МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ **“БАКАЛАВР”**

Тема: «Вебсайт кав'ярні з використанням HTML, CSS та JS»

Виконавець: Владислав САШКІН

Керівник: к.т.н., доцент Олена ТОЛСТІКОВА

Нормоконтролер: к.т.н., доцент Вікторія СИДОРЕНКО

КИЇВ 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *комп'ютерних інформаційних технологій*

Спеціальність *122 «Комп'ютерні науки»*

Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
Аліна САВЧЕНКО
«_____»_____ 2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Сашкіна Владислава Руслановича

(прізвище, ім'я, по батькові здобувача вищої освіти в родовому відмінку)

1. Тема кваліфікаційної роботи: «Вебсайт кав'ярні з використанням HTML, CSS та JS»,

затверджена наказом ректора «05» квітня 2024 р. № 517/ст.

2. Термін виконання роботи: з 06 травня 2024 року по 16 червня 2024 року

3. Вихідні дані до роботи: вебсайт кав'ярні на основі засобів HTML, CSS та JS з використанням Visual Studio Code.

4. Зміст пояснювальної записки: 1. Огляд та аналіз предметної області. 2. Інструменти та технології розробки вебсайту. 3. Створення вебсайту кав'ярні.

5. Перелік обов'язкового ілюстративного матеріалу: 1. Структура вебсайту. 2. Верстка елементів вебсайту. 3. Меню напоїв. 4.Сторінка «Кошик». 5. Згенерований QR-код.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1	Ознайомлення з постановкою задачі та вивчення літератури. Розробка та затвердження плану кваліфікаційної роботи.	06.05.24-08.05.24	
2	Огляд та аналіз предметної області	08.05.24-12.05.24	
3	Вибір інструментів та технологій розробки	13.05.24-17.05.24	
4	Розробка вебсайту	18.05.24-23.05.24	
5	Оформлення та друк пояснювальної записки кваліфікаційної роботи.	24.05.24-30.05.24	
6	Проходження нормоконтролю, перепліт пояснювальної записки.	31.05.24	
7	Підготовка презентації та доповіді для виступу.	06.06.24-10.06.24	

7. Дата видачі завдання «06» травня 2024 р.

Керівник кваліфікаційної роботи _____ Олена ТОЛСТІКОВА
(підпис керівника)

Завдання прийняв до виконання _____ Владислав САШКІН
(підпис здобувача вищої освіти)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Вебсайт кав'ярні з використанням HTML, CSS та JS» містить: 51 сторінка, 26 рисунків, 13 інформаційних джерел.

Об’єкт дослідження – розробка вебсайту кав’ярні.

Предмет дослідження –вебсайт кав’ярні на основі засобів HTML, CSS та JS.

Мета кваліфікаційної роботи – розробка функціонального, зручного та привабливого вебсайту для кав'ярні з використанням технологій HTML, CSS та JavaScript, який забезпечить представлення асортименту продукції та підвищення рівня обслуговування.

Методи дослідження – аналіз літературних джерел, дослідження та аналіз уже існуючих сервісів.

Наукова новизна: У кваліфікаційній роботі представлено новий підхід до розробки вебсайту кав'ярні з використанням фреймворку Bootstrap. Використання Bootstrap дозволяє значно прискорити процес розробки, забезпечуючи при цьому високу якість та адаптивність вебсайту. Основна увага приділяється поєднанню простоти та функціональності, що дозволяє створити швидкий, легкий у використанні та візуально привабливий вебсайт. Це забезпечує легкість в обслуговуванні та розширенні вебсайту, а також можливість швидкого навчання та впровадження нових функцій.

Результати роботи: У результаті виконаної кваліфікаційної роботи було створено функціональний вебсайт кав'ярні з використанням HTML, CSS, JavaScript та фреймворку Bootstrap. Вебсайт містить головну сторінку з інформацією про кав'ярню, меню продукції з детальним описом та цінами, галерею зображень, форму зворотного зв'язку для клієнтів та розділ новин та акцій. Використання Bootstrap дозволило забезпечити адаптивний дизайн, який коректно відображається на різних пристроях та екранах. Вебсайт оптимізовано для швидкого завантаження та зручної навігації. Усі функціональні елементи протестовані та забезпечують належний рівень інтерактивності та зручності використання для користувачів.

Ключові слова: ВЕБСАЙТ, HTML, CSS, JAVASCRIPT, КАВ’ЯРНЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

HTML	–	HyperText Markup Language
CSS	–	Cascading Style Sheets
JS	–	JavaScriptx
IDE	–	Integrated Development Environment
WEB	–	World Wide Web
UI	–	User Interface
UX	–	User Experience
VS Code	–	Visual Studio

ВСТУП

Розвиток технологій та зростання доступності Інтернету кардинально змінили бізнес-середовище, зокрема в сфері обслуговування. Зараз наявність вебсайту для підприємства, незалежно від його розміру, є важливою складовою маркетингової стратегії. Вебсайт не лише забезпечує представлення інформації про послуги, але й дозволяє створити ефективну платформу для комунікації з клієнтами. Особливо актуальним це стає для невеликих закладів, таких як кав'ярні, де якісний вебсайт може стати вирішальним фактором у залученні та утриманні клієнтів.

Ця кваліфікаційна робота присвячена розробці вебсайту для кав'ярні з використанням сучасних вебтехнологій, таких як HTML, CSS та JavaScript. Основною метою є створення інтерактивного та естетично привабливого вебсайту, який не тільки представлятиме інформацію про заклад та його меню, але й забезпечуватиме зручність для користувачів.

У ході роботи буде детально проаналізовано кожен етап створення вебсайту, від планування структури та дизайну до впровадження інтерактивних функцій. Основна увага приділятиметься створенню естетично привабливого та функціонального інтерфейсу, що забезпечить зручність для користувачів та відповідатиме сучасним стандартам вебдизайну.

Кінцевим результатом проекту стане повнофункціональний вебсайт кав'ярні, який ефективно представлятиме інформацію про послуги закладу, покращуючи взаємодію з клієнтами.

Актуальність теми кваліфікаційної роботи полягає в необхідності адаптації бізнесів до сучасних цифрових реалій, де вебсайт відіграє ключову роль у взаємодії з клієнтами. З огляду на зростання конкуренції в сфері обслуговування, зокрема серед кав'ярень, наявність якісного вебсайту може стати важливим фактором для успіху бізнесу.

Об’єктом дослідження кваліфікаційної робити є процес створення вебсайту для кав'ярні.

Предмет дослідження – вебтехнології та методи їх застосування для розробки інтерактивного та функціонального вебсайту.

Мета кваліфікаційної роботи – розробити та впровадити сучасний, функціональний та естетично привабливий вебсайт для кав'ярні, що забезпечить ефективну комунікацію з клієнтами та підвищить рівень їх задоволеності послугами закладу.

Відповідно до поставленої мети роботи визначено основні **завдання дослідження**:

- Виконати аналіз предметної області;
- Дослідити сучасні тенденції та найкращі практики веброзробки кав’ярень;
- Провести огляд програмних засобів для розробки вебсайтів;
- Реалізувати вебсайт на основі зібраних даних;
- Впровадити інтерактивні функції для покращення користувацького досвіду;
- Провести аналіз отриманого результату;

Для досягнення поставленої мети й виконання завдань використано наступні методи: обробки літературних джерел, огляд сучасних вебтехнологій.

Практичне значення отриманих результатів. Результати цієї роботи можуть бути використані як основа для розробки вебсайтів для інших малих підприємств у сфері обслуговування та в освітніх цілях.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Основні поняття

Вебсайт - це цифрова платформа, що складається з взаємопов'язаних вебсторінок, розміщених на вебсервері. Ці вебсторінки призначені для доступу за допомогою гіперпосилань і мають єдиний інтерфейс та візуальний стиль. Крім того, вебсайт може містити додаткові ресурси, такі як зображення, відео та інші цифрові ресурси, що розширюють його зміст.

Компоненти вебсайту:

Вебсайт складається з декількох взаємопов'язаних компонентів, які разом формують його присутність в Інтернеті. Кожен компонент відіграє життєво важливу роль у функціональності, естетиці та зручності використання вебсайту.

1. Вебхостинг

Вебхостинг - це послуга, яка надає фізичне місце для зберігання файлів вебсайту та доступу до них в Інтернеті. Вебхостинг - це, по суті, сервер, який зберігає дані вебсайту і доставляє їх на пристрої користувачів за запитом.

Існують різні типи вебхостингу, зокрема віртуальний хостинг, виділений хостинг, віртуальні приватні сервери (VPS) і хмарний хостинг. Кожен тип пропонує різні рівні ресурсів, безпеки та масштабованості відповідно до вимог вебсайту.

2. Адреса вебсайту (URL)

Адреса вебсайту, також відома як Uniform Resource Locator (URL), - це унікальний ідентифікатор, який вказує на місцезнаходження вебсторінки або вебсайту в Інтернеті. Користувачі вводять URL-адресу у веббраузер, щоб отримати доступ до потрібного вебсайту.

URL складається з декількох компонентів, включаючи протокол (наприклад, HTTP або HTTPS), доменне ім'я (наприклад, www.example.com) і необов'язковий шлях або параметри, які вказують на конкретний ресурс на сервері.

3. Домашня сторінка

Домашня сторінка є основною точкою входу на вебсайт і слугує відправною точкою для користувачів. Зазвичай вона відображає загальну тему, брендинг і навігаційну структуру вебсайту. На домашній сторінці часто розміщується основна інформація, тематичний контент і посилання на інші розділи вебсайту.

Розробка ефективної домашньої сторінки вимагає ретельного врахування принципів користувацького досвіду (UX), таких як чітка навігація, помітні заклики до дії та привабливі візуальні ефекти, що привертають увагу відвідувачів.

4. Дизайн вебсайту

Дизайн вебсайту охоплює візуальні та функціональні аспекти вебсайту, включаючи макет, кольорові схеми, типографіку та графічні елементи. Він спрямований на створення привабливого та зручного інтерфейсу, який відповідає меті вебсайту та ідентичності бренду.

Ключові елементи дизайну вебсайту включають адаптивні макети (сумісні з різними пристроями), інтуїтивно зрозуміле навігаційне меню, доступний контент для підвищення задоволеності та залученості користувачів.

5. Контент

Контент - це інформація, представлена на вебсторінках, включаючи текст, зображення, відео та інтерактивні елементи. Високоякісний контент має важливе значення для передачі повідомлення вебсайту, залучення відвідувачів та покращення видимості в пошукових системах.

Ефективне створення контенту передбачає розуміння цільової аудиторії, створення переконливої розповіді, оптимізацію для SEO (пошукової оптимізації) та підтримання узгодженості на всьому вебсайті.

6. Структура навігації

Структура навігації визначає організацію та ієрархію сторінок вебсайту, полегшуючи навігацію для користувачів. Вона складається з меню, посилань та ієрархічних зв'язків, які спрямовують відвідувачів до різних розділів вебсайту.

Добре продумана навігація підвищує зручність використання, дозволяючи користувачам швидко та інтуїтивно знаходити інформацію. Поширені моделі навігації включають верхнє меню, навігацію по бічній панелі і посилання в нижньому колонтитулі.

Типи вебсайтів:

Статичні вебсайти:

Статичний вебсайт складається з вебсторінок, які доставляються користувачеві в тому вигляді, в якому вони були збережені, без жодної обробки на стороні сервера для створення контенту. Ці вебсторінки зазвичай створюються з використанням базових технологій, таких як HTML, CSS і JavaScript. Ключовою характеристикою статичного вебсайту є те, що кожна сторінка є окремим документом і не змінюється залежно від дій користувача або введення даних. Сервер просто повертає вже існуючі файли в браузер користувача за запитом.

Переваги статичних вебсайтів:

Швидкість: статичні вебсайти працюють швидко, оскільки не потребують обробки на сервері для динамічної генерації контенту.

