ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку для служби доставки їжі з використанням мов розмітки HTML-CSS, мови програмування Java Script»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Максим РЕБРИК
(підпис)

Виконав: здобувач вищої освіти групи ПД-44

Максим РЕБРИК

Керівник: Володимир САДОВЕНКО
к.ф.-м.н., доцент

Рецензент: _____

Київ `

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ребрика Максима Сергійовичу

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку для служби доставки їжі з використанням мов розмітки HTML-CSS, мови програмування Java Script» керівник кваліфікаційної роботи к.ф.-м.н., доцент Володимир САДОВЕНКО, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
   3.1. Науково-технічна література.
   3.2. Мова HTML.
   3.3. Мова CSS.
   3.4. Офіційна документація Node.js.
   3.5. Мова Java Script.
   3.6. GitHub.
   3.5. Git.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   4.1. Аналіз актуальності та огляд існуючих додатків.

4.2. Аналіз інструментів реалізації та розробка структури веб-додатка доставки їжі.
4.3. Реалізація програмної частини додатку.
4.4. Висновки.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до веб-додатку.
3. Програмні засоби реалізації.
4. Діаграма використання.
5. Діаграма активності.
6. Схема бази даних
7. Екранні форми
8. Апробація результатів дослідження

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Аналіз актуальності та дослідження існуючих веб-додатків служби доставки їжі	12.03-19.03.2024	
4	Проектування веб-додатку служби доставки їжі	20.03-29.03.2024	
5	Програмна реалізація веб-додатку служби доставки їжі	30.03-14.04.2024	
6	Тестування веб-додатку служби доставки їжі	15.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач(ка) вищої освіти _____ Максим РЕБРИК
(підпис)

Керівник
кваліфікаційної роботи _____ Володимир САДОВЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 40 стор., 2 табл., 25 рис., 10 джерел.

Мета роботи – спрощення та покращення процесу замовлення доставки їжі за рахунок застосування веб-додатку, створеного на мовах HTML, CSS, JS та їхні фреймворки.

Об'єкт дослідження – процес служби доставки їжі.

Предмет дослідження – веб-додаток служби для доставки їжі та різноманітних страв.

Короткий зміст роботи: В роботі проаналізовано алгоритми та методи для обробки природньої мови в контексті задачі служби доставки їжі. Проаналізовано інструментальні засоби для веб-додатку для служби доставки їжі: HTML, CSS, JS, React. Розроблено алгоритм роботи веб-додатку та програмно реалізовані ключові функціональні можливості, зокрема: можливість залишати відгуки, панель зворотного зв'язка, кнопка повторного замовлення, можливість додавати до списку улюбленого, реєстрація та авторизація, збереження замовлень, стійкість до навантажень, система відстеження замовлень. Проведено функціональне тестування веб-додатку. Веб-додаток був створений використовуючи мови програмування такі, як HTML, CSS, JavaScript.

ЗМІСТ

ВСТУП

Створення веб-додатка для служби доставки їжі в сучасних умовах виявляється особливо стратегічним важливим завданням, оскільки відбувається значущі трансформації від вибору та покупки товарів. За останні кілька років спостерігається значний ріст популярності самої послуги доставки їжі, що посилює попит на веб-додатки у цій галузі. Крім того, зі зростанням популярності зростає і конкуренція серед служб доставки, що змушує стежити за останніми технологічними тенденціями та постійно вдосконалювати свої додатки. Для користувачів веб-додатки стають зручним і ефективним способом замовити їжу безпосередньо зі свого смартфона чи ПК, не виходячи з дому або офісу. Також, завдяки збору та аналізу даних про замовлення та вподобання користувачів, веб-додатки можуть надавати персоналізовані рекомендації, що підвищує їх задоволення від використання сервісу. Такий веб-додаток може не лише задовольнити практичні потреби клієнтів у доставці продуктів, але й відкрити нові можливості для оптимізації процесів доставки за допомогою сучасних технологій, зменшуючи часу очікування та покращення якості обслуговування, що робить цю тему досить перспективною для подальшого розвитку та інновації.

Мета роботи – спрощення та покращення процесу замовлення доставки їжі за рахунок застосування веб-додатку, створеного на мовах HTML, CSS, JS та їхні фреймворки.

Завдання роботи:

– Аналіз потреб і специфіки доставки їжі для визначення основних вимог вимог до веб-додатку.

– Розробка веб-додатку доставки їжі, до якої входить структура, основні модулі та функції системи.

– Реалізація веб-додатку доставки їжі мовами програмувань HTML, CSS та Java Script.

Об’єкт дослідження – процес служби доставки їжі.

Предмет дослідження – веб-додаток служби для доставки їжі та

різноманітних страв.

Новизна цієї роботи полягає у специфічному підході до автоматизації та оптимізації процесів доставки їжі, а також у реалізації сучасного стилю веб-додатку, створеного з урахуванням потреб користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ ЗАСОБІВ ДЛЯ ЇЇ ВИРІШЕННЯ

1.1 Загальні особливості служби доставки їжі

У сучасному світі існує безліч ресторанів, кафе та закладів харчування, які пропонують різноманітні страви, включаючи десерти, фаст-фуд, японську кухню та багато іншого. Однак не завжди у всіх є можливість відвідати ці місця, щоб насолодитися улюбленими або новими стравами. У зв'язку з цим були створені зовані служби доставки їжі, які замість вас можуть оперативно забрати замовлення в потрібному ресторані та швидко доставити його прямо до вашого місця проживання або місце знаходження, що дає змогу не витрачати час та власних сил, щоб замовити страву.

Через те дані служби доставки їжі надають зручність та велику ефективність в ритмі нашого життя. А саме:

– Широкий вибір є однією з ключових переваг служб доставки їжі. Різні компанії пропонують дуже великий асортимент ресторанів, кафе або супермаркетів на вибір, де клієнти можуть замовити будь-які продукти або вже готові страви. Це дозволяє покупцям вибирати саме ті страви, які вони бажають, із різних кухень світу або спеціалізованих закладів з певними видами кухні. Незалежно від того, чи це сніданок, обід, вечеря або перекус, клієнти мають можливість замовити їжу, яка відповідає їхнім смакам і перевагам. Такий широкий вибір забезпечує задоволення потреб споживачів і робить процес замовлення їжі онлайн ще більш зручним і привабливим;

– Зручність та ефективність є ще однією важливою перевагою служб доставки їжі. Ці служби можуть значно заощадити час користувача, доставлячи замовлення до вказаного місця призначення швидко та ефективно. Крім того, вони володіють великою гнучкістю, яка дозволяє доставити замовлення в потрібний час, зручний для користувача. Наприклад, якщо вам необхідно отримати їжу під час обідньої перерви або вечері після роботи, ви можете запланувати доставку відповідно до вашого графіку. Це робить служби доставки

їжі ідеальним вибором для людей, які мають обмежений час або хочуть максимально ефективно використовувати свій час;

– Безпека та зручність є невід'ємною частиною служб доставки їжі, особливо в контексті пандемії COVID-19. Завдяки цим службам, клієнти можуть отримувати свої замовлення безпечно, уникнувши непотрібних контактів з іншими людьми. Процес замовлення та доставки відбувається без особистого контакту з кур'єром чи персоналом ресторану, оскільки оплата може бути здійснена онлайн, а саме замовлення доставлене прямо до дверей клієнта. Це дозволяє клієнтам не лише насолоджуватися своїми улюбленими стравами, а й зберігати своє здоров'я та безпеку в період пандемії, дотримуючись рекомендацій щодо соціального дистанціювання та гігієни.

1.2 Аналіз існуючих веб-додатків служби доставки їжі

1.2.1 Веб-Додаток Zakaz.ua

Zakaz.ua – веб-додаток створений українськими розробниками. Онлайн-сервіс доставки продуктів з супермаркетів. Він має досить простий і швидкий спосіб замовлення товарів, пропонуючи користувачам можливість купувати продукти з улюблених супермаркетів, таких, як: Ашан, Metro, Novus та ін.

Даний веб-додаток використовує персоналізовані рекомендації на основі попередніх покупок клієнтів. Продукти організовані за категоріями з можливістю фільтрації за ціною, брендом чи типом товару. Сайт забезпечує безпечні та зручні способи оплати.

Zakaz.ua має такі переваги:

– Добре пророблена пошукова система. Система пошуку на Zakaz.ua дуже ефективна та швидко знаходить необхідний товар, дозволяючи користувачам додати його до кошика без зайвих зусиль;

– Наявність Чат-Бота та чату для зворотного зв'язку. Zakaz.ua пропонує користувачам можливість отримати допомогу за допомогою Чат-Бота або спілкування з оператором у чаті, що підвищує рівень обслуговування та

забезпечує зручність користувачів;

– Окремі та унікальні сторінки для кожного закладу. Кожний заклад має свою окрему сторінку, що сприяє зручності вибору та ознайомленню з асортиментом товарів.

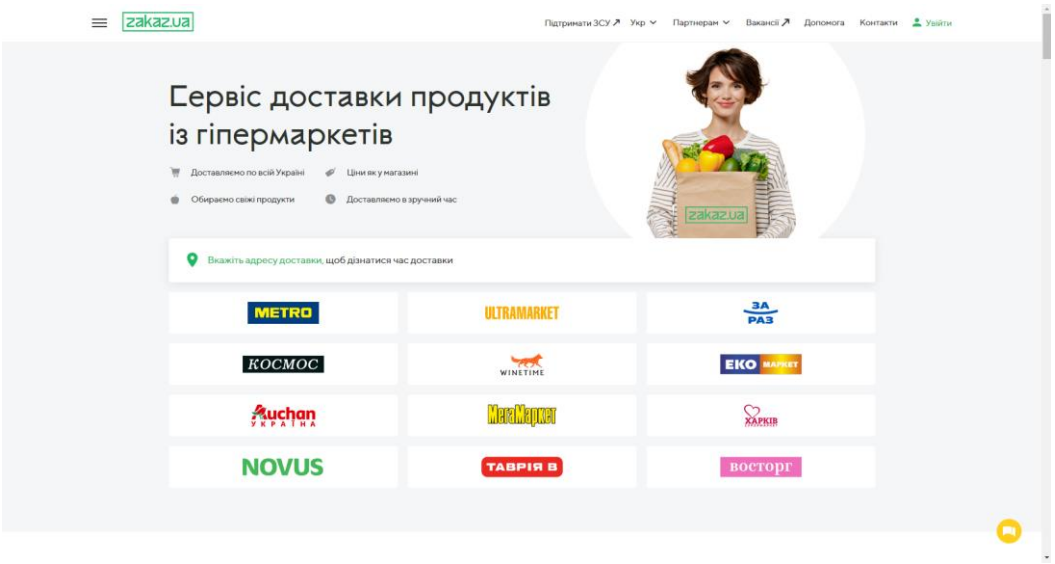


Рис. 1.1 Головне меню веб-додатку «Zakaz.ua»

Недоліки:

– Погана оптимізація сайту: Завантаження сторінок на Zakaz.ua може бути повільним через погану оптимізацію, що може призвести до незручностей у користувачів;

– Необов'язкове вказання адреси перед додаванням товарів до кошика. Платформа не дозволяє користувачам додавати товари до кошика без попереднього вказання адреси доставки, що може бути незручним у деяких випадках;

– Поганий дизайн. На сьогоднішній день дизайн Zakaz.ua може бути не дуже привабливим або застарілим, що може вплинути на користувачів;

– Неможливість додавання товарів з різних закладів до одного кошика. Система не дозволяє користувачам додавати товари з різних закладів до кошика одночасно, що може збільшувати кількість кроків для замовлення та ускладнювати процес покупок;

1.2.2 Веб-Додаток Такфур

Такфур - онлайн-супермаркет, який забезпечує надійну та оперативну доставку продуктів харчування прямо до дому або іншого місця призначення в Києві та області. Сервіс пропонує широкий асортимент товарів, включаючи свіжі овочі, фрукти, м'ясо, молочні продукти та багато іншого.

Однак, поточний веб-додаток має обмежений вибір продуктів і страв, що може не задовольнити всі потреби користувачів. Дизайн сайту також потребує покращення, оскільки його інтерфейс не є достатньо привабливим і сучасним.

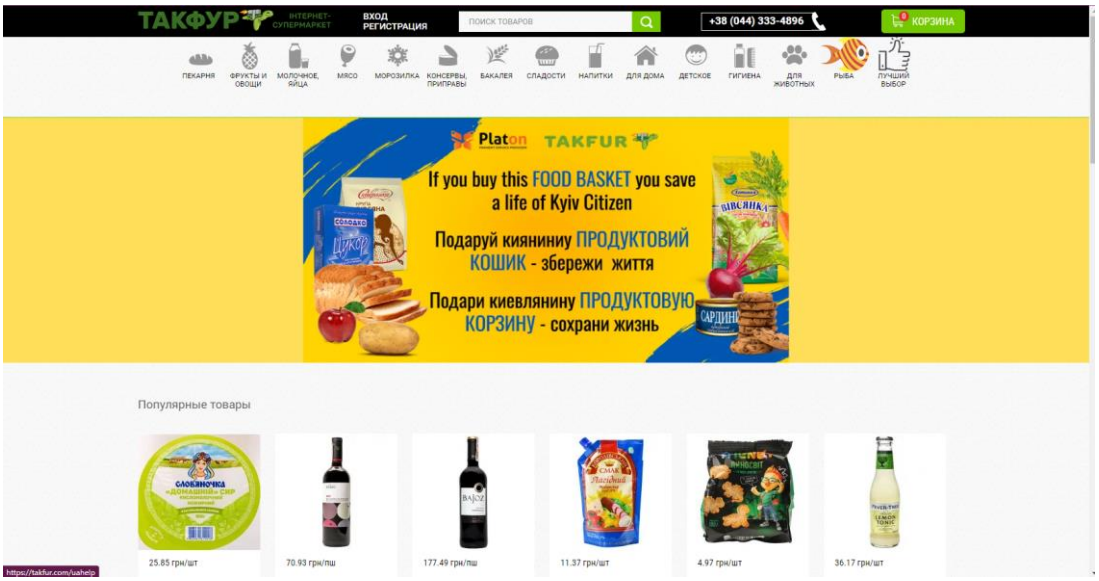


Рис. 1.2 Головне меню веб-додатку «Такфур»

Однак, поточний веб-додаток має обмежений вибір продуктів і страв, що може не задовольнити всі потреби користувачів. Дизайн сайту також потребує покращення, оскільки його інтерфейс не є достатньо привабливим і сучасним.

Переваги:

– Проста авторизація та реєстрація через номер телефон дозволяє користувачам швидко та зручно отримати доступ до системи;

– Оптимізований сайт, який швидко працює та ефективно підбирає товари, забезпечуючи комфортне користування платформою.

Недоліки:

– Наявність жахливої візуальної частини та неякісно розташованих віджетів може відволікати користувачів та зменшувати задоволення від використання платформи;

– Відсутність чат-бота ускладнює процес комунікації з підтримкою та може збільшувати час очікування відповіді на запитання;

– Відсутність фільтрації за конкретними параметрами товару обмежує можливості користувачів у пошуку необхідних продуктів;

– Відсутність трекінгу відслідковування товару ускладнює контроль над статусом замовлення та може призвести до незручностей у випадку доставки.

1.2.3 Веб-Додаток Glovo

Glovo - міжнародна служба доставки, яка діє в багатьох країнах, зокрема в Україні. Glovo надає багато різноманітних можливостей замовляти їжу з ресторанів, але й продукти з супермаркетів, ліки, квіти та інших видів товарів.

