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 та оптимізація комерційного вебсайту для презентації можливостей трейд-бота»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та технології спеціальності
126 Інформаційні системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТбз_41
Любий В.Р.
Керівник: Флегантов Л.О.
Рецензент: Ковальчук С.Б.

Полтава – 2024 року

ВСТУП

В сучасному світі неможливо уявити життя без інтернету, який дійсно став невід'ємною частиною більшості людей. Значну роль у процесі користування інтернетом відіграють браузери – оглядачі вебсайтів, оскільки саме вони є посередниками між людиною та послугами мережі. Сучасні вебсайти дають змогу людям будувати власний бізнес, вирішувати безліч комплексних задач майже у всіх сферах діяяльності, від інформаційного пошуку й навчання до комерції, бізнесу, фінансової діяльності, медицини, розваг тощо. Серед бізнес-сфер цікавим нарямком комерції виступає фінансова діяльність через інтернет. Ффінансові ринки стають все більш доступними для широкої аудиторії, і багато людей зацікавлені в торгівлі акціями, валютами та криптовалютами.

Актуальність теми кваліфікаційної роботи пов'язана із тим, що не кожен із користувачів інтернет-послуг має достатні знання і досвід, щоб успішно здійснювати фінансові операції на ринку. Тому автоматизовані трейд-боти стають популярними інструментами для автоматичної торгівлі й доволі раціональним способом виступає їх інтеграція з вебсайтом. При цьому вирішується комплексний підхід та знаходження нових способів комбінування усталених технологій побудови функціональних і комерційно ефективних вебсайтів. Вебсайти компаній, які будують власний бізнес і хочуть, щоб клієнти мали змогу скористатись зручним та сучасним інтерфейсом, щоб зробити вибір на користь їхньої компанії, вимагають особливої уваги.

Метою даного проекту є здійснення обґрунтованого вибору технологій розробки, якісного дизайну та SEO-оптимізації комерційного вебсайту, який буде презентувати можливості трейд-бота і залучати нових користувачів та сприяти популяризації сервісу.

Завданнями кваліфікаційної роботи є:

– дослідити сучасні тенденції дизайну та засобів досягнення функціональності й властивостей комерційних вебсайтів;

– здійснити вибір середовища розробки для створення вебсайту ;

– надати опис бізнес-моделі, яка буде використовуватися для комерційної діяльності сайту;

– розробити макет вебсайту та продумати розташування необхідного контенту;

– проаналізувати і застосувати на прикладах інструментальні засоби та методи просування вебсайту;

– здійснити технічну реалізацію вебсайту з використанням сучасних технологій та заходів з оптимізації для покращення продуктивності та швидкості завантаження.

Об'єктом дослідження виступає процес розроблення адаптивного комерційного вебсайту із використанням сучасних вебтехнологій.

Предметом дослідження є технології дизайну та розробки сучасного вебсайту на прикладі комерційного вебсайту, а також засоби оптимізації сайту для пошукових систем.

Методи дослідження: аналітико-синтетичний, інформаційно-пошуковий, робота з репозитаріями програмних кодів, технічного аудиту вебсайтів, графічний та інші.

Практична значущість полягає у розробці вебсайту комерційного спрямування, який має сучасний дизайн і висвітлює переваги та функціональні можливості трейд-бота Proton Trade, має форму зворотнього зв'язку, щоб користувач міг реалізувати всі запити щодо послуг компанії та залишити заявку в разі зацікавлення послугами. Дизайн сайту має адаптивні властивості. Його можна використати для просування будь-яких інших послуг, удосконалити.

Результати роботи апробовані в рамках щорічної студентської наукової конференції кафедри інформаційних систем та технологій ПДАУ в 2024 р.

Структура і обсяг кваліфікаційної роботи: пояснювальна записка складається із вступу, трьох розділів основної частини, висновків, списку використаних джерел та додатків. Вона викладена на 58 сторінках формату А4, містить 39 рисунків, 2 таблиці та додатки. У роботі використано 36 науково-технічних інформаційних джерел.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБЛЕННЯ ВЕБСАЙТІВ КОМЕРЦІЙНОГО СПРЯМУВАННЯ ТА ДИЗАЙНУ ІНТЕРФЕЙСІВ

Аналіз основних понять та предмету дослідження вебсайтів

Розробка сайтів це ніша в програмуванні, що весь час оновлюється, адаптується та покращується в цілому, і, наразі є актуальною, такою буде і надалі. Але, щоб бути сучасним розробником, потрібно постійно слідкувати за змінами в технологіях і нововведеннями в них.

Задаючись питанням, що взагалі таке вебсайт, можна знайти таке визначення: «… вебсайт – це сукупність програмних, інформаційних, а також медійних засобів, логічно пов'язаних між собою. По суті ж веб-сайт – це віддзеркалення успішності фірми, її обличчя [1]».

Якісний вебсайт повинен відповідати багатьом критеріям, з них можна виділити 10 найважливіших:

1. Зміст або наповнення сайту корисною інформацією. Сайт повинен залучати відвідувача якісними статтями, оригінальними фотографіями та зображеннями. Повноцінний зміст формується власними силами, або залученням праці професійних копірайтерів та фотографів.

2. Структура, тобто зручне розміщення інформації на вебсайті. Сайт повинен бути організований так, щоб випадковий відвідувач легко знайшов інформацію на тему, що його цікавить, по системі вкладок. При створенні сайту слід пам'ятати, що потенційний користувач не готовий довго розумітися на складній системі переходів від сторінки до сторінки. Користувач воліє прості та зрозумілі рішення.

3. Оформлення або дизайн. Стиль сайту є важливим для формування позитивного враження про нього. В оформленні вітається яскрава індивідуальність, здатна вселяти довіру до змісту сайту. Важливо, щоб у гонитві

за неповторним виглядом сайту, його творець не забував про інтереси відвідувачів, і не налякав їх надто оригінальним оформленням.

4. Оновлення вмісту. Сайт живе, поки на ньому з'являються нові статті, картинки, чудові фотографії та відеозвіти про події. Без свіжої інформації, сайт приречений на забуття та загибель.

5. Адреса або розміщення в інтернеті. Щоб сайт викликав довіру, слід подбати про лаконічну його назву та розміщення на платному хостингу. Серйозні компанії ніколи не користуються послугами «безкоштовних сайтів», це негативно позначається на їхньому іміджі.

6. Швидкість завантаження сайту. Користувач не чекатиме на завантаження більше чотирьох або п'яти секунд. Тому при створенні сайту засікайте період завантаження за допомогою секундоміра.

7. Продумана система зв'язку із відвідувачем. Контакти для зв'язку з адміністрацією сайту мають бути легко знайдені. Передбачте кілька варіантів: зв'язок через електронний лист, за номером телефону, через онлайн-консультанта. Якісний зворотний зв'язок допомагає створити коло постійних відвідувачів сайту.

8. Індексація сайту, його просування. Основний алгоритм пошуку інформації в Мережі – через пошукові системи та ключові запити. Тому контент сайту повинен містити ключові слова та цілі словосполучення, за допомогою яких пошукові системи переадресовуватимуть користувачів на конкретний сайт.

9. Реклама. Сайт без реклами виглядає підозрілим. Сайт з надмірною кількістю реклами, з нав'язливими банерами і величезними вікнами, що спливають, викликає бажання відразу ж залишити його назавжди. Турбота про додатковий дохід через рекламу не повинна шкодити самому існуванню сайту.

10. Інтеграція із соціальними мережами . Практично весь час в інтернеті користувачі проводять у улюблених соціальних мережах. Використання будь-яких способів розміщення посилань на робочий сайт у мережах значно підвищує індекс його відвідуваності та рейтинг компанії [2].

Більш детально зупинимося на змісоному наповненні поняття вебдизайн.

Вебдизайн – це проектування та візуалізація продуктів, їх макетів та прототипів. Сьогодні він не пов'язаний з кодом та включає декілька аспектів:

- візуалізація прототипу;
- проектування сайту або додатка;
- графічний дизайн;
- взаємодію з користувачем.

Професійний вебдизайнер сучасності знає й розуміє основні принципи сайтостворення, розбирається у верстанні та маркетингу [3]. Втім, веброзробник не обов'язково має розумітися в дизайні, він повинен лише знати як його реалізувати за допомогою певних інструментів і зробити це якісно, неякісний продукт розробляти просто немає сенсу – він буде не потрібен ані користувачу, ані замовнику.

Адаптивність сайту – це можливість правильного відображення на пристроях з різними технічними характеристиками [4]. Тобто це здатність вебсайту підлаштовуватись під пристрій, з якого його переглядають і залежить вона від ширини екрану користувача.

Опис існуючих методів розробки комерційних вебсайтів

Існують різні методи розробки інтернет-магазинів, кожен з яких має свої особливості та нюанси. Розглянемо далі кілька найпоширеніших підходів.

1. Платформи електронної комерції. Платформи електронної комерції надають комплексні рішення й частіше використовуються саме для створення інтернет-магазинів. Ці платформи, такі як Shopify, BigCommerce і Magento, пропонують низку функцій, включаючи управління каталогом товарів, функціональність кошика для покупок, платіжні шлюзи і системи управління замовленнями. Вони часто постачаються шаблонами та темами, які налаштовуються, що полегшує бізнесу створення та управління інтернет-магазинами без глибоких технічних знань.

2. Системи управління контентом (CMS). Для створення комерційних сайтів можна використовувати такі платформи CMS, як WordPress, Joomla і Drupal. Вони пропонують плагіни або розширення для електронної комерції, такі як WooCommerce для WordPress, які додають функціональність електронної комерції до CMS [5].

Платформи CMS забезпечують гнучкість у дизайні, управлінні контентом та інтеграції зі сторонніми сервісами. Вони підходять для компаній, які вже мають вебсайт, створений на CMS, і хочуть розширити його в напрямку електронної комерції.

3. Індивідуальна розробка. Вона передбачає створення індивідуального вебсайту з нуля відповідно до конкретних бізнес-вимог. Цей метод зазвичай передбачає наймання професійного веброзробника або команди розробників, які володіють знаннями у сфері вебдизайну, мов програмування та рішень для електронної комерції. Процес починається з планування та розробки структури, макета та інтерфейсу вебсайту. Потім розробники впроваджують необхідні функції, такі як управління каталогом товарів, кошиком для покупок, платіжними шлюзами та системами обробки замовлень. Індивідуальна розробка пропонує максимальну гнучкість і можливості кастомізації, але вимагає більше часу, ресурсів і технічної експертизи порівняно з іншими методами.

4. Headless Commerce. «Headless Commerce означає відокремлення frontend-додатків електронної комерції від backend [6]. Архітектура, яка відокремлює презентаційний рівень інтерфейсу від внутрішньої функціональності електронної комерції дозволяє компаніям створювати висококастомізовані інтерфейси з використанням сучасних технологій, таких як React або Vue.js, одночасно використовуючи спеціалізовані API-інтерфейси електронної комерції або мікросервіси для бекенд-частини.

«API (від англ. Application programming interface – інтерфейс програмування додатків) – набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) для використання в зовнішніх програмних продуктах [7]». «Безголова» комерція пропонує гнучкість,

масштабованість і можливість надавати послідовний досвід на різних каналах, таких як веб, мобільні пристрої та пристрої інтернету речей.

5. Аутсорсинг. Компанії можуть віддати розробку своїх інтернет-магазинів на аутсорсинг спеціалізованим агентствам з веброзробки або фрілансерам. Цей варіант дозволяє бізнесу використовувати досвід професіоналів, які спеціалізуються на розробці електронної комерції. Аутсорсинг може бути економічно вигідним рішенням для бізнесу з обмеженими технічними ресурсами або специфічними вимогами до проєкту.

Для прикладу розглянемо платформу електронної комерції Shopify.

Shopify – це повноцінна комерційна платформа, яка дозволяє створювати, розвивати та керувати бізнесом [8]. Головні переваги Shopify:

1. Зручний інтерфейс, який дозволяє компаніям легко створювати свої інтернет-магазини. Він надає широкий вибір професійно розроблених тем, які можна налаштувати відповідно до ідентичності бренду. Теми адаптовані для мобільних пристроїв, що забезпечує безперебійний досвід покупок на всіх пристроях. Компанії також можуть налаштовувати дизайн, змінюючи кольори, шрифти і макети або створюючи власні теми за допомогою інструментів розробки тем Shopify.

2. Надійна система управління продуктами, яка дозволяє компаніям створювати та керувати своїм каталогом продуктів. Підтримує різні типи продуктів, включаючи фізичні товари, цифрові завантаження та послуги. Компанії можуть додавати описи продуктів, зображення, ціни та інформацію про відстеження запасів. Shopify також пропонує такі функції, як варіації продуктів, організація продуктів у колекції та інструменти пошукової оптимізації (SEO) для оптимізації видимості продуктів.

3. Безпечна система кошика для покупок, яка дозволяє клієнтам додавати товари до свого кошика, розраховувати вартість доставки та переходити до оформлення замовлення. Він підтримує кілька платіжних шлюзів, включаючи Shopify Payments, що дозволяє компаніям приймати платежі кредитними картками безпосередньо через платформу. Shopify також інтегрується зі

сторонніми платіжними шлюзами, надаючи бізнесу гнучкість у виборі найбільш підходящого варіанту для своїх клієнтів.

4. Інтуїтивно зрозуміла система управління замовленнями, яка дозволяє компаніям обробляти замовлення, виставляти рахунки, відстежувати статус замовлення та надсилати клієнтам автоматичні сповіщення про замовлення. Вона надає функції управління запасами, які допомагають компаніям відстежувати рівень запасів, налаштовувати автоматичне оновлення запасів та отримувати сповіщення про низький рівень запасів. Shopify також інтегрується з транспортними перевізниками, щоб оптимізувати виконання замовлень і генерувати транспортні етикетки

5. Платформа має великий магазин додатків, який пропонує широкий спектр розширень для покращення функціональності інтернет-магазинів.

Компанії можуть вибирати з різних додатків, які надають додаткові функції, такі як маркетинг електронною поштою, інтеграція з соціальними мережами, відгуки клієнтів, відновлення покинутих кошиків і розширена аналітика. Магазин додатків дозволяє компаніям налаштовувати свої інтернет-магазини відповідно до конкретних потреб та галузевих вимог.

6. Shopify – це хмарна платформа, що означає, що компаніям не потрібно турбуватися про хостинг або управління серверами. Платформа забезпечує надійну продуктивність, швидке завантаження сторінок і легко справляється зі стрибками трафіку. Shopify створений для роботи з компаніями будь-якого розміру, від невеликих стартапів до підприємств, забезпечуючи масштабованість і здатність рости разом з бізнесом.

7. Цілодобова підтримка клієнтів через різні канали, включаючи чат, електронну пошту та телефон. Він надає вичерпну документацію, навчальні посібники та інструкції, які допомагають бізнесу орієнтуватися та максимально використовувати функції платформи. Shopify також має активний форум спільноти, де користувачі можуть спілкуватися, ділитися досвідом та шукати порад від колег-підприємців.

