Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Бондаря Богдана Сергійовича*
(ПІБ)

академічної групи *122-19з-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-сайту для ресторану з використанням технологій HTML/CSS/JS та фреймворку Bootstrap*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Каштан В. Ю.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19з-1* *Бондаря Б. С.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-сайту для ресторану з використанням технологій HTML/CSS/JS та фреймворку Bootstrap*

затверджена наказом ректора НТУ «ДП» від *11.04.2023* № *256-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ ***доц. Кабак Л.В.***
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ _____
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Об’єкт розробки: сайт ресторану з використанням технологій HTML/CSS/JS та фреймворку Bootstrap.

Мета кваліфікаційної роботи: розробка сайту для ресторану з використанням технологій HTML, CSS, JS та фреймворку Bootstrap. Веб-сайт буде розроблено для демонстрації меню ресторану, розташування та іншої важливої інформації для потенційних клієнтів.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні веб сайту ресторану головною ціллю якого є показати меню закладу для залучення потенційниї клієнтів.

Актуальність даного програмного продукту визначається великим попитом на подібні розробки, багато людей відвідують заклади харчування а також замовляють їжу додому. Тому опція перегляду меню на сайту чи оформлення замовлення з доставкою додому є важливими критеріями для закладу харчування у сучасному світі.

Список ключових слів: ВЕБ САЙТ, РЕСТОРАН, HTML, CSS, JS, BOOTSTRAP, FRAMEWORK, ЗАМОВЛЕННЯ, МЕНЮ, ОНЛАЙН, АДАПТИВНИЙ, ВЕБ-ДИЗАЙН, КЛІЄНТ.

ABSTRACT

Object of development: restaurant website using HTML/CSS/JS technologies and the Bootstrap framework.

The purpose of the qualification work: development of a website for a restaurant using HTML, CSS, JS and the Bootstrap framework. The website will be designed to showcase the restaurant's menu, location and other important information to potential customers.

In the introduction, the analysis and current state of the problem is considered, the purpose of the qualification work and the field of its application are specified, the justification of the relevance of the topic is given, and the statement of the task is clarified.

In the first section, the subject area is analyzed, the relevance of the task and the purpose of the development is determined, the task statement is formulated, and the requirements for software implementation, technologies and software tools are specified.

In the second section, available solutions are analyzed, platforms for development are selected, program design and development is performed, program operation, algorithm and structure of its functioning are described, as well as program calling and loading, input and output data are determined, and the composition of technical means parameters is characterized.

In the economic section, the labor intensity of the developed information system is determined, the cost of work on creating the program is calculated, and the time for its creation is calculated.

The practical significance lies in the creation of a restaurant website, the main purpose of which is to show the restaurant's menu to attract potential customers.

The relevance of this software product is determined by the great demand for such developments, many people visit restaurants and also order food at home. Therefore, the option to view the menu on the website or place an order with home delivery are important criteria for a restaurant in today's world.

Keyword List: WEBSITE, RESTAURANT, HTML, CSS, JS, BOOTSTRAP, FRAMEWORK, ORDER, MENU, ONLINE, RESPONSIVE, WEB DESIGN, CLIENT.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

HTML — мова розмітки веб-сторінки;
CSS – мова стилів веб-сторінки;
JS - мова програмування JavaScript;
HTML-тег - елемент розмітки, укладений у кутові дужки (< >);
Фреймворк – набір інструментів які забезпечують основу для розробки програмного забезпечення;
CSS-властивість - атрибут або характеристика, яку можна застосувати до елемента HTML;

ВСТУП

Темою даної кваліфікаційної роботи є розробка сайту ресторану з використанням технологій HTML/CSS/JS та фреймворку Bootstrap.

Метою даної кваліфікаційної роботи є представлення розробки сайту для ресторану з використанням технологій HTML/CSS/JS та фреймворку Bootstrap. Веб-сайт розроблений таким чином, щоб бути адаптивним, зручним і візуально привабливим, щоб залучити потенційних клієнтів до ресторану.

У сучасну епоху цифрових технологій веб-сайт є потужним інструментом для бізнесу, щоб створити свою присутність в Інтернеті та охопити ширшу аудиторію. Зокрема, ресторани можуть отримати значну користь від веб-сайту, який демонструє їхнє меню, місце розташування та атмосферу, а також дозволяє клієнтам робити виклик при натисканні на кнопку – посилання для оформлення замовлення по телефону. Розробивши адаптивний, зручний, візуально привабливий веб-сайт, ресторан може залучити нових клієнтів і створити позитивний досвід для користувачів.

Розробка веб-сайту здійснюватиметься з використанням технологій HTML, CSS та JavaScript разом із фреймворком Bootstrap. Використання технологій HTML, CSS і JavaScript дозволить створити візуально привабливий та інтерактивний веб-сайт, зручний для користувача та простий у навігації. Крім того, використання фреймворку Bootstrap дозволить веб-сайту бути адаптивним і доступним на різних пристроях, включаючи настільні комп’ютери, планшети та смартфони.

Bootstrap — це фреймворк із відкритим вихідним кодом, розроблений Twitter. Це набір компонентів і інструментів HTML, CSS і JavaScript, які спрощують процес створення адаптивних і візуально привабливих веб-сайтів. Bootstrap надає адаптивну систему сіток, попередньо оформлені компоненти (наприклад, кнопки, форми та панелі навігації) і широкий спектр службових класів для оптимізації веб-розробки та забезпечення узгодженого дизайну на різних пристроях і розмірах екрана.

Загалом ця дипломна робота має на меті продемонструвати важливість веб-

розробки в ресторанній індустрії та переваги, які добре розроблений веб-сайт може принести бізнесу.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Розробка веб-сайтів стала важливим аспектом створення онлайн-присутності для компаній у різних галузях. У випадку ресторанів добре спроектований і функціональний веб-сайт відіграє вирішальну роль у залученні клієнтів, демонстрації меню, наданні контактної інформації та створенні чудової взаємодії з користувачем. Використання можливостей HTML, CSS, JavaScript і фреймворку Bootstrap забезпечує чудову основу для створення професійного та візуально привабливого веб-сайту для ресторану.

HTML (HyperText Markup Language) є основою розробки веб-сайтів, дозволяючи розробникам ефективно структурувати та організовувати вміст. За допомогою HTML можна визначити структуру веб-сайту ресторану, включаючи заголовки, абзаци, списки, зображення та інші важливі елементи. Він забезпечує необхідну основу для подання інформації в логічний і семантично значущий спосіб.

CSS (каскадні таблиці стилів) доповнює HTML, дозволяючи налаштовувати та стилізувати веб-сторінки. Це дозволяє розробникам визначати кольори, шрифти, макети та інші візуальні властивості, забезпечуючи єдиний і візуально привабливий дизайн веб-сайту ресторану. CSS відіграє ключову роль у створенні унікальної та привабливої естетики, яка відображає ідентичність бренду ресторану та створює приємний досвід користувача.

JavaScript додає веб-сайту ресторану інтерактивність і динамічність. Використовуючи JavaScript, розробники можуть створювати інтерактивні елементи, такі як меню зі спадними меню, повзунки зображень, контактні форми та інші функції, які покращують залучення користувачів. JavaScript також

дозволяє отримувати дані в реальному часі, дозволяючи відображати спеціальні пропозиції, наявність бронювання тощо, надаючи відвідувачам актуальну та актуальну інформацію.

Фреймворк Bootstrap є цінним інструментом для прискорення розробки веб-сайту. Завдяки попередньо розробленим і адаптивним компонентам CSS і JavaScript Bootstrap спрощує створення веб-сайту ресторану. Він пропонує систему сіток для створення адаптивних макетів, готових до використання навігаційних меню, форм, кнопок та інших компонентів інтерфейсу користувача, що зменшує потребу у великому кодуванні з нуля. Крім того, Bootstrap забезпечує сумісність із різними пристроями, забезпечуючи безперебійне відображення та роботу веб-сайту на комп'ютерах, планшетах і мобільних пристроях.

1.2. Призначення розробки та галузь застосування

Метою розробки веб-сайту ресторану з використанням HTML, CSS, JavaScript і фреймворку Bootstrap є створення сильної присутності ресторану в Інтернеті та забезпечення безперебійного цифрового досвіду для клієнтів. Веб-сайт слугує потужним маркетинговим інструментом, що дозволяє ресторану демонструвати свої унікальні пропозиції, залучати нових клієнтів і підвищувати залученість і задоволеність клієнтів.

Добре розроблений веб-сайт для ресторану дозволяє потенційним клієнтам досліджувати меню, переглядати привабливі візуальні зображення їжі та напоїв і збирати відповідну інформацію, таку як години роботи, місцезнаходження та контактні дані. Використовуючи технологію HTML, CSS, JavaScript і Bootstrap, веб-сайт може запропонувати інтуїтивно зрозумілий і зручний інтерфейс, забезпечуючи позитивний досвід перегляду для відвідувачів.

Крім того, веб-сайт ресторану, розроблений за допомогою HTML, CSS, JavaScript і Bootstrap, сумісний із широким спектром пристроїв, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони. Це гарантує, що клієнти можуть безперешкодно отримати доступ до веб-сайту та його функцій з будь-

якого пристрою, який вони віддають перевагу, ще більше підвищуючи доступність і зручність.

Підсумовуючи, мета розробки веб-сайту ресторану з використанням HTML, CSS, JavaScript і фреймворку Bootstrap полягає в тому, щоб створити привабливу та інформативну цифрову платформу, яка покращує взаємодію з клієнтами, стимулює розвиток бізнесу та зміцнює присутність ресторану в Інтернеті.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

– освітня програма спеціальності 122 «Компьютерні науки»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 256-с від 11.04.2023;

завдання на кваліфікаційну роботу на тему «Розробка веб-сайту для ресторану з використанням технологій HTML/CSS/JS та фреймворку Bootstrap».

1.4. Постановка завдання

Завданням даної роботи є створення швидкого та інтуітивно зрозумілого сайту ресторану. Веб-сайт повинен бути розроблений таким чином, щоб бути адаптивним, зручним і візуально привабливим, щоб залучити потенційних клієнтів до ресторану.

Основними характеристиками розробки повинні бути:

1. Адаптивний дизайн: веб-сайт повинен бути оптимізований для різних пристроїв, включаючи настільні ПК, ноутбуки, планшети та смартфони.

2. Зручна навігація: Веб-сайт повинен мати зрозумілу та просту у користуванні систему навігації, що дозволяє відвідувачам швидко та легко знаходити потрібну інформацію.

3. Візуально привабливий дизайн: веб-сайт повинен мати привабливий дизайн, який відображає бренд ресторану та створює позитивний досвід для користувачів.

4. Демонстрація меню ресторану: веб-сайт повинен мати розділ меню, який демонструє страви ресторану, ціни та опис страв.

5. Місцезнаходження та контактна інформація: веб-сайт повинен надавати точне місцезнаходження та контактну інформацію, включаючи адресу, номер телефону та електронну адресу.

Поставлена задача може бути досягнута при виконанні наступних вимог:

- вивчення предметної області завдання;
- вибір платформи розробки;
- написання програмного коду;
- розробка довідника для пояснення користування додатком.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

- інтуїтивно зрозумілий інтерфейс користувача;
- перегляд меню страв з їх описом та ціною;
- перегляд контактної інформації а також розташування закладу на Google Maps;
- можливість здійснити виклик до ресторану натиснувши на кнопку-посилання якщо використовується мобільний пристрій;

1.5.2. Вимоги до інформаційної безпеки

Додаток запаковано в один файл, крім цього у користувача немає необхідності реєструватися і створювати аккаунт на сайті. Тому ніяких вимог до інформаційної безпеки немає.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для швидкого та коректного відображення сайту пристрій з якого він переглядається повинен відповідати наступним технічним вимогам:

- операційна система Windows XP-11, MacOS, Linux, Android, IOS;
- сучасний браузер з підримкою Javascript, фреймворків Bootstrap та jQuery;
- наявніть оперативної пам’яті не менше 512 мб для швидкого функціонування сайту;

1.5.4 Вимоги до інформаційної та програмної сумісності

Переглянути сайт можливо просто перейшовши за посилання або ввівши власноруч url-адресу до адресного рядка браузера.

Для швидкого та коректного відображення сайту слід дотримуватись наступних програмних вимог:

- операційна система Windows XP-11, MacOS, Linux, Android (бажано версія не нижче 4.4.2), IOS(бажано версія не нижче 6.0);
- сучасний браузер з підримкою Javascript, фреймворків Bootstrap та jQuery;

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи має бути веб додаток створений за допомогою технологій HTML, CSS, JavaScript, а також з використанням фреймворку Bootstrap для заохочування потенційних кліентів веб-сайту.

Основний функціонал додатка полягає в тому щоб потенційний клієнт ресторану міг переглянути меню страв, їх опис та ціну онлайн не виходячи з дому.

Крім цього, для кращої і якісної роботи програми, повинен бути присутнім наступний додатковий функціонал:

– перегляд усіх наявних категорій та окремих для кожної категорії страв ресторану;

– здійснення виклику при натисканні на кнопку-посилання з номер телефону для оформлення заказу;

– перегляд контактних даних(адреса та графік роботи), а також розташування закладу за допомогою блоку який використовую фрейм з Google Maps;

2.2. Опис застосованих математичних методів

Під час проектування та розробки даної інформаційної системи використовувалися лише прості арифметичні дії. Математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Під час проектування, основним інструментом та набором шаблонів стане фреймворк Bootsttrap.

Bootstrap — це популярна інтерфейсна платформа веб-розробки з відкритим вихідним кодом, яка надає колекцію попередньо розроблених компонентів HTML, CSS і JavaScript для створення адаптивних і орієнтованих на мобільні пристрої веб-інтерфейсів.