Економічна ефективність: Хостинг статичного вебсайту зазвичай обходиться дешевше, оскільки він не потребує підтримки різних мов програмування на стороні сервера.

Простота: Їх легко розробляти і підтримувати, особливо для невеликих вебсайтів, які не потребують частого оновлення контенту або складних функціональних можливостей.

Безпека: Статичні вебсайти менш вразливі до певних типів кібератак, оскільки їм не вистачає можливостей обробки на стороні сервера.

Недоліки статичних вебсайтів:

Обмежена інтерактивність: Вони не можуть запропонувати динамічний контент або користувацький досвід без додаткових технологій.

Проблеми з обслуговуванням: Оновлення контенту на декількох сторінках може бути громіздким, оскільки зміни потрібно вносити вручну в кожен HTML-файл.

Динамічні вебсайти:

Динамічний вебсайт генерує вебсторінки динамічно під час виконання, пристосовуючи їх до запитів кожного користувача. На відміну від статичних вебсайтів, динамічні вебсайти обробляються на стороні сервера за допомогою скриптових мов, таких як PHP, Node.js або ASP.NET. Ці мови дозволяють серверу генерувати контент у відповідь на дії користувачів або запити до бази даних.

Переваги динамічних вебсайтів:

Інтерактивність: Динамічні вебсайти можуть реагувати на введення даних користувачем, що дозволяє використовувати такі функції, як форми, системи входу та персоналізований контент.

Інтеграція з базами даних: Вони можуть взаємодіяти з базами даних для зберігання та отримання інформації, що робить їх придатними для таких додатків, як сайти електронної комерції або соціальні мережі.

Управління контентом: Оновлення та модифікації можна вносити централізовано за допомогою серверних скриптів, автоматично поширюючи зміни на весь вебсайт.

Масштабованість: Динамічні вебсайти можуть обробляти складні функціональні можливості та легше адаптуватися до вимог, що змінюються, ніж статичні вебсайти.

Недоліки динамічних вебсайтів:

Повільніша продуктивність: Динамічна генерація контенту вимагає обробки на стороні сервера, що може призвести до затримок порівняно з обслуговуванням попередньо створених статичних файлів.

Складність: Розробка та підтримка динамічних вебсайтів часто включає в себе більш складні концепції та технології програмування.

Безпека: Динамічні вебсайти можуть бути більш вразливими до певних вразливостей безпеки через їхню інтерактивну та керовану даними природу.

1.2. Порівняння сайтів аналогів

1.2.1. Кав’ярня «Вірменка»

Вебсайт кав'ярні "Вірменка" (Рис. 1.1) є першим результатом за запитом «Кав’ярня» у пошуковій системі Google. Сайт вирізняється приємним мінімалістичним дизайном, що забезпечує зручність користування. Однак, існують певні недоліки, які можуть вплинути на загальне враження від ресурсу.

Одним із основних недоліків є відсутність зображень кавових напоїв, що ускладнює користувачам вибір. Крім того, деякі позиції меню мають неповні описи, що не дозволяє клієнтам отримати повну інформацію про склад та особливості напоїв. Включення фотографій напоїв та більш детальних описів їх складу могло б значно покращити навігацію по сайту і зробити його більш привабливим для відвідувачів.

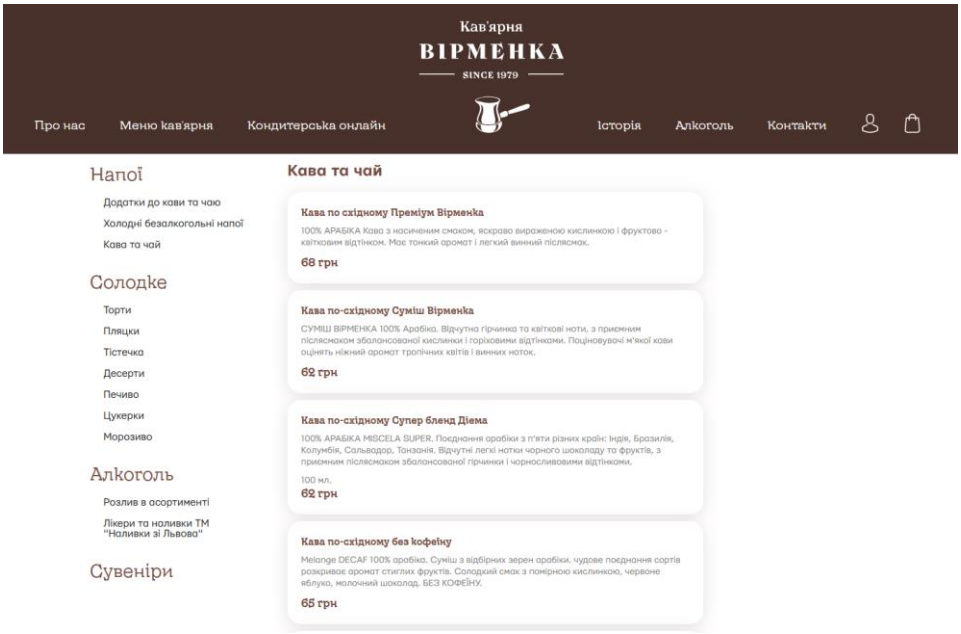

Рис. 1.1. Кав’ярня «Вірменка»

1.2.2. Кав’ярня «Каффа»

Сайт «Каффа» (Рис. 1.2) є другою кав’ярнею, що з’являється у результатах пошуку Google. Одразу привертає увагу перевантаженість інтерфейсу, який доцільно було б розділити на декілька окремих сторінок для зручності

користування. Деякі розділи, наприклад, текст на фоні піщаних дюн, мають недостатній контраст, що ускладнює читання.

У меню відсутні описи напоїв та фотографії, представлено лише назви та ціни. Це може створювати труднощі для користувачів, які бажають дізнатися більше про склад або вигляд напою перед замовленням. Для покращення користувацького досвіду, варто додати детальні описи кожного напою та зображення, що сприятиме інформованості клієнтів.

Також слід зазначити, що структура сайту не є інтуїтивно зрозумілою, що може впливати на навігацію і загальне сприйняття ресурсу.

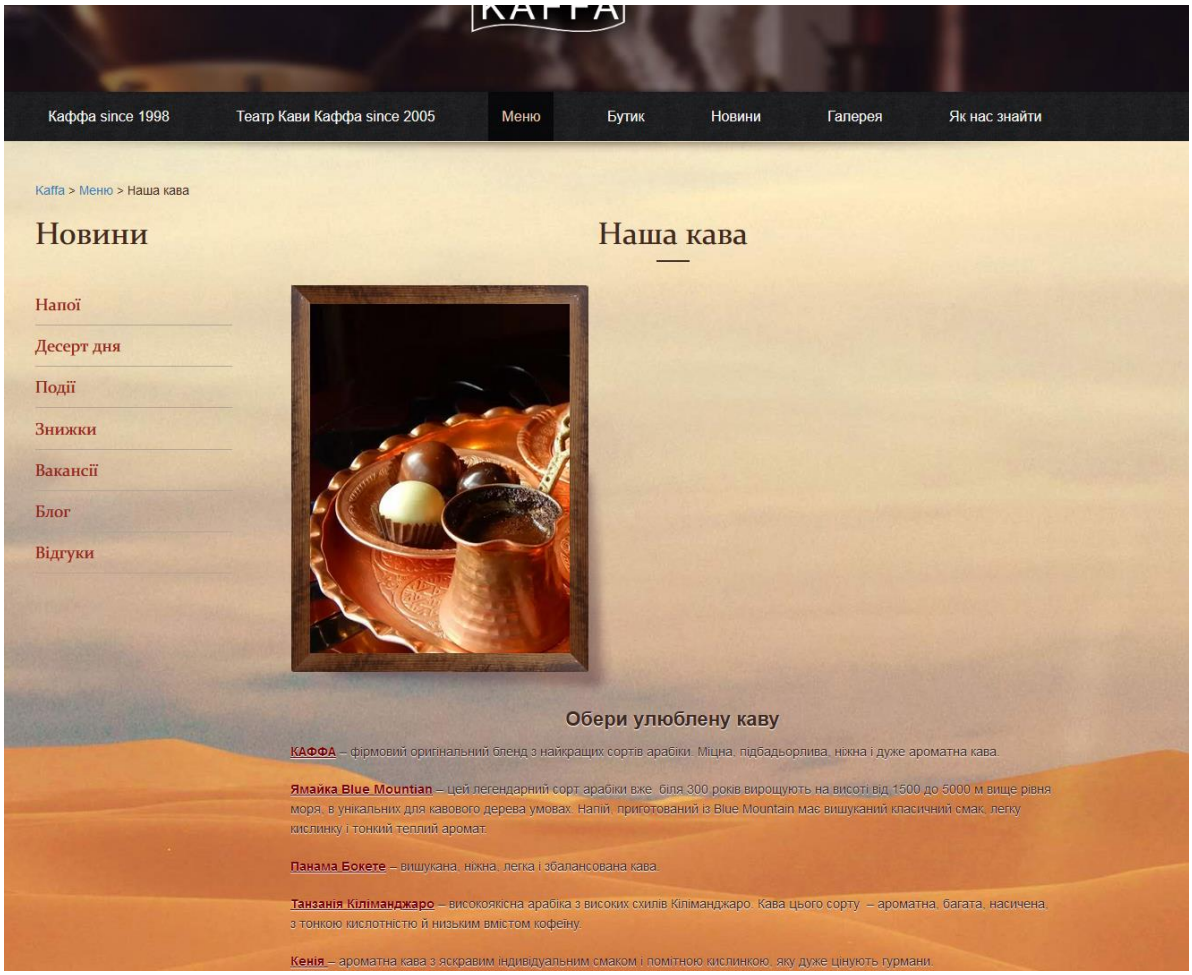

Рис. 1.2. Кав’ярня «Каффа»

1.2.3. Кав’ярня «Львівська копальня кави»

Прикладом сайту кав’ярні є «Львівська копальня кави» (Рис. 1.3). Дизайн сайту виконано у мінімалістичному стилі, з використанням добре підібраної кольорової палітри. Біля кожного напою розміщена фотографія з його зображенням, а також надано опис, що включає склад, спосіб приготування та

смакові характеристики. Користувачі мають можливість додавати напої до списку улюблених.

До недоліків сайту можна віднести відсутність стандартизованого опису напоїв з точним складом. Крім того, на сайті відсутня функція замовлення напою, що ставить під сумнів доцільність функції додавання до улюблених. Для покращення користувацького досвіду було б доцільно інтегрувати систему онлайн-замовлень, що дозволило б клієнтам легко замовляти напої через сайт.