Головні недоліки Shopify:

1. Структура витрат працює за моделлю, що базується на підписці, а це означає, що підприємства повинні сплачувати щомісячну плату за використання платформи. Хоча існують різні тарифні плани, витрати можуть зростати, особливо для малого бізнесу або стартапів з обмеженим бюджетом. Крім того, при використанні сторонніх платіжних шлюзів замість Shopify Payments може стягуватися комісія за транзакції, що може додатково вплинути на загальні витрати.

2. Обмеження кастомізації. Хоча Shopify пропонує широкий вибір тем і варіантів дизайну, що налаштовуються, можуть існувати обмеження, коли мова йде про повну свободу дизайну.

Для налаштування зовнішнього вигляду та функціональності певних елементів можуть знадобитися знання кодування або допомога розробника. Компанії з унікальними вимогами до дизайну або складними потребами в кастомізації можуть виявити, що можливості Shopify дещо обмежені.

3. Залежність від додатків. Магазин додатків Shopify надає широкий спектр розширень та інтеграцій для розширення функціональності інтернет-магазину.

Однак, значна залежність від сторонніх додатків може збільшити витрати, оскільки багато з них мають додаткову плату. Крім того, використання великої кількості додатків може потенційно вплинути на загальну продуктивність і швидкість завантаження вебсайту.

4. Контроль даних. Як хостингова платформа, Shopify зберігає контроль над базовою інфраструктурою та серверами. Хоча вони забезпечують безпеку даних і резервне копіювання, бізнес має менший контроль над своїми даними порівняно з рішеннями, розміщеними на власному хостингу.

Залежність від інфраструктури Shopify означає, що компанії покладаються на здатність платформи підтримувати конфіденційність і безпеку даних.

5. Обмежений контроль SEO. Shopify надає базові функції SEO та інструменти оптимізації, рівень контролю та кастомізації може бути обмеженим у порівнянні з автономними системами управління контентом. Компанії можуть

мати меншу гнучкість у впровадженні передових методів SEO або оптимізації певних елементів свого вебсайту для пошукових систем.

6. Проблеми міграції платформ. Якщо бізнес вирішить перейти на іншу платформу електронної комерції в майбутньому, міграція інтернет-магазину з Shopify може бути складним завданням.

Хоча Shopify надає інструменти для експорту певних даних, можуть виникнути складнощі з перенесенням інформації про товари, налаштувань SEO та даних клієнтів на нову платформу. Процес міграції може вимагати технічних знань і ретельного планування для забезпечення плавного переходу.

7. Обмежені багатомовні можливості. Вбудовані багатомовні можливості Shopify дещо обмежені, доступні мови: англійська, французька, німецька, іспанська, бразильська португальська, італійська, японська та голландська.

Хоча він пропонує варіанти перекладу для певних елементів, створення повністю локалізованого багатомовного магазину може вимагати використання сторонніх додатків або індивідуальної розробки. Це може додати складнощів і витрат для компаній, які орієнтуються на кілька мовних ринків.

Нижче наведено демонстрацію вигляду потенційного сайту, який розробляється, використовуючи Shopify (рис. 1.1).

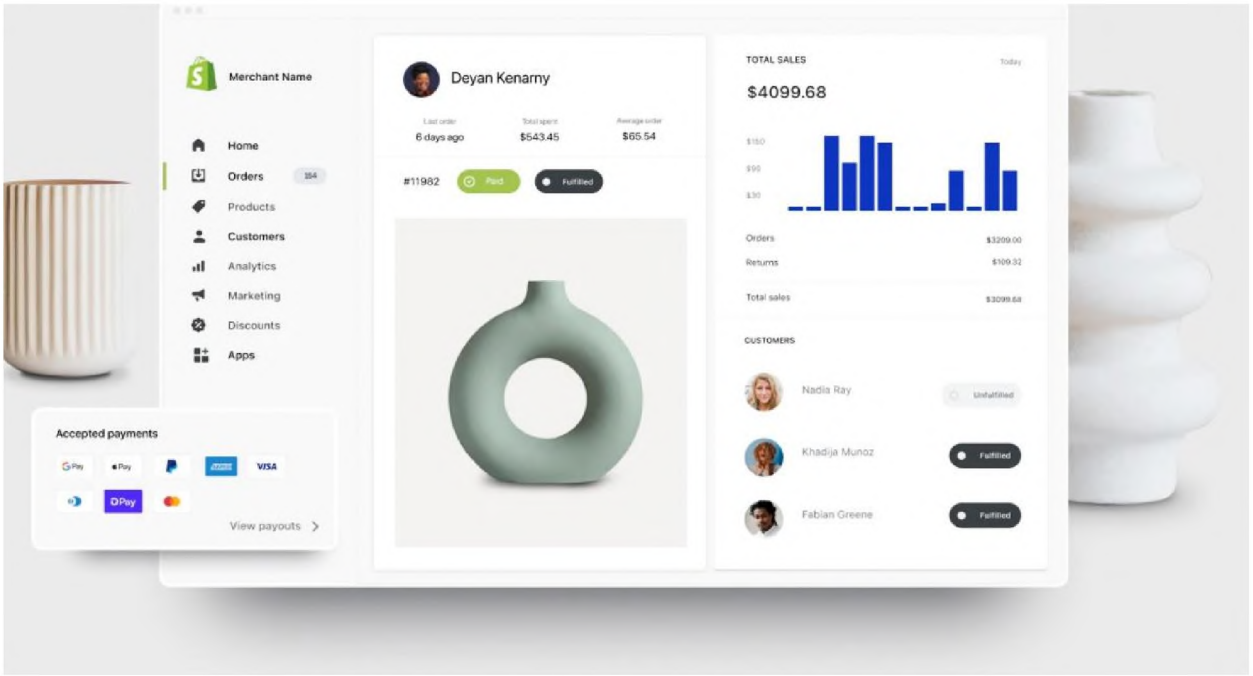


Рисунок 1.1 – Приклад збору даних Shopify [9]

Отже, Shopify пропонує низку переваг для бізнесу, який прагне створити та керувати своїми інтернет-магазинами. Зручний інтерфейс, широкий вибір тем, надійні функції для управління продуктами та виконання замовлень роблять його популярним вибором серед платформ електронної комерції. Крім того, наявність численних додатків та інтеграцій дозволяє компаніям налаштовувати свої інтернет-магазини відповідно до конкретних потреб.

Однак важливо враховувати і потенційні недоліки використання Shopify. Структура витрат, обмеження кастомізації, залежність від додатків та обмежений контроль над даними і SEO – це фактори, які компанії повинні зважити на переваги платформи. Крім того, при розгляді Shopify як довгострокового рішення слід враховувати проблеми, пов'язані з міграцією платформи та обмеженими багатомовними можливостями.

Сучасні технології забезпечення адаптивності вебсайту

Адаптивність сайту дає можливість комфортно, в зручному вигляді переглядати вебсайт на будь-якому пристрої. Це забезпечується шляхом використання так званих «брейкпоінтів» – певних точок «зломів» сторінки, коли ширина вікна браузера стає менше певного значення. Наприклад, Full HD монітори ПК мають ширину екрану 1920 пікселів, більш старі монітори, зазвичай, 1440 пікселів в ширину. Піксель – абсолютна величина у веброзробці, тому відносно неї можна чітко зрозуміти як саме варто відображати сторінку при тій, чи іншій ширині вікна.

Автор статті «100 % правильний спосіб робити адаптивні брейкпоінти в CSS» [10] надає на розгляд малюнок з точками і пропонує поєднати їх у групи, потім поєднує їх у зелені овали, таким чином позначивши зони ширин екрану, після чого додає їх на шкалу з написами, що визначають ширину девайса. Потім автор розділяє ці зони на прямокутники, замість овалів, що вже більше схоже на екран якогось девайса. В результаті отримуємо картинку з візуальним

позначенням розмірів екранів девайсів в цілому, що полегшує розуміння точок «зломів» і дає змогу розуміти, з якого місця потрібно робити перехід на новий девайс. Так як більшість людей користується смартфонами та персональними комп'ютерами, можна звернути на них окрему увагу при розробці.

На рис. 1.2 також приведена статистика використання девайсів різної ширини екрану, отже можна побачити тенеднції по рокам та який відсоток людей користується тим чи іншим типом девайсів [11]. За даними цієї компанії в 2017 р. було подолано позначку 50 % надходження інтернет-трафіку через малоекранні девайси. Надалі цей показник постійно зростає.

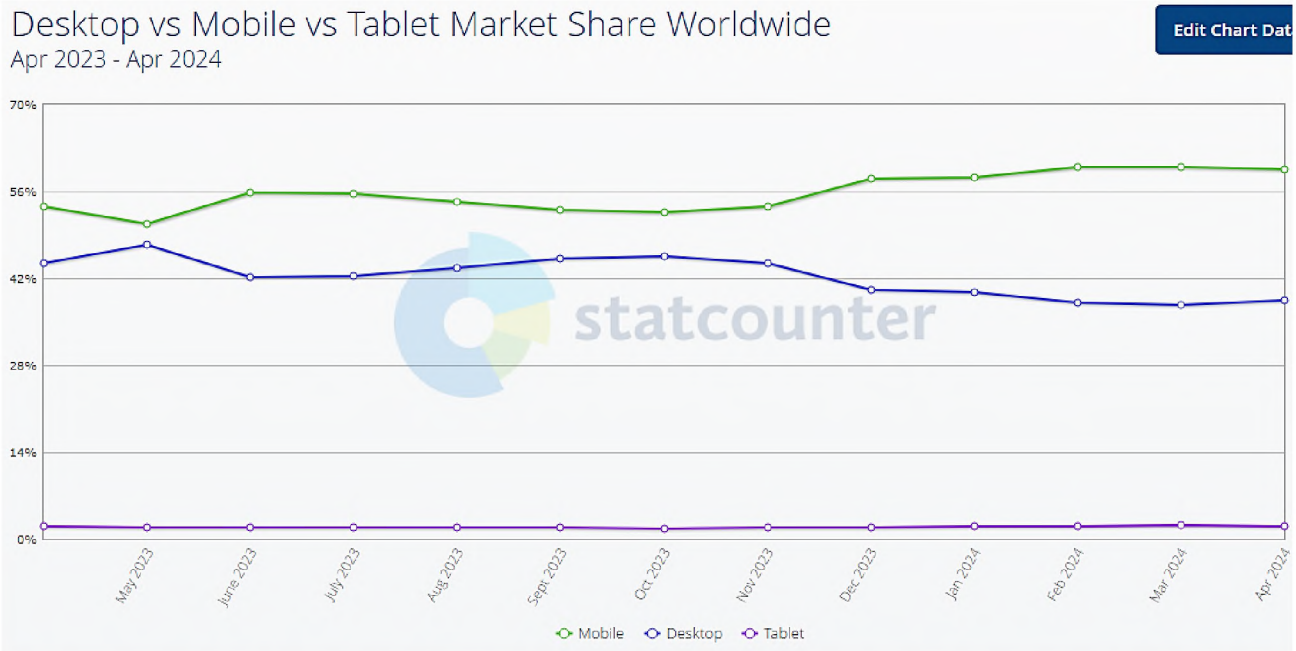


Рисунок 1.2 – Статистика використання девайсів різної ширини екрану [11]

При адаптації потрібно враховувати, що розмір зображення на великому екрані може займати різний відсоток площі при різній роздільній здатності екрану. Така невизначеність може створювати проблеми з зображенням, тому подавати розміри краще у відносних одиницях.

Кількість дій, що потрібні для адаптивності, також можна значно скоротити, якщо використовувати відносні одиниці виміру. Піксель – абсолютна одиниця, 10 пікселів це завжди 10 пікселів, а от, наприклад, відсотки відносні – вони залежать від ширини батьківського елементу. Таким чином 50 % – це

половина батьківського блоку. Якщо нам потрібно відштовхуватись від ширини або висоти екрану ми можемо використати інші одиниці виміру – vw та vh. Перша залежить від ширини, друга – від висоти. Отже, якщо ми задаємо блоку розмір 100vh, то його висота завжди займатиме всю висоту екрану користувача, незалежно від типу пристрою. Ці самі одиниці виміру працюють і зі звичайним текстом, так що можна не зменшувати його розмір на кожному етапі, а зробити це лише один або кілька разів. На рис. 1.3 наведено статистичні дані за популярністю різних роздільних здатностей екранів пристроїв та частота їх використання [12].

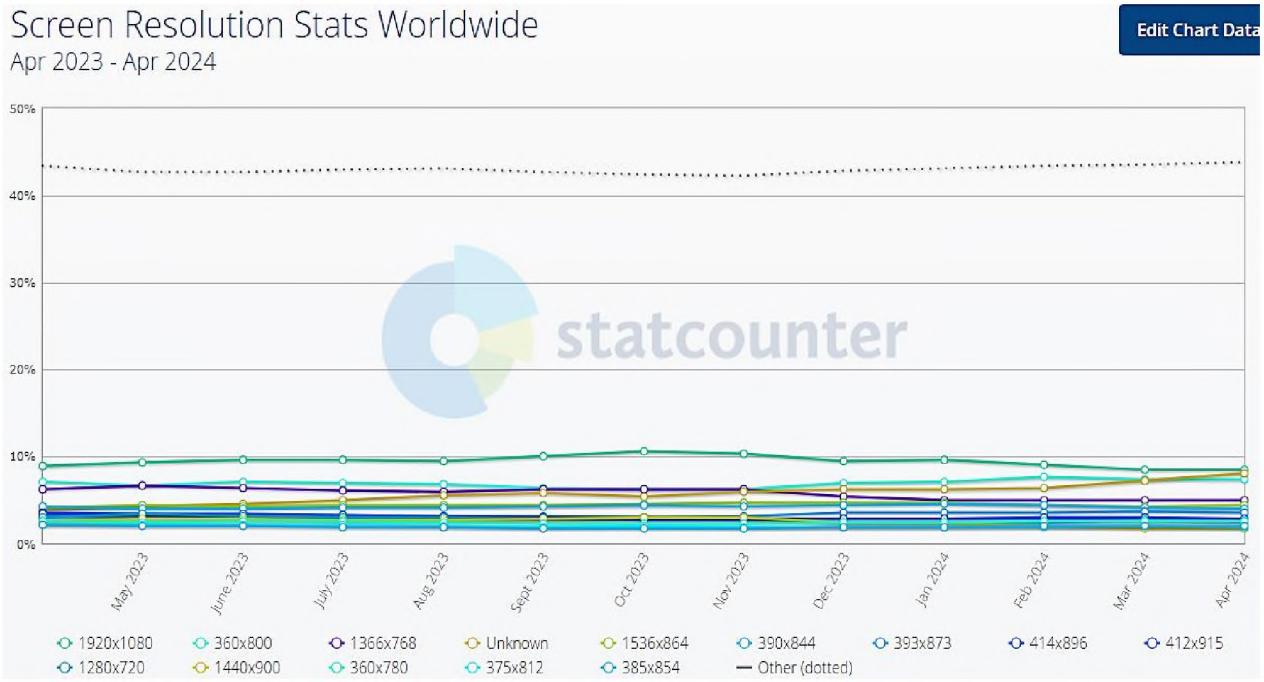


Рисунок 1.3 – Статистика використання екранів з різними параметрами роздільної здатності в світі

Немаловажливим фактором є використання так званої гнучкої верстки на флексбоксах (від англ. flexible – гнучкий) з униканням задання чітких розмірів і відступів, здебільшого у блоків. Гнучка верстка передбачає, зокрема, використання властивостей так званих флексбоксів (Flex-box), за допомогою яких легко можна керувати вмістом блоку, напрямком розташування елементів в ньому, відстанню між ними, їх вирівнюванням та інше [13].