Основні переваги використання Bootstrap для веб-розробки:

1. Швидка розробка: Bootstrap надає широкий спектр попередньо розроблених компонентів, таких як кнопки, форми та меню навігації, які можна легко налаштувати та інтегрувати у веб-проект. Це прискорює процес розробки та скорочує кількість часу, необхідного для кодування та проектування.
2. Узгодженість: компоненти Bootstrap відповідають узгодженій мові дизайну та оптимізовані для сумісності з різними браузерами, що полегшує створення узгодженого та професійно виглядаючого веб-інтерфейсу.
3. Адаптивність: Bootstrap розроблено для створення адаптивного веб-інтерфейсу, який автоматично адаптується до різних розмірів екрана та типів пристроїв, таких як настільні ПК, планшети та смартфони. Це допомагає гарантувати, що веб-вміст доступний і простий у навігації для всіх користувачів, незалежно від їх пристрою.
4. Налаштування: Bootstrap надає широкий спектр параметрів налаштування, що дозволяє веб-розробникам легко змінювати зовнішній вигляд і поведінку компонентів відповідно до унікальних потреб їх проекту.

Що стосується адаптивного веб-дизайну, Bootstrap містить потужну систему сітки, яка дозволяє розробникам створювати гнучкі та адаптивні макети для своїх веб-сторінок. Ця система сіток використовує комбінацію класів CSS і елементів HTML для створення сітки стовпців і рядків, які можна легко налаштувати для різних розмірів екрана.

Наприклад, система сітки дозволяє розробникам визначати кількість стовпців, які повинен охоплювати певний елемент на певному розмірі пристрою. Це гарантує, що вміст на веб-сторінці впорядковано таким чином, щоб він був оптимізований для пристрою, який використовується для його перегляду.

Система сітки Bootstrap базується на макеті з 12 стовпцями, який забезпечує гнучку та адаптивну структуру для розробки веб-інтерфейсів. Система сітки використовує класи CSS і елементи HTML для визначення сітки з 12 стовпців однакової ширини, які можна використовувати для створення складних макетів веб-сторінок.

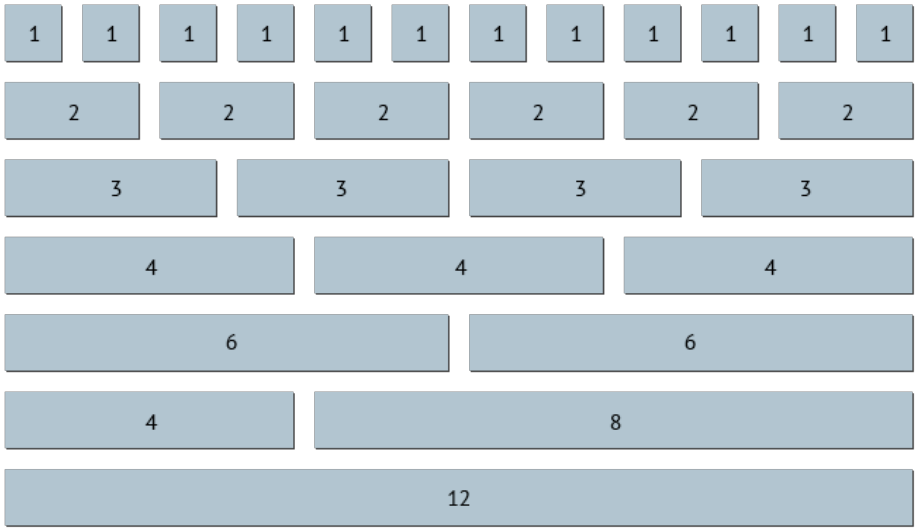


Рис. 2.1. Система сітки Bootstrap з 12 стовбцями

Щоб використовувати систему сітки, ви спочатку визначаєте елемент контейнера, наприклад тег , який має клас контейнера. Всередині контейнера ви можете створювати рядки, використовуючи тег із класом рядків. Тоді кожен рядок може містити один або кілька стовпців, які визначаються за допомогою тегу з такими класами, як col-sm-6 або col-lg-4.

Класи, які використовуються для визначення стовпців у системі сітки Bootstrap, базуються на комбінації розміру екрана та ширини стовпця. Серед доступних розмірів екрана:

- xs (дуже маленький): для екранів менше 576 пікселів
- sm (маленький): для екранів від 576 пікселів до 768 пікселів
- md (середній): для екранів від 768 пікселів до 992 пікселів
- lg (великий): для екранів від 992 пікселів до 1200 пікселів
- xl (дуже великий): для екранів розміром понад 1200 пікселів

Ширина стовпця визначається як число від 1 до 12, яке представляє кількість стовпців, які має охоплювати певний елемент. Наприклад, col-md-6 створить стовпець, який охоплює 6 із 12 стовпців на екранах середнього розміру.

Ось приклад простого макета Bootstrap із використанням сітки:

```
<div class="container">
  <div class="row">
    <div class="col-sm-6 col-md-4">Column 1</div>
    <div class="col-sm-6 col-md-4">Column 2</div>
    <div class="col-sm-12 col-md-4">Column 3</div>
  </div>
</div>
```

У цьому прикладі ми маємо елемент-контейнер з одним рядком, який містить три стовпці. На маленьких екранах (sm) кожен стовпець охоплює 6 із 12 стовпців, створюючи по два стовпці на рядок, за виключенням останнього, так як він має клас “col-sm-12”, але це можна змінити замінивши клас на “col-sm-6”

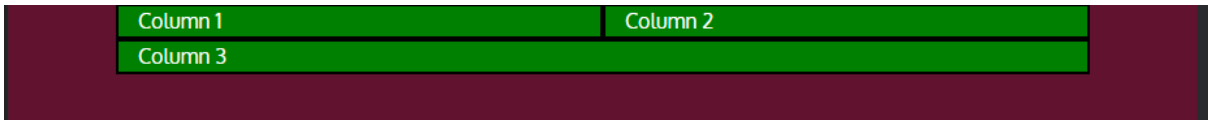


Рис. 2.2. останній рядок з класом “col-sm-12”

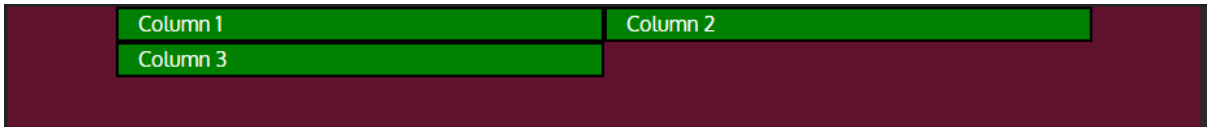


Рис. 2.3. останній рядок з класом “col-sm-6”

На середніх екранах (md) кожен стовпець охоплює 4 із 12 стовпців, створюючи по три стовпці на рядок.

Рис. 2.4.

Усе це здійснюється за допомогою CSS-властивості “width”.

Властивість CSS "width" встановлює ширину елемента. Вона визначає ширину вмісту елемента і не включає відступи, межі чи поля. Значення можна

встановити в пікселях, відсотках, em або інших одиницях вимірювання довжини. Наприклад, встановив ширину елемента на «50%» елемент займе половину ширини батьківського елемента. Якщо для ширини встановлено значення «auto», елемент займатиме стільки горизонтального простору, скільки вимагає його вміст. Якщо ширина встановлена на "100%", елемент займатиме всю ширину свого батьківського елемента.

У нашому прикладі ми використовували класи “col-sm-4”, “col-md-4” там col-sm-12”. Кожен з цих класів відповідне значення CSS-властивості “width”, тобто у випадку з класом “col-sm-4”,“col-md-4”, елементи з будь-яким з цих класів буде займати 33.3% від ширини батьківського елемента або вікна браузера, відповідно елемент з класом “col-sm-12” займатиме 100% ширини.

Також варто додати, на що саме впливає використання буквосполучен “sm”, “md” та інших і як це пов’язано з такою річчю як точки зупину (breakpoints) в CSS. У Bootstrap точки зупину стосуються певної ширини екрана, при якій змінюється макет веб-сторінки. Bootstrap використовує мобільний підхід до веб-дизайну, що означає, що стилі та макет за замовчуванням оптимізовані для пристроїв з невеликим екраном, таких як смартфони.

Точки зупину використовуються для визначення різних правил CSS для різних розмірів екрана, що дозволяє адаптувати веб-сторінку та забезпечити кращу взаємодію з користувачем на пристроях різних розмірів. Bootstrap має попередньо визначені точки зупину, які зазвичай використовуються у веб-дизайні. Ці контрольні точки визначаються таким чином:

- xs: надзвичайно малі пристрої (менше 576 пікселів)
- sm: маленькі пристрої (576 пікселів і більше)
- md: пристрої середнього розміру (768 пікселів і більше)
- lg: великі пристрої (992 пікселів і більше)
- xl: дуже великі пристрої (1200 пікселів і більше)
- xxl: надзвичайно великі пристрої (1400 пікселів і більше)

Наприклад, розробник може визначити правило CSS для точки зупину sm, щоб змінити макет веб-сторінки, коли розмір екрана становить 576 пікселів або

більше. Правило CSS може налаштувати розмір шрифту або змінити макет меню, щоб краще відповідати великому екрану.

Система сітки Bootstrap також використовує точки зупину, щоб визначити кількість стовпців, які має займати елемент на різних розмірах екрана. Наприклад, стовпець може займати 12 стовпців на малих пристроях (точка зупину sm), але лише 6 стовпців на середніх пристроях (точка зупину md). Це дає змогу адаптувати макет і забезпечити узгоджену взаємодію з користувачем на пристроях різного розміру.

Використовуючи наш попередній приклад з трьомо стовбцями, розглянемо як саме працюють точки зупину і де можна побачити їх властивості. Я буду демонструвати як саме працюють точки зупину на прикладі браузеру Google Chrome, але ця функція присутня майже у кожному сучаному браузері.

1. Переходимо до вікна розробника натиснувки ПКМ - переглянути код або просто натискаємо клавішу F12.
2. Поруч з меню на панелі розробника знаходимо кнопку “toggle devive toolbar”.

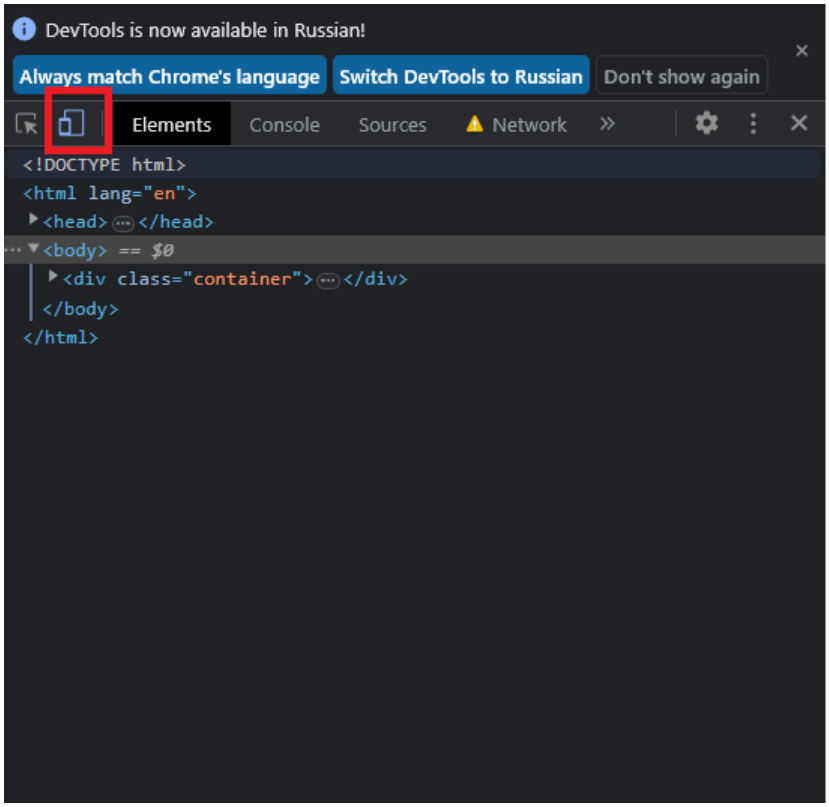


Рис 2.5. Демонстрація кнопки “toggle devive toolbar” у браузері Google Chrome

3. Далі з’являється вікно у якому ми можемо симулювати різні розміру екрану на якому буде відображатися наша веб-сторінка просто розтягуючи вікно зліва направо чи навпаки.

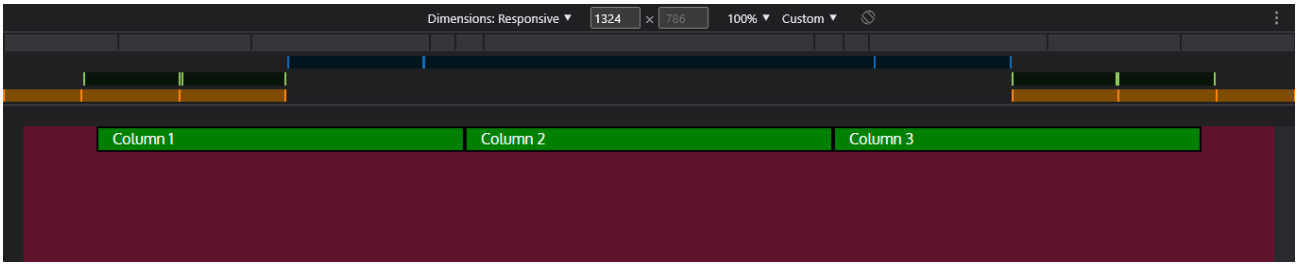


Рис 2.6. Демонастрація точок зупину у панелі розробника Google Chrome

Тут нарешті ми і можемо візуально побачити які саме точки зупину має конкретна веб-сторінка і яке саме мінімальне і максимальне значення вона має у пікселах. На малюнку нижче ми бачимо що під час того як усі три стовпці займають один рядок, активна точка зупину має мінімальне значення 992px і це означає, що поки ширина вікна чи екрану пристрою будет 992px або більше, то буде активна CSS-властивість “width” з потрібним нам значенням, як показано на малюнку з властивістю у вікні розробника браузера.