Користувачі можуть вибирати серед широкого асортименту ресторанів і магазинів, доступних у їхньому районі який веб-додаток може запропонувати. Також Glovo пропонує функцію відстеження замовлення в режимі реального часу, що дозволяє бачити, де знаходиться кур'єр і коли прибуде доставка до місця призначення, яке вказав користувач, за допомогою чого надає користувачам можливість добре розрахувати свій час.

В Glovo можна виділити такі переваги та недоліки:

– Glovo має дуже привабливий та зручний користувацький інтерфейс. В якому дуже добре та оптимізовано розроблена анімація віджетів, які також добре розташовані на сторінці;

– Glovo оптимізований під мобільні пристрої, також має власний мобільний застосунок, за допомогою, якого користувач може легко та з любого місця замовити собі смачненькі страви не використовуючи ПК;

– Glovo має дуже добре опрацьовану пошукову систему, де користувач може знайти, легко улюблений заклад або вже по готовим запитам, які частіше за всього задають користувачі. Також головним аспектом є те, що користувач може

робити пошук продуктів в окремо вибраному закладі, який він обрав через веб-додаток;

– Головна особливість, що робить Glovo відомою, - це широкий вибір закладів, серед яких користувач може знайти свій улюблений;

– Glovo має дуже добру користувацьку підтримку, де користувач може спокійно оформити заявку через їхній веб-додаток та відправити на трудовлаштування до їхньої компанії, щоб допомагати її в розвитку.

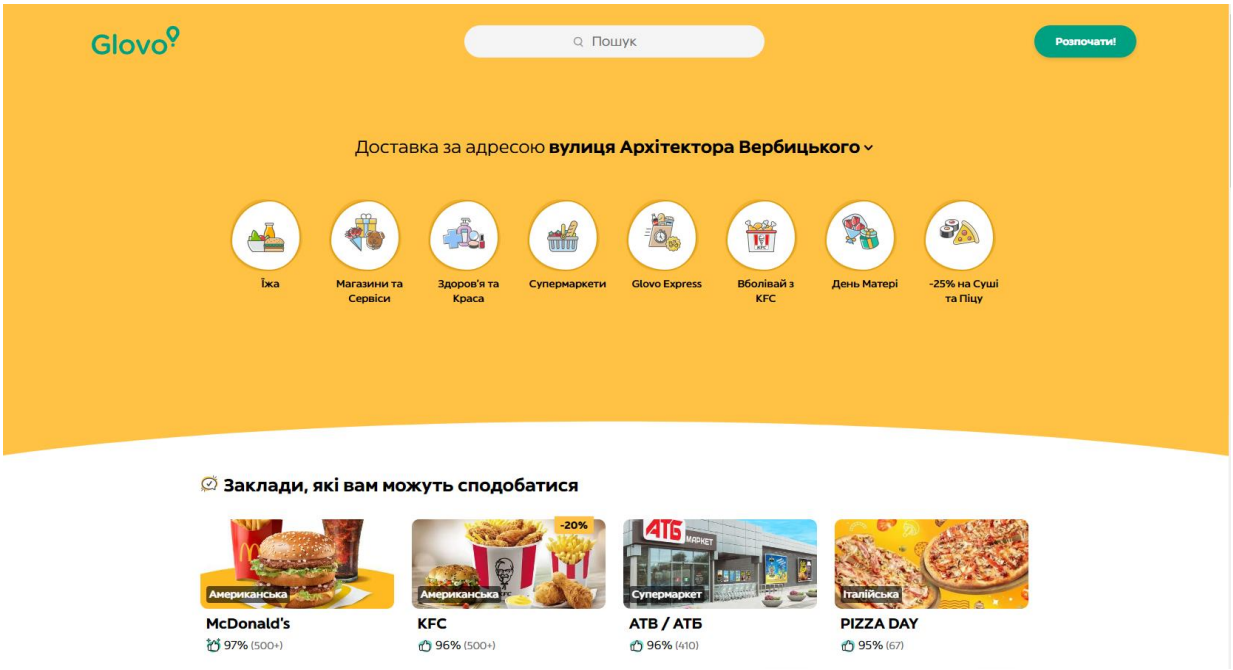


Рис. 1.3 Головне меню веб-додатку «Glovo»

Недоліки:

– Відсутність чат-бота для користувачів. Glovo не надає чат-бота, який може би відповідати на запитання користувачів або надавати необхідну підтримку через автоматизовані повідомлення;

– Неефективний функціонал кнопки перекладача тексту. У додатку Glovo проблеми з функціоналом кнопки перекладача тексту. Багато слів не перекладаються належним чином, що може призвести до нерозуміння та недоліків у спілкуванні між користувачами та сервісом.

Зведені результати аналізу характеристик розглянутих додатків наведено у таблиці 1.1.

Таблиця 1.1

Зведені результати аналізу характеристик веб-додатків

Показник	**Glovo**	**Zakaz.ua**	**Такфур**	**ReDelivery**
Система відгуків	-	+	-	+
Можливість замовлення без реєстрації на сайті	+	-	-	+
Наявність списку улюбленого	-	+	-	+
Наявність чат-бота	-	+	-	+
Можливість залишити чайові	-	-	-	+
Швидкість завантаження сторінки	549 ms	1800 ms	1080 ms	435ms

1.3 Огляд використаних програмних засобів для реалізації

1.3.1 Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) [1] – є безкоштовним середовищем розробки коду, яку заснувала компанія від Microsoft. VS Code отримала свою популярність у використанні завдяки своїй легкості, швидкості та широкому спектру функцій, які роблять його унікальним інструментом для розробників. А можна виділити такі особливості VS Code:

– Менше займає місце на диску та швидкісний у роботі. VS Code займає дуже малу кількість пам’яті на диску і дуже ефективно працює навіть на дуже слабких ПК, що робить його ідеальним вибором для розробників;

– Підтримка великої кількості мов програмувань. VS Code підтримує велику кількість мов програмування, а саме можна виділити: Python, HTML, CSS, Java, Java Script та багато інших, які існують на сьогоднішній день. Через те VS Code забезпечує гнучкість прі створюванні проектів;

– Велика кількість плагінів. VS Code має дуже велику кількість плагінів, за допомогою яких розробник може налаштовувати власне середовище під себе. Крім цього VS Code має великий каталог розширень, які додають нові можливості та інструменти до середовища;

– Можливість інтегрувати з Git. VS Code має вбудовану підтримку з Git, що легко дозволяє керувати різними версіями коду та виконувати швидко коміти, переглядати історію змін і працювати з гілками, притому що це все писати власноруч через термінал не обов'язково;

– Мультиплатформеність. VS Code можна використовувати на різних платформах, а саме: Windows, macOS та Linux;

– Інтелектуальне автодоповнення. Має дуже потужне розширення IntelliSense. Дане розширення допомагає розробнику швидко писати код автоматично доповнюючи його і також надає підказки розробнику під час написання код, крім цього він має документацію API, що значно підвищує ефективність і точність розробки.

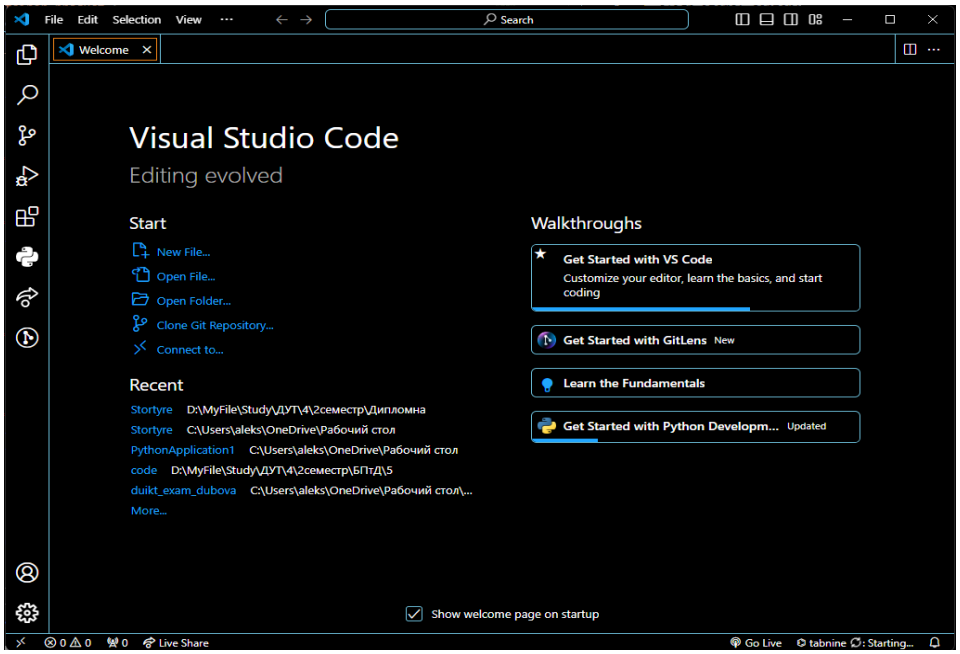


Рис. 1.4 Інтерфейс середовища розробки «Visual Studio Code»

У висновку VS Code на сьогоднішній є дуже потужним, ефективним та гнучким розширенням для середовище, який надає велику можливість для створення проектів. Через те було обрано VS Code для створення веб-додатку для ефективного створення потужного застосунку за допомогою даного середовище.

1.3.2 Мова програмування HTML

HTML [4] (Hyper Text Markup Language) – стандартна мова розмітки для створення веб-сторінок. HTML визначає структуру веб-сторінки, використовуючи теги в коді, які включають в себе різні атрибути, щоб формувати текст, зображення, відео та інші видів елементів на веб-сторінці.

За допомого HTML тегів можливо створити на веб-сторінці основну структуру сторінки, а саме: заголовки, таблиці, списки, підзаголовки та інші блоки. А теги позначають таким чином: <html>, <body>, <p>, та ін. Також теги містять в собі атрибути, які надають додаткову інформацію про дані елементи. Атрибути позначаються так: href, src, alt.

Але основною особливістю HTML виділяється гіпертекстом. Він дозволяє створювати гіперпосилання на інші різні веб-сторінки, що робить можливим навігацію в Інтернеті.

Також має сумісність з CSS та JavaScript. HTML дуже легко і швидко можна інтегрувати з CSS для створення візуальної частини сторінки для гарної її відображення, а з JavaScript надає до сторінки можливість додавати динамічні функції.

HTML являється стандартом, підтримуваним усіма основними веб-браузерами, що забезпечують коректне відображення веб-сторінки, незалежно, який пристрій або платформу користувач буде використовувати.

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/modern-normalize/2.0.0/modern-normalize.min.css"
    integrity="sha512-4xo8b1KMVCiXpTaLzQSLSw3KFOVPWhm/TRtuPVc4WG6kUgjH6J03IBuG7JZPkcWMxJ5huwaBpOpnwYE1P/m6wg=="
    crossorigin="anonymous" referrerpolicy="no-referrer" />
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
    href="https://fonts.googleapis.com/css2?family=Sarabun:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;1,100;1,200;1,30
    rel="stylesheet">
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="../css/menu.css">
  <title>Menu</title>
</head>
```

Рис. 1.5 Шматок коду написаний на мові HTML

1.3.3 Мова програмування CSS

CSS [5] (Cascading Style Sheets) – мова стилів, яка використовується для створення візуального вигляду, які реалізовані мовами розмітки, такими як HTML. CSS дозволяє розробникам змінювати вигляд веб-сторінок без впливу на їх структуру. Можна виділити такі особливості CSS:

– За допомогою CSS можливо створювати стилі для елементів HTML для їхнього гарного вигляду. А саме що можна налаштувати це: кольори, шрифти, відступи, межі, тіні та багато іншого. В результаті чого ми отримуємо дуже гарні та налаштовані елементи HTML, замість стандартного вигляду елементу;

– Також CSS надає дуже великий набір селекторів, що дозволяє точно вибрати елемент HTML та налаштувати його. Тобто ми можемо вказати назву в елементі HTML class або id, за допомогою чого зможемо дуже точно налаштувати той необхідний елемент, який нам потрібен, замість всіх;

– CSS надає унікальність написання медіа-запитів. За допомогою них розробник, може створювати адаптивні сторінки, які при зміні розширення екрану пристрою або зміні розміру самої веб-сторінки змінює свій візуальних вигляд і елементів під кожний розмір пристрою;

– CSS дозволяє створювати різні якісні анімації та переходи між станами елементів, що додає динаміки і додаткову привабливість веб-сторінкам;

– Також CSS має дуже зручні та потужні сучасні технології для створення макетів, а саме Flexbox та Grid, що полегшує розробникам розміщення

елементів на сторінці у складних структурах;

– CSS дозволяє використовувати зовнішні стилі, зберігаючи їх у окремих файлах. Це сприяє організованості коду, дозволяє повторне використання стилів і полегшує їхнє оновлення.

```css
html {
  padding: 0;
  margin: 0;
}

body {
  font-family: "Sarabun", sans-serif;
  line-height: normal;
  font-style: normal;
  font-weight: 400;
  font-size: 16px;
  line-height: 1.5;
  letter-spacing: 2%;
  color: #434455;
  background-color: var(--Main, #F6F1EE);
  margin: 0;
  padding: 0;

```

Рис. 1.6 Демонстрація коду написаний на CSS

На сьогоднішній день CSS є ключовою технологією для створення привабливою візуальної частини веб-сторінки і дозволяє легко керувати зовнішнім виглядом веб-сторінки і створювати адаптивний дизайн, що робить його доступним та зручним для користувачів на різних пристроях.

1.3.4 Мова програмування JS

JavaScript [3] (JS) - динамічна мова програмування, призначена для створення інтерактивних і функціональних веб-сторінок. Як одна з ключових технологій веб-розробки, вона забезпечує чудову динамічну взаємодію з користувачами. JavaScript виділяється такими характеристиками:

– JavaScript зазвичай виконується на стороні клієнта, тобто веб-браузера, що дозволяє створювати швидкі інтерфейси без необхідності постійного звернення до сервера;

– JavaScript дозволяє створювати функції, такі як перевірка форм, динамічне оновлення контенту, анімації, спливаючі вікна та багато іншого;

– JavaScript має багатий набір вбудованих функцій та об'єктів, які дозволяють працювати з текстом, числами, датами, масивами, а також маніпулювати елементами DOM [6] (Document Object Model). DOM – програмний інтерфейс для веб-документів, який представляє веб-сторінку. Він дозволяє JavaScript і іншим мовам програмування динамічно змінювати елементи та їхній вміст, забезпечуючи інтерактивність і динамічну взаємодію з користувачами;

– JavaScript можна дуже легко інтегрувати з HTML та CSS, а також може взаємодіяти з іншими різними видами мовами програмування та технологіями за допомогою API та веб-сервісів;

– JavaScript підтримує асинхронні операції через функції зворотного виклику, команд async/await, що дозволяє виконувати завдання у фоновому режимі без блокування головного потоку. Це дуже важливо для операція, що забирають велику кількість часу, типу таких як завантаження даних з мережі або доступ до баз даних, оскільки він дозволяє користувацькому інтерфейсу залишатися швидким і чуйним;

– JavaScript має дуже велику кількість різноманітних фреймворків і бібліотек, таких як React, Angular, Vue.js, jQuery, які спрощують розробку складних веб-додатків та знижують обсяг коду, що потрібно писати вручну і допомагає розробникам швидко реалізувати власний продукт.

