МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ

Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
Аліна САВЧЕНКО
«_____» _____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНИЙ ПРОЄКТ, ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ **"БАКАЛАВР"**
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
"ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: "Web додаток підбору нових комплектуючих для персонального комп'ютера**"**

Виконавець: студент групи УС-412 Драпогуз Роман Сергійович

Керівник: к.т.н., доцент Холявкіна Тетяна Володимирівна

Нормоконтролер: Олександр ШЕВЧЕНКО

Київ – 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *Комп'ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«____»_____2023р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Драпогуза Романа Сергійовича

(прізвище, ім'я, по батькові)

1. **Тема роботи**: «Web додаток підбору нових комплектуючих для персонального комп'ютера» затверджена наказом ректора від 01.05.2023р. № 623/ст.
2. **Термін виконання роботи** з 15.05.2023 р. по 25.06.2023 р.
3. **Вихідні дані до роботи**: Web додаток для підбору нових комплектуючих для персонального комп'ютера.
4. **Зміст пояснювальної записки**: вибір мов програмування, дослідження різних мов програмування, аналіз важливих факторів, які мають вирішальне значення для людей при виборі збірок ПК, комплексний аналіз різноманітних інтегрованих середовищ розробки програмного забезпечення.
5. **Перелік обов'язкового графічного матеріалу**: робоче вікно Visual Studio Code, застосунок FileZilla, компоненти у Figma.

6. Календарний план-графік

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1.	Проаналізувати літературу та джерела за темою дипломного проекту.	15.05.2023 – 17.05.2023	
2.	Розроблення та затвердження плану дипломного проекту.	18.05.2023 – 20.05.2023	
3.	Провести консультації з науковим керівником щодо створення першого розділу.	21.05.2023– 23.05.2023	
4.	Розробка розділу 1	24.05.2023– 02.06.2023	
5.	Розробка розділу 2	03.06.2023– 09.06.2023	
6.	Розробка розділу 3	10.06.2023 – 13.06.2023	
7.	Висновки та оформлення пояснювальної записки дипломного проекту.	14.06.2023 – 16.05.2023	
8.	Підписання необхідних документів у встановленому порядку.	17.05.2023 – 19.05.2023	

7. Дата видачі завдання: «__15__» __травня____2023 р.

Керівник дипломного проекту _____ Тетяна ХОЛЯВКІНА
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання_____ Роман ДРАПОГУЗ
(підпис студента) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Web додаток підбору нових комплектуючих для персонального комп’ютера» складається із вступу, трьох розділів, висновку, списку бібліографічних посилань містить 77 сторінок, 40 рисунків, 1 таблицю. Список бібліографічних посилань складається з 10 найменувань.

Об'єктом дослідження **є**: веб-додаток, який призначений для надання користувачам можливості підібрати нові комплектуючі для їх персонального комп'ютера.

Предметом дослідження є: процес підбору нових комплектуючих для персонального комп'ютера за допомогою веб-додатку.

Мета роботи: є розробка веб-додатку, який забезпечує користувачам можливість зручного та точного підбору комплектуючих для їх персонального комп'ютера. Основною метою дослідження є визначення ефективності та функціональності такого додатку, а також його впливу на задоволення потреб користувачів у процесі вибору комплектуючих.

Методи дослідження:

1. метод проведення опитування;
2. метод тестування користувачами;
3. метод аналізу даних;
4. метод порівняльного аналізу;
5. метод експертної оцінки;
6. метод розроблення прототипів.

Результати дослідження: розроблений веб додаток «Підбір нових комплектуючих для персонального комп’ютера».

ПЕРСОНАЛЬНИЙ КОМП’ЮТЕР, WEB-ДОДАТОК, JAVASCRIPT, HTML, КОМПЛЕКТУЮЧІ, VS CODE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

HTML (HyperText Markup Language)	мова розмітки для створення веб-сторінок
JS (JavaScript)	мова програмування
CSS (Cascading Style Sheets)	мова стилів
SCSS (Sassy CSS)	препроцесор CSS
PHP (Hypertext Preprocessor)	мова програмування
IDE (Integrated Development Environment)	інтегроване середовище розробки
SEO (Search Engine Optimization)	процес оптимізації веб-сайту

ВСТУП

Світ компонентів персонального комп'ютера (ПК) величезний і складний, з безліччю опцій, доступних для кожної частини. Для тих, хто не розбирається в техніці, вибір правильних компонентів може бути складним завданням. Є багато вебсайтів, які пропонують можливість вибрати компоненти та зібрати їх, але часто ці сайти більше зосереджені на отриманні прибутку, ніж на допомозі своїм клієнтам. У результаті багато людей отримують системи, які не оптимізовані для їхніх потреб, або, що ще гірше, з компонентами, які несумісні один з одним (Рис. 1.1).

Щоб вирішити цю проблему, ми вирішили створити веб-додаток під назвою «Підбір нових комплектуючих для ПК». Ця програма буде актуальною, оскільки надасть користувачам інформацію, необхідну для прийняття зважених рішень про те, які компоненти вибрати і як їх зібрати та полегшить людям створення ПК, оптимізованих для їхніх конкретних потреб, не витрачаючи надмірних коштів на непотрібні компоненти або не сумісними один з одним компонентами.

Веб-додаток «Підбір нових комплектуючих для ПК» включатиме дані про компоненти ПК, включаючи відеокарти, процесори, материнські плати, оперативну пам'ять, охолодження та блоки живлення. Кожен компонент матиме власну спеціальну сторінку з детальною інформацією про його характеристики, включно з тим, з якими іншими компонентами він добре працюватиме. Крім того, ми надамо готові збірки для кожного компонента разом із поясненнями, чому були обрані ці компоненти та як вони працюють разом. Ми також додамо збірки, які були створені та надані іншими користувачами, що дозволить людям вчитися на досвіді інших.

Актуальність сайту - сайт збірки ПК дуже актуальний, оскільки він спрощує складний процес створення комп'ютера. Він пропонує широкий вибір компонентів, перевірку сумісності та покрокові інструкції. Цей сайти економить час, забезпечують оптимальний вибір компонентів і дають змогу користувачам легко створювати персоналізовані ПК.

Новизна сайту - на відміну від сайтів, орієнтованих на отримання прибутку, ми надаємо перевагу допомозі клієнтам. Ми надаємо важливу інформацію для прийняття обґрунтованих рішень, дозволяючи користувачам створювати оптимізовані ПК відповідно до їхніх потреб, уникаючи проблем із сумісністю та непотрібних витрат а також можна вибрати збірку відштовхуючись від одної деталі якщо вона вже є на руках.

РОЗДІЛ 1
ПІДБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВЕБ-ДОДАТКІВ

Неможливо переоцінити важливість вибору відповідного програмного забезпечення для веб-додатків. Успіх веб-програми часто значною мірою залежить від вибору програмного забезпечення, яке використовується для її розробки. Вибір правильного програмного забезпечення може зробити процес розробки плавнішим і ефективнішим, тоді як неправильний вибір може призвести до численних проблем, таких як проблеми сумісності, вразливості безпеки та обмеження продуктивності.

Фактори, які необхідно брати до уваги під час процесу вибору програмного забезпечення, включають функціональність і особливості програмного забезпечення, масштабованість, безпеку, простоту використання та вартість.

Вибір програмного забезпечення також залежить від конкретних вимог веб-додатку. Тому перед вибором програмного забезпечення важливо провести ретельний аналіз функціональних і нефункціональних вимог веб-додатку. Цей аналіз допомагає визначити необхідні функції та функції програмного забезпечення, які відповідатимуть потребам програми та її призначених користувачів.

Крім того, при виборі програмного забезпечення також необхідно враховувати технічні обмеження, такі як сумісність з існуючими системами, наявні ресурси та засоби розробки. Неврахування цих технічних факторів може призвести до проблем сумісності, обмеження продуктивності та інших проблем, які можуть перешкодити загальному успіху веб-програми.

Рис. 1.1. Проблема подібних сайтів

1.1. Аналіз вимог

Цей аналіз має вирішальне значення для вибору відповідного програмного забезпечення для розробки веб-додатку. Зосередимось на визначенні функціональних і нефункціональних вимог, потреб і переваг користувачів, а також технічних обмежень і обмежень для веб-програми.

Функціональні вимоги до веб-додатку включають функції та функції, необхідні для правильної роботи веб-сайту. Веб-сайт призначений для надання користувачам детальної інформації про конкретні компоненти для ПК, такі як відеокарти, процесори, материнські плати, оперативна пам’ять, охолодження та блоки живлення. Таким чином, функціональні вимоги до веб-додатку включатимуть:

1. Детальна інформація про кожен компонент – веб-додаток має надавати детальну інформацію про кожен компонент, включаючи його функції, характеристики та сумісність з іншими компонентами.

2. Перевірка сумісності – веб-програма повинна мати засіб перевірки сумісності, який може рекомендувати користувачам сумісні компоненти.

3. Попередньо зібрані збірки – веб-програма повинна мати попередньо зібрані збірки, які можуть допомогти користувачам у виборі компонентів, сумісних один з одним.

Нефункціональні вимоги – це якості або характеристики веб-програми, які безпосередньо не пов’язані з її функціональністю, але є критичними для її загальної продуктивності. Для веб-програми нефункціональні вимоги включатимуть:

1. Зручність використання – веб-додаток має бути простим у використанні та навігації.

2. Надійність – веб-додаток має бути надійним і доступним у будь-який час.

3. Безпека – веб-додаток має бути безпечним, а дані користувача – захищеними.

4. Продуктивність – веб-додаток має бути швидким і чуйним, з мінімальним часом завантаження сторінки.

Технічні обмеження – це технічні аспекти, які необхідно враховувати під час розробки веб-додатку. Ці обмеження можуть включати:

Таблиця 1.1.

Технічні обмеження

Сумісність із різними веб-браузерами	веб-програма має бути сумісною з різними веб-браузерами, включаючи Google Chrome, Mozilla Firefox, Safari та інші.
Мобільна адаптивність	веб-додаток має адаптуватися до мобільних пристроїв і бути доступним на різних пристроях, включаючи смартфони, планшети та ноутбуки.
Масштабованість	веб-додаток має бути масштабованим для майбутніх оновлень і розширень.
Інтеграція зі сторонніми інструментами	веб-додаток має мати можливість інтегруватися зі сторонніми інструментами та службами, такими як платформи соціальних мереж і інструменти електронного маркетингу.

Аналіз допоміг визначити функціональні та нефункціональні вимоги, потреби та переваги користувачів, а також технічні обмеження та обмеження для веб-програми.

1.2. Критерії оцінювання

Щоб вибрати програмне забезпечення, яке найкраще відповідатиме вимогам веб-додатку для вибору нових компонентів для ПК, важливо встановити критерії оцінки. Ці критерії повинні відображати конкретні потреби та переваги користувачів, а також технічні обмеження та обмеження проекту.

Одним з важливих критеріїв оцінки варіантів програмного забезпечення для веб-додатку є функціональність. Програмне забезпечення має бути здатним обробляти великі обсяги даних і складні обчислення, необхідні для точного рекомендування компонентів для конкретних потреб користувача. Він також повинен бути здатний відображати ці дані в організованому та легкому для розуміння вигляді.

Масштабованість є ще одним важливим фактором, який слід враховувати при виборі програмного забезпечення для веб-додатку. Оскільки сайт отримує більше користувачів і додається більше даних, програмне забезпечення повинно мати можливість впоратися зі збільшеним навантаженням, не стаючи повільним або нестабільним. Це гарантує, що веб-додаток може продовжувати ефективно обслуговувати користувачів у міру його зростання.

Простота використання — ще один важливий фактор, який слід враховувати, оскільки веб-програма має бути доступною та інтуїтивно зрозумілою для всіх користувачів, незалежно від їхнього технічного досвіду. Це означає, що вибране програмне забезпечення повинно мати зручний інтерфейс і чіткі інструкції щодо його ефективного використання.

Нарешті, вартість є важливим фактором, оскільки вибране програмне забезпечення має бути доступним у межах бюджету, виділеного на проект. Це означає, що програмне забезпечення має забезпечувати співвідношення ціна-якість із набором функцій і можливостей, які виправдовують його вартість.

Загалом, критерії оцінки вибору програмного забезпечення для веб-додатку для вибору нових компонентів для ПК повинні включати такі фактори, як функціональність, масштабованість, простота використання та вартість. Ці критерії допоможуть переконатися, що вибране програмне забезпечення відповідає потребам і впо-

добанням користувачів, а також дотримується технічних обмежень і обмежень проекту.

1.3. Процес вибору програмного забезпечення

Першим кроком у процесі вибору програмного забезпечення є проведення дослідження ринку та визначення доступних варіантів програмного забезпечення, які можна використовувати для веб-додатку. Дослідження має бути спрямоване на визначення варіантів програмного забезпечення, які відповідають вимогам веб-додатків і можуть ефективно виконувати необхідні завдання.

Після визначення доступних варіантів програмного забезпечення наступним кроком є їх оцінка на основі попередньо визначених критеріїв. Критерії оцінки повинні бути розроблені на основі вимог, визначених у попередньому розділі. Критерії можуть включати такі фактори, як функціональність, простота використання, масштабованість, безпека, вартість і підтримка.

Після встановлення критеріїв оцінки кожен варіант програмного забезпечення слід порівняти та порівняти, щоб визначити найкращий варіант для веб-додатку. Порівняння має базуватися на критеріях оцінки, і кожен варіант програмного забезпечення слід оцінювати за кожним критерієм. Варіанти програмного забезпечення можна ранжувати залежно від того, наскільки вони відповідають критеріям оцінки.

Після порівняння та порівняння варіантів програмного забезпечення слід вибрати найкращий варіант на основі критеріїв оцінки та рейтингу. Вибране програмне забезпечення має відповідати вимогам, визначеним під час аналізу вимог, і має ефективно виконувати необхідні завдання.

1.4. Вибране програмне забезпечення

Цей розділ має на меті представити вичерпне пояснення щодо вибору коду Visual Studio (VS Code) як програмного забезпечення, якому віддається перевага для поточного проекту. Оцінюючи плюси та мінуси Visual Studio Code, цей розділ надає

розуміння процесу прийняття рішень, висвітлюючи сильні та слабкі сторони програмного забезпечення.

Причини вибору коду Visual Studio:

1. Універсальний і легкий. Основним фактором, який обумовив вибір Visual Studio Code, є його універсальність і легкість. Як розробник, я ціную IDE, яка може легко працювати з різними мовами програмування та фреймворками. Широка підтримка мови Visual Studio Code в поєднанні з його здатністю ефективно працювати в системах з обмеженими ресурсами робить його ідеальним вибором для мого проекту.

2. Широка екосистема та налаштування. Величезна екосистема розширень Visual Studio Code зіграла вирішальну роль у процесі прийняття рішень. Наявність численних розширень дозволяє створити середовище розробки, яке можна налаштовувати відповідно до моїх конкретних потреб. Ця гнучкість дає мені змогу покращити мій робочий процес і підвищити продуктивність шляхом інтеграції інструментів і функцій, які відповідають вимогам мого проекту.

3. Сумісність між платформами. Іншим вирішальним фактором у виборі Visual Studio Code є його кросплатформна сумісність. Завдяки можливості працювати в різних операційних системах, включаючи Windows, macOS і Linux, Visual Studio Code забезпечує безперебійну співпрацю та дозволяє мені працювати ефективно незалежно від використовуваної платформи.

Плюси Visual Studio Code:

1. Ефективний досвід кодування. Visual Studio Code забезпечує інтуїтивно зрозумілий і ефективний досвід кодування завдяки таким функціям, як інтелектуальне завершення коду, підсвічування синтаксису та надійні можливості налагодження. Ці функції значно сприяють прискоренню процесів розробки та покращенню якості коду.

2. Активна підтримка спільноти. Процвітаюча спільнота, що оточує Visual Studio Code, забезпечує доступ до розширеної документації, навчальних посібників і великої кількості ресурсів. Ця мережа підтримки виявилася безцінною у вирішенні проблем, відкритті нових функцій і в курсі останніх подій.

Мінуси коду Visual Studio:

1. Крива навчання. Крива навчання, пов’язана з Visual Studio Code, може стати проблемою, особливо для новачків або тих, хто переходить з інших IDE. Набуття навичок у використанні його широкого набору функцій і налаштування IDE відповідно до особистих уподобань може вимагати початкових інвестицій часу та зусиль.

2. Відсутність розширених функцій IDE. Порівняно з повноцінними IDE, такими як Visual Studio, Visual Studio Code може не мати певних розширених функцій, необхідних для спеціалізованих сценаріїв розробки. У таких випадках розробникам може знадобитися покладатися на зовнішні інструменти або розширення, щоб доповнити функціональність IDE.

