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

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-сайту для сервісного центру з ремонту техніки Apple»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
*(код, найменування спеціальності)*
освітньо-професійної програми Інформаційні системи та технології
*(назва)*

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

_____ Юрій АНДРУЩЕНКО
*(підпис)* *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти Юрій АНДРУЩЕНКО
група ІСДМ-61

Керівник: Аліна ТУШИЧ
*науковий ступінь, вчене звання* д.т.н.

Рецензент:
*науковий ступінь, вчене звання* _____
Ім'я, ПРІЗВИЩЕ

**Київ 2023**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

**ЗАТВЕРДЖУЮ**
Завідувач кафедрою ІПЗАС

_____ Каміла СТОРЧАК
«_____» _____ 2023 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Андрущенко Юрій Михайлович
*(прізвище, ім'я, по батькові здобувача)*

1. Тема кваліфікаційної роботи: Розробка web-сайту для сервісного центру з ремонту техніки Apple

керівник кваліфікаційної роботи Аліна ТУШИЧ д.т.н.
*(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)*
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, основні властивості об'єктів в HTML,CSS, динамічність сторінки із JS, JQuery, властивості мережевого протоколу SMTP.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Проектування макету сайту та аналіз інформаційної системи
Створення структури інформаційної системи та її оформлення

Розробка динамічної поведінки інтерфейсу та впровадження системи SMTP

5. Перелік графічного матеріалу: *презентація*
    1. Створення структури веб-сайту
    2. Оформлення сторінок
    3. Додавання динамічності та інтерактивності
    4. Робота із формою

6. Дата видачі завдання«19» жовтня 2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір відповідного науково-технічного матеріалу | 01.09.2023-20.09.2023 | Виконав |
| 2 | Відомості про програмування та створення web-сайтів | 20.09.2023-01.10.2023 | Виконав |
| 3 | Створення макету і структури сторінки | 04.10.2023-22.10.2023 | Виконав |
| 4 | Оформлення сторінки відповідно макету | 22.10.2023-06.11.2023 | Виконав |
| 5 | Додавання динамічності сайту, та впровадження мережевого протоколу | 07.11.2023-15.11.2023 | Виконав |
| 6 | Тестування моделі на різних пристроях | 15.11.2023-10.12.2023 | Виконав |
| 7 | Виставлення на хостинг | 10.12.2023-13.12.2023 | виконав |
| 8 | Розробка демонстраційних матеріалів (Презентація) | 13.12.2023-20.12.2023 | Виконав |

Здобувач вищої освіти _____ Юрій АНДРУЩЕНКО
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Аліна ТУШИЧ
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 79 стор., 109 рис., 5 джерел.

*Мета роботи* – проєктування та розробка веб-сайту, для сервісного центру.

*Об'єкт дослідження* – процес проектування, та створення інформаційної системи під виглядом веб-сторінки.

*Предмет дослідження* – інформаційна система для роботи сервісного центру.

*Короткий зміст роботи*: Було створено повноцінний макет, із головною сторінкою формою для заповнення, складом команди та контактним блоком, завдяки додатку Figma. Створена основна структура за допомогою HTML, та заданий певний стиль оформлення для нашого проекту. За допомогою JavaScript, були додані певні динамічні об'єкти, та додали змогу відправки форми на пошту завдяки PHP. Виконана робота по адаптації для мобільних девайсів, та виставлення на безкоштовний хостинг для перегляду.

КЛЮЧОВІ СЛОВА: ДИЗАЙН ІНТЕРФЕЙСУ, ВЗАЄМОДІЯ КОРИСТУВАЧА З ВЕБ-САЙТОМ, МОБІЛЬНА АДАПТАЦІЯ, МЕТА-ТЕГИ.

## ABSTRACT

Text part of the master's qualification work: 79 pages, 109 pictures, 5 sources.

The purpose of the work design and development of a website for a service center.

Object of research – design process and creation of an information system in the form of a web page.

Subject of research – information system for service center operation.

Summary of the work: A complete layout was created, with a main page, a form to fill in, a team composition and a contact block, thanks to the Figma app. A basic structure has been created using HTML, and a certain design style has been set for our project. With the help of JavaScript, certain dynamic objects were added, and the possibility of sending the form to the mail thanks to PHP was added. Completed work on adaptation for mobile devices, and posting on free hosting for viewing.



KEYWORDS: INTERFACE DESIGN, USER INTERACTION WITH THE WEBSITE, MOBILE ADAPTATION, META-TAGS.

**ЗМІСТ**

ВСТУП ..... 12
1 ПОЯВА ПЕРШОГО В СВІТІ САЙТУ ..... 16
2.1 Історія появи мови розмітки HTML ..... 16
2.2 Основні теги HTML ..... 17
2.3 Метадані документа ..... 18
2.4 Гіперпосилання ..... 19
2.5 Базові теги ..... 20
2.6 Форматування тексту ..... 21
2.7 Зображення ..... 23
2.8 Списки ..... 23
2.9 Таблиці HTML ..... 24
2.10 Коментарі ..... 25
3 CSS ЯК ІНСТРУМЕНТ ОФОРМЛЕННЯ ..... 26
3.1 Введення в CSS ..... 26
3.2 Робота із CSS ..... 26
3.3 Селектори CSS ..... 27
3.4 Блокова модель ..... 30
3.5 Фон та кордони ..... 32
3.6 Позиціювання ..... 33
3.7 Обтікання ..... 35
3.8 Z-index ..... 35
3.9 Робота із текстом ..... 36
4 JavaScript ЯК РЕСУРС ДЛЯ ПОКРАЩЕННЯ САЙТУ ..... 40
4.1 Основні поняття JavaScript ..... 40
4.1.1 Відомості про DOM ..... 41
4.1.2 Обробник подій ..... 42
4.1.3 Змінні ..... 42
4.1.4 Структура даних ..... 43
4.1.5 Функції ..... 44
4.2 Підключення JavaScript ..... 45

4.3 JQuery в полегшенні роботи із JS ..... 46
4.3.1 Селектори jQuery ..... 46
4.3.2 Події ..... 47
4.4 Під’єднання JQuery ..... 48
5 РОЛЬ PHP У СТВОРЕННІ Web-сайту ..... 50
5.1 Змінні в PHP ..... 51
5.2 Функції PHP ..... 52
6 ЕТАПИ СТВОРЕННЯ ВЕБ-СТОРІНКИ ..... 53
6.1 Створення макету ..... 53
6.2 Робота із редактором Visual Studio Code ..... 56
6.2.1 Створення структури документа ..... 57
6.2.2 Оформлення веб-сайту ..... 61
6.2.3 Додавання динамічності та інтерактивності ..... 70
6.2.4 Відправа форми за допомогою PHP ..... 74
6.2.5 Налаштування відображення сайту на телефонах ..... 75
6.2.6 Виставлення на хостинг ..... 78
Висновок ..... 81
Перелік посилань ..... 82
Демонстраційні матеріали (Презентація) ..... 83

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

| | | |
|---|---|---|
| HTML | HyperText Markup Language | Стандартизована мова розмітки документів для перегляду веб-сторінок в браузері. |
| WWW | World Wide Web | Розподілена система, що надає доступ до пов'язаних між собою документів. |
| HTTP | HyperText Transfer Protocol | Протокол передачі гіпертексту. |
| URL | Uniform Resource Locator | Система уніфікованих адрес електронних ресурсів. |
| CSS | Cascading Style Sheets | Каскадні таблиці стилів. |
| SGML | Standard Generalized Markup Language | Стандартна узагальнена мова розмітки. |
| XUL | XML User Interface Language | Мова розмітки для створення динамічних інтерфейсів на основі XML. |
| SVG | Scalable Vector Graphics | Масштабована векторна графіка. |
| XML | eXtensible Markup Language | Мова розмітки, що нагадує HTML. |
| JS | JavaScript | Мультипарадигменна мова програмування. |
| JQuery | JQuery | Набір функцій JavaScript |
| HTTP | HyperText Transfer Protocol | Протокол прикладного рівня передачі даних. |
| HTTPS | HyperText Transfer Protocol Secure | Розширення протоколу HTTP для підтримки шифрування з метою підвищення безпеки. |
| ОС | Operating System | Операційна система |
| PHP | Hypertext Preprocessor | Скриптова мова програмування, призначена для розробки веб-застосунків |
| SMTP | Simple Mail Transfer Protocol | Протокол передачі електронної пошти |
| DOM | Document Object Model | Представляє структуру документа |

# ВСТУП

Розвиток інформаційних технологій ніколи не стоїть на місці, а тільки прискорюється із часом . І на сьогоденні , одним із найбільших ресурсів інформації є веб-сайти. Кожного дня, ми беремо в руки телефон, заходимо в пошукову систему, і задаємо певний запит, на який ми шукаємо відповідь.

Давайте розглянемо терміни "сторінка" і "сайт". Веб-сторінка - це текстовий файл, створений мовою HTML, який можна відобразити у вікні браузера. Для створення і редагування HTML-файлу можна використовувати будь-який текстовий редактор. Якщо відкрити HTML-файл за допомогою інструментів перегляду, таких як Google, він покаже текст та графіку.