```
'use strict';

import { pizza, burgers, pita } from './menu-list.js';
import * as mobileMenu from '../js/mobile-menu.js';

const menu = document.querySelector('.menu-list');
const cartButton = document.querySelector('.cart-container');
const cartPcButton = document.querySelector('.cart-button');
const cartCounter = document.querySelector('.cart-counter');
const cartCounterMobile = document.querySelector('.cart-counter-mobile');
const backdrop = document.querySelector('.backdrop');
const modalMenuCloseBtn = document.querySelector('.close-modal-menu-btn');
const countPlusBtn = document.querySelector('.modal-count-plus-btn');
const countMinusBtn = document.querySelector('.modal-count-minus-btn');
const itemsCounter = document.querySelector('.counter');

let favStorage = localStorage.getItem('fav');
let favArray = JSON.parse(favStorage);
let addToCartBtn = document.querySelector('.modal-menu-add-button');
let counter = 1;
```

Рис. 1.7 Демонстрація коду написаний на JS

JavaScript є незамінним інструментом для веб-розробників, дозволяючи створювати інтерактивні, динамічні та зручні веб-сторінки.

1.3.5 GitHub

GitHub [10] — це веб-платформа для хостингу та спільної розробки програмного забезпечення, заснована на системі контролю версій Git. GitHub надає як безкоштовні, так і платні послуги для користувачів, а також має ряд функцій, які полегшують процес розробки та спільного використання коду.

GitHub був заснований у 2008 році Томом Престон-Вернером, Крісом Ванстретом, Пі Джей Хайеттом та Скоттом Чаконом. Платформа швидко стала популярною серед розробників та компаній завдяки зручності використання та потужним можливостям для спільної роботи. У 2018 році GitHub був придбаний корпорацією Microsoft за $7.5 мільярдів.

Основні функції та можливості GitHub:

– GitHub дозволяє користувачам створювати репозиторії, де зберігається вихідний код проєктів. Репозиторії можна зробити публічними (доступними для всіх користувачів) або приватними (для обмеженої кількості користувачів).

– GitHub заснований на Git, що дозволяє відстежувати зміни в коді, працювати з гілками, об'єднувати їх та повертатися до попередніх версій, якщо це необхідно. Це особливо корисно для командної роботи, оскільки кожен розробник може працювати над своєю гілкою, не заважаючи іншим.

– GitHub надає інструменти для спільної роботи, такі як Pull Request (запит на внесення змін), Issues (проблеми або завдання) та Projects (дошки для управління проєктами). Ці функції допомагають командам організувати робочий процес, обговорювати ідеї та відстежувати прогрес.

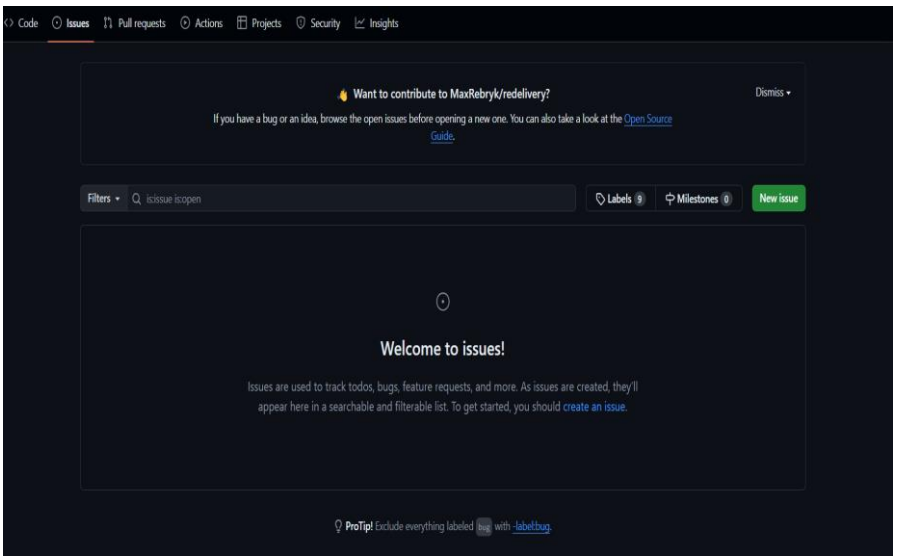


Рис. 1.8 Інструмент GitHub «Issues»

– GitHub дозволяє налаштовувати права доступу для різних користувачів та команд, що забезпечує безпеку коду та даних. Розробники можуть запрошувати співробітників, обмежувати доступ до приватних репозиторіїв та використовувати двофакторну автентифікацію для додаткового захисту.

– За допомогою GitHub Actions розробники можуть автоматизувати різні завдання, такі як тестування коду, деплой на сервер або оновлення документації. Інтеграція з системами безперервної інтеграції та безперервної доставки (CI/CD) дозволяє прискорити цикл розробки та покращити якість продукту.

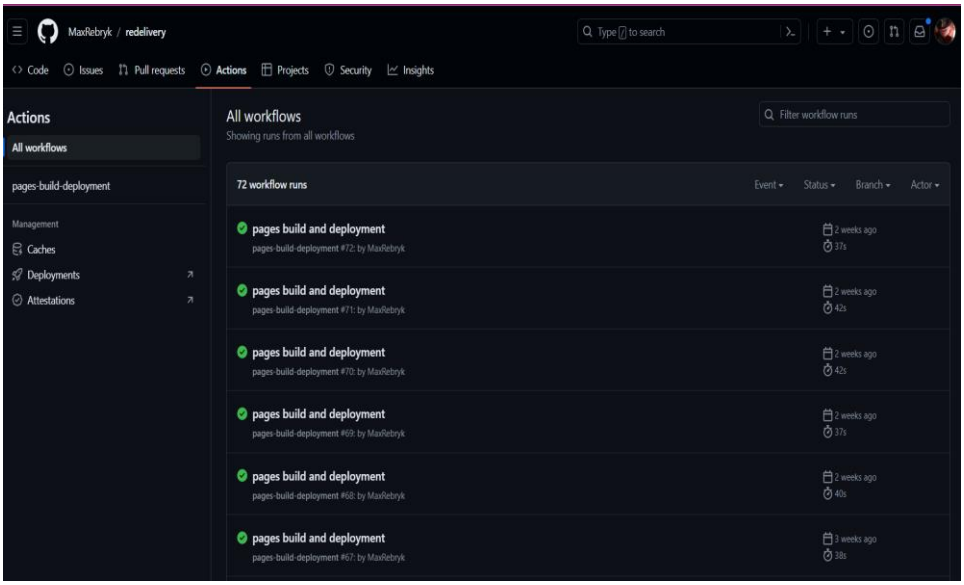


Рис. 1.9 GitHub Actions

– GitHub є домом для мільйонів проєктів з відкритим вихідним кодом, де розробники з усього світу можуть вносити свій вклад, використовувати чужі напрацювання та ділитися своїми проєктами. Це створює потужну спільноту, яка сприяє розвитку технологій та обміну знаннями.

– GitHub Marketplace пропонує безліч інтеграцій та додатків, які розширюють функціональність платформи. Це можуть бути інструменти для аналізу коду, управління завданнями, моніторингу та багато іншого.

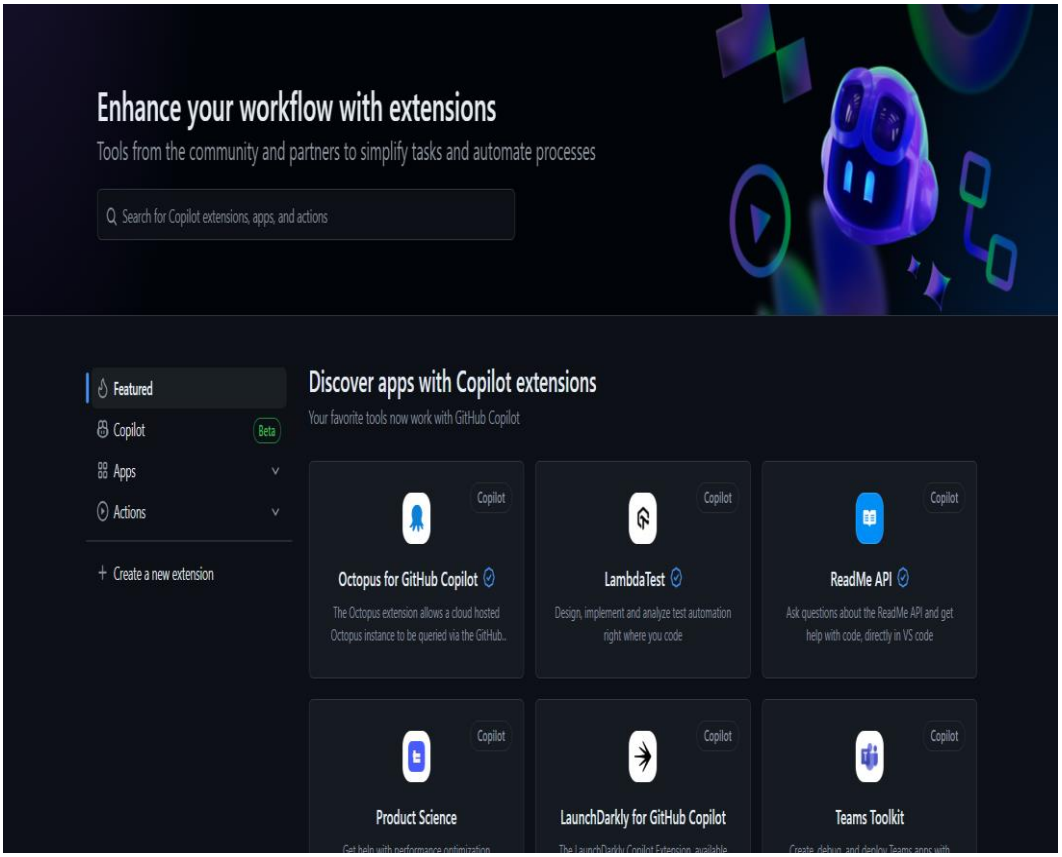


Рис. 1.10 GitHub Marketplace

Приклади використання GitHub:

– Розробники часто використовують GitHub для зберігання та демонстрації своїх проєктів потенційним роботодавцям.

– Компанії та команди розробників використовують GitHub для спільної роботи над проєктами, відстеження прогресу та управління завданнями.

– GitHub використовується в освітніх цілях для навчання студентів програмуванню, а також для менторства та спільної роботи в навчальних

проєктах.

GitHub став невід'ємною частиною екосистеми розробки програмного забезпечення, надаючи потужні інструменти та можливості для розробників усіх рівнів.

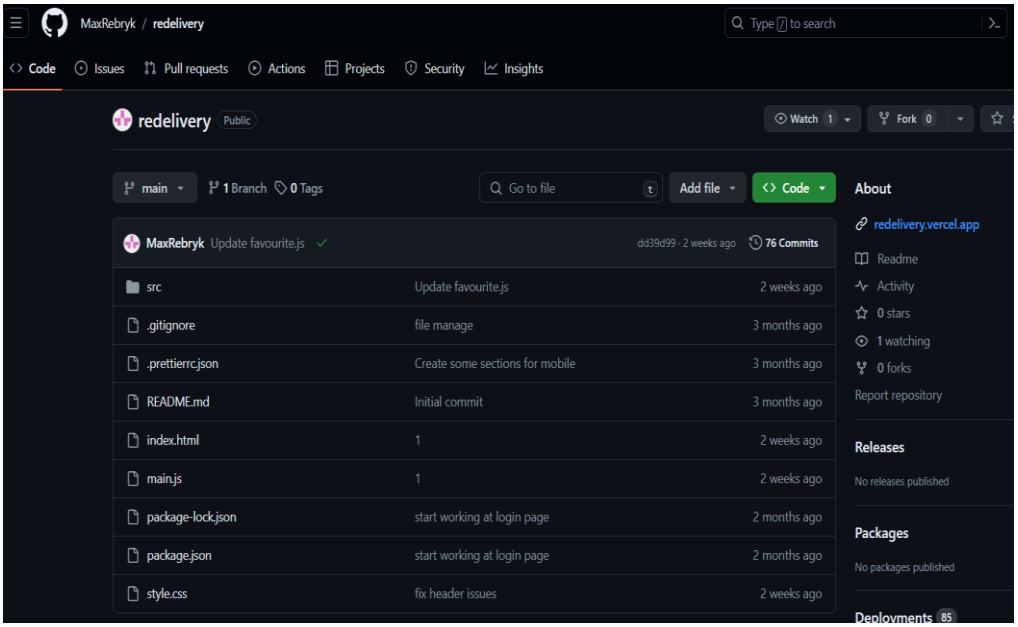


Рис. 1.11 Демонстрація GitHub

1.3.6 Git

Git [9] — це розподілена система керування версіями, яка використовується для відстеження змін у вихідному коді під час розробки програмного забезпечення. Git був створений Лінусом Торвальдсом у 2005 році для підтримки розробки ядра Linux, і з того часу став найпопулярнішою системою контролю версій у світі.

У 2005 році Лінус Торвальдс створив Git як відповідь на втрату доступу спільноти розробників Linux до системи контролю версій BitKeeper. Git став швидкою, ефективною та розподіленою системою, що здобула широку популярність серед розробників програмного забезпечення.

Основні особливості та принципи роботи Git:

– GitHub використовується в освітніх цілях для навчання студентів програмуванню, а також для менторства та спільної роботи в навчальних

проєктах.

– Git оптимізований для високої швидкості. Операції, які зазвичай займають більше часу в централізованих системах, у Git виконуються миттєво, оскільки вони виконуються локально.

– Git забезпечує цілісність даних за допомогою використання SHA-1 хешування для кожного файлу та коміту. Це означає, що будь-яка зміна даних призведе до зміни хешу, що дозволяє виявляти будь-які порушення цілісност.

– Git відстежує зміни файлів на основі знімків (snapshots), а не різниць (deltas). Кожного разу, коли ви комітите зміни, Git зберігає знімок всього проєкту. Якщо файли не змінювалися, Git зберігає посилання на попередні версії файлів.

– Git дозволяє створювати та працювати з гілками легко і швидко. Гілки в Git легкі та дешеві, що заохочує розробників створювати нові гілки для кожної нової функції або виправлення. Злиття гілок також ефективне і добре підтримується.

– Git добре масштабується і підходить як для невеликих проєктів, так і для великих з відкритим вихідним кодом.

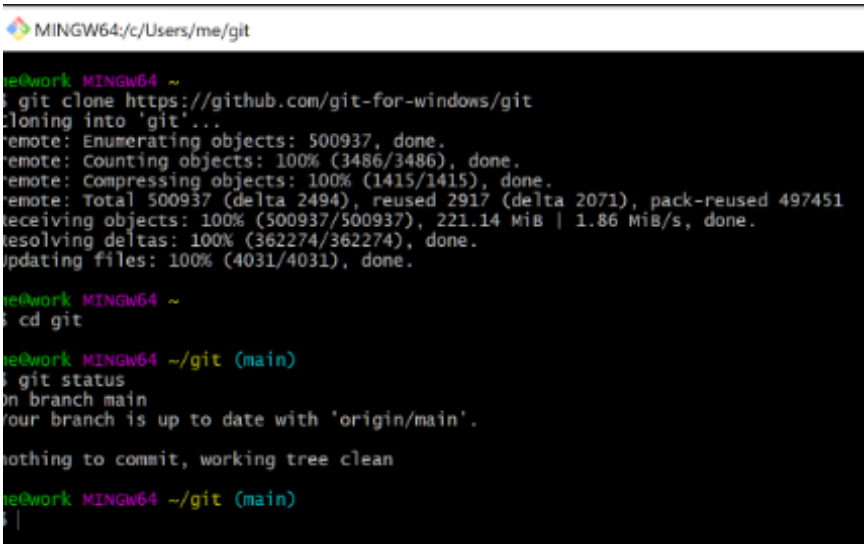


Рис. 1.12 Демонстрація команд GIT

Git є потужним інструментом для керування версіями, який надає розробникам можливість ефективно співпрацювати, відстежувати зміни та

зберігати цілісність проєктів. Завдяки своїй розподіленій природі та високій швидкості, Git став стандартом де-факто в світі розробки програмного забезпечення.

Принципи роботи з Git:

– Кожен коміт повинен містити лише одну логічну зміну або виправлення. Це полегшує відстеження змін та виявлення проблем.

– Розробники повинні комітити часто, щоб мінімізувати ризик втрати даних та спростити злиття змін.

– Коментарі до комітів повинні бути інформативними та чіткими, щоб інші розробники могли легко зрозуміти суть змін.

– Кожна нова функція або виправлення повинні розроблятися в окремій гілці. Після завершення розробки та тестування, гілка зливається з основною гілкою.

1.4 Об’єкт, предмет, мета та задачі роботи

Мета роботи – покращення процесу замовлення доставки їжі за рахунок застосування веб-додатку, створеного на мовах програмування HTML, CSS, JS та їхні фреймворки.

Об’єкт дослідження – процес служби доставки їжі.

Предмет дослідження – веб-застосунок для служби доставки їжі.

Основні задачі роботи:

– Проаналізувати існуючі аналоги веб-додатків для служб доставок їжі та порівняти їх.

– Створити функціональні і нефункціональні вимоги до веб-додатку.

– Зробити дослідження та вибрати засоби розробки для веб-додатку служби доставки їжі та розробити модель архітектури веб-додатку.