Вибір Visual Studio Code як програмного забезпечення, якому надається перевага, пояснюється його універсальністю, легким дизайном, розгалуженою екосистемою та сумісністю між платформами. Незважаючи на те, що він представляє криву навчання та може не мати деяких розширених функцій IDE, переваги ефективного кодування та настроюваного середовища переважують ці обмеження. Visual Studio Code дає розробникам змогу оптимізувати свій робочий процес, залучати спільноту підтримки та бездоганно працювати на кількох платформах. Розглядаючи плюси та мінуси, я впевнений, що Visual Studio Code сприятиме продуктивному та успішному результату проекту.

1.4.1. HTML - Гіпертекстова розмітка

Одним із основних інструментів веб-розробки є HTML або мова розмітки гіпертексту. Це стандартна мова розмітки, яка використовується для створення та дизайну веб-сторінок. HTML є фундаментальною частиною веб-розробки, і розробникам важливо чітко розуміти його можливості та обмеження.

Він використовується для структурування веб-вмісту, створення посилань між сторінками та відображення мультимедійного вмісту, наприклад зображень і відео. HTML також використовується для додавання стилів до веб-сторінок, таких як шрифти, кольори та макет.

Однією з головних переваг HTML є його простота. Це проста мова, яку легко вивчити навіть для початківців. HTML має відносно простий синтаксис і базується на тегах, які використовуються для визначення структури та вмісту веб-сторінки.

Ще однією перевагою HTML є його сумісність з різними платформами та пристроями. HTML розроблено таким чином, щоб бути незалежним від платформи, тобто його можна використовувати в широкому діапазоні операційних систем і пристроїв.

Підсумовуючи, HTML є основним інструментом у веб-розробці та необхідним для створення статичних веб-сторінок. HTML — це універсальна мова, яку легко вивчати та використовувати, і вона сумісна з широким спектром платформ і пристроїв. Для більш складних веб-додатків HTML часто використовується в поєднанні з іншими мовами програмування, такими як JavaScript і CSS.

1.4.2. JavaScript - Сценарії

JavaScript (JS) — ще один інструмент, який можна використовувати при розробці веб-сайту. JS — популярна мова програмування на стороні клієнта, яка використовується для створення інтерактивних і динамічних веб-сторінок. У цьому підрозділі ми обговоримо використання JS при розробці веб-додатку для вибору нових компонентів для ПК.

Однією з головних переваг використання JS є його здатність створювати динамічні та інтерактивні інтерфейси користувача. Це робить його ідеальним вибором для створення веб-додатків, де користувачі можуть взаємодіяти з різними компонентами та робити вибір відповідно до своїх вимог. Наприклад, коли користувач вибирає певний компонент, такий як відеокарта, JS може бути використаний для відображення додаткової інформації про цей компонент або пропозиції сумісних компонентів на основі вибору користувача.

JS також відомий своєю сумісністю з різними веб-браузерами, що робить його надійним вибором для веб-розробки. Це означає, що користувачі можуть отримати

доступ до веб-сайту з різних браузерів і пристроїв, і веб-сайт все одно працюватиме належним чином.

З іншого боку, є також деякі недоліки, які слід враховувати при використанні JS. Однією з першочергових проблем є проблема із сумісністю браузера. Незважаючи на те, що JS сумісний з різними браузерами, певні функції можуть не підтримуватися деякими браузерами, що може призвести до неузгодженості взаємодії з користувачем. Розробники повинні протестувати веб-сайт у різних браузерах і пристроях, щоб переконатися, що він функціонує належним чином.

1.4.3. CSS (SCSS) - Каскадні таблиці стилів

У контексті розробки веб-сайту для вибору нових компонентів для ПК каскадні таблиці стилів (CSS) і їхній препроцесор Sassy CSS (SCSS) є важливими інструментами для розробки та стилізації інтерфейсу веб-сайту.

CSS — це мова, яка використовується для оформлення документів HTML. Це дозволяє розробникам відокремлювати вміст від презентації, що означає, що HTML-документи містять лише вміст сторінки, тоді як файли CSS містять інструкції щодо того, як має бути представлений цей вміст. CSS надає широкий спектр варіантів стилю, включаючи колір, шрифт, розмір, макет і анімацію, що робить його важливим інструментом для створення візуально привабливих веб-сайтів.

SCSS — це препроцесор для CSS, який надає додаткові функції CSS, такі як вкладення, змінні, міксини та успадкування. Це дозволяє розробникам писати більш модульний код CSS, який зручно підтримувати, що полегшує оновлення та масштабування дизайну веб-сайту.

Переваги використання CSS і SCSS у веб-розробці численні. По-перше, вони забезпечують узгоджений вигляд і відчуття на веб-сайті, що покращує взаємодію з користувачем і допомагає користувачам легше орієнтуватися на сайті. По-друге, вони відокремлюють вміст від презентації, полегшуючи оновлення та зміну дизайну веб-сайту без зміни коду HTML. Ця модульність робить веб-сайт більш зручним для обслуговування та масштабованим. По-третє, CSS і SCSS скорочують час заванта-

ження веб-сайту, зберігаючи інструкції стилю окремо від вмісту HTML, дозволяючи браузеру швидше завантажувати веб-сайт.

Незважаючи на переваги використання CSS і SCSS, існують і деякі недоліки. Одним із найбільш істотних недоліків використання CSS є те, що може бути складно підтримувати узгодженість у кількох браузерах. Браузери мають різні реалізації CSS, а це означає, що веб-сайт, який чудово виглядає в одному браузері, може не виглядати так само в іншому.

1.4.4. PHP - Серверний скриптінг

Однією з мов, які можна використовувати для сценаріїв на стороні сервера, є PHP, яка відома своєю простотою та універсальністю.

PHP (Hypertext Preprocessor) — широко розповсюджена серверна мова сценаріїв із відкритим кодом, розроблена для веб-розробки. Він в основному використовується для створення динамічних веб-сторінок і веб-додатків. PHP можна вбудовувати в код HTML, дозволяючи розробникам створювати динамічний вміст на льоту. Він також може взаємодіяти з різноманітними базами даних, що робить його ідеальним для розробки веб-додатків на базі даних.

Однією з ключових переваг PHP є простота використання. Код PHP можна написати швидко й легко навіть тим, хто тільки починає програмувати. Це робить його популярним вибором для початківців і тих, хто тільки починає веб-розробку.

Ще однією перевагою PHP є його універсальність. PHP можна використовувати для розробки широкого спектру веб-додатків, від маленьких особистих блогів до великомасштабних веб-сайтів електронної комерції.

Одним із головних недоліків PHP є його безпека. Оскільки PHP є мовою з відкритим кодом, його вихідний код є у вільному доступі для всіх. Це означає, що зловмисники можуть перевіряти код на наявність уразливостей і використовувати їх. Щоб зменшити цей ризик, розробники повинні бути пильними, захищаючи свої PHP-додатки та оновлюючи останні виправлення безпеки.

Іншим потенційним недоліком PHP є його продуктивність. Хоча PHP може бути швидким і ефективним, він також може бути повільним і ресурсомістким, якщо його не оптимізувати належним чином. Розробники повинні ретельно писати ефективний код і оптимізувати продуктивність своїх програм.

1.4.5. jQuery - Бібліотека JavaScript

jQuery — популярна бібліотека JavaScript, яка спрощує процес створення інтерактивних і динамічних веб-сторінок. Його розроблено, щоб полегшити маніпулювання HTML DOM (об'єктною моделлю документа), обробку подій, створення анімації та виконання запитів AJAX, серед іншого. jQuery — це програмне забезпечення з відкритим вихідним кодом, яке підтримується спільнотою розробників і широко поширене в галузі веб-розробки.

Однією з головних переваг використання jQuery є те, що він зменшує кількість коду, необхідного для виконання стандартних завдань веб-розробки. Наприклад, вибір елемента в DOM і маніпулювання його властивостями можна виконати лише за допомогою кількох рядків коду jQuery, а не більш детального та схильного до помилок JavaScript, який був би необхідний інакше. Це робить jQuery потужним інструментом як для досвідчених, так і для початківців веб-розробників, оскільки він дозволяє їм зосередитися на бажаній функціональності, а не на основних деталях впровадження.

jQuery також має велику кількість плагінів, які надають додаткові функції, такі як візуалізація даних, перевірка форм і галереї зображень. Ці плагіни створені на основі основної бібліотеки jQuery, і їх можна легко включити у веб-сторінку. Це дозволяє веб-розробникам легко додавати складні функціональні можливості на свої сайти без необхідності самостійно писати код.

1.4.6. BEM - Блок-елемент-модифікатор

BEM або Block Element Modifier — це методологія для написання масштабованого та підтримуваного коду CSS. У цьому підрозділі ми обговоримо принципи BEM і те, як його можна використовувати для покращення CSS веб-сайту.

BEM — це угода про найменування, яка має на меті зробити структуру HTML і назви класів CSS більш читабельними та зручними для обслуговування. Угода про іменування передбачає поділ коду на блоки, елементи та модифікатори.

Блок — це окремий компонент на сторінці, який виконує певну функцію, наприклад панель навігації або кнопка. Блоки мають назви за допомогою описового та унікального ідентифікатора, наприклад .navbar або .button.

Елементи — це компоненти, з яких складається блок. Елементи називаються назвою блоку, за якою йде подвійне підкреслення (__), а потім описова назва, наприклад .navbar__item або .button__text.

Модифікатори використовуються для зміни зовнішнього вигляду або поведінки блоку або елемента. Модифікатори називаються назвою блоку або елемента, за якою йде подвійне тире (--), а потім описова назва, наприклад .navbar--sticky або .button--disabled.

Використання методології BEM допомагає гарантувати, що класи CSS можна використовувати повторно та не впливають на інші компоненти сторінки. Використовуючи стандартизовану угоду про найменування, розробники можуть легко ідентифікувати та змінювати компоненти, не турбуючись про конфлікти імен або побічні ефекти.

Окрім покращення зручності обслуговування коду CSS, BEM також може покращити продуктивність веб-сайту. BEM заохочує розробників писати більш конкретні селектори CSS, які можуть допомогти зменшити кількість правил, застосованих до сторінки. Це, у свою чергу, може допомогти зменшити розмір файлу CSS і покращити час завантаження веб-сайту.

Підсумовуючи, BEM — це методологія для написання масштабованого та підтримуваного коду CSS, який може покращити продуктивність і модульність веб-

сайту. Розділивши код на блоки, елементи та модифікатори, розробники можуть створити більш стандартизовану та читабельну кодову базу, полегшуючи підтримку та зміну CSS веб-сайту.

1.4.7. FileZilla - FTP-клієнт

FileZilla — популярне програмне забезпечення з відкритим кодом, яке використовується для передачі файлів між локальним комп'ютером і віддаленим сервером. Це безкоштовний клієнт FTP (протокол передачі файлів), який простий у використанні та підтримує різні протоколи передачі файлів, такі як SFTP (протокол безпечної передачі файлів) і FTPS (FTP через SSL/TLS). У цьому підрозділі ми обговоримо особливості FileZilla і чому це хороший вибір для передачі файлів між локальним комп'ютером і віддаленим сервером.

Однією з головних причин використання FileZilla є простота використання. Інтерфейс користувача простий і зручний для навігації, що робить його чудовим вибором для новачків. Користувачі можуть легко підключитися до віддаленого сервера, ввівши адресу сервера, ім'я користувача та пароль. FileZilla також пропонує функцію перетягування, що дозволяє користувачам легко переміщувати файли між локальним комп'ютером і віддаленим сервером.

FileZilla також має надійні можливості керування файлами. Користувачі можуть переглядати та редагувати віддалені файли безпосередньо з програмного забезпечення, що полегшує керування файлами на сервері. Програмне забезпечення також має вбудовану чергу передавання файлів, що дозволяє користувачам передавати кілька файлів одночасно.

Безпека є головним пріоритетом, коли йдеться про передачу файлів між локальним комп'ютером і віддаленим сервером, і FileZilla пропонує різні функції безпеки для забезпечення безпечної передачі файлів. Він підтримує SFTP, який використовує шифрування для безпечної передачі файлів. Також підтримується FTPS, який забезпечує шифрування SSL/TLS для передачі файлів. FileZilla також дозволяє ко-

ристувачам безпечно зберігати дані про підключення до сервера за допомогою функції, захищеної паролем головного пароля.

Крім того, FileZilla є програмним забезпеченням з відкритим вихідним кодом, що означає, що воно безкоштовне для використання та може бути налаштовано відповідно до вимог користувача. Відкритий код також означає, що програмне забезпечення має велику спільноту розробників, які постійно працюють над удосконаленням і оновленням програмного забезпечення.

Підсумовуючи, FileZilla є чудовим вибором для передачі файлів між локальним комп'ютером і віддаленим сервером. Він пропонує простий та інтуїтивно зрозумілий інтерфейс користувача, надійні можливості керування файлами та різноманітні функції безпеки для забезпечення безпечної передачі файлів. Його кросплатформна сумісність і природа з відкритим вихідним кодом роблять його універсальним і настроюваним вибором для користувачів, які працюють на різних платформах.

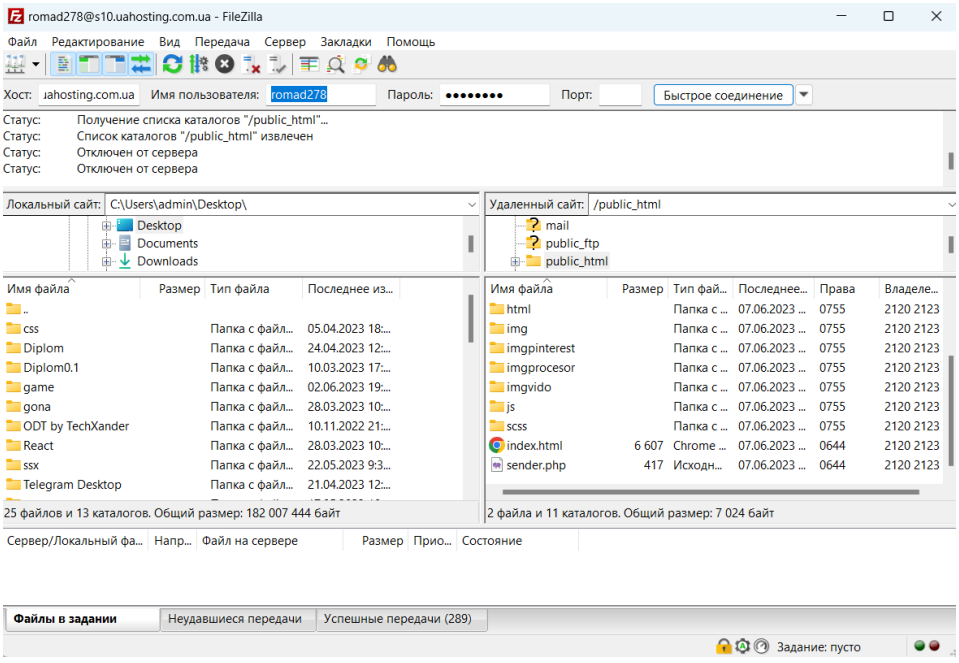

Рис. 1.2. Застосунок FileZilla

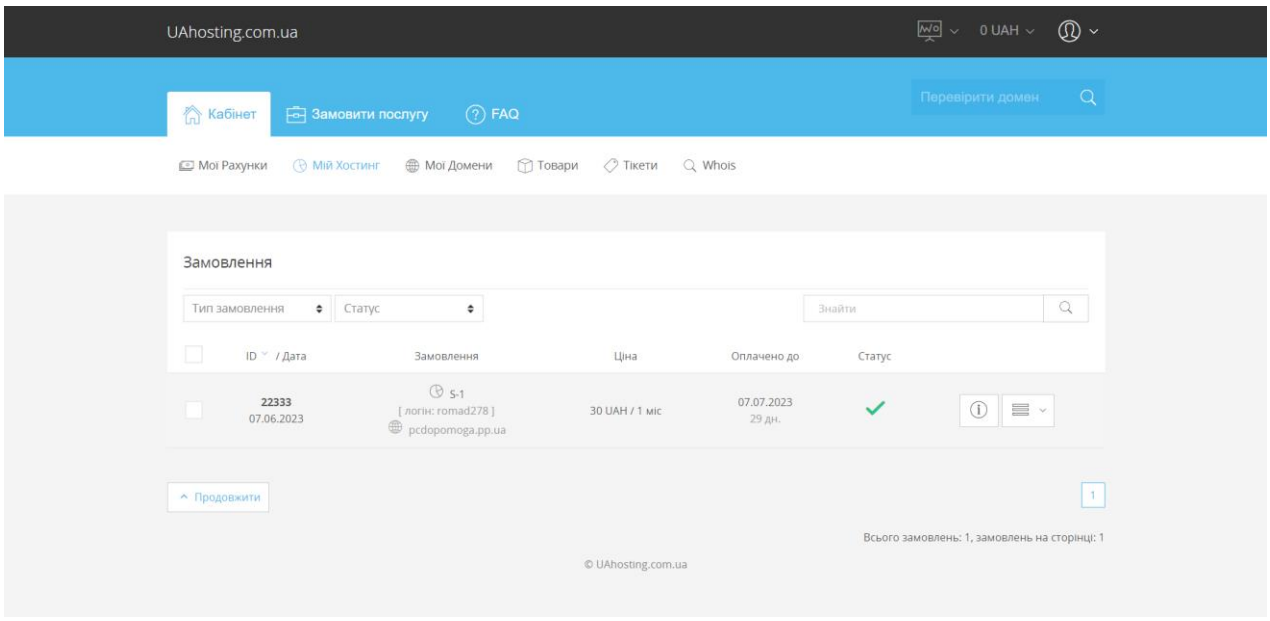

Рис. 1.3. Хостинг проекту

1.5. Висновок до розділу 1

У розділі 1 ми обговорили важливість вибору правильного програмного забезпечення для ПК і те, як це може вплинути на загальну продуктивність системи. Ми почали з опису різноманітних факторів, які слід враховувати під час вибору програмного забезпечення, зокрема сумісності, безпеки, простоти використання та вартості.

