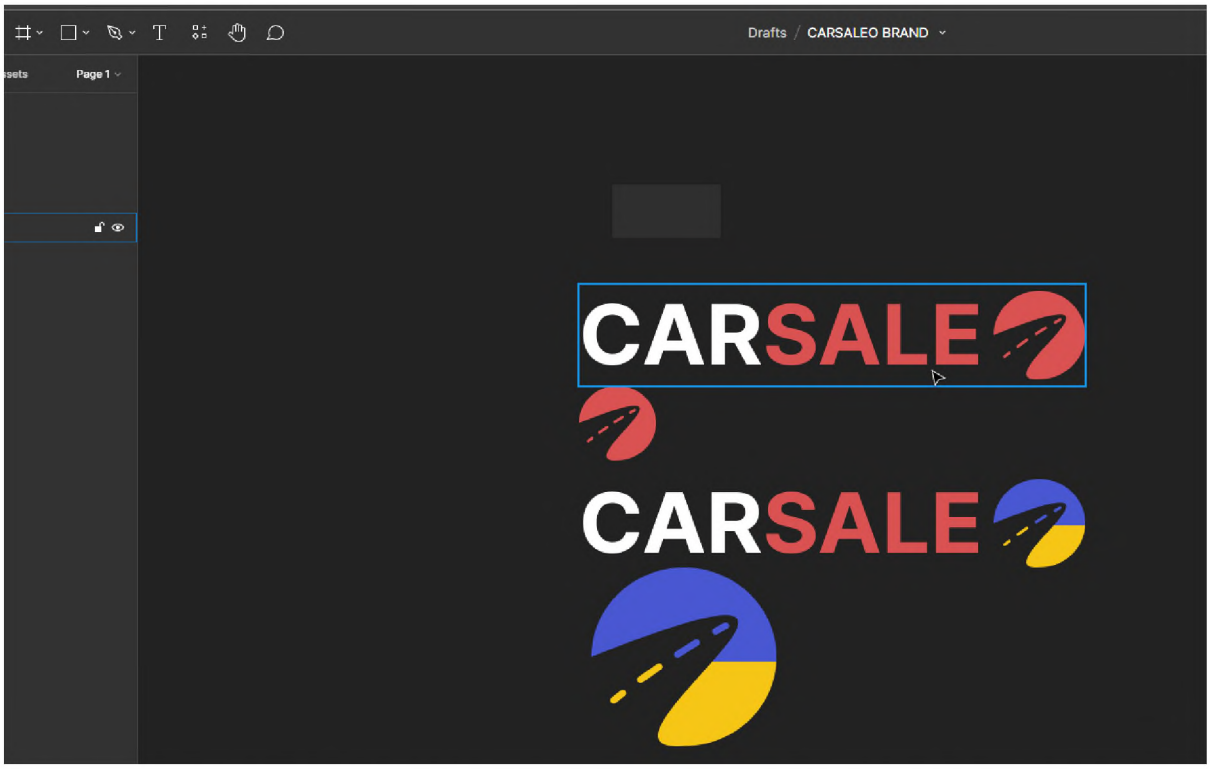

Рисунок 2.1 – Створення логотипу вебсайту в сервісі для розробки інтерфейсів Figma

Загалом, Figma є потужним інструментом для дизайну та прототипування, що дозволяє дизайнерам працювати швидше, ефективніше та співпрацювати з командою у режимі реального часу. Проаналізувавши сучасні технології для розробки вебсайтів (див. розділ 1), їх переваги та можливості, було обрано для розробки вебсайту наступні.засоби.

Для клієнтської частини вебсайту:

ReactJS – це один з найпопулярніших фреймворків для розробки інтерфейсів користувача. Він забезпечує швидку та ефективну розробку вебдодатків за рахунок використання компонентної архітектури. ReactJS дозволяє легко створювати перевикористовувані компоненти та швидко оновлювати лише ті частини сторінки, які змінилися. Це дозволяє знизити час розробки та поліпшити продуктивність вебдодатку.

Приклад синтаксису представлено на рис. 2.2.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.scss';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import { store } from './redux/store'

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <Provider store={store}>
      <App />
    </Provider>
  </BrowserRouter>
);
```

Рисунок 2.2 – Приклад синтаксису ReactJS

Styled Components – це бібліотека для створення компонентів з вбудованими стилями. Вона забезпечує зручний спосіб написання CSS-стилів для ваших компонентів та дозволяє зберігати їх разом з компонентами. Це дозволяє працювати зі стилями прямо в JavaScript – це компонент, який в стилі підсовує функцію від якихось аргументів. У Styled Component треба звертатися до функції, і вона, по суті, може робити будь-що, повертаючи будь-яке рядкове значення для стилю цього компонента [24]. Приклад синтаксису представлено на рис. 2.3.

```
const Image = styled.img`
    background: black;
    border: none;
    max-height: 140px;
    height: 100%;
    max-width: 100%;
    border-top-left-radius: 5px;
    border-bottom-left-radius: 5px;
    object-fit: cover;
    object-position: center center;
```

Рисунок 2.3 – Приклад синтаксису Styled components

Axios – це бібліотека для виконання HTTP-запитів з JavaScript. Вона дозволяє легко взаємодіяти зі сторонніми API та забезпечує зручний спосіб обробки відповідей від сервера.

SCSS – це препроцесор CSS, який дозволяє використовувати змінні, функції та інші покращення, що полегшують написання CSS-коду. Використання SCSS дозволяє зменшити кількість повторюваного коду та полегшити розуміння стилів вебдодатку. Він працює на основі синтаксису CSS, але додає деякі нові функціональні можливості. SCSS дозволяє використовувати змінні для зберігання значень, таких як кольори, розміри шрифтів, відступи та інші [25].

Для серверної частини вебсайту були обрані наступні технології.

NodeJS – це платформа для розробки серверних додатків з використанням JavaScript. Вона забезпечує швидку та ефективну розробку серверних додатків за рахунок використання неблокуючого вводу-виводу та асинхронної обробки запитів. При розробці Node.js за основу було взято двигун виконання JavaScript під назвою V8, який був створений компанією Google і використовувався в браузері Google Chrome. Оскільки після створення Node.js Javascript код можна запустити фактично в будь-якому середовищі, за допомогою цієї бібліотеки можна написати не лише Front-End, а й серверну частину вебпрограми [26].

Express - це фреймворк для розробки вебдодатків з використанням Node.js. Він дозволяє швидко створювати API для взаємодії Front-End та Back-End. Приклад створення сервера ExpressJS на рис. 2.4.

```
async function start() {
  try {
    // const uri = process.env.MONGODB_URI;
    await mongoose.set("strictQuery", false);
    await mongoose.connect(`mongodb+srv://${process.env.DB_USER}:${process.env.
DB_PASS}@cluster0.xoc2ydp.mongodb.net/${process.env.DB_NAME}
?retryWrites=true&w=majority`, { useNewUrlParser: true })
    app.listen(process.env.PORT || 3001, () => console.log("Server started"))
    console.log(app.all)
  } catch (error) {
    console.log(error)
  }
}

start()
```

Рисунок 2.4 – Приклад створення сервера ExpressJS

В Express є багато корисних функцій, таких як обробка маршрутів, робота з HTTP-запитами та відповідями, обробка запитів з різними HTTP-методами (GET, POST, PUT, DELETE) та багато іншого.

Bcrypt – це криптографічна бібліотека для Node.js, яка дозволяє захистити паролі користувачів. Вона забезпечує хешування паролів з використанням солі, що ускладнює атаки зламу пароля. Bcrypt є стандартом для захисту паролів у більшості вебдодатків. Шифрування паролю коритсувача та запис в базу даних продемонстровано на рис. 2.5.