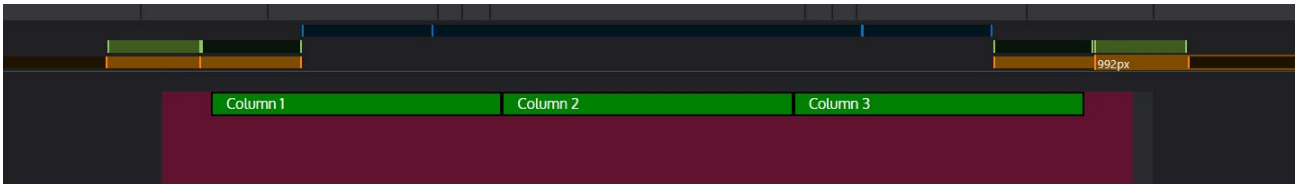


Рис 2.7. активна точка зупину з мінімальним порогом 992px

```
@media (min-width: 992px)
.col-md-4 {                                grid-framework.less:48
  width: 33.33333333%;
}
```

Рис 2.8. css-властивість, яка буде виконуватися лише якщо активна відповідна точка зупину

У другому випадку, на приведеному малюнку нижче, активна точка зупину з мінімальним значенням в 768px і підповідним значенням CSS-властивості “width” яке дорівнює вже 50%, і саме тому один з наших div елементів буде займати вже

50% від батьківського елемента, отже на одному рядку поміститься лише два таких елементи, а третій переміститься на наступний рядок.

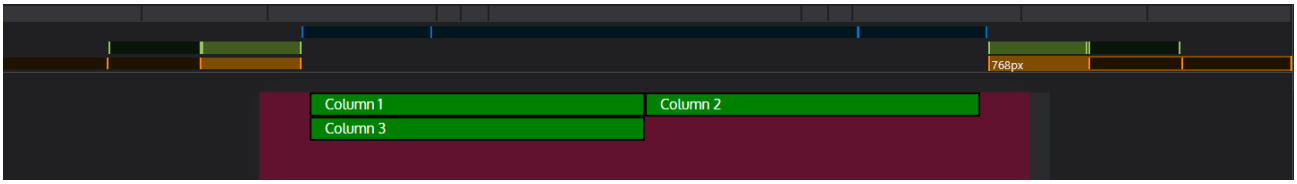


Рис 2.9 активна точка зупину з мінімальним порогом 768px

```
@media (min-width: 768px)
.col-sm-6 {                                grid-framework.less:48
  width: 50%;
}
```

Рис 2.10 відповідна css-властивість, яка виконується при мінальній ширині екрану 768px

І нарешті, у третьому випадку, усі три стовбця будуть займати 100%, тому що ніякі з точок зупину не будуть активними, а тому, за замовчуванням, усі блокові елементи, а у нашому випадку це , будуть займати усю ширину батьківського елементу або вікна браузера, як це показано на малюнку нижче.

Рис 2.11

Але звідкіля саме з’явилися ці CSS-властивості “width”, які задають розміри наших блоків, будь то 33%, 50% або інше значення? Відповідь на це проста – з CSS-класів фреймворку Bootstrap, які ми призначили нашим елементам.

Column 3

Ці класи і містять CSS-властивість “width” яка і змінює ширину наших елементів. Вони знаходяться у окремому CSS файлі фреймворку Bootstrap, який

ми підключаємо за допомгою тегу <link> у шапці нашого html файлу (тег <head>).

```
<head>
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link rel="stylesheet" href="css/bootstrap.css">
  <link rel="stylesheet" href="css/styles.css">
</head>
```

Також у цій же секції нашого html-файлу ми підключаємо css-файл з нашими власними стилями, які будуть перезаписувати вже існуючи стилі фреймворку Bootsrap, якщо буде така необхідність, а тому підключати власні стилі потрубно обов'язкого після стилів Bootstrap, тому що html код винонується послідовно і те саме стосується підключаємих зовнішніх файлів.

Давайте розглянемо більш детально вміст класів Bootstrap які ми використовували для наших елементів:

```
@media (min-width: 992px) {
  .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6, .col-md-7, .col-md-8, .col-md-9, .col-md-10, .col-md-11, .col-md-12 {
    float: left;
@media (min-width: 768px) {
  .col-sm-1, .col-sm-2, .col-sm-3, .col-sm-4, .col-sm-5, .col-sm-6, .col-sm-7, .col-sm-8, .col-sm-9, .col-sm-10, .col-sm-11, .col-sm-12 {
    float: left;
  }
  .col-sm-6 {
    width: 50%;
  }
}
```

Для початку треба зрозуміти що таке саме означає властивість “@media” та текст після цього у дужках.

У CSS, @media є правилом, що використовується для визначення різних стилів для екранів і пристроїв різного розміру(роздільної здатності). Це дозволяє створювати сторінки з адаптивним веб-дизайном , які можуть підлаштовуватися в залежності від пристрою який використовується для перегляду певної веб-

сторінки. Правило @media починається з ключового слова @media, за яким слідує медіа-запит, який визначає властивість, за якої мають застосовуватися стилі всередині правила. Медіа-запит зазвичай включає певну властивість пристрою, наприклад розмір екрана, роздільну здатність або орієнтацію.

Нижче наведено приклад, як @media-правило може бути використано для привласнення окремих стилів, коли екран пристрою, з якого переглядать веб-сторінку, ширше за 768px:

```
@media (min-width: 768px) {
  body {
    font-size: 18px;
  }
}
```

У цьому прикладі, @media правило активується тоді, коли ширина екрану дорівнює як мінімум 768px, і лише у випадку якщо пристрій відповідає цих критерії, то тільки тоді код у фігурних дужках будет виконуватися.
А саме, якщо пристрій будет з екраном з мінамальною шириною в 768px, то буде застосована CSS-властивість “font-size” яка встановить розмів шрифту 18px для елементу <body> і всіх елементів які належать йому, але лише за умови, якщо ширина екрану пристрою буде як мінімум 768px, у інших випадках @media-правило і усі CSS-властивості всередині фігурних дужок будуть ігноруватися браузером.

Тепер поверномося до аналізу властивостей класів Bootstrap які ми використовували для наших трьох елементів, а саме властивості класів .col-md-4 та col-sm-6. Обидва ці класи мають CSS-властивість “float” зі значенням “left”, “float” може приймати різні значення, але основні, які ми будемо використовувати – це “left”, “right” та “auto”.

За допомогою CSS-властивості “float: left” ми примусово переміщуємо елемент до лівого краю іншого елементу, вмістом якого є поточний елемент. Також будь-який вміст, який слідує за ним, буде обтікати його з правого боку, тобто елемент з властивістю “float: left” не буде резервувати під себе увесь рядок і витісняти наступні два елементи на наступний рядок, як це

відбувається за замовчуванням, але це за умови що усі три елементи будуть мати властивість “float: left”.

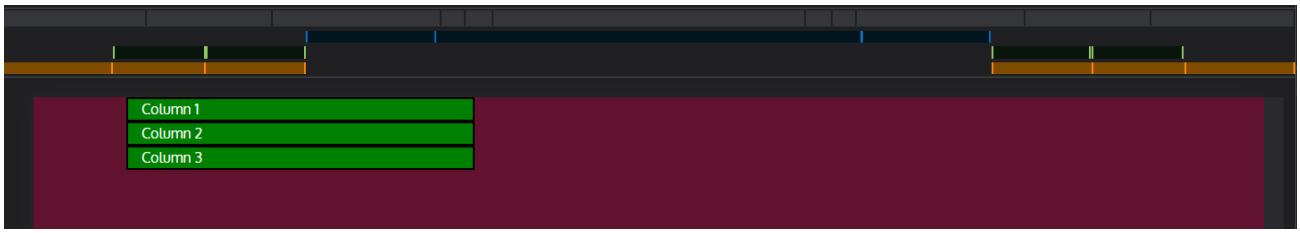


Рис 2.12

На малюнку вище ми можемо переконатися у цьому. Всі елементи мають параметр «width: 33%” і займають третину від від батьківського елементу, але параметр “float: left” деактевовано, і хоча рядок і має місце для ще двох таких самих елементів, перший елемент усе рівно резервує під себе увесь рядок і “виштовхує” наступні елементи які йдуть одразу за ним на наступний рядок, те саме відбувається і з другим елементов, який виштовхует третій елемент на наступний рядок.

Усе це спричинено тим, що за замовчуванням, блокові елементи резервують під себе увесь рядок і не мають привласненого значення CSS-властивості “float”. Але додавши цю властивість, ми змінюємо поведінку елементу за замовчуванням і таким чином перший елемент і наступні за ним групуються у лівій бік веб-сторінки один за одним:

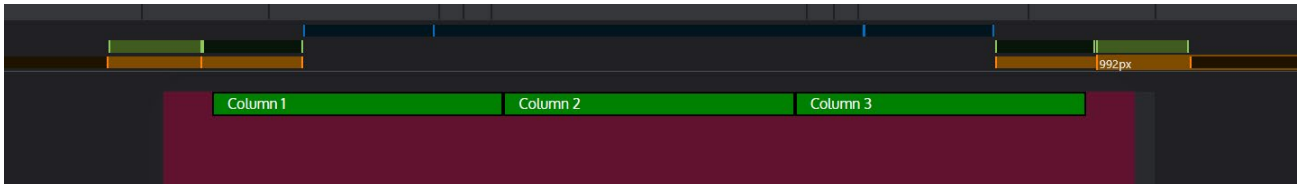


Рис 2.13

```
  }
  .col-md-4 {
    width: 33.33333333%;
  }
}
```

Також варто додати пояснення, як саме наші елементи, маючи два класи(col-sm-6 col-md-4) використовують їх таким чином, що вони не конфліктують між собою.

Як ми знаємо, обидва ці класи містять відповідні @media-правила, для . col-sm-6 – це “min-width: 768px”, а для col-md-4 – “min-width: 992px”. Обидва ці класи ми призначили нашим div-елементам, але тільки один клас буде використовуватися і не буде конфліктувати з іншим, навіть враховуючи те, що @media-правило “min-width: 768px” за логікою повинне виконуватися і тоді, коли буде активне @media-правило нашого другого классу (min-width: 992px). Виконуватися будуть CSS-властивості лише одного класу, тому що у зовнішньому файлі Bootstrap.css оголошення класів з приставкою “md” йде після оголошення класів з приставкою “sm” і тому, коли ширина екрана чи браузера більша за 992px, то клас з приставкою “sm” виконувався бы все одно, але тому що ми маэмо другий клас який оголошену після нього, то він перезаписує CSS-властивості попереднього класу.

Крім усього вище сказаного, Bootstrap також надає кілька попередньо розроблених компонентів, таких як навігаційні меню, форми та таблиці, які вже оптимізовані для адаптивного дизайну. Це допомагає забезпечити належне відображення цих компонентів на будь-якому пристрої без необхідності додаткового кодування чи налаштування.

Загалом, Bootstrap надає потужний набір інструментів і компонентів, які спрощують створення адаптивних і орієнтованих на мобільні пристрої веб-інтерфейсів, оптимізованих для зручності використання та доступності перегляду веб-сайтів на різних пристроях.

2.4. Опис використаних технологій та мов програмування

У процесі розробки веб-сайту вікористовувалися наступні мови програмування та технології:

- HTML;
- CSS;
- Javascript;
- AJAX;
- SAP;
- HTTP;
- JSON;

HTML (HyperText Markup Language) — стандартна мова розмітки, яка використовується для створення та структурування вмісту веб-сторінок. Вона складається з набору тегів або елементів, які визначають структуру та представлення інформації у веб-документі. Теги HTML використовуються для розмітки різних типів вмісту, наприклад заголовків, абзаців, зображень, посилань тощо, що дозволяє веб-переглядачам інтерпретувати та відображати вміст відповідно.

CSS (каскадні таблиці стилів) — це мова таблиць стилів, яка використовується для опису візуального представлення документа, написаного в HTML або XML. Вона надає набір правил і властивостей, які визначають, як мають відображатися елементи на веб-сторінці, включаючи їх макет, кольори, шрифти, розміри та інші візуальні характеристики. Відокремлюючи презентаційний рівень від структури документа, CSS забезпечує легкий і послідовний стиль для кількох веб-сторінок і дозволяє створювати візуально привабливі та адаптивні дизайни для веб-сайтів.

JavaScript — це інтерпретована мова програмування високого рівня, яка в основному використовується для додавання інтерактивності та динамічних функцій веб-сторінок. Це дозволяє розробникам створювати інтерактивні елементи, маніпулювати та змінювати вміст веб-сторінки, обробляти події, виконувати обчислення, надсилати мережеві запити та багато іншого. JavaScript зазвичай використовується у веб-розробці для покращення взаємодії з користувачем і створення динамічних інтерактивних веб-додатків.

AJAX (асинхронний JavaScript і XML) — це техніка, яка використовується в JavaScript для асинхронного надсилання та отримання даних із сервера без втручання в поточну сторінку. Це дозволяє оновлювати частини веб-сторінки, не вимагаючи повного перезавантаження сторінки. Традиційно, коли користувач взаємодіє з веб-сторінкою, браузер надсилає запит на сервер, який у відповідь надсилає нову сторінку HTML. Цей процес спричиняє перезавантаження всієї сторінки, що призводить до помітної затримки та менш плавної взаємодії з користувачем.