Потім ми заглибились у різні типи програмного забезпечення, які необхідні для оптимальної роботи ПК. Детально обговорювалися HTML, JavaScript, CSS, PHP, jQuery, BEM і FileZilla. Ці компоненти необхідні для створення та підтримки функціонального та естетично привабливого веб-сайту. HTML використовується для структурування вмісту веб-сторінки, тоді як CSS використовується для його стилізації та форматування. JavaScript використовується для додавання інтерактивності та динамічного вмісту веб-сторінок, тоді як PHP використовується для створення сценаріїв на стороні сервера. jQuery — це популярна бібліотека JavaScript, яка спрощує написання сценаріїв на стороні клієнта, а BEM — це методологія для організації CSS. Нарешті, FileZilla — популярний FTP-клієнт, який дозволяє користувачам завантажувати файли веб-сайтів і керувати ними.

Підсумовуючи, вибір правильного програмного забезпечення для ПК має вирішальне значення для оптимальної продуктивності, стабільності та безпеки системи. Це вимагає ретельного розгляду таких факторів, як сумісність, простота використання, вартість і безпека. Компоненти, розглянуті в розділі 1, включаючи HTML, JavaScript, CSS, PHP, jQuery, BEM і FileZilla, усі необхідні для створення та підтримки функціонального та естетично привабливого веб-сайту. Важливо постійно оновлювати програмне забезпечення та регулярно встановлювати виправлення безпеки, щоб забезпечити захист системи від зловмисних атак.

РОЗДІЛ 2
РОЗРОБКА WEB ДОДАТКУ «ПІДБІР НОВИХ КОМПЛЕКТУЮЧИХ ДЛЯ ПК»

2.1. Проектування розробки програмного інтерфейсу.

Дизайн веб-сайту «Нові компоненти ПК» складається з різних розділів і елементів, які покращують взаємодію з користувачем і полегшують навігацію.

У верхній частині сайту заголовок помітно відображає назву веб-сайту в центрі, встановлюючи ідентичність бренду. Крім того, квадрат у правому куті заголовка вказує номер версії сайту, надаючи інформацію про оновлення та покращення.

Основна частина сайту містить лівостороннє спадне меню, що складається з п’яти кнопок. Коли натискається кожна кнопка, відкриваються ще три кнопки, розширюючи доступні параметри для користувачів. Одна з цих кнопок надає важливу інформацію про сайт і дозволяє користувачам надсилати повідомлення та запити. Решта чотири кнопки демонструють різні готові колекції, і після натискання користувачі можуть переглянути детальну інформацію про кожну колекцію.

У середній частині корпусу є шість кнопок, прикрашених ілюстраціями частин комп’ютера, включаючи процесор, материнську плату, відеокарту, блок живлення, охолодження та операційну систему. Натискання будь-якої з цих кнопок призводить користувачів до спеціальної сторінки з детальною інформацією про відповідний компонент комп’ютера. Ця функція дозволяє користувачам збирати конкретну інформацію про кожну частину, допомагаючи їм приймати обґрунтовані рішення.

Щоб додати інтерактивності, на сайті є прихована антистресова кнопка. При виявленні та натисканні ця кнопка змінює свій колір випадковим чином, забезпечуючи користувачам моментальне відволікання та розвагу.

Нижня частина сайту містить добре структурований нижній колонтитул із трьома колонками, кожна з яких служить певній меті. Перший стовпець містить назви різних компонентів ПК. Користувачі можуть натиснути назву будь-якого компонента, щоб отримати доступ до спеціальної сторінки з детальною інформацією про цю конкретну частину. Ця функція спрощує процес швидкого пошуку потрібної інформації.

У другому стовпці представлено три назви книг для ПК разом із посиланнями для доступу до цих книг. Цей розділ має на меті надати користувачам додаткові ресурси та довідкові матеріали для покращення їхніх знань і розуміння компонентів ПК.

Третій стовпець зосереджений на контактах веб-сайту та містить важливу інформацію, таку як номер телефону та поштову адресу. Крім того, кнопка «Надіслати повідомлення» дозволяє користувачам зручно зв’язуватися з адміністраторами сайту з будь-якими запитаннями, відгуками чи запитами, які вони можуть мати.

Загалом ця структура сайту пропонує зручний інтерфейс зі зрозумілою навігацією та легким доступом до відповідної інформації. Заголовок демонструє назву сайту та номер версії, забезпечуючи впізнаваність бренду та інформуючи користувачів про оновлення. На корпусі є спадне меню, готові колекції та детальна інформація про компоненти ПК. Прихована кнопка-антистрес додає елемент несподіванки та залучення. Нижній колонтитул забезпечує швидкий доступ до компонентів ПК, книг для ПК та контактів веб-сайтів, підвищуючи зручність використання та полегшуючи взаємодію з користувачем.

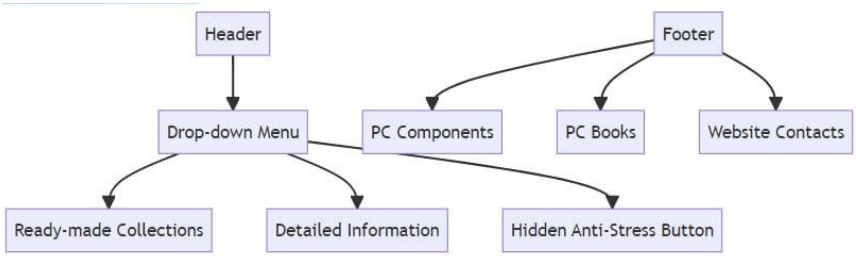

Рис. 2.1. Карта побудови web додатку

2.1.1. Створення каркасів або прототипів низької точності

Є кілька причин, чому каркаси або прототипи сторінок із низькою точністю можуть бути корисними в процесі розробки:

1. Швидка ітерація. Каркаси сторінки низької якості дозволяють швидко та ітеративно змінювати дизайн. Оскільки вони менш деталізовані та займають менше часу на створення порівняно з високоякісними прототипами, дизайнери можуть легко експериментувати з різними варіантами макета, інформаційною ієрархією та моделями взаємодії. Цей ітеративний процес допомагає вдосконалити загальну концепцію дизайну перед тим, як інвестувати значні ресурси у високоточну розробку.

2. Зосередьтеся на структурі. Каркаси сторінок із низькою точністю допомагають дизайнерам зосередитися на структурних аспектах інтерфейсу користувача та потоку інформації. Усунувши візуальні перешкоди та деталі, дизайнери можуть забезпечити інтуїтивно зрозумілу та логічну архітектуру та навігацію сайту. Цей ранній фокус на загальній структурі закладає надійну основу для зручного та ефективного використання.

3. Співпраця та зворотній зв’язок. Каркаси сторінок із низькою точністю є ефективними інструментами спілкування для співпраці й отримання відгуків від зацікавлених сторін, членів команди чи потенційних користувачів. Простота фреймів спонукає до обговорення та сприяє зосередженню уваги на основній функціональності та вмісті. Цей ранній зворотний зв’язок може сприяти прийняттю обґрунтованих проектних рішень і узгодити розуміння командою цілей проекту.

4. Ефективність витрат і часу. Розробка прототипів високої точності може зайняти багато часу та ресурсів. Починаючи з низькоякісних рамок сторінок, дизайнери можуть заощадити час і ресурси на початкових етапах процесу проектування. Це дозволяє їм швидко досліджувати різні можливості дизайну, визначати потенційні проблеми та перевіряти дизайнерські рішення перед тим, як приступити до більш детальної розробки.

5. Дизайн, орієнтований на користувача. Рамки сторінки з низькою точністю переміщують фокус із візуальної естетики на дизайн, орієнтований на користувача. Розставляючи пріоритети для потреб користувачів, взаємодії та організації вмісту, дизайнери можуть гарантувати, що сайт або програма відповідає основним вимогам і ефективно передає свою мету. Такий підхід допомагає створити міцну основу для дизайнерського рішення, орієнтованого на користувача.

Хоча каркас сторінки низької якості бракує візуального блиску та деталей, їхня мета полягає в тому, щоб сприяти дослідженню, ітерації та ранньому відгуку. Використовуючи ці низькоточні прототипи, дизайнери можуть створювати більш ефективні та орієнтовані на користувача проекти, оптимізуючи час, ресурси та співпрацю.

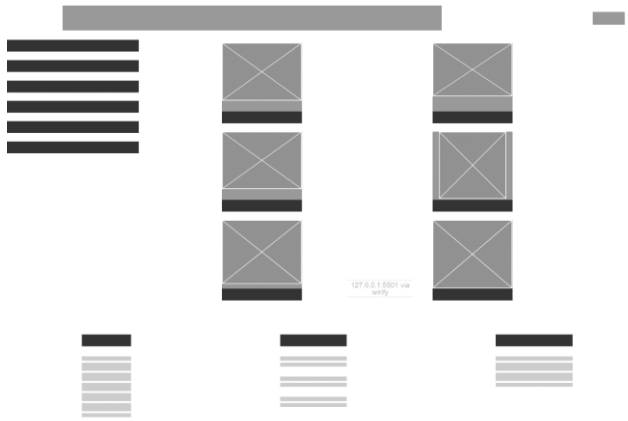

Рис. 2.2. Каркасна конструкція головної сторінки

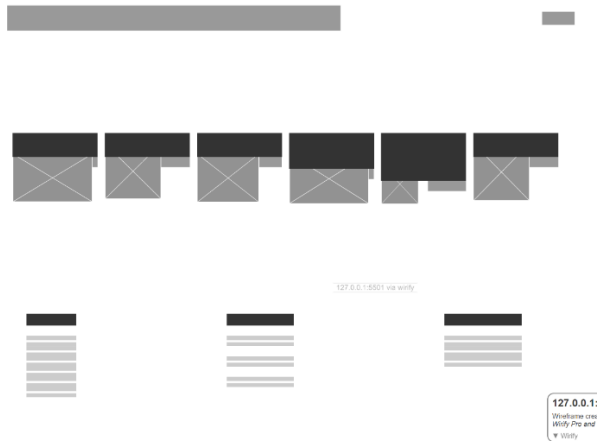

Рис. 2.3. Каркасна конструкція сторінки «Відео карта»

2.1.2. Ітеративне проектування та прототипування

Визначаємо високоякісні прототипи за допомогою Figma, мультиплатформенної служби веб-дизайну, та робимо такі дії:

1. Визначаємо свої цілі. Чітко окреслюємо свої цілі та завдання для прототипу. Зрозумійємо, чого ми хочемо досягти та які аспекти дизайну ми хочемо перевірити чи продемонструвати а в данному випадку сайт «Підбір нових комплектуючих для ПК».

2. Збираємо вимоги. Збираємо всю необхідну інформацію, включаючи вимоги користувачів, інструкції щодо дизайну, вміст і будь-які конкретні функції або взаємодії, які потрібно включити в прототип.

3. Спланувуємо потік користувачів. Спланувуємо потік користувачів і навігацію прототипу. Визначаємо ключові екрани та взаємодії, які мають бути представлені в прототипі, і спланувуємо, як користувачі рухатимуться через інтерфейс. Зображено на рис. 2.2.

4. Каркаси. Починаємо зі створення каркасів низької точності, щоб створити базову структуру та структуру інтерфейсу. Використовуємо прості фігури та заповнювачі для представлення різних елементів, таких як кнопки, текст, зображення та меню навігації. Зображено на рис. 2.2.-2.3.

5. Візуальний дизайн. Коли каркаси встановлені, можем почати додавати елементи візуального дизайну. Використовуючи інструменти дизайну Figma, щоб створювати візуально привабливі інтерфейси, включаючи типографіку, колірні схеми, піктограми та зображення, які відповідають цілям вашого бренду та дизайну.

Рис. 2.4. Візуальний дизайн головної сторінки

6. Бібліотека компонентів. Створюємо бібліотеку компонентів у Figma, щоб забезпечити узгодженість та ефективність процесу проектування. Розробляємо багаторазові компоненти, такі як кнопки, елементи форми та заголовки, які можна легко копіювати та змінювати на кількох екранах.

Рис. 2.5. Компонент у Figma головного заголовка

Рис. 2.6. Компоненти у Figma кнопок

7. Інтерактивне створення прототипів. Використовуємо інтерактивні функції Figma, щоб втілити свій прототип у життя. Додаємо інтерактивні елементи, такі як кнопки, посилання та переходи, щоб імітувати взаємодію користувача та продемонструвати функціональність дизайну.

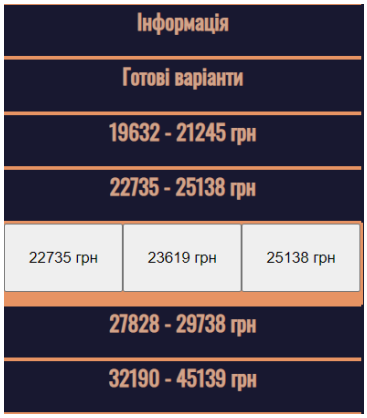

Рис. 2.7. Інтерактивні елемент, такій як перехід

8. Тестуємо та повторюємо. Проводимо тестування зручності використання нашого прототипу, щоб зібрати відгуки та ідеї від користувачів. Визначаємо сфери вдосконалення та повторіть свій дизайн на основі отриманих відгуків. Внесемо необхідні зміни, щоб покращити зручність використання та досвід користувача.

9. Співпраця та зворотний зв’язок. Використовуємо функції співпраці Figma, щоб поділитися своїм прототипом із зацікавленими сторонами, клієнтами чи членами команди. Збираємо відгуки та коментарії та внесемо до необхідні зміни на основі отриманих відгуків.

10. Документація. Задокументуємо дизайнерські рішення, взаємодії та функціональні можливості прототипу. Ця документація слугуватиме довідником для розробників та інших зацікавлених сторін, залучених до етапу впровадження.

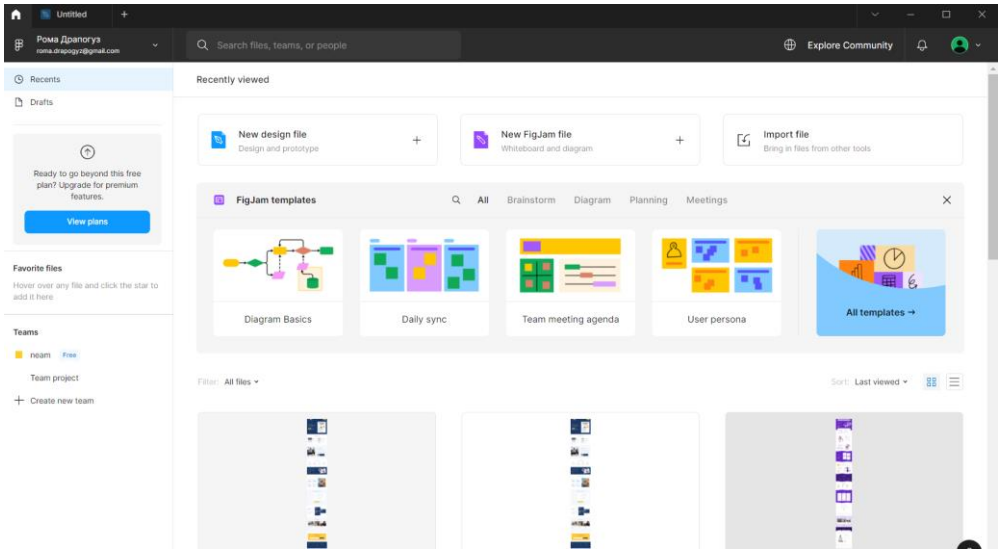

Рис. 2.8. Етап впровадження у Figma

2.2. Процес створення веб-додатку

Початковий крок у процесі створення веб-додатку передбачає отримання та налаштування необхідного середовища розробки. У першому розділі ми ознайомили з обраними технологіями розробки веб-додатку «Підбір нових компонентів для ПК». Ми вибрали Visual Studio Code як інтегроване середовище розробки (IDE).