```
const salt = bcrypt.genSaltSync(10)
const hash = bcrypt.hashSync(password, salt)

const newUser = new User({
    email,
    password: hash,
    fullName
})
```

Рисунок 2.5 – Шифрування паролю користувача та запис в базу даних

JWT (jsonwebtoken) – це бібліотека для створення та перевірки JSON Web Token (JWT). JWT - це стандарт, який використовується для автентифікації та авторизації користувачів у вебдодатках. JWT дозволяє створювати токени, які містять інформацію про користувача та його права доступу, які можуть передаватись між сервером та клієнтом для забезпечення безпеки додатка.

Nodemon - це інструмент, який автоматично перезапускає сервер при зміні файлів в проекті. Це дозволяє значно зменшити час, який потрібний для розробки, оскільки розробник може зосередитись на написанні коду, не витрачаючи часу на ручне перезапуску сервера при кожній зміні файлу. Nodemon створений для розробників, що працюють з серверною частиною проектів, такими як вебдодатки чи API.

MongoDB – це документо-орієнтована база даних, яка дозволяє зберігати дані у вигляді JSON-подібних документів. Вона дуже популярна в середовищі Node.js та часто використовується для зберігання даних у вебдодатках. MongoDB

дозволяє швидко зберігати та отримувати дані з бази даних, а також має велику кількість інструментів для адміністрування бази даних. Вона також має вбудовану підтримку MapReduce-style Aggregation та Geospatial індексів [27]. Модель схеми MongoDB новин на вебсайті відображено на рис. 2.6.

```
import mongoose from "mongoose";

const NewsSchema = new mongoose.Schema(
  {
    title: {
      type: String
    },
    text: {
      type: String
    },
    image: {
      type: String
    }
  },
  { timestamps: true }
)

export default mongoose.model('News', NewsSchema)
```

Рисунок 2.6 – Модель схеми MongoDB новин на вебсайті

Також MongoDB дозволяє розробникам працювати з гнучкими схемами даних. Можна зберігати документи різної структури в одній колекції, що дає можливість адаптувати схему даних під змінні потреби додатка. Це особливо корисно при розробці вебдодатків, де структура даних може змінюватися з часом.

РОЗДІЛ 3
РЕЗУЛЬТАТИ РОЗРОБКИ, ТЕСТУВАННЯ ТА ВИПУСК ВЕБСАЙТУ

3.1 Програмна реалізація основних функцій вебсайту

Перед початком розробки треба визначити структуру сторінок вебсайту, кількість та переходи між ними. Структура та призначення сторінок вебсайту:

1. Головна. Головна сторінка має важливе завдання зацікавити відвідувачів і надати їм загальне уявлення про сайт. Її мета полягає в тому, щоб швидко і ефективно передати основну інформацію про вас, вашу компанію, послуги або продукти, що пропонуються.

2. Всі публікації. Сторінка всіх публікацій – головне джерело інформації, на цій сторінці користувачі будуть знаходитися та взаємодіяти з вебсайтом більшість часу;

3. Публікація. Друга по важливості сторінка після всіх публікацій, є сторінка однієї публікації, ці дві сторінки працюють разом і є невід'ємними.

4. Реєстрація/Авторизація. Сторінки авторизації виконують важливу функцію, тому що від працездатність їх залежить вся робота сайту. Треба уважно протестувати їх тому що вони можуть бути підвержені різноманітним уразливостям, які можуть бути використані зловмисниками для незаконного доступу або злому безпеки користувачів.

Ось декілька типових уразливостей, на які варто звернути увагу та попередити на етапі розробки:

- атаки на перехоплення сесії;
- недостатня перевірка введених даних;
- недостатня захист від атак на перелом паролю (brute force);
- недостатній захист від капчі. Капча використовується для відокремлення людей від автоматизованих ботів. Для захисту клієнтської частини від спаму та ботів використовується Google ReCAPTCHA, React компонент ReCAPTCHA продемонстровано на рис. 3.1.

```
<ReCAPTCHA
  sitekey="*API KEY*"
  onChange={handleCaptcha}
  hl="uk"
  size={(width < 400) ? "compact" : "normal"}
/>
```

Рисунок 3.1 – React компонент Google ReCAPTCHA

5. Профіль користувача. Сторінка профілю користувача відповідає за надання інформації про користувача, його публікації та статистику;

6. Створення публікації. Сторінка з формою, яка передає інформацію від користувача до бази даних;

7. Умови використання. Сторінка з умовами використання (Terms of Service) є важною складовою будь-якого вебсайту або онлайн-платформи. Основною метою цієї сторінки є встановлення правових рамок та умов, за якими користувачі можуть використовувати вебсайт.

Далі продемонстровані функції та можливості, які були впроваджені на сторінки. Ось основні з них:

1. На головній сторінці є слайдер з актуальною інформацією вебсайта або рекламою. Головна сторінка показана на рисунку 3.2.

2.

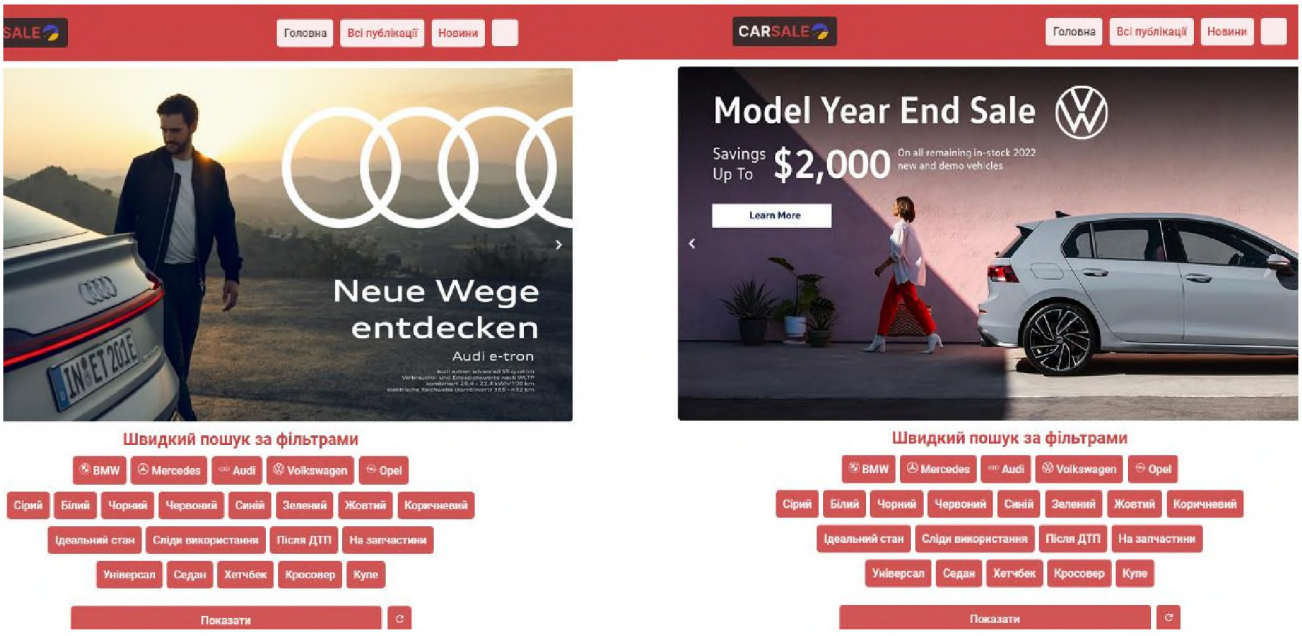

Рисунок 3.2 – Головна сторінка вебсайту з роботою слайдера

Окрім того, на головній сторінці є «Швидкий пошук за фільтрами» для спрощення пошуку та для пришвидшення роботи з вебсайтом. Є змішаний список популярних параметрів фільтрів, де можна одразу вибрати та перейти на сторінку з результатами пошуку.

Слайдер працює за принципом каруселі, при використанні цього компонента передається властивість data, яка містить масив зображень.

Компонент використовує хук useState для збереження даних про розміри вікна браузера (windowSize) та поточного зміщення (offset) каруселі. Також використовується хук useEffect для відслідковування змін розмірів вікна та додавання/видалення подій зміни розміру вікна. Використання React хуків в коді можна побачити на рис. 3.3.