За допомогою AJAX JavaScript може робити асинхронні запити до сервера у фоновому режимі, не перериваючи взаємодії користувача зі сторінкою. Це досягається за допомогою об’єкта XMLHttpRequest або нових методів, таких як fetch API. Сервер може відповідати даними в різних форматах, включаючи XML, JSON або простий текст.

Як тільки сервер відповідає, JavaScript може обробляти дані та динамічно оновлювати певні частини сторінки. Це може включати завантаження нового вмісту, надсилання даних форми або виконання інших дій без перезавантаження всієї сторінки.

За допомогою Javscript наш сайт буде працювати як SPA (Single Page Application) чи Односторінкова програма. Суть такого веб-сайту полягає в тому що він містить усього одну веб-сторінку, а необхідні її фрагменти будуть завантажуватися з сервера та динамічно за допмогою JS та технології AJAX вставлятися усередину існуючого HTML–коду чи оновлювати його. Таким чином, коли користувач буде натискати на будь-яке посилання, то з серверу буде завантажуватися не окрема html-сторінка, а лише частина сторінки яку потрібно оновити. Цей підхід забезпечує плавніший і оперативніший досвід роботи з користувачем, оскільки лише необхідні дані витягуються із сервера, а переходи сторінок обробляються на стороні клієнта без необхідності повного перезавантаження сторінки.

HTTP, скорочення від Hypertext Transfer Protocol, — це протокол зв’язку, який широко використовується для передачі гіпертекстових і гіпермедійних

документів у Всесвітній павутині. Це основа передачі даних для Інтернету та дозволяє веб-браузерам і веб-серверам спілкуватися та обмінюватися інформацією.

HTTP працює як клієнт-серверний протокол, де клієнт, як правило, веб-браузер, надсилає запит серверу, а сервер відповідає запитаними даними. Ці дані можуть включати HTML-документи, зображення, відео або будь-який інший тип вмісту, доступний в Інтернеті.

Протокол працює за моделлю запит-відповідь, де клієнт ініціює запит, вказуючи метод (наприклад, GET, POST, PUT, DELETE) разом із URL-адресою (уніфікованим покажчиком ресурсу) потрібного ресурсу. Сервер обробляє запит і генерує відповідну відповідь, яка включає код статусу, що вказує на успішне або невдале виконання запиту, і запитувані дані, якщо це можливо.

HTTP базується на протоколі без стану та з'єднання, що означає, що кожна взаємодія запиту та відповіді не залежить від попередніх чи майбутніх взаємодій. Цей дизайн забезпечує масштабованість і простоту, але вимагає додаткових механізмів, таких як файли cookie або сеанси, для підтримки стану та забезпечення більш складної взаємодії.

Згодом було розроблено різні версії HTTP, причому HTTP/1.1 донедавна була найпоширенішою версією. Впровадження HTTP/2, а пізніше HTTP/3 покращило продуктивність, безпеку та ефективність завдяки введенню таких функцій, як мультиплексування, натискання на сервер і покращену обробку одночасних запитів.

JSON розшифровується як JavaScript Object Notation, це легкий формат обміну даними, який широко використовується для представлення структурованих даних. Це текстовий формат, який легко читати та писати людям, а також машинам легко аналізувати та генерувати. JSON часто використовується як формат даних для передачі та зберігання даних у веб-додатках.

JSON базується на колекції пар ключ-значення та структур даних, таких як масиви та об'єкти. Він тісно пов'язаний із синтаксисом мови програмування

JavaScript, але не обмежується JavaScript і може використовуватися з різними мовами програмування.

У JSON дані організовані в об’єкти, які взяті у фігурні дужки {}. Кожен об’єкт складається з однієї або кількох пар «ключ-значення», де ключ — це рядок, укладений у подвійні лапки “”, а значенням може бути будь-який дійсний тип даних JSON, зокрема рядки, числа, логічні значення, значення null, масиви або вкладені об’єкти.

Окрім об’єктів, JSON також підтримує масиви, які є впорядкованими наборами значень, укладених у квадратні дужки []. Масиви можуть містити кілька типів даних JSON і можуть бути вкладеними для створення складніших структур.

JSON забезпечує спосіб представлення структурованих даних у переносимий і незалежний від платформи спосіб. Він зазвичай використовується для обміну даними між клієнтом і сервером у веб-додатках, а також для зберігання параметрів конфігурації, відповідей API та інших форм серіалізації даних. Простота та універсальність JSON зробили його популярним вибором у сучасній розробці програмного забезпечення.

2.5. Опис структури програми та алгоритмів її функціонування

Структура веб-додатку складається з декілької секцій головної сторінки, а також інших веб-сторінок які будуть динамічно завантажуватись до головної секці веб-сторінки яка знаходиться посередині та оновлювати її вміст (технологія SAP).

Загалом сайт буде містити три секції:

- header – містить лого сайту, назву, навігаційну панель а також кнопку-посилання для здійснення дзвінка до рестарану;
- головна секція усередині елемента з Bootstrap-класом “main-content” - містить фонове зображення та 3 посилання у вигляді плиток, “Меню”, “Особливі страви”, а також плитку з відображенням розташуваня ресторану на на Google Maps;
- footer – містить адресу закладу, його графік роботи а також відгуки

клієнтів;

Нижче на рисунку наведено діаграму потенційних дій користувача при переході на головну сторінку сайту:

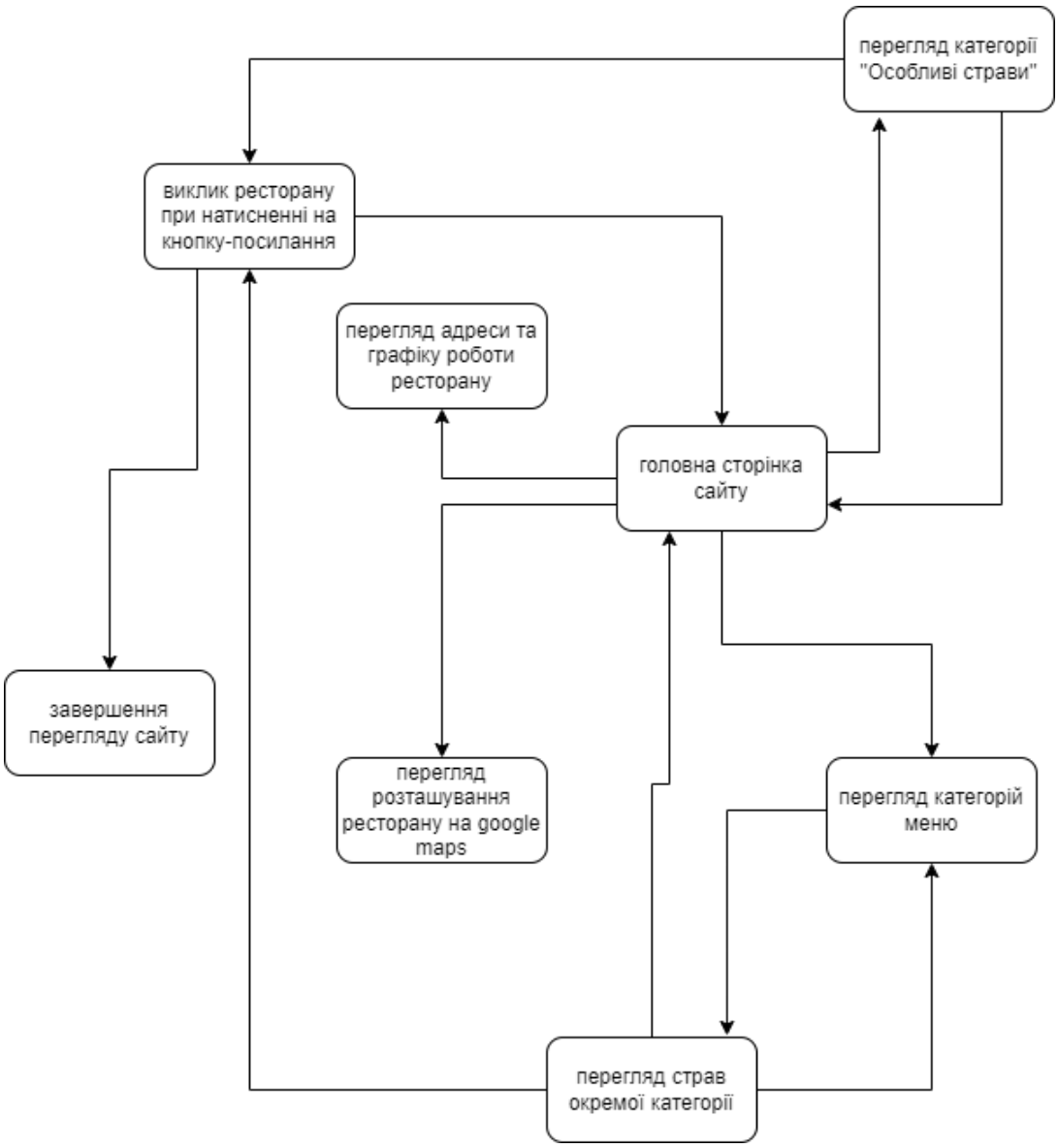


Рис 2.14. Діаграма потенційних дій користувача веб-додатка

Веб-додаток буде працювати за моделлю SAP, тобто складитя лише з однієї сторінки, яка буде оновлювати свій вміст відповідно до дій користувача. Головна сторінка буде динамічно оновлюватися за допомогою JS функцій та технології AJAX. Усередині неї знаходиться елемент з id “main-content”, який відповідає за головну секцію, вміст якої і буде оновлюватися динамічно. Наступні

JS-функції відповідають за відправлення запиту серверу для отримання html-сніпетів (частин коду) для подальшого вставляння цього коду за допомогою Javascript. Також за допомогою цих самих функцій будуть отримуватись дані з файлів .json при відправленні запитів до сервера.

```
sendGetRequest = function (requestUrl, responseHandler, isJsonResponse) {
   var request = new XMLHttpRequest();
   request.onreadystatechange = function() {
      handleResponse(request, responseHandler, isJsonResponse);
    };
   request.open("GET", requestUrl, true);
   request.send(null);
 };
function handleResponse(request, responseHandler, isJsonResponse) {
 if ((request.readyState == 4) && (request.status == 200)) {
   // Default to isJsonResponse = true
   if (isJsonResponse == undefined) {
    isJsonResponse = true;
   }
   if (isJsonResponse) {
    responseHandler(JSON.parse(request.responseText));
   }
   else {
    responseHandler(request.responseText);
   }
 }
}
```

Під час виклику цієї функції, їй передається у вигляді аргументу шлях до сніпету з html-кодом який потрібно отримати з серверу, для головної сторінки це файл “home-snippet.html”, він має наступний код:

```
<div class="jumbotron">
  <img src="images/jumbotron_768.jpg" alt="Picture of restaurant" class="img-responsive visible-
xs">
</div>
<div id="home-tiles" class="row">
  <div class="col-md-4 col-sm-6 col-xs-12">
    <a href="#" onclick="loadMenuCategories();">
      <div id="menu-tile"><span>меню</span></div>
    </a>
  </div>
  <div class="col-md-4 col-sm-6 col-xs-12">
    <a href="single-category.html">
      <div id="specials-tile"><span>особливі страви</span></div>
    </a>
  </div>
  <div class="col-md-4 col-sm-12 col-xs-12">
    <a href="https://goo.gl/maps/GbjmFBcFShmUMw4TA?coh=178572&entry=tt" target="_blank">
```

```
    <div id="map-tile">
      <iframe
src="https://www.google.com/maps/embed?pb=!1m13!1m8!1m3!1d165.35299790239148
          !2d35.0492271616566866!3d48.46328362205484!3m2!1i1024!2i768!4f13.1!3m2!1m1!2z
          NDjCsDI3JzQ3LjciTiAzNcKwMDInNTcuNyJF!5e0!3m2!1sru!2sua!4v1683219856628!5m
2!1sru!2sua" width="100%"
        height="250" style="border:0;" allowfullscreen="" loading="lazy"
        referrerpolicy="no-referrer-when-downgrade"></iframe>
      <span>карта</span>
    </div>
  </a>
 </div>
</div>
```

Далі викликається функція яка отримає цей код та вставить його усередину елемента з id “main-content”.

```
var homeHtml = "snippets/home-snippet.html";
sendGetRequest(homeHtml, function (responseText) {
    document.querySelector("#main-content").innerHTML = responseText; }, false);
```

Після цього головна сторінка матиме наступний вигляд:

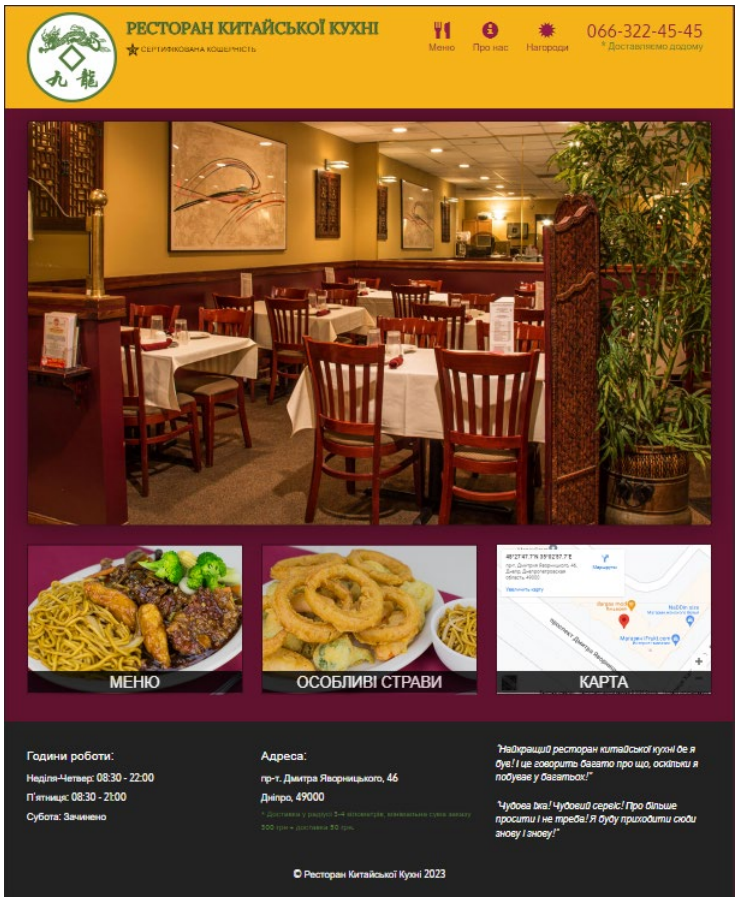


Рис 2.15. Вигляд головної сторінки

Для порівняння, якщо закоментувати використаний JS-код, то після завантаження головної сторінки div-елемент “main-content” залишиться пустим і сайт буде відображати лише секції header та footer.