Метою саме нашого проєкту було створення одно сторінкового сайту, у якого була головна сторінка, із основною інформаціє, сторінка із формою для заповнення, яка б передавалась на пошту, для подальшої обробки, сторінка із основним виконавчим складом та контактний блок. Веб-сторінки були оформленні за допомогою каскадних таблиць стилів, були вказані розміри тексту, кольори, відступи внутрішні так і зовні, виконане позиціювання для кращого відображення, заданні різні анімації для об’єктів. Додані динамічні об’єкти, які відкриваються по натисканню, або фіксуються в процесі прокрутки сторінки. Проведена оптимізація SMTP запиту, для коректного прийняття та відправки даних із форми заявки.

## 1 ПОЯВА ПЕРШОГО В СВІТІ САЙТУ

Фізик Тім Бернерс-Лі, працюючи у Європейській організації ядерних досліджень CERN в Женеві, став піонером першого в історії веб-сайту. У березні 1989 року він висунув ідею використання гіпертексту для передачі даних через глобальну мережу Інтернет. Вже у 1990 році Бернерс-Лі створив перший веб-сайт під адресою info.cern.ch. На цьому сайті був детальний опис нової технології WWW (World Wide Web), яка базувалася на принципах інтернет-адресації URL, протоколу передачі інформації HTTP та мови розмітки гіпертексту HTML.

Заздалегідь перед створенням першого сайту, Бернерс-Лі розробив Enquire в 1980 році. Цей проект мав на меті створення єдиного інформаційного простору, який об'єднував би Інтернет, комп'ютери користувачів і гіпертекст. Це стало можливим завдяки принципам випадкових асоціацій. Enquire використовував гіпертекст для забезпечення зручного доступу до інформації на web-сторінках, дозволяючи зв'язувати їх між собою за допомогою посилань. Бернерс-Лі використовував цей підхід, щоб демонструвати переваги нововведень своїм колегам у CERN. Таким чином, вчений не лише висунув концепцію першого сайту, але й використовував попередні дослідження, зокрема Enquire, для практичної реалізації інновацій у веб-середовищі.

info.cern.ch, на той момент, визнавався як перший веб-сервер, заснований на комп'ютері NeXT. Розташований у лабораторії CERN, цей сервер отримав почесне звання першого в світі сайту. Перша web-сторінка містила інформацію про проект першого в історії інтернет-браузера WWW, технічні деталі створення інших web-сторінок та дані про гіпертекст. Також там надавалися докладні пояснення щодо проведення пошуку інформації в Інтернеті, принципи інсталяції браузерів, серверів і порядок роботи з ними. Цей сайт став повноцінним Інтернет-каталогом, із якого Тім Бернерс-Лі пізніше опублікував перелік посилань на нові веб-сайти.

З 1991 року почалася експансія web-серверів до інших європейських установ, і невдовзі американські колеги приєдналися, використовуючи наукові розробки. В Стенфордському центрі лінійного прискорювача SLAC з’явився сервер. Якщо на

кінець 1992 року було всього 26 відомих web-серверів у світі, то до жовтня 1993 року ця кількість зросла до 200.

Цікаво відзначити, що ПК NeXT, на якому працював Тім Бернерс-Лі, був представлений в CERN на виставці Microcosm як перший в світі web-сервер, web-редактор і гіпермедіа браузер. Стандарт WWW був визнаний і затверджений в Женеві у травні 1991 року в CERN, при цьому Тім Бернерс-Лі отримав почесне звання "батька" ключових веб-технологій: URI/URL, HTTP, HTML.

В наш час Тім Бернерс-Лі очолює власний Консорціум Всесвітньої Павутини (WWW Consortium), що було засновано ним. На сьогодні, компанія виконує роль розробки та впровадження нових стандартів.

# 2 ОСНОВНІ СКЛАДОВІ МОВИ HTML, ТА ОСНОВНІ ОСОБЛИВОСТІ

## 2.1 Історія появи мови розмітки HTML

Мова гіпертекстової розмітки HTML (HyperText Markup Language) була розроблена на початку 1990-х років працівником Європейської організації з ядерних досліджень (CERN) Тімом Бернерс-Лі. HTML ґрунтується на мові SGML (Standard Generalized Markup Language – Стандартна узагальнена мова розмітки).

Головною концепцією SGML було створення платформонезалежної мови для розмітки документів. Цю ідею втілив колектив під керівництвом Чарльза Гольдфарба з компанії IBM, який вирішив відокремити структуру документа від його оформлення.

Засада розмітки ґрунтувалася на введенні у код спеціальних керуючих конструкцій, що визначали елементи структури, не надаючи вказівок щодо їх візуального відображення. Оформлення документа винесено в окремий файл, відомий як таблиця стилів. Така структура дозволяла відображати однакову інформацію у різних візуальних варіантах.

Розробники системи прийшли до висновку, що документи можуть бути надійно оброблені лише за умови відповідності єдиному стандарту, і всі документи, що не відповідають цьому стандарту, слід відкидати. З цією метою був введений механізм визначення типів документів (DTD – Document Type Definition). DTD, подібно до таблиці стилів, є зовнішнім файлом відносно відповідного документа.

## 2.2 Основні теги HTML

### <DOCTYPE>

Тег <DOCTYPE> призначений для визначення типу поточного документа (DTD - Document Type Definition). Це необхідно для того, щоб браузер розумів, як правильно інтерпретувати веб-сторінку. HTML існує у різних версіях, а також існує XHTML (EXtensible HyperText Markup Language, розширена мова розмітки гіпертексту), яка подібна до HTML, але має відмінний синтаксис. Додавання тегу <DOCTYPE> у перший рядок коду дозволяє браузеру чітко визначити, за яким стандартом слід відображати веб-сторінку, уникнути непорозумінь і забезпечити правильне відображення контенту.

### <html lang="ua">

Lang атрибут використовується для визначення мови елемента: мову, на якій написані не редаговані елементи, або мову, на якій користувач повинен редагувати редаговані елементи.

### <head>

HTML-тег <head> призначений для включення елементів, що допомагають браузеру у взаємодії з даними. Крім того, всередині контейнера <head> можуть розміщуватися мета-теги, які забезпечують зберігання інформації для браузерів і пошукових систем. Такі мета-теги можуть включати назву документа, скрипти, стилі та інші параметри.

### <title>

HTML-тег <title> встановлює заголовок документа, який відображається у вікні браузера або на вкладці сторінки. Вміст елемента містить тільки текст, і будь-які теги всередині цього елемента будуть проігноровані.

### <body>

HTML тег <body> призначений для зберігання змісту веб-сторінки (контенту), що відображається у вікні браузера. Весь основний контент, та скрипти потрібно розташовувати саме всередині тегу <body>. До цієї інформації відноситься текст, зображення, теги, таблиці, скрипти.

Тег <body> використовується для задання позиціювання елементів на сторінці, кольору тексту, відступів, шрифту який буде задаватися на всю сторінку та інше.Але набагато краще, та зручніше використовувати CSS стилі, застосовуючи їх до селектора body . Тим паче, що більше тегів вже підтримується різними браузерами.

```
<!DOCTYPE html>
<html lang="ua">
<head>
  <meta charset="UTF-8">

  <title>AppleServiceTeam</title>
</head>
<body>
```

Рисунок 2.2.1 – Приклад структури сайту

## 2.3 Метадані документа

Метадані містять інформацію про сторінку, включаючи стилі, скрипти та дані, які допомагають програмному забезпеченню (пошукові системи, браузери і т.д.) використовувати та відображати сторінку. Інформацію про стилі та скрипти можна визначити на сторінці або посилати на інший файл, який містить необхідні дані.

### <base>

HTML-тег <base> встановлює основну адресу (URL) для всіх відносних адрес (URLs) у документі. Дозволяє лише один елемент <base> у документі.

### <link>

Тег <link> визначає відносини між поточним документом та зовнішнім ресурсом. Зазвичай використовується для посилання на stylesheets і створення іконок сайту, включаючи іконки у стилі "favicon" та для домашніх екранів та додатків мобільних пристроїв.

**<meta>**

Елемент <meta> представляє метадані, такі як <base>, <link>, <script>, <style>, чи <title>. Використовується для надання додаткової інформації про документ.

## 2.4 Гіперпосилання

Гіперпосилання є невід'ємною частиною функціонування Інтернету, визначаючи його мережевий характер. Ця стаття надає синтаксис для створення посилань та обговорює кращі практики щодо їх використання.

Гіперпосилання – це одна з ключових інновацій у світі Інтернету, визначаючи його унікальні можливості. З самого початку вони визначали особливість інтерактивності Інтернету, роблячи його справжньою "мережею". Гіперпосилання дозволяють з’єднувати наші документи з будь-яким іншим документом або ресурсом за нашим вибором. Ці посилання також надають можливість зв'язувати конкретні частини документів і робити веб-додатки доступними за простими веб-адресами, відзначаючи їх від традиційних локальних програм.