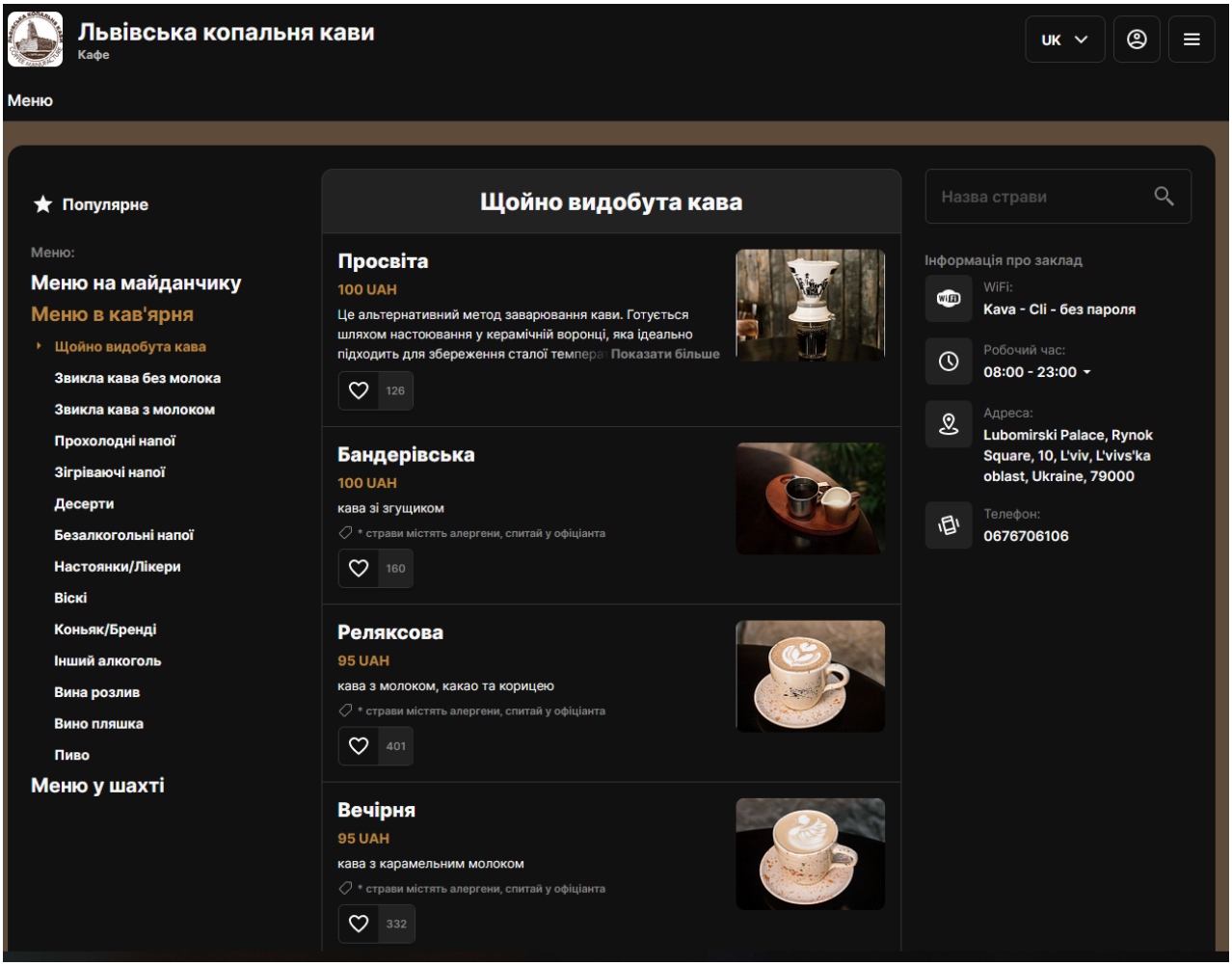

Рис. 1.3. Кав’ярня «Львівська копальня кави»

1.3. Постановка задачі

Постановка задачі полягає в тому, щоб створити мінімалістичний та зручний для користувачів вебсайт для кав’ярні, який одразу надавав би всю необхідну інформацію про різні види кави і не був перевантаженим зайвою інформацією. На кожній сторінці з описом кави повинна бути представлена детальна інформація про напій, включаючи склад, метод приготування, а також

якісне зображення, що допоможе клієнтам швидко зорієнтуватися і вибрати бажаний напій.

Окрім цього, важливо, щоб користувачі мали можливість додавати обрані види кави до списку улюблених напоїв. Це дозволить їм з легкістю знаходити свої улюблені напої під час кожного відвідування сайту, без необхідності повторного пошуку. Також потрібно реалізувати функцію додавання кави до кошика, що дозволить користувачам швидко і просто оформити замовлення. Функціонал кошика повинен включати можливість перегляду всіх доданих напоїв, редагування кількості або видалення позицій, а також остаточного підтвердження замовлення для його покупки.

Головна мета полягає у створенні простого, елегантного та інтуїтивно зрозумілого інтерфейсу, який забезпечить приємний користувацький досвід. Вебсайт повинен бути оптимізований для швидкого доступу до інформації, спрощуючи процес вибору і замовлення кавових напоїв. Це сприятиме задоволенню потреб клієнтів, роблячи процес взаємодії з сайтом легким та приємним. Дизайн повинен бути сучасним, привабливим і відповідати стилю кав’ярні, щоб підкреслити її унікальність та створити позитивне враження у відвідувачів.

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі було визначено основні поняття вебсайту, його ключові компоненти та типи. Провівши аналіз аналогічних вебсайтів, виділено їх переваги та недоліки. На основі цього аналізу стало очевидним, що вебсайт повинен відповідати мінімалістичному стилю та забезпечувати інтуїтивно зрозумілий інтерфейс.

В ході аналізу було також визначено, що для досягнення високого рівня користувацького досвіду важливо сконцентруватися на наданні всієї необхідної інформації у доступній формі. Це включає зображення напоїв, їх детальний склад, а також можливість зручного замовлення напоїв та додавання їх до списку улюблених. Вебсайт повинен бути зразком ефективного та зручного інструменту для користувачів, які шукають конкретну інформацію про напої та хочуть швидко зробити замовлення.

РОЗДІЛ 2
ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБСАЙТУ

2.1. Мова розмітки HTML

HTML, абревіатура від Hypertext Markup Language (мова розмітки гіпертексту), слугує основою для створення структурованого та відформатованого контенту на вебсторінках. Вона використовує систему тегів розмітки для визначення різних елементів, таких як текст, зображення, посилання та мультимедійні компоненти. Розуміння HTML передбачає розуміння його текстової природи, структури на основі тегів, ієрархії документів та інтеграції з іншими вебтехнологіями.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="/css/style.css" />
    <link rel="stylesheet" href="/css/navbar_animation.css" />
    <title>Home</title>
  </head>
  <body>
    <!--Navbar-->
    <header>
      <nav class="navbar fixed-top navbar-expand-lg navbar-dark p-md-3">
        <div class="container">
          <a href="#carouselExampleCaptions" class="navbar-brand">Вовки</a>
          <button type="button" class="navbar-toggler">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarNav">
            <div class="mx-auto"></div>
            <ul class="navbar-nav">
              <li>
                <a href="/index.html" class="nav-link text-white">Головна</a>
              </li>
```

Рис. 2.1. Приклад коду HTML

Ключові поняття HTML

1. Текстовий формат документів:

Файли HTML - це звичайні текстові документи, що зберігаються з розширеннями .html або .htm. Ці файли складаються з тексту, доповненого тегами HTML, які визначають структуру і зміст вебсторінки.

2. Теги та елементи:

HTML використовує теги розмітки, укладені в кутові дужки (< >) для визначення елементів. Теги часто зустрічаються парами, з відкриваючим тегом (<tag>) і закриваючим тегом (</tag>). Наприклад, позначає початок абзацу, а - його кінець.

3. Структура документа:

HTML-документи мають певну структуру:

- Оголошення <!DOCTYPE> визначає версію HTML.
- Елемент <html> інкапсулює весь документ, що містить секції <head> і <body>.
- Секція <head> містить метадані (наприклад, заголовок, кодування символів, посилання на таблиці стилів/скрипти).
- <body> містить видимий вміст вебсторінки.

4. Ієрархічний склад:

HTML підтримує вкладення тегів для створення ієрархічної структури документа. Наприклад, елемент (розділ) може охоплювати кілька елементів (абзац), кожен з яких містить вбудовані теги, такі як <a> (прив’язка) або (сильний наголос).

5. Атрибути:

Теги можуть містити атрибути, що пропонують додаткову інформацію. Наприклад:

- Тег <img> використовує атрибут src для джерела зображення.
- Тег <a> використовує атрибут href для призначення гіперпосилання.

6. Процес візуалізації:

Коли веббраузер зустрічає документ HTML, він аналізує вміст, інтерпретуючи теги та атрибути. Цей процес визначає спосіб відображення вмісту на екрані користувача, включаючи стилі, визначені CSS (каскадні таблиці стилів).

1. Гіперпосилання:

HTML містить теги <a> (прив’язки) для створення гіперпосилань, що дозволяє користувачам переходити між вебсторінками або отримувати доступ до зовнішніх ресурсів.

2. Вбудовування мультимедіа:

HTML полегшує вбудовування мультимедійних елементів за допомогою таких тегів, як <img> (зображення), <video> (відео) і <audio> (аудіо).

3. Створення форми:

HTML пропонує такі елементи форми, як <form>, <input>, <textarea> і <button> для створення інтерактивних форм для введення та надсилання користувачами.

4. Спеціальні можливості:

HTML містить такі функції доступності, як альтернативний текст для зображень і семантичні елементи (<nav>, <header>), щоб покращити зручність використання для людей з обмеженими можливостями.

5. Кросбраузерність:

HTML універсально підтримується сучасними веббраузерами, забезпечуючи послідовний перегляд на різних пристроях і платформах.

2.2. CSS (Cascading Style Sheets)

Каскадні таблиці стилів (CSS) - це фундаментальний компонент веброзробки, який використовується для визначення візуального представлення вебдокументів. CSS використовується для визначення макета, дизайну та естетичних властивостей документів HTML (HyperText Markup Language), XML (eXtensible Markup Language) та SVG (Scalable Vector Graphics). За допомогою CSS веброзробники можуть точно контролювати зовнішній вигляд контенту на різних пристроях і платформах.

```css
.navbar-nav .nav-link {
  position: relative;
  color: white;
  text-decoration: none;
  padding-bottom: 5px;
}

.navbar-nav .nav-link::after {
  content: "";
  position: absolute;
  width: 0;
  height: 2px;
  bottom: 0;
  left: 50%;
  transform: translateX(-50%);
  background-color: #fff;
  transition: width 0.3s ease;
}