Важливою деталлю у розробці сучасних сайтів є доступність для людей усіх категорій, в тому числі людей з вадами зору, такими як дальтоніки, люди з поганим зором, або люди, що взагалі не бачать. Про таких людей теж не можна забувати, і, якщо стоїть мета розробити сучасний, доступний сайт, то ми маємо це враховувати. Наразі існують посібники для вебу з метою підвищення доступності контенту: міжнародний стандарт WCAG2.0 для користувачів з різними обмеженнями здоров'я. Вони розроблялися для людей з порушеннями здоров'я, однак використання тих самих принципів підвищить рівень комфорту в роботі з сайтом та здорових людей. Адже люди можуть просто втомитися або читати сайт з маленького телефону з темним екраном, на якому погано видно текст [14].

Шрифт потрібно задавати у спеціальних відносних одиницях – rem та em, перша за замовчуванням дорівнює 16 пікселів, для цього варто розрахувати скільки rem у шрифтах, заданих на макеті, для цього я створив змінні, яким перерахував ці значення, таким чином 18 пікселів дорівнює 1,125 rem (рис. 1.4), текст, розмір якого заданий за допомогою цих одиниць виміру, користувач може з легкістю збільшити в налаштуваннях браузера, на відміну від пікселів, що номінально не збільшуються і не зменшуються.

```
//font variables
$text-16: 1rem;
$text-18: 1.125rem;
$text-20: 1.25rem;
$text-24: 1.5rem;
```

Рисунок 1.4 – Змінні, з перерахуванням розмірів у одиниці rem

Також за стандартом WCAG2.0 по сайту має бути реалізована навігація за допомогою клавіатури, зазвичай це можна зробити клавішою Tab, до того ж сайт має бути зацикленим і дійшовши до кінця, користувач повинен повертатись на початок сайту, а не потрапляти в тупик.

РОЗДІЛ 2
ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ТА МЕТОДИ РОЗРОБЛЕННЯ ДИЗАЙНУ І ВЕРСТКИ ВЕБСАЙТУ

Обгрунтування вибору програмного середовища розробки вебсайту

Програмісти багато часу проводять за написанням та налагодженням коду. Щоб написати робочий код, вистачить простої програми, яка вміє редагувати текстовий вміст файлу, наприклад, NotePad++. Але так писати код складно, довго, і в процесі розробки допускається безліч помилок, які важко знайти. Існують як платні редактори, і безкоштовні. На сьогоднішній день найбільш популярними та потужними з них вважаються Visual Studio Code, Sublime Text, Atom, WebStorm [15].

Сучасні редактори коду вміють підсвічувати синтаксис на будь-якій мові програмування, виставляти автоматичні відступи, доповнювати код, скорочуючи процес розробки, розділятись на кілька робочих областей, встановлювати велику кількість розширень, налаштовувати зовнішній вид та інструменти, мають консоль, дебагер та систему контролю версій, як от Git.

Але в сучасному світі з'явилася величезна кількість засобів для розробки, різноманітних текстових редакторів та довершених інтегшрованих середовищ розробки кодів (IDE), тому було б доцільно виділити основні переваги цих застосунків та відібрати їх для порівняння за допомогою оцінки їх балами за результатами статистики відгуків та проведеного якісного аналізу (табл. 2.1). На початку розглянемо детальніше характеристики порівнюваних застосунків:

– Visual Studio Code – це легкий, але потужний редактор вихідного коду, який працює на вашому робочому столі та доступний для Windows, macOS та Linux. Він поставляється з вбудованою підтримкою JavaScript, TypeScript і Node.js, а також має багату екосистему розширень для інших мов (наприклад, C++, C#, Java, Python, PHP, Go) і середовищ виконання (наприклад, .NET і Unity) [16].

Visual Studio Code це простий та елегантний редактор коду для проєктів різноманітної складності, розробників будь-якого рівня. Він відносно «важкий» і при цьому дуже гнучко налаштовується під будь-які запити розробника, незалежно від напрямку його діяльності. Має зручний інтерфейс на будь-який смак з можливістю вибору тем, від світлої до самої темної, систему контролю версій, дебагер, автоматичний пошук помилок в коді. IntelliSense – автодоповнювач коду у зв'язці з Emmet значно пришвидшує процес розробки, іноді достатньо написати лише 1 символ і редактор вже запропонує вам потрібну підказку. Саме ця IDE буде використана в ході розробки випускної кваліфікаційної роботи;

– Sublime Text 3 – це кроссплатформений текстовий редактор, розроблений для верстальщиків та програмістів і дозволяє використовувати від Erlang до C-подібних мов програмування. Він досить мінімалістичний, в редакторі відсутні панель інстументів та діалогові вікна. Але при детальному перегляді у ньому можна знайти плагіни, розумне автодоповнення коду, виділення багатьох елементів відразу, що дозволяє швидко замінювати назви змінних чи файлів, функція полегшення доступу до файлів та багато інших корисних функцій. Sublime Text досить простий в засвоєнні і підійде програмісту з будь-якими задачами, також виділяють швидкість роботи редактора через його «легкість», адже він не нагромаджений зайвими функціями, доки вони вам не знадобляться.

Даний редактор має відкритий вихідний код і є безкоштовним, але є можливість покупки платної версії, але зазвичай безкоштовної версії достатньо для легкої та комфортної розробки.

Серед недоліків можна виділити хіба що баги, які іноді виникають з певними плагінами, тому варто уважно підходити до їх вибору;

– WebStorm – це інтегроване середовище для розробки на JavaScript та пов'язаних з ним технологіях. Як і інші IDE JetBrains, WebStorm дозволяє автоматизувати рутинну роботу та легко справлятися зі складними завданнями, роблячи розробку більш цікавою [17]. Він, як і всі сучасні IDE, забезпечує

автодоповнення коду, його аналіз прямо по ходу розробки, навігацію по коду, інтеграцію з системами контролю версій, такими як Git та рефакторинг коду. Рефакторинг реалізований якісно і надає змогу змінювати код JavaScript і HTML в різних файлах та папках проекту, що значно полегшує налагодження роботи додатку, що розробляється;

За замовчуванням WebStorm не має таких можливостей, що описані вище, але в ньому інстальовані більше 100 різних доповнень, які забезпечують легку підтримку і зручну розробку завдяки використанню фреймворків, бібліотек і плагінів. Також, заслугою даного редактора коду є так званий Live Edit, що відразу інстальований в ньому, він дозволяє зберігаючи файл, відразу бачити внесені зміни, наприклад на сайті, що досить вагомо полегшує розробку і зменшує кількість непотрібних дій розробника.

Звичайно, редактор підтримує усі сучасні мови програмування останніх версій, мови серверного взаємозв'язку, такі як Node.js, Maria.db та інші. Дана IDE є взірцем якісного продукту, але при цьому є досить навантаженою, тобто «важкою» і за доступ до більшості функцій потрібно буде купити платну підписку, що коштує $129.00(близько 3812 гривень) на рік, але це не висока ціна для такого роду застосунку.

– Adobe Dreamweaver – візуальний HTML-редактор Adobe. Редактор не користувався особливою популярністю у розробників, але в версії, що недавно вийшла, з'явилося безліч додаткових можливостей, таких як сучасний користувальницький інтерфейс і гнучкий механізм для швидкого написання коду. Ці функції спрощують роботу веб-дизайнерів і розробників інтерфейсу користувача, дозволяючи створювати проекти, писати код і керувати веб-сайтами, які чудово виглядають на будь-якому екрані [18]. Серед явних плюсів редактора можна виділити реалізацію власного «Live Edit», як в IDE WebStorm, що надає змогу розробляти в режимі реального часу і бачити зміни безпосередньо без оновлення сторінки браузера. Також присутні синтаксичні підказки, перевірка помилок, як і інших редакторах а також попередньою обробкою CSS-коду.

Серед недоліків є такі вагомі, як платність рішення, адже Adobe Dreamweaver, як і інші додатки компанії Adobe, входить в пакети, що купуються разом з іншими, іноді не потрібними вам функціями. Конкретно ця IDE входить до пакету Adobe Creative Cloud, що також включає в себе Photoshop, Adobe Stock та інші програмні рішення;

– Vim – це спеціалізований текстовий редактор програмістів. Він призначений для роботи з великими обсягами коду без миші [19]. Саме вилучення миші з процесу розробки є головною перевагою даного текстового редактора, адже для того, щоб зробити якусь дію в проєкті не потрібно відривати руки від клавіатури, що значно пошвидшує швидкість розробки. Для того щоб наприклад видалити рядок коду потрібно відірвати руку від клавіатури, взяти мишу, навести її на потрібний рядок, виділити його, натиснути кнопку Delete та повернути руки на клавіатуру. Цей процес досить сильно сповільнює процес розробки, тому в Vim можна це зробити просто за домопогою клавіатури певною комбінацією дій, що при наявності досвіду роботи з ним займає в рази менше часу, аніж звичний всім спосіб з мишою. Звичайно, цей редактор коду більше підходить розробникам на Java або C-подібних мовах, але для верстальщика він також може бути корисний.

На підтвердження цього твердження у статті про Vim наводиться такий вислів: «…користувач Vim прибирає руки з клавіатури, щоб поїсти або почухати ріпу. Решту часу він короткими командами обробляє текст у кілька разів швидше, ніж будь-яка людина з мишею.

Після звикання робота в Vim можна порівняти з грою на гітарі: ви керуєте текстом, ніби він продовження вашої руки. Фільми про хакерів бачили? Ось такий ефект [19]».

Серед недоліків звичайно можна виділити важкість в освоєнні, адже редактор є свого роду унікальним і щоб розробляти за допомогою нього потрібно вивчити цілий ряд команд, також плагіни працюють гірше, ніж в інших, більш продвинутих IDE, також відсутня функція автодоповнення коду, що може призвести до синтаксичних помилок у новачків. Саме тому дане рішення

підходить лише для досвідчених розробників, які досконально знають синтаксис і можуть передбачити помилки, що можуть виникнути в ході розробки продукту.

Отже серед основних переваг сучасних редакторів коду можна виділити наступні:

– безкоштовність; з малої літери всі пункти
– автодоповнення коду;
– можливість редагування в режимі реального часу або, так званий, Live Edit;
– зручність;
– простота інтерфейсу;
– гнучкість;
– простота освоєння;
– практичність.

Далі заносимо оцінки виділених характеристик редакторів коду в табл. 2.1 і визначаємо бали за шкалою від 1 до 10, де 1 – абсолютно не задовільно, а 10 – повністю відповідає на основі досконалості даних переваг у редактора.

Таблиця 2.1 – Порівняльна характеристика середовищ розробки

Критерії порівняння	Бальна оцінка за критеріями кожного із середовищ розробки				
	VS Code	Sublime Text	WebStorm	Dreamweaver	Vim
Безкоштовність	10	9	5	5	10
Автодоповнення коду	10	9	10	9	5
Live Edit	10	10	10	9	9
Зручність	10	8	10	8	10
Простота інтерфейсу	8	8	7	8	6
Гнучкість	10	8	10	7	8
Простота освоєння	9	9	9	9	4
Практичність	9	8	10	8	10
Середній бал якості	9.5	8.63	8.88	7.88	7.75

Отже, після розрахунку середнього балу якості найбільше переваг має редактор коду Visual Studio Code (результати – 9,5).

При розрахунку було використано спеціальний інструмент формул в текстових таблицях (рис. 2.1).

Критерії порівняння	Бальна оцінка за критеріями кожного із середовищ розробки		Vim
	VS Code		
Безкоштовність	10		10
Автодоповнення коду	10		5
Live Edit	10		9
Зручність	10		10
Простота інтерфейсу	8		6
Гнучкість	10		8
Простота освоєння	9		4
Практичність	9		10
Середній бал якості	9.5		

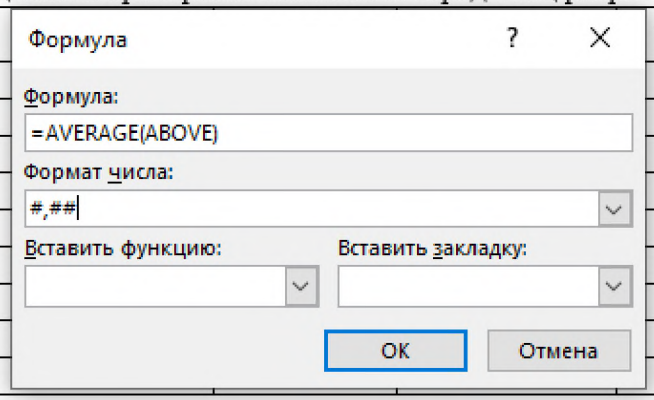


Рисунок 2.1 – Використання інструменту «Формула» в текстових таблицях

Отже, за результатами проведеного аналізу визначено, що VS Code є найоптимальнішим варіантом редактору коду для розробника, через свою зручність, гнучкість та безкоштовність. Саме через ціну і простоту редакторів середній бал WebStorm та Vim значно знизився, хоча без сумніву всі розглянуті вище редактори є гарними програмними засобами і доцільні для використання, але в індивідуальних умовах. Ключові можливості та характеристики Visual Studio Code [20] розглянемо більш детально по пунктах.

1. Інтерфейс користувача. VS Code має чистий та інтуїтивно зрозумілий користувацький інтерфейс з бічною панеллю для навігації, центральною областю редагування та рядком стану для відображення інформації та інструментів. Він забезпечує вільний робочий простір, дозволяючи розробникам зосередитися на своєму коді (рис. 2.2 – 2.3).

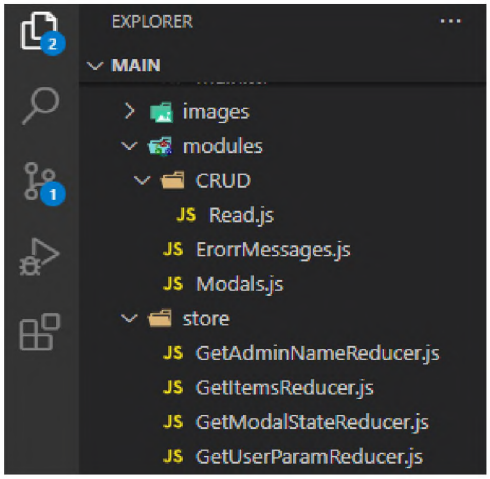


Рисунок 2.2 – Вигляд бічної панелі Visual Studio Code

Бічна панель Visual Studio Code забезпечує швидкий доступ до різних функцій та можливостей редактора коду. Бічна панель містить іконки навігації, такі як Провідник, Розширення та інші, що дозволяють користувачам легко перемикатися між різними файлами та інструментами в редакторі.