Почнемо з того, що налаштування середовища розробки має вирішальне значення, оскільки воно надає необхідні інструменти та ресурси для створення веб-додатку. Зазвичай це передбачає завантаження та встановлення необхідних програмних компонентів і фреймворків, які полегшать процес розробки.

У контексті нашого веб-додатку «Вибір нових компонентів для ПК» ми ретельно визначили та вибрали конкретні технології, які найкраще відповідатимуть нашим вимогам. Ці технології були детально описані в першому розділі нашого шляху розробки.

Серед різноманітних доступних варіантів ми вибрали Visual Studio Code як IDE. Visual Studio Code — це дуже популярний і універсальний редактор коду, який пропонує широкий спектр функцій і розширень, що робить його ідеальним для розробки веб-додатків. Він забезпечує зручний інтерфейс і підтримує кілька мов програмування, що робить його адаптованим до потреб різних проектів.

Вибираючи Visual Studio Code, ми розуміємо, що це середовище розробки є надійним, ефективним і добре підходить для поставленого завдання. Цей вибір дозволяє нам скористатися широкими можливостями та широкою підтримкою спільноти, які пропонує IDE.

Підсумовуючи, початковий етап розробки веб-додатків включає придбання та налаштування середовища розробки. Ми спеціально вибрали Visual Studio Code як нашу бажану IDE для проекту «Вибір нових компонентів для ПК». Це рішення гарантує, що ми маємо надійний і універсальний набір інструментів для ефективного створення нашої веб-програми.

2.2.1. Налаштування Visual Studio Code для розробки веб-додатку «Підбір нових компонентів для ПК»

Етапи налаштування Visual Studio Code:

1. Завантаження та інсталяція Visual Studio Code. Щоб налаштувати Visual Studio Code, почнемо із завантаження останньої версії з офіційного веб-сайту. Виберемо відповідний інсталятор залежно від використовуваної операційної системи (Windows, macOS або Linux). Запустимо програму встановлення та дотримуємося вказівок на екрані, щоб завершити процес встановлення.

2. Налаштування коду Visual Studio для розробки веб-додатків. Запускаємо Visual Studio Code після встановлення. Установлюємо відповідні розширення: Visual

Studio Code пропонує широкий спектр розширень, які покращують його функціональність для веб-розробки. Деякі важливі розширення для розробки веб-додатків включають інтеграцію HTML, CSS, JavaScript і Git. Установлюємо ці розширення, перейшовши до перегляду розширень у коді Visual Studio та знайшовши потрібні.

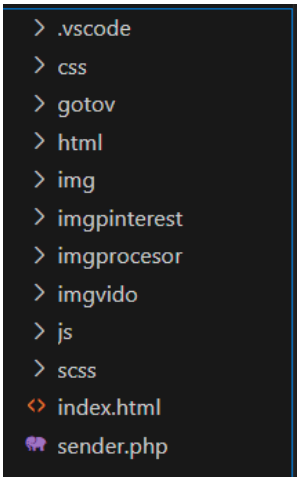

Рис. 2.9. Розширення які ми використовували у коді Visual Studio

3. Налаштування інтерфейсу користувача та параметрів. Visual Studio Code забезпечує настроюваний інтерфейс користувача (UI) відповідно до індивідуальних уподобань і підвищення продуктивності. Змінюємо макет інтерфейсу користувача, колірні теми та параметри шрифту відповідно до особистих уподобань. Налаштуємо комбінації клавіш і фрагменти коду, щоб спростити робочий процес розробки. Налаштуємо параметри на панелі «Параметри», доступній через меню «Файл» або за допомогою комбінації клавіш (Ctrl + ,).

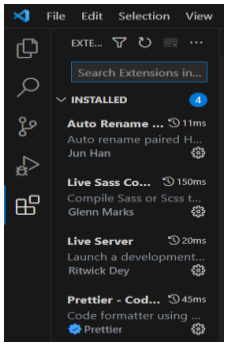

Рис. 2.10. Параметри на панелі «Параметри»

4. Використання вбудованого терміналу. Visual Studio Code включає інтегрований термінал, що усуває необхідність перемикатися між IDE та окремим вікном терміналу. Інтегрований термінал підтримує взаємодію командного рядка та дозволяє запускати сценарії, компілювати код і виконувати різні завдання розробки. Отримаємо доступ до вбудованого терміналу, перейшовши до меню «Вигляд» і вибравши «Термінал» або скориставшись ярликом (Ctrl + `).

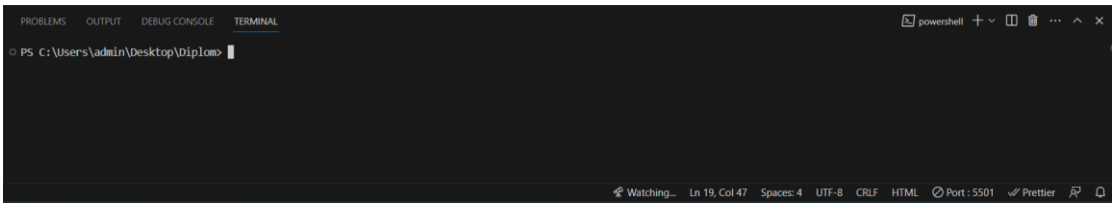

Рис. 2.11. Термінал в Visual Studio

5. Використання IntelliSense і функцій навігації по коду. Visual Studio Code надає потужні можливості IntelliSense, які допомагають у завершенні коду, пропозиціях і виявленні помилок. Він підтримує різні мови програмування, включаючи HTML, CSS, JavaScript та багато інших. IntelliSense допомагає оптимізувати кодування, надаючи пропозиції з урахуванням контексту, зменшуючи кількість помилок і покращуючи якість коду. А також Prettier – code formatter який форматує код, який можна інтегрувати з Visual Studio Code, щоб забезпечити послідовне та стандартизоване форматування коду.

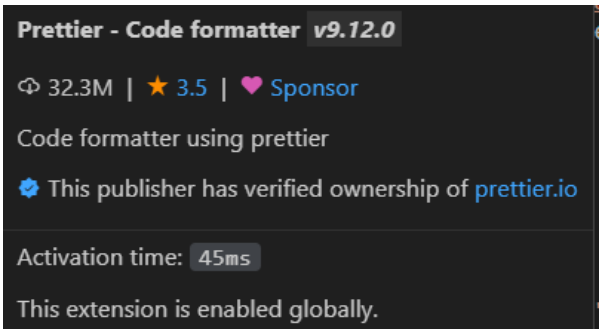

Рис. 2.12. Prettier code formatter

А також Live sass compiler який дозволяє компілювати файли Sass або SCSS у CSS у реальному часі, коли ви вносите зміни.

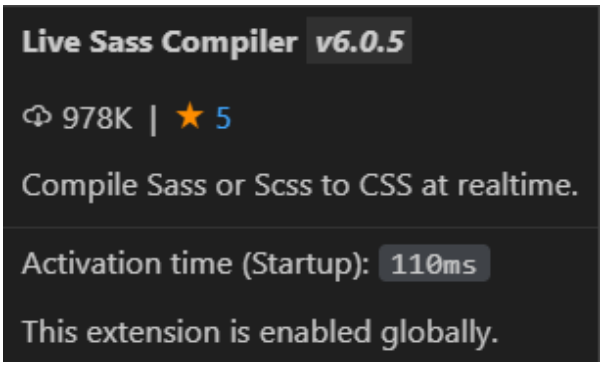

Рис. 2.13. Live sass compiler

6. Налагодження веб-додатків. Visual Studio Code пропонує надійні можливості налагодження веб-додатків, що дозволяє розробникам ефективно виявляти та виправляти проблеми. Налаштуємо налагодження, встановлюючи точки зупину, перевіряючи змінні та покроково проходячи процес виконання коду. Налагодження можна виконувати локально або віддалено, залежно від вимог веб-додатку.

Рис. 2.14. Ефект при виявляні проблеми

7. Використовуємо автоматизацію завдань за допомогою Visual Studio Code. Visual Studio Code надає функції автоматизації завдань для автоматизації повторюваних завдань розробки. Він підтримує такі програми запуску завдань, як Grunt, Gulp і npm-скрипти, що дозволяє бездоганно інтегруватись із процесами збирання. Визначаємо завдання у файлі "tasks.json" у межах проекту та виконуємо їх безпосередньо з Visual Studio Code.

8. Співпраця та обмін кодом. Visual Studio Code полегшує співпрацю та обмін кодом між розробниками.

2.2.2. Використання HTML в проекті веб-додатку «Підбір нових компонентів для ПК»

HTML (Hypertext Markup Language) є основним будівельним блоком веб-розробки, і його використання в проекті веб-додатку «Вибір нових компонентів для ПК» відіграє вирішальну роль у створенні структури, вмісту та інтерактивності додатка. Ось докладний опис того, як HTML використовується в проекті:

1. Структура та семантика:

- HTML використовуємо для визначення загальної структури веб-додатку. Він дає змогу створювати різні елементи, такі як верхні та нижні колонтитули, навігаційні меню, бічні панелі та розділи вмісту;

- Семантичні теги HTML, такі як `<header>`, `<nav>`, `<main>`, `<section>` і `<footer>`, використовуються для надання значення та структури різним частинам програми;

- Використовуючи семантичні теги, проект забезпечує доступність і покращує пошукову оптимізацію (SEO).

2. Презентація змісту:

- HTML відповідає за представлення текстового та мультимедійного вмісту у веб-додатку;

- Заголовки (`<h1>` до `<h6>`) використовуються для структурування та ієрархічної організації вмісту, забезпечуючи ясність і легкість навігації;

- Абзаци (`<p>`) використовуються для представлення текстової інформації в структурованому вигляді;

- Списки (`<ul>`, `<ol>`, `<li>`) використовуються для створення маркірованих і нумерованих списків для представлення інформації в стислому та читабельному форматі;

- HTML також дозволяє вставляти медіаконтент, наприклад зображення (`<img>`), відео (`<video>`) і аудіо (`<audio>`).

3. Форми та введення користувача:

- HTML використовується для створення інтерактивних форм, які дозволяють користувачам вводити дані, надсилати інформацію та взаємодіяти з веб-програмою;

- Такі елементи форми, як `<input>`, `<select>`, `<textarea>` і `<button>`, використовуються для запису введених користувачем даних і забезпечення засобів взаємодії;

- Різні типи введення (текст, електронна пошта, пароль тощо) і атрибути (обов'язкові, заповнювачі тощо) використовуються для покращення взаємодії з користувачем і забезпечення перевірки.

4. Гіперпосилання та навігація:

- Якорі HTML (`<a>`) і атрибут `href` використовуються для створення гіперпосилань, що дозволяє переходити між різними сторінками або розділами в програмі;

- Навігаційні меню та панелі навігації створюються за допомогою списків HTML і стилів CSS для забезпечення легкого доступу до різних частин програми.

5. Інтеграція з CSS і JavaScript:

- HTML є основою для інтеграції CSS і JavaScript у проект веб-додатку;

- CSS використовується для стилізації та візуального покращення елементів HTML, визначення кольорів, типографіки, макета та загального вигляду програми;

- JavaScript використовується для додавання інтерактивності та динамічної поведінки елементам HTML, уможливлюючи такі дії, як перевірка форми, маніпулювання даними та обробка подій.

6. Метадані та SEO:

- Розділ `<head>` HTML використовується для надання метаданих та інформації про веб-програму;

- Тег <title> визначає назву веб-сторінки, яка відображається у вкладці або вікні браузера;

- Мета-теги, такі як `<meta name="description">` і `<meta name="keywords">`, використовуються для надання опису та ключових слів, релевантних вмісту, покращуючи SEO та видимість у пошуковій системі.

7. Чуйний дизайн і доступність:

- HTML відіграє важливу роль у створенні адаптивного веб-дизайну, який адаптується до різних розмірів екрана та пристроїв;

- Елементи HTML структуровані та оформлені за допомогою медіа-запитів CSS, щоб забезпечити оптимальний перегляд на різних пристроях, включаючи настільні ПК, планшети та мобільні пристрої;

- Атрибути HTML, такі як `alt` для зображень і атрибути `aria` для спеціальних можливостей, використовуються для покращення доступності, гарантуючи, що веб-програма доступна для використання окремими особами.

2.3. Реалізація елементів HTML у веб-додатку.

Продемонструємо HTML елементи, використані при створенні головної сторінки web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів:

Рис. 2.15. Початкова частина HTML коду

Цей код є початковою частиною HTML-документа і визначає тип документа та мову, в якій він написаний. Давайте розглянемо детальніше:

1. <!DOCTYPE html> - цей рядок вказує на тип документа і називається декларацією типу документа (Document Type Declaration або DOCTYPE). В даному випадку, "html" вказує, що це HTML-документ.

2. <html lang="uk"> - цей тег позначає початок HTML-документа і визначає мову, в якій написаний документ. У даному випадку, значення атрибута "lang" встановлено на "uk", що вказує, що цей документ написаний українською мовою.

Таким чином, цей код визначає, що ми маємо справу з HTML-документом, написаним українською мовою. Це важливо для веб-браузера, оскільки це допомагає йому правильно інтерпретувати і відображати контент з урахуванням правил і особливостей української мови.

```
<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>Помічник комплектуючих ПК</title>
    <link rel="stylesheet" href="css/style.css">
    <script src="https://kit.fontawesome.com/4c44893623.js" crossorigin="anonymous"></script>
</head>
```

Рис. 2.16. Розмітка, яка знаходиться всередині елементу head

Наданий код представляє частину HTML-розмітки, яка знаходиться всередині елементу `<head>`. Давайте розберемо кожен елемент окремо:

1. `<meta charset="UTF-8">` - цей елемент встановлює кодування символів документа на UTF-8. UTF-8 є широко використовуваним стандартом кодування, який підтримує велику кількість символів і мов.

2. `<meta http-equiv="X-UA-Compatible" content="IE=edge">` - цей елемент вказує на режим сумісності з браузером Internet Explorer і встановлює налаштування `X-UA-Compatible` на `IE=edge`. Це означає, що браузер Internet Explorer повинен використовувати найсучасніший можливий режим відображення.

3. `<meta name="viewport" content="width=device-width, initial-scale=1.0">` - цей елемент встановлює параметри відображення (viewport) для мобільних пристроїв.

4. `<title>Помічник комплектуючих ПК</title>` - цей елемент встановлює заголовок (title) сторінки, який буде відображатися у вкладці браузера або як заголовок сторінки в пошукових системах.

5. `<link rel="stylesheet" href="css/style.css">` - цей елемент встановлює зовнішній файл стилів (CSS) для сторінки. Вказується шлях до файлу стилів

(`css/style.css`), який знаходиться в папці "css". Файл стилів використовується для візуального оформлення елементів на сторінці.

6.`<script src = "https:// kit.fontawesome.com/ 4c44893623.js" crossorigin ="anonymous"> </script>` - цей елемент завантажує зовнішній JavaScript-файл з бібліотекою Font Awesome. Зазначений файл містить набір іконок.

```
<body>
    <div class="header">
        <div class="header__logo">
            <i class="fa-solid fa-desktop"></i>
            <a> Підбір нових комплектуючих для персонального комп'ютера </a>
        </div>
        <div class="header__version">
            <a>Версія 0.1</a>
        </div>
    </div>
```

Рис. 2.17. Розмітка для веб-сторінки з простою структурою заголовка

Наданий код представляє HTML-розмітку для веб-сторінки з простою структурою заголовка (header) і його складовими елементами. Давайте розберемо кожен елемент починаючи з найвищого рівня вкладеності:

1. `<body>` - це основний контейнер для всього вмісту сторінки, що знаходиться всередині нього.

2. `<div class="header">` - цей елемент представляє заголовок сторінки і містить два дочірні елементи.