.navbar-nav .nav-link:hover::after {
  width: 100%;
}

@media (max-width: 991px) {
  .navbar-nav .nav-link::after {
    display: none;
  }
}
```

Рис. 2.2. Приклад коду CSS

Призначення та функціональність

Основна мета CSS - відокремити вміст від представлення, дозволяючи розробникам ефективно стилізувати HTML-документи. Застосовуючи правила CSS до елементів HTML, розробники можуть визначати такі атрибути, як колір, розмір, макет і позиціонування, забезпечуючи послідовний і привабливий користувацький досвід у різних веббраузерах.

Синтаксис і структура

Правила CSS складаються з селекторів та декларацій. Селектори вказують на певні елементи HTML, тоді як декларації визначають властивості стилів, що застосовуються до цих елементів. Декларація складається з властивості (наприклад, color, font-size) і відповідного значення (наприклад, blue, 12px). Кілька декларацій можна згрупувати в межах правила для спільного застосування стилю до елемента.

Селектори

Селектори в CSS визначають, на які елементи в HTML-документі буде впливати заданий набір правил стилів. Вони можуть вибирати елементи на основі

їх типу (наприклад, <p> для абзаців), класу (наприклад, .highlight для елементів з класом "highlight"), ідентифікатора (наприклад, #header для елементів з ідентифікатором "header"), атрибутів або ієрархічних зв'язків (наприклад, батьківський-дочірній).

Правила стилізації

CSS-стилістика охоплює широкий спектр властивостей, які визначають зовнішній вигляд і поведінку HTML-елементів. До загальних властивостей відносяться:

- Колір і фон: Визначає колір тексту (color), колір фону (background-color) та зображення (background-image).
- Типографіка: Керує властивостями шрифту, такими як розмір (font-size), стиль (font-style), вага (font-weight) і сімейство (font-family).
- Макет і модель коробки: Визначає розміри (width, height), поля (margin), відступи (padding) і межі (border).
- Позиціонування: Позиціонує елементи за допомогою таких властивостей, як position, top, left, right і bottom.
- Гнучкі рамки та сітка: Пропонує розширені методи верстки за допомогою таких властивостей, як display, flex-direction, justify-content, align-items тощо.

Методи стилізації

CSS дозволяє використовувати різні вдосконалені методи стилізації, в тому числі:

- Медіа-запити: Адаптує макети на основі характеристик пристрою (наприклад, розміру екрану, орієнтації) за допомогою правил @media.
- Анімації та переходи: Додає ефекти руху за допомогою @keyframes, властивостей переходів та анімації.
- Псевдокласи та псевдоелементи: Націлює елементи на основі стану (:hover, :focus) або генерує віртуальні елементи (::before, ::after).

Таблиці стилів

Правила CSS зазвичай зберігаються у зовнішніх таблицях стилів (файлах .css), що полегшує підтримку та забезпечує узгодженість на різних вебсторінках.

Крім того, CSS можна вбудовувати безпосередньо в HTML за допомогою тегів <style> або застосовувати вбудовано до окремих елементів HTML.

Інтеграція з HTML

Щоб застосувати стилі CSS до вмісту HTML, розробники посилають зовнішні таблиці стилів в HTML за допомогою тегів <link> або вбудовують CSS безпосередньо в теги <style> в розділі <head> документа. Селектори, визначені в CSS, вказують, які елементи HTML повинні відповідати певним правилам стилів.

2.2.1. Bootstrap

Популярність Bootstrap у сфері веброзробки свідчить про його широкий інструментарій, ретельно спроектований, щоб надати розробникам багатий набір готових компонентів та утиліт. Відомий своєю безшовною інтеграцією в процеси розробки, Bootstrap пропонує величезну кількість інструментів, створених для того, аби прискорити перетворення дизайнерських концепцій на повнофункціональні вебелементи, і все це без втрати якості.

Центральним елементом привабливості Bootstrap є його адаптивна система сітки, ключовий компонент сучасного вебдизайну, який полегшує створення гнучких макетів для пристроїв з різним розміром екрану. Ця система сітки в поєднанні з широким набором варіантів типографіки надає розробникам гнучкість у створенні привабливих і зручних інтерфейсів, які легко адаптуються до постійно змінюваного ринку інтернет-споживання.

Крім того, в арсеналі Bootstrap є безліч CSS-компонентів, кожен з яких ретельно розроблений і ретельно задокументований..

Від кнопок і форм до навігаційних панелей і каруселей, ці компоненти слугують будівельними блоками для створення користувацького досвіду.

Також модульна природа Bootstrap заохочує розробників до експериментів, дозволяючи їм поєднувати та підбирати компоненти відповідно до конкретних вимог проекту.

Окрім основних можливостей, потужна екосистема плагінів та розширень Bootstrap ще більше посилює його привабливість, надаючи розробникам унікальну гнучкість у створенні вебсайтів відповідно до унікальних вимог та вподобань. Чи то інтеграція динамічних функцій за допомогою плагінів

JavaScript, чи то тонка настройка візуальної стилістики за допомогою спеціальних стилів CSS, Bootstrap дозволяє розробникам розкрити свій творчий потенціал і втілити цифрове бачення в життя з неймовірною точністю.

Більше того, широке розповсюдження Bootstrap у спільноті веброзробників сприяє створенню активної спільноти з можливостями для спільної роботи та обміну знаннями. Використання ресурсів спільноти дозволяє розробникам легше долати виклики, вирішувати проблеми та знаходити інноваційні рішення з меншими витратами та більшою ефективністю. Крім того, обширна документація, навчальні посібники та онлайн-ресурси, пов'язані з Bootstrap, слугують незамінними помічниками на шляху опанування, надаючи розробникам знання та досвід, необхідні для розкриття всього потенціалу технології.

2.3. Мова програмування JavaScript

JavaScript - це широко використовувана мова програмування, відома своєю універсальністю в різних сферах, включаючи веброзробку, серверні додатки, розробку мобільних додатків, пристроїв IoT (Інтернет речей) тощо. Вона стала популярним вибором як для початківців, так і для досвідчених розробників завдяки своїй гнучкості та широкому застосуванню.

```js
const carousel = document.getElementById("carouselExampleCaptions");
const carouselItems = carousel.querySelectorAll(".carousel-item");
const prevButton = carousel.querySelector(".carousel-control-prev");
const nextButton = carousel.querySelector(".carousel-control-next");
const indicators = carousel.querySelectorAll(".carousel-indicators button");

let intervalId;
let lastInteractionTime = Date.now();

// Функція для перемикання каруселі до вказаного айтему
function goToSlide(slideIndex) {
  // Видаляємо активний клас у всіх айтемів і кнопок індикаторів
  carouselItems.forEach((item) => item.classList.remove("active"));
  indicators.forEach((indicator) => indicator.classList.remove("active"));

  // Додаємо активний клас до зазначеного айтему і кнопки індикатора
  carouselItems[slideIndex].classList.add("active");
  indicators[slideIndex].classList.add("active");
}

// Функція для перемикання каруселі до наступного айтему
function nextSlide() {
  const currentSlideIndex = Array.from(carouselItems).findIndex((item) =>
    item.classList.contains("active")
  );
  const nextSlideIndex = (currentSlideIndex + 1) % carouselItems.length;
  goToSlide(nextSlideIndex);
}