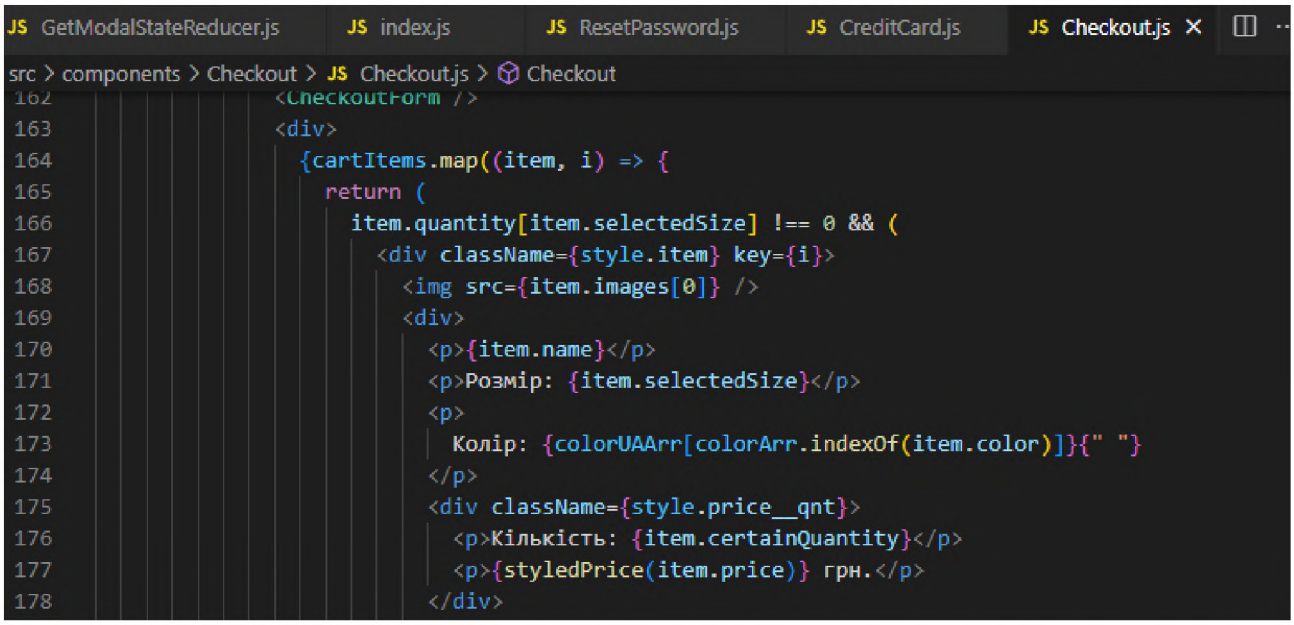


Рисунок 2.3 – Вигляд центральної області Visual Studio Code (фрагмент)

2. Крос-платформна сумісність. Visual Studio Code доступний для Windows, macOS та Linux, що забезпечує сумісність та однакову роботу з різними операційними системами.

3. Розширення та маркетплейс. Однією з визначних особливостей VS Code є його широка екосистема розширень. Розробники можуть налаштовувати та покращувати свій досвід кодування, встановлюючи широкий спектр розширень для різних мов програмування, фреймворків та інструментів. На Visual Studio Code Marketplace розміщено величезну колекцію розширень, створених спільнотою.

4. Мовна підтримка. Комплексна мовна підтримка для багатьох мов програмування, включаючи такі популярні, як JavaScript, Python, C++, Java, HTML, CSS та багато інших. Вона забезпечує підсвічування синтаксису, завершення коду, форматування, розбиття на рядки та можливості налагодження, специфічні для кожної мови.

5. Інтегрований термінал. Дозволяє розробникам запускати команди, компілювати код і виконувати різні завдання без перемикання на окремий термінальний додаток. Це полегшує безперебійний робочий процес розробки в самому редакторі (рис. 2.4).

Рисунок 2.4 – Вигляд інтегрованого терміналу Visual Studio Code

6. Інтеграція з Git. VS Code має вбудовану інтеграцію з Git, що дозволяє контролювати версії та співпрацювати з колегами по команді (рис. 2.5). Розробники можуть легко переглядати та керувати репозиторіями Git'a, фіксувати зміни, переглядати різниці та виконувати операції з Git'ом, не виходячи з редактора [21].

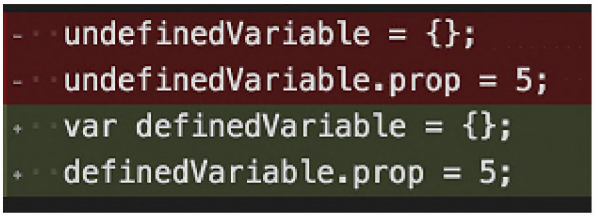


Рисунок 2.5 – Вигляд різних версій проєкту Visual Studio Code

7. Можливості налагодження. Visual Studio Code забезпечує надійну підтримку налагодження для різних мов і платформ. Вона пропонує покрокове налагодження, перевірку змінних та інші важливі функції налагодження, які допомагають розробникам виявляти та виправляти проблеми в коді.

8. IntelliSense. Інтелектуальна функція завершення коду у VS Code, яка надає контекстно-залежні пропозиції під час введення коду (рис. 2.6). Вона

пропонує автозавершення, підказки щодо підписів функцій та документацію, щоб пришвидшити кодування та зменшити кількість помилок [22].

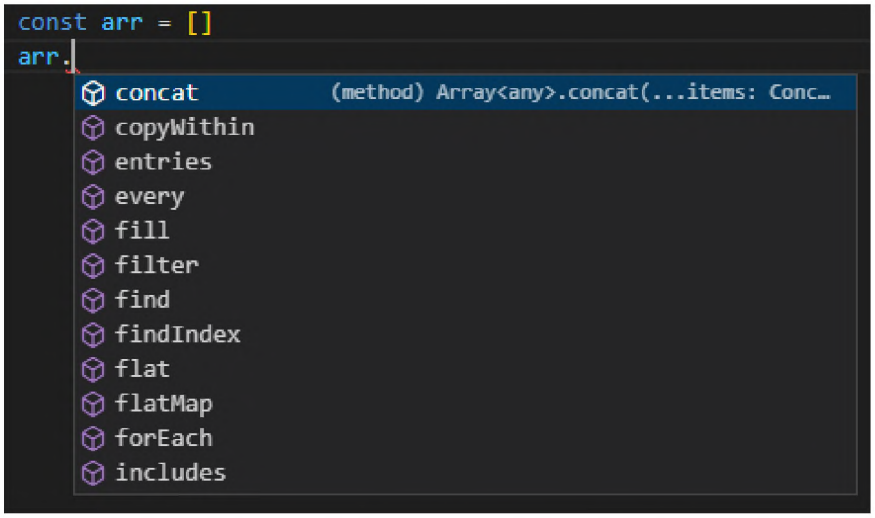


Рисунок 2.6 – Вигляд IntelliSense Visual Studio Code

9. Автоматизація завдань. VS Code підтримує автоматизацію завдань за допомогою інтегрованого бігуна завдань. Розробники можуть визначати власні завдання збірки, запускати скрипти і виконувати різні автоматизовані завдання в редакторі, підвищуючи продуктивність і ефективність робочого процесу.

10. Кастомізація. Visual Studio Code дуже добре налаштовується, що дозволяє розробникам персоналізувати своє середовище кодування. Користувачі можуть змінювати теми, встановлювати розширення, налаштовувати параметри та створювати власні комбінації клавіш, щоб пристосувати редактор до своїх уподобань [23].

Загалом, Visual Studio Code надає потужне та гнучке середовище кодування з багатим набором функцій, розширень та можливостей налаштування, що робить його популярним вибором для розробників усіх рівнів. Його легкість, широка мовна підтримка та динамічна екосистема сприяють його широкому розповсюдженню у спільноті розробників.

Опис обраних мов програмування та додаткових засобів розробки

В ході виконання кваліфікаційної роботи для проєкту були обрані такі технології як HTML, CSS та JavaScript, а також бібліотеки React, Redux, Firebase, Cloudinary і препроцесори до них.

HTML – це код, який використовується для структурування вебсторінки та її вмісту. Наприклад, вміст може бути структурований в межах набору абзаців, списку маркованих пунктів або за допомогою зображень і таблиць даних [18]. Іншими словами можна сказати, що HTML це основа (скелет) сайту, що визначає його структуру та складові деталі. Він складається з елементів, або так званих тегів, що містять в собі певний контент або атрибути. Атрибути можуть змінювати або доповнювати функції елемента, вони поділяються на обов'язкові й необов'язкові. Наприклад, посилання у вигляді тегу <a> не діє без атрибута href, що містить в собі саме посилання. Контент, у свою чергу, розміщують тільки в парні теги, що відкриваються і закриваються. Для прикладу візьмемо той самий тег посилання <a>, зазвичай він має такий вигляд – Назва посилання, де «Назва посилання» і є контент (вміст) тегу. В редакторі коду це має такий вигляд (рис. 2.7):

```
<a href="https://www.pdau.edu.ua/">Посилання</a>
```

Рисунок 2.7 – Приклад запису парного тегу-посилання

Також теги бувають непарні, наприклад, мета-теги: вони не мають контенту, лише атрибути, від яких і залежить їхній функціонал, наприклад, кодування сторінки (рис. 2.8):

```
<meta charset="UTF-8">
```

Рисунок 2.8 – Приклад непарного мета-тег кодування файлу

Саме з такого роду тегів, як описано вище, складається структура сайту, за допомогою різних рівнів вкладеності і взаємозв'язку між ними. Приклад базової структури сторінки наведено на рис. 2.9.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <a href="https://www.pdau.edu.ua/">Посилання</a>
</body>
</html>
```

Рисунок 2.3 – Базова структура вебсторінки

CSS – мова каскадних таблиць стилів (від англ. Cascading Style Sheets), яка використовується для опису подання документа, написаного в HTML або XML (включаючи діалекти XML, такі як SVG, MathML або XHTML). CSS описує, як елементи мають відображатися на екрані, на папері, у мовленні або на інших носіях [19].

Якщо HTML можна умовно уявити скелетом сайту, то каскадна таблиця стилів це його шкіра, та все інше, що знаходиться поверх каркасу. Саме вона визначає який вигляд має сайт, дає можливість гнучко його налаштовувати та навіть додавати нові елементи, не змінюючи основний HTML документ. Нові версії CSS мають безліч нових функцій, властивостей. Так що інколи можливо застосувати CSS замість використання JavaScript у створенні певних елементів сайту, наприклад створити меню з випадаючими списками (рис. 2.10), аккордеон чи випливаючу підказку.

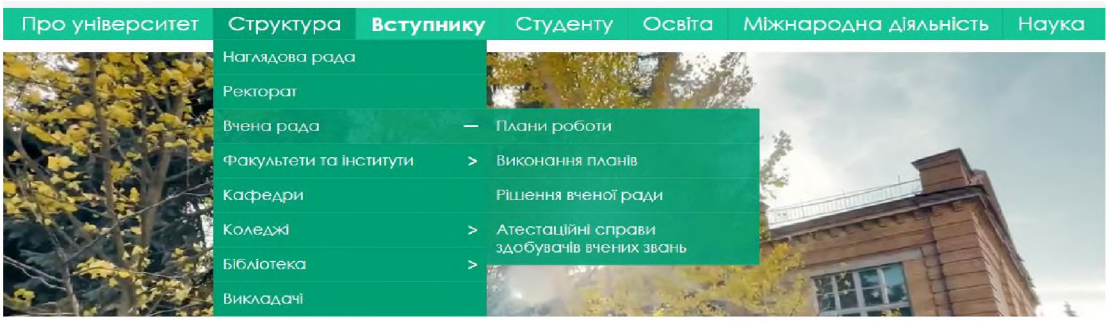


Рисунок 2.10 – Приклад випадаючого меню на вебсайті ПДАУ

Приклад коду меню, зображеного на рисунку 2.10, можна розібрати за структурою та зрозуміти, навіщо використовувались ті, чи інші контейнери. Наприклад, на верхньому рівні знаходиться обгортка всього списку, основних пунктів меню, що надає змогу розмістит їх у рядок та зробити відступи, як зображено в додатку А (рис. А.1).

На рівень нижче знаходиться пункт меню, при наведенні на який з'являється випадаюче меню, реалізовано це за допомогою комбінації CSS властивостей (додаток А, рис. А.2) та JS-функцій, зображених на рис. 2.11. Далі за ієрархією знаходиться посилання з назву пункту меню, натиснувши на яке може перемістити на сторінку з її вмістом. А на рівень нижче знаходить саме приховане меню, що початково не відображається на сторінкі за допомогою задання стилю CSS, а потім з'являється за допомогою JS.

```
let el = document.getElementsByClassName('menu-item');

for (let i = 0; i < el.length; i++) {
    el[i].addEventListener("mouseenter", showSub, false);
    el[i].addEventListener("mouseleave", hideSub, false);

}

function showSub(e) {
    if (this.children.length > 1) {
        this.children[1].style.height = "auto";
        this.children[1].style.overflow = "visible";
        this.children[1].style.opacity = "1";
    } else {
        return false;
    }
}

function hideSub(e) {
    if (this.children.length > 1) {
        this.children[1].style.height = "0px";
        this.children[1].style.overflow = "hidden";
        this.children[1].style.opacity = "0";
    } else {
        return false;
    }
```

Рисунок 2.11 – Приклад JavaScript-коду для відкривання випадаючого меню

Звичайно, за допомогою CSS можна робити різноманітні анімації для покращення user experience, адже користувачу завжди приємніше взаємодіяти з мінімалістичними анімаціями, аніж зі статичним сайтом.

JavaScript – мова програмування, яка дозволяє вам створювати динамічно оновлюваний контент, керує мультимедіа, анімує зображення. Це мова програмування, яка дозволяє вам застосовувати складні речі на вебсайті – кожен раз, коли на web сторінці відбувається щось більше, ніж просто її статичне відображення – відображення контенту, що періодично оновлюється, інтерактивних карт, анімація 2D/3D графіки, або прокручування відео в плеєрі і т.д. – можете бути впевнені, що, швидше за все, не обійшлося без JavaScript [24].

Отже, знову прибігаючи до аналогії з людиною, можна сказати, що JS це мізки людини, які за потреби говорять скелету та шкірі як потрібно себе поводити й рухатися при певних умовах. Також він дає можливість зберігати змінні локально, чи видалено, взаємодіяти з контентом сайту, замінюючи, чи коректуючи його, перебирати великі масиви даних, виводити потрібну користувачу інформацію на вебсайт, реагувати на дії користувача та багато інших корисних речей без яких важко уявити сучасний вебсайт.

Зазвичай, код JS завантажується на сайт після появи структури сторінки та його зовнішнього вигляду, адже якщо він спробує отримати якийсь елемент з DOM-вузла, а його на цей момент ще не буде у вікні браузера, то в кращому випадку ми отримаємо помилку. DOM є основним API-інтерфейсом на вебсайті.

Для того, щоб уникнути такої помилки можна підключити скрипт на початку HTML-документа і додати йому артибут «defer». Якщо встановлено атрибут defer, він вказує, що сценарій завантажується паралельно з аналізом сторінки та виконується після завершення аналізу сторінки [25]. Підключення такого скрипта зображено на рисунку 2.12.