3. `<div class="header__logo">` - цей елемент містить логотип заголовка, який представлений іконкою та текстовою посиланням. `<i class="fa-solid fa-desktop"></i>` - це елемент `<i>` з класом `fa-solid fa-desktop`, що використовується для відображення іконки. Ймовірно, використовується якась зовнішня бібліотека (наприклад, Font Awesome), яка надає цей клас для відображення іконки "desktop" (настільний комп'ютер). `<a> Підбір нових комплектуючих для персонального комп'ютера </a>` - це текстове посилання, яке містить заголовок для підбору нових комплектуючих для персонального комп'ютера.

4. `<div class="header__version">` - цей елемент містить інформацію про версію і також містить текстове посилання. `<a>Версія 0.1</a>` - це текстове посилання, яке містить інформацію про версію (у даному випадку "Версія 0.1").

Таким чином, код створює заголовок сторінки з логотипом (іконкою) і описом, а також інформацією про версію.

```
<div class="center">

    <div class="center__garmohka">
        <div class="garmohka__item">
            <h3>Інформація</h3>
        </div>
        <div class="garmohka__content"><a href="/html/Meta.html"><button>Діяльність</button></a><a
                href="/html/contact.html"><button>Контакти</button></a>
        </div>

        <div class="garmohka__item">
            <h3>Готові варіанти</h3>
        </div>

        <div class="garmohka__item">
            <h3>19632 - 21245 грн</h3>
        </div>
        <div class="garmohka__content"><a href="/html/19632.html"><button>19632 грн</button></a><a
                href="/html/20636.html"><button>20636
                    грн</button></a><a href="/html/21245.html"><button>21245
                    грн</button></a>
        </div>
```

Рис. 2.18. Розмітка для блоку з класом "center" і його складовими елементами

Наданий код представляє HTML-розмітку для блоку з класом "center" і його складовими елементами. Давайте розберемо кожен елемент починаючи з найвищого рівня вкладеності:

1. `<div class="center">` - цей елемент представляє блок з класом "center", який використовується, для вирівнювання вмісту по центру.

2. `<div class="center__garmohka">` - цей елемент представляє внутрішній блок з класом "center__garmohka".

3. `<div class="garmohka__item">` - цей елемент представляє окремий елемент (item) всередині блоку "garmohka". Його вмістом є заголовок (`<h3>Інформація</h3>`), який представляє собою текстовий заголовок третього рівня.

4. `<div class="garmohka__content">` - цей елемент представляє блок з класом "garmohka__content". Він містить два елементи посилання (`<a>`) з кнопками всере-

дині них. Кнопки використані для навігації до інших сторінок з інформацією про сайт та контакти. Кнопка "Діяльність" - має текст "Діяльність" і вказує посилання на сторінку з шляхом "/html/Meta.html". Кнопка "Контакти" - має текст "Контакти" і вказує посилання на сторінку з шляхом "/html/contact.html".

5. Далі слідують подібні блоки "garmohka__item" і "garmohka__content" зі своїми власними заголовками (`<h3>`) і посиланнями в кнопках. Кожен блок має заголовок, який відображає діапазон цін, і блок з посиланнями, які ведуть до сторінок з готовими збірками.

У загальному розумінні, цей код представляє блок зі списком інформації про різні готові збірки за вказаною ціною.

```
<div class="center__antistres"></div>
```

Рис. 2.19. Кнопка антистрес на сторінці

Цей елемент створює контейнерний блок, який використовується для розміщення кнопки антистрес на сторінці. У даному випадку, блок має клас "center__antistres". Проте, оскільки цей блок не містить вмісту або додаткових елементів, його зовнішній вигляд буде залежати від визначених стилів CSS та Js для класу "center__antistres".

```
<div class="center__componentu">
    <a href="html/materinka.html">
        <div class="materinka">

            <img src="/img/materinka.png">
            <h3>Материнська плата</h3>
        </div>
    </a>
```

Рис. 2.20. Кнопка на головній сторінці с посиланням на сторінку з материнками

Наданий код представляє HTML-розмітку, яка містить два блоки з класами "center__componentu" і "center__componentu2". Давайте розберемо кожен елемент починаючи з найвищого рівня вкладеності:

1. `<div class="center__componentu">` і `<div class="center__componentu2">` - ці елементи представляють контейнерні блоки з класами "center__componentu" і "center__componentu2" відповідно. Ім'я "componentu" використовується для розміщення кнопок с компонетами пк.

2. `<a href="html/materinka.html">` - цей елемент `<a>` встановлює посилання на сторінку "html/materinka.html". Він створює активне посилання, на яке можна натиснути, щоб перейти на вказану сторінку з материнками.

3. `<div class="materinka">` - цей елемент `<div>` представляє блок з класом "materinka". Ім'я "materinka" було придумано для розмуння де саме материньска плата. Внутрішній вміст блоку містить зображення і заголовок.

4. `<img src="/img/materinka.png">` - цей елемент `<img>` встановлює зображення для компонента "Материнська плата". Атрибут `src` вказує шлях до зображення, в даному випадку "/img/materinka.png".

5. `<h3>Материнська плата</h3>` - цей елемент `<h3>` встановлює заголовок "Материнська плата" для компонента.

Всі ці елементи розташовані всередині посилання `<a>`, що означає, що при натисканні на блок з класом "materinka" користувач буде перенаправлений на сторінку "html/materinka.html". Вміст блоку відображатиметься як активне посилання, зображення та текст "Материнська плата".

І так само повторюємо код з іншими компонетами як: відео карта, процесор, охолодження, оперативна пам’ять та блок живлення.

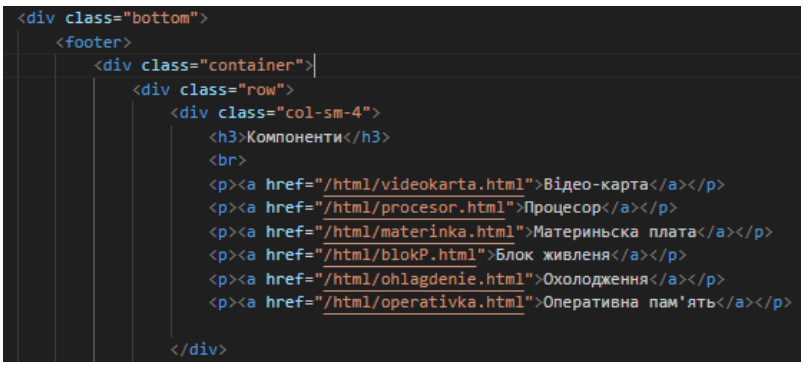

Рис. 2.21. Нижньої частина сторінки

Наданий код містить початок створення нижньої частини сторінки (bottom) і футера. Ось опис кожного елемента:

1. `<div class="bottom">` - цей елемент `<div>` представляє нижню частину сторінки і містить футер.

2. `<footer>` - елемент `<footer>` відображає футер сторінки. Футер містить додаткову інформацію про сторінку, посилання, контакти та іншу додаткову інформацію для користувачів.

3. `<div class="container">` - цей елемент `<div>` створює контейнер для розміщення вмісту футера.

4. `<div class="row">` - елемент `<div>` з класом "row" визначає рядок, в якому будуть розташовані різні колонки футера.

5. `<div class="col-sm-4">` - цей елемент `<div>` з класом "col-sm-4" визначає колонку футера, яка займає 1/3 ширини контейнера на пристроях з розміром екрану sm (наприклад, на малих екранах). У цій колонці розташовуються інформація, посилання та контакти.

Цей фрагмент коду описує початок структури футера з однією колонкою, яку можна заповнити додатковим вмістом. Залежно від продовження коду, можуть бути додані інші колонки для створення більш складеної структури футера.

6. `<h3>Компоненти</h3>` - цей елемент `<h3>` є заголовком першої колонки меню. Він відображає текст "Компоненти".

7. `<p><a href="/html/videokarta.html">Відео-карта</a></p>` - цей елемент `<p>` містить гіперпосилання `<a>` на сторінку "/html/videokarta.html". Текст гіперпосилання "Відео-карта" відображається в меню. При натисканні на це гіперпосилання користувач буде перенаправлений на вказану сторінку зі списком відеокарт.

8. `<p><a href="/html/operativka.html">Оперативна пам'ять</a></p>` - аналогічно до попереднього елемента, ці елементи `<p>` містять гіперпосилання `<a>` з відповідними текстами (назвами компонентів) і посиланнями на відповідні сторінки зі списками компонентів.

Компоненти меню включають "Відео-карта", "Процесор", "Материнська плата", "Блок живлення", "Охолодження" і "Оперативна пам'ять". Кожен з цих пунктів меню є посиланням на відповідну сторінку зі списком компонентів.

```
<div class="col-sm-4">
    <h3>Інфо-центр</h3>
    <br>
    <p><a href="https://www.amazon.com/Building-Perfect-Robert-Bruce-Thompson/dp/1449388248">Building
        the Perfect PC</a></p>
    <br>
    <p><a href="https://www.amazon.com/Building-PC-Dummies-3rd/dp/0764507826">Building a PC For
        Dummies</a>
    </p>
    <br>
    <p><a href="https://www.amazon.com/Ultimate-Guide-Building-Gaming-Century/dp/B0BF381XJZ">The
        Ultimate Guide</a></p>
</div>

<div class="col-sm-4">
    <h3>Контакти</h3>
    <br>

    <p><a href="tel:+380678700232">+38 067 870 02 </a></p>
    <p>E-mail: drapromario@mail.com</p>
    <p><a href="/html/contact.html">Відправити повідомлення</a></p>

</div>
```

Рис. 2.22. Посилання в нижній частині сайту на книги та контакти

Наданий код містить дві колонки (`<div class="col-sm-4">`) у футері. Ось опис кожного елемента:

1. `<div class="col-sm-4">` - цей елемент `<div>` з класом "col-sm-4" визначає другу колонку футера, яка займає 1/3 ширини контейнера на пристроях з розміром екрану sm (наприклад, на малих екранах). У цій колонці розміщена інформація про "Інфо-центр".

2. `<h3>Інфо-центр</h3>` - заголовок `<h3>` "Інфо-центр" вказує на тему, яка міститься в цій колонці.

3. `<p><a href="https://www.amazon.com/Building-Perfect-Robert-Bruce-Thompson/dp/1449388248">Building the Perfect PC</a></p>` - цей рядок створює посилання `<a>` на книгу "Building the Perfect PC" на Amazon.

4. `<p><a href="https://www.amazon.com/Building-PC-Dummies-3rd/dp/0764507826" >Building a PC For Dummies</a></p>` - цей рядок створює посилання `<a>` на книгу "Building a PC For Dummies" на Amazon.

5. `<p><a href="https://www.amazon.com/Ultimate-Guide-Building-Gaming-Century/dp/B0BF381XJZ">The Ultimate Guide</a></p>` - цей рядок створює посилання `<a>` на книгу "The Ultimate Guide" на Amazon.

6. `<div class="col-sm-4">` - цей елемент `<div>` з класом "col-sm-4" визначає третю колонку футера, яка також займає 1/3 ширини контейнера на пристроях з розміром екрану sm. У цій колонці розміщена інформація про "Контакти".

7. `<h3>Контакти</h3>` - заголовок `<h3>` "Контакти" вказує на категорію інформації, яка міститься в цій колонці.

8. `<p><a href="tel:+380678700232">+38 067 870 02 </a></p>` - цей рядок містить посилання `<a>` на телефонний номер з кодом країни.

9. `<p>E-mail: drapromario@mail.com</p>` - цей рядок відображає текст "E-mail: drapromario@mail.com". Ймовірно, це електронна адреса контактної особи або організації, пов'язаної з веб-сайтом.

10. `<p><a href="/html/contact.html">Відправити повідомлення</a></p>` - цей рядок містить посилання `<a>` з текстом "Відправити повідомлення". Посилання спрямовує на сторінку "/html/contact.html", ця сторінка, де користувач може надіслати повідомлення або зв'язатися зі службою підтримки, контактними особами та формою зворотного зв'язку.

Продемонструємо HTML елементи, використані при створенні сторінки з збірками відео-карт web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

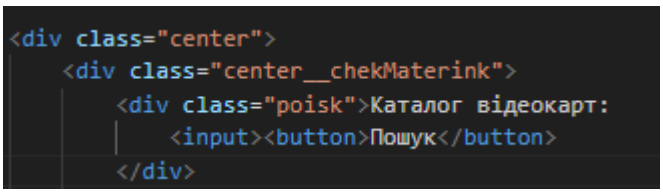

Рис. 2.23. Пошук відео-карт по каталогу

Наданий код містить елемент `<div>` з класом "center", який містить ще один `<div>` з класом "center__chekMaterink". В середині "center__chekMaterink" розташований елемент `<div>` з класом "poisk", що представляє собою поле пошуку в каталозі відеокарт:

1. `<div class="center">` - цей елемент виступає як контейнер, який вирівнює свої дочірні елементи у центрі;

2. `<div class="center__chekMaterink">` - цей елемент відображає блок, який містить деякі перевірки або фільтри для вибору материнських плат;

3. `<div class="poisk">` - цей елемент містить поле пошуку в каталозі відеокарт;

4. `"Каталог відеокарт: "` - цей текст відображає мітку або заголовок для поля пошуку, який показує, що саме шукається у каталозі;

5. `<input>` - цей елемент представляє поле введення, де користувач може вводити текст для пошуку;

6. `<button>Пошук</button>` - цей елемент представляє кнопку "Пошук", за допомогою якої можна ініціювати пошук в каталозі відеокарт після введення користувачем необхідного запиту.

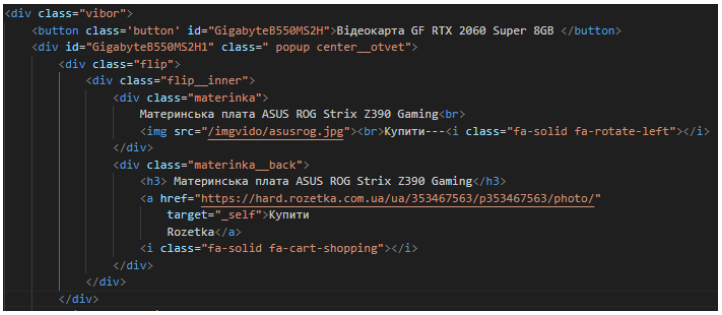

Рис. 2.24. Материнка яка підібрана під дану відеокарту

Наданий код містить елемент `<div>` з класом "vibor", який включає в себе кнопку та розгортаючийся блок з інформацією про вибрану відеокарту:

1. `<div class="vibor">` - цей елемент представляє контейнер для вибору компонента (в даному випадку відеокарти).

2. `<button class='button' id="GigabyteB550MS2H">Відеокарта GF RTX 2060 Super 8GB </button>` - цей елемент представляє кнопку, яка показує назву вибраної відеокарти. У даному випадку, кнопка має текст "Відеокарта GF RTX 2060 Super 8GB" та ідентифікатор "GigabyteB550MS2H".

3. `<div id="GigabyteB550MS2H1" class="popup center__otvet">` - цей елемент є розгортаючимся блоком, який містить додаткову інформацію про вибрану відеокарту. Він має ідентифікатор "GigabyteB550MS2H1" та класи "popup" і "center__otvet".

4. `<div class="flip">` - цей елемент представляє блок для здійснення ефекту перевертання (flip) вмісту в середині.

5. `<div class="flip__inner">` - цей елемент є внутрішнім блоком для забезпечення ефекту перевертання (flip) всередині блока "flip".

6. `<div class="materinka">` - цей елемент відображає компонент "материнська плата" і містить назву материнської плати, зображення та посилання на її придбання.

7. `<div class="materinka__back">` - цей елемент відображає зворотню сторону компонента "материнська плата" і містить детальнішу інформацію про неї та посилання для покупки.

Загалом, код створює вибір відеокарти та надає користувачу можливість дізнатися більше про цей компонент та придбати його. Точно такий же приклад використовується на інших компонентах таких як: відео-карта, процессор, охолодження, блок-живлення, оперативна-пам'ять. Також є пункти деталі де не має самої деталі бо вона не потрібна в цій збірці і показує просто текст чуму намає цієї деталі.