```
const [windowSize, setWindowSize] = useState([window.innerWidth, window.
innerHeight])

useEffect(() => {
    const handleWindowResize = () => {
        setWindowSize([window.innerWidth, window.innerHeight]);
    };

    window.addEventListener('resize', handleWindowResize);

    return () => {
        window.removeEventListener('resize', handleWindowResize);
    };
}, []);
```

Рис. 3.3 – Використання хука useState та хука useEffect в коді

Компонент також використовує функції hendleLeftArrowClick та hendleRightArrowClick для обробки кліків на стрілки вліво та вправо відповідно. Ці функції перевіряють, чи пройшло більше 500 мс з останнього виконання функції, і змінюють offset залежно від напрямку руху (додаток Б, рис. Б.1).

Додатково використовуються два ефекти (useEffect). Перший ефект відслідковує зміни розміру вікна та оновлює windowSize, а також створює масив елементів для каруселі (items). Другий ефект встановлює події touchstart та touchmove для обробки руху по дотику на мобільних пристроях.

Нарешті, компонент повертає JSX-розмітку, яка включає стрілки вліво та вправо для переміщення по каруселі, вікно для відображення зображень та підкреслення активного зображення. Зображення залежать від поточного значення offset та windowSize.

На сторінці «Всі публікації» є розширена версія фільтрів, де можна підібрати по любому параметру авто. Також на сторінці «Всі публікації» є бокси з знайденими публікаціями авто та коротка інформація по цим публікаціям вигляд фільтрів на клієнтській частині (додаток А, рисунок А.1).

Роботу фільтрів можна описати як роботу комбінації компонентів React, стану Redux та серверної частини на Node.js з використанням Express.js.

Клієнтська частина (React+Redux):

1. Створюються компоненти фільтрів, такі як випадаючі списки, чекбокси, радіокнопки і т.д., які будуть відображати доступні опції для фільтрації автомобілів.
2. Компоненти фільтрів взаємодіють зі станом Redux, диспетчеруючи дії для оновлення фільтрів.
3. При зміні вибраних опцій фільтрів, дії відправляються до Redux store, який зберігає стан фільтрів.
4. Зміни фільтрів можуть тригерувати запити до серверної частини для отримання оновлених даних відповідно до вибраних фільтрів.

Серверна частина (Node.js + Express.js):

1. Створюється серверна логіка для обробки запитів фільтрації товарів.
2. За допомогою Express.js встановлюються маршрути, які обробляють запити клієнта.
3. При отриманні запиту фільтрації, серверна частина виконує необхідні операції для відбору товарів згідно з вибраними фільтрами.
4. Результати фільтрації повертаються клієнту у відповідь на запит.

Під час реалізації фільтрів важливо забезпечити синхронізацію стану фільтрів між клієнтською та серверною частинами. Це означає, що вибрані фільтри повинні бути передані до сервера, щоб виконати фільтрацію на стороні

сервера, а потім результати повернути назад до клієнта для відображення відфільтрованих товарів.

3. Бокси з публікаціями можна побачити на рисунку (додаток А, рисунок А.1). Ще на сторінці «Всі публікації» є гарна підгрузка публікацій, такий прийом зараз є досить популярним, його назва Skeleton React Loader, багато вебсайтів та вебдодатків використовують таку загрузку.

Skeleton React Loader - це компонент, який використовується для створення ефекту скелету (skeleton loading) під час завантаження контенту на сторінці в React.

Коли сторінка завантажується, асинхронно запитується контент з сервера, і це може зайняти певний час. У цей час користувач може бачити порожні блоки або нерозроблені елементи, що може створювати погане враження. Щоб уникнути цього, використовуються скелетні завантажувачі. Момент загрузки можна побачити на рис. 3.4

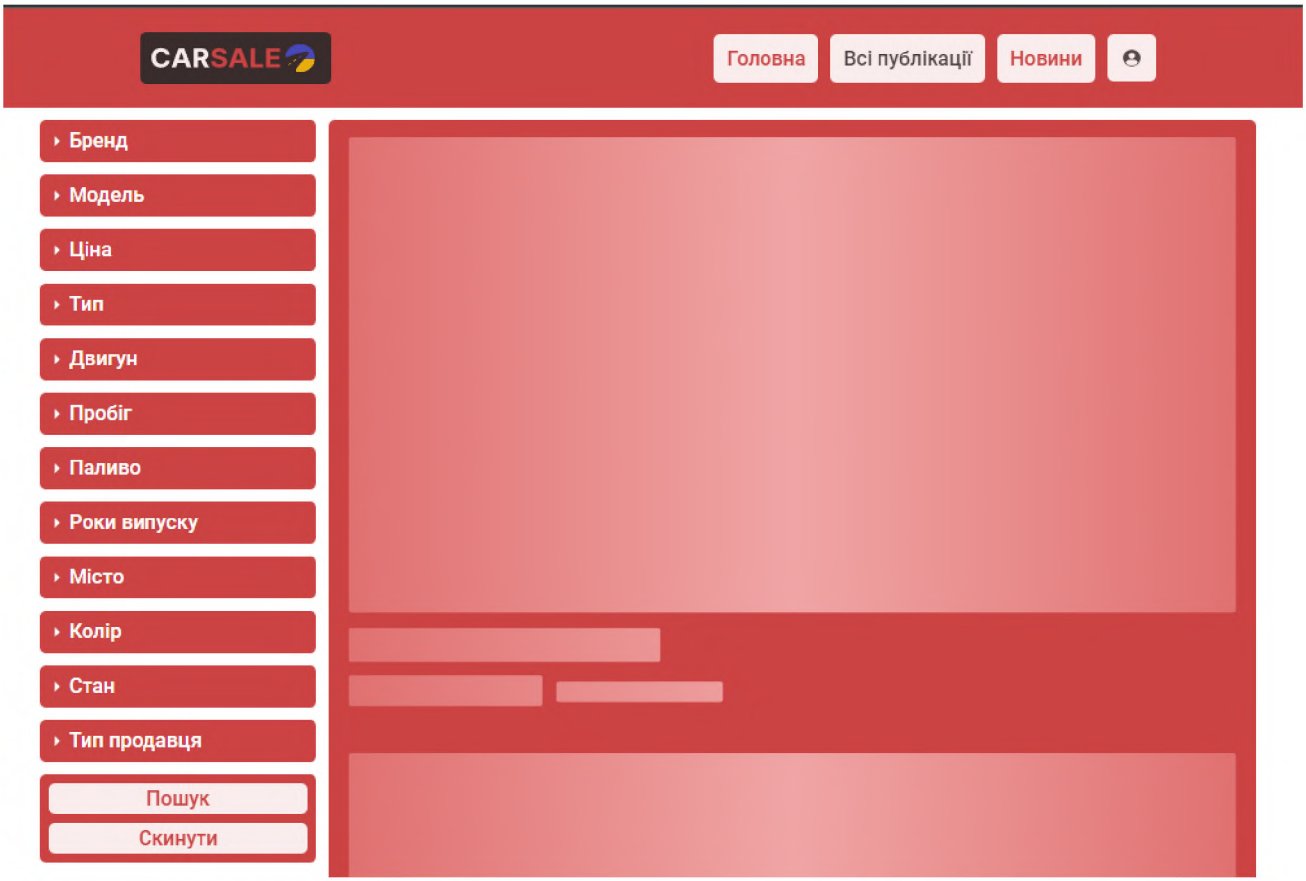