Створення гіперпосилань в Інтернеті вимагає використання певного синтаксису. У коді HTML це може виглядати наступним чином:

```
<a href="посилання_на_ресурс">Текст або зображення, які стануть посиланням</a>
```

Рисунок 2.4.1 – Приклад гіперпосилання

Тут href визначає адресу (URL) ресурсу, на який веде посилання. Текст або зображення всередині тегу <a> слугує активною областю посилання.

Гіперпосилання визначають важливий аспект взаємодії в Інтернеті, забезпечуючи ефективний обмін інформацією та зручний доступ до різноманітних ресурсів.

### 2.5 Базові теги

HTML-теги заключаються в углові дужки <>. В закривальному тезі також використовується коса риска (слеш) </>. Нижче наведено теги, які використовуються частіше за інші.

- Теги <h1>-<h6> для заголовків
- Тег  для визначення абзаців
- Тег <img> для вставки зображень
- Тег <a> для вставки посилань

Теги <h1> до <h6> використовуються для структурування заголовків. В HTML є 6 рівнів заголовків, від <h1> до <h6>. <h1> використовується для позначення найважливіших заголовків, тоді як <h6> використовується для найменш важливих заголовків.

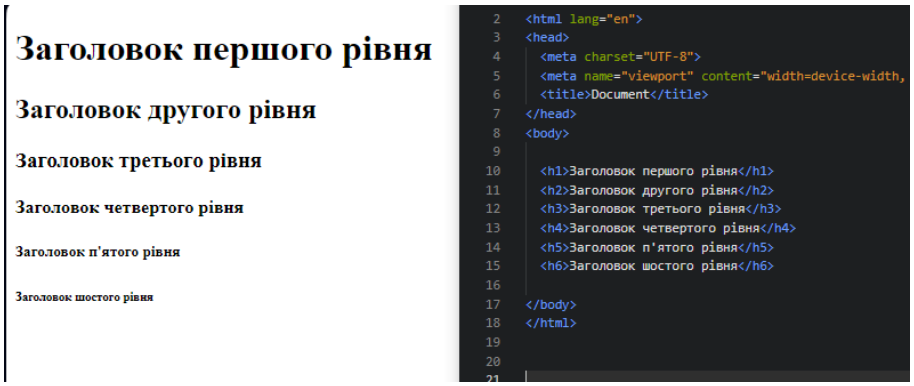



Рисунок 2.5.1 – Приклад заголовків

Парний тег використовується для розділення абзаців. За замовчуванням браузери автоматично додають відступи до  і після .

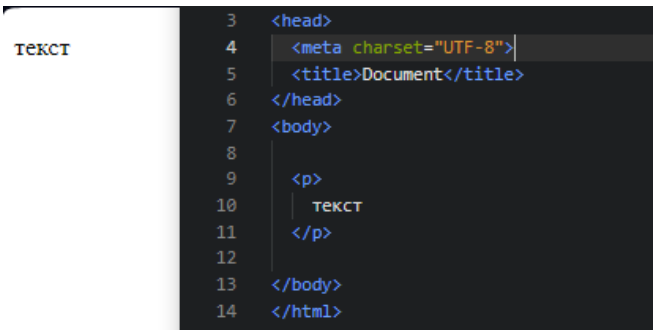



Рисунок 2.5.2 – Приклад абзацу

## 2.6 Форматування тексту

Вигляд тексту на екрані залежить від використаних HTML-тегів для його форматування. Теги форматування поділяються на дві категорії: теги фізичної розмітки, які відповідають за стильове оформлення (жирний шрифт, курсив, шрифт і т. д.) тексту, та теги логічної розмітки, які несуть смислове навантаження (наприклад, вказують пошуковим системам, яким чином оцінювати веб-сторінку за вмістом певних слів).

HTML теги <b> і  використовуються для визначення напівжирного зображення шрифту. Їх різниця полягає в тому, що тег <b> є тегом фізичної розмітки і використовується для виділення тексту без особливого акценту на його важливість. Тег , навпаки, визначає текст як особливо важливий. Зміст тегу  має велику вагу для пошукових систем, і пристрої, які використовують текстове відтворення, можуть відзначити його особливою інтонацією.

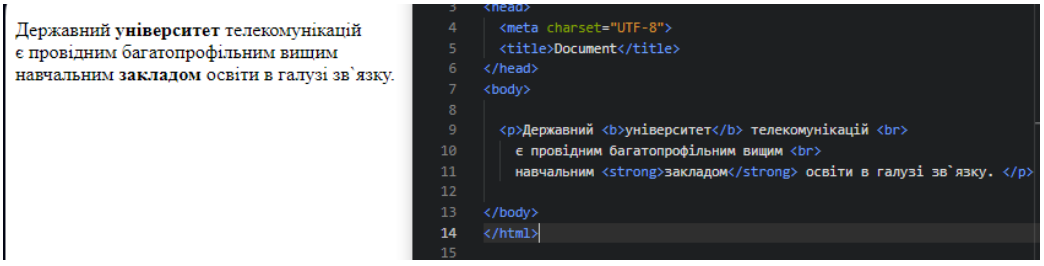



Рисунок 2.6.1 – Оформлення за допомогою <b> і 

Теги <i> та  використовуються для задання курсивного зображення шрифту. Тег <i> є елементом фізичної розмітки, що означає, що вкладений текст

відрізняється лише візуально і не має особливого семантичного значення для браузерів та пошукових систем. Тег , навпаки, використовується для експресивного виділення фрагменту тексту, приділяючи йому емоційний акцент.

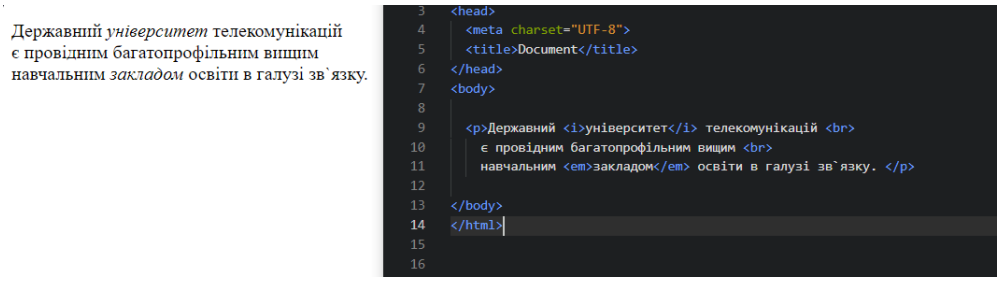



Рисунок 2.6.2 – Оформлення за допомогою <i> та 

Тег <pre> використовується для вставки попередньо відформатованого тексту в HTML-документ. У тексті, який включений в цей тег, зберігаються всі прогалини та розриви рядків, і браузери відображають їх без об'єднання пробілів, що йдуть поспіль, як це роблять за замовчуванням.

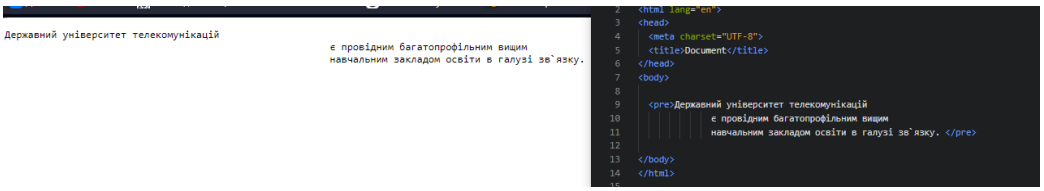



Рисунок 2.6.3 – Оформлення за допомогою <pre>

Тег <small> встановлює розмір тексту шрифту, який є на один розмір менше, ніж у батьківського елемента. У стандарті HTML5 цей тег також використовується для включення інформації щодо авторських прав, а також для визначення дрібного або юридичного шрифту.

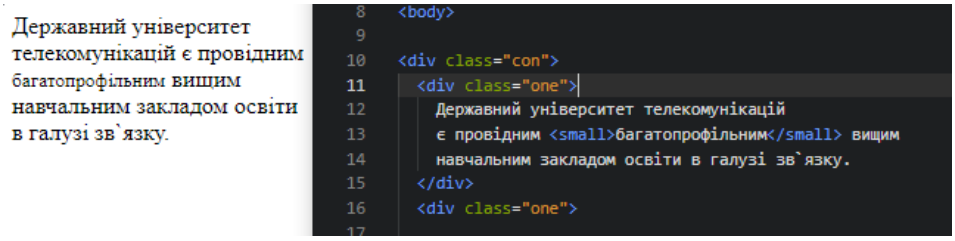



Рисунок 2.6.4 – Оформлення за допомогою <small>

Тег <del> використовується для визначення частини тексту, яка була вилучена з документа. Тег <s> використовується для позначення тексту, який втратив актуальність або більше не є важливим у контексті.