```
<script src="js/script.js" defer></script>
```

Рисунок 2.12 – Синтаксис підключення скрипта JavaScript

API (Application programming interface) – це готові частини коду, в даному випадку на мові JS, які вже розроблені та впроваджені досвідченими розробниками для полегшення рутинної роботи іншим розробникам. Вони дозволяють нам використовувати певні надлаштування та готові методи над

JavaScript та значно полегшують роботу розробника з мовою JS. Мабуть улюблена метафора більшості розробників це «винаходити велосипед» – немає необхідності придумувати як реалізувати певний функціонал, який вже давно до нас реалізований, до того ж при розробці власними руками швидше за все виникнуть певні баги та так звані «костилі».

API-інтерфейс DOM (Document Object Model) дозволяє легко маніпулювати розміткою та стилями, створювати, змінювати чи видаляти вміст сайту, динамічно змінюючи його.

API Canvas, наприклад, дозволяє створювати анімовану графіку на сайті, легко маніпулюючи його елементами, за допомогою даного API можна дійсно цікаво та унікально прикрасити свій сайт, зробивши акценти в потрібних місцях або щось, що запам'ятається користувачу.

Окремої уваги заслуговують препроцесори до мов, що розглядаються – це Pug для HTML та SCSS для CSS. Процес написання HTML і CSS може виявитися дещо виснажливим і вимагати безлічі одних і тих же завдань знову і знову. Такі різні завдання, як правило, невеликі, знижують ефективність. На щастя, ці та кілька інших неефективних завдань були визнані і виклик ним кинули препроцесори. Що стосується HTML і CSS, одні з найпопулярніших мов препроцесора – це Haml і Sass. Haml перетворюється на HTML, а Sass перетворюється на CSS [26]. В свою чергу Pug є надлаштуванням над Haml, так званим шаблонізатором, але є його дещо покращеною версію і також слідує принципу Don't repeat yourself (DRY; з англ. – «не повторюйся») і сприяє добре структурованій розмітці, допомогаючи досвідченим розробникам працювати більш комфортно та уникати рутини (рис. 2.13).

```
html(lang="ru")
  head
    meta(charset="utf-8")
    title= "Учимся работать с шаблонизатором Pug"
  body
      h1 Hello, World!
```

Рисунок 2.13 – Приклад синтаксису коду на Pug

Pug покращує візуальне сприйняття коду для людини, знайомої з його синтаксисом. Sass це свого роду розширення, створене для спрощення каскадних таблиць стилів (CSS), це скриптова метамова, тобто мова програмування, що описує іншу [26]. Цей модуль також входить в Haml і має схожий з ним та Pug зовнішній вигляд, отже розробники можуть з легкістю розуміти синтаксис та структуру цих мов програмування.

Коли йдеться про Sass, як правило, ми маємо на увазі препроцесор та мову в цілому. Тим не менш, використовуючи Sass (препроцесор) ми можемо використовувати два різні синтаксиси (рис. 2.14):

– Sass (відступи);
– SCSS (CSS-подібний синтаксис).

SASS

```
$color: red
$color2: lime

a
  color: $color
  &:hover
    color: $color2
```

SCSS

```
$color: #f00;
$color2: #0f0;