– Розробити веб-додаток служби для доставки їжі на основі обраних інструментів та визначених вимог.

– Провести тестування веб-додатку.

2 ПРОЕКТУВАННЯ WEB-ЗАСТОСУНКУ

2.1 Проектування структури веб-додатку служби доставки їжі

Дана структура веб-додатку [6] для служби доставки їжі була створена та зображена за допомогою draw.io. За допомогою цієї структури ми зобразимо, як будуть пов’язані між собою веб-сторінки веб-додатку для служби доставки їжі.

Для реалізація структури веб-додатку для служби доставки їжі робилася за методом ієрархічної. Ця структура передбачає вкладеність однієї категорії в іншу при цьому для кожної окремої послуги чи продукту формується гілка.

Головна перевага ієрархічної структури є те, що вона допомагає користувачам набагато краще орієнтуватися на нашому веб-сайті та дозволяє їм швидко знайти необхідну інформацію.

На рисунку 2.1, продемонстровано ієрархічну структуру веб-додатку для служби доставки їжі.

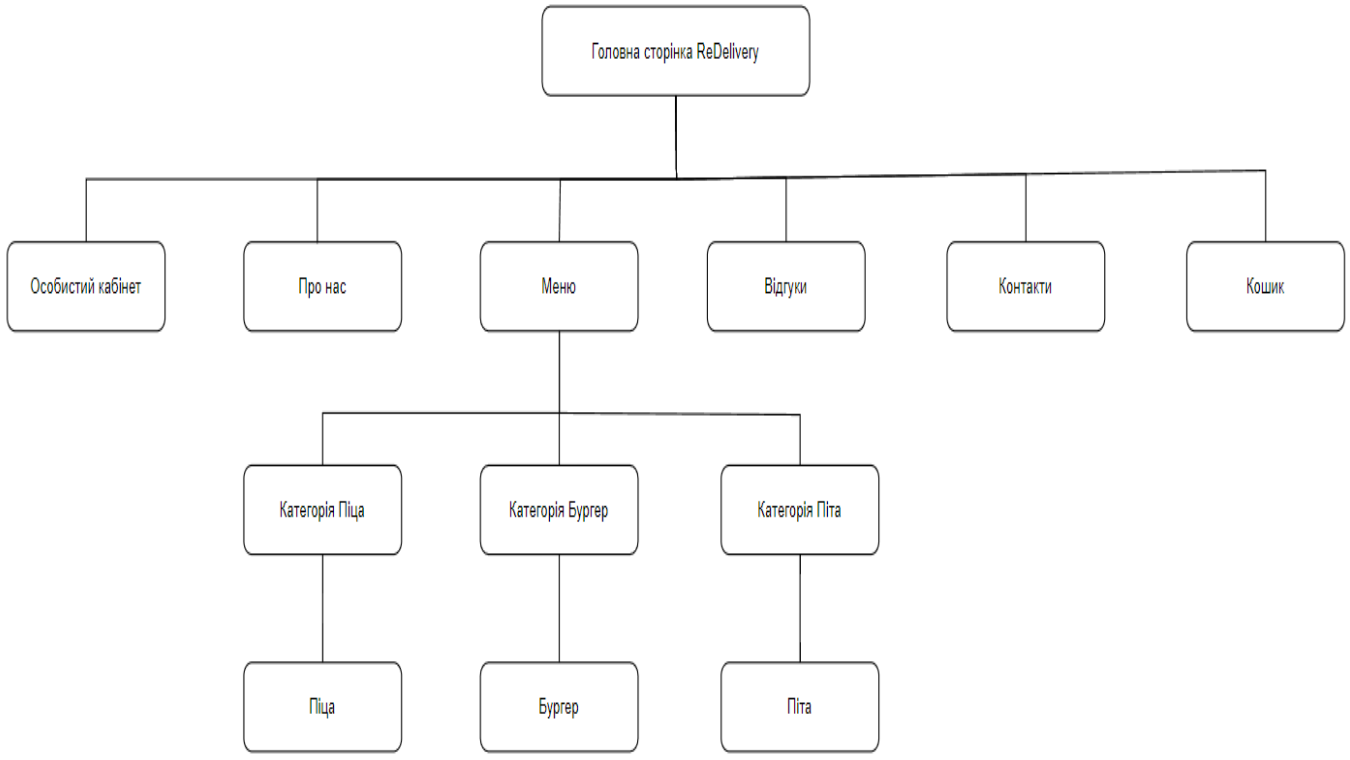


Рис. 2.1 Ієрархічна структура веб-додатку для служби доставки їжі

На даній структурі, зображено, як користувач може з головної сторінки потрапити на такі, як: Особистий кабінет, Про нас, Меню, Відгуки, Контакти, Кошик.

Також, якщо користувач перейде на сторінку меню, він з цієї сторінки може легко перейти до наступної з трьох наведених категорій: Категорія піца, Категорія бургер, Категорія піта. І відштовхуючись від цих категорій, він може обрати необхідний продукт, який його цікавить і перейти на його сторінку.

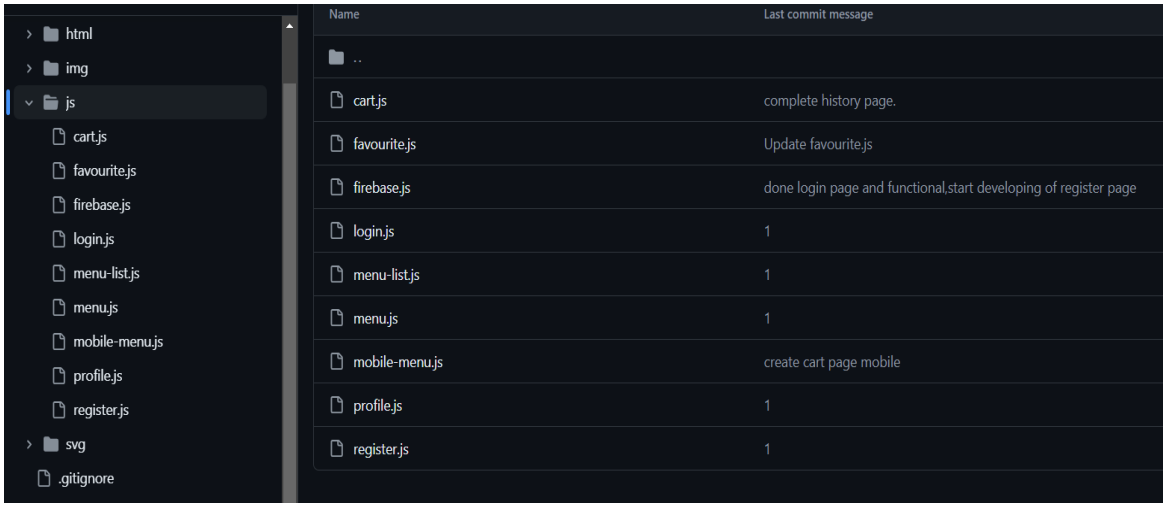


Рис. 2.2 Файлова структура

2.2 Моделювання вимог веб-додатку для служби доставки їжі.

Після детального аналізу аналогічних веб-додатків для служб доставки їжі, виявлення їх недоліків та запозичення корисних аспектів, були сформульовані наступні функціональні та нефункціональні вимоги, за допомогою яких можна реалізувати якісний та оптимізований веб-продукт.

Функціональні вимоги:

– Можливість залишати відгуки - можливість ділитися своїм враженням та досвідом з іншими клієнтами, допомагаючи їм зробити більш обдуманий вибір.

– Панель зворотного зв’язку - можливість ділитися своїм враженням та досвідом з іншими клієнтами, допомагаючи їм зробити більш обдуманий вибір.

– Кнопка повторного замовлення - забезпечує зручність та ефективність покупок, оскільки вона дозволяє клієнтам швидко та легко відновити попередні

замовлення без необхідності повторного пошуку або вибору товарів.

– Можливість додавати улюблені страви до списку бажаного - для створення зручного і персоналізованого досвіду покупок, оскільки вона дозволяє клієнтам відокремлювати товари, які їх зацікавили, для подальшого огляду та покупки.

– Реєстрація та авторизація – користувачі можуть створювали власний кабінет, безпечно входити в систему та безпечно керувати своїми даними та відслідковувати свої замовлення.

Нефункціональні функції:

– Стійкість до навантаження – здатність системи витримувати навантаження від час періодів пікового попиту втрати продуктивності або недоступності сервісу.

– Система відстеження стану замовлення – дозволяє користувачам в реальному часі переглядати поточний статус замовлень, починаючи від моменту підтвердження замовлення до його доставки, що забезпечує прозорість.

– Збереження замовлень – користувачі можуть в особистому кабінеті переглядати свої минулі зроблені замовлення та при необхідності зробити повторне замовлення.

2.3 Проектування структури бази даних веб-додатку

Структуру бази даних для веб-додатку служби доставки їжі було розроблено за допомогою draw.io. В цій структурі відображено всі таблиці, їхні атрибути та взаємозв'язки між ними, що забезпечує повне розуміння даних та їх інтеграцію у систему.

Схема бази даних - це логічне представлення структури даних у базі даних. Вона визначає структуру таблиць, їх взаємозв'язки та обмеження цих взаємозв'язків. Основними компонентами схеми бази даних є таблиці, поля (стовпці), відносини (зв'язки між таблицями), ключі (первинні, зовнішні), індекси та обмеження цілісності даних.

На рисунку 2.2 буде зображено структуру бази даний до веб-додатку для служби доставки їжі.

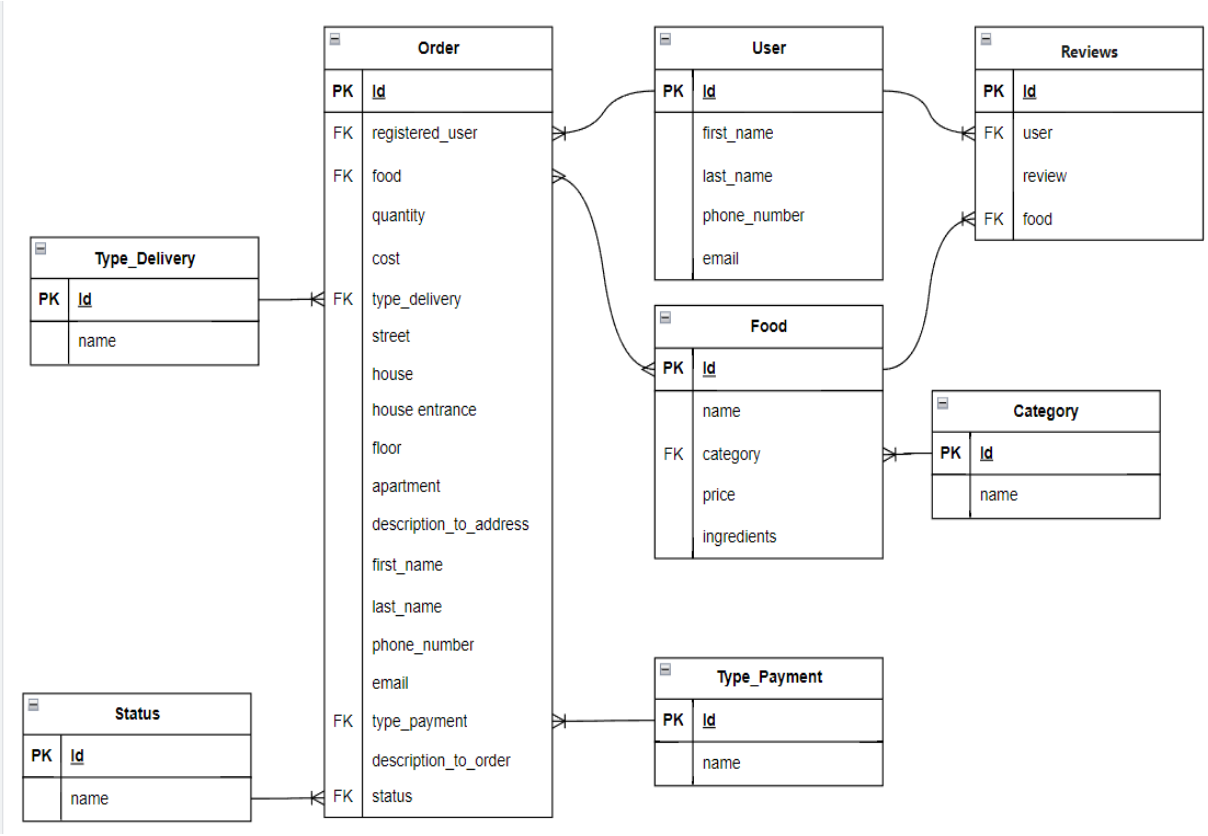


Рис. 2.3 Ієрархічна структура веб-додатку для служби доставки їжі

Дана структура база даних до веб-додатку для служби доставки їжі, містить такі таблиці:

– Order – дана таблиця містить собі дані, про залишені замовлення користувачів, як зареєстрованих та й не зареєстрованих і яка також має багато зв'язків між різними таблицями.

– User – ця таблиця містить дані про зареєстрованих користувачів, які зареєструвалися на веб-додатку. Має зв’язок з таблицями Reviews та Order, один-до-багатьох.

– Food – ця таблиця містить дані про страви, які відображаються на веб-додатку. Має зв’язок з таблицею Order - багато-до-багатьох та таблицями Reviews та Category – один-до-багатьох та багато-до-одного.

– Reviews – ця таблиця містить дані всі відгуків зареєстрованих

користувачів на конкретну страву. Має зв’язок з таблицями User та Food, багато-до-одного.

– Category – ця таблиця містить всіх способів можливих категорій страв. Має зв’язок з таблицею Food, один-до-багатьох.

– Type_Payment – ця таблиця містить інформацію про тип способів оплати замовлень. Має зв’язок з таблицею Order, один-до-багатьох.

– Type_Delivery – ця таблиця містить інформацію всіх видів способів отримання замовлення. Має зв’язок з таблицею Order, один-до-багатьох.

– Status – ця таблиця містить інформацію про всі видів статусів, яке може мати замовлення. Має зв’язок з таблицею Order, один-до-багатьох.

2.4 Проектування інтерфейсу користувача

2.4.1. Основні вимоги до юзабіліті веб-додатку для служби доставки їжі

Юзабіліті сайту – визначає наскільки зручно розроблений інтерфейс для користувача. Тобто визначає загальний ступінь зручності веб-сторінки при використанні користувачем.

Для створення веб-додатку для служби доставки їжі були поставлені такі основні вимоги юзабіліті:

– Швидке завантаження сторінок.

– Оптимізація під мобільні пристрої.

– Однаковий стиль всіх веб-сторінок.

– Заповнення кожної веб-сторінки, щоб користувачі розуміли тематику відвіданого ними відвіданої сторінки.

– Легкий спосіб закриття діалогових вікон, які спливають.

– Налагодження сторінки 404.

– Розробити спливаючі підказки, якщо кнопка немає назви, то при наведенні курсором миші, щоб відображалася підказка з підписом кнопки її дії.

– Додати невелику анімацію до кнопок, для їх привабливості та динамічності.

2.4.2. Реалізація діаграми активності для веб-додатку.

Діаграма активності або (task flow), використовується для реалізації візуалізації різних наявних процесі, завдань і рішень, які користувач виконує для досягнення конкретної мети в веб-додатку. Вона допомагає зрозуміти і оптимізувати взаємодію користувачів з веб-додатком.

Основні елементи діаграми:

– Завдання. Описує дії або кроки, які користувач виконує. Кожне завдання може бути представлене у вигляді окремого блоку чи вузла на діаграмі.

– Послідовність. Вказує на порядок виконання завдань. Стрілки між вузлами демонструють напрямок дій.

– Вибір. Момент, коли користувач повинен зробити вибір або прийняти рішення. Ці вузли можуть розподіляти напрямок завдань на декілька гілок, залежно від прийнятого рішення.

– Початок і кінець. Вузли які позначають на діаграмі, початкову і кінцеву точку процесу.

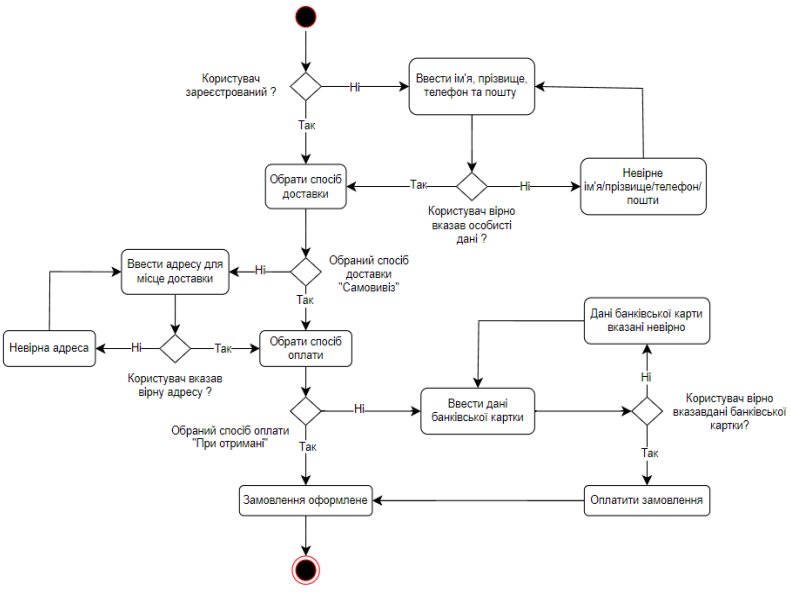


Рис. 2.4 Діаграма активності для веб-додатку для служби-доставки їжі

В даній діаграмі зображено повний процес для успішного оформлення замовлення, від початку до кінця, зі всіма можливими варіантами прийняття рішеннями

3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ СЛУЖБИ ДОСТАВКИ ЇЖІ

3.1 Діаграма класів веб-додатку для служби доставки їжі