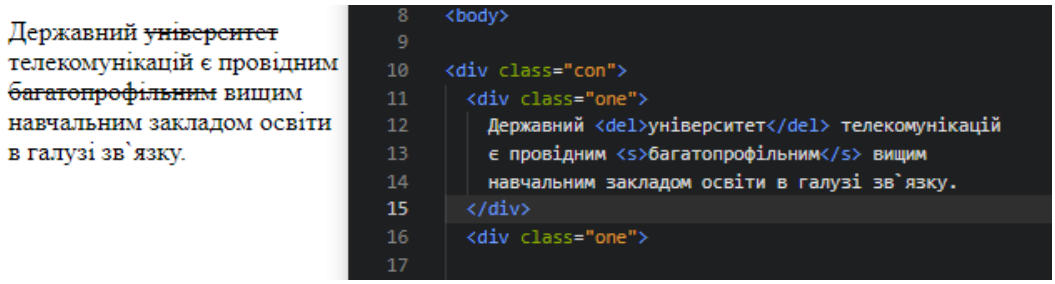



Рисунок 2.6.5 – Оформлення за допомогою <del> та <s>

## 2.7 Зображення

Атрибут src є необхідним, оскільки він вказує шлях зображення. Значенням атрибута src може бути ім'я файлу або його URL. Атрибут alt також є обов'язковим для тега <img>. Його значенням є текст, що пояснює, який показується в браузері до завантаження зображення. Браузер також відображає цей текст під час наведення миші на зображення.


Рисунок 2.7.1 – Вставка зображення

## 2.8 Списки

Мова гіпертекстової розмітки підтримує три типи списків, для кожного з яких використовуються свої теги.

Маркований список містить ненумеровані елементи без певної послідовності. Для створення маркованого списку використовується блоковий елемент <ul>.

Кожен елемент списку починається з тега відкриття <li> і закінчується закриваючим тегом </li>..

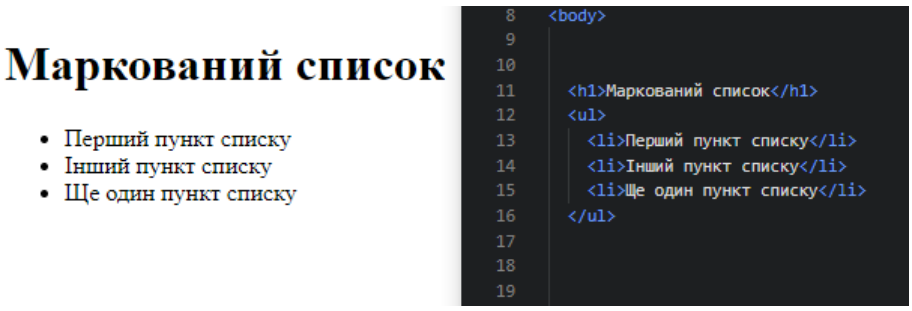



Рисунок 2.8.1 – Маркований список

Нумерований список містить елементи певної послідовності і використовує блоковий елемент <ol>. Кожен елемент нумерованого списку починається з тега <li> (відкривається) і закінчується закриваючим тегом </li>. Пункти списку автоматично отримують номери.

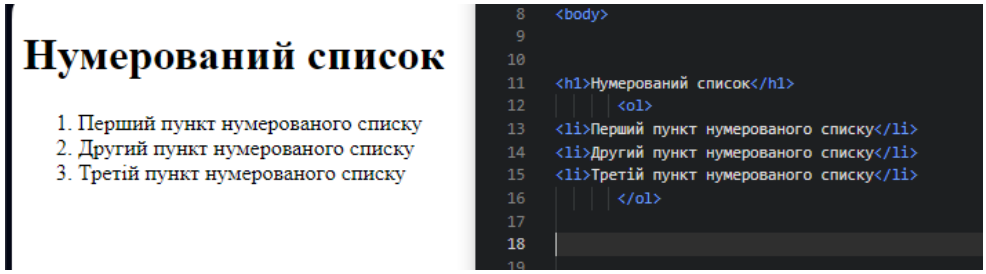



Рисунок 2.8.2 – Нумерований список

### 2.9 Таблиці HTML

Для створення таблиць у HTML використовується елемент <table>, який є контейнером для елементів, що визначають вміст таблиці.Рядки таблиці визначаються за допомогою парного блокового тегу <tr> (від англ. "table row" - рядок таблиці). Кожен рядок таблиці записується окремо за допомогою тегу <tr>.

Елементи у рядках, або комірки, визначаються з використанням тегу <td> (від англ. "table data" - дані таблиці). Кожна комірка представлена окремим тегом <td>, в якому знаходиться вміст таблиці, такий як числа, текст і т.д. Заголовок рядка чи стовпця таблиці визначається тегом <th> (від англ. "table head" - заголовок таблиці)

і розміщується в першому рядку таблиці. В браузері він автоматично вирізняється жирним шрифтом.

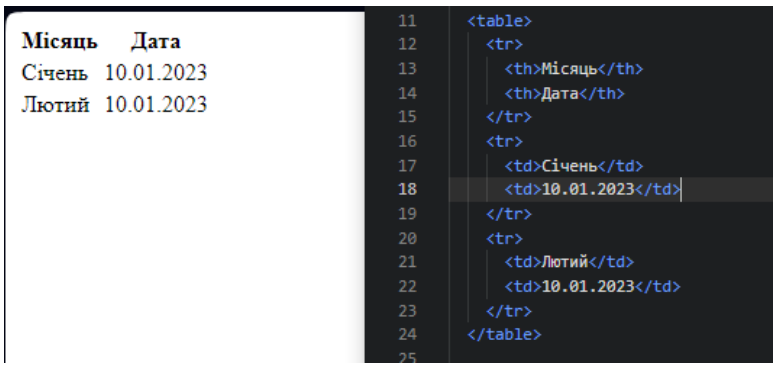



Рисунок 2.9.1 – Таблиця

## 2.10 Коментарі

HTML коментарі використовуються для написання пояснень, нагадувань або додаткових описів до відрізків HTML коду. Вони сприяють полегшенню роботи з кодом, дозволяючи швидко знайти необхідні частини і розібратися в намірах програміста, який писав код (це особливо важливо, коли над створенням веб-сайту працює кілька людей). Коментарі також можна використовувати для тимчасового вилучення будь-якого блоку коду, не видаляючи його. Достатньо лише обгорнути цей відрізок у теги <!-- ... -->, і браузер не відображатиме його.

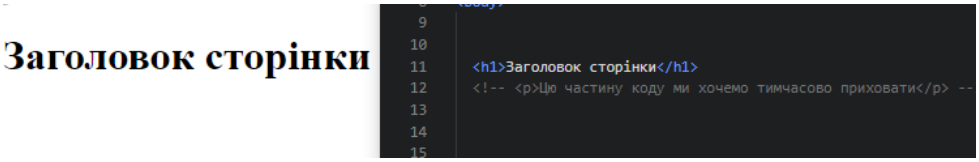



Рисунок 2.10.1 – Коментарі

# 3 CSS ЯК ІНСТРУМЕНТ ОФОРМЛЕННЯ

## 3.1 Введення в CSS

CSS (каскадні таблиці стилів) використовується для стилізації та компонування веб-сторінок. Цей інструмент дозволяє змінювати шрифт, колір, розмір та інтервал вмісту, розбивати його на кілька стовпців, додавати анімацію та інші декоративні елементи. Модуль CSS надає чудову можливість ознайомитися з основами цієї технології, вивчити синтаксис та почати використовувати CSS для надання стилів HTML.

CSS є однією з ключових мов у вільній веб-розробці та є стандартизованою специфікацією W3C. Стандарт CSS поділяється на кілька рівнів: CSS1 зараз вважається застарілим, CSS2.1 рекомендований для використання, а CSS3, розбитий на різні модулі, продовжує розвиватися в рамках стандартизації.

## 3.2 Робота із CSS

Спочатку нам слід повідомити HTML-документ, що ми використовуємо певні CSS-правила. Існують три основні методи застосування CSS до HTML-документу, але зараз ми розглянемо найбільш звичайний і корисний спосіб - використання зв'язування CSS із заголовком вашого документа.

Для цього створюється файл у тій же папці, що й документ HTML, styles.css. Розширення .css показує, що це файл CSS. Щоб зв'язати styles.css в index.html, додається наступний рядок десь усередині <head> HTML документа:

```
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" href="style2.css">
</head>
<body>
```

Рисунок 3.2.1 – Підключення стилів до документа

### 3.3 Селектори CSS

CSS-селектори використовуються для стилізації HTML елементів на веб-сторінці. Існує широкий спектр селекторів CSS, що дозволяє точно вибирати елементи для стилізації. Селектори представляють собою вирази, які вказують браузеру, які елементи HTML слід стилізувати, використовуючи визначені властивості CSS всередині блоку оголошення стилів.

Розуміння того, який саме селектор потрібен, дуже допомагає підібрати потрібний елемент.

До цієї групи належать **селектори HTML-елементів**, таких як <h1>.