a {
    color: $color;
    &:hover {
        color: $color2;
    }
}
```

CSS

```
a {
    color: red;
}
    a:hover {
        color: lime;
    }
```

Рисунок 2.14 – Різниця в синтаксисі SASS, SCSS та CSS

Отже, Sass має Ruby-подібний синтаксис, схожий на Pug, тобто для позначення вкладеності використовуються пробіли або таби, а SCSS більше схожий на CSS, в який він згодом звичайно компілюється і у файлі з розширенням .css можна з легкістю замінити розширення на .scss і він буде працювати так, як і до цього. Для заміни розширення файлу .scss в .css потрібно провести декілька більше маніпуляцій з кодом, але в цілому це робиться досить швидко та просто. Використання цих препроцесорів може дуже значно полегшити розробку верстальнику і позбавити його зайвої рутини та зробити весь процес розробки більш приємним.

React – це потужна бібліотека JavaScript, яка використовується для створення динамічних користувацьких інтерфейсів та досвіду. Розроблена та підтримувана Facebook з березня 2013 року. React широко використовується такими великими компаніями, як Instagram, Uber, Netflix, Twitter та багатьма іншими. Розробники цінують його за здатність створювати великі, швидкі та масштабовані вебдодатки.

«Одна з чудових частин React – це те, як з'являються додатки в міру їхнього створення. Якщо описувати цей процес у вигляді кроків, то отримаємо такий шлях [27]:

1. Декомпозиція UI у вигляді компонентів;
2. Відтворення даних без інтерактивності;
3. Визначення мінімальних станів для завдання інтерактивності;
4. Вибір компонентів, що володіють і змінюють стан;
5. Визначення зворотного потоку даних.

В основу React покладені компоненти, елементи, Virtual DOM і JSX.

Елементи – це об'єкти JavaScript, які представляють HTML-елементи (описують DOM-елементи). Вони є будівельними блоками React.

JSX – це препроцесор, що спрощує створення елементів і компонентів React. З ним потрібно набагато менше зусиль порівняно з класичним Java Script під час написання коду. JSX трансформується в JavaScript перед запуском у браузері. Він не є обов'язковим під час використання React. Робота з React без JSX комфортна і зручна, особливо якщо немає потреби налаштовувати компіляцію у своєму середовищі збірки.

Кожен елемент JSX є просто синтаксичним цукром для виклику «React.createElement (component, ...). Тому все, що можна зробити за допомогою JSX, також можна зробити і за допомогою звичайного JavaScript.

Virtual DOM – представляє легковажну копію звичайного DOM. І відмінною особливістю React є те, що ця бібліотека працює саме з віртуальним DOM, а не звичайним [28]. Компоненти в React дозволяють розділити користувацький інтерфейс на самодостатні частини, які можна використовувати

незалежно. Подібно до функцій JavaScript, компоненти приймають користувацькі дані (так звані пропси) і генерують React-елементи, які визначають візуальне представлення на екрані. Кожен компонент має властивості та стан, що забезпечує додаткову гнучкість та інтерактивність у додатку.

React надає ефективний спосіб керування станом елементів, використовуючи віртуальний DOM, що допомагає підвищити продуктивність. Завдяки декларативному API React розробники можуть зосередитися на описі бажаного результату, не турбуючись про зміни, що відбуваються в основі. Такий декларативний підхід спрощує розробку додатків, полегшуючи написання коду.

Однією з сильних сторін React є його здатність надавати детальний контроль над розміром додатку. Розробники можуть вибірково включати лише необхідні компоненти, забезпечуючи гнучкість при переході від односторонніх додатків (SPA) до мікросервісів. Це дозволяє повторно використовувати існуючі частини програми, що підвищує ефективність та масштабованість коду.

Крім більш швидкої розробки, перевикористання коду дає змогу уникнути великої кількості помилок. Якщо створити правильно спроєктовані та зручні компоненти, які планується використовувати в майбутньому знову, то потрібно буде писати менше коду, якщо з’явиться рішення створити за їхньою допомогою новий користувацький інтерфейс. У разі виникнення помилок зможете передбачити причину їхньої появи [18].

Завдяки перевикористанню коду стало набагато простіше створювати мобільні застосунки. Код, який був написаний під час роботи над сайтом, може бути повторно використаний для розробки мобільного застосунку. При цьому не потрібно наймати дві великі команди розробників.

Redux – це бібліотека управління передбачуваними станами для JavaScript-додатків, яка зазвичай використовується з такими фреймворками, як React та Angular. Вона забезпечує централізований підхід до управління станом додатку, що полегшує розуміння, тестування та підтримку складної логіки, пов'язаної зі станами [29].

Основна концепція Redux обертається навколо ідеї єдиного незмінного дерева станів. Весь стан програми зберігається в одному JavaScript-об'єкті, відомому як «сховище». Сховище відповідає за зберігання поточного стану і відправку дій для його зміни (рис. 2.15).

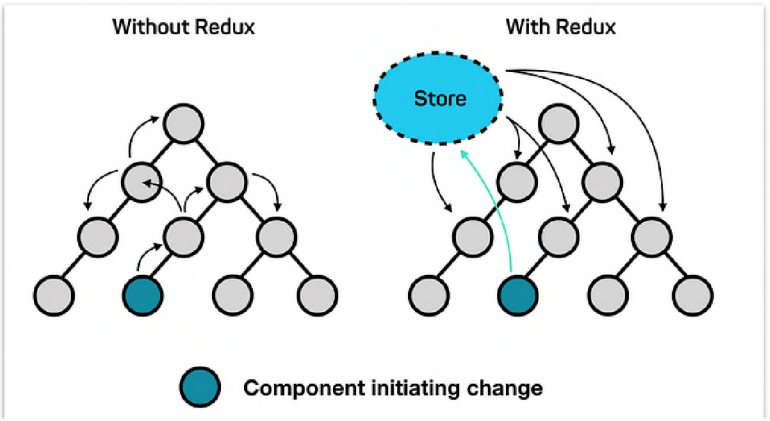


Рисунок 2.15 – Схематичне представлення, як працює Redux [30]

Стан в Redux не змінюється безпосередньо. Натомість зміни вносяться шляхом диспетчеризації дій, які є звичайними JavaScript-об'єктами з властивістю типу, що описує дію, яка виконується. Дії зазвичай викликаються взаємодією користувача або асинхронними операціями.

Редуктори відіграють важливу роль у Redux. Редуктор - це чиста функція, яка отримує на вхід поточний стан і дію, а на виході повертає новий стан на основі типу дії. Редуктори відповідають за незмінне оновлення стану, гарантуючи, що початковий стан залишається незмінним.

Редукція сприяє односпрямованому потоку даних. Коли виконується дія, вона проходить через редуктори, які відповідно оновлюють стан. Будь-які компоненти, зацікавлені у зміні стану, можуть підписатися на сховище і отримувати оновлення щоразу, коли стан змінюється.

Для підключення компонентів до сховища Redux у React-додатках зазвичай використовується бібліотека під назвою "react-redux". Вона надає функцію підключення, яка дозволяє компонентам отримувати доступ до стану зі сховища та відправляти дії.

Незважаючи на свої переваги, Redux може внести додаткову складність у невеликі або менш складні додатки. Він рекомендується для додатків зі значною кількістю станів або тих, що потребують розширених функцій управління станами.

«Firebase – це платформа для розробки додатків, яка допомагає створювати та розвивати програми та ігри, які подобаються користувачам. Підтримується Google та користується довірою мільйонів компаній по всьому світу [31]. Firebase здатна вирішити проблеми, описані вище.

Firebase надає комплексну платформу для розробки бекенд-систем, яка спрощує весь процес розробки та долає проблеми традиційної розробки бекенд-систем. Кілька ключових способів, за допомогою яких Firebase вирішує ці проблеми, описано нижче.

Синхронізація даних в режимі реального часу. База даних Firebase в режимі реального часу забезпечує безперебійну синхронізацію даних між декількома клієнтами в режимі реального часу. Розробники можуть легко оновлювати та отримувати дані, а зміни автоматично поширюються на всі підключені пристрої. Ця можливість роботи в режимі реального часу має вирішальне значення для таких додатків, як інструменти для спільної роботи, програми для чату та потокової передачі даних у реальному часі.

Спрощене управління інфраструктурою: Firebase позбавляє розробників необхідності налаштовувати та підтримувати сервери, керувати базами даних або працювати зі складними конфігураціями інфраструктури. Вона надає повністю керовану внутрішню інфраструктуру, що дозволяє розробникам зосередитися на створенні функцій та функціональності, а не турбуватися про управління серверами.

Автентифікація та залучення користувачів. Firebase пропонує надійну систему автентифікації користувачів, яка підтримує різні методи автентифікації, включаючи електронну пошту/пароль, логіни в соціальних мережах і сторонніх постачальників автентифікації. Це спрощує управління користувачами та контроль доступу, забезпечуючи безпечний і безперешкодний вхід для

користувачів. Firebase також включає такі функції, як хмарний обмін повідомленнями, push-повідомлення та персоналізовану взаємодію з користувачами, що покращує загальний користувацький досвід.

Масштабованість та продуктивність. Інфраструктура Firebase спроєктована таким чином, щоб легко масштабуватися для задоволення зростаючих потреб додатків. Вона автоматично управляє процесом масштабування, забезпечуючи високу доступність і продуктивність навіть під час пікових навантажень. Це позбавляє розробників необхідності керувати та налаштовувати додаткову інфраструктуру вручну, зменшуючи складнощі, пов'язані з масштабуванням бекенд-систем.

Cloudinary – це комплексна хмарна платформа для управління медіа, яка надає широкий спектр інструментів і сервісів для ефективного управління та доставки цифрових медіа-активів (рис. 2.9). Вона покликана спростити процес зберігання, оптимізації, маніпулювання та доставки зображень і відео для веб та мобільних додатків.

Однією з ключових особливостей Cloudinary є його потужні можливості управління зображеннями та відео. Він пропонує автоматичну оптимізацію зображень і відео, що дозволяє розробникам надавати високоякісні медіа з оптимальним розміром файлів і часом завантаження. Cloudinary підтримує різноманітні формати зображень і відео, а також може обробляти зміну розміру, обрізання, обертання та інші перетворення на льоту, що полегшує адаптацію медіа-активів до різних пристроїв і розмірів екранів.

Cloudinary також надає потужний набір функцій для керування медіа-активами. Він пропонує безпечне зберігання, резервне копіювання та керування версіями, гарантуючи, що медіа-активи захищені, до них можна легко отримати доступ і відновити за потреби. Крім того, Cloudinary пропонує безперешкодну інтеграцію з популярними системами управління контентом (CMS) та фреймворками для розробки, що дозволяє легко інтегрувати можливості управління медіа в існуючі робочі процеси.

Опис бізнес-моделі та процес макетування вебсайту

Для успішного функціонування вебсайту про переваги використання Proton Trade необхідна чітка бізнес-модель. Однією з можливих стратегій стала комісійна модель, де користувачі вебсайту оплачують комісію за використання трейд-бота та отримання прибутку від автоматичної торгівлі. Також можна розглянути модель підписки, де користувачі мають можливість обрати різні плани підписки залежно від їхніх потреб та отримувати додаткові функції та переваги. Завдання частин вебсайту – максимально чітко та зрозуміло й наочно показати вигоди від використання трейд-бота.

Для початку необхідно створити орієнтовний макет різних частин вебсторінки. Загальний план – структура у вигляді лендінгу. Макет комерційного вебсайту повинен бути розроблений таким чином, щоб забезпечити безперебійний і візуально привабливий досвід перегляду (UI-дизайн). Він повинен бути добре організованим, з чіткою ієрархією, яка дозволяє користувачам легко знаходити інформацію та продукти, які вони шукають. Схема макету показана на рис. 2.16.

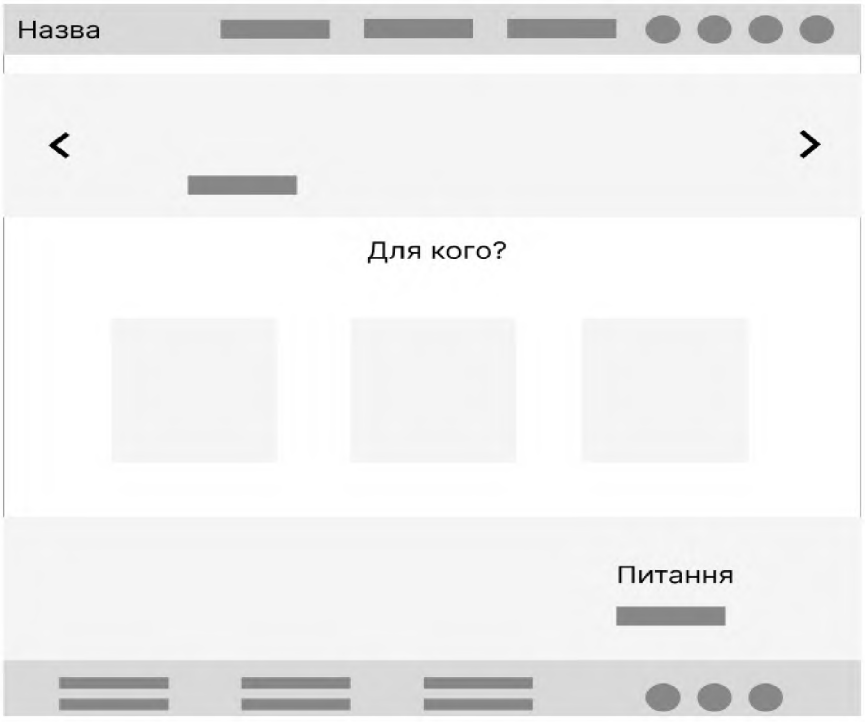


Рисунок 2.16 – Вигляд макету головної сторінки

Макет головної сторінки буде містити динамічні графічні елементи, які при певних діях і інтервалі часу змінюються в розмірах, основну частину з висвітленням можливостей трейд-боту та блок з посиланням на сторінку допомоги, де користувач зможе відправити листа

Візуальні елементи, такі як зображення, кольори та типографіка, повинні використовуватися стратегічно, щоб покращити загальну естетику сайту та відобразити ідентичність його призначення. Сайт за планом виготовлений в стилі мінімалізму. В якості графіки використано популярний сторітеллінг, іконки (зображення формату .png). Графічний проєкт основної частини вебсайту показано на рис. 2.17.

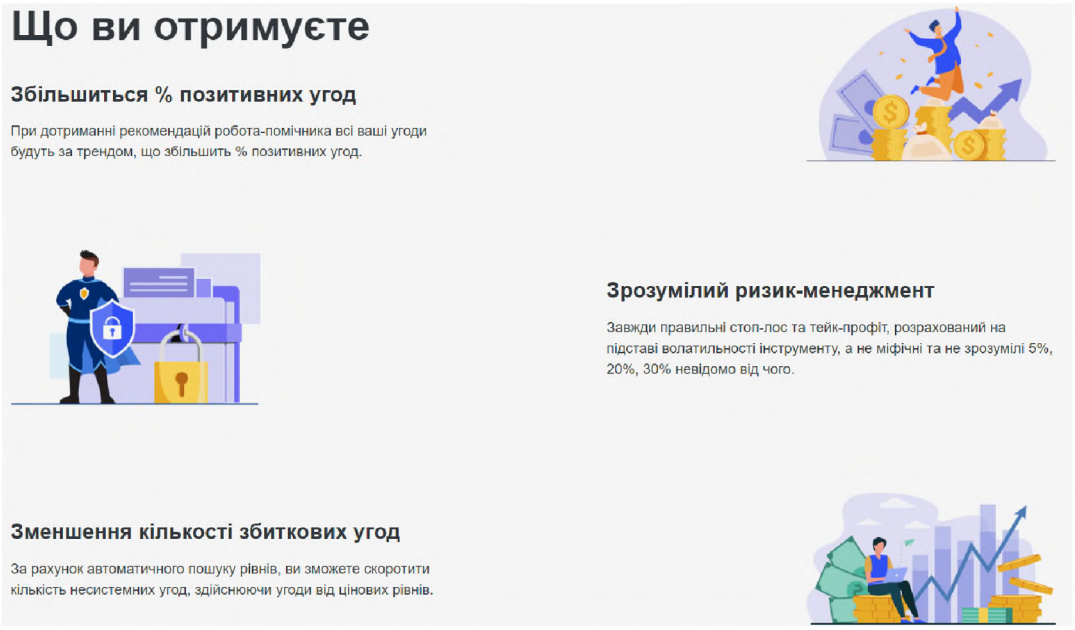


Рисунок 2.17 – Мінімалістичний стиль основної частини сайту і сторітеллінг

Окремим блоком представлено картки функцій робота – помічника. В якості змістового наповнення в текстовій формі представлені пояснення основних функцій робота. При цьому враховано потреби користувача. Кожний інформаційний блок виділений фоновою областю та іконкою-символом (рис. 2. 18)

Всі представлені елементи було заплановано зробити динамічними із використанням таких ефектів, як спливаючі зображення при прокручуванні сайту. Для цього підібрано інструменти технологій CSS, бібліотек JavaScript.

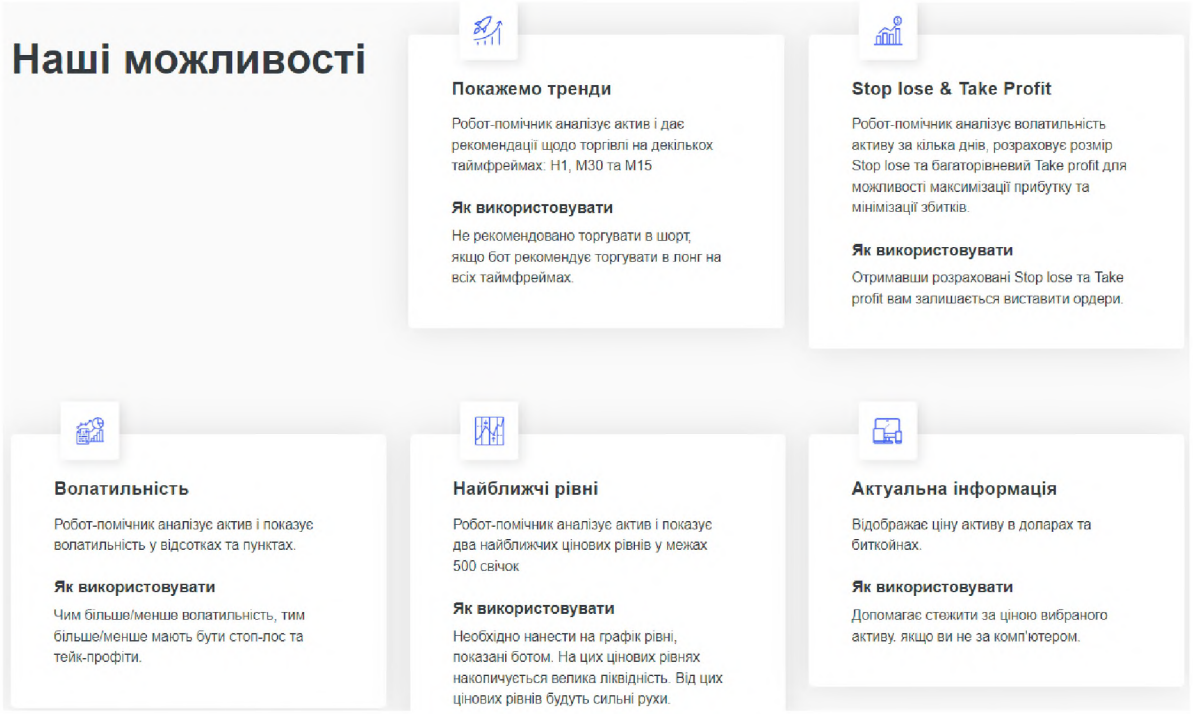


Рисунок 2.18 – Вигляд макету основної частини сторінки

Нижня частина вебсайту містить посилання на партнерів та форму зворотного зв’язку (рис. 2.19).

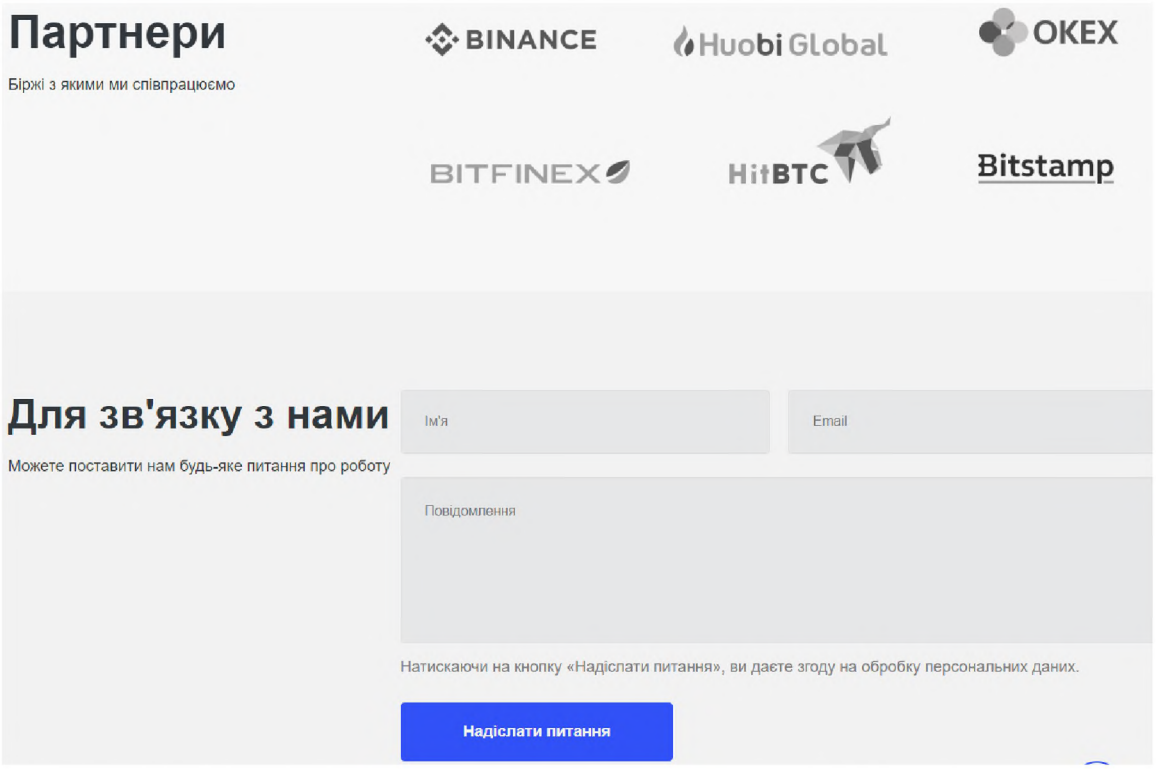


Рисунок 2.19 – Вигляд макету нижньої частини сторінки з формою зворотного зв’язку

Трьом частинам лендінгу відповідають такі елементи навігації, як горизонтальне меню та стрілки для вертикального пересування. Три пункти свідчать про розбивку основної частини на три основних екранних блоки (рис. 2.20).

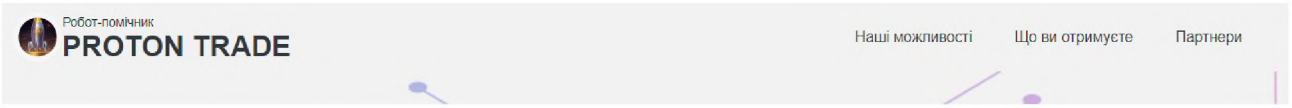


Рисунок 2.20 – Вигляд верхньої частини макету з горизонтальним меню

В наступній частині представлено опис структури коду вебсайту як набору програмних файлів та елементів коду, які відповідають за окремі функції виконаного вебсайту. Окремо продумана адаптивність вебсайту до різних екранів.

РОЗДІЛ 3
ПРАКТИЧНІ РЕЗУЛЬТАТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ КОМЕРЦІЙНОГО ВЕБСАЙТУ

1 Технічні прийоми створення основних розділів вебсайту

Важливою частиною створення будь-якого сайту є семантична оптимізація коду. Семантична оптимізація коду фокусується на використанні відповідних тегів HTML для відображення структури та ієрархії контенту. Це допомагає пошуковим системам зрозуміти контекст і релевантність інформації, що потенційно покращує видимість сайту в результатах пошукової видачі. На рис. 3.1 представлено структуру верхньої частини, в якій використано з поміж інших семантичні теги <header> <nav>.

```
<!DOCTYPE html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="assets/css/style.min.css">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="robots" content="index, follow">
    <meta name="google" content="notranslate">
    <meta name="format-detection" content="telephone=no">
    <meta name="description" content="">

    <title>Proton Trade</title>
</head>
<body>

    <div class="page">
        <header class="header"  id="top">
            <div class="container">
                <div class="header__wrapper">
                    <div class="header__logo logo">
                        <div class="logo__img">
                            <img src="assets/images/logo.png" alt="Logo Proton Trade">
                        </div>
                        <div class="logo  text">
                    </div>

                    <nav class="header__nav">
                        <ul class="header__list">
                            <li>
                                <a class="header__link lift" href="#capabilities" >
                                    Наші можливості
                                </a>
                            </li>
                            <li >
                                <a class="header__link lift" href="#you-get">
                                    Що ви отримуєте
                                </a>
                            </li>
                            <li>
                        </ul>
                    </nav>
```

Рисунок 3.1 – HTML-кодування верхньої частини вебсайту Proton Trade

Використовуючи описові та змістовні імена для елементів HTML, розробники можуть зробити свій код більш зрозумілим і легким для сприйняття. Це не тільки приносить користь іншим розробникам, які можуть працювати над проектом, але й сприяє загальній легкості обслуговування кодової бази. Перше що видно на головній сторінці це header (верхній колонтитул або «шапка»). Після нього зображений банер з назвою теми сайту (див. рис. 3.1). Повний HTML-код вебсторінки представлено в додатку Б. Якщо брати до уваги інші файли, а саме зображення, то вони знаходяться в кореневій папці – у папці «images», де знаходяться всі необхідні зображення.

Ще одним ключовим аспектом семантичної оптимізації коду є розмежування проблем, пов'язаних із контентом, представленням і поведінкою. Такий підхід дозволяє розробникам підтримувати чітке розділення HTML, CSS і JavaScript, що полегшує оновлення та модифікацію певних аспектів вебсайту, не впливаючи на інші.

Для управління функціями головної сторінки створені 2 файли: перший це файл з розширенням «.js», який слугує для написання коду; інший файл із розширенням «.css», який призначений для стилю різних елементів коду. Підключення останнього представлено в рядку в частині <head>:

<link rel="stylesheet" href="assets/css/style.min.css">.

SCSS (Sassy CSS) – це препроцесор, який розширює можливості звичайного CSS. Він надає додаткові можливості, такі як змінні, вкладеність та інші, які роблять написання стилів більш ефективним та організованим [32]. Повний лістинг таблиці стилів (компактної версії) представлено в додатку В.

При верстанні вебсайту використані сучасні функції CSS, які дозволяють досягнути динамічних ефектів. Наприклад, для появи окремих блоків під час прокручування або натискання одного з пунктів меню використано спеціальну функцію transition: 1s; }, записану в класі .demo, а також властивості об’єкта в різних епізодах із вказанням максимального розміру блоку від появи до розкриття (.demo._active). Фрагмент відповідного запису стилів у файлі style.css представлено на рис. 3.2.