Рисунок 3.4 – Момент завантаження сторінки «Всі публікації»

4. На сторінці однієї публікації можна побачити всю інформацію про вибрану публікацію, сторінка поділена на розділи, перший розділ з короткою інформацією, далі другий розділ з детальною інформацією, в третьому зазначений продавець авто, а в четвертому і п’ятому інформація про це авто з зовнішніх джерел. Приклад публікації передавлений (додаток А, рисунок А.2)

5. Сторінка профілю представляє собою невелике меню та основний блок, де в меню можна вийти з аккаунту або додати оголошення. В основному блоці можна налаштувати профіль користувача та переглянути статистику. Також на додаток є ще один розділ для дилерських акаунтів. Дилерський акаунт представляє собою акаунт з більшою репутацією та має можливість відфільтруватись при пошуку. Для отримання дилерського акаунту, треба зв’язуватися з адміністрацією сайту.

Сторінку профілю можна переглянути на рис. 3.5.

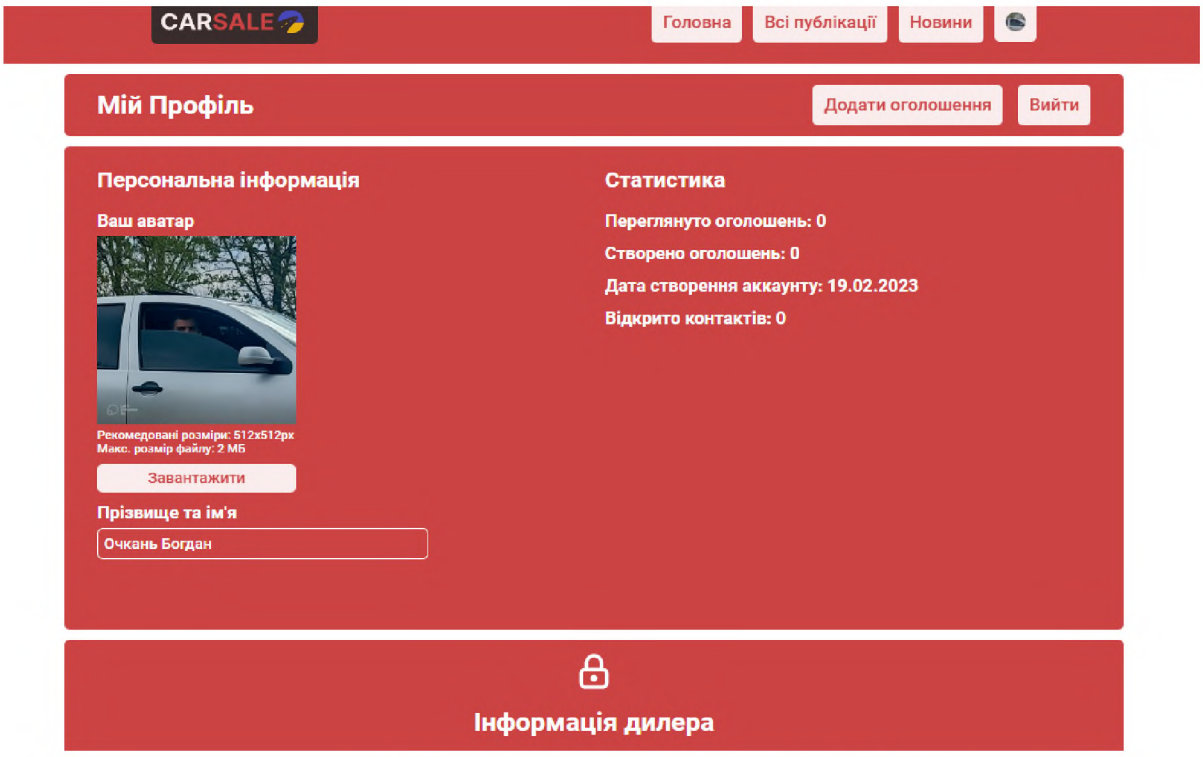

Рисунок 3.5 – Сторінка профілю

6. Сторінка створення публікації має два основних розділи, попередній перегляд, де можна подивитися як буде виглядати оголошення після заповнення

та публікації, другий розділ це сама форма деталей про авто та контактна інформація. (див. додаток А, рисунок А.3)

В цілому, вебсайт надає зручність, функціональність та різноманітні можливості для користувачів.

3.2 Інтеграції вебсайту із зовнішніми сервісами

Інтеграція зовнішніх сервісів до вебсайту за допомогою API (Application Programming Interface) є важливим аспектом розробки та функціонування сучасних сайтів. API дозволяє взаємодіяти з іншими програмними продуктами та сервісами, використовуючи стандартизовані методи та протоколи комунікації.

Прикладний програмний інтерфейс (інтерфейс програмування застосунків, інтерфейс прикладного програмування, API) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає засоби для швидкої розробки програмного забезпечення. API може бути для веббазованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек [28]. Розглянемо більш детально, чому інтеграція зовнішніх сервісів до сайту через API є важливою.

1. Розширення функціональності: інтеграція зовнішніх сервісів дозволяє розширити функціональність вашого вебсайту. Можна використовувати функції та дані з інших платформ або сервісів, щоб надати нові можливості.

2. Обмін даними: за допомогою API можна обмінюватися даними між вашим сайтом та зовнішніми сервісами. Це дозволяє отримувати актуальну інформацію, таку як погода, новини, геолокація тощо, а також надсилати дані на інші платформи, наприклад, для обробки платежів або збереження даних.

3. Підвищення продуктивності: використання зовнішніх сервісів через API дозволяє ефективно використовувати наявні ресурси та функціонал, що зменшує навантаження на сервери та прискорює відповідь вебсайту.

4. Інтеграція з соціальними медіа: API соціальних медіа дозволяють відображати живі стрічки з соціальних мереж на вебсайті, додавати кнопки поділу контенту, авторизацію через облікові записи соціальних мереж тощо. Це сприяє залученню аудиторії, підвищує взаємодію та розповсюдження контенту.

5. Партнерство та інтеграція зі сторонніми сервісами: Інтеграція зовнішніх сервісів може бути корисною для партнерства з іншими компаніями або створення власних додатків, що використовують інші платформи або сервіси.

Приклади API-інтеграцій включають платіжні шлюзи, соціальні мережі, картографічні сервіси (наприклад, Google Maps), сервіси електронної пошти, системи аналітики, мобільні додатки тощо.

Для створення вебсайту було використано чотири API, а саме:

1. Google ReCAPTCHA – це система, що була початково розроблена в університеті Карнегі Мелон і базується на використанні CAPTCHA для оцифровування текстів книг заодно із захистом вебсайтів від доступу ботами до обмежених ресурсів. 16 вересня 2009 року Google придбав reCAPTCHA. У цей час reCAPTCHA оцифровує архіви газети New York Times. Вже опрацьовано випуски The New York Times за двадцять років і очікується, що у 2010-му буде оцифровано архіви ще за 110 років [29]. Її використання можна обґрунтувати тим що вона добре захищає від бот систем при підборі паролю для злому акаунту, або для створення великої кількості акаунтів бот системою.

2. НБУ Курс валют – це JSON файл з актуальним курсом валют, потрібен для конвертування однієї валюти в іншу за актуальним курсом.

3. JSON (JavaScript Object Notatio) – це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передавання структурованої інформації через мережу (завдяки процесу, що називають серіалізацією) [30].

4. ChatGPT (Generative Pre-trained Transformer, укр. Породжувальний попередньо тренований трансформер) – чат-бот зі штучним інтелектом,

розроблений лабораторією OpenAI. Це велика статистична модель мови, оптимізована для ведення діалогів та відлагоджена завдяки технікам навчання з учителем та навчання з підкріпленням. В основі прототипу лежить модель OpenAI GPT-3.5 — покращена версія GPT-3 [31]. Його використання забезпечує оцінку нейромережі, спочатку йому надається інформація про публікацію та з налаштуваннями йому відправляється прохання оцінити співвідношення ціни та якості і доцільності покупки цього авто.

5. Baza-gai – це сервіс який надає послуги отримання відкритої інформації по державному номеру авто, було надіслано повідомлення до керівництва сайта та був запит на API ключ, мені був надісланий API ключ та далі було зроблено запити при створені публікації до їх сервісу.

3.3 Проведення попереднього тестування вебсайту

Тестування сайту є невід'ємною та критично важливою частиною процесу розробки та підтримки вебсайтів. Воно спрямоване на перевірку функціональності, якості, безпеки та коректності роботи сайту перед його впровадженням або після внесення змін. Ось деякі ключові аспекти, що підкреслюють важливість тестування сайту.

1. Виявлення помилок: тестування дозволяє виявити помилки, дефекти та несправності в роботі сайту перед тим, як він буде запущений для використання користувачами. Це дозволяє уникнути проблем, які можуть негативно вплинути на досвід користувачів та репутацію вашого бренду.

2. Покращення користувальницького досвіду: тестування сайту допомагає впевнитися, що вебсайт надає зручний та задовільний користувальницький досвід. Можна перевірити навігацію, швидкість завантаження сторінок, функціональність взаємодії з елементами сайту, адаптивність до різних пристроїв та браузерів. Покращення користувальницького досвіду сприяє залученню та утриманню відвідувачів на вашому сайті.

3. Забезпечення функціональності: тестування допомагає перевірити, чи працюють всі функції та сервіси на сайті належним чином. Можена перевірити форми зворотного зв'язку, функцію пошуку, обробку платежів, реєстрацію користувачів та інші важливі елементи функціональності сайту. Це дозволяє упевнитися, що сайт виконує свої завдання без помилок та перебоїв.

4. Забезпечення безпеки: тестування допомагає виявити потенційні слабкі місця та ризики безпеки на сайті. Можена перевірити, чи належним чином захищений сайт від хакерських атак, перехоплення даних та інших загроз. Захист інформації користувачів та забезпечення конфіденційності даних є важливими аспектами, які потрібно перевірити перед запуском сайту.

5. Оптимізація продуктивності: тестування дозволяє оцінити продуктивність сайту та виявити можливі проблеми, що можуть сповільнювати завантаження сторінок або зменшувати швидкість роботи сайту в цілому. Можена перевірити час відгуку сервера, оптимізацію зображень, кешування та інші аспекти, що впливають на продуктивність.