Діаграма класів є одним із ключових елементів мов моделювання UML (Unified Modeling Language). Вона використовується для візуалізації структури системи, відображаючи класи, їх атрибути, методи, а також відносини між класами. Це потужний інструмент для об'єктно-орієнтованого проєктування і програмування, який допомагає розробникам зрозуміти та планувати архітектуру системи.

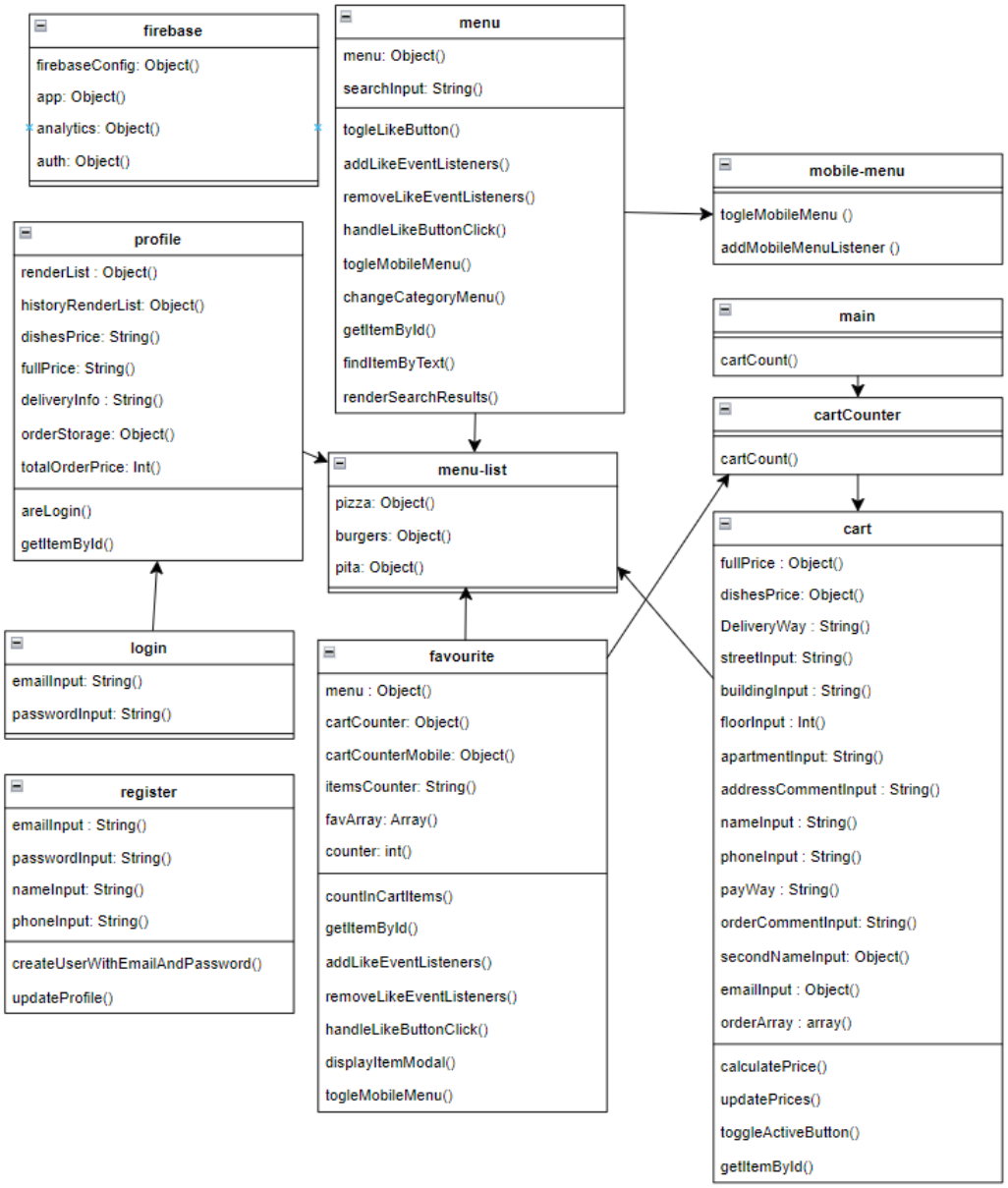


Рис. 3.1 Діаграма класів веб-додатку для служби доставки їжі

Як зображено на рисунку 3.1, зображенні різні класи, які мають власні функції та атрибути. А саме зображенні такі класи: register, main, menu, cartCounter, firebase, favorite, profile, login.

3.2 Специфікація ключових класів веб-додатку для служби доставки їжі

В даному веб-додатку для служби доставки їжі були створені та реалізовані всі класи, які відповідають вимогам до веб-додатку. А саме були реалізовані такі класи:

– register.js – в даному файлі реалізовано функція реєстрації користувача, де користувач вказує всі необхідні дані, який потребує клас. Також в цьому файлі реалізовані функції, які перевіряють, чи за даною поштою вже створений за тією поштою особистий кабінет, якщо ні то аккаунт створюється, а в іншому випадку користувач отримує помилку.

– main.js – даний файл бере дані з іншого класу та передає до головної сторінки, в якої відображаються дані.

– menu.js – даний файл приймає дані та опрацьовує дані всіх наявних страв та доданих товарів до кошику користувачем, крім того також відображаються, дані які додані у список бажаного. Також в даному файлі реалізована функція пошуку, за допомогою чого користувач може шукати ту страву, яку він бажає. Ще реалізовані такі функції, як додавання та прибирання з списку бажаного страв, коли користувач додає до свого списку бажаних страв, щоб в каталозі це також відмітилося, і в протилежному випадку, коли користувач прибирає зі списку бажаного.

– menu-list.js – даний файл, зберігає дані різних видів страв, які включають в собі такі зміни, як: id, назва, ціна, зображення.

– cart.js – даний файл створений, для отримання даних веденим користувачем. Коли користувач вводить особисті дані та адресу доставки, то функція їх спочатку перевіряє, чи вказані вірно дані та відповідають відповідному

формату. Також в цьому файлі розписана окрема функція, де підраховує всю суму товарів, доставку та чаєві, якщо користувач їх залишав. І якщо користувач успішно оформив замовлення, то система зберігає замовлення в сховище.

– cartCounter.js – в даному файлі реалізована функція підрахунку всіх залишених замовлень конкретним користувачем.

– favorite.js – в даному файлі реалізована відображення списку бажаного користувача, в якому він додавав товари. Спочатку він приймає всі необхідні дані про страви, які є в базі даних, а бере дані користувача, які він додав до свого списку, та відображає йому його список на сторінці списку бажаного. Також створена функція додавання та прибирання зі списку бажаного страв, де можна його редагувати в середині. І додатково також, отримує всі необхідні дані про користувача, які необхідно відобразити для нього.

– profile.js – в даному файлі реалізовано функція отримання та відображення всій особистих даних користувача. Також зроблена функція зроблена для отримання всіх замовлень зроблених користувачем та виводить йому на головному профілі. Також реалізована функція для оновлення особистий даних користувача, якщо при необхідності потрібно йому буде змінити. Крім цього зроблена перевірка, перед тим, як користувач захоче зайти до свого кабінету, система перевірить, чи користувач зареєструвався на сайті, якщо ні, то замість відкриття сторінки профілю користувача, відкриється діалогове вікно, де буде запропоновано користувачу зареєструватися на сайті.

– firebase.js – в даному файлі реалізовані вже готові функції для авторизації користувача та надає додаткову безпеку від зловмисників.

– login.js – в даному файлі використовує функції з файлу fatebase.js та використовує їх для авторизації користувача, якщо користувач не вірно вказав дані, то система виведе користувачеві помилку про недостовірність ведених їм дані.

3.3 Опис функціонування веб-додатку з наведеними екранними формами

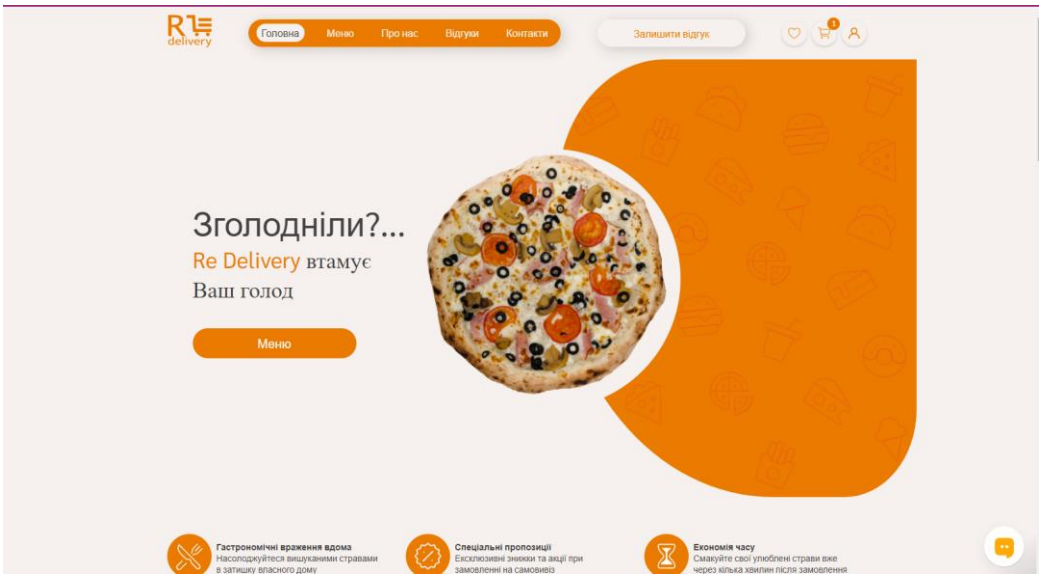


Рис. 3.2 Сторінка веб-додатку «Головне меню»

В даній сторінці зображене головне меню сторінки, звідки користувач може відразу перейти до меню всіх страв натиснувши на кнопку «Меню». Також в даній сторінці описано коротко про саму компанії чим вони спеціалізуються та які послуги надають своїм клієнтам.

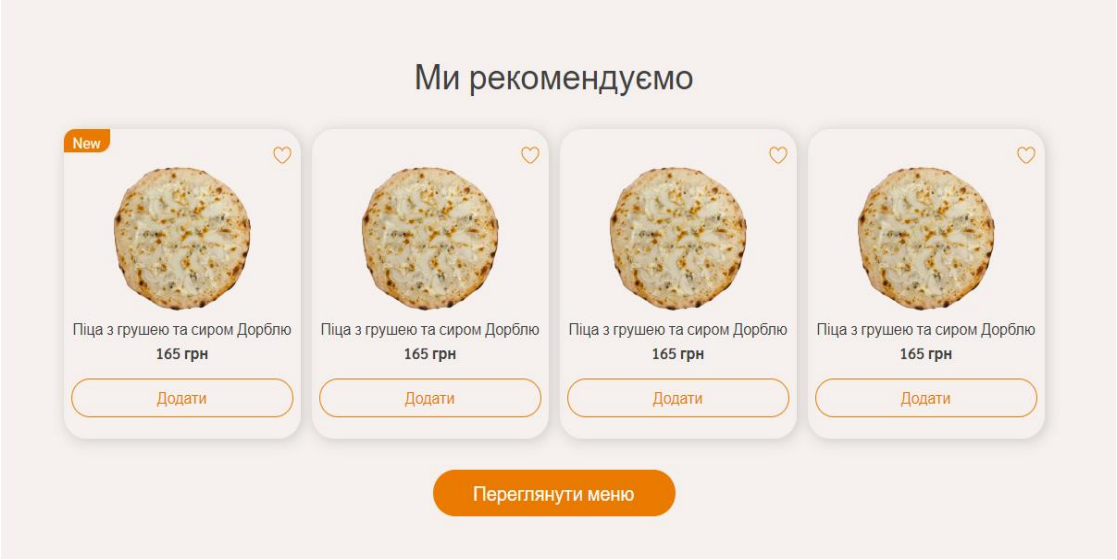


Рис. 3.3 Сторінка веб-додатку «Головне меню» з відображенням різними пропозиціями

Якщо опуститися нижче, то можна побачити різні пропозиції, який сервіс може запропонувати користувачеві, та відразу додати до свого списку бажаного або кошику.

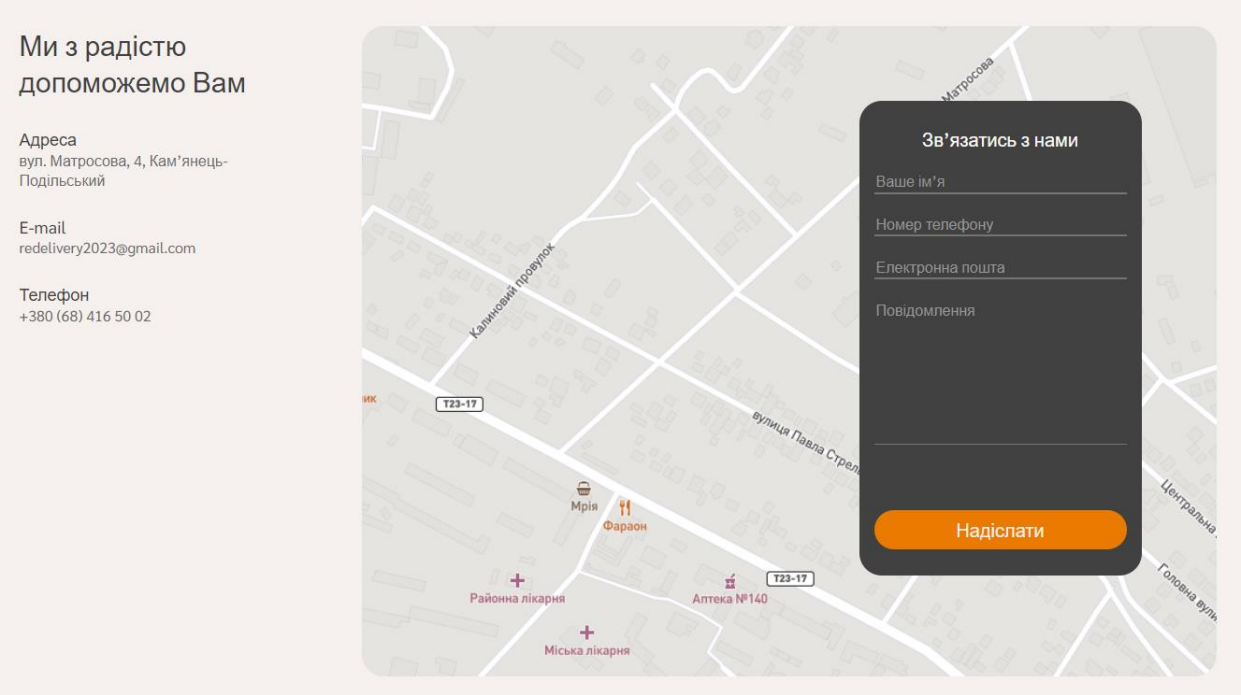


Рис. 3.4 Сторінка веб-додатку «Головне меню» з панеллю зворотного зв'язку.

На той ж самій сторінці, якщо користувач опуститься нижче, то помітить панель для заповнення форми зворотного зв'язку, де користувач, зможе при необхідності, якщо в нього виникли питання або проблеми, вказавши свої дані, щоб служба підтримка мала можливість з ним зв'язатися.

Рис. 3.5 Верхівка всього веб-додатку

На даній верхівці зображено логотип компанії, панель для швидкого переміщення між сторінками, кнопка «Залишити відгук» для того, щоб користувач міг залишити особисті почуття та оцінку за отриманні послуги. Також через цю шапку веб-додатку користувач може перейти до свого особистого кабінету або створити новий профіль, зайти у кошик та за до свого списку бажаного.

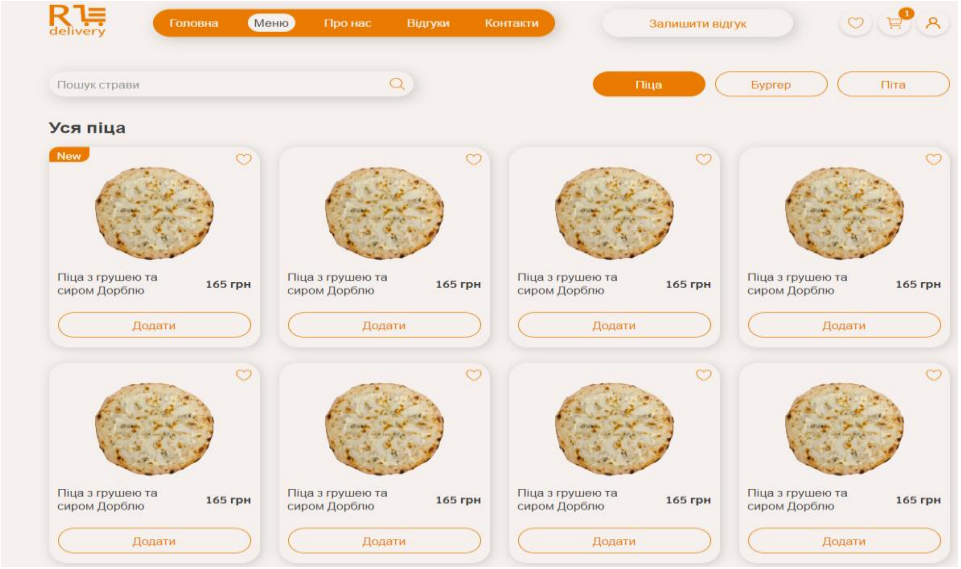


Рис. 3.6 Сторінка веб-додатку «Меню»

На сторінці «Меню» користувач може ознайомити з всіма асортиментом доступних страв, які може запропонувати компанія. Також тут реалізована пошукова система, за допомогою, якою користувач може легко та швидко знайти йому необхідну страву, яку він бажає або відсортувати страви за конкретними категоріями страв, що спростить пошук необхідної страви користувачеві.

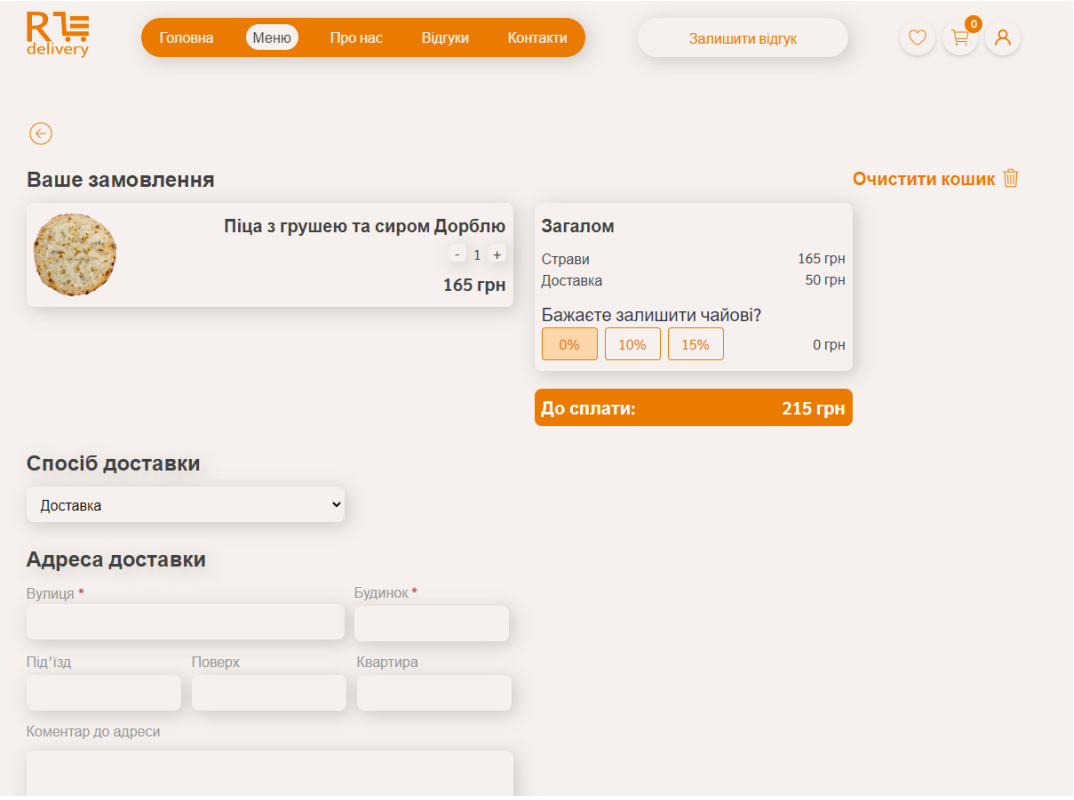


Рис.3.7 Сторінка веб-додатку «Кошику»

На цій сторінці користувач може побачити всі додані страви до свого кошика, які він додав до свого списку і побачити всю суму замовлення. Також користувач має можливість, якщо має бажання залишати чаєві при його бажанні вказавши це в кошику. Крім цього користувач може відразу оформити замовлення на тій ж самій сторінці, заповнивши всі необхідні форми. А якщо користувач передумав оформлювати замовлення, то є кнопка, яка очистить повністю весь кошик, прибравши всі страви, які додавав користувач.

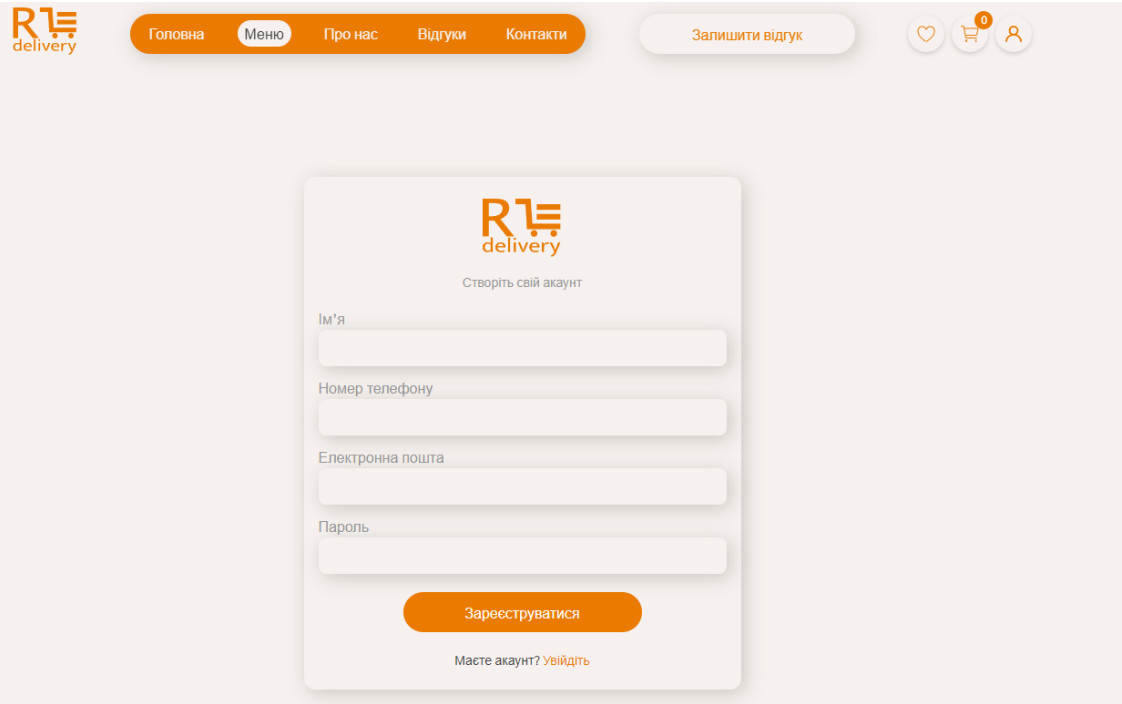


Рис. 3.8 Сторінка веб-додатку «Реєстрації»

На цій сторінці зображена форма реєстрації для користувача, де він вказує всі необхідні дані, та створює особистий кабінет, якщо в нього він відсутній.

4 ТЕСТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ СЛУЖБИ ДОСТАВКИ ЇЖІ.

Веб-додаток для служби доставки їжі протестований буде на сайті WebPageTest.org. Це безкоштовний онлайн-сервіс для тестування продуктивності веб-сторінок. Він дозволяє розробникам аналізувати швидкість завантаження особистих сторінок. Даний сайт дуже зручний, через те що він детальний звіт про сайт, який сервіс зміг проаналізувати сайт, а саме: розмір ресурсів, час завантаження, ефективність кешування та інші видів показників. Вони допомагають розробнику ідентифікувати та виправити проблеми на сайті та покращити швидкість їх завантаження.

Рисунок 4.1 – Результат тесту веб-додатку

В результаті ми отримуємо такі результати веб-додатку для служби доставки їжі. . В таблиці 4.1 буде зображенні отриманні результаті після тесту.

Таблиця 4.1

Отриманні результаті після тесту

Метрика	Опис	Значення
Час до першого байта (TTFB)	Коли почав завантажуватися контент?	.703C
Початок рендеру	Коли пікселі вперше почали з'являтися?	1.700C