Рисунок 3.3.2 – Селектор HTML-елементів

До групи належать і **селектори класів**:

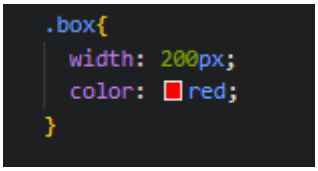



Рисунок 3.3.3 – Селектор класів

**Селектори ідентифікаторів** (ID):


Рисунок 3.3.4 – Селектор класів

До цієї групи належать псевдокласи, що стилізують певний стан елемента. Псевдоклас :hover, наприклад, застосовує правило, тільки якщо на елемент наведено курсор миші. Приклади псевдокласів:

- :link - стиль для посилань, на які ще не було здійснено переходу;
- :visited - стиль для посилань, які вже були відвідані;
- :hover - стиль для елементів, на які наведено курсор;
- :focus – стиль для інтерактивних елементів, які отримали фокус через клавішу Tab;
- :active — стиль для активованих елементів;
- :valid — стиль для форм, які успішно пройшли перевірку на відповідність типу даних;
- :invalid — стиль для форм, чий вміст не відповідає вказаному типу;
- :enabled – стиль для всіх активних форм;
- :disabled — стиль для форм, які вимкнуті і неактивні;
- :in-range – стиль для полів, значення яких знаходяться в певному діапазоні;
- :out-of-range – стиль для полів, значення яких не входять до заданого діапазону;
- :lang() — стиль для елементів, які вказані для певної мови;
- :not(селектор) — стиль для елементів, які не відповідають вказаному селектору;
- :target – стиль для елемента з символом # в посиланні, яке вказує на інший розділ сторінки;
- :checked – стиль для вибраних користувачем елементів.


Рисунок 3.3.1.1 – Псевдоклас

Селектори **структурних псевдокласів** типу вказують на конкретні типи дочірніх елементів:

• :nth-of-type() – вибирає елемент, враховуючи його тип;

• :first-of-type - вибирає перший елемент вказаного типу;

• :last-of-type - вибирає останній елемент вказаного типу;

• :nth-last-of-type() — вибирає елемент, оглядаючи його розташування та тип;

• :only-of-type - вибирає єдиний елемент вказаного типу серед усіх.

**Селектори псевдоелементів** використовуються для додавання особливих символів чи інформації за допомогою властивості content:

• :first-letter – вибирає першу літеру кожного абзацу, застосовується тільки до блочних елементів;

• :first-line - вибирає перший рядок кожного абзацу, застосовується тільки до блочних елементів;

• :before — виводить зміст, який вставляється перед елементом;

• :after — виводить зміст, який вставляється після елемента.

І остання група селекторів дозволяє об'єднувати їх, щоб легко знаходити конкретні елементи всередині документа. У наступному прикладі ми використовуємо комбінатор дочірніх елементів (>) для знаходження дочірнього елемента <article>:

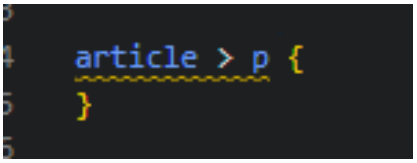



Рисунок 3.3.1.2 – Об’єднання селекторів

### 3.4 Блокова модель

Кожен CSS-елемент вкладений у блок, і розуміння поведінки цих блоків — ключ до уміння задавати розкладку за допомогою CSS, тобто вбудовувати одні елементи щодо інших елементів. Блокові та інлайнові елементи визначаються їхньою поведінкою в контексті потоку сторінки та щодо інших елементів на сторінці.

Якщо елемент визначений як блоковий, то він буде мати такі особливості:

- Починається з нового рядка.
- Розширюється вздовж рядка так, щоб заповнити весь простір, доступний у його контейнері. Зазвичай це означає, що блок буде мати таку саму ширину, як і його контейнер, заповнюючи 100% доступного простору.
- Застосовуються властивості width і height.
- Зовнішні та внутрішні відступи, рамка відсувають інші елементи від нього.

При створенні блокового елемента у CSS ми розглядаємо наступні компоненти:

- Вміст: Це область, де відображається ваш контент. Розмір цієї області можна налаштувати за допомогою властивостей, таких як width та height.
- Внутрішній відступ: Він визначає відступи, розташовані навколо вмісту як порожній простір. Розмір цих відступів контролюється padding x.
- Рамка: Рамка оточує вміст та внутрішні відступи. Розмір та стиль рамки можна налаштувати за допомогою властивостей, таких як border .
- Зовнішній відступ: Це зовнішній шар, що містить вміст, внутрішній відступ і рамки. Його розмір контролюється за допомогою властивостей, таких як margin.


Рисунок 3.4.1 – Компоненти блоку

У звичайній блоковій моделі, якщо вказані атрибути width і height для елемента, це встановлює ширину та висоту його вмісту. Після цього будь-які відступи та рамки додаються до цих значень, щоб отримати загальний розмір елемента.

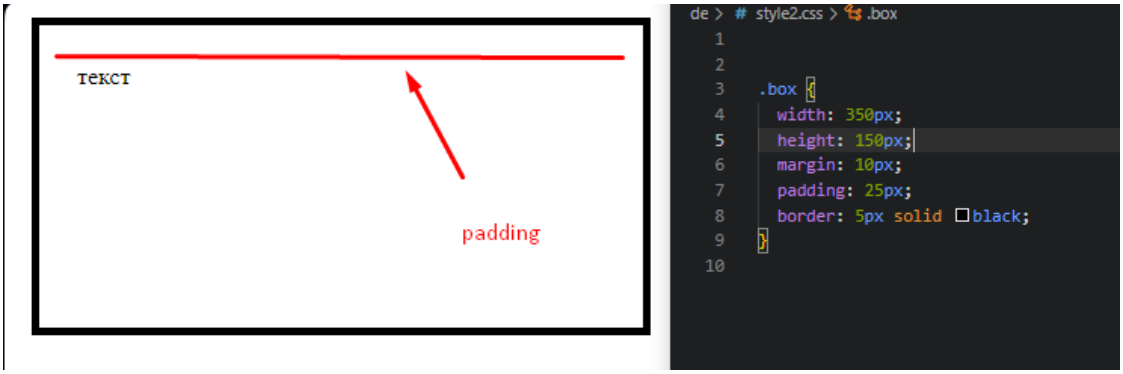



Рисунок 3.4.2 – Внутрішні відступи

Зовнішній відступ представляє собою невидимий простір, який оточує ваш елемент, відштовхуючи інші елементи від нього. Цей відступ може бути як позитивним, так і негативним. Використання від'ємного значення може спричинити перекриття деяких елементів сторінки. Незалежно від того, чи використовується стандартна чи альтернативна блокова модель, зовнішній відступ завжди додається після обчислення розміру видимого блоку.

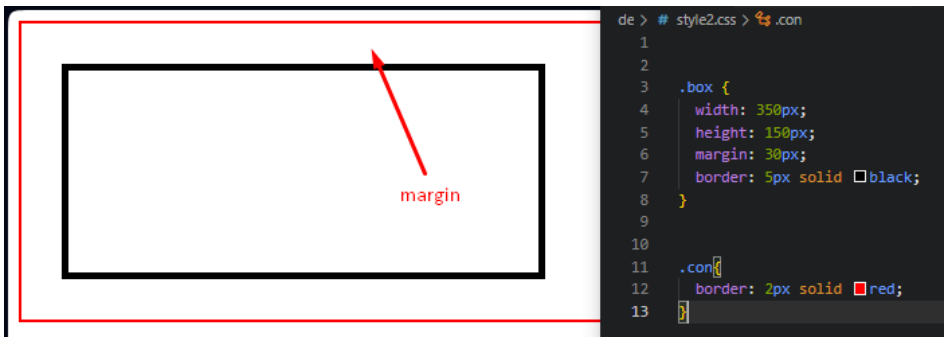



Рисунок 3.4.3 – Зовнішні відступи

### 3.5 Фон та кордони

Властивість background-color встановлює колір фону для CSS-елемента. Ця властивість приймає будь-який допустимий колір <color>. Колір, встановлений через background-color, застосовується до самого контенту і відступів від нього (padding).


Рисунок 3.5.1 – Зміна кольору фона

Властивість background-image дозволяє використовувати зображення в якості фону для елемента.

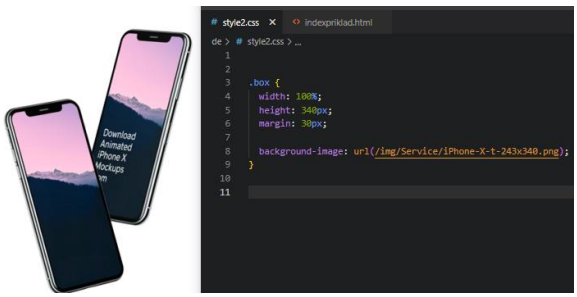



Рисунок 3.5.1 – Фон картинка

Властивість background-repeat регулює спосіб повторення фонового зображення. Її можна налаштовувати за допомогою наступних значень:

- no-repeat: призводить до припинення повторення фонового зображення у всіх напрямках.
- repeat-x: викликає повторення фонового зображення горизонтально.
- repeat-y: спричиняє повторення фонового зображення вертикально.
- repeat: призначає повторення фонового зображення як по горизонталі, так і по вертикалі. Це значення встановлено за замовчуванням.