На етапі тестування адаптації, коли перевіряється працездатність веб-додатку на різних пристроях і розмірах екранів, можуть виникати деякі типові помилки. Однак, існує кілька кращих практик, які можуть допомогти уникнути цих помилок:

Респонсивний дизайн: розробляючи вебдодаток, слід звертати увагу на респонсивний дизайн, що дозволяє адаптувати вигляд і макет сторінки до різних розмірів екранів. Використовування медіа-запитів та гнучких одиниць виміру, такі як відносні відсотки або вьюпорт, для забезпечення правильного відображення на різних пристроях.

В цілому, тестування сайту є важливим етапом процесу розробки, який допомагає забезпечити якість, функціональність, безпеку та задоволення користувачів. Воно дозволяє виявити проблеми та недоліки до того, як сайт буде доступний публіці, що забезпечує успішне функціонування та репутацію вашого вебпроекту. Для прикладу візьмемо сторінку всіх публікацій, тому що вона користувач буде проводити більшість часу на ній.

Перевірка адаптації здійснювалася на різних розмірах екрану. Для ширини екрану 1024 пікселів результат можна побачити на рис. 3.6

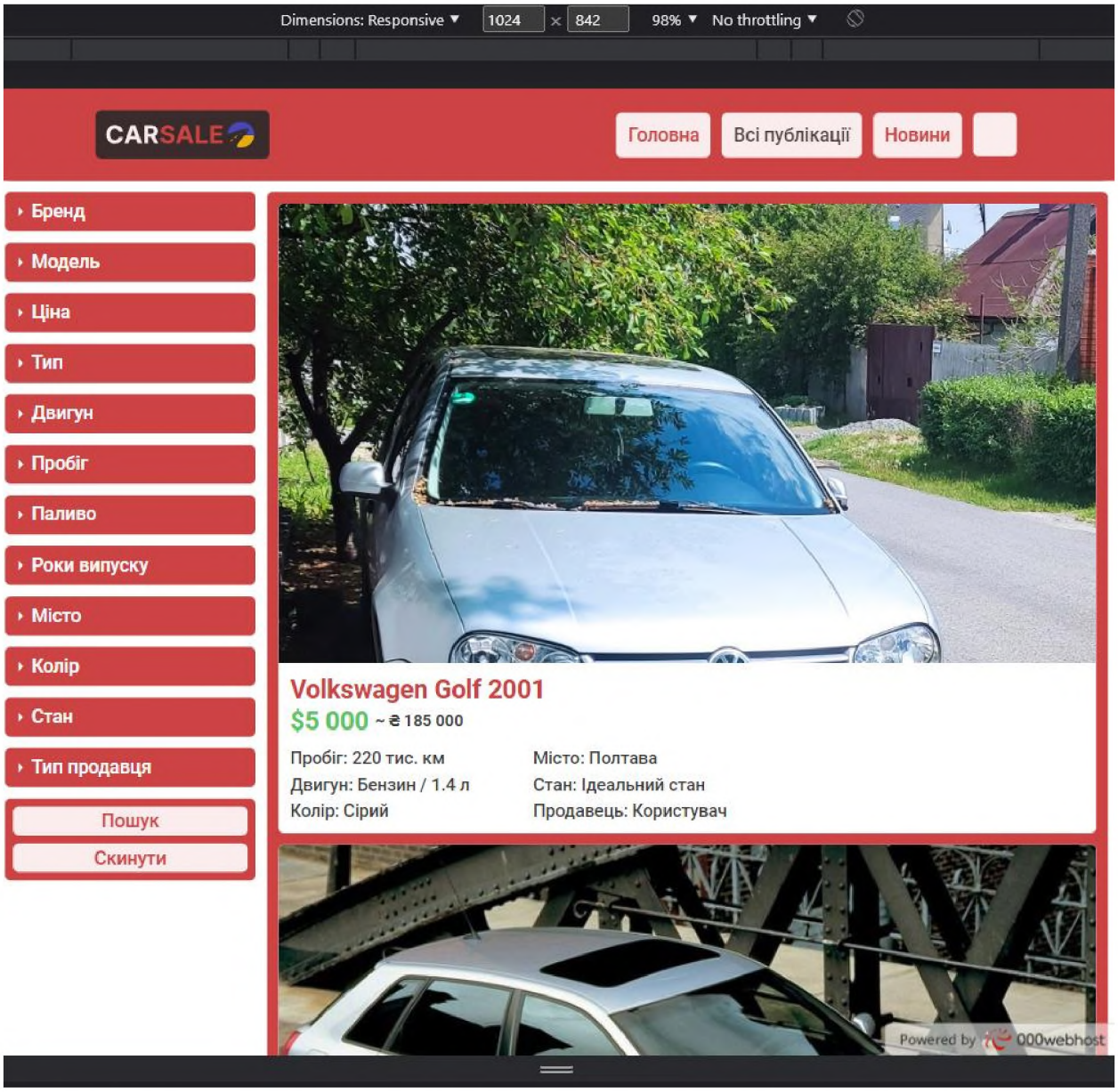

Рисунок 3.6 – Вигляд вебсторінки для ширини екрану 1024 пікселів

Багато настільних комп'ютерів, які продаються на ринку, підтримують роздільність екрана 1024x768 або вище. Це включає в себе як персональні комп'ютери, такі як ПК з операційною системою Windows або Mac, так і сервери з операційними системами Linux.

3. Окреме тестування проводилося для ширини екрану 768 пікселів. Результат можна побачити на рис. 3.7

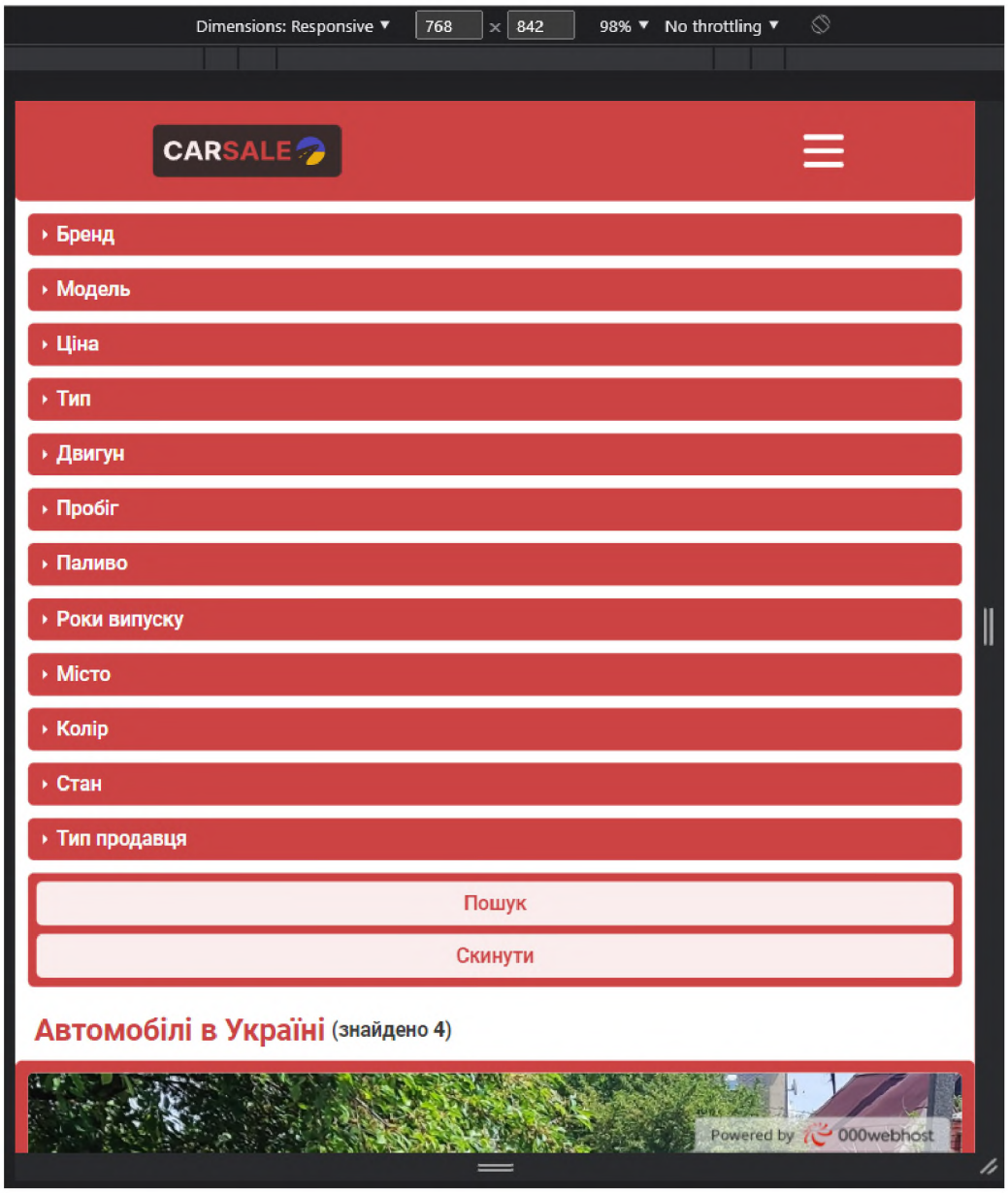

Рисунок 3.7 – Ширина екрану 768 пікселів (планшет)

На визначеному моменті, екран переходе в другий режим, це працюють правила CSS, а саме @media запити.

Медіа-запити дозволяють визначати, які стилі повинні бути застосовані до елементів на основі характеристик пристрою, таких як розмір екрану, орієнтація, роздільна здатність тощо.

Тестування вигляду та читабельності вебсторінки на екрані смартфону. Ширина екрану для прикладу взята 375 пікселів. Результат можна побачити на рис. 3.8.

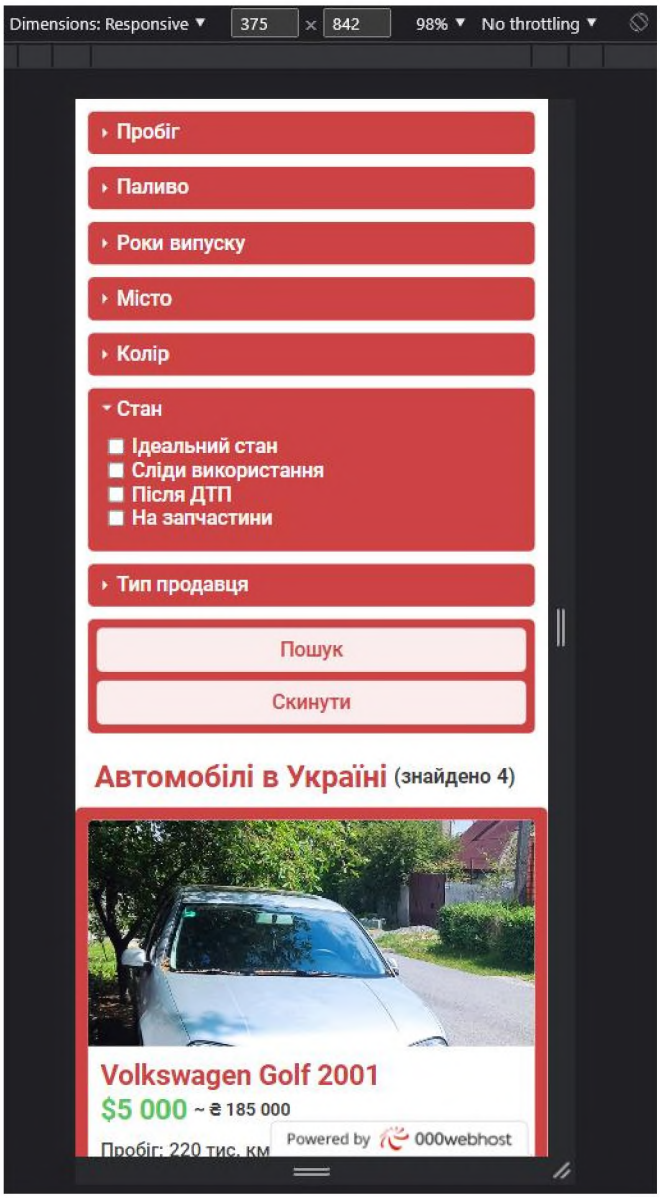

Рисунок 3.8 – Ширина екрану 375 пікселів

Нижче 375 пікселів зазвичай не роблять адаптацією, тому що актуальні розміри мобільних пристроїв зазвичай більше цієї ширини. Після ретельного тесту адаптації можна зробити висновок, що створення респонсивного дизайну спрощує розробку та адаптацію.

Обрання відповідних сервісів для релізу вебсайту є важливим кроком, який впливає на його доступність, швидкість та зручність використання.

Для релізу сайту було обрано:

– Для серверної частини вебсайту, сервіс replit, він надає можливість розгортання серверної частини вебсайту без складнощів. Він дозволяє

розробникам легко створювати, тестувати та розгортати серверні додатки без складнощів. Інтерфейс Replit продемонстровано на рис. 3.9.

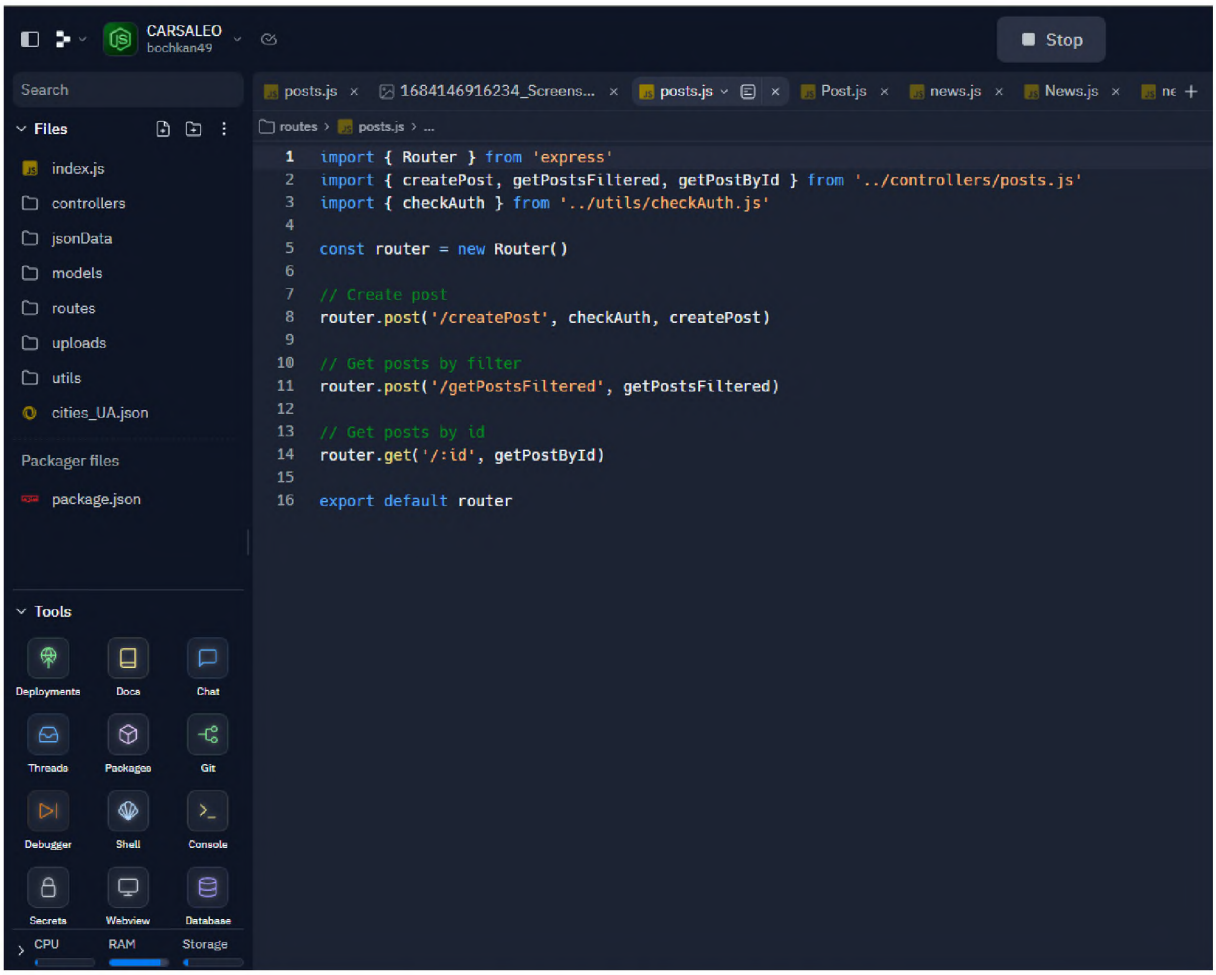

Рисунок 3.9 – Інтерфейс Replit

Для клієнтської частини вебсайту було обрано 000webhost, це простий та безкоштовний хостинг вебсайтів зі зручним інтерфейсом та інтуїтивно-зрозумілими функціями. Він дозволяє безкоштовно розмістити клієнтську частину вебсайту та забезпечує доступ до необхідних ресурсів для оптимальної роботи сайту.

000webhost має стабільну інфраструктуру, що дозволяє забезпечити надійність та доступність вебсайту. Панель керування 000webhost зображено на рис. 3.10.

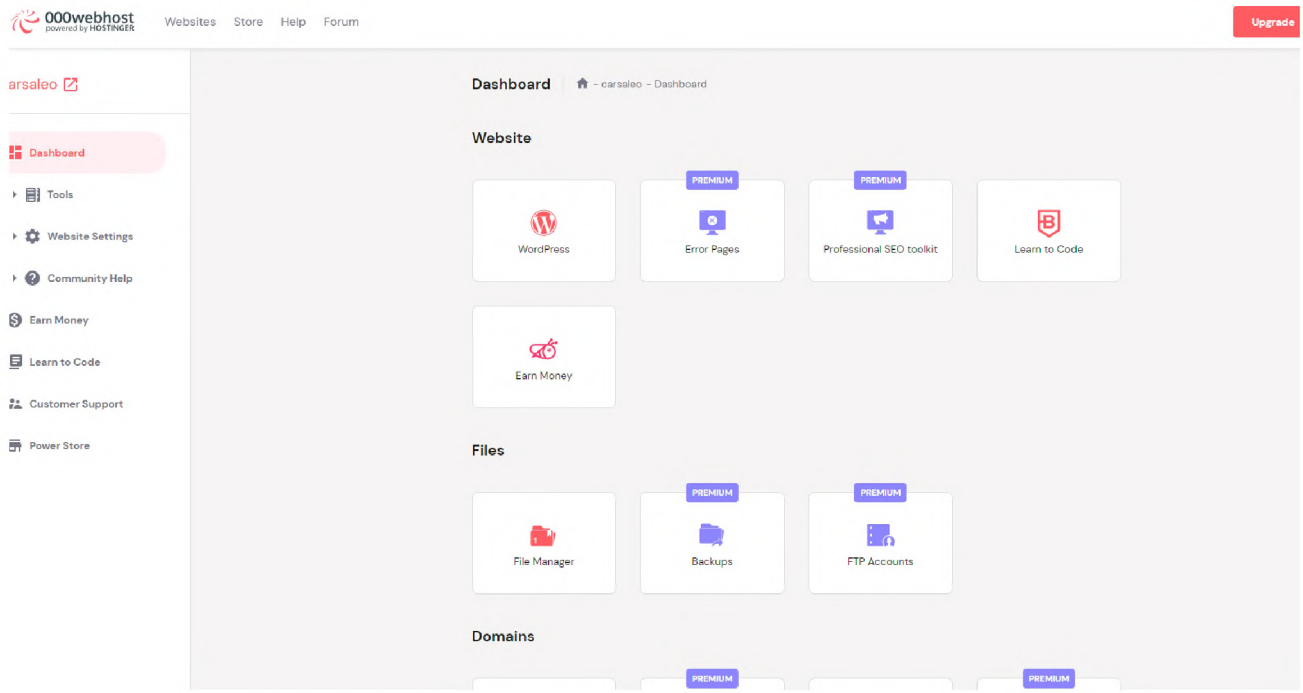

Рисунок 3.10 – Панель керування 000webhost