Продемонструємо HTML елементи, використані при створенні сторінки з інформацією web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

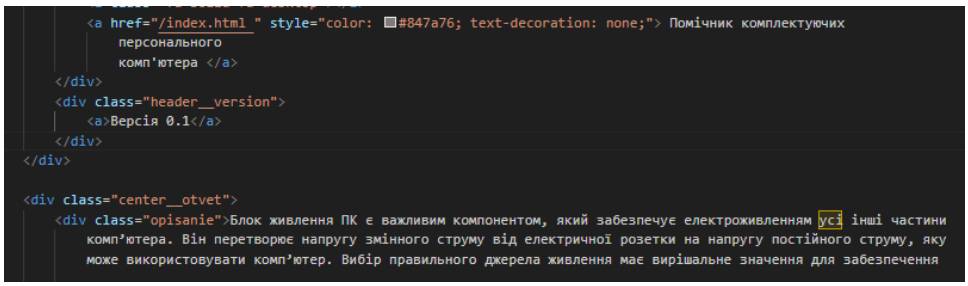

Рис. 2.25. Заголовок, логотип, версія, блок опису та інформацію про блок живлення ПК

Цей код представляє розмітку HTML сторінки. Основні елементи коду включають заголовок, логотип, версію, блок опису та інформацію про блок живлення ПК:

1. `<body>` - тег, який визначає тіло документа;

2. `<a href="/index.html " style="color: #847a76; text-decoration: none;">` - посилання на головну сторінку зі стилізацією тексту;

3.`<div class="header__version">` - блок з версією;

4. `<a>Версія 0.1</a>` - текстове посилання, яке відображає версію програми;

5. `<div class="center__otvet">` - блок з центральною інформацією;

6. `<div class="opisanie">` - блок з описом.

Цей код, при використанні відповідного CSS стилю, може створювати веб-сторінку з заголовком, логотипом, версією та інформацією.

Продемонструємо HTML елементи, використані при створенні сторінки з контактами web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

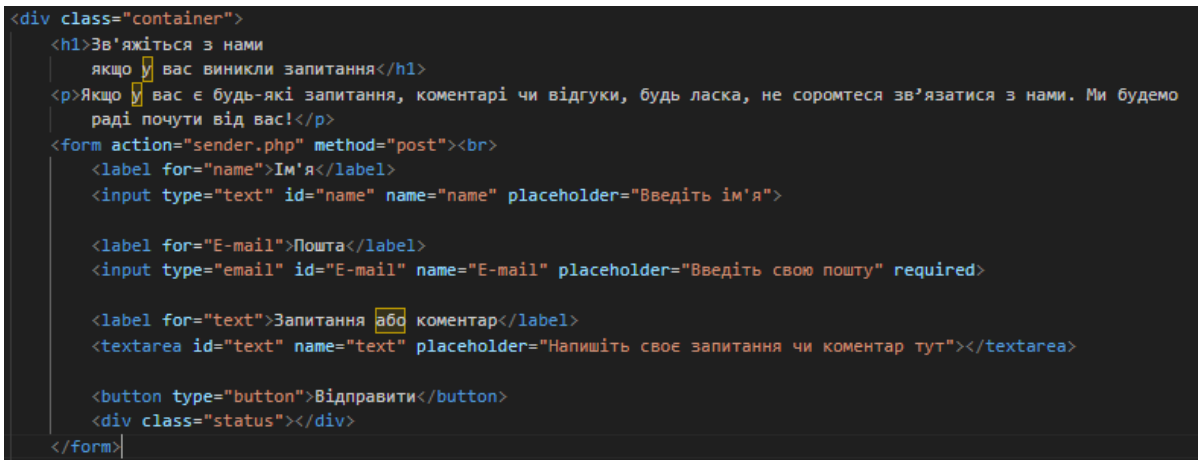

Рис. 2.26. Форма зв'язку

Цей код є HTML-кодом для створення форми зв'язку. Він складається з різних елементів, які дозволяють користувачам вводити свої дані та надсилати їх на сервер. Ось докладний опис кожного елемента коду:

1. `<div class="container">` - це контейнер, який містить всю форму.

2. `<h1>Зв'яжіться з нами якщо у вас виникли запитання</h1>` - заголовок форми, що повідомляє користувачам, що вони можуть зв'язатися з компанією.

3. `<p></p>` - параграф з додатковою інформацією для користувачів.

4. `<form action="sender.php" method="post">` - відкриваючий тег форми. `action` вказує URL або шлях до обробника форми на сервері, а `method` вказує метод відправки даних (у цьому випадку використовується метод "POST").

5. `<label for="name">Ім'я</label>` - мітка для полі імені.

6. `<input type="text" id="name" name="name" placeholder="Введіть ім'я">` - текстове поле для введення імені користувача. `id` та `name` використовуються для ідентифікації поля на сервері, а `placeholder` встановлює підказку для користувача.

7. `<label for="E-mail">Пошта</label>` - мітка для полі електронної пошти.

8. `<input type="email" id="E-mail" name="E-mail" placeholder="Введіть свою пошту" required>` - поле для введення електронної пошти користувача. `type` встановлено як "email" для автоматичної перевірки правильності формату електронної пошти. Опція `required` вказує, що поле є обов'язковим для заповнення.

9. `<label for="text">Запитання або коментар</label>` - мітка для полі введення запит.

10. `<button type="button">Відправити</button>` - це кнопка "Відправити", яка дозволяє користувачам надіслати введені дані з форми. У даному випадку, `type` встановлено як "button", що означає, що кнопка не має стандартної поведінки відправки форми. Це означає, що ми додали JavaScript-код для обробки натискання кнопки та відправки даних.

11. `<div class="status"></div>` - це блок `<div>` з класом "status. Цей елемент використовується для відображення повідомлень про статус відправки форми або будь-яких повідомлень про помилки.

```
<div class="cont">
    <div class="medix">
        <h3>Написати можна в такий час:</h3>
        <p>с 10:00 до 18:00 пн-пт</p>
        <p>с 13:00 до 15:00 сб-нд</p>
    </div>
    <div class="social-media">
        <a href=" https://t.me/,,,,,rOmA,,,,,,."><i class="fa-brands fa-telegram"></i></a>
        <a href="https://www.instagram.com/change_phonenumber/"><i
                class="fa-brands fa-square-instagram"></i></a>
        <a href="https://www.facebook.com/profile.php?id=100064151343934"><i
                class="fa-brands fa-facebook"></i></a>
    </div>
</div>
```

Рис. 2.27. Інформація роботи сайту та посилання на соцмережі

Цей код представляє розмітку HTML-сторінки і містить два блоки з класами "cont" і "medix", а також блок з класом "social-media". Ось опис кожного елемента коду:

1. `<div class="cont">` - це зовнішній контейнер, який містить обидва блоки "medix" і "social-media". Клас "cont" використовується для стилізації.

2. `<div class="medix">` - це блок, який містить інформацію про години роботи.

3. `<div class="social-media">` - цей блок містить посилання на соціальні медіа. Кожне посилання `<a>` має атрибут `href`, який вказує URL-адресу профілю та сторінки відповідної соціальної мережі. Використовується FontAwesome, де `<i>` еле-

мент має клас "fa-brands" для вказання, що це іконка відповідає соціальній мережі, і вказується клас конкретної соціальної мережі (наприклад, "fa-telegram", "fa-square-instagram", "fa-facebook").

Загалом, цей код відображає розклад роботи та посилання на соціальні медіа. Він може використовуватись для відображення інформації про графік роботи сайту.

2.4. Реалізація елементів JS у веб-додатку.

Продемонструємо елементи, використані при створенні кнопки анітсрес головної сторінки web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

```
const collorArr =[1,2,3,4,5,6,7,8,9,'a','b','c','d','e','f'];
const centerAntistres = document.getElementsByClassName('center__antistres');
centerAntistres[0].addEventListener('click',function() {
  let color='#';
  for (i=0; i < 6; i++){
    color +=collorArr[Math.round(Math.random() * ( collorArr.length-1))];
  }
  centerAntistres[0].style.background = color;
})
```

Рис. 2.28. Масив для створення випадкових кольорів

Цей код JavaScript виконує такі дії:

1. Він визначає масив під назвою `collorArr`, який містить послідовність чисел від 1 до 9 і букв від 'a' до 'f'. Цей масив пізніше використовується для створення випадкових кольорів.

2. Вибирає елемент із назвою класу `'center__antistres'` за допомогою методу `document.getElementsByClassName()`. Вибраний елемент призначається змінній `centerAntistres`.

3. Він додає слухач подій до першого елемента з назвою класу `'center__antistres'`. Подія, яка прослуховується, є подією клацання. Коли клацнути елемент, буде виконано код усередині функції.

4. Усередині функції слухача подій оголошується нова змінна `color` та ініціалізується значенням `'#'`. Ця змінна буде зберігати згенерований випадковий колір.

5. Цикл for використовується для створення випадкового кольору. Він виконує шість разів (від 0 до 5), щоб створити шістнадцятковий код кольору з шести символів.

6. У межах кожної ітерації циклу за допомогою `Math.random()` і `Math.round()` генерується випадковий індекс. Цей індекс використовується для отримання випадкового елемента з масиву `collorArr`.

7. Випадково вибраний елемент із `collorArr` об'єднується зі змінною `color`.

8. Нарешті, колір тла першого елемента в масиві `centerAntistres` встановлюється на згенерований колір за допомогою властивості `style.background`.

Таким чином, цей код вибирає елемент із назвою класу `'center__antistres'` і додає до нього прослуховувач подій клацання. Після клацання елемента генерується випадковий шестизначний шістнадцятковий код кольору, який застосовується як колір фону елемента.

Продемонструємо елементи, використані при створенні випадаючого меню головної сторінки web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

```
const garmohkaItem = document.querySelectorAll('.garmohka__item');

for(i=0;i<garmohkaItem.length;i++){
    garmohkaItem[i].addEventListener('click',showGarmohkaContent)
}
function showGarmohkaContent(){
    this.nextElementSibling.classList.toggle('show')
}
```

Рис. 2.29. Випадаюче меню

Цей код JavaScript виконує такі дії:

1. Вибирає всі елементи з іменем класу `'garmohka__item`` за допомогою методу `document.querySelectorAll()`. Вибрані елементи зберігаються у змінній `garmohkaItem` як NodeList.

2. Він встановлює цикл `for` для повторення кожного елемента в `garmohkaItem` NodeList. Цикл починається з індексу 0 і продовжується до довжини NodeList `garmohkaItem`.

3. Усередині циклу до кожного елемента додається слухач подій за допомогою методу addEventListener(). Подія, яка прослуховується, є подією клацання. Коли клацнути будь-який з елементів, буде виконано код у функції `showGarmohkaContent`.

4. Визначено функцію showGarmohkaContent. Він викликається, коли клацнути елемент `'garmohka__item'`. Функція отримує доступ до вибраного елемента за допомогою ключового слова `this`.

5. Всередині функції `showGarmohkaContent` властивість `nextElementSibling` використовується для доступу до наступного однорідного елемента клацнутого елемента. Передбачається, що це елемент, що містить вміст, який буде показано або приховано.

6. Метод `classList.toggle()` викликається в `nextElementSibling` для перемикання присутності класу `'show'`. Це означає, що якщо елемент не має класу `'show'`, його буде додано, а якщо він уже має клас, його буде видалено.

Підсумовуючи, цей код додає прослуховувач події клацання до кожного елемента з іменем класу `'garmohka__item'`. Коли клацнути будь-який із цих елементів, він перемикає видимість наступного однорідного елемента, додаючи або видаляючи клас `'show'`.

Продемонструємо елементи, використані при створенні спливаючих вікон web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

```
const button = document.querySelectorAll('.button');
const popup = document.querySelectorAll('.popup');

for(i=0;i<button.length;i++){
  button[i].addEventListener('click', toggleOtvet);
}
function toggleOtvet(){
  for(i=0;i<button.length;i++){
    button[i].nextElementSibling.classList.remove('active');
  }
  this.nextElementSibling.classList.add('active');
}
```

Рис. 2.30. Код який додає прослуховувач події клацання до кожного елемента з іменем класу "button"

Цей код JavaScript виконує такі дії:

1. Вибирає всі елементи з іменем класу `"button`` за допомогою методу `document.querySelectorAll()`. Вибрані елементи зберігаються у змінній `button` як NodeList.

2. Вибирає всі елементи з іменем класу `"popup`` за допомогою методу `document.querySelectorAll()`. Вибрані елементи зберігаються у змінній `popup` як NodeList.

3. Він встановлює цикл `for` для проходження кожного елемента в `button` NodeList. Цикл починається з індексу 0 і продовжується до довжини `button` NodeList.

4. Усередині циклу до кожного елемента додається слухач подій за допомогою методу addEventListener(). Подія, яка прослуховується, є подією клацання. Коли клацнути будь-який з елементів, буде виконано код у функції `toggleOtvet`.

5. Визначено функцію `toggleOtvet`. Він викликається, коли клацають елемент `"button"`. Функція отримує доступ до вибраного елемента за допомогою ключового слова `this`.

6. Усередині функції `toggleOtvet` налаштовано ще один цикл `for` для повторного повторення кожного елемента в `button` NodeList. Цей цикл використовується для видалення класу «активний» з наступних однорідних елементів усіх елементів.

7. У внутрішньому циклі властивість `nextElementSibling` використовується для доступу до наступного однорідного елемента кожного елемента ``button``. Потім клас `'active'` видаляється з кожного з цих елементів за допомогою методу `classList.remove()`.

8. Після видалення класу `'active'` з усіх наступних однорідних елементів, клас `'active'` додається до наступного однорідного елемента елемента `'button'`, за допомогою `classList.add()`. метод.

Таким чином, цей код додає прослуховувач події клацання до кожного елемента з іменем класу "button". Коли клацнути будь-який із цих елементів, він видаляє клас `'active'` з усіх наступних однорідних елементів `'button'` і додає клас `'active'` до наступного однорідного елемента `'button " елемент. Цей код використовується для перемикання видимості (спливаючих вікон), пов’язаних із певними кнопками.

Продемонструємо елементи, використані при створенні перевірки форми web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

```
jQuery(document).ready(function () {

   jQuery('form button').click( function() {
       var form = jQuery(this).closest('form');

       if ( form.valid() ) {
           form.css('opacity','.5');
           var actUrl = form.attr('action');

           jQuery.ajax({
               url: actUrl,
               type: 'post',
               dataType: 'html',
               data: form.serialize(),
               success: function(data) {
                   form.html(data);
                   form.css('opacity','1');
                   form.find('.status').html('форма відправлена успішно');

               },
               error:    function() {
                    form.find('.status').html('серверная помилка');
               }
           });
       }
   });


});
```

Рис. 2.31. Код який використовує jQuery для обробки події натискання елементів кнопки у формі

Цей код JavaScript використовує бібліотеку jQuery для виконання таких дій:

1. Він загортає код у функцію `jQuery(document).ready()`, яка забезпечує виконання коду лише після завершення завантаження DOM.

2. Вибирає елементи `button` у `form` за допомогою селектора ``'form button'``.

3. Він приєднує обробник події клацання до вибраних елементів `button` за допомогою методу `.click()`. Якщо натиснути будь-яку з цих кнопок, буде виконано код у наданій функції.

4. Усередині функції обробки подій клацання він знаходить найближчий елемент `form` до натиснутої кнопки за допомогою методу `.closest()` і зберігає його в змінній `form`.

5. Він перевіряє, чи дійсна форма, викликаючи метод `.valid()` для об'єкта `form`. Ймовірно, це стосується механізму перевірки форми, реалізованого в іншому місці.

6. Якщо форма дійсна, вона встановлює непрозорість форми на 0,5 за допомогою методу `.css()`.

7. Він отримує значення атрибута `action` з `form` за допомогою методу `.attr()` і призначає його змінній `actUrl`.

8. Він робить запит AJAX (асинхронний JavaScript і XML) за допомогою методу `jQuery.ajax()`.

9. Запит AJAX налаштовано з такими параметрами:

- `url` встановлюється на значення `actUrl`, отримане раніше.

- `type` встановлено на ``'post'``, що вказує на запит POST.

- `dataType` встановлено на ``'html'``, що вказує на очікуваний тип даних відповіді.

- `data` встановлюється на дані серіалізованої форми, отримані за допомогою методу `.serialize()`.

- `success` — це функція зворотного виклику, яка виконується, якщо запит AJAX виконано успішно. Він приймає `data`, повернуті сервером, як аргумент.

- У зворотному виклику `success` він замінює вміст `form` на `data`, отриманий із сервера за допомогою методу `.html()`.

- Встановлює непрозорість `форми` на 1 за допомогою методу `.css()`.

- Він встановлює вміст елементів із класом `.status` у `form` на 'форма надіслана успішно' (форму надіслано успішно).

- `error` - це функція зворотного виклику, яка виконується, якщо запит AJAX стикається з помилкою.

- Всередині зворотного виклику `error` він встановлює вміст елементів з класом `.status` у `form` на 'серверная помилка' (помилка сервера).