// Функція для перемикання каруселі до попереднього айтему
function prevSlide() {
  const currentSlideIndex = Array.from(carouselItems).findIndex((item) =>
    item.classList.contains("active")
  );
  const prevSlideIndex =
    (currentSlideIndex - 1 + carouselItems.length) % carouselItems.length;
  goToSlide(prevSlideIndex);
```

Рис. 2.3. Приклад коду JS

Ключові особливості JavaScript:

Мова високого рівня:

JavaScript - це мова високого рівня, яка абстрагується від низькорівневих деталей базового обладнання. Вона має автоматичне керування пам'яттю за допомогою збирача сміття, що звільняє розробників від завдань ручного розподілу пам'яті, характерних для таких мов, як C. JavaScript пропонує надійні конструкції для роботи зі складними структурами даних та об'єктами.

Динамічне виконання:

Будучи динамічним, JavaScript виконує багато операцій під час виконання, які статичні мови виконують під час компіляції. Ця динамічна природа надає такі переваги, як динамічна типізація, пізнє зв'язування, рефлексія та підтримка функціональних парадигм програмування. Такі концепції, як зміна об'єктів під час виконання, закриття та більш просунуті мовні можливості, притаманні JavaScript.

Динамічна типізація:

JavaScript має динамічну типізацію, що дозволяє змінним зберігати значення будь-якого типу. Це означає, що змінну можна перепризначити зі значенням іншого типу під час виконання. Крім того, JavaScript має вільну типізацію, що означає, що вона не застосовує суворі правила до об'єктів, забезпечуючи гнучкість, але жертвуючи певною безпекою типів у порівнянні з такими мовами, як TypeScript.

Інтерпретована мова:

JavaScript зазвичай називають інтерпретованою мовою, що означає, що вона не потребує етапу компіляції перед виконанням. Хоча сучасні движки JavaScript використовують компіляцію JIT (Just-In-Time) для оптимізації продуктивності, розробники взаємодіють з JavaScript як з інтерпретованою мовою без необхідності явної компіляції.

Багатопарадигмальність:

JavaScript підтримує декілька парадигм програмування, включаючи об'єктно-орієнтоване програмування (ООП) з використанням прототипів і класів ES6, функціональне програмування з першокласними функціями та імперативне

програмування, подібне до мов на кшталт C. На відміну від мов, які нав'язують певні парадигми, JavaScript дозволяє розробникам обирати найбільш підходящий підхід для своїх додатків.

Різноманітні застосування JavaScript:

- Веброзробка: JavaScript є фундаментальною мовою для написання сценаріїв на стороні клієнта у веброзробці, забезпечуючи інтерактивну та динамічну взаємодію з користувачем у веббраузерах.
- Серверні додатки: У Node.js JavaScript використовується для створення масштабованих та ефективних серверних додатків, пропонуючи уніфіковану мову як для клієнтської, так і для серверної розробки.

Розробка мобільних додатків: Такі технології, як React Native, використовують JavaScript для створення крос-платформних мобільних додатків, використовуючи спільну кодову базу для iOS та Android.

- IoT та вбудовані системи: Фреймворки JavaScript, такі як Johnny-Five та Espruino, дозволяють програмувати мікроконтролери та пристрої Інтернету речей, розширюючи сферу застосування JavaScript до апаратного забезпечення.
- Інші застосування: JavaScript можна використовувати в різних інших сферах, включаючи програми для смарт-годинників, десктопні програми (за допомогою Electron), розробку ігор (за допомогою фреймворків на кшталт Phaser) тощо.

Широке розповсюдження JavaScript забезпечує його інтеграцію в нові технології та фреймворки, що робить його універсальною мовою, яка постійно розвивається і підходить для різноманітних сценаріїв програмування. Незалежно від того, чи ви початківець, чи досвідчений розробник, JavaScript забезпечує надійну основу для створення сучасних програмних додатків на різних платформах і для різних сценаріїв використання.

2.4. Вибір текстового редактору

Текстовий редактор є основним інструментом для створення коду, що вимагає високої точності, надійності та ефективності у використанні. Таким чином, критерії, що розглядаються, охоплюють зручність використання,

багатство функцій, розширюваність, продуктивність і підтримку спільноти, причому кожен параметр піддається ретельному оцінюванню.

Зручність використання є ключовим фактором у процесі відбору. Складний або неефективний інтерфейс знижує продуктивність і відволікає від основної задачі - розробки коду. Обраний редактор повинен мати інтуїтивно зрозумілий макет, що забезпечує швидку навігацію та безперешкодну інтеграцію основних функціональних можливостей. Крім того, вміння взаємодіяти з системами контролю версій та інтеграція інструментів налагодження підвищує узгодженість робочого процесу, полегшуючи спільну роботу в командах розробників.

Функціональне багатство є ще одним незамінним критерієм, що визначає ефективність редактора в роботі з тонкощами веброзробки. Повноцінна підтримка HTML, JavaScript і CSS є основою, а допоміжні функції, такі як підсвічування синтаксису, автозавершення і розбиття коду, сприяють підвищенню ефективності та дотриманню стандартів кодування. Розширені можливості, такі як керування фрагментами та багатокурсорне редагування, ще більше підвищують точність та гнучкість у створенні коду.

Розширюваність є ключовим параметром, що відображає гнучкість та динамічність середовища, в якому відбувається розробка. Здатність редактора адаптуватися до сторонніх плагінів і розширень дає можливість розробникам пристосовувати середовище до вимог конкретного проекту та індивідуальних уподобань. Така гнучкість необхідна для інтеграції систем виконання завдань, пакувальників або статичних конструкторів сайтів, що дозволяє безперешкодно впроваджувати методології та технології, що стрімко розвиваються.

Продуктивність є одним з найважливіших критеріїв вибору, що підкреслює важливість швидкого реагування та стабільності. Текстовий редактор повинен демонструвати стабільну роботу в різних операційних середовищах і з різною складністю кодової бази, забезпечуючи безперебійне кодування навіть у ресурсномістких сценаріях.

Нарешті, підтримка спільноти означає життєздатність екосистеми редактора. Активна та зацікавлена спільнота сприяє колективному навчанню, усуненню помилок та постійному вдосконаленню можливостей редактора.

Обширні колекції навчальних посібників, документації та онлайн-форуми слугують безцінними ресурсами, що підвищують кваліфікацію розробників і розширюють можливості редактора.

2.4.1. Notepad++

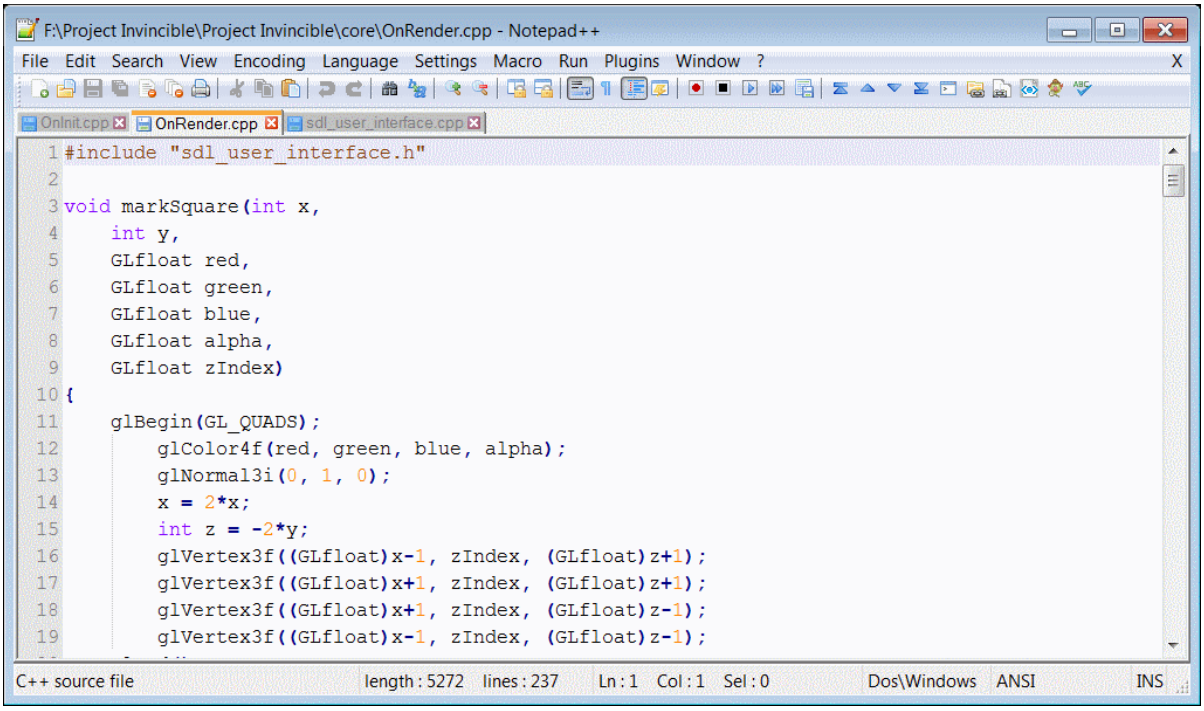

Рис. 2.1. Інтерфейс редактора Notepad++

У сфері текстових редакторів Notepad++ є конкурентом, що характеризується простотою та універсальністю. Його інтерфейс, хоча і є досить примітивним у порівнянні з сучасними аналогами, проте відрізняється зручністю та простотою у використанні. Notepad++ чудово справляється з основними завданнями редагування коду, забезпечуючи надійну підтримку підсвічування синтаксису HTML, JavaScript і CSS, доповнену численними мовними плагінами для розширення функціональності. Його легка архітектура забезпечує швидкий запуск і мінімальне споживання ресурсів, що ідеально підходить для розробників, які працюють в умовах обмежених ресурсів або прагнуть спростити процес редагування.

Проте, хоча Notepad++ вирізняється простотою, йому бракує додаткових функцій та можливостей розширення. Хоча цього достатньо для невеликих проектів, його обмежена підтримка сучасних процесів розробки, таких як

інтегрований контроль версій або складні інструменти налагодження, може виявитися недостатньою для масштабних і складних проектів. Крім того, його екосистема плагінів, хоч і велика, але не має тієї глибини та різноманітності, що є у більш сучасних редакторах, що обмежує адаптивність до парадигм розробки, які постійно змінюються.

Що стосується продуктивності, то Notepad++ зберігає відмінну стабільність і швидкість реагування навіть при роботі з кодовими базами середнього розміру. Його швидка робота забезпечує мінімальний вплив на ефективність робочого процесу, дозволяючи розробникам зосередитися на створенні коду без зайвих відволікань та затримок. Проте зі збільшенням розміру та складності проектів Notepad++ може демонструвати певні обмеження в обробці ресурсоємних завдань, що потенційно може призвести до падіння продуктивності під час виконання таких операцій, як синтаксичний аналіз великих файлів або складних операцій пошуку та заміни.

Підтримка спільноти користувачів Notepad++ залишається досить активною, хоча і більш спеціалізованою порівняно з іншими популярними редакторами. Користувачі редактора беруть активну участь у розробці плагінів, вирішенні проблем та обміні досвідом на онлайн-форумах і в спільнотах. Незважаючи на відсутність великих ресурсів і мереж підтримки більших платформ, спільнота Notepad++ залишається цінним ресурсом для розробників, які шукають допомоги та порад щодо розширення можливостей редактора.

Таким чином, Notepad++ пропонує надійне і легке рішення для основних завдань редагування коду, що підходить для розробників, які ставлять на перше місце простоту і ефективність у своєму робочому процесі. Його простий інтерфейс у поєднанні з належною мовною підтримкою та продуктивністю робить його привабливим варіантом для малих і середніх проектів або розробників з мінімалістичними уподобаннями. Однак, відсутність розширених функцій і можливостей розширення може створювати обмеження для більших і складніших проектів, що вимагає ретельного вивчення вимог проекту і пріоритетів розвитку перед прийняттям рішення про його використання.

2.4.2. Visual studio

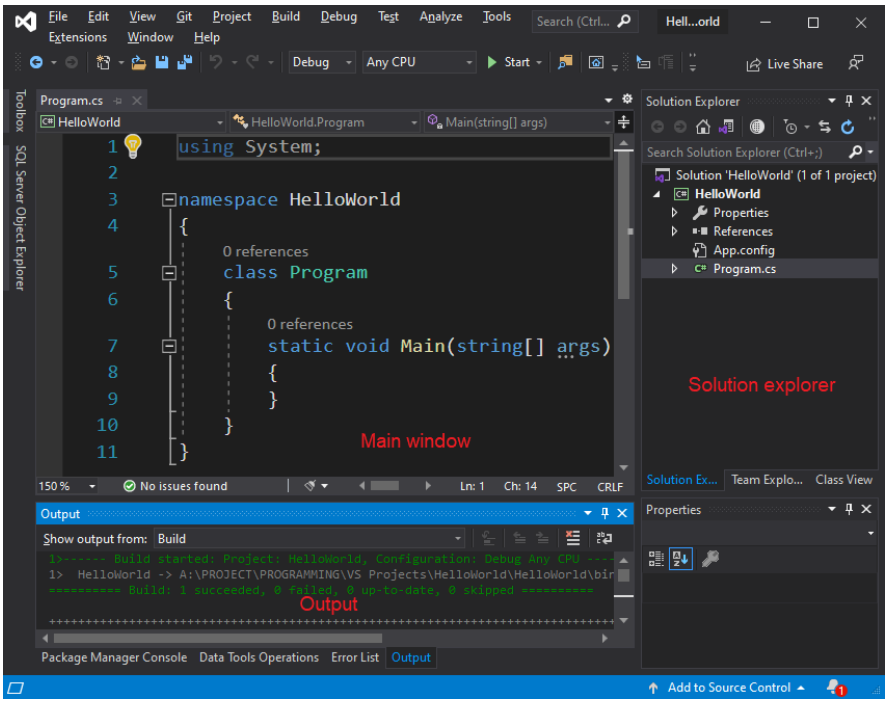

Рис. 2.2. Інтерфейс редактора Visual Studio

Visual Studio є одним з найпопулярніших текстових редакторів, оскільки має широкий набір функцій, що відповідають вимогам веброзробки. Відомий своєю зручним середовищем і широким інструментарієм, Visual Studio пропонує багатогранний підхід до створення коду та управлінням проектами. Його інтегроване середовище розробки (IDE) охоплює повний набір інструментів, включаючи сучасні засоби відлагодження, редагування та тестування, які органічно інтегровані в єдиний інтерфейс.

Відмінною рисою Visual Studio є унікальна здатність підтримувати великомасштабні проекти та розробку корпоративного рівня. Надійна архітектура редактора та сучасні інструменти проєктного управління враховують особливості складних веброзробок, сприяючи безперешкодній співпраці та інтеграції контролю версій у великих командах розробки. Крім того, вбудована інтеграція Visual Studio зі службами Azure дозволяє розробникам спростити процеси розгортання та моніторингу, сприяючи створенню повного життєвого циклу від початку розробки до впровадження.

Крім того, Visual Studio чудово підходить для крос-платформної розробки завдяки вбудованій підтримці .NET Core, Xamarin та інших фреймворків, що

полегшує обмін кодом і його повторне використання в різних операційних системах. Ця універсальність поширюється і на веброзробку, дозволяючи розробникам використовувати уніфікований інструментарій для створення адаптивних, ефективних вебдодатків, які виходять за межі окремих платформ.

Однак, за широкий набір функцій Visual Studio доводиться розплачуватися високою складністю та значними ресурсними витратами. Розширений інструментарій редактора може стати надто складним для початківців або тих, хто прагне мінімалістичного середовища розробки програмного забезпечення. Крім того, його пропрієтарна природа обмежує можливості персоналізації порівняно з більш гнучкими альтернативами.

Таким чином, Visual Studio є чудовим варіантом для проектів веброзробки, що характеризуються масштабом, складністю та вимогами корпоративного рівня. Потужний набір функцій, бездоганна інтеграція з технологіями Microsoft та підтримка крос-платформної розробки роблять його сильним конкурентом серед текстових редакторів. Однак його складність і вимогливість до ресурсів можуть створити проблеми для розробників, які прагнуть простоти або більшої гнучкості свого середовища розробки.

2.4.3. Visual studio code

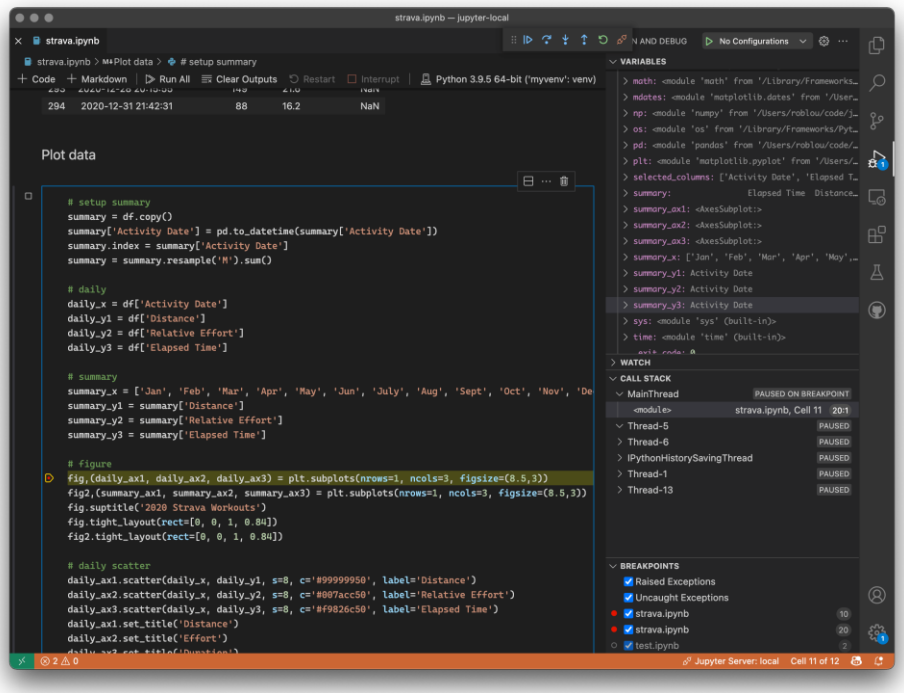

Рис. 2.3. Інтерфейс редактора Visual Studio Code

Visual Studio Code (VS Code), відомий своєю універсальністю та ефективністю, реалізує мінімалістичний, але ефективний підхід до редагування коду. Його користувацький інтерфейс дотримується тонкого балансу між простотою та функціональністю, створюючи середовище, сприятливе для створення якісного коду.

Завдяки безшовній інтеграції систем контролю версій та інструментів налагодження, VS Code сприяє оптимізації робочого процесу, сприяючи співпраці та підвищенню продуктивності команд розробників.

Набір функцій VS Code є надійним і всеосяжним, задовольняючи будь-які потреби вебразробників. Основні мови, такі як HTML, JavaScript і CSS, повністю підтримуються і доповнюються безліччю функцій, включаючи підсвічування синтаксису, автозавершення і форматування коду. Особливої уваги заслуговує багата бібліотека розширень, що охоплює широкий спектр функціональних можливостей - від автоматизації завдань до розширеної мовної підтримки. Така гнучкість дозволяє розробникам налаштовувати середовище редагування відповідно до вимог конкретного проекту, тим самим підвищуючи ефективність та якість коду.

Продуктивність - відмінна риса VS Code, що характеризується швидким відгуком і стабільністю навіть при роботі з великими кодовими базами або ресурсоємними завданнями. Завдяки легкій архітектурі та ефективному використанню ресурсів редактор забезпечує плавне написання коду на різних операційних системах та апаратних конфігураціях. Ця надійність лежить в основі придатності редактора для вимогливих сценаріїв розробки, де продуктивність не підлягає обговоренню.

Крім цього, VS Code може похвалитися активною та зацікавленою спільнотою, яка сприяє його постійному вдосконаленню та розвитку. Безліч навчальних посібників, документації та онлайн-форумів надають широку підтримку розробникам, які прагнуть максимізувати свої навички та реалізувати весь потенціал редактора. Крім того, активна екосистема розробки забезпечує своєчасні оновлення та вдосконалення, що дозволяє VS Code відповідати постійно розвиваючомуся середовищу технологій веброзробки.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі було проведено детальний аналіз засобів розробки, таких як HTML, CSS та JavaScript, що є основними технологіями для створення вебсторінок та вебдодатків. HTML (HyperText Markup Language) відповідає за структуру та вміст вебсторінки, забезпечуючи основу для всіх вебдокументів. CSS (Cascading Style Sheets) використовується для опису зовнішнього вигляду HTML-елементів, включаючи кольори, шрифти, макети та інші аспекти дизайну, що дозволяє зробити вебсторінки привабливими та зручними для користувачів. JavaScript, у свою чергу, додає динамічність і інтерактивність, дозволяючи розробникам створювати складні функціональні можливості, такі як валідація форм, анімації та обробка подій.

Для пришвидшення роботи та спрощення процесу розробки було обрано фреймворк Bootstrap. Він надає готові компоненти та стилі, що значно скорочує час на створення інтерфейсу користувача. Використання Bootstrap дозволяє розробникам швидко і легко створювати сучасні та адаптивні дизайни без необхідності писати великий обсяг власного CSS-коду.

Також у другому розділі було проведено порівняння найбільш популярних текстових редакторів, які використовуються для написання коду. Зокрема, були розглянуті такі редактори, як Notepad++, Visual Studio та Visual Studio Code. В результаті аналізу було визначено, що Visual Studio Code є найбільш оптимальним вибором. Він має зручний інтерфейс, високу швидкість роботи і постійно оновлюється, що робить його потужним інструментом для розробників.

РОЗДІЛ 3
СТВОРЕННЯ ВЕБСАЙТУ КАВ’ЯРНІ

3.1. Створення вебсайту

3.1.1. Структура вебсайту

Вебсайт кав'ярні відображається у вигляді структурованого комплексу, що складається з трьох основних директорій: "pages", "images", "scripts" і "css" (Рис. 3.1). Кожна з цих директорій має свою визначену функцію, сприяючи в цілому функціональності та естетиці вебсайту.

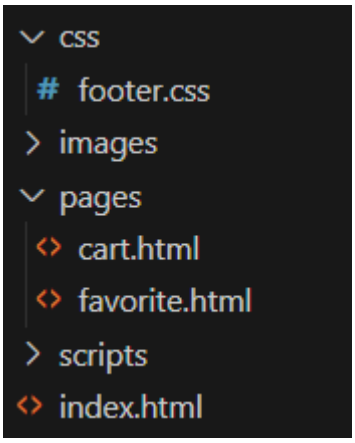

Рис. 3.1. Структура вебсайту

Перш за все, директорія "pages" призначена для зберігання файлів сторінок. Вона включає в себе всі необхідні HTML файли, які формують контент та структуру окремих сторінок вебсайту. Тут відбувається розміщення усієї інформації, що відображається перед користувачем.

Директорія "images" призначена для зберігання всіх використовуваних на вебсайті зображень. Це можуть бути фотографії, ілюстрації або будь-які інші графічні елементи, які допомагають зробити вебсайт більш привабливим та інформативним для відвідувачів.

Директорія "scripts" містить файли JavaScript, які відповідають за функціональність та взаємодію вебсайту з користувачем. JavaScript використовується для створення різноманітних інтерактивних елементів, а також для виконання різноманітних обчислень і операцій на клієнтському боці.

Нарешті, директорія "css" містить файли стилів, які відповідають за візуальне оформлення вебсайту. Вони містять правила форматування, кольори, шрифти та інші параметри, які визначають зовнішній вигляд інтерфейсу вебсайту.

Крім цих директорій, в кореневій частині вебсайту розташований файл "index.html", який є головною сторінкою та відображається користувачеві при переході на домашню сторінку. Цей файл визначає загальну структуру вебсайту та посилається на всі необхідні ресурси, що розташовані у вищезгаданих директоріях, забезпечуючи цілісність та правильне відображення вмісту.

3.1.2. Головна сторінка

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
      crossorigin="anonymous"
    />
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css"
    />
  </head>
```