Використання таких сервісів дозволяє зосередитися на розробці вебсайту, мінімізуючи складнощі та час, пов'язаний зі створенням серверної та клієнтської частини.

3.4 Оцінювання економічної та технічної ефективності вебсайту

Розглянемо наближений перелік витрат, пов'язаних із створенням та розміщенням вебсайту. Оскільки використовується безкоштовна версія програмного продукту Visual Studio Code як основу, до складу витрат на створення вебсайту включаються такі елементи:

1. Витрати на розміщення в мережі інтернет (хостинг та домен).
2. Заробітна плата для програміста (FullStack розробник).
3. Витрати на необхідні матеріали для комп'ютера.

Вартість розробки вебсайту може бути дуже гнучкою, при проектуванні визначаються безліч деталей, функцій, можливостей тощо. З іншого боку

компанії та агентства які займаються розробкою сайтів мають різні рівні та вартості на послуги.

Для визначення середньої вартості розробки було взято статистку середньої вартості розробки інтернет магазину [32]. В розділі ціни за типом, є тип інтернет-магазин, ціна вказана 100000 грн.

Якщо ж брати середню заробітну плату Fullstack JavaScript розробника за статистикою, вона становить 2600 доларів [33]. Тому можна зробити висновок що розробка вебсайту буде коштувати приблизно 2600 доларів США. В цей перелік буде входити розробка клієнтської частини, серверної частини, просте налаштування SEO та просте тестування.

Припустимо, що компанія звернулася до веброзробника для створення сайту, і вони домовилися, що веброзробник зможе написати сайт протягом 4 тижнів за загальну вартість 2600 доларів. За потреби можна домовитись з розробником на підтримку вебсайту та виправлення помилок в подальшому, але це виконується за потреби, наприклад при покращені дизайну сайту, додаванні нових функцій або доробки теперішніх функцій.

В розрахунках використовується середній курс долара відносно гривні на травень 2024 року, який становить 1 долар США = 39,8 грн.

Розглянемо розрахунок витрат за рік на створення та утримання вебсайту (табл. 3.1).

Таблиця 3.1 – Витрати за рік на утримання та створення вебсайту

Найменування	Сумма у долларах США	Сумма в грн
Послуга веброзробника (разова)	2600	93600
Хостинг та домен	156	5616
Використання інтернет (абонплата)	80	2880
Разом:	2836	102096

При роботі з програмним забезпеченням при створенні вебсайту необхідно задіяти як мінімум один персональний комп’ютер або ноутбук (табл. 3.2).

Таблиця 3.2 – Вартість необхідного комп'ютерного обладнання

Найменування	Ціна 1 шт. у доларах США	Ціна 1шт. в грн	Кількість, шт.
Комп'ютер	500	18000	1

Метод лінійної амортизації є одним зі способів розрахунку амортизації основних засобів, включаючи комп'ютерну техніку. Для проведення розрахунку амортизації за допомогою формули 3.1 потрібно знати середньорічну вартість основних засобів (С) та норму амортизаційних відрахувань (Н). Формула для розрахунку амортизації за методом лінійної амортизації виглядає так:

$$A = C \times H / 100\%, \quad (3.1)$$

де А – сума амортизаційних відрахувань за рік, грн;

Н – норма амортизаційних відрахувань, %;

С – середньорічна вартість основних засобів.

Відповідно отримаємо: 500$ × 20% / 100% = 100$

Отже, сума амортизаційних відрахувань за рік становить 100 доларів США. Загальний кошторис витрат на виготовлення вебсайту наведено в табл. 3.3.

Таблиця 3.3 – Кошторис витрат на розробку вебсайту

Найменування економічних елементів витрат	Сума, $	Сума, грн
Витрати на оплату праці	2600	93 600
Хостинг та домен сайту	156	5 616
Амортизація основних фондів	100	3 600
Інші витрати (комунальні послуги)	820	29 520
Разом:	3200	132 336

Комунальні послуги включають в себе інтернет та енергоживлення для комп'ютера, з витратою енергії приблизно 60 кВт/год на робочий день. Вартість енерговитрат (див табл. 3.3) розрахована за тарифами на 01.05.2024 р.