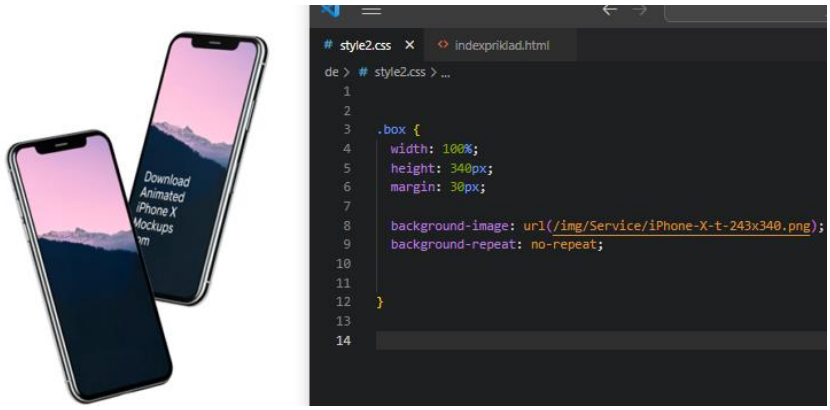



Рисунок 3.5.2 – Вимкнення повтору фона

### 3.6 Позиціювання

Позиціонування дозволяє відсторонювати елементи від стандартного потоку документа та змінювати їх поведінку, наприклад, розташовувати один над іншим або зберігати на тому ж місці всередині вікна браузера.

Існує різні типи позиціонування, які можна використовувати для елементів HTML. Для задання конкретного типу позиціонування для елемента використовується властивість position.

**Статичне позиціонування** є значенням за замовчуванням, яке отримує кожен елемент і просто вказує "розмістіть елемент у його стандартному положенні в потоці документа".

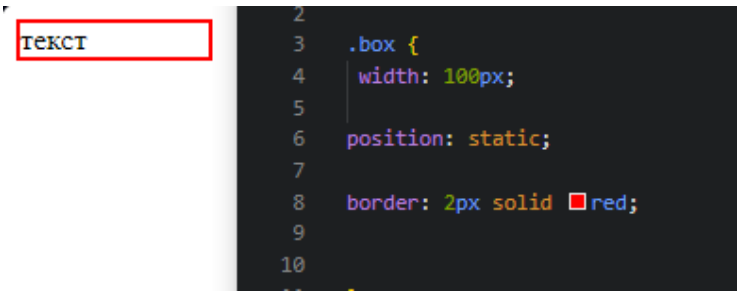



Рисунок 3.6.1 – Позиціювання статичне

**Відносне позиціонування** - це перший тип позиціонування, який ми розглядаємо. Воно дуже схоже на статичне позиціонування, за винятком того, що ви можете змінювати остаточне положення об'єкта, який позиціонується і займає своє місце в макеті нормального потоку. Це включає можливість перекривання інших елементів на сторінці.

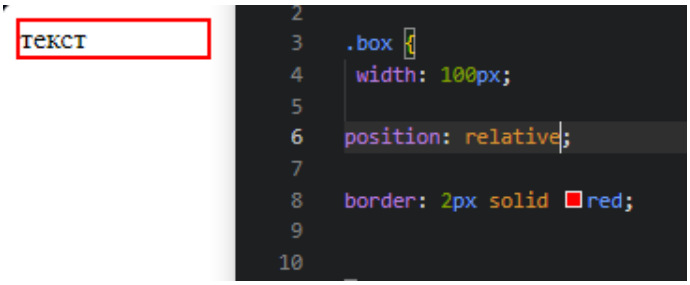



Рисунок 3.6.1 – Позиціювання відносне

**Абсолютне позиціювання** - витягує елемент із загального потоку, при цьому визначається нове місце відносно його непереміщуваного батьківського елемента. Якщо такого батьківського елемента немає, то нове місце визначається відносно вікна браузера (при цьому ширина елемента встановлюється автоматично за його вмістом). Розташування обчислюється на основі властивостей left, top, right та bottom. Вплив на положення взятий у рахунок значення властивості position батьківського елемента: якщо вона встановлена як static або батька немає, відлік координат ведеться від краю вікна браузера.

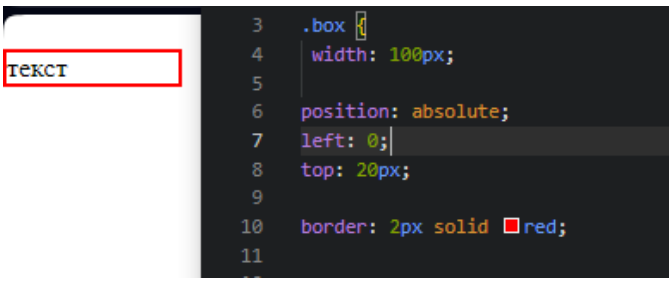



Рисунок 3.6.1 – Позиціювання абсолютне

### 3.7 Обтікання

Обтікання за допомогою властивості float дозволяє зсувати блок вліво або вправо, в залежності від потрібного результату. Плаваючий блок буде зміщуватися в напрямку, заданому властивістю float, до тих пір, поки його край не досягне краю блока або іншого елемента, який перешкоджає.

Для правильного використання float рекомендується вказати ширину плаваючого блоку, щоб виділити місце для додаткового контенту. Коли місце для плаваючого елемента закінчується по горизонталі, він розпочинає зміщення вниз, інші елементи відсукаються навколо нього.

Основні умови для регулювання поведінки плаваючих блоків визначаються за допомогою властивості float. Розміри блоку враховують рамку та внутрішні відступи, але зовнішні відступи контенту не з'єднуються.

### 3.8 Z-index

Властивість CSS z-index визначає порядок накладання позиціонованого елемента, його дочірніх елементів або флекс-елементів вздовж осі z. Елементи з великим значенням z-index будуть перекривати елементи з меншим значенням z-index, встановлюючи порядок їх відображення на екрані.

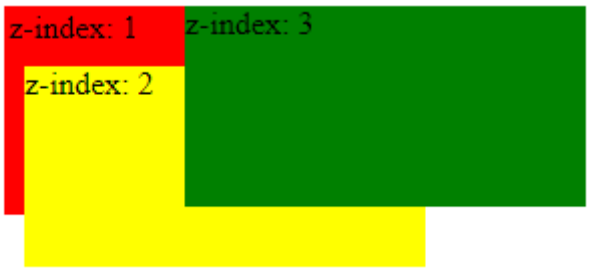



Рисунок 3.8.1 – Порядок накладання

### 3.9 Робота із текстом

**Властивість color** установлює колір тексту або вмісту переднього плану для вибраних елементів. Ця властивість визначає кольоровий спектр для тексту, а також може впливати на інші елементи переднього плану, такі як підкреслення або накладення, розміщені на тексті за допомогою text-decoration.

color:red

color:green

color:green

Рисунок 3.9.1 – Задання кольору тексту

**Властивість font-size** в CSS визначає розмір шрифту для конкретного елемента. Вона дозволяє встановити розмір тексту на веб-сторінці та зазвичай вказується в одиницях виміру, таких як пікселі (px).

font-size: 20px

font-size: 28px

font-size: 34px

Рисунок 3.9.2 – Задання розміру тексту

**Властивість text-decoration** в CSS визначає стиль оформлення тексту, такий як підкреслення, лінії над текстом або лінії під текстом. Ця властивість дозволяє змінювати зовнішній вигляд тексту в залежності від дизайну сторінки.

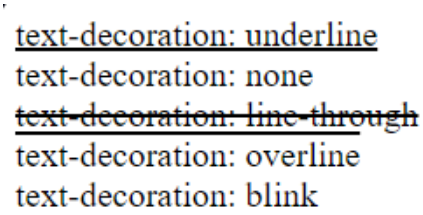



Рисунок 3.9.3 – Властивість text-decoration

**Властивість text-align** в CSS визначає спосіб горизонтального вирівнювання тексту всередині блокового елемента. Вона використовується для визначення того, як текст буде розташований горизонтально в межах свого контейнера.

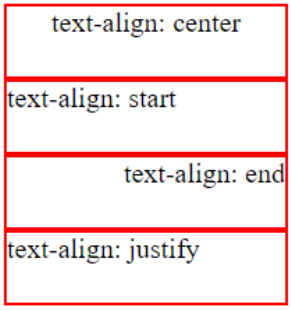



Рисунок 3.9.4 – Властивість text-align

В CSS **властивість font-weight** використовується для визначення товщини шрифту тексту. Вона регулює, наскільки тонким або жирним буде відображений текст на веб-сторінці. Ця властивість приймає числові значення в діапазоні від 100 до 900, ключові слова, такі як "normal" і "bold", або спеціальне значення "bolder".