Перше змістовне відображення (FCP)	Як швидко почали з'являтися текст та зображення?	1.745C
Індекс швидкості	Як швидко сторінка виглядала придатною до використання?	1.702C
Найбільше змістовне відображення (LCP)	Коли найбільший видимий контент завершив завантаження?	1.745C
Кумулятивне зміщення макета (CLS)	Наскільки сильно зміщувався дизайн під час завантаження?	0
Загальний час блокування (TBT)	Чи був заблокований основний потік?	.000C
Вага сторінки	Скільки байтів завантажено?	112KB

ВИСНОВКИ

Згідно результатів даної дипломної роботи було розроблено веб-додаток для служби доставки їжі використовуючи мови програмування HTML, CSS та Java Script.

В ході виконання дипломної роботи було виконано наступні ключові пункти:

– Проаналізували всі існуючі аналоги веб-додатків для служб доставок їжі та порівняли їх, в результаті чого було сформовано таблицю зі всіма недоліками та перевагами аналогів.

– Розроблені функціональні і нефункціональні вимоги до веб-додатку для служби доставки їжі.

– Зроблено дослідження та обрано засоби розробки для веб-додатку служби доставки їжі та розроблено модель архітектури веб-додатку.

– Розроблено веб-додаток служби для доставки їжі на основі обраних інструментів та визначених вимог.

– Проведено тестування веб-додатку служби для доставки їжі.

Робота пройшла апробацію:

1. Ребрик М.С., Садовенко В.С. Існуючі рішення веб-додатків для служби доставки їжі. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 18 квітня 2024 року., Збірник тез. К.: ДУІКТ, 2024. С.9-10.

2. Ребрик М.С., Садовенко В.С. Розробка веб-додатку для служби доставки їжі. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2024 року., Збірник тез. К.: ДУІКТ, 2024. С.198-199.

Створений веб-додаток має значний потенціал, щодо його вдосконалення в майбутньому та розширенням іншим функціоналом залежності від розвитку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Visual Studio Code [Електронний ресурс] // Microsoft. – 2024 – Режим доступу до ресурсу: https://code.visualstudio.com/docs.
2. Anthony Accomazzo. Fullstack React: The Complete Guide to ReactJS and Friends. / - 2017 – с. 1-836.
3. Jon Duckett. HTML & CSS Design and Build Web Sites./ 2011 – с. 1-512
4. Ivo Balbaert, Adrian Salceanu. Web Development with Julia and Genie. / 2022 – с. 1-254.
5. Marijn Haverbeke. A Modern Introduction to Programming. / 2014 – с. 1-472.
6. David Flanagan. JavaScript: The Definitive Guide: Activate Your Web Pages. / 2011. с. 1 – 1096.
7. Jon Duckett. HTML and CSS: Design and Build Websites. / 2011. с. 1 – 490.
8. Martine Dowden, Michael Gearon. Tiny CSS Projects. / 2023. с. 1 – 392.
9. Git [Електронний ресурс] // Software Freedom Conservancy. – 2024 – Режим доступу до ресурсу: https://www.git-scm.com/doc.
10. GitHub [Електронний ресурс] // GitHub, Inc. – 2024 – Режим доступу до ресурсу: https://docs.github.com/en.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА WEB-ЗАСТОСУНКУ ДЛЯ СЛУЖБИ ДОСТАВКИ ЇЖІ З ВИКОРИСТАННЯМ МОВ РОЗМІТКИ HTML-CSS, МОВИ ПРОГРАМУВАННЯ JAVA SCRIPT

Виконав студент 4 курсу
групи ПД-44
Ребрик Максим Сергійович
Керівник роботи
к.ф.-м.н., професор кафедри Садовенко Володимир Сергійович

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – спрощення процесу взаємодії клієнтів і служби доставки їжі за рахунок використання web-застосунку, створеного мовами HTML, CSS, Java Script.

Об’єкт дослідження – процес взаємодії клієнтів зі службою доставки їжі.

Предмет дослідження – web-застосунок служби для забезпечення взаємодії клієнтів і служби доставки їжі.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати існуючі аналоги web-застосунків для служб доставок їжі та провести порівняльний аналіз.
2. Розробити функціональні і нефункціональні вимоги до web-застосунку для служби доставки їжі.
3. Провести аналіз та вибрати засоби розробки для web-застосунку для служби доставки їжі.
4. Розробити web-застосунок служби для доставки їжі на основі обраних інструментів та визначених вимог.
5. Спроектувати модель архітектури, діаграму активності, діаграму варіантів використання, схему бази даних та мапу сайту для web-застосунку служби для доставки їжі.
6. Провести тестування web-застосунку служби доставки їжі.

АНАЛІЗ АНАЛОГІВ

Показник	Glovo	Zakaz.ua	Такфур	ReDelivery
Система відгуків	+	+	-	+
Можливість замовлення без реєстрації на сайті	-	-	-	+
Наявність списку улюбленого	+	+	-	+
Наявність чат-бота	-	+	-	+
Можливість залишити чайові	+	-	-	+
Швидкість завантаження сторінки	549ms	1800ms	1080ms	435ms

ВИМОГИ ДО WEB-ЗАСТОСУНКУ

Функціональні вимоги:

1. Можливість здійснення пошуку страв в меню за назвою, для забезпечення швидкого доступу до потрібної страви користувачу
2. Можливість ділитися своїм враженням та досвідом з закладом через форму зворотного зв'язку, допомагаючи йому покращити надання своїх послуг.
3. Додавати улюблені страви до списку бажаного.
4. Користувачі можуть створювали власний кабінет, безпечно входити в систему та безпечно керувати своїми даними та відслідковувати свої замовлення.
5. Дозволяє користувачам в реальному часі переглядати поточний статус замовлень, починаючи від моменту підтвердження замовлення до його доставки, що забезпечує прозорість.
6. Користувачі можуть в особистому кабінеті переглядати свої минулі зроблені замовлення та при необхідності зробити повторне замовлення.

Нефункціональні вимоги:

1. Здатність системи витримувати навантаження від час періодів пікового попиту втрати продуктивності або недоступності сервісу.
2. Дизайн web-застосунку, який адаптується до розміру дисплея пристрою користувача
3. HTML-розмітка яка допомагає пошуковим системам зрозуміти про що написано на кожній сторінці або окремому блоці контенту та про що web-застосунок загалом.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Visual Studio Code

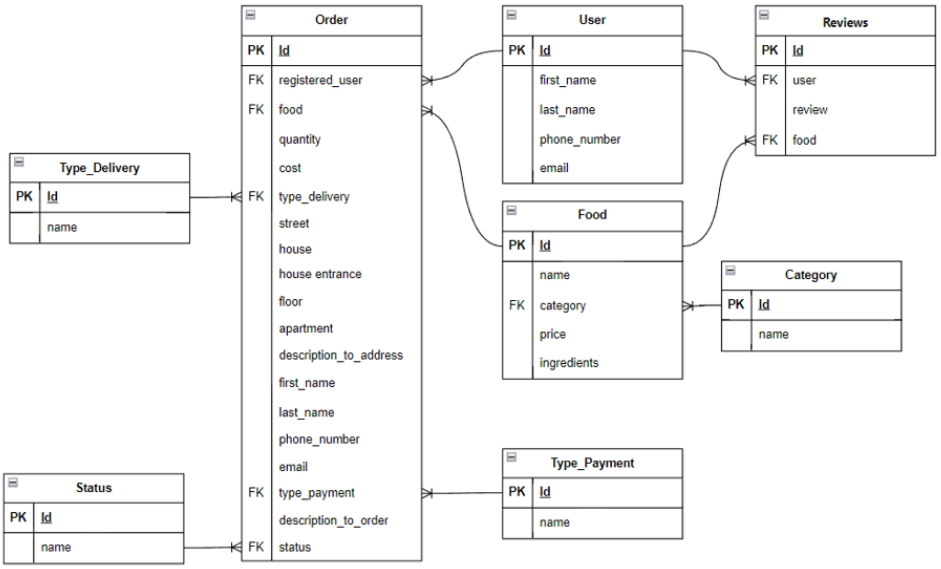
GitHub

Git

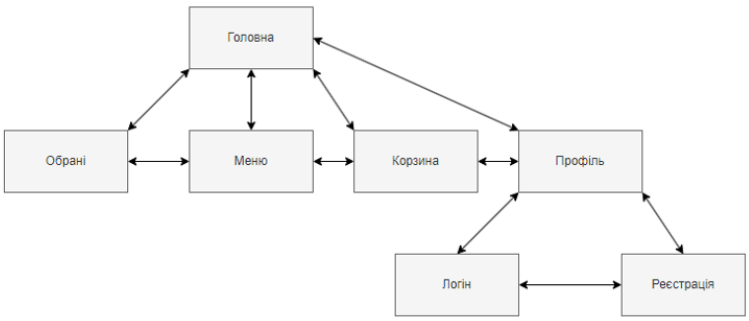
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ WEB-ЗАСТОСУНКУ

СХЕМА КЛІЄНТ-СЕВЕРНОЇ АРХІТЕКТУРИ WEB-ЗАСТОСУНКУ

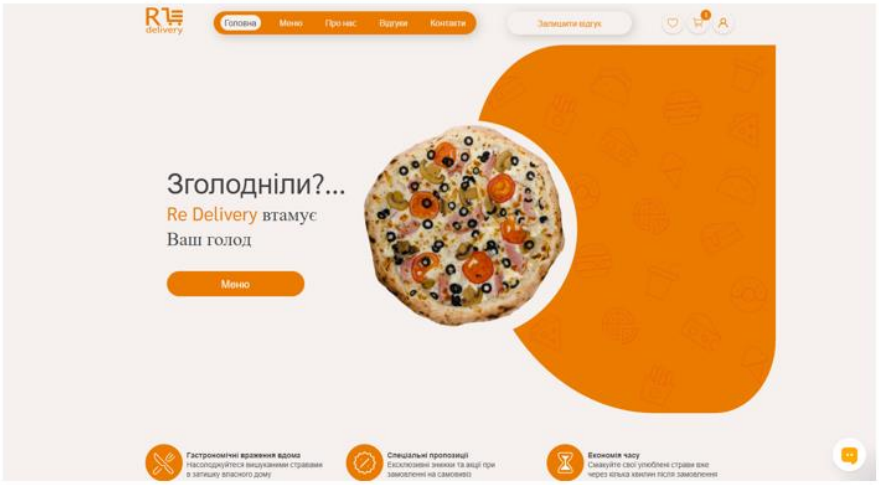
СХЕМА БАЗИ ДАНИХ



МАПА САЙТУ

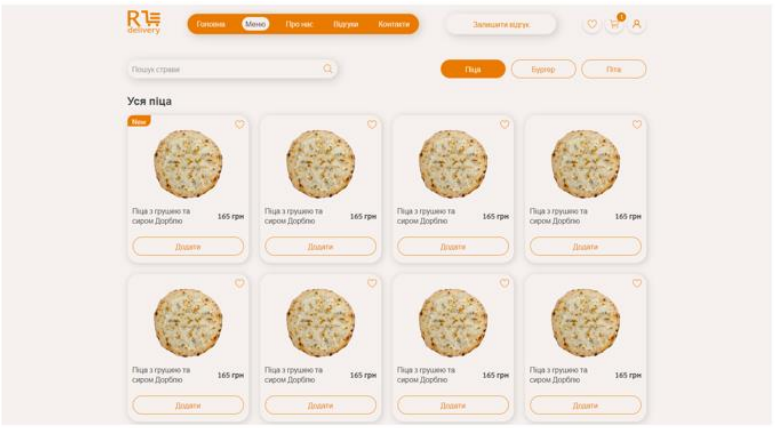


ЕКРАННІ ФОРМИ



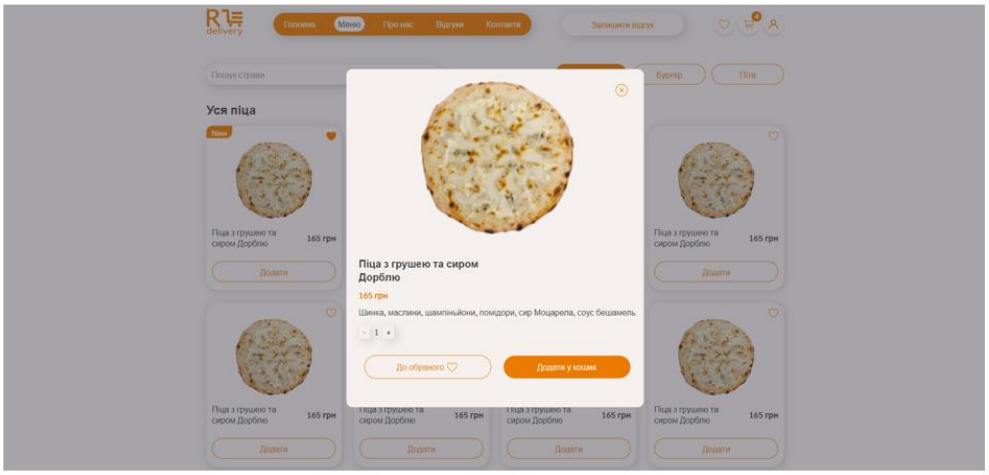
Головна сторінка

ЕКРАННІ ФОРМИ



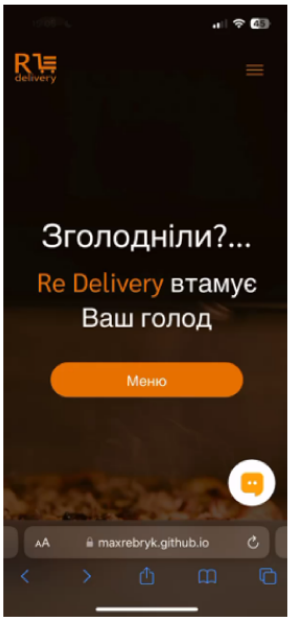
Сторінка меню страв

ЕКРАННІ ФОРМИ



Сторінка меню страв, після натискання кнопки додати

ВІДЕО РОБОТИ WEB-ЗАСТОСУНКУ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Ребрик М. С. Існуючі рішення веб-додатків для служби доставки їжі / Садовенко В.С., Ребрик М.С. // V Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез. 18 квітня 2024 року, ДУІКТ, м. Київ – К: ДУІКТ, 2024. Подано до друку.
2. Ребрик М. С. Розробка веб-додатку для служби доставки їжі / Садовенко В.С., Ребрик М.С. // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24 квітня 2024 року, ДУІКТ, м. Київ – К: ДУІКТ, 2024. с.198.

ВИСНОВКИ

1. Проаналізовано існуючі аналоги web-застосунків для служб доставок їжі та було проведено порівняльний аналіз, в результаті чого було сформовано таблицю з результатами аналізу.
2. Розроблені функціональні і нефункціональні вимоги до web-застосунку для служби доставки їжі.
3. Проведено аналіз та обрані засоби розробки: HTML, CSS, Java Script, Visual Studio Code, Git, GitHub для web-застосунку служби доставки їжі.
4. Спроектовано модель архітектури, діаграми варіантів використання, діаграми активності, схему бази даних та мапа сайту для web-застосунку для служби доставки їжі.
5. Розроблено web-застосунку для забезпечення взаємодії клієнтів зі службою доставки їжі на основі обраних інструментів та визначених вимог.
6. Проведено мануальне та функціональне тестування, в тому числі тестування продуктивності за допомогою сервісу WebPageTest.org.

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО МОДУЛЯ