Рис. 3.2. Елемент <head>

Для початку, у відкритому тезі <html> вказується мова сторінки - українська. У відкритому тезі <head>, яка містить інформацію про сторінку, встановлюються налаштування метатегів. Метатег <meta charset="UTF-8" /> визначає кодування символів як UTF-8, що дозволяє коректно відображати символи різних мов у веббраузері. Метатег <meta name="viewport"

content="width=device-width, initial-scale=1.0" /> налаштовує масштабування вебсторінки для пристроїв з різною шириною екрану, щоб забезпечити оптимальний вигляд на мобільних та інших пристроях. Назва вкладеного документа, яка відображається на вкладці браузера, задається у тегу <title>.

Після цього, в <head> також використовуються дві посилання <link> для підключення зовнішніх ресурсів. Перше посилання підключає стилі з бібліотеки Bootstrap версії 5.3.3, яка містить готові CSS-стилі для оформлення вебсторінок. У цьому випадку, використовується CDN (Content Delivery Network) - мережа, яка поширює ресурси через різні сервери по всьому світу, щоб прискорити їх завантаження. Друге посилання підключає значки з бібліотеки Bootstrap Icons версії 1.11.3, яка містить іконки для використання у вебсторінках. Використання CDN для завантаження цих ресурсів дозволяє швидко та зручно використовувати їх у проекті без необхідності завантажувати файли локально.

```
<!--Navbar-->
<div class="container-fluid">
  <div class="row">
    <div class="col-sm-auto bg-light sticky-top">
      <div
        class="d-flex flex-sm-column flex-row flex-nowrap bg-light align-items-center sticky-top"
      >
        <a
          href="/"
          class="d-block p-3 link-dark text-decoration-none"
          title=""
          data-bs-toggle="tooltip"
          data-bs-placement="right"
          data-bs-original-title="Icon-only"
        >
          <i class="bi-cup-hot-fill fs-1"></i>
        </a>
        <ul
          class="nav nav-pills nav-flush flex-sm-column flex-row flex-nowrap mb-auto mx-auto text-center justify-content-between w-100 px-3 align-items-center"
        >
          <li class="nav-item">
            <a
              href="/index.html"
              class="nav-link py-3 px-2"
              title=""
              data-bs-toggle="tooltip"
              data-bs-placement="right"
              data-bs-original-title="Home"
            >
              <i class="bi-house fs-1"></i>
            </a>
          </li>
```

Рис. 3.3. Навігаційне меню

Ця частина створює бічну навігаційну панель для. Вона включає в себе посилання на головну сторінку "/", а також посилання на інші сторінки, такі як "/index.html", "/pages/favorite.html", "/pages/cart.html". Кожне посилання має клас "nav-link", щоб стилізувати його як елемент навігації. Для додавання іконок до посилань використовуються іконки з бібліотеки Bootstrap Icons, наприклад, "bi-house", "bi-heart", "bi-cart2". Іконки мають клас "fs-1", що встановлює їх розмір.

Навігаційна панель також має світлий фоновий колір "bg-light" та закріплена так (sticky-top), щоб вона залишалася видимою при прокручуванні сторінки донизу.

```
<!--Main Image-->
<div class="col-md vh-100 text-center">
  <img
    src="/images/14723637_dw21batch4-sasi-27.svg"
    alt=""
    class="vh-100"
  />
</div>
```

Рис. 3.4. Декоративне зображення

В цій частині створюється блок зображення, яке розміщується по центру сторінки та займає всю доступну висоту екрану.

Код розміщено всередині , який має клас клас "vh-100", який встановлює висоту елементу на висоту екрану.

Усередині знаходиться <img> тег, який відображає саме зображення.

```
<div class="container d-grid border-top gap-5 pt-5">
  <div class="row justify-content-md-center">
    <div class="col-md-4 d-flex">
      <img
        src="/images/americano_hd.png"
        alt=""
        class="float-start col-md-6 rounded"
      />
      <div class="d-flex flex-column justify-content-between">
        <div>
          <h1>Американо</h1>
          <p class="text-secondary">
            30мл еспресо <br />
            90мл води
          </p>
        </div>
        <div class="d-flex">
          <button
            type="button"
            class="btn btn-dark py-2 px-5 w-100 rounded-0 rounded-start border-end"
            id="americano-cart"
            onclick="addToCart('americano')"
          >
            Замовити
          </button>
          <button
            type="button"
            class="btn btn-dark py-1 px-2 bi-heart px-3 rounded-0 rounded-end"
            id="americano-heart"
            onclick="addToFav('americano')"
          ></button>
        </div>
      </div>
    </div>
```

Рис. 3.5. Меню

Ця частина відображає кавові напої на вебсторінці. Вони розміщені всередині контейнера (елемент з класом "container"), який використовує сітку Bootstrap для організації вмісту.

Усередині контейнера розміщуються два блоки, кожен з яких представляє один кавовий напій. Кожен блок відображається в ряді, використовуючи клас "row justify-content-md-center", який центрує блоки по горизонталі.

У кожному блоку є зображення кавового напою (елемент <img>), яке відображається зліва, та опис кавового напою (елементи <h1> та <p>). Опис включає назву напою та його склад. Крім того, для кожного напою є дві кнопки: одна для додавання товару у кошик і друга для додавання його у список обраних.

Кожна кнопка має відповідні обробники подій onclick, які викликають JavaScript функції addToCart() та addToFav() з аргументами, що вказують на конкретний кавовий напій.

```
<!--Footer-->
<footer class="footer">
  <div class="bg-light py-5 py-xl-8 py-xxl-9 border-top border-light">
    <div class="container overflow-hidden">
      <div class="row gy-5 gy-md-0 align-items-md-center">
        <div class="col-xs-12 col-md-7 order-1 order-md-0">
          <div class="copyright text-center text-md-start">
            &copy; 2024. All Rights Reserved.
          </div>
        </div>

        <div class="col-5 text-end text-secondary">
          <p>Час роботи: 11:00 - 21:00</p>
        </div>
      </div>
    </div>
  </div>
</footer>
```

Рис. 3.6. Футер

Цей фрагмент створює нижній колонтитул (футер) для вебсторінки. Футер - це розділ вебсторінки, який зазвичай містить інформацію про авторські права, контактну інформацію або інші важливі елементи, які повинні бути доступні на всіх сторінках сайту.