Рис 2.16. Вигляд сторінки яка має лише секцію header та footer

При натисканні на кнопку “Меню” буде показан перелік категорій меню страв ресторану. Перелік категорій завантажується під час відправлення запиту до сервера з вказанням шляху до файла з розширенням .json. Він містить повну та коротку назву кожної категорії.

```
{
  "name": "Закуски",
  "short_name": "В",
}
```

Повна назва потрібна для відображення назви категорії на сторінці, коротка, яка складається з однієї букви потрібна для завантаження зображення яке відповідає конкрутній категорії а також для виклику функції яка буде завантажувати перелік страв після вибору конкретної категорії.

Сніпет який відповідає за одну категорію має наступний html-код:

```
<div class="col-md-3 col-sm-4 col-xs-6 col-xxs-12">
  <a href="#" onclick="loadMenuItems('{{short_name}}');">
    <div class="category-tile">
      <img width="200" height="200" src="images/menu/{{short_name}}/{{short_name}}.jpg"
alt="{{name}}">
      <span>{{name}}</span>
    </div>
  </a>
</div>
```

У цьому сніпеті як можна побачити присутній текст-заповнювач(placeholder) з ключами “name” та “short_name”, ці участки коду будуть замінені значеннями з .json файлу під час виклику нижче описаної функції.

Далі, коли користувач натисне на кнопку-посилання “Меню” на навігаційній панелі чи у головніій секції, буде викликана функція для отримання цього html-сніпету з сервера та вставляння його усередину з id “main-content”, а також будуть отримані дані з .json файлу для того щоб підставити значення ключів “name” та “short_name” усередину html-коду.

```
 var allCategoriesUrl = "data/categories.json";
 var categoriesTitleHtml = "snippets/categories-title-snippet.html";
 var categoryHtml = "snippets/category-snippet.html";

 var insertHtml = function (selector, html) {
  var targetElem = document.querySelector(selector);
  targetElem.innerHTML = html;
 };

 var insertProperty = function (string, propName, propValue) {
  var propToReplace = "{{" + propName + "}}";
  string = string.replace(new RegExp(propToReplace, "g"), propValue);
  return string;
 };

  loadMenuCategories = function () {
   sendGetRequest(allCategoriesUrl, buildAndShowCategoriesHTML);
  };

 function buildAndShowCategoriesHTML(categories) {
  sendGetRequest(categoriesTitleHtml, function (categoriesTitleHtml) {
     sendGetRequest(categoryHtml, function (categoryHtml) {
        var categoriesViewHtml = buildCategoriesViewHtml(categories,
categoriesTitleHtml,categoryHtml);
        insertHtml("#main-content", categoriesViewHtml);
       }, false);
    },
    false
  );
 }

 function buildCategoriesViewHtml(categories, categoriesTitleHtml, categoryHtml) {
  var finalHtml = categoriesTitleHtml;
  finalHtml += "<section class='row'>";
  for (var i = 0; i < categories.length; i++) {
```

```
    var html = categoryHtml;
    var name = "" + categories[i].name;
    var short_name = categories[i].short_name;
    html = insertProperty(html, "name", name);
    html = insertProperty(html, "short_name", short_name);
    finalHtml += html;
  }
  finalHtml += "</section>";
  return finalHtml;
}
```

Після виклику фнукції loadMenuCategories буде відображено перелік категорій:

Рис 2.17. Сторінка з переліком категорій меню

На цій сторінці перелічені усі категорії меню страв ресторану. Коли потенційний клієнт обере одну з категорій та натисне на неї, буде викликана функція яка завантажить перелік усіх страв окремої категорії. Перелік страв оеркмої категорії буде завантажуватися так само з файлу .json як і перелік категорій. Для завантаження переліку страв нам звону знадобиться сніпет з html-кодом, відповідаючий за 1 страву з її описом, усередині якого ми будемо замінювати код усередині фігурних скобок значеннями які буде отримано з .json файлу.

```
<div class="menu-item-tile col-md-6">
  <div class="row">
    <div class="col-sm-5">
      <div class="menu-item-photo">
        <div>{{short_name}}</div>
        <img class="img-responsive" width="250" height="150"
src="images/menu/{{catShortName}}/{{short_name}}.jpg" alt="Item">
      </div>
      <div class="menu-item-price">{{price_small}}<span> {{small_portion_name}}</span>
{{price_large}} <span>{{large_portion_name}}</span></div>
    </div>
    <div class="menu-item-description col-sm-7">
      <h3 class="menu-item-title">{{name}}</h3>
      <p class="menu-item-details">{{description}}</p>
    </div>
  </div>
  <hr class="visible-xs">
</div>
```

Далі пісдя вибору користувачем окремої категорії буде викликана функція loadMenuItems і функції які викликає вона, значення у фігурних дужках(placeholders) будуть замінені отриманими значеннями з файлу ,json і буде завантажено та відображено перелік страв окремої категорії. Також кожна страва буде мати окремий опис і ціну за малу порцію (100г) та велику порцію (500г).

```
var menuItemsTitleHtml = "snippets/menu-items-title.html";
var menuItemHtml = "snippets/menu-item.html";
var menuItemsUrl = "data/menu_items/";
```

```
loadMenuItems = function (categoryShort) {
  sendGetRequest(
    menuItemsUrl + categoryShort + ".json",
    buildAndShowMenuItemsHTML
  );
};

function buildAndShowMenuItemsHTML(categoryMenuItems) {
  sendGetRequest(menuItemsTitleHtml,function (menuItemsTitleHtml) {
      sendGetRequest(menuItemHtml, function (menuItemHtml) {
          var menuItemsViewHtml =
buildMenuItemsViewHtml(categoryMenuItems,menuItemsTitleHtml,menuItemHtml);
          insertHtml("#main-content", menuItemsViewHtml);},false);
    },false);
}

function buildMenuItemsViewHtml(categoryMenuItems,menuItemsTitleHtml,menuItemHtml) {
  menuItemsTitleHtml =
insertProperty(menuItemsTitleHtml,"name",categoryMenuItems.category.name);
  menuItemsTitleHtml =
insertProperty(menuItemsTitleHtml,"special_instructions",categoryMenuItems.category.special_instr
uctions);
  var finalHtml = menuItemsTitleHtml;
  finalHtml += "<section class='row'>";
  var menuItems = categoryMenuItems.menu_items;
  var catShortName = categoryMenuItems.category.short_name;
  for (var i = 0; i < menuItems.length; i++) {
    var html = menuItemHtml;
    html = insertProperty(html, "short_name", menuItems[i].short_name);
    html = insertProperty(html, "catShortName", catShortName);
    html = insertItemPrice(html, "price_small", menuItems[i].price_small);
    html = insertItemPortionName(html,"small_portion_name",menuItems[i].small_portion_name);
    html = insertItemPrice(html, "price_large", menuItems[i].price_large);
    html = insertItemPortionName(html,"large_portion_name",menuItems[i].large_portion_name);
    html = insertProperty(html, "name", menuItems[i].name);
    html = insertProperty(html, "description", menuItems[i].description);
    if (i % 2 != 0) {
      html += "<div class='clearfix visible-lg-block visible-md-block'></div>";
    }
    finalHtml += html;
  }
  finalHtml += "</section>";
  return finalHtml;
}

function insertItemPrice(html, pricePropName, priceValue) {
  if (!priceValue) {
    return insertProperty(html, pricePropName, "");
  }
  priceValue = priceValue.toFixed(2) + " грн";
  html = insertProperty(html, pricePropName, priceValue);
  return html;
```

```
}

function insertItemPortionName(html, portionPropName, portionValue) {
  if (!portionValue) {
    return insertProperty(html, portionPropName, "");
  }
  portionValue = "(" + portionValue + ")";
  html = insertProperty(html, portionPropName, portionValue);
  return html;
}
```

Після виклику фнукції loadMenuItems буде відображено перелік усіх страв окремої категорії.

Рис 2.18. Сторінка з переліком страв окремої категорії

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Программа отримує вхідні дані у вигляді натискання користувача по одному з посилань на сторінці після чого в залежності від того на яке саме посилання було натиснуто, завантажується необхідна інформація. Ця інформація є вихідними даними. До неїї належать перелік страв при виборі якоїсь окремої категорії, а такоє перелік категорій при натисканні на кнопку-послиання “Меню” яка знаходиться на навігаційній панелі а також у вигляді плитки з фоновим зображенням посередині сторінки.

В залежності від того на яке послиання натисне користувач, також завантажуються дані з сервера які знаходяться усередині .json файлу. Ці дані відповідають за перелік страв, їх назву, опис, ціну за малу ма велику порцію. Також цей файл містить перелік усіх категорій меню ресторану. Дані які знаходяться усередині .json файлу, а потім вставляються усередину html-кода, також є вихідними даними від яких залежить кінцевий вид сторінки яка буде відображена користувачу.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Веб-додаток не є вимогливим до технічних характеристик пристрою з якого він будет завантажуватися, будь це ПК, ноутбук, планшет чи смартфон під управлінням одної з сучасних операційниї систем. Для завантаження веб-додатку використовується лише браузер пристрою.

Додаток було розроблено на ПК під управлінням операційної системи Windows 10, характеристики цього ПК наведені нижче:

- операційна система Windows 10;
- процесор Intel® Xeon® E3-1230 v2 3.3GHz-3.7GHz;
- материнська плата Gigabyte GA-B75-D3V (1 x PCI-E 3.0 x16, 1 x PCI-E

3.0 x16 (x4), 4 x DDR3 DIMM, Audio, Video, Gigabit LAN);

- 4 модулі оперативної пам’яті, 3 планки виробництва Samsung та одна планка виробництва Goodram, усі працюють на частоті 1600 МГц.
- відеоадаптер NVIDIA GeForce GTX 1070 (8 ГБ);
- HDD-диск Western Digital WDC WD5000AADS-00S9B0 на 500ГБ та SSD-диск ADATA XPG GAMMIX S11 Pro на 512ГБ;
- монітор з роздільною здатністю 1920 x 1080 пікселів;
- пристрої вводу: USB-миша та PS/2 клавіатура;
- доступ до мережі інтернет 100 МБіт/сек;

2.7.2. Використані програмні засоби

Під час розробки веб-додатка були використані наступні програмні засоби:

- Visual Studio Code – для роботи з кодом;

VS Code — це легкий редактор вихідного коду з можливістю налаштування, розроблений Microsoft. Він пропонує простий та інтуїтивно зрозумілий інтерфейс, відмінну підтримку мови з такими функціями, як IntelliSense, і широку екосистему розширень. Він включає інтегрований термінал, можливості налагодження, інтеграцію контролю версій, автоматизацію завдань і підтримує співпрацю в реальному часі за допомогою Live Share. Він широко використовується розробниками завдяки своїй універсальності та функціям підвищення продуктивності.

- браузер Google Chrome – для перегляду отриманого результату, відладки JS-коду, зміни деяких значень «на льоту» та миттєвий перегляд результату змін, а також для тестування додатку з екранами різних пристроїв за допомогою функції симуляції різних пристроїв.

Google Chrome – це веб-браузер, який пропонує низку функцій та інструментів, розроблених для задоволення потреб розробників. Він забезпечує надійне середовище розробки за допомогою вбудованих

інструментів розробника, що дозволяє розробникам перевіряти та змінювати веб-сторінки, налагоджувати код JavaScript, контролювати мережеву активність та оптимізувати роботу веб-сайту. Широке застосування Chrome робить його необхідним для тестування крос-браузерної сумісності, гарантуючи безперебійну роботу веб-програм на різних платформах. Екосистема розширень браузера пропонує додаткові інструменти для підвищення продуктивності для редагування коду, контролю версій і робочих процесів веб-розробки. Chrome також підтримує прогресивні веб-програми (PWA), що дозволяє розробникам створювати веб-програми з автономною функцією та доступом до функцій пристрою. Google надає розробникам розширену документацію, навчальні посібники та ресурси, а також активно взаємодіє зі спільнотою через форуми та події.

– Операційна система Windows 10;

2.7.3. Виклик та завантаження програми

Для перегляду сторінки з веб-додатком потрібно лише ввести URL адресу сторінки у адресний рядок браузера. Для текстування та розробки, веб-додаток було розгорнуто на локальному сервері за допомогою розширення “Live-server” усередині середовища VS Code.

2.7.4. Опис інтерфейсу користувача

Враховуючи те ще, що сайт побудовано за допоомогою фреймворку Bootstrap, то при його перегляді з різних пристроїв він будет мати різний вигляд, адаптований під розмір екрану.