```
'use strict';

import * as mobileMenu from '../js/mobile-menu.js';

import { pizza, burgers, pita } from '../js/menu-list.js';

const tipsButtonsUl = document.querySelector('.tips-
button-ul');
const renderContainer =
document.querySelector('.render-item-container');
const cleanOrderBtn = document.querySelector('.clean-
btn');
const mediumTipsBtn = document.querySelector('.tips-
ten-button');
const maxTipsBtn = document.querySelector('.tips-
fifteen-button');
const fullPrice = document.querySelector('.full-price');
const tipsPrice = document.querySelector('.tips-price');
const dishesPrice = document.querySelector('.dishes-
price');
const backMenuBtn = document.querySelector('.back-
menu-btn');
const deliveryForm = document.querySelector('.delivery-
form');
const DeliveryWay = document.querySelector('.delivery-
way-class');
const streetInput = document.querySelector('#street');
const buildingInput =
document.querySelector('#building');
const floorInput = document.querySelector('#floor');
const apartmentInput =
document.querySelector('#apartment');
const addressCommentInput =
document.querySelector('#address-comment');
const nameInput = document.querySelector('#name');
const phoneInput = document.querySelector('#phone');
const payWay = document.querySelector('#pay-way');
const orderCommentInput =
document.querySelector('#order-comment');
const secondNameInput =
document.querySelector('#second-name');
const emailInput = document.querySelector('#email');
const backdrop = document.querySelector('.backdrop');
const loginRedirectBtn =
document.querySelector('.login-redirect-btn');
const closeModalBtn = document.querySelector('.close-
modal-menu-btn');

let orderStorage = localStorage.getItem('order');
let orderArray = JSON.parse(orderStorage);

let totalOrderPrice = 0;
let tips = 0;

const favButton = document.querySelector('.fav-button');
const profileButton = document.querySelector('.profile-
button');
const cartButton = document.querySelector('.cart-
button');

cartButton.addEventListener('click', event => {
  window.location.href = './menu.html';
});

favButton.addEventListener('click', event => {
  window.location.href = './favourite.html';
});

profileButton.addEventListener('click', event => {
  window.location.href = './profile.html';
});

mobileMenu.addMobileMenuListener();

function calculatePrice() {
  tips = 0;
  if (mediumTipsBtn.classList.contains('is-active')) {
    tips = (totalOrderPrice * 10) / 100;
    tipsPrice.textContent = `${tips} грн`;
  }
  if (maxTipsBtn.classList.contains('is-active')) {
    tips = (totalOrderPrice * 15) / 100;
    tipsPrice.textContent = `${tips} грн`;
  }
  dishesPrice.textContent = `${totalOrderPrice} грн`;
  tipsPrice.textContent = `${tips} грн`;
  fullPrice.textContent = totalOrderPrice + tips + 50 + '
грн';
}

function updatePrices() {
  // Перераховуємо вартість всіх страв

  let totalDishesPrice = 0;
  orderArray.forEach(item => {
    const menuItem = getItemById(item.itemId);
    totalDishesPrice += menuItem.price * item.count;
  });
  dishesPrice.textContent = `${totalDishesPrice} грн`;
  tips = 0;
  if (mediumTipsBtn.classList.contains('is-active')) {
    tips = (totalDishesPrice * 10) / 100;
    tipsPrice.textContent = `${tips} грн`;
  }
  if (maxTipsBtn.classList.contains('is-active')) {
    tips = (totalDishesPrice * 15) / 100;
    tipsPrice.textContent = `${tips} грн`;
  }
  tipsPrice.textContent = `${tips} грн`;
  // Перераховуємо повну вартість замовлення з
урахуванням tips
  const totalFullPrice = totalDishesPrice + tips + 50; //
Додаємо 50 грн за доставку
  fullPrice.textContent = `${totalFullPrice} грн`;
}
```