У відкритому тезі <footer> вказується клас "footer", який використовується для стилізації футера за допомогою CSS. У вкладеному блоку встановлюється стиль фону (світлий), відступи зверху та знизу. Також додається верхня рамка, що розділяє футер від вмісту сторінки.

У вкладеному блоку з класом "container overflow-hidden" створюється контейнер для організації вмісту футера. У внутрішньому блоку використовується сіткова система для розташування елементів у рядок.

Далі вказані два блоки, які представляють дві колонки. Перша колонка займає всю доступну ширину на малих пристроях і 7/12 ширини на пристроях середнього розміру (md), вона містить інформацію про авторські права. Друга колонка займає 5/12 ширини на пристроях середнього розміру і містить інформацію про час роботи.

У другій колонці текст вирівнюється по правому краю, а текстовий колір встановлюється як вторинний, сам текст дає користувачу інформацію про робочий час.

```
<script src="/scripts/addToCart.js"></script>
<script src="/scripts/addToFav.js"></script>
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
  crossorigin="anonymous"
></script>
```

Рис. 3.7. Підключення файлів JavaScript

В самому кінці файлу у нас вказуються посилання на JavaScript-файли, які відповідають за функціонал додавання товарів до кошика та до списку обраних, а також підключає JavaScript-файл з бібліотекою Bootstrap.

3.1.3. Улюблене

```
<!--Content-->
<div class="col-sm">
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-md-7 mt-3">
        <div id="fav-items">
        </div>
      </div>
    </div>
  </div>
</div>
</div>
```

Рис. 3.8. Блок улюблених напоїв

Сторінка з улюбленими напоями повторно використовує навігаційне меню та футер з основної сторінки. Для відображення самих напоїв створюється блок з розміткою (Рис. 3.8) та підключається скрипт (Рис. 3.8).

```
let fav = JSON.parse(localStorage.getItem("fav")) || [];

let favItemsContainer = document.getElementById("fav-items");

fav.forEach((item) => {
  let itemDiv = document.createElement("div");
  itemDiv.classList.add("card", "mb-3");
  itemDiv.innerHTML = `
<div class="card-body d-flex justify-content-between align-items-center">
    <img src="${itemData[item].image}" alt="${item}" class="mr-3" style="max-width: 100px;">
    <div>
        <h5 class="card-title">${itemData[item].name}</h5>
    </div>
    <div class="d-flex align-items-center"> <!-- Added container for buttons -->
        <button class="btn btn-dark bi-cart2 me-1" id="${item}-cart" onclick="addToCart('${item}')"></button>
        <button class="btn btn-danger bi-trash3" onclick="removeItem('${item}')"></button>
    </div>
</div>

  favItemsContainer.appendChild(itemDiv);
});

function removeItem(item) {
  fav = fav.filter((i) => i !== item);
  localStorage.setItem("fav", JSON.stringify(fav));
  location.reload();
}
```

Рис. 3.9. Скрипт генерації улюблених напоїв

Скрипт витягує дані про улюблені елементи з локального сховища браузера за допомогою методу localStorage.getItem. Якщо дані не знайдено, створюємо

порожній масив. Далі отримуємо контейнер для улюблених елементів за допомогою document.getElementById("fav-items").

Після цього використовуємо forEach для перебору кожного елемента у масиві fav. Для кожного елемента створюється новий div з класом "card" для відображення карточки елемента. Внутрішній HTML цього div містить зображення елемента, його назву та дві кнопки - для додавання у кошик та видалення зі списку улюблених.

3.1.4. Кошик

```
<!--Content-->
<div class="col-sm">
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-md-7 mt-3">
        <div id="cart-items">
        </div>
        <div class="d-flex justify-content-end">
          <button
            type="button"
            class="btn btn-dark col-md-3"
            onclick="generateQRCode()"
          >
            Придбати
          </button>
        </div>
        <div
          class="modal fade"
          id="qrCodeModal"
          tabindex="-1"
          aria-labelledby="qrCodeModalLabel"
          aria-hidden="true"
        >
          <div class="modal-dialog modal-dialog-centered">
            <div class="modal-content">
              <div class="modal-header">
                <h5
                  class="modal-title text-center"
                  id="qrCodeModalLabel"
                >
                  Ваше замовлення
                </h5>
                <button
                  type="button"
                  class="btn-close"
                  data-bs-dismiss="modal"
                  aria-label="Close"
                ></button>
              </div>
              <div class="modal-body" id="qrCodeContent">
                <!-- QR code -->
              </div>
            </div>
          </div>
        </div>
```

Рис. 3.10. Сторінка кошика

Для сторінки кошика було створено динамічну генерацію обраних товарів, подібну до генерації улюблених товарів (Рис 3.9). Після цього знаходиться кнопка "Придбати", яка активує функцію generateQRCode() при натисканні.

Наступною є модальне вікно, яке міститься у з класом "modal fade". Це вікно використовується для відображення QR-коду після натискання кнопки "Придбати". Інформація про замовлення буде відображатися в заголовку модального вікна, а QR-код - у вмісті модального вікна.

```
function generateQRCode() {
    let cart = JSON.parse(localStorage.getItem("cart")) || [];

    let cartData = JSON.stringify(cart);

    let qr = qrcode(0, 'M');
    qr.addData(cartData);
    qr.make();

    let qrCodeContainer = document.getElementById("qrCodeContent");

    qrCodeContainer.innerHTML = "";

    let qrCodeImage = qr.createImgTag(10);

    qrCodeImage = qrCodeImage.replace('<img ', '<img style="display: block; margin: 0 auto;" ');

    qrCodeContainer.innerHTML = qrCodeImage;

    let qrCodeModal = new bootstrap.Modal(document.getElementById("qrCodeModal"));
    qrCodeModal.show();
}
```

Рис 3.11. Скрипт генерації QR-коду

JavaScript-функція (Рис. 3.10), яка генерує QR-код з даними кошика товарів працює наступним чином:

1. Отримує дані кошика з локального сховища браузера.
2. Перетворює дані кошика у рядок.
3. Створює QR-код і додає до нього дані.
4. Очищає контейнер для QR-коду і додає нове зображення QR-коду.
5. Відкриває модальне вікно з QR-кодом.

3.1.5. Скрипти

Перша частина коду скрипта додавання до кошика (Рис. 3.12) встановлює прослуховувач подій, який очікує завантаження всіх елементів DOM (Document

Object Model). Коли це станеться, виконується функція, яка перевіряє, чи є вже деякі елементи у корзині. Це доцільно використовувати для ініціалізації даних користувача при завантаженні сторінки.

```
document.addEventListener("DOMContentLoaded", function () {
    function checkCart() {
      let cart = JSON.parse(localStorage.getItem("cart")) || [];
      cart.forEach(function (item) {
        const button = document.getElementById(item + "-cart");
        if (button) {
          button.classList.remove("btn-dark");
          button.classList.add("btn-success");
        }
      });
    }

    checkCart();
  });

  function addToCart(item) {
    let cart = JSON.parse(localStorage.getItem("cart")) || [];

    const index = cart.indexOf(item);
    if (index > -1) {
      cart.splice(index, 1);
      document.getElementById(item + "-cart").classList.remove("btn-success");
      document.getElementById(item + "-cart").classList.add("btn-dark");
    } else {
      cart.push(item);
      document.getElementById(item + "-cart").classList.remove("btn-dark");
      document.getElementById(item + "-cart").classList.add("btn-success");
    }

    localStorage.setItem("cart", JSON.stringify(cart));
  }
```

Рис. 3.12. Скрипт додавання до кошика

Функція addToCart(item) відповідає за додавання або видалення товару з корзини. Вона приймає параметр item, який є ідентифікатором товару. Спочатку функція отримує дані збережені в локальному сховищі браузера про товари у корзині. Потім вона перевіряє, чи вже є даний товар у корзині: якщо так, він видаляється, якщо ні - додається. Після зміни стану товару у корзині, відповідний елемент DOM (кнопка) оновлюється, змінюючи його клас стилю для відображення відповідного стану (додано або видалено з корзини). Нарешті, дані про корзину оновлюються в локальному сховищі браузера, щоб зберегти зміни.

Кнопки для додавання до списку улюблених працюють аналогічним чином.

```
const itemData = {
  americano: {
    name: "Американо",
    description: ["30мл еспресо", "90мл води"],
    image: "/images/americano_hd.png",
  },
  latte: {
    name: "Латте",
    description: ["60мл еспресо", "30мл молока", "30мл спіненого молока"],
    image: "/images/latte_hd.png",
  },
  breve: {
    name: "Бреве",
    description: ["60мл еспресо", "30мл вершків"],
    image: "/images/coffee_breve_hd.png",
  },
  flat_white: {
    name: "flat_white",
    description: ["90мл еспресо", "30мл спіненого молока"],
    image: "/images/flat_white_hd.png",
  },
```

Рис. 3.13. Список напоїв

Для коректної роботи інших скриптів було створено файл з об’єктом itemData, який містить інформацію про кожен напій, доступний у магазині.

Для кожного напою, визначеного як ключ у itemData, є об'єкт, що містить такі дані:

- name: Назва напою.
- description: Масив рядків, які містять опис складових напою.
- image: Шлях до зображення напою.

3.2. Огляд роботи вебсайту

При заході на сайт користувача зустрічає чорно-біла ілюстрація кавової плантації (Рис. 3.14), яка одразу налаштовує на атмосферу затишку та автентичності. Використання чорно-білих тонів підкреслює класичний характер зображення, викликаючи асоціації з історією та майстерністю вирощування кави. Такий дизайнерський прийом сприяє формуванню у користувачів довіри до бренду та викликає інтерес до подальшого перегляду сайту.

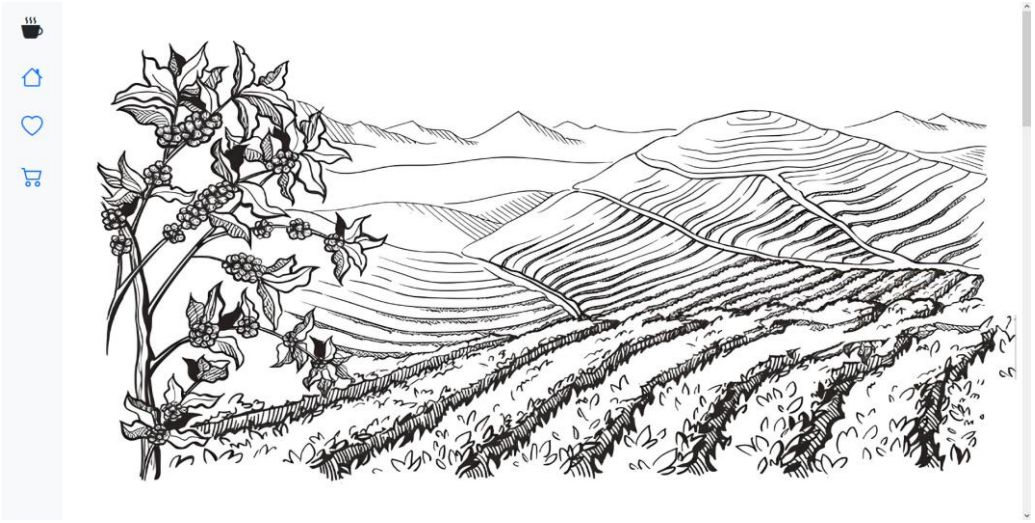

Рис. 3.14. Ілюстрація кавової плантації

На головній сторінці представлено всі доступні напої, які є на сайті (Рис. 3.14). Біля кожного напою наведено його склад та зображення, що сприяє зручному пошуку потрібного напою для користувача.

Для зручності користувачів передбачено функцію швидкого додавання напоїв до улюблених або до кошика. При цьому, зміна кольору кнопок сигналізує про успішне виконання дії, що забезпечує наочний зворотний зв'язок і дозволяє уникнути необхідності переходу на інші сторінки для перевірки коректності виконання операції. Це підвищує ефективність взаємодії з сайтом, спрощуючи процес вибору та покупки напоїв.