Для прикладу нижче наведено декілька зображень які демонструють вигляд головної сторінки при перегляді з різних пристроїв. Для того щоб симулювати екрани різних пристроїв використовувся браузер Chrome та функція “device mode” яку можна увімкнути увійшовки до панелі розробника (F12) та натиснувши кнопку “device toggle toolbar” або одночасно клавіші Ctrl+Shift+M.

Рис 2.19. Інтерфейс головної сторінки

Такий вигляд має головна сторінка при перегляді з ноутбуку, ПК чи іншого пристрою що має екран більше середнього розміру.

На рисунку нижче зображено який вигляд має та сама сторінка але вже при перегляді з планшету Apple Ipad.

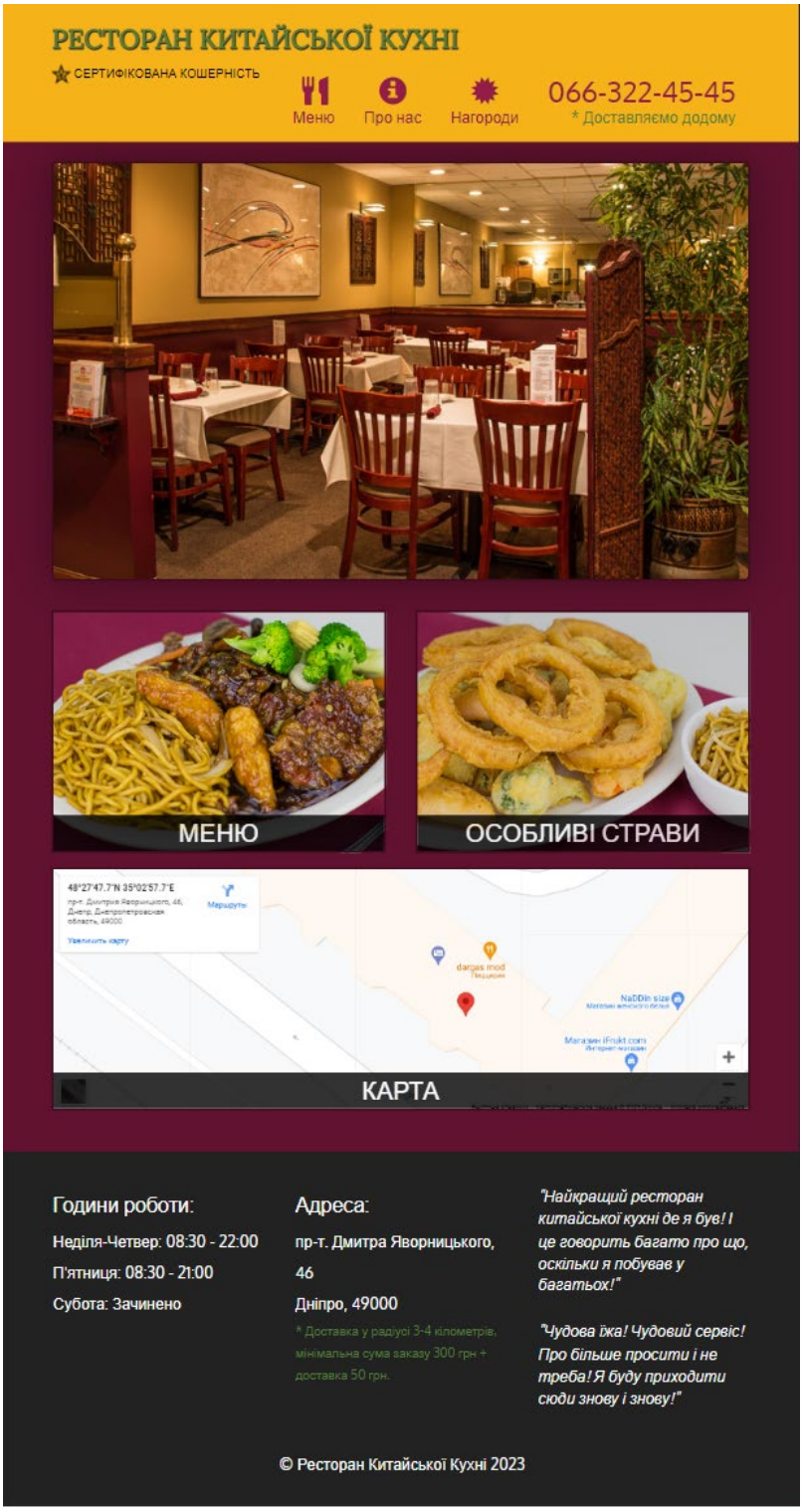


Рис 2.20. Виглчд сторінки при перегляді з планшету

Як можна побачити, тепер сайт не відображає лого, це зроблено для економії місця. Так як розмір екрану зменшився, то місця для розташування усіх елементів усередині секції “header” також стало менше, а тому було прийнято рішення під час розробки додати CSS-правило яке буде приховувати лого під час перегляду сайту з планшетів чи мобільних пристроїв.

Також сайт змінив вигляд у головній секції що має три плитки, “Меню”, “Особливі страви” та плитку що містить фрейм з розташування ресторану на Google Maps. Під час перегляду з ноутбуку чи настільного комп’ютера усі три плитки знаходилися у одному рядку, тепер ми спостерігаємо що у одному рядку знаходяться лише дві плитки, а третя знаходиться на наступному рядку. Така поведінка була досягнута використанням спеціальних Bootstrap css-класів які працюють за «системою сітки з 12 стовбців». До усіх трьох плиток плиток був застосований клас «col-md-4» що означає відображення трьох елементів у одному рядку під час перегляду сайтів з пристроїв які мають екран більше середнього(ноутбуки, ПК). Також до перших двох плиток було застосовано клас «col-sm-6» а до останньої що містить що містить фрейм з Google Maps клас «col-sm-12». Це і є ключовим моментом у відображенні цих елементів, тому як цифра 6 у назві Bootstrap-класу вказує на те що елемент буде займати половину ширини батьківського елемента чи вікна браузера, а цифра 12 – усю ширину. Тому перші два елемента знаходятся на одному рядку, а третій на наступному під час перегляду з планшету.

Нижче на рисунку наведено приклад як виглядає сайт при перегляді з смартфону Samsung Galaxy S8+.

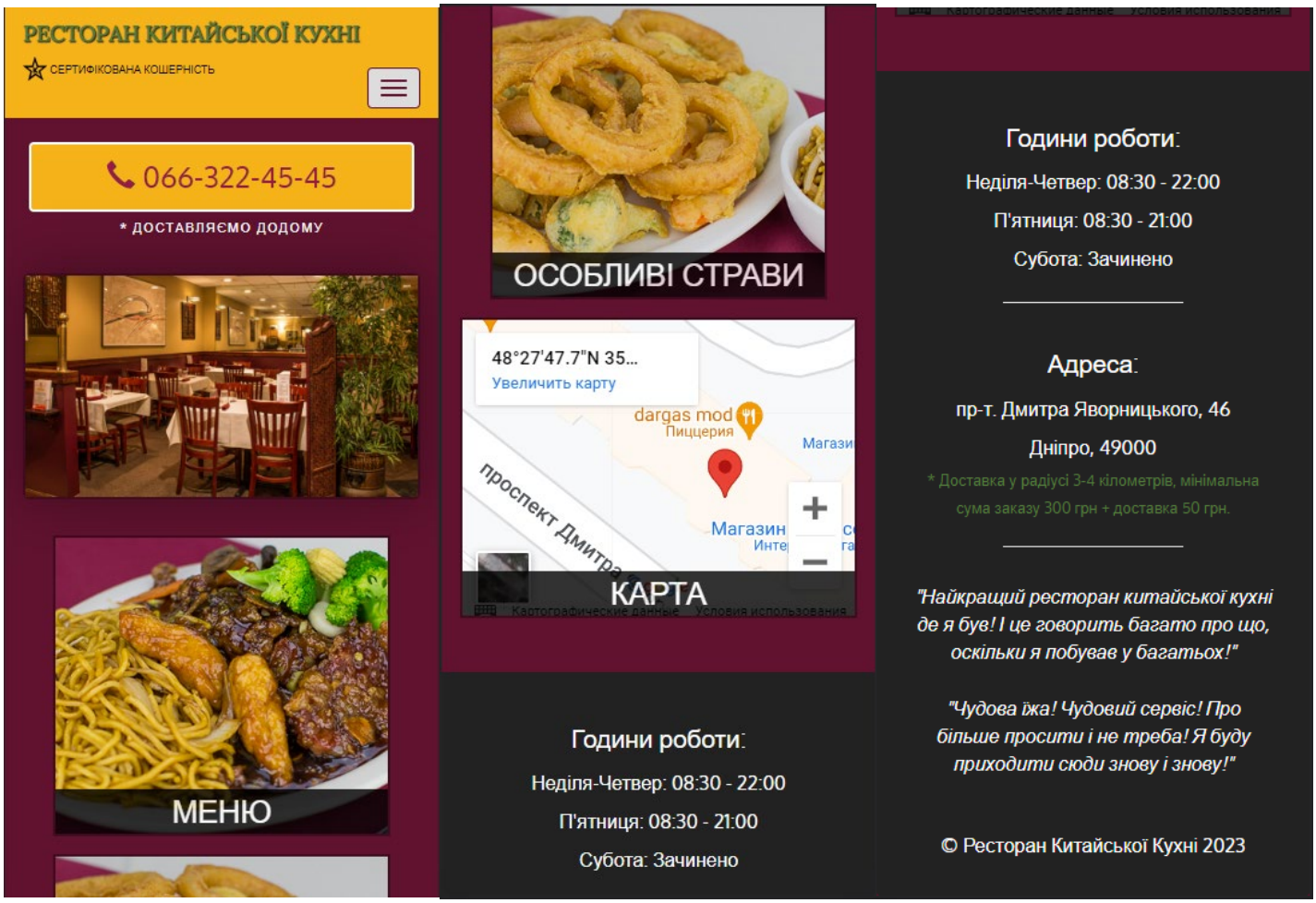
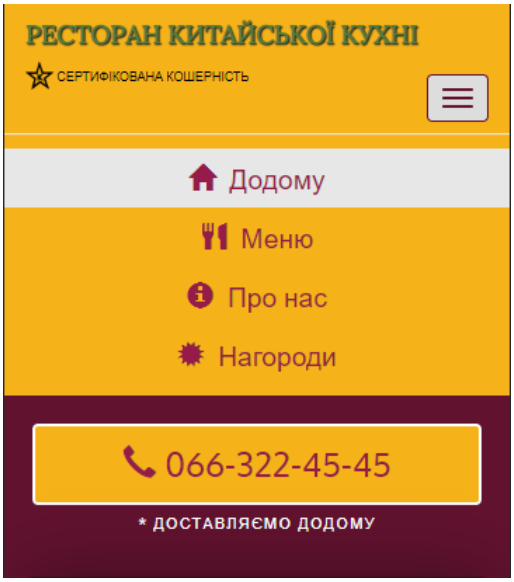


Рис 2.21. Виглчд сторінки при перегляді з мобільного пристрою

Під час перегляду сайту з мобільного пристрою з’являється кнопка яка відповідає за розгортання навігаційного меню.



Для того щоб при натисканні на цю кнопку з’являлось випадюче меню, ми використали Bootstrap-клас “collapse” а також у атрибуті “target” нашої кнопки вказали елемент який відповідає за усі пунки навігаційного меню, а саме це елемент <ul> який містить усі пункти, а кожен окремий пункт знаходиться усередині елементів <li>.

Також для того щоб меню згорталося при натисканні поза його межами була додана наступня JS-функція:

```
document.addEventListener("DOMContentLoaded", function (event) {
  document.querySelector("#navbarToggle").addEventListener("blur", function (event) {
    var screenWidth = window.innerWidth;
    if (screenWidth < 768) {
      $("#collapsable-nav").collapse('hide');
    }
  });
});
```

Також варто додати, що кнопку-посилання з номером телефону було винесено за межі навігаційної панелі під час перегляду з мобільного пристрою. Це було досягнуто вкористанням Bootstrap-класів “visible-xs” для кнопки яка видна під час перегляду з мобільного пристрою та знаходиться поза межами навігаційної панелі, а також класу “hidden-xs” для кнопки яка видна під час перегляду з ПК чи планшету для того щоб приховату кнопку при натисканні на кнопку випадаючого меню.

РОЗДІЛ 3
ЕКОНОМІЧНІЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми – 1250;
2. Коефіцієнт складності програми – 1,3;
3. Коефіцієнт корекції програми в ході її розробки – 0,08;
4. Годинна заробітна плата програміста– 125 грн/год;

Заробітна плата була вирахувана виходячи з даних наданих сайтом dou.ua («Українська спільнота програмістів»). Виходячи з цих даних, станом на грудень 2022 року, середня медіанна заробітна плата Junior JavaScript Developer з досвідом роботи до 1 року становить 600 доларів США. При курсі валют НБУ на початок травня 2023 року один американський долар дорівнює 36,56 грн, тому середня медіанна зарплата в гривнях дорівнює 21936 грн. При нормованому 8-годинному графіку, обсяг робочих годин у місяць буде становити 176. З цього виходить що погодинна зарплатня буде становити 125 гривень.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0.9;
7. Вартість машино-години ЕОМ – 10 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за наступною формулою:

$$t = t_O + t_U + t_A + t_N + t_{отл} + t_Д, \text{ людино-годин}, \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \quad (3.2)$$

де q – передбачуване число операторів (1250);

C – коефіцієнт складності програми (1,3);

p – коефіцієнт корекції програми в ході її розробки (0,08).

Підставивши необхідні значення у формули отримуємо наступний результат:

$$Q = 1250 \cdot 1{,}3 \cdot (1 + 0{,}08) = 1755$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин}, \quad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2);

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. Враховуючи, що стаж програміста становить до 1 року, то коефіцієнт дорівнює 0.9.