```
.demo {
  background: linear-gradient(#4867ff, #4867ff), url(../../assets/images/web.png) center/cover no-repeat;
  background-blend-mode: overlay;
  position: relative;
  opacity: 0;
  margin-left: -100%;
  transition: 1s; }
  .demo._active {
    margin-left: 0;
    opacity: 1; }
  @media (max-width: 768px) {
    .demo {
      padding-bottom: 1rem; } }
```

Рисунок 3.2 – Запис властивостей елементів, які динамічно з'являються на головній частині вебсторінки

У середній частині вебсайту необхідно привернути увагу до партнерських компаній, що вже використовують трейд-бот. Це зроблено у вигляді трансформації зображень на сайті з логотипами цих компаній. Властивості трансформації кожної частини блоку прописані в кількох класах: .partners__company, .partners__descript, та активований .partners__descript._active (рис. 3.3, повний перелік див. в додатку В).

```
.partners__company {
  flex-basis: 66%;
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  align-items: flex-end; }
  @media (max-width: 1200px) {
    .partners__company {
      flex-basis: 75%; } }
.partners__img {
  margin: 0 auto;
  opacity: 0;
  transform: scale(0);
  transition: 1.6s; }
  .partners__img._active {
    opacity: 1;
    transform: scale(1); }
```

Рисунок 3.3 – Приклад використання функції трансформації (збільшення) зображень в CSS із заданими властивостями

Як і було передбачено, для цільової аудиторії необхідно мати доступ до сайту з будь-яких пристроїв, тому адаптивності елементів було приділено значну

увагу. Так, класичний приклад забезпечення гнучкості блоків за допомогою властивостей display: flex; , flex-wrap: wrap; , flex-basis наведено на рис. 3.4.

```
862  .partners__company {
863    flex-basis: 66%;
864    display: flex;
865    justify-content: space-between;
866    flex-wrap: wrap;
867    align-items: flex-end; }
868    @media (max-width: 1200px) {
869      .partners__company {
870        flex-basis: 75%; } }
871  .partners__img {
```

Рисунок 3.4 – Адаптивні flex-властивості блоків

Для нижньої частини вебсайту показані властивості для зміни порядку розташування блокових елементів при перегляді на малому екрані, зокрема, flex-direction: column;, а також граничні розміри зменшення (рис. 3.5).

```
892  .contact-us__wrapper {
893    display: flex;
894    justify-content: space-between; }
895    @media (max-width: 768px) {
896      .contact-us__wrapper {
897        flex-direction: column;
898        text-align: center; } }
899  @media (max-width: 768px) {
900    .contact-us__caption {
901      padding-bottom: 5rem; } }
902  @media (max-width: 481px) {
903    .contact-us__caption {
```

Рисунок 3.5 – Набір flex-властивостей для блоків нижньої частини вебсайту

Перевірка адаптивності наведена на рис. 3.6 – 3.7.

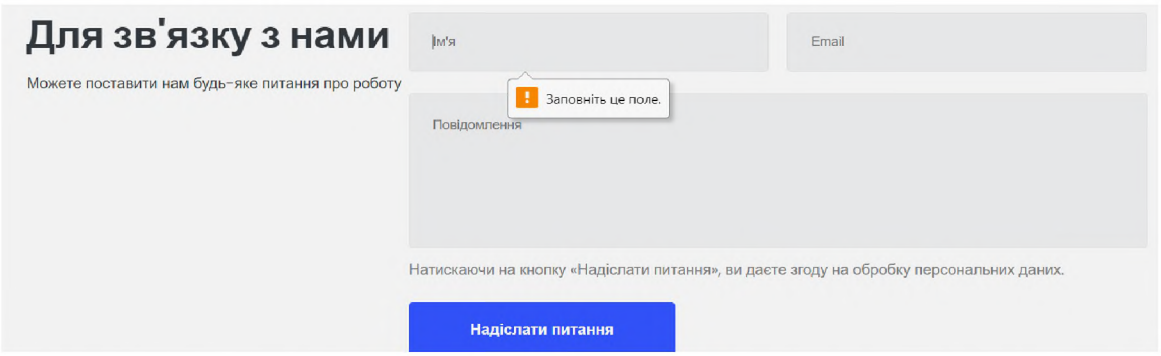


Рисунок 3.6 – Вигляд форми контактів та заголовку на звичайному екрані

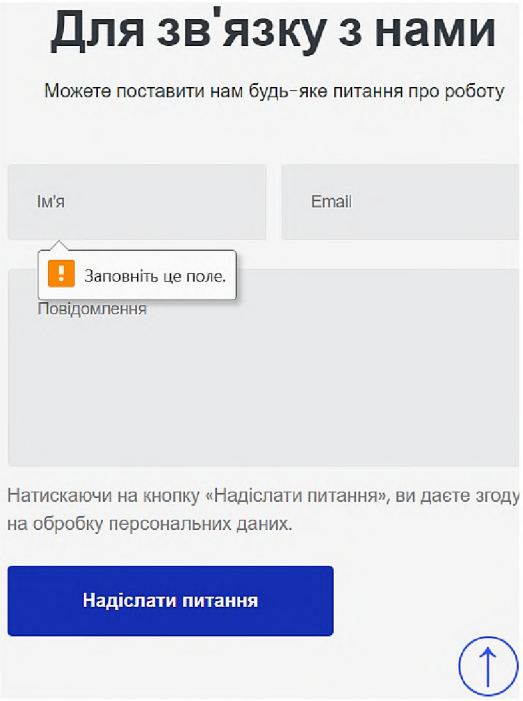


Рисунок 3.7 – Розташування елементів форми на екрані смартфона

Для роботи з об’єктами використано набір функцій JavaScript. Підключення файлу app.js розміщено в нижній частині вебсайту у вигляді скрипта (рис. 3.8), що дозволяє уникнути сповільнення завантаження та роботи з вебсайтом у браузері.

```
302        </div>
303
304        <script src="assets/js/app.js"></script>
305    </body>
306    </html>
```

Рисунок 3.8 – Запис посилання на файл app.js в нижній частині HTML-коду

Повний лістинг цього файлу представлено в додатку Г. Для прикладу розглянемо один із методів, який використаний для роботи з елементами. Мова йде про статичний метод Object.defineProperties(), який визначає нові або змінює існуючі властивості безпосередньо об’єкта, повертаючи об’єкт. Синатксис методу записується Object.defineProperties(obj, props). При цьому:

obj - об’єкт, для якого потрібно визначити або змінити властивості;

props - Об’єкт, ключі якого представляють назви властивостей, які потрібно визначити або змінити, а значення – об’єкти, що описують ці властивості. Кожне

значення в props має бути або дескриптором даних, або дескриптором доступу; не може бути обох [33].

Відповідний фрагмент запису методу наведено на рис. 3.9 разом із визначенням функції __webpack_require__.t.

```
48  /******/        Object.defineProperty(exports, '__esModule', { value: true });
49  /******/    };
50  /******/
51  /******/    // create a fake namespace object
52  /******/    // mode & 1: value is a module id, require it
53  /******/    // mode & 2: merge all properties of value into the ns
54  /******/    // mode & 4: return value when already ns object
55  /******/    // mode & 8|1: behave like require
56  /******/    __webpack_require__.t = function(value, mode) {
57  /******/        if(mode & 1) value = __webpack_require__(value);
58  /******/        if(mode & 8) return value;
59  /******/        if((mode & 4) && typeof value === 'object' && value && value.__esModule) return value;
60  /******/        var ns = Object.create(null);
61  /******/        __webpack_require__.r(ns);
62  /******/        Object.defineProperty(ns, 'default', { enumerable: true, value: value });
63  /******/        if(mode & 2 && typeof value != 'string') for(var key in value) __webpack_require__.d(ns,
64  /******/        return ns;
65  /******/    };
```

Рисунок 3.9 – Приклад опису методу Object.defineProperties у файлі .JS

Використання методів та функцій JS дозволяє управляти та обробляти дані більш коректно, у тому числі завдяки алгоритмам розгаження та циклам (див. рис. 3.9).

3.2 Методи оптимізації вебсайту та елементи тестування працездатності

При розробленні вебсайту враховано вимоги до пошукової оптимізації вебсайту та збільшення переглядів цільовою аудиторією. Заходи з оптимізації можна поділити на кілька груп.

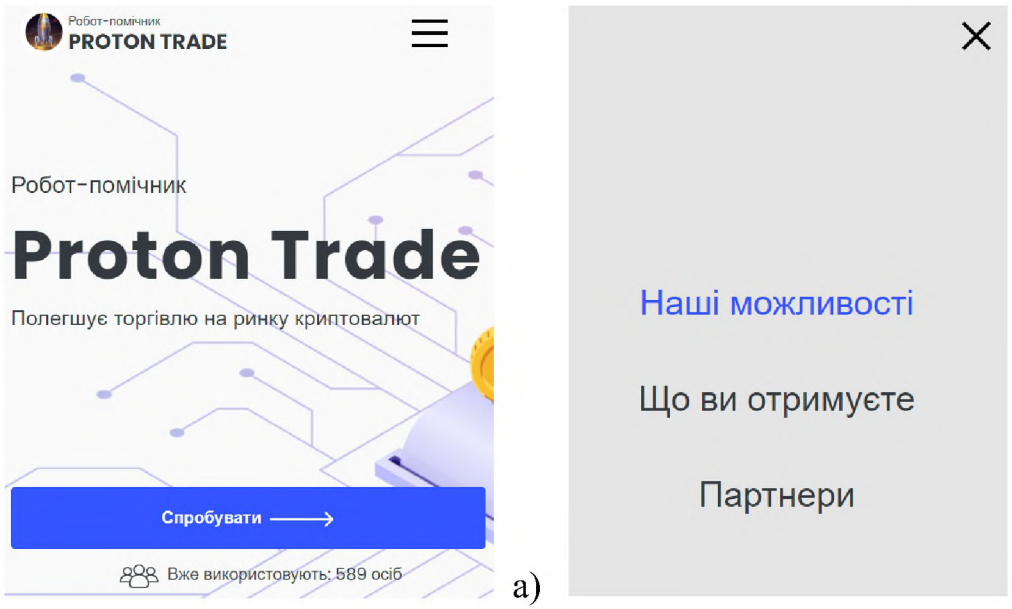
1. Графіка та мультимедіа. Вставлені візуальні елементи, які доповнюють тематику сторінки, такі як ілюстрації, фотографії та відео. При цьому використані формати, що підтримуються веббраузерами, для оптимального завантаження та відтворення. Окремо обрані спеціальні шрифти з бібліотеки Google, які надають виразності тексту.

2. Оптимізація та тестування на різних пристроях. Готовий вебсайт проходив попереднє тестування з метою усунення як недоліків у написанні коду, так і для правильного відображення на різних екранах. У попередньому розділі було зазначено про використання флекс-елементів для досягнення адаптивності різних блоків. Для збереження функцій горизонтального меню на малих екранах застосовано спеціальне бургер-меню, яке компактно згортає всі посилання в «бургер». Властивості, які формують бургер-меню, наведені на рис. 3.10.