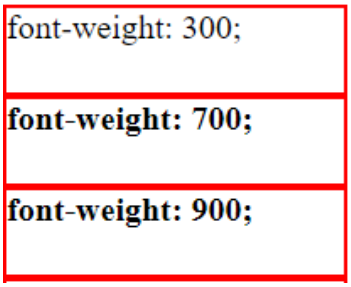



Рисунок 3.9.5 – Властивість fon-weight

**Властивість line-height** в CSS визначає простір між рядками тексту всередині блокового елемента. Ця властивість дозволяє контролювати вертикальний інтервал між рядками тексту, що впливає на вигляд та читабельність текстового контенту всередині елемента.

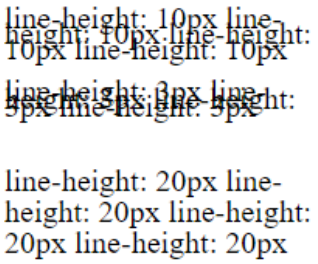



Рисунок 3.9.6 – Властивість line-height

**Властивість letter-spacing** у CSS встановлює інтервал між символами тексту всередині блокового елемента. Вона надає можливість контролювати відстань між літерами з метою досягнення певного ефекту оформлення.

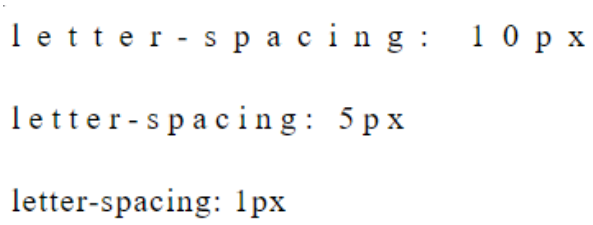



Рисунок 3.9.7 – Властивість letter-spacing

Властивість text-shadow в CSS дозволяє надавати тексту тінь всередині блокового елемента. Використання цієї властивості дозволяє створювати різноманітні ефекти тіні, що поліпшують видимість тексту та роблять його більш привабливим на веб-сторінках.

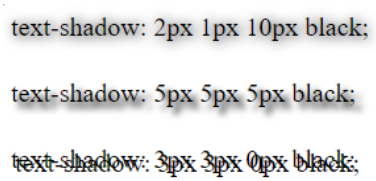



Рисунок 3.9.8 – Властивість text-shadow

Властивість text-transform в CSS дозволяє змінювати відображення тексту щодо його регістру. Вона корисна для стилізації текстового контенту відповідно до дизайнерських вимог. Синтаксис виглядає так:

```
TEXT-TRANSFORM: UPPERCASE;

text-transform:lowercase;

Text-Transform:Capitalize
```

Рисунок 3.9.9 – Властивість text-transform

# 4 JAVASCRIPT ЯК РЕСУРС ДЛЯ ПОКРАЩЕННЯ САЙТУ

## 4.1 Основні поняття JavaScript

JavaScript є мовою програмування, яка дозволяє реалізовувати складну взаємодію на веб-сторінках. Кожен раз, коли ви переглядаєте веб-сторінку, вона не обмежується простим відображенням статичного вмісту. JavaScript дозволяє додавати динаміку, оновлювати вміст в реальному часі, вбудовувати інтерактивні карти, а також створювати 2D/3D анімації.

Ядро JavaScript включає низку стандартних можливостей, що дозволяють виконувати наступне:

- Маніпуляція DOM (Document Object Model): Завдяки JavaScript можна динамічно змінювати структуру та вміст HTML-документів, що дозволяє створювати веб-сторінки з живою інтерактивністю.
- Обробка подій: Мова дозволяє створювати реакції на події, такі як кліки, натискання клавіш, завантаження сторінки і багато інших, що робить можливим створення інтерактивних інтерфейсів.
- Змінні: За допомогою ключових слів, таких як var, let або const, JavaScript дозволяє створювати та використовувати змінні для зберігання даних.
- Структури даних: Вбудовані структури даних, такі як масиви та об'єкти, допомагають організовувати та обробляти інформацію ефективно.
- Функції: JavaScript підтримує визначення та виклик функцій, що дозволяє структурувати код та використовувати його повторно.
- Асинхронне програмування: Використовуючи проміси та функції зворотного виклику, JavaScript ефективно обробляє асинхронні операції, наприклад, отримання даних з сервера.

- Обробка помилок: JavaScript має механізми для ефективної обробки винятків, що дозволяє вирішувати ситуації, коли виникають помилки.
- Взаємодія з HTTP-запитами: З використанням об'єкта XMLHttpRequest або методів Fetch API, JavaScript може взаємодіяти з сервером, отримуючи або відправляючи дані асинхронно.
- Взаємодія з браузерними API: JavaScript може використовувати різноманітні API браузера для доступу до різноманітних функцій, таких як робота з локальним сховищем, геолокацією, камерою та іншими.

### 4.1.1 Відомості про DOM

DOM (Document Object Model або Модель Об'єктів Документа) - це структура, яка відображає структуру HTML-документа або XML-документа у вигляді дерева об'єктів, доступного для взаємодії з використанням JavaScript чи інших мов програмування. DOM надає програмістам зручний інтерфейс для динамічних змін вмісту, структури та стилів веб-сторінки.

Основні концепції та складові DOM:

- Document: Представляє весь документ та його вміст. Це основний об'єкт, від якого розпочинається вся структура DOM.
- Element: Представляє HTML-теги на сторінці, такі як , <p>.
- Attribute: Характеристика елемента, така як атрибут "class" чи "id". Ці характеристики можна змінювати та зчитувати за допомогою JavaScript.
- Text Node: Представляє текстовий вміст всередині елемента.
- Attribute Node: Представляє атрибут елемента.
- Method: Методи дозволяють взаємодіяти з DOM. Наприклад, getElementById, getElementsByTagName.

Операції, які можна виконати з використанням DOM:

- Зміна вмісту: Додавання, видалення або зміна елементів та їх вмісту .

- Зміна атрибутів: Можливість змінювати значення атрибутів елементів відкриває можливості для динамічної зміни характеристик елементів.
- Зміна стилів: Динамічне змінення стилів елементів дозволяє адаптувати вигляд веб-сторінки в реальному часі.
- Взаємодія з подіями: Додавання обробників подій, таких як клік чи наведення миші.
- Зміна структури: Додавання чи видалення елементів для зміни структури дерева DOM.

### 4.1.2 Обробник подій

JavaScript надає можливості для призначення обробників подій, які визначають поведінку коду у відповідь на різні дії користувача або системи.

Основні концепції обробки подій:

- Визначення обробника подій: Обробники подій - це функції, які викликаються у відповідь на певну подію. Наприклад, клік мишею, натискання клавіші, завантаження сторінки.
- Об'єкт події: Кожен обробник подій отримує об'єкт події, який містить інформацію про саму подію.
- Видалення обробників: Для уникнення можливих конфліктів та звільнення ресурсів, обробники подій можна видаляти за допомогою методу removeEventListener.
- Зупинка обробки події: Метод event.preventDefault() використовується для зупинки стандартної поведінки браузера.

### 4.1.3 Змінні

Змінні в JavaScript служать як контейнери для зберігання значень, таких як числа, які можуть використовуватися у математичних операціях, або рядки, які можуть бути використані як текстові дані у складних конструкціях. Однак,

особливість змінних полягає в тому, що їхні значення можуть змінюватися протягом виконання програми.

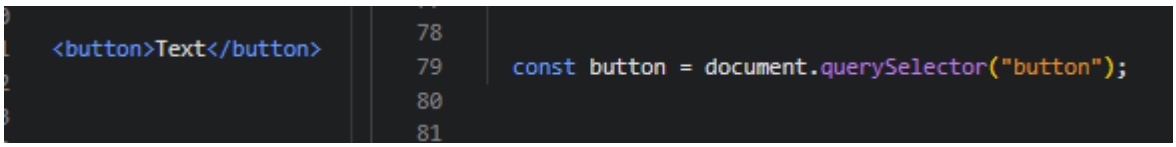



Рисунок 4.1.3.1 – Оголошення змінної JS, яка містить зв’язок із button, яка оголошена в HTML

4.1.4 Структура даних

В JavaScript є декілька структур даних, а саме:

**Масиви** в JavaScript представляють собою упорядковані колекції значень, які можуть включати елементи будь-якого типу даних.

```
let myArray = [1, 'Hello', true, 3];
```

Рисунок 4.1.4.1 – Оголошення масиву

**Об'єкти** в JavaScript представляють собою невпорядковані колекції пар ключ-значення. Ключі можуть бути рядками або символами, а значення можуть мати будь-який тип даних.

```
let myObject = {
  name: 'Yurii',
  age: 22,
  isStudent: true
};
```

Рисунок 4.1.4.2 – Оголошення об’єкта

**Стрічка** в JavaScript – це послідовність символів, яку обмежують лапки.

```
let hello = 'Hello, World!';
```

Рисунок 4.1.4.3 – Оголошення стрічки

**Числа** включають цілі числа та числа з плаваючою комою.

```
let numb = 22;
```

Рисунок 4.1.4.4 – Оголошення числа

**Мапа** в JavaScript - це структура даних, яка представляє собою колекцію пар ключ-значення, де ключі можуть бути представлені будь-яким типом даних.

```
let myMap = new Map();
myMap.set('name', 'Yurii');
myMap.set('age', 22);
```

Рисунок 4.1.4.5 – Оголошення мапи

4.1.5 Функції

Функція в мові JavaScript - це спеціальний тип об'єкта, який дозволяє формалізувати логіку поведінки та обробки даних в межах мови програмування.