Підставивши необхідні значення в формулу (3.3), отримаємо результат витрат праці на вивчення опису задачі:

$$t_u = \frac{1755 \cdot 1{,}2}{85 \cdot 0.9} = 27{,}52, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25)*k}, \text{ людино-годин, } (3.4) \mid$$

Підставимо необхідні значення у формулу (3.4) та отримаємо результат витрат праці на розробку алгоритму рішення задачі:

$$t_a = \frac{1755}{20 \cdot 0{,}9} = 97{,}5, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) \cdot k}, \text{ людино-годин.} \quad (3.5)$$

$$t_n = \frac{1755}{25 \cdot 0{,}9} = 78, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4...5) \cdot K}; \qquad (3.6)$$

$$t_{\text{отл}} = \frac{1755}{5 \cdot 0.9} = 390, \text{ людино-годин}$$

– за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1,2 \cdot t_{отл;} \quad (3.7)$$

$$t^{к}_{отл} = 1,2 \cdot 390 = 468 \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до} \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису;

$$t_{д} = \frac{Q}{(15...20) \cdot K}; \quad (3.9)$$

$$t_{д} = \frac{1755}{20 \cdot 0.9} = 97.5 \text{ людино-годин,}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації;

Підставимо необхідні значення та отримаємо наступний результат:

$$t_{до} = 0,75 \cdot t_{др} ; \quad (3.10)$$

$$t_{до} = 0,75 \cdot 97,5 = 73,12, \text{ людино-годин.}$$

$$t_{д} = 97,5 + 73,12 = 170,62, \text{ людино-годин.}$$

Використовуючи формулу (3.1) вирахуємо трудомісткість розробки програмного забезпечення:

t = 50 + 27,52 + 97,5 + 78 + 390 + 170,62 = 813,64, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно близько 813,64 людино-годин для розробки даного програмного забезпечення.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $К_{по}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{по} = З_{зп} + З_{мв}\text{ , грн,} \qquad (3.11)$$

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{зп} = t \cdot С_{пр}\text{ , грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{пр}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата програміста становить 125 грн/год, отримуємо:

$$З_{зп} = 813{,}64 \cdot 125 = 101705 \text{ грн.}$$

Вартість машинного часу $З_{мв}$, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{мч}$ – вартість машино-години ЕОМ, грн/год.

Підставввимо у формулу (3.13) необхідні значення та визначимо вартість часу необхідного для налагодження програми на ЕОМ:

$$З_{мв} = 390 \cdot 10 = 3900 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{по} = 101705 + 3900 = 105605 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \quad \text{,міс}$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Витрати на створення програмного продукту:

$$T = \frac{813{,}64}{1 \cdot 176} = 4{,}62 \text{ міс.}$$

Висновки. Веб-додаток має вартість 105605 грн. Ймовірний очікуваний час розробки – 4,62 місяців при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Цей термін пов’язаний з кількістю операторів і включає в себе час для дослідження та розробку алгоритму розв’язання задачі, розробку дизайну і створення документації. На розробку веб-додатку буде витрачено 813,64 людино-годин.

ВИСНОВКИ

Під час виконання цього дипломного проекту було створено зручний та практичний веб-додаток, у його створенні допомогли технології HTML/CSS/JS, а також фреймворк Bootstrap.

Веб-додаток було створено з ціллю покращення сервісу потенційних клієнтів, а також для розширення клієнтської бази. Завдяки наявності сайту, потенційні клієнти ресторану можуть завчасно переглянути перелік категорій страв, а також перелік страв для кожної окремої категорія. Вони також можуть ознайомитися з описом кожної страви, а також її ціною. Окрім цього сторінка закладу має необхідну контактну інформацію яка допоможе потенційному клієнту закладу дізнатися графік роботи, точну адресу (з можливістю миттєвого перегляду розташування за допомогою наявності елемента з фреймом Google Maps), а також переглянути відгуки вже побувавших у ресторані людей. Також для більш зручного використання на сторінкудодано кнопку-посилання з номер телефону закладу яка здійснює виклик при натисканні на неї, це дозволяє швидко зв’язатися з закладом і оформити замовлення з доставкою додому або отримати іншу необхідну інформацію.

Використання фреймворку Bootstrap значно полегшило процес розробки. Завдяки попередньо розробленим компонентам і адаптивній сітковій системі, Bootstrap запропонував міцну основу для створення веб-сайту, зручного для перегляду незалежно від пристрою чи розміру його екрану, тому що за допомогою Bootstrap, елементи сторінки підлаштовуються автоматично, щоб зберегти її цілісність й загальний дизайн.

В результаті роботи над цим дипломним проектом, було отримано веб-додаток, який має практичну цінність, тому що вирішує реальні проблеми такі як покращення досвіду потенційних клієнтів ресторану, а також збільшення клієнтської бази закладу, що у свою чергу прямим чином позитивно впливає на його прибуток.

Під час роботи над дипломним проектом по створенню веб-додатку для ресторану були досягнуті наступні цілі:

- проаналізовано предменту область задачі яку потрібно розв’язати;

– обрано раціональний шлях та необхідні інструменти для створення веб-додатку;
– написано необхідний HTML/CSS/JS код для функціонування веб-додатку;

В економічному розділі було вирахувано вартість розробки даного програмного застосунку, яка становить 105605 грн. Також було визначено час, який необзідний одному молодшому веб-розробнику для створення програмного застосунка, цей час становить 4,62 місяці або 813,64 людино-годин.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bootstrap documentation - https://getbootstrap.com/docs/5.3/getting-started/introduction;
2. Руководство по Bootstrap 4 - https://webref.ru/layout/bootstrap4;
3. Bootstrap 5 Tutorial - https://www.w3schools.com/bootstrap5;
4. Руководство по JavaScript - https://metanit.com/web/javascript;
5. CSS Tutorial - https://www.w3schools.com/css/default.asp;
6. HTML Tutorial - https://www.w3schools.com/html/default.asp;
7. Дэн Сидерхолм. CSS3 для веб-дизайнеров. 2012. 144 с.
8. З. Джилленуотер. Сила CSS3. Освой новейший стандарт веб-разработок. 2012. 304 с.
9. Эрик А. Мейер. CSS - каскадные таблицы стилей. Подробное руководство. 2008. 576 с.
10. Бен Фрейн. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. 2014. 304 с.
11. Дженнифер Нидерст Робинс. HTML5, CSS3 и JavaScript. Исчерпывающее руководство. 2014. 528 с.
12. Эстель Вейл. HTML5. Разработка приложений для мобильных устройств. 2015. 480 с.
13. Николас Закас. JavaScript для профессиональных веб-разработчиков. 2013. 960 с.
14. Дэвид Флэнаган. JavaScript. Подробное руководство. 2012. 1080 с.
15. Visual Studio Code - https://uk.wikipedia.org/wiki/Visual_Studio_Code;
16. Documentation for Visual Studio Code - https://code.visualstudio.com/docs;
17. Chrome DevTools - https://developer.chrome.com/docs/devtools;
18. Современный учебник JavaScript - https://learn.javascript.ru;
19. Рівень заробітної плати Junior Web Developer - https://jobs.dou.ua/salaries/?period=2022-12&position=Junior%20SE&technology=JavaScript&experience=0;

ДОДАТОК А

КОД ПРОГРАМИ

index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>>Ресторан китайської кухн</title>
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/styles.css">
    <link href='https://fonts.googleapis.com/css?family=Oxygen:400,300,700' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Lora' rel='stylesheet' type='text/css'>
  </head>
<body>
  <header>
    <nav id="header-nav" class="navbar navbar-default">
      <div class="container">
        <div class="navbar-header">
          <a href="index.html" class="pull-left visible-md visible-lg">
            <div id="logo-img"></div>
          </a>
          <div class="navbar-brand">
            <a href="index.html">
              <h1>Ресторан китайської кухні</h1>
            </a>
            <p>
              <img src="images/star-k-logo.png" alt="Kosher certificate">
              <span>Сертифікована кошерність</span>
            </p>
          </div>
          <button id="navbarToggle" type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#collapsable-nav"
            aria-expanded="false">
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
        </div>
        <div id="collapsable-nav" class="collapse navbar-collapse">
          <ul id="nav-list" class="nav navbar-nav navbar-right">
            <li id="navHomeButton" class="visible-xs active">
              <a href="index.html">
                <span class="glyphicon glyphicon-home"></span> Додому</a>
            </li>
            <li id = "navMenuButton">
              <a href="#" onclick="loadMenuCategories();">
                <span class="glyphicon glyphicon-cutlery"></span><br class="hidden-xs"> Меню</a>
            </li>
            <li>
              <a href="#">
                <span class="glyphicon glyphicon-info-sign"></span><br class="hidden-xs"> Про нас</a>
            </li>
            <li>
              <a href="#">
```

```
            <span class="glyphicon glyphicon-certificate"></span><br class="hidden-xs"> Нагороди</a>
          </li>
          <li id="phone" class="hidden-xs">
            <a href="tel:066-322-45-45">
              <span>066-322-45-45</span></a>
            <div>* Доставляємо додому</div>
          </li>
        </ul>
      </div>
    </div>
  </nav>
 </header>
 <div id="call-btn" class="visible-xs">
  <a class="btn" href="tel:066-322-45-45">
    <span class="glyphicon glyphicon-earphone"></span>
    066-322-45-45
  </a>
 </div>
 <div id="xs-deliver" class="text-center visible-xs">* Доставляємо додому</div>
 <div id="main-content" class="container"></div>
 <footer class="panel-footer">
  <div class="container">
    <div class="row">
      <section id="hours" class="col-sm-4">
        <span>Години роботи:</span><br>
        Неділя-Четвер: 08:30 - 22:00<br>
        П'ятниця: 08:30 - 21:00<br>
        Субота: Зачинено
        <hr class="visible-xs">
      </section>
      <section id="address" class="col-sm-4">
        <span>Адреса:</span><br>
        пр-т. Дмитра Яворницького, 46<br>
        Дніпро, 49000
        <p>* Доставка у радіусі 3-4 кілометрів, мінімальна сума заказу 300 грн + доставка 50 грн.</p>
        <hr class="visible-xs">
      </section>
      <section id="testimonials" class="col-sm-4">
        <p>"Найкращий ресторан китайської кухні де я був! І це говорить багато про що, оскільки я побував у
багатьох!"</p>
        <p>"Чудова їжа! Чудовий сервіс! Про більше просити і не треба! Я буду приходити сюди знову і знову!"</p>
      </section>
    </div>
    <div class="text-center">&copy; Ресторан Китайської Кухні 2023</div>
  </div>
 </footer>
 <script src="js/jquery-2.1.4.min.js"></script>
 <script src="js/bootstrap.min.js"></script>
 <script src="js/ajax.js"></script>
 <script src="js/script.js"></script>
</body>
</html>
```

categories-title-snippet.html

```
<h2 id="menu-categories-title" class="text-center">Категорії меню</h2>
```

home-snippet.html

```
<div class="jumbotron">
  <img src="images/jumbotron_768.jpg" alt="Picture of restaurant" class="img-responsive visible-xs">
</div>
<div id="home-tiles" class="row">
  <div class="col-md-4 col-sm-6 col-xs-12">
    <a href="#" onclick="loadMenuCategories();">
      <div id="menu-tile"><span>меню</span></div>
    </a>
  </div>
  <div class="col-md-4 col-sm-6 col-xs-12">
    <a href="#" onclick="loadMenuItems('SP');">
      <div id="specials-tile"><span>особливі страви</span></div>
    </a>
  </div>
  <div class="col-md-4 col-sm-12 col-xs-12">
    <a href="https://goo.gl/maps/GbjmFBcFShmUMw4TA?coh=178572&entry=tt" target="_blank">
      <div id="map-tile">
        <iframe src="https://www.google.com/maps/embed?pb=!1m13!1m8!1m3!1d165.35299790239148
            !2d35.049227161656866!3d48.46328362205484!3m2!1i1024!2i768!4f13.1!3m2!1m1!2z
            NDjCsDI3JzQ3LjciTiAzNcKwMDInNTcuNyJF!5e0!3m2!1sru!2sua!4v1683219856628!5m2!1sru!2sua"
width="100%"
          height="250" style="border:0;" allowfullscreen="" loading="lazy"
          referrerpolicy="no-referrer-when-downgrade"></iframe>
        <span>карта</span>
      </div>
    </a>
  </div>
</div>
```

category-snippet.html

```
<div class="col-md-3 col-sm-4 col-xs-6 col-xxs-12">
  <a href="#" onclick="loadMenuItems('{{short_name}}');">
    <div class="category-tile">
      <img width="200" height="200" src="images/menu/{{short_name}}/{{short_name}}.jpg" alt="{{name}}">
      <span>{{name}}</span>
    </div>
  </a>
</div>
```

menu-items-title.html

```
<h2 id="menu-categories-title" class="text-center">{{name}}</h2>
```

menu-item.html

```
<div class="menu-item-tile col-md-6">
  <div class="row">
    <div class="col-sm-5">
      <div class="menu-item-photo">
        <div>{{short_name}}</div>
        <img class="img-responsive" width="250" height="150" src="images/menu/{{catShortName}}/{{short_name}}.jpg"
alt="Item">
      </div>
      <div class="menu-item-price">{{price_small}}<span> {{small_portion_name}}</span> {{price_large}}
<span>{{large_portion_name}}</span></div>
    </div>
    <div class="menu-item-description col-sm-7">
      <h3 class="menu-item-title">{{name}}</h3>
      <p class="menu-item-details">{{description}}</p>
```