```
.burger {
  display: none;
  z-index: 2;
  width: 3.4rem;
  height: 2.6rem;
  position: absolute;
  top: 2rem;
  right: 5rem;
  font-size: 0;
  color: transparent; }
  @media (max-width: 768px) {
    .burger {
      display: block; } }
  @media (max-width: 481px) {
    .burger {
      top: 2rem;
      right: 1rem; } }
  .burger:before, .burger:after,
  .burger span {
    display: block;
    width: 100%;
    height: 3px;
    background-color: #000;
    position: absolute;
    left: 0; }
  .burger:before, .burger:after {
    content: "";
    transition: transform 0.5s linear; }
  .burger:before {
    top: 0; }
  .burger:after {
    bottom: 0; }
  .burger span {
    top: 50%;
    transform: translateY(-50%);
    transition: opacity 0.2s linear; }
  .burger.active span {
    opacity: 0; }
```

Рисунок 3.10 – Правила стилів для оформлення бургер-меню на вебсайті

Результат роботи властивостей бургер-меню показано на рис. 3.11 при перегляді сайту на смартфоні.



а) б)

Рисунок 3.11 – Вигляд бургер-меню при перегляді на смартфоні в згорнутому вигляді (а) та активованому стані (б)

Адаптивний дизайн спрямований на те, щоб вебсайти були візуально привабливими, незалежно від пристроїв, з яких користувачі отримують доступ до них (стаціонарного ПК, ноутбука, планшета або смартфона [34]. Адаптивний дизайн був розроблений для всього сайту (рис. 3.12).

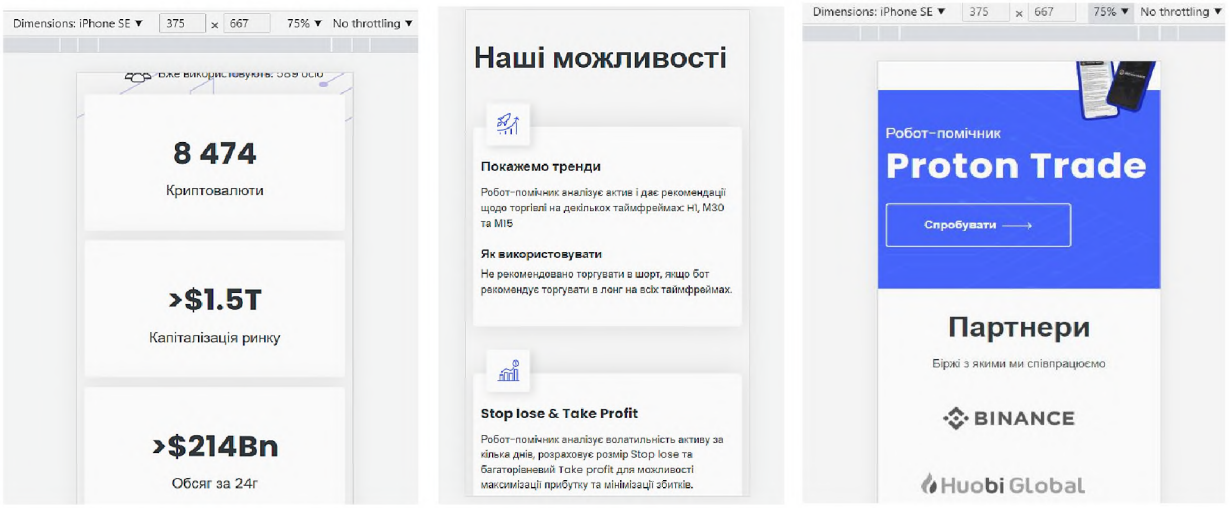


Рисунок 3.12 – Адаптивний формат частин вебсайту на екрані планшету IPad Air

Перед розгортанням вебсайту в мережі, було проведене тестування, щоб переконатися в її правильному відображенні та функціональності на різних пристроях та браузерах за допомогою сервісу [35]. Виявлені проблеми

виправлені, а також оптимізоване завантаження сторінки для поліпшення її продуктивності та швидкості завантаження (рис. 3.13.)

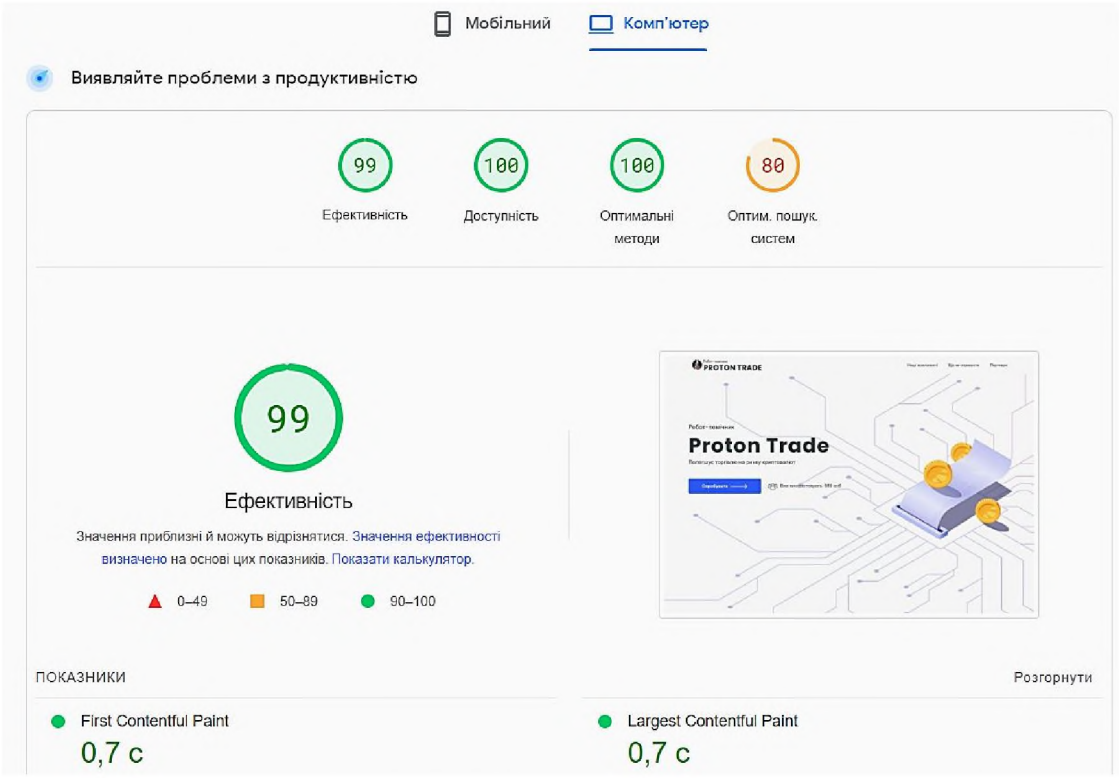


Рисунок 3.13 – Результати технічної веб-аналітики сайту

3. Оптимізація контенту та ключові слова. Заголовок і підзаголовок: яскравий та змістовний заголовок, який привертає увагу користувачів, а також підзаголовок, який уточнює основне повідомлення (рис. 3.14).

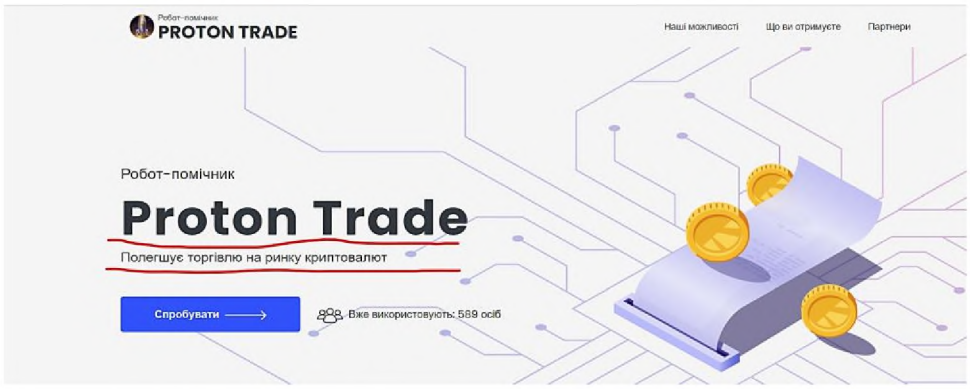


Рисунок 3.14 - Заголовки та підзаголовки на сайті

В основній частині опис трейд-бота включає текст про основні переваги та функціональні можливості Proton Trade. Пояснено, як він працює, які стратегії

використовуються та які можливості надає користувачам. В окремих блоках продемонстровано реальні результати торгівлі за допомогою трейд-бота (рис. 3.15). При цьому виведено тільки необхідну інформацію, виразні шрифти.

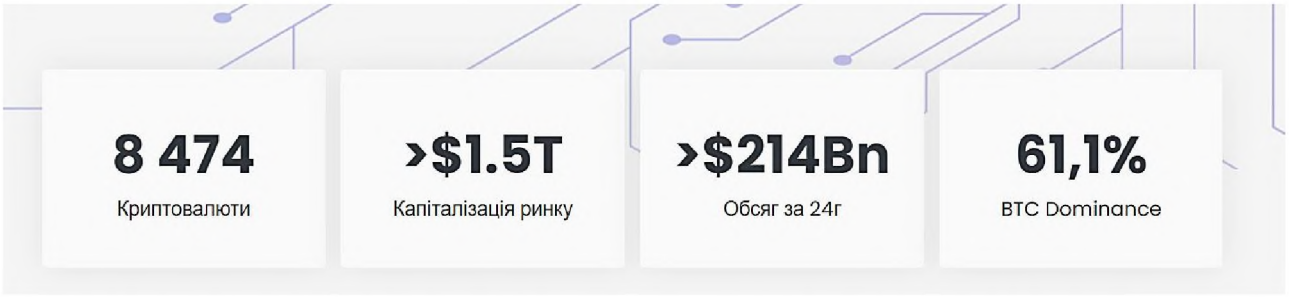


Рисунок 3.15 – Візуальні елементи для демонстрації результатів роботи бота

Так само, в розділі «Що ви отримуєте» наведено в аргументованій формі переваги використання трейд-бота Proton Trade (наприклад, див. рис. 2.18). Готовий код розробленого вебсайту розміщено в репозиторії GitHub [36].

3.3 Перевірка та опис економічної ефективності розробки

Перевірка та опис економічної ефективності розробки проєкту передбачає детальну оцінку різних аспектів для визначення його життєздатності та ефективності у створенні економічних переваг.

Аналіз витрат. Для розглянутого проєкту розробки, що включає створення вебдодатку з використанням РЕЬД5, CSS3, SCSS, JS, React та деяких прикладних програм, були враховані фактори витрат, описані нижче.

1. Інструменти та програмне забезпечення для розробки. Використовував Visual Studio Code як редактор коду, який є безкоштовним і широко використовуваним інструментом у спільноті розробників. Він допоміг ефективно писати та керувати кодом.

2. Хостинг та інфраструктура. Firebase пропонує безкоштовний план хостингу, що дозволило мені розгорнути свій вебдодаток без жодних витрат на

хостинг. Ця економічна опція забезпечила зручну та надійну платформу для розміщення додатку.

3. Сторонні бібліотеки та пакети. Ретельно відібрав бібліотеки та пакети з відкритим вихідним кодом для свого проєкту, що дозволило уникнути будь-яких витрат, пов'язаних з придбанням або ліцензуванням.

4. Витрати на персонал. Оскільки працював над проєктом самостійно, додаткових витрат на персонал не було. Загалом же використовується тариф на оплату праці програміста, вебдизайнера середнього рівня.

5. Витрати на інтеграцію та API. Розробляючи вебдодаток також врахував будь-які потенційні витрати, пов'язані з інтеграцією сторонніх сервісів або використанням зовнішніх API. Залежно від конкретних може виникнути потреба в інтеграції з платіжними шлюзами, постачальниками послуг або іншими зовнішніми системами.

Інтеграція з цими сервісами часто пов'язана з додатковими витратами, такими як комісія за транзакції, підписка або плата за використання. Важливо оцінити моделі ціноутворення та умови цих інтеграцій, щоб переконатися, що вони відповідають бюджету та вимогам проєкту. З вище описаних витрат, могли бути витрати тільки за оплату за замовлення та комісію.

6. Обслуговування та підтримка. При розрахунках економічної вартості потрібно врахувти потенційні майбутні витрати на обслуговування та підтримку вебдодатку. Це включає такі фактори, як поточне обслуговування сервера (за наявності), виправлення помилок і розширення функцій. Якщо використовувати проєкт для багатьох користувачів, то потрібно врахувати витрати на базу даних.

Враховуючи вищезазначені фактори і використовуючи безкоштовні ресурси, такі як Visual Studio Code та безкоштовний хостинг, при підготовці роботи ефективно керував витратами. Ця економічна ефективність дозволила зосередитися на успішному виконанні проєкту без фінансових обмежень. Розрахунок типових витрат наведено в табл. 3.1.

Таблиця 3.1 – Перелік витрат при розробці комерційного вебсайту, станом на початок 2024 р.

№ п/п	Найменування послуги (зміст)	Вартість, грн
1	Планування та дизайн. Збір вимог, структурування та UX дизайну	9217,00
2	Frontend розробка. HTML/CSS кодування, реалізація UI	18536,00
3	Backend розробка. Налаштування бази даних, логіка на стороні сервера	18536,00
4	Інтеграція API (платіжний шлюз, тощо)	3000,00
5	Тестування та QA. Забезпечення якості, виправлення помилок	9217,00
6	Впровадження та запуск. Налаштування сервера, реєстрація домену використовуючи Firebase	370,00
7	Поточне обслуговування. Оновлення, виправлення помилок, технічна підтримка (терміном дії на місяць)	7414,00
8	Сумарні витрати на розробку, встановлення та обслуговування комерційного вебсайту	66290,00

Хоча етап розробки пройшов успішно, слід визнати, що в майбутньому можуть виникнути непередбачувані технічні проблеми. Ці проблеми можуть включати проблеми сумісності між різними версіями бібліотек або потенційні вразливості безпеки, які можуть з'явитися. Щоб зменшити ці ризики, необхідно залишатися в курсі останніх версій та проводити регулярний моніторинг.

В результаті докладених зусиль додаток було розроблено ефективно, з використанням економічно вигідних ресурсів та з дотриманням найкращих практик. Проактивно реагуючи на потенційні ризики, пом'якшені виклики і створено безпечний, масштабований і зручний вебдодаток.

Загалом, поєднання використання економічно ефективних ресурсів, проведення ретельної оцінки ризиків та впровадження відповідних заходів сприяло успішному завершенню проєкту з розробки комерційного вебсайту.

ВИСНОВКИ

На основі здійсненого аналізу принципів розробки комерційного вебсайту можна зробити наступні висновки на завершення роботи.

1. Детальне дослідження теоретичних основ розробки вебсайту дозволили систематизувати і належним чином представити матеріал про:

– поняття та функціонування вебсайту загалом;

– розуміння принципів поведінки споживачів в інтернеті при користуванні вебдодатками;

– аналіз та вивчення потреб для користувачів такого типу сайтів;

– існуючі методи і технічні прийоми розробки вебсайтів.

2. Здійснено вибір технологій та обґрунтування методів розробки комерційного вебсайту, а саме:

– принципи та методики розробки UX/UI Design, спрямовані на поліпшення користувацького досвіду та інтерфейсу вебсайту;

– безкоштовний та потужний редактор коду Visual Studio Code, який підтримує розширення для HTML, CSS та JavaScript;

– HTML5 як основна мова розмітки, що використовується для створення структури та вмісту вебсторінок;

– мова каскадних таблиць стилів CSS, яка використовується для оформлення вебсторінок, має спецальні функції для задання зовнішнього вигляду елементів, кольорів, шрифтів тощо;

– мова сценаріїв препроцесора SASS, яка інтерпретується або компілюється в каскадні таблиці стилів;

– мова програмування JavaScript, яка дозволяє надавати вебсторінкам інтерактивність та динамічні функціональні можливості;

– Gulp – таск-менеджер для автоматичного виконання завдань, написаний на мові програмування JavaScript;

– Git як розподілена система керування версіями файлів та спільної роботи;

– адаптивний дизайн для забезпечення читабельності сайту на різних екранах;

– GitHub Pages – вебхостинг, який пропонує GitHub для розміщення статичних вебсторінок для користувачів GitHub, блогів користувачів, проектної документації або навіть цілих книг. Вебсайти розміщуються в субдоменах домену github.io або в іншому домені, який придбав користувач.

3. Практичні аспекти розроблення комерційного вебсайту показали, що створення візуально привабливого та інтуїтивно зрозумілого користувацького інтерфейсу є життєво важливим для покращення досвіду перегляду та залучення клієнтів.

4. Оптимізація сайту для пошукових систем є ключовим аспектом, який допоможе підвищити видимість та привернути цільову аудиторію. Використання відповідних ключових слів у заголовках, метатегах та контенті, а також збільшення кількості зовнішніх посилань на сайт сприятимуть підвищенню рейтингу у пошукових системах

Враховуючи та впроваджуючи теоретичні основи розробки вебсайтів, обираючи відповідні методи та звертаючись до практичних аспектів, робота виконана в запланованому обсязі.

Розробка та оптимізація такого вебсайту, як Proton Trade, є складним та багатогранним процесом. Врахування вищезазначених аспектів дозволить створити привабливий, функціональний та ефективний вебсайт, який буде відповідати потребам користувачів та сприятиме досягненню бізнес-цілей.