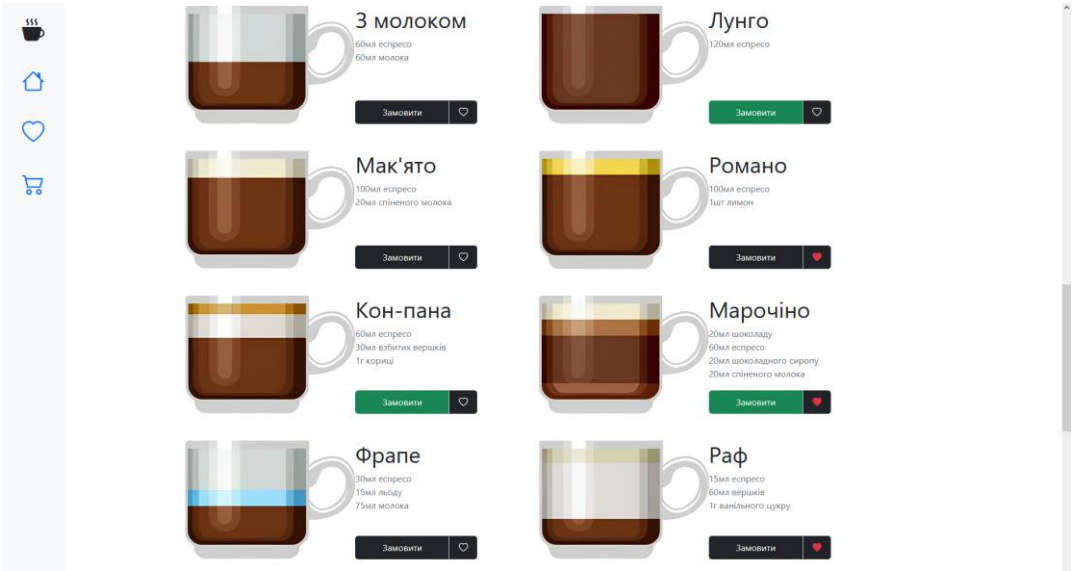

Рис. 3.15. Меню напоїв

На сторінці "Улюблене" (Рис. 3.16) представлено перелік напоїв, які користувач позначив як улюблені. Ця сторінка надає можливість швидко переглядати та керувати обраними напоями. Користувач може додати будь-який з напоїв до кошика для здійснення покупки або видалити напій зі списку улюблених, якщо він більше не бажає зберігати його в цьому переліку. Це забезпечує зручність і швидкість у здійсненні покупок, дозволяючи користувачу легко доступати до своїх найулюбленіших напоїв.

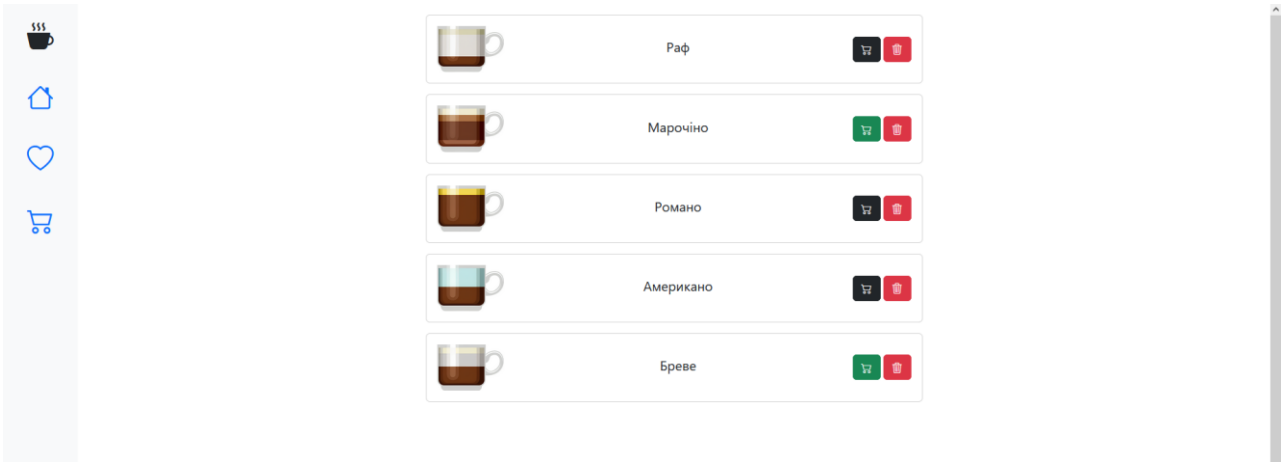

Рис. 3.16. Сторінка «Улюблене»

На сторінці «Кошик» відображається детальний перелік напоїв, які користувач додав до свого кошика під час процесу вибору (Рис. 3.17). Користувач має можливість переглядати цей список, оцінювати кожен вибраний напій та вносити корективи за потреби, видаляючи ті напої, які він більше не бажає придбати. Якщо користувач задоволений усіма обраними позиціями, він може завершити процес купівлі, натиснувши кнопку «Придбати».

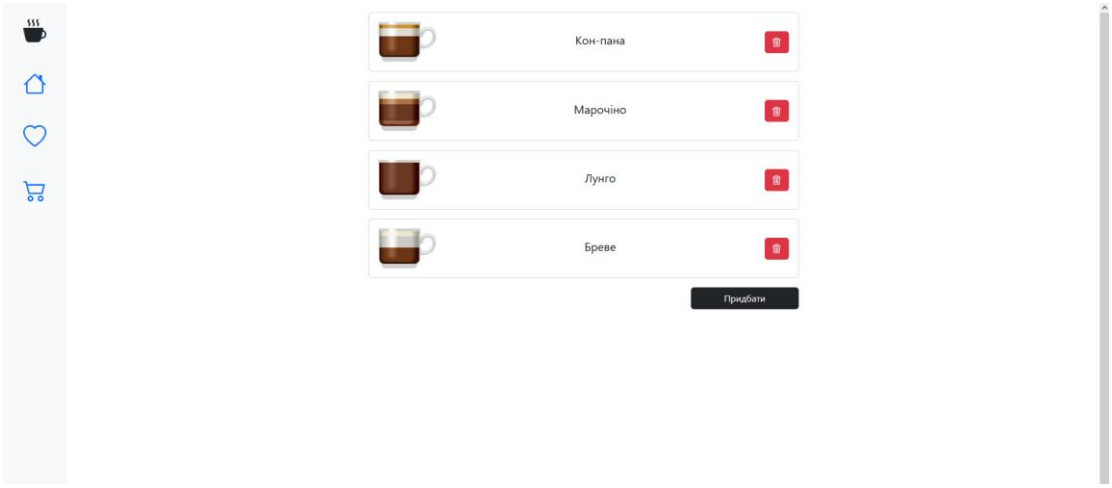

Рис. 3.17. Сторінка «Кошик»

Після цього система автоматично згенерує унікальний QR-код, який буде відображено на екрані (Рис. 3.18). Цей QR-код є необхідним для підтвердження замовлення. Користувач повинен пред'явити його персоналу, щоб завершити процес оформлення та отримати свої напої. Це забезпечує швидкий і зручний спосіб завершення покупки та отримання замовлених напоїв.

Рис. 3.18. Згенерований QR-код

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі було детально продемонстровано процес розробки вебсайта для кав’ярні, використовуючи редактор коду Visual Studio Code. Було створено декілька ключових сторінок, таких як головна сторінка, сторінка улюблених напоїв та сторінка кошика для покупок, а також окремі елементи цих сторінок.

Кожна сторінка містить навігаційне меню, яке дозволяє користувачам легко переміщатися між різними розділами сайту, та футер, який дає додаткову інформацію.

На головній сторінці було реалізоване меню напоїв, що включає в себе інформацію про доступні напої, їхні ціни та зображення. Кожен напій на головній сторінці можна добавити до кошика або до улюбленого.

На сторінці улюблених напоїв користувачі можуть переглядати список обраних кавових напоїв.

Сторінка кошика для покупок дозволяє користувачам переглядати та керувати своїми замовленнями. Вона містить перелік напоїв, які були додані до кошика видаляти зайві позиції. Кожна позиція в кошику супроводжується зображенням, назвою та кнопкою для видалення. Під списком напоїв знаходиться кнопка для оформлення замовлення.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено вебсайт кав’ярні, який включає всі необхідні функції для зручного та швидкого замовлення напоїв. Під час роботи проведено ґрунтовний аналіз предметної області та аналогічних сайтів конкурентів, що дозволило визначити основні вимоги до вебсайту, виявити сильні та слабкі сторони існуючих рішень і сформувати критерії для створення унікального та функціонального продукту.

Детально проаналізовано інструменти та технології веброзробки, включаючи HTML, JavaScript, CSS та фреймворк Bootstrap, що забезпечили швидку розробку та кросплатформеність вебсайту. Текстовий редактор Visual Studio Code обрано як оптимальний інструмент для написання коду завдяки його численним розширенням та зручному інтерфейсу.

Створено інтуїтивно зрозуміле навігаційне меню та список напоїв, а також реалізовано можливість додавання напоїв до кошика та списку улюблених, що робить процес замовлення більш зручним та персоналізованим. На сторінці кошика впроваджено функціонал для генерування QR-коду, який значно пришвидшує процес замовлення та оплати.

Сайт розроблено у мінімалістичному стилі, що полегшує взаємодію користувача з інтерфейсом. Простий та чистий дизайн допомагає користувачам зосередитися на основних функціях, покращуючи користувацький досвід і роблячи процес замовлення більш ефективним.

Крім того, розроблений сайт передбачає можливість подальшого розширення та інтеграції з іншими сервісами. Це дозволяє додавати нові функції та оновлювати існуючі без значних витрат часу та ресурсів. Такий підхід гарантує, що вебсайт залишатиметься актуальним і відповідатиме потребам користувачів, забезпечуючи їм найкращий досвід при взаємодії з кав’ярнею. Таким чином, виконана робота охоплює всі ключові аспекти розробки сучасного вебсайту для кав’ярні, від аналізу до реалізації, забезпечуючи зручний спосіб замовлення напоїв за допомогою новітніх вебтехнологій та сучасних підходів до дизайну.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Muhammed Cindioglu. HTML from A to Z 2023. – 152 s.
2. Austin French. CSS Beginners' Basic Fundamental Guide (2022 Crash Course for Newbies), 2022 – 76 C
3. Daniel Charles Foreman, Daniel Foreman Bootstrap 5 Foundations 2021 – 140 s.
4. David Weathers. UX/UI Design 2021 For Beginners A Simple Approach to UX/UI Design for Intuitive Designers, 2021 – 78 s.
5. Mike McGrath. HTML, CSS & JavaScript in easy steps 2020 – 480 s.
6. Flavio Copes. The JavaScript Beginner’s Handbook 2020. – 76 s.
7. Abhilasha Sinha, Alok Ranjan, Ranjit Battewad. JavaScript for Modern Web Development 2020 – 440 s.
8. Документація CSS. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 23.05.2024)
9. Офіційна документація Bootstrap. URL: https://getbootstrap.com/docs/ (дата звернення: 23.05.2024)
10. Офіційна документація Visual Studio Code. Documentation. URL: https://code.visualstudio.com/docs/editor/whyvscode (дата звернення: 23.05.2024)
11. Важливість UX / UI дизайну. URL: https://makeitclear.com/what-is-the-importance-of-ui-and-ux-for-a-website/ (дата звернення: 23.05.2024)
12. Порівняльна характеристика текстових редакторів. URL: https://stackshare.io/stackups/notepad-plus-plus-vs-visual-studio-code-vs-visual-studio (дата звернення: 23.05.2024)
13. Положення про кваліфікаційні роботи (проєкти) здобувачів вищої освіти Національного авіаційного університету СМЯ НАУ П 03.01(10) – 03 – 2024