Підсумовуючи, цей код використовує jQuery для обробки події натискання елементів кнопки у формі. Після натискання кнопки виконується перевірка форми, і якщо форма дійсна, надсилається запит AJAX на сервер. Відповідь від сервера потім використовується для оновлення вмісту форми та відображення відповідного повідомлення про успіх або повідомлення про помилку.

2.5. Реалізація елементів PHP у веб-додатку.

Продемонструємо елементи, використані при створенні перевірки форми web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

```php
<?php
    $name = $_POST['name'];
    $email = $_POST['E-mail'];
    $text = $_POST['text'];

    $to = "roma.drapogyz@gmail.com";
    $date = date ("d.m.Y");
    $time = date ("h:i");
    $from = $email;
    $subject = "Заявка с сайта";


    $msg="
    Имя: $name /n
    Почта: $email /n
    Текст: $text";
    mail($to,$subject,"dfgdf",$headers);

?>

<p>Привіт, форма відправлена</p>
```

Рис. 2.32. Код PHP який отримує дані форми

Цей код PHP виконує такі дії:

1. Він отримує дані з форми HTML за допомогою суперглобальної змінної `$_POST`. Зокрема, він присвоює значення поля введення з назвою `"name"` змінній `$name`, значення поля введення з назвою `"E-mail"` змінній `$email`, і значення поля введення з іменем `"text"` до змінної `$text`.

2. Він встановлює адресу електронної пошти одержувача на `"roma.drapogyz@gmail.com`` шляхом призначення її змінній `$to`.

3. Він отримує поточну дату й час за допомогою функції `date()` і призначає їх змінним `$date` і `$time` відповідно.

4. Він присвоює значення змінної `$email` змінній `$from`, яка представляє адресу електронної пошти відправника.

5. Встановлює тему електронного листа `"Заявка с сайта``, призначаючи її змінній `$subject`.

6. Він створює тіло повідомлення електронної пошти шляхом об’єднання значень змінних `$name`, `$email` і `$text` у змінну `$msg`.

7. Для надсилання електронного листа використовується функція `mail()`. Функція приймає адресу електронної пошти одержувача (`$to`), тему електронного листа (`$subject`), тіло повідомлення (`$msg`) і додаткові додаткові заголовки (`$headers`) як аргументи. Однак у наданому коді змінна `$headers` не визначена.

8. Після надсилання листа виводиться HTML-код `<p>Привіт, форма надіслана</p>`.

Підсумовуючи, цей код PHP отримує дані форми, створює повідомлення електронної пошти, використовуючи дані форми, і надсилає електронний лист на вказану адресу одержувача. Після надсилання електронного листа відображається просте HTML-повідомлення, яке вказує на те, що форму успішно надіслано.

2.6. Використання функцій SCSS

Продемонструємо приклади використання таких функцій SCSS, як міксини, вкладення та частини в процесі розробки інтерфейсу користувача.

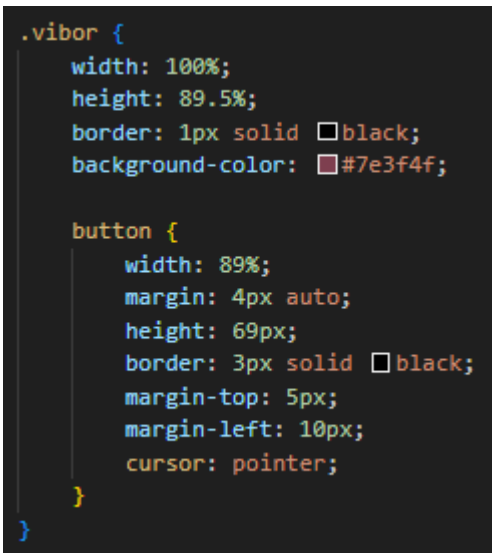

Рис. 2.33. Код SCSS який стилізує елементи з іменем класу

Цей код SCSS визначає стилі для класу під назвою `.vibor`. Ось розбивка коду:

1. Селектор `.vibor` вибирає елементи з назвою класу `.vibor`;

2. Властивість `width` встановлює ширину вибраних елементів на 100% від їх батьківського контейнера;

3. Властивість `height` встановлює висоту вибраних елементів на 89,5% їхнього батьківського контейнера;

4. Властивість `border` додає суцільну чорну рамку розміром 1 піксель до вибраних елементів;

5. Властивість `background-color` встановлює колір фону вибраних елементів на `#7e3f4f`;

6. Вкладений селектор `button` націлений на елементи `button` у класі `.vibor`;

7. Властивість `width` встановлює ширину вкладених елементів `button` на 89% від їх батьківського контейнера;

8. Властивість `margin` встановлює автоматичні горизонтальні поля та верхнє та нижнє поле розміром 4 пікселя для вкладених елементів `button`;

9. Властивість `height` встановлює висоту вкладених елементів `button` на 69 пікселів;

10. Властивість `border` додає суцільну чорну рамку розміром 3 пікселя до вкладених елементів `button`;

11. Властивість `margin-top` додає 5px верхнє поле до вкладених елементів `button`;

12. Властивість `margin-left` додає ліве поле розміром 10 пікселів до вкладених елементів `button`;

13. Властивість `cursor` встановлює стиль курсора миші на `pointer` для вкладених елементів `button`, вказуючи, що їх можна натиснути.

Підсумовуючи, цей код SCSS стилізує елементи з іменем класу `.vibor` і встановлює їх ширину, висоту, рамку та колір фону. Він також стилізує вкладені елементи `button` у клас `.vibor`, встановлюючи їх ширину, висоту, рамку, поля та стиль курсору.

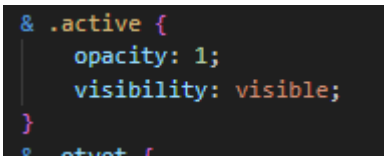

Рис. 2.34. Код SCSS який націлений на елементи з іменем класу active

Цей код SCSS визначає стилі для елементів із назвою класу `.active`, які також вкладені в інший батьківський елемент. Ось розбивка коду:

1. Символ «&» відноситься до батьківського селектора, який у цьому випадку представляє батьківський елемент, що містить клас «.active»;

2. Селектор `.active` вибирає елементи з назвою класу `.active`, які є нащадками батьківського елемента;

3. Властивість `opacity` встановлює непрозорість елементів `.active` на 1, роблячи їх повністю видимими;

4. Властивість `visibility` встановлює видимість елементів `.active` на `visible`, гарантуючи, що вони видимі, а не приховані.

Підсумовуючи, цей код SCSS націлений на елементи з іменем класу `.active`, вкладені в інший батьківський елемент. Він встановлює для них непрозорість на 1 і видимість на видимість, що робить їх повністю видимими на сторінці.

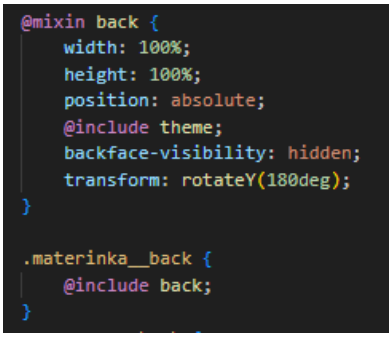

Рис. 2.35. Міксин та використання цього міксину

Цей код SCSS визначає міксин під назвою `back` і застосовує його до класу під назвою `.materinka__back`. Ось розбивка коду:

1. Директива `@mixin` визначає міксин з назвою `back`. Міксин — це багаторазовий блок коду CSS, який можна включити в кілька селекторів.

2. Усередині міксину `back` визначені такі властивості:

- Властивість `width` встановлює ширину елементів на 100% від їх батьківського контейнера.

- Властивість `height` встановлює висоту елементів на 100% від їх батьківського контейнера.

- Властивість `position` встановлює позицію елементів на `абсолютну`.

- `@include theme` включає інший міксин під назвою `theme` (не показано в наданому коді). Це дозволяє включати додаткові стилі, визначені в міксіні `theme`.

- Властивість `backface-visibility` встановлює для backface-visibility елементів значення `hidden`, запобігаючи видимості задньої поверхні в 3D-перетвореннях.

- Властивість `transform` застосовує тривимірне обертання до елементів, повертаючи їх навколо осі Y на 180 градусів.

3. Селектор `.materinka__back` націлює елементи з назвою класу `.materinka__back`.

4. Директива `@include` використовується для включення міксину `back` у селектор `.materinka__back`. Це застосовує всі стилі, визначені в міксині `back`, до елементів із класом `.materinka__back`.

Підсумовуючи, цей код SCSS визначає міксин під назвою `back` зі стилями для ширини, висоти, позиції, теми, видимості задньої поверхні та трансформації. Потім він застосовує цей міксин до класу `.materinka__back`, у результаті чого елементи з класом `.materinka__back` успадковують ці стилі.

2.7. Висновок до розділу 2

У процесі розробки веб-додатку "Підбір нових компонентів для ПК" було виконано ряд завдань, які сприяли успішній реалізації проекту. Починаючи з визначення структури та організації інтерфейсу, було розроблено каркаси та прототипи низької точності для визначення загального напрямку проекту. Ітеративне проектування та прототипування були використані для вдосконалення інтерфейсу та забезпечення зручності використання додатку. Для реалізації веб-додатку було використано Visual Studio Code, яке було налаштовано для зручної розробки. HTML використовувався для створення структури та розмітки веб-сторінок, в той час як JS використовувався для додавання динамічних функцій та взаємодії з користувачем. PHP було використано для реалізації бізнес-логіки та взаємодії з базою даних. Окрім того, було використано функції SCSS для поліпшення оформлення і забезпечення кращої організації стилів. Це дозволило зменшити повторюваний код і спростити процес розробки. Усі ці кроки допомогли створити функціональний веб-додаток "Підбір нових компонентів для ПК". В результаті було досягнуто мети проекту - забезпечити користувачам зручний і точний підбір комплектуючих для їх персонального комп'ютера.

РОЗДІЛ 3
ПРОЦЕС ДОСЛІДЖЕННЯ ТА РОЗРОБКИ, АГРЕГУВАННЯ ІНФОРМАЦІЇ

3.1. «Методологія вибору ідеально сумісністних компонентів ПК».

Ефективна навігація в лабіринтовому світі компонентів ПК вимагає ретельного підходу для людей, які прагнуть знайти тонкий баланс між продуктивністю, сумісністю та доступністю. З великою кількістю вибору, доступного на ринку, стає все більш важливим озброїтися необхідними знаннями та інструментами для прийняття обґрунтованих рішень.

Я обрав 3 методи для підбору ідеально сумісних компонентів ПК для полегшення розуміння людиною складу персональних комп'ютерів.

Ефективна навігація в заплутаному світі компонентів ПК і розуміння складу персональних комп’ютерів є завданням, яке часто спантеличує багатьох людей. Щоб спростити цей складний процес, було розроблено три різні методи, кожен з яких спрямований на те, щоб допомогти користувачам зрозуміти нюанси складання ПК і вибрати компоненти, які легко інтегруються один з одним. Ці методи дають користувачам необхідні знання для прийняття обґрунтованих рішень на основі різних критеріїв, включаючи ціну, сумісність і загальну інформацію про компоненти ПК.

Спосіб 1. Вибір збірки ПК з урахуванням ціни.

Перший метод має на меті надати користувачам легкодоступний варіант складання ПК, ретельно підібраний на основі фактора ціни.

Цей метод пропонує користувачам перевагу спрощеного процесу вибору, оскільки вони можуть швидко оцінити та порівняти ціни на різні попередньо налаштовані збірки ПК.

Зосереджуючись на ціні, люди можуть узгодити свої очікування зі своїми фінансовими можливостями, забезпечуючи збірку, яка відповідає їхнім вимогам, без надмірного навантаження на їхні ресурси.

Крім того, супровідний опис підкреслює обґрунтування вибору конкретних компонентів і пояснює, як їх інтеграція оптимізує загальну продуктивність ПК.

Спосіб 2. Детальний аналіз збору ПК, відштовхуючись від конкретного компоненту.

Другий метод покращує розуміння користувачами складання ПК, надаючи повне уявлення про компоненти які найкраще підходять до того компоненту який у вас вже є або відтовхуетесь саме від нього . Користувачі можуть досліджувати готову збірку ПК і заглиблюватися в деталі, які можуть містити інформацію про окремі частини або збірку в цілому. Ця детальна інформація дає користувачам можливість глибше зрозуміти вибрані компоненти та їх сумісність один з одним.

Детально досліджуючи компонент від якого ви відщтовхуетесь, користувачі можуть не тільки дізнатися про їхні особливості та можливості, але й оцінити їх сумісність із наявними компонентами, якими вони, можливо, вже володіють. Цей підхід виявляється особливо цінним для людей, які прагнуть оновити свої ПК, або тих, хто бажає перепрофілювати певні компоненти зі своїх поточних установок. Озброївшись цими знаннями, користувачі можуть приймати зважені рішення щодо сумісності компонентів і впевнено продовжувати збирання чи оновлення ПК.

Спосіб 3. Вичерпна інформація про компоненти.

Третій метод призначений для людей, які прагнуть глибше зрозуміти ті компоненти ПК які мають дуже багато моделей та сумісностей та їх можливі комбінації . Оскільки ринок пропонує безліч різновидів компонентів, таких як системи охолодження, блоки живлення та модулі оперативної пам’яті, стає вкрай важливим володіти загальними знаннями про ці компоненти та їх сумісність один з одним. Цей метод надає користувачам платформу для доступу до вичерпної інформації про різні компоненти, що дозволяє їм приймати зважені рішення під час збирання своїх ПК.

Включення цих методів у процес прийняття рішень дає змогу користувачам оптимізувати збірку ПК, зберігаючи при цьому сумісність і економічну ефективність.

На завершення можна сказати, що три представлені тут методи пропонують різні підходи до оптимізації процесу вибору компонентів ПК. Зосереджуючись на ціні, надаючи детальну інформацію про складання та пропонуючи вичерпну інформацію про компоненти, користувачі можуть з більшою впевненістю орієнтуватися в складності складання ПК.

3.2. Агрегування інформації: підбір оптимальних збірок із різноманітних джерел.

У прагненні зібрати повну компіляцію компонентів було проведено ретельне дослідження з різних авторитетних джерел. Ці збірки, які охоплюють багатство знань і досвіду, якими поділилися люди на численних платформах, були ретельно переглянуті та проаналізовані, щоб визначити найкращі та найцінніші ідеї.

Процес розпочався з широкого дослідження та вивчення різноманітних вебсайтів, де люди щедро ділилися своїм досвідом, пропонуючи на розгляд безліч збірок. Ці джерела слугували безцінним сховищем знань з перших вуст, надаючи унікальні погляди на різні теми. Від форумів ентузіастів до спеціалізованих блогів ці платформи пропонували скарбницю інформації, яка полегшувала цілісну оцінку збірок.

Щоб забезпечити найвищу якість і надійність відібраних збірок, був реалізований суворий процес перевірки. Кожну збірку було піддано ретельній перевірці, при цьому уважно враховували довіру та досвід авторів. Вивчаючи репутацію та послужний список джерел, було оцінено достовірність спільного досвіду, гарантуючи, що лише найбільш надійні та глибокі збірки були включені до остаточної компіляції.

Процес відбору найкращих збірок був кропіткою роботою, яка передбачала прискіпливу оцінку багатьох факторів. Відповідні критерії включали відповідність і

застосовність збірок для бажаної мети, глибину та широту наданої інформації. Завдяки проникливому оку та багатству накопичених знань було відібрано найцінніші збірки для включення в кураторську збірку.

Основна мета цього ретельного процесу курування полягала в тому, щоб надати користувачам вишуканий вибір збірок, які охоплювали б найрелевантнішу та надійнішу інформацію. Просіюючи величезну кількість джерел і виявляючи проникливість, мета полягала в тому, щоб представити вичерпну компіляцію, яка слугувала б надійним ресурсом для людей, які шукають добре обізнані погляди на колекції.

Важливо зазначити, що підібрані збірки покликані служити відправною точкою для подальших розвідок і досліджень.

Підсумовуючи, процес курування оптимальних збірок передбачав ретельне дослідження, аналіз і перевірку з різних джерел. Завдяки ретельному оцінюванню достовірності, актуальності та консенсусу було створено вдосконалену компіляцію, яка надає користувачам надійний ресурс для прийняття обґрунтованих рішень.