Таким чином, згідно попередньої оцінки, загальна вартість проєкту складається з 132,336 грн та 18,000 грн, що дорівнює 150,336 грн.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено аналіз та обрання оптимальних технологій розробки вебсайту. Було детально вивчено структуру сучасних вебсайтів та встановлено значущість як клієнтської, так і серверної частини. Для розробки клієнтської частини вебсайту було вибрано мову програмування JavaScript та використано інструмент Figma для розробки інтерфейсу. Серверна частина була успішно розгорнута на платформі Replit.

Під час створення інтернет-магазину було враховано вимоги до SPA-додатків (Single Page Application), що дозволяє знизити час завантаження сторінки та поліпшити користувацький досвід. Для забезпечення швидкої роботи сайту було використано NodeJS, який забезпечує масштабованість та високу продуктивність.

Дизайн сайту був розроблений з використанням бібліотеки Styled Component, що дозволяє легко керувати стилями та забезпечити їх перевикористовування. Крім того, вебсайт має Full-Stack структуру, що дозволяє працювати з базою даних та здійснювати операції з її збереженням та оновленням.

Було проведено огляд існуючих підходів до розроблення сучасних веб-сайтів, зокрема вивчено різні методики та практики. Також розглянуті базові технології розроблення як клієнтської, так і серверної частини веб-сайту, що включало аналіз мов програмування та інструментів для їх розробки.

Обґрунтовано важливість вибору технологій та інтерфейсу вебсайту. Викладені були принципи UX/UI та виконано вибір та розробку кольорової палітри, що сприяє привабливості та гармонії. Окремий наголос зроблено на значенні доступності, що допомагає забезпечити рівний доступ до вебсайту для всіх користувачів.

Ще було проведено розробку, тестування та випуск вебсайту. Були реалізовані основні функції та пройдено їх тестування для перевірки працездатності. Додатково були інтегровані зовнішні сервіси з метою

покращення функціональності вебсайту. Після успішного тестування вебсайт був готовий до релізу. Крім того, було проведено економічне обґрунтування проекту, що дозволило оцінити вартість та ефективність розробки.

Було проведено дослідження для з'ясування потреб та очікувань потенційних клієнтів. Були проаналізовані існуючі інтернет-магазини, що пропонують автомобілі, та виявлені їх переваги та недоліки. На основі цього була розроблена концепція вебсайту, яка містить у собі найбільш ефективні та зручні функції для покупців та продавців.

Таким чином, створення інтернет-магазину з продажу автомобілів на основі технологій React, NodeJS та Styled Component успішно завершено. Результатом проекту є зручний та безпечний інтерфейс для покупців та продавців автомобілів, який надає можливість здійснювати швидке та зручне замовлення, отримувати детальну інформацію про автомобілі та спілкуватися з менеджерами вебсайту.

Загальною метою цієї кваліфікаційної роботи було створення ефективного, привабливого та функціонального вебсайту з використанням сучасних технологій та методологій розробки. Завдяки аналізу, обранню оптимальних рішень та детальній розробці, ціль була досягнута.

Отриманий результат є значущим з практичної та наукової точок зору. З практичної точки зору, створений вебсайт може бути використаний в реальному бізнес-середовищі для продажу автомобілів, забезпечуючи зручність та ефективність процесу купівлі та продажу.

Використання сучасних технологій та методологій розробки, таких як React, NodeJS та Styled Component, дозволяє забезпечити швидку та масштабовану роботу вебсайту, покращити користувацький досвід та забезпечити безпеку взаємодії.