```
    </div>
  </div>
  <hr class="visible-xs">
</div>
```

styles.css

```
body {
  font-size: 16px;
  color: #fff;
  background-color: #61122f;
  font-family: 'Oxygen', sans-serif;
}
#header-nav {
  background-color: #f6b319;
  border-radius: 0;
  border: 0;
}
#logo-img {
  background: url('../images/restaurant-logo_large.png') no-repeat;
  width: 150px;
  height: 150px;
  margin: 10px 15px 10px 0;
}
.navbar-brand {
  padding-top: 25px;
}
.navbar-brand h1 {
  font-family: 'Lora', serif;
  color: #557c3e;
  font-size: 1.5em;
  text-transform: uppercase;
  font-weight: bold;
  text-shadow: 1px 1px 1px #222;
  margin-top: 0;
  margin-bottom: 0;
  line-height: .75;
}
.navbar-brand a:hover, .navbar-brand a:focus {
  text-decoration: none;
}
.navbar-brand p {
  color: #000;
  text-transform: uppercase;
  font-size: .7em;
  margin-top: 15px;
}
.navbar-brand p span {
  vertical-align: middle;
}
#nav-list {
  margin-top: 10px;
}
#nav-list a {
  color: #951C49;
  text-align: center;
}
#nav-list a:hover {
  background: #E7E7E7;
}
#nav-list a span {
  font-size: 1.8em;
```

```
}
#phone {
  margin-top: 5px;
}
#phone a {
  text-align: right;
  padding-bottom: 0;
}
#phone div {
  color: #557c3e;
  text-align: right;
  padding-right: 15px;
}
.navbar-header button.navbar-toggle, .navbar-header .icon-bar {
  border: 1px solid #61122f;
}
.navbar-header button.navbar-toggle {
  clear: both;
  margin-top: -30px;
}
.container .jumbotron {
  box-shadow: 0 0 50px #3F0C1F;
  border: 2px solid #3F0C1F;
}
#menu-tile, #specials-tile, #map-tile {
  height: 250px;
  width: 100%;
  margin-bottom: 15px;
  position: relative;
  border: 2px solid #3F0C1F;
  overflow: hidden;
}
#menu-tile:hover, #specials-tile:hover, #map-tile:hover {
  box-shadow: 0 1px 5px 1px #cccccc;
}
#menu-tile {
  background: url('../images/menu-tile.jpg') no-repeat;
  background-position: center;
}
#specials-tile {
  background: url('../images/specials-tile.jpg') no-repeat;
  background-position: center;
}
#menu-tile span, #specials-tile span, #map-tile span {
  position: absolute;
  bottom: 0;
  right: 0;
  width: 100%;
  text-align: center;
  font-size: 1.6em;
  text-transform: uppercase;
  background-color: #000;
  color: #fff;
  opacity: .8;
}
.panel-footer {
  margin-top: 30px;
  padding-top: 35px;
  padding-bottom: 30px;
  background-color: #222;
  border-top: 0;
}
.panel-footer div.row {
```

```
  margin-bottom: 35px;
}
#hours, #address {
  line-height: 2;
}
#hours > span, #address > span {
  font-size: 1.3em;
}
#address p {
  color: #557c3e;
  font-size: .8em;
  line-height: 1.8;
}
#testimonials {
  font-style: italic;
}
#testimonials p:nth-child(2) {
  margin-top: 25px;
}
.category-tile {
  position: relative;
  border: 2px solid #3F0C1F;
  overflow: hidden;
  width: 200px;
  height: 200px;
  margin: 0 auto 15px;
}
.category-tile span {
  position: absolute;
  bottom: 0;
  right: 0;
  width: 100%;
  text-align: center;
  font-size: 1.2em;
  text-transform: uppercase;
  background-color: #000;
  color: #fff;
  opacity: .8;
}
.category-tile:hover {
  box-shadow: 0 1px 5px 1px #cccccc;
}
#menu-categories-title {
  margin-bottom: 50px;
}

.menu-item-tile {
  margin-bottom: 25px;
}
.menu-item-tile hr {
  width: 80%;
}
.menu-item-tile .menu-item-price {
  font-size: 1.1em;
  text-align: right;
  margin-top: -15px;
  margin-right: -15px;
}
.menu-item-tile .menu-item-price span {
  font-size: .6em;
}
.menu-item-photo {
  position: relative;
```

```
  border: 2px solid #3F0C1F;
  overflow: hidden;
  padding: 0;
  margin-right: -15px;
  margin-left: auto;
  margin-bottom: 20px;
  max-width: 250px;
}
.menu-item-photo div {
  position: absolute;
  bottom: 0;
  right: 0;
  width: 80px;
  background-color: #557c3e;
  text-align: center;
}
.menu-item-description {
  padding-right: 30px;
}
h3.menu-item-title {
  margin: 0 0 10px;
}
.menu-item-details {
  font-size: .9em;
  font-style: italic;
}
.test1 {
  clear: both;
}

@media (min-width: 1200px) {
  .container .jumbotron {
    background: url('../images/jumbotron_1200.jpg') no-repeat;
    height: 675px;
  }
  .navbar-brand p img {
    height: 22px;
  }
}

@media (min-width: 992px) and (max-width: 1199px) {
  #logo-img {
    background: url('../images/restaurant-logo_medium.png') no-repeat;
    width: 100px;
    height: 100px;
    margin: 5px 5px 5px 0;
  }
  .container .jumbotron {
    background: url('../images/jumbotron_992.jpg') no-repeat;
    height: 558px;
  }
  .navbar-brand h1 {
    font-size: 1.3em;
  }
  #nav-list a {
    font-size: 0.9em;
  }
  .navbar-brand p {
    font-size: 0.6em;
  }
  .navbar-brand p img {
    height: 19px;
  }
```

```
}

@media (min-width: 768px) and (max-width: 991px) {
  .container .jumbotron {
    background: url('../images/jumbotron_768.jpg') no-repeat;
    height: 432px;
  }
  .navbar-brand p img {
    height: 20px;
  }
}

@media (max-width: 767px) {
  .navbar-brand {
    padding-top: 10px;
    height: 80px;
  }
  .navbar-brand h1 {
    padding-top: 10px;
    font-size: 5vw;
  }
  .navbar-brand p {
    font-size: .6em;
    margin-top: 12px;
  }
  .navbar-brand p img {
    height: 20px;
  }
  #collapsable-nav a {
    font-size: 1.2em;
  }
  #collapsable-nav a span {
    font-size: 1em;
    margin-right: 5px;
  }
  #call-btn > a {
    font-size: 1.5em;
    display: block;
    margin: 0 20px;
    padding: 10px;
    border: 2px solid #fff;
    background-color: #f6b319;
    color: #951c49;
  }
  #xs-deliver {
    margin-top: 5px;
    font-size: .7em;
    letter-spacing: .1em;
    text-transform: uppercase;
  }
  .container .jumbotron {
    margin-top: 30px;
    padding: 0;
  }
    #menu-tile, #specials-tile {
    width: 360px;
    margin: 0 auto 15px;
  }

  .panel-footer section {
    margin-bottom: 30px;
    text-align: center;
  }
```

```
  .panel-footer section:nth-child(3) {
    margin-bottom: 0;
  }
  .panel-footer section hr {
    width: 50%;
  }

  .menu-item-photo {
    margin-right: auto;
  }
  .menu-item-tile .menu-item-price {
    text-align: center;
  }
  .menu-item-description {
    text-align: center;;
  }
}

@media (max-width: 479px) {
  .navbar-brand h1 {
    padding-top: 5px;
    font-size: 5vw;
  }
  #menu-tile, #specials-tile {
    width: 280px;
    margin: 0 auto 15px;
  }
  .col-xxs-12 {
    position: relative;
    min-height: 1px;
    padding-right: 15px;
    padding-left: 15px;
    float: left;
    width: 100%;
  }
}
```

ajax.js

```
sendGetRequest = function (requestUrl, responseHandler, isJsonResponse) {
    var request = new XMLHttpRequest();
    request.onreadystatechange = function() {
        handleResponse(request, responseHandler, isJsonResponse);
      };
    request.open("GET", requestUrl, true);
    request.send(null);
  };
function handleResponse(request, responseHandler, isJsonResponse) {
  if ((request.readyState == 4) && (request.status == 200)) {
    if (isJsonResponse == undefined) {
      isJsonResponse = true;
    }
    if (isJsonResponse) {
      responseHandler(JSON.parse(request.responseText));
    }
    else {
      responseHandler(request.responseText);
    }
  }
}
```

script.js

```
document.addEventListener("DOMContentLoaded", function (event) {
  document.querySelector("#navbarToggle").addEventListener("blur", function (event) {
    var screenWidth = window.innerWidth;
    if (screenWidth < 768) {
      $("#collapsable-nav").collapse('hide');
    }
  });
});

var homeHtml = "snippets/home-snippet.html";
var allCategoriesUrl = "data/categories.json";
var categoriesTitleHtml = "snippets/categories-title-snippet.html";
var categoryHtml = "snippets/category-snippet.html";
var menuItemsTitleHtml = "snippets/menu-items-title.html";
var menuItemHtml = "snippets/menu-item.html";
var menuItemsUrl = "data/menu_items/";

sendGetRequest(homeHtml, function (responseText) {
  document.querySelector("#main-content").innerHTML = responseText;
}, false);

var insertHtml = function (selector, html) {
  var targetElem = document.querySelector(selector);
  targetElem.innerHTML = html;
};

var insertProperty = function (string, propName, propValue) {
  var propToReplace = "{{" + propName + "}}";
  string = string.replace(new RegExp(propToReplace, "g"), propValue);
  return string;
};

var switchMenuToActive = function () {
  var classes = document.querySelector("#navHomeButton").className;
  classes = classes.replace(new RegExp("active", "g"), "");
  document.querySelector("#navHomeButton").className = classes;
  classes = document.querySelector("#navMenuButton").className;
  if (classes.indexOf("active") == -1) {
    classes += " active";
    document.querySelector("#navMenuButton").className = classes;
  }
};

loadMenuCategories = function () {
  sendGetRequest(allCategoriesUrl, buildAndShowCategoriesHTML);
};

loadMenuItems = function (categoryShort) {
  sendGetRequest(
    menuItemsUrl + categoryShort + ".json",
    buildAndShowMenuItemsHTML
  );
};

function buildAndShowCategoriesHTML(categories) {
  sendGetRequest(categoriesTitleHtml, function (categoriesTitleHtml) {
    sendGetRequest(categoryHtml, function (categoryHtml) {
      switchMenuToActive()
      var categoriesViewHtml = buildCategoriesViewHtml(categories, categoriesTitleHtml, categoryHtml);
```

```
      insertHtml("#main-content", categoriesViewHtml);
    }, false);
  },
    false
  );
}

function buildCategoriesViewHtml(categories, categoriesTitleHtml, categoryHtml) {
  var finalHtml = categoriesTitleHtml;
  finalHtml += "<section class='row'>";
  for (var i = 0; i < categories.length; i++) {
    var html = categoryHtml;
    var name = "" + categories[i].name;
    var short_name = categories[i].short_name;
    html = insertProperty(html, "name", name);
    html = insertProperty(html, "short_name", short_name);
    finalHtml += html;
  }
  finalHtml += "</section>";
  return finalHtml;
}

function buildAndShowMenuItemsHTML(categoryMenuItems) {
  sendGetRequest(menuItemsTitleHtml,function (menuItemsTitleHtml) {
      sendGetRequest(menuItemHtml, function (menuItemHtml) {
          switchMenuToActive()
          var menuItemsViewHtml = buildMenuItemsViewHtml(categoryMenuItems,menuItemsTitleHtml,menuItemHtml);
          insertHtml("#main-content", menuItemsViewHtml);},false);
    },false);
}

function buildMenuItemsViewHtml(categoryMenuItems,menuItemsTitleHtml,menuItemHtml) {
  menuItemsTitleHtml = insertProperty(menuItemsTitleHtml,"name",categoryMenuItems.category.name);
  menuItemsTitleHtml =
insertProperty(menuItemsTitleHtml,"special_instructions",categoryMenuItems.category.special_instructions);
  var finalHtml = menuItemsTitleHtml;
  finalHtml += "<section class='row'>";
  var menuItems = categoryMenuItems.menu_items;
  var catShortName = categoryMenuItems.category.short_name;
  for (var i = 0; i < menuItems.length; i++) {
    var html = menuItemHtml;
    html = insertProperty(html, "short_name", menuItems[i].short_name);
    html = insertProperty(html, "catShortName", catShortName);
    html = insertItemPrice(html, "price_small", menuItems[i].price_small);
    html = insertItemPortionName(html,"small_portion_name",menuItems[i].small_portion_name);
    html = insertItemPrice(html, "price_large", menuItems[i].price_large);
    html = insertItemPortionName(html,"large_portion_name",menuItems[i].large_portion_name);
    html = insertProperty(html, "name", menuItems[i].name);
    html = insertProperty(html, "description", menuItems[i].description);
    if (i % 2 != 0) {
      html += "<div class='clearfix visible-lg-block visible-md-block'></div>";
    }
    finalHtml += html;
  }
  finalHtml += "</section>";
  return finalHtml;
}

function insertItemPrice(html, pricePropName, priceValue) {
  if (!priceValue) {
    return insertProperty(html, pricePropName, "");
  }
  priceValue = priceValue.toFixed(2) + " грн";
```

```
  html = insertProperty(html, pricePropName, priceValue);
  return html;
}

function insertItemPortionName(html, portionPropName, portionValue) {
  if (!portionValue) {
    return insertProperty(html, portionPropName, "");
  }
  portionValue = "(" + portionValue + ")";
  html = insertProperty(html, portionPropName, portionValue);
  return html;
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

Перелік файлів на магнітному носії

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна_робота_Бондарь.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна_робота_Бондарь.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Бондарь.rar	Архів. Містить коди програми і скомпільовану програму.
Презентація	
Бондарь.ppt	Презентація кваліфікаційної роботи