```
function toggleActiveButton(btn) {
  const buttons =
tipsButtonsUl.querySelectorAll('button');
  buttons.forEach(button => {
    if (button.classList.contains('is-active')) {
      button.classList.remove('is-active');
    }
  });
  btn.classList.add('is-active');
}

function getItemById(itemId) {
  const itemFromPizza = pizza.find(item => item.id ===
parseInt(itemId));
  const itemFromBurgers = burgers.find(item => item.id
=== parseInt(itemId));
  const itemFromPita = pita.find(item => item.id ===
parseInt(itemId));
  if (itemFromPizza) return itemFromPizza;
  if (itemFromBurgers) return itemFromBurgers;
  if (itemFromPita) return itemFromPita;
}

document.addEventListener('DOMContentLoaded',
event => {
  orderArray.forEach((item, index) => {
    let itemId = item.itemId;
    const menuArray = getItemById(itemId);
    let counter = item.count;
    totalOrderPrice += menuArray.price * counter;
    const markup = `<div class="item-container"
id="item-${index}">
          <picture>
             <source
             srcset="
             ${menuArray.imgDesktop} 1x,
             ${menuArray.imgDesktop2x} 2x
             "
             media="(min-width:1158px)"
             height="300"
             width="360"
             />

             <source
             srcset="${menuArray.imgMobile2x} 2x"
             media="(max-width: 767px)"
             width="83"
             height="83"
             />
             <img class="item-img" src="${
               menuArray.imgMobile
             }" alt="pizza" width="83" height="83">
          </picture>

          <div class="item-info-container">

          <h3 class="item-
header">${menuArray.name}</h3>


          <div class="item-count-container">
             <button class="item-count-minus-btn"
type="button" data-info="${
               menuArray.id
             }">-</button>
             <p class="counter">${counter}</p>
             <button class="item-count-plus-btn"
type="button" data-info="${
               menuArray.id
             }">+</button>
           </div>
           <p class="item-price">${menuArray.price *
counter} грн</p>
           </div>
        </div>`;

    renderContainer.insertAdjacentHTML('beforeend',
markup);
  });

  calculatePrice();

  const countPlusBtns =
document.querySelectorAll('.item-count-plus-btn');
  const countMinusBtns =
document.querySelectorAll('.item-count-minus-btn');
  const itemsCounters =
document.querySelectorAll('.counter');
  const itemPrices = document.querySelectorAll('.item-
price');
  let counters = Array.from({ length:
countPlusBtns.length }, () => 1);

  for (let i = 0; i < countPlusBtns.length; i++) {
    countPlusBtns[i].addEventListener('click', event => {
      const dataId = countPlusBtns[i].getAttribute('data-
info');

      let orderStorage = localStorage.getItem('order');
      orderArray = JSON.parse(orderStorage);
      orderArray.forEach((item, index) => {
        if (item.itemId === dataId) {
          item.count += 1;
          itemsCounters[i].innerHTML = item.count;
          counters[i] = item.count;
          let itemFromMenuList =
getItemById(item.itemId);
          totalOrderPrice = itemFromMenuList.price *
item.count;
          let updatedPrice = itemFromMenuList.price *
item.count;
          itemPrices[i].textContent = `${updatedPrice} грн`;
        }
      });
      let updatedOrderJSON =
JSON.stringify(orderArray);
      localStorage.setItem('order', updatedOrderJSON);
      calculatePrice();
      updatePrices();
    });

    countMinusBtns[i].addEventListener('click', event =>
{
      if (counters[i] > 1) {
        const dataId =
countMinusBtns[i].getAttribute('data-info');
```

```
      let orderStorage = localStorage.getItem('order');
      orderArray = JSON.parse(orderStorage);
      orderArray.forEach((item, index) => {
        if (item.itemId === dataId) {
          item.count -= 1;
          itemsCounters[i].innerHTML = item.count;
          counters[i] = item.count;
          let itemFromMenuList =
getItemById(item.itemId);
          totalOrderPrice = itemFromMenuList.price *
item.count;
          let updatedPrice = itemFromMenuList.price *
item.count;
          itemPrices[i].textContent = `${updatedPrice}
грн`;
        }
      });
      let updatedOrderJSON =
JSON.stringify(orderArray);
      localStorage.setItem('order', updatedOrderJSON);
      calculatePrice();
      updatePrices();
    }
  });
 }
});

tipsButtonsUl.addEventListener('click', event => {
 const targetButton = event.target;
 if (targetButton.tagName === 'BUTTON') {
  toggleActiveButton(targetButton);

  calculatePrice();
  updatePrices();
 }
});

cleanOrderBtn.addEventListener('click', event => {
 localStorage.removeItem('order');
 let order = [];
 let orderJSON = JSON.stringify(order);
 localStorage.setItem('order', orderJSON);

 renderContainer.innerHTML = '';
 tipsPrice.textContent = `0 грн`;
 fullPrice.textContent = `0 грн`;
 dishesPrice.textContent = `0 грн`;
});

backMenuBtn.addEventListener('click', event => {
 history.back();
});

deliveryForm.addEventListener('submit', event => {

'use strict';

// Імпорт функцій з SDK, які вам потрібні
import { initializeApp } from 'firebase/app';
import { getAnalytics } from 'firebase/analytics';
import { getAuth, signInWithEmailAndPassword } from
'firebase/auth'; // Доданий імпорт auth
 event.preventDefault();
 let deliveryInfo = {};

 deliveryInfo.deliveryWay = DeliveryWay.value;
 deliveryInfo.street = streetInput.value;
 deliveryInfo.building = buildingInput.value;
 deliveryInfo.floor = floorInput.value;
 deliveryInfo.apartment = apartmentInput.value;
 deliveryInfo.addressComment =
addressCommentInput.value;
 deliveryInfo.name = nameInput.value;
 deliveryInfo.secondName = secondNameInput.value;
 deliveryInfo.email = emailInput.value;
 deliveryInfo.phone = phoneInput.value;
 deliveryInfo.payWay = payWay.value;
 deliveryInfo.orderComment =
orderCommentInput.value;
 deliveryInfo.tips = tips;
 let deliveryInfoJSON = JSON.stringify(deliveryInfo);
 localStorage.setItem('deliveryInfo', deliveryInfoJSON);

 let confirmedOrder = localStorage.getItem('order');
 let confirmedOrderArray =
JSON.parse(confirmedOrder);

 let confirmedOrderJSON =
JSON.stringify(confirmedOrderArray);
 localStorage.setItem('confirmedOrder',
confirmedOrderJSON);
 if (localStorage.getItem('orderHistory') !== 'true') {
   let orderHistoryArray =
     JSON.parse(localStorage.getItem('orderHistory')) ||
[]; // Отримання попередньої історії замовлень
   orderHistoryArray.push(orderArray); // Додавання
нового замовлення до історії
   localStorage.setItem('orderHistory',
JSON.stringify(orderHistoryArray)); // Збереження
оновленої історії замовлень
 }
 localStorage.removeItem('order');
 let order = [];
 let orderJSON = JSON.stringify(order);
 localStorage.setItem('order', orderJSON);

 backdrop.classList.add('is-open');
});

loginRedirectBtn.addEventListener('click', event => {
 window.location.href = './login.html';
});

closeModalBtn.addEventListener('click', event => {
 window.location.href = './menu.html';
});

// TODO: Add SDKs for Firebase products that you want
to use
// https://firebase.google.com/docs/web/setup#available-
libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId
is optional
```

```
export const firebaseConfig = {
  apiKey:
'AIzaSyAerKKFXmuky6LCeiAWuifTbVVPXZ9Tclw',
  authDomain: 're-delivery-cfa80.firebaseapp.com',
  projectId: 're-delivery-cfa80',
  storageBucket: 're-delivery-cfa80.appspot.com',
  messagingSenderId: '425549554900',
  appId: '1:425549554900:web:3f8efb60ffb068d54fc741',
  measurementId: 'G-LEEL210TP7',
};

// Ініціалізація Firebase
export const app = initializeApp(firebaseConfig);
export const analytics = getAnalytics(app);
export const auth = getAuth(app); // Отримання об'єкта
auth

'use strict';

import * as mobileMenu from '../js/mobile-menu.js';
import { pizza, burgers, pita } from '../js/menu-list.js';

const renderList = document.querySelector('.current-
order-items-list');
const historyRenderList =
document.querySelector('.history-items-list');
const dishesPrice = document.getElementById('dishes-
price');
const fullPrice = document.getElementById('full-price');
const tipsPrice = document.getElementById('tips-price');
const favButton = document.querySelector('.fav-button');
const profileButton = document.querySelector('.profile-
button');
const cartButton = document.querySelector('.cart-
button');
let deliveryInfo = localStorage.getItem('deliveryInfo');
let deliveryInfoArray = JSON.parse(deliveryInfo);
let orderStorage =
localStorage.getItem('confirmedOrder');
let orderArray = JSON.parse(orderStorage);

let totalOrderPrice = 0;
let tips = deliveryInfoArray.tips;

cartButton.addEventListener('click', event => {
  window.location.href = './menu.html';
});

favButton.addEventListener('click', event => {
  window.location.href = './favourite.html';
});

profileButton.addEventListener('click', event => {
  console.log(1);
  window.location.href = './profile.html';
});

function cartCount() {
  const cartCounter = document.querySelector('.cart-
counter');
  const orderStorage = localStorage.getItem('order');
  let orderArray = JSON.parse(orderStorage);
  // Перевірка на наявність масиву
  if (orderArray) {
    cartCounter.textContent = orderArray.length;
  } else {
    cartCounter.textContent = '0'; // Якщо масив
порожній
  }
}

cartCount();

function areLogin() {
  // Перевіряємо, чи є користувач увійшов без
очікування DOMContentLoaded
  const userLoggedIn =
JSON.parse(localStorage.getItem('userLoggedIn'));
  if (userLoggedIn) {
    return true;
  } else {
    window.location.href = './login.html';
  }
  console.log(userLoggedIn);
  // Повертаємо true, якщо користувач увійшов, false в
іншому випадку
}

areLogin();

function getItemById(itemId) {
  const itemFromPizza = pizza.find(item => item.id ===
parseInt(itemId));
  const itemFromBurgers = burgers.find(item => item.id
=== parseInt(itemId));
  const itemFromPita = pita.find(item => item.id ===
parseInt(itemId));
  if (itemFromPizza) return itemFromPizza;
  if (itemFromBurgers) return itemFromBurgers;
  if (itemFromPita) return itemFromPita;
}

mobileMenu.addMobileMenuListener();

document.addEventListener('DOMContentLoaded',
event => {
  renderList.textContent = '';
  orderArray.forEach((item, index) => {
    let itemId = item.itemId;
    const menuArray = getItemById(itemId);
    let counter = item.count;
    totalOrderPrice += menuArray.price * counter;
    const markup = ` <li class="current-order-item">
    <div class="item-container" id="item-${index}">
      <div class="item-picture-container">
        <picture>
          <source srcset="
          ${menuArray.imgDesktop} 1x,
          ${menuArray.imgDesktop2x} 2x
          " media="(min-width:1158px)" height="80"
width="80" />

          <source srcset="${
            menuArray.imgMobile2x
          } 2x" media="(max-width: 767px)"
width="83"
```

```
            height="84" />
          <img class="menu-img" src="${
            menuArray.imgMobile
          }" alt="pizza" width="83"
            height="84">
        </picture>
      </div>

      <div class="item-info-container">

        <h3 class="item-
header">${menuArray.name}</h3>


        <div class="item-count-container">

          <p class="counter">${counter} x</p>

        </div>
        <p class="item-price">${menuArray.price *
counter} грн</p>
      </div>
    </div>
</li>`;

    renderList.insertAdjacentHTML('beforeend', markup);
  });
  dishesPrice.textContent = `${totalOrderPrice} грн`;
  tipsPrice.textContent = `${tips} грн`;
  fullPrice.textContent = totalOrderPrice + tips + 50 + '
грн';
});

document.addEventListener('DOMContentLoaded',
event => {
  historyRenderList.textContent = '';
  let orderHistoryArray =
JSON.parse(localStorage.getItem('orderHistory'));
  orderHistoryArray.forEach((order, index) => {
    let markupArray = order.map(item => {
      let itemId = item.itemId;
      const menuHistoryArray = getItemById(itemId);
      let counter = item.count;
      return `  <li class="history-orders-items">
        <div class="item-picture-container">
          <picture>
            <source srcset="
              ${menuHistoryArray.imgDesktop} 1x,
              ${menuHistoryArray.imgDesktop2x} 2x
            " media="(min-width:1158px)" height="80"
width="80" />

            <source srcset="${
              menuHistoryArray.imgMobile2x
            } 2x" media="(max-width: 767px)" width="83"
height="84" />
            <img class="menu-img" src="${
              menuHistoryArray.imgMobile
            }" alt="pizza" width="83" height="84">
          </picture>
          </div>
          <div class="history-item-info-container">
            <h3 class="item-header">${counter}x
${menuHistoryArray.name}</h3>
            <p class="item-
price">${menuHistoryArray.price * counter} грн</p>
          </div>
        </div>
      </li>`;
    });
    historyRenderList.insertAdjacentHTML('afterbegin',
markupArray.join(''));
  });
});
```