3.3. Дослідження та аналіз користувачів

Аудиторія, яка відвідує веб-сайти, присвячені вибору компонентів для ПК, різноманітна та охоплює цілу низку людей із різними інтересами та досвідом у галузі технологій. Ці веб-сайти приваблюють широкий спектр користувачів, включаючи ентузіастів ПК, геймерів, професіоналів та окремих людей, які хочуть створити або оновити свої ПК. Давайте дослідимо різні категорії відвідувачів цих сайтів та їхні характеристики:

1. Ентузіасти ПК. Ентузіасти ПК захоплюються технологіями та люблять бути в курсі останніх досягнень компонентів ПК. Вони володіють глибокими знаннями про специфікації апаратного забезпечення, контрольні показники продуктивності та галузеві тенденції. Ці люди часто мають досвід створення та налаштування ПК і часто відвідують сайти вибору компонентів, щоб дослідити та порівняти різні варіанти. Вони дуже залучені до спільноти ПК і активно беруть участь у форумах і обговореннях, щоб поділитися своїм досвідом і отримати пораду.

2. Геймери. Ігри є основною рушійною силою популярності компонентів ПК. Геймери відвідують ці сайти, щоб вибрати компоненти, які можуть покращити їхній ігровий досвід, забезпечуючи високу продуктивність, графічні можливості та сумісність з останніми іграми. Вони віддають пріоритет таким факторам, як графічні карти, процесори та оперативна пам'ять, які можуть задовольнити складні вимоги до ігор.

Геймерам потрібна детальна інформація про специфікації, контрольні показники та відгуки користувачів, щоб приймати зважені рішення при виборі компонентів для своїх ігрових установок.

3. Будівельники своїми руками. Багато користувачів відвідують ці веб-сайти, тому що вони воліють створювати власні ПК, а не купувати готові системи. Ці будівельники DIY мотивовані такими факторами, як індивідуальне налаштування, економічна ефективність і задоволення від збирання власних машин. Вони покладаються на сайти вибору компонентів, які направлятимуть їх через процес, надаючи інформацію про сумісність, пропоновані конфігурації та відгуки користувачів, щоб забезпечити успішну збірку.

4. Користувачі-початківці. Початківці користувачі з обмеженими технічними знаннями також відвідують ці сайти, коли їм потрібно оновити свій ПК або придбати нові компоненти. Вони покладаються на веб-сайти, щоб навчити їх основам компонентів ПК, пояснити технічні терміни та спростити процес вибору. Ці користувачі часто шукають зручний інтерфейс, чіткі пояснення та рекомендації, адаптовані до їхніх потреб.

Загалом відвідувачі сайтів вибору комплектуючих для ПК мають спільні переваги та очікування. Вони шукають вичерпну інформацію про продукт, включаючи детальні характеристики, ціни, наявність і відгуки користувачів. Вони цінують інтуїтивно зрозумілий інтерфейс користувача, який забезпечує легку навігацію та функції порівняння. Ці користувачі очікують точної та актуальної інформації, надійних рекомендацій і платформи, яка допоможе їм приймати зважені рішення на основі їхніх конкретних вимог і бюджетних обмежень.

Для веб-сайтів, орієнтованих на вибір компонентів ПК, важливо задовольняти різноманітні потреби своєї аудиторії. Надаючи точну та детальну інформацію, зручні інтерфейси, персоналізовані рекомендації та можливості для взаємодії з громадою, ці сайти можуть ефективно обслуговувати різні категорії користувачів і надавати їм можливість приймати обґрунтовані рішення при виборі компонентів ПК.

3.4. Створення персонажів і сценарії користувачів

А зараз створимо персонажів і сценарії користувачів.

Персона 1. Технічно підкований ентузіаст:

- Ім'я: Алекс Томпсон.
- Вік: 30.
- Посада: розробник програмного забезпечення.
- Довідкова інформація: Алекс — ентузіаст технологій із глибокою пристрастю до комп’ютерів і створення нестандартних установок. Вони мають глибокі знання про компоненти ПК і завжди в курсі останніх досягнень галузі. Алекс витрачає значну кількість часу на дослідження, порівняння та тестування різних компонентів для створення високопродуктивних систем. Вони активно беруть участь в онлайн-форумах і із задоволенням діляться своїм досвідом із спільнотою ПК.
- Цілі:
  - Щоб бути в курсі останніх компонентів ПК і вдосконалень;
  - Щоб створити потужну та передову систему ПК для ігор і роботи;
  - Взаємодіяти зі спільнотою ПК і вносити свої знання та досвід.
- Виклики:
  - Пошук надійних джерел інформації про сумісність компонентів, продуктивність і контрольні показники;
  - Переконайтеся, що вибрані компоненти відповідають конкретним вимогам і забезпечують бажаний рівень продуктивності;
  - Залишаючись у межах свого бюджету, водночас отримуючи високоякісні компоненти.

- Сценарій користувача:

- Алекс планує створити новий ігровий ПК і хоче вивчити новітні відеокарти, доступні на ринку. Вони відвідують веб-сайт вибору компонентів ПК і переходять до розділу відеокарти. Алекс фільтрує параметри на основі їх бюджету, бажаного рівня продуктивності та сумісності з існуючою системою. Вони ретельно аналізують технічні характеристики, порівнюють відгуки користувачів і вивчають контрольні показники продуктивності, щоб прийняти обґрунтоване рішення. Після вибору відповідної відеокарти Алекс переходить на сторінку оформлення замовлення, впевнений у своєму виборі.

Персона 2. Користувач-початківець:

- Ім'я: Сара Міллер.
- Вік: 25.
- Посада: асистент з маркетингу.
- Довідкова інформація: Сара має обмежені технічні знання, але хоче оновити наявний комп’ютер, щоб покращити його продуктивність для своїх робочих завдань. Вона не знайома з останніми компонентами ПК і вважає процес вибору непосильним. Сара шукає просту для розуміння платформу, яка допоможе їй у процесі вибору компонентів і надасть рекомендації на основі її потреб і бюджету.
- Цілі:

- Щоб оновити свій ПК, щоб підвищити продуктивність для виконання робочих завдань;

- Отримати базове розуміння компонентів ПК та їх сумісності;

- Знайти зручну платформу, яка спрощує процес вибору компонентів.

- Виклики:

- Розуміння технічного жаргону та специфікацій, пов’язаних із компонентами ПК;

- Забезпечення сумісності між вибраними компонентами та існуючою системою;

- Почуття впевненості у своєму виборі та уникнення типових пасток під час вибору компонентів.

- Сценарій користувача:

- Сара відвідує веб-сайт вибору компонентів ПК, щоб дослідити варіанти оновлення оперативної пам’яті свого ПК. Вона не впевнена щодо вимог сумісності та збільшення продуктивності, на яке вона може розраховувати. Вона переходить до розділу RAM і знаходить спрощене пояснення різних типів RAM та їх сумісності з різними системами. Веб-сайт пропонує інструмент, за допомогою якого вона може ввести поточні технічні характеристики ПК, і на основі цього рекомендує сумісні модулі оперативної пам’яті. Сара дотримується рекомендацій і починає купувати вибрану оперативну пам’ять, відчуваючи впевненість, що прийняла обґрунтоване рішення.

Ці персонажі та сценарії користувачів допомагають проілюструвати різноманітні потреби та поведінку користувачів, які відвідують веб-сайти вибору компонентів ПК. Розуміючи свої цілі, виклики та сценарії, дизайнери та розробники можуть адаптувати веб-додаток відповідно до конкретних вимог кожного персонажа, забезпечуючи персоналізований і орієнтований на користувача досвід.

3.5. Висновок до розділу 3

У процесі дослідження та розробки методології вибору ідеально сумісних компонентів для персонального комп'ютера, було проведено ряд кроків, які сприяли створенню оптимального та зручного додатку.

Агрегування інформації було важливим етапом, де з різних джерел зібрано та відібрано необхідну інформацію про компоненти ПК. Це дозволило створити базу даних з оптимальними збірками компонентів для різних користувачів.

Дослідження та аналіз користувачів були проведені з метою розуміння їх потреб, вимог та переваг. Створення персонажів та сценаріїв користувачів допомогло визначити типові вимоги та сценарії використання, що дозволило розробити веб-додаток з максимальною відповідністю їх потребам.

Загальний результат проекту полягав у створенні ефективної методології вибору сумісних компонентів ПК. Ця методологія дозволяє користувачам швидко та точно знайти оптимальні компоненти для свого персонального комп'ютера. Вона базується на зібраній інформації, дослідженні користувачів та їх потреб, що забезпечує високу релевантність та точність результатів.

В результаті розробки веб-додатку "Методологія вибору ідеально сумісніcних компонентів ПК", було досягнуто мети - надання користувачам інструменту для зручного та швидкого вибору компонентів ПК, що відповідають їх вимогам та специфікаціям.

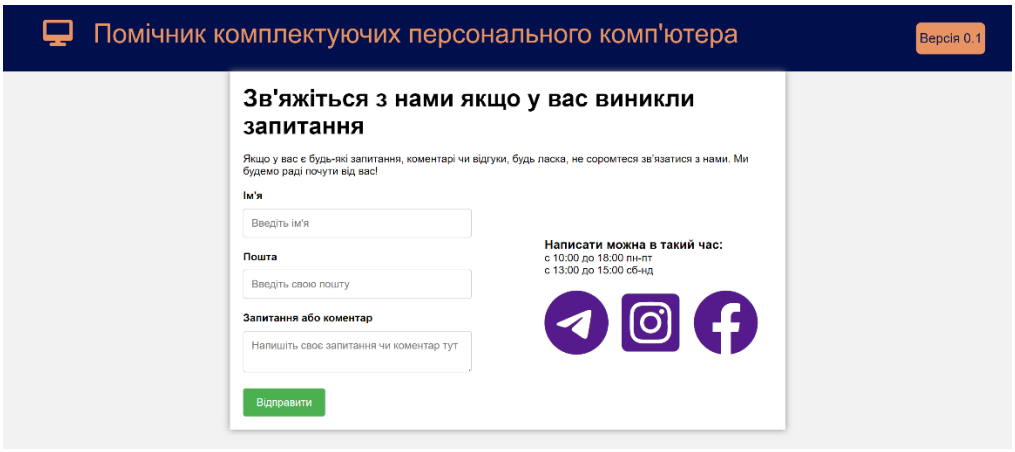

Рис. 3.1. Контакти

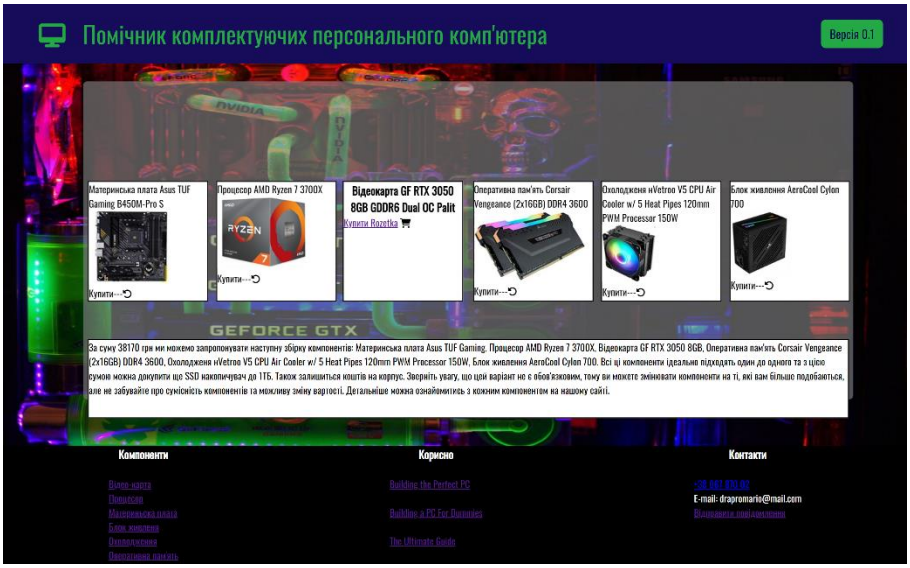

Рис. 3.2. Готовий варіант за ціною

ВИСНОВКИ

Підсумовуючи, провівши ретельний аналіз вимог, критеріїв оцінки та процесу відбору програмного забезпечення для проекту «Вибір нових компонентів для ПК», ми визначили набір інструментів, які найкраще відповідають заявленим цілям. Вибране програмне забезпечення включає HTML для гіпертекстової розмітки, JavaScript для сценаріїв, CSS (SCSS) для каскадних таблиць стилів, PHP для серверних сценаріїв, jQuery як бібліотеку JavaScript, BEM для методології модифікатора блокових елементів і FileZilla як FTP-клієнт . Процес створення веб-додатку включав визначення структури та організації інтерфейсу, створення каркасів або прототипів низької точності, а також використання ітераційного проектування та методів прототипування. Visual Studio Code було налаштовано як основне інтегроване середовище розробки, що полегшує розробку веб-програми «Вибір нових компонентів для ПК».

На додаток до цих технічних аспектів була розроблена методологія вибору ідеально сумісних компонентів ПК. Це передбачало збір інформації з різних джерел і підбір оптимальних колекцій, адаптованих до конкретних потреб користувачів. Було проведено дослідження та аналіз користувачів, щоб отримати уявлення про їхні переваги, поведінку та вимоги. Крім того, створення персонажів і користувацьких сценаріїв допомогло уявити взаємодію користувачів і оптимізувати процес проектування. Підсумовуючи, проект «Вибір нових компонентів для ПК» успішно задовольнив вимоги, застосувавши комплексний підхід до вибору програмного забезпечення, розробки веб-додатків і дизайну, орієнтованого на користувача. Вибрані програмні засоби, такі як HTML, JavaScript, CSS (SCSS), PHP, jQuery, BEM і FileZilla, зіграли вирішальну роль в успішній реалізації проекту. Ретельний аналіз вимог, критеріїв оцінки та дослідження користувачів забезпечили відповідність кінцевої веб-програми очікуванням цільової аудиторії, сприяючи легкому вибору сумісних компонентів ПК.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. P. Regan IT Essentials: PC Hardware and Software Labs and Study Guide / P. Regan; Cisco Press – Indianapolis : 2008. – 476 с. – Електрон. аналог друк. вид.: режим доступу: https://docplayer.net/1212653-It-essentials-pc-hardware-and-software-labs-and-study-guide-third-edition-patrick-regan-cisco-press.html (дата звернення 25.05.2023 р). – Назва з екрана.
2. Томсон Л. Web розробка PHP і MySQL / Томсон Л., Веллінг. Л.; Sams Publishing – United States of America : 2013. – 893 с. - Електрон. аналог друк. вид.: режим досту-пу: https://repository.unikom.ac.id/32751/1/php%20and%20mysql%20web%20dev.pdf (дата звернення: 10.05.2023)). – Назва з екрана.
3. J. Wiley HTML & CSS Design and Build Websites / J. Wiley; Cisco Press – Indianapolis : 2011. – 514 с. – Електрон. аналог друк. вид.: режим доступу: https://wtf.tw/ref/duckett.pdf (дата звернення 19.05.2023 р). – Назва з екрана.
4. R. Nixon Learning PHP, MySQL & JavaScript / R. Nixon; Cisco Press – United States of America : 2018. – 829 с. – Електрон. аналог друк. вид.: режим досту-пу: https://sd.blackball.lv/library/learning_php_mysql_and_javascript_(2018).pdf (дата звернення 20.05.2023 р). – Назва з екрана.
5. A. Freeman Pro jQuery / A. Freeman; Cisco Press – United States of America : 2012. – 989 с. – Електрон. аналог друк. вид.: режим доступу: https://sd.blackball.lv/library/Pro_jQuery.pdf (дата звернення 15.05.2023 р). – Назва з екрана.
6. Andrew S. Tanenbaum structured computer organization / T. Austin; Cisco Press – United States of America : 2013. – 801 с. – Електрон. аналог друк. вид.: режим доступу: https://csc-knu.github.io/sys-prog/books/Andrew%20S.%20Tanenbaum%20%20Structured%20Computer%20Organization.pdf (дата звернення 10.05.2023 р). – Назва з екрана.
7. Дакетт Дж. Web дизайн із набором HTML, CSS, JavaScript і jQuery / Дж. Да-кетт, [пер. з англ. М. А. Райтмана]; Видавництво «Е» - Київ : 2017 – 640 с. Елект-

рон. аналог друк. вид.: режим доступу: http://vk.academy.lv/file/ Dakett_Dzh_Javascript_i_jQuery_Interaktivnaya_web-razrabotka.pdf (дата звернення: 01.05.2023) – Назва з екрана.

8. Маркотт, І. Адаптивний web дизайн / І. Маркотт. - Нью Йорк : A Book Apart, 2014. - 150 с. Електрон. аналог друк. вид.: режим доступу: https://sd.blackball.lv/library/Pro_jQuery.pdf (дата звернення 20.05.2023 р). – Назва з екрана.