Оголошення функції, включає ключове слово function та наступні елементи:

- Ім'я функції.
- Список параметрів, приймаються функцією, укладений у круглі дужки ().
- Інструкції, які будуть виконані після виклику функції, укладені у фігурні дужки { }.

```
function square(number) {
  return number + number;
}
```

Рисунок 4.1.5.1 – Оголошення функції

Існують різні види функцій JavaScript:

- Оголошення функції (Function Declaration):

```
function myFunction(parameter1, parameter2) {
  // тіло функції
}
```

Рисунок 4.1.5.2 – Оголошення функції Declaration

- Функціональний вираз (Function Expression):

```
const myFunction = function(parameter1, parameter2) {
    // тіло функції
};
```

Рисунок 4.1.5.3 – Оголошення функції Expression

- Анонімна функція (Anonymous Function):

```
const myFunction = function(parameter1, parameter2) {
    // тіло функції
};
```

Рисунок 4.1.5.4 – Оголошення функції Anonymous

- Функція зворотного виклику (Callback Function):

```
function SUMAXY(x, y, suma) {
    const result = x + y;
    suma(result);
}

SUMAXY(2, 3, function(result) {
    console.log('Результат операції:', result);
});
```

Рисунок 4.1.5.5 – Оголошення функції Callback

- Стрілкова функція (Arrow Function):

```
const addNumbers = (a, b) => a + b;
```

Рисунок 4.1.5.6 – Оголошення функції Arrow

### 4.2 Підключення JavaScript

JavaScript використовується в HTML-сторінці аналогічно CSS. Так само, як CSS використовує елементи <link> для зовнішніх стилів і <style> для вбудованих у HTML, для JavaScript потрібен лише один елемент у світі HTML — елемент <script>. Існує два варіанти підключення JS до документа HTML. Перший варіант це додавання коду на пряму, в документі HTML:

```
<script>
    alert("код");
</script>
```

Рисунок 4.2.1 – Підключення напряму

Другий варіант, це підключення зовнішнього файлу, в якому і буде писатися весь код:


Рисунок 4.2.2 – Підключення зовнішнього файлу

## 4.3 JQuery в полегшенні роботи із JS

jQuery - це бібліотека, розроблена на базі мови програмування JavaScript.

Представлена у 2006 році, jQuery зараз широко використовується на різних веб-сайтах, особливо на тих, що побудовані на платформі WordPress. Можна розглядати jQuery як фреймворк, який надає багато інструментів для ефективної роботи з веб-документами.

Ця бібліотека значно полегшує вирішення різних типових завдань, таких як валідація форм, створення слайдерів і т. д. Використання jQuery дозволяє значно швидше вирішувати такі завдання порівняно з чистим JavaScript, завдяки великому набору готових засобів.

Основні функції jQuery:

- Використання різноманітних правил таблиць стилів CSS.
- Обробка подій в документі.
- Створення анімацій та застосування різних ефектів в документі.
- Маніпуляція та відображення об'єктів у DOM-структурі сторінки.
- Застосування технології AJAX.

### 4.3.1 Селектори jQuery

Селектори в jQuery служать для вибору та управління HTML елементами. Розглянемо основні селектори:

**Селектор елементів** jQuery вибирає елементи на основі імені елемента. В даному прикладі, по натисканню. Кнопки, будуть приховуватись всі елементи «p».

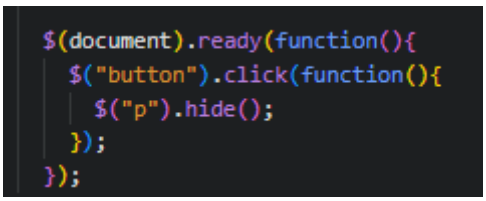



Рисунок 4.3.1.1 – Приклад селектора елементів

**Селектор #id** в jQuery використовує атрибут id HTML-тегу для знаходження конкретного елемента. В даному прикладі, елемент у якого стояв #id, зникне по натисканню кнопки.


Рисунок 4.3.1.2 – Приклад селектора #id

**Селектор .class** в jQuery використовується для знаходження елементів з конкретним класом. В даному прикладі, елементи у яких був прописаний даний клас, зникнуть по натисканню кнопки.


Рисунок 4.3.1.3 – Приклад селектора .class

4.3.2 Події

Події в бібліотеці jQuery дозволяють реагувати на дії користувача або на зміни в документі. Робота з подіями включає встановлення обробників подій, які викликаються при настанні конкретних подій. Покращення взаємодії із Html, за допомогою JQuery .

Робота з обробниками подій включає встановлення їх за допомогою методу .on() або використання скорочених форм, таких як .click(), .hover(), і так далі.

```
$('#myElement').on('click', function() {
    // Код, який викликається при кліку на елемент
});
```

Рисунок 4.3.2.1 – Приклад встановлення обробника для події кліку

В кожному обробнику подій передається об'єкт події, що містить деталі щодо самої події.

```
$('#myElement').on('click', function(event) {
    console.log('Координати кліку:', event.pageX, event.pageY);
});
```

Рисунок 4.3.2.2 – Приклад використання об'єкта події для отримання координат кліку.

Метод event.preventDefault() використовується для блокування стандартної поведінки браузера, що пов'язаної з певною подією. Нижче подано приклад використання для зупинки переходу за посиланням:

```
$('#myLink').on('click', function(event) {
    event.preventDefault();
    // Код, який викликається при кліку, зупиняючи перехід за посиланням
});
```

Рисунок 4.3.2.3 – Приклад використання для зупинки переходу за посиланням.

### 4.4 Під’єднання JQuery

Для включення бібліотеки jQuery до веб-сайту вставляється посилання в розділ <head> вашого HTML-документа. Ось приклад тегу <script>, який можна використовувати для підключення jQuery з офіційного CDN.

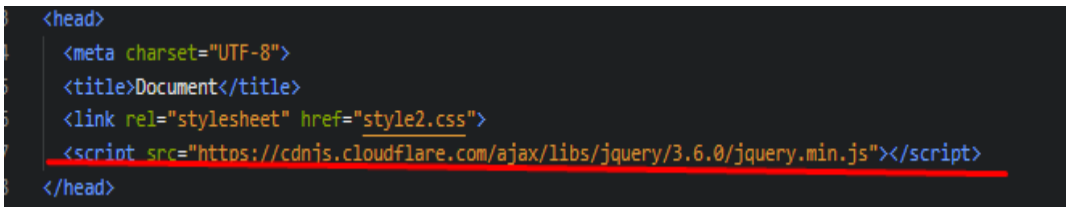



Рисунок 4.4.1 – Приклад підключення JQuery

Також, можна підключити локальний файл, якщо завантажити бібліотеку на свій ПК.

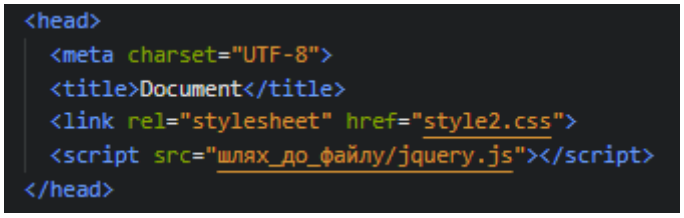



Рисунок 4.4.2 – Приклад підключення JQuery локально

# 5 РОЛЬ PHP У СТВОРЕННІ WEB-САЙТУ

PHP - це мова програмування загального призначення, яку широко використовують для створення веб-застосунків та скриптів. Вона використовується для розробки динамічних веб-сайтів, обробки веб-форм, взаємодії з базами даних та вирішення різноманітних завдань у галузі веб-розробки.

Основні функціональності PHP включають:

- Створення динамічних веб-сайтів: PHP дозволяє розробляти веб-сторінки, які змінюються динамічно в залежності від користувацьких дій або інших факторів, що сприяє створенню інтерактивних та динамічних веб-сайтів.
- Обробка форм: Зручна інтеграція з HTML-формами дозволяє PHP легко обробляти дані, що вводяться користувачами через веб-форми. Це є ключовим для реалізації функцій зворотного зв'язку, авторизації та збереження користувацької інформації.
- Взаємодія з базами даних: PHP надає інструменти для з'єднання з різними системами управління базами даних та виконання операцій з даними, таких як додавання, зчитування, оновлення та видалення.
- Обробка зображень: Завдяки вбудованим функціям для обробки та маніпулювання зображеннями, PHP стає ефективним інструментом для розробки графічних додатків та роботи з мультимедійним контентом.
- Робота з файлами: PHP дозволяє читати та записувати в файли на сервері, що спрощує управління файловою системою та зберігання даних.
- Створення кукі та сесій: Мова дозволяє зберігати стан інформації між різними запитами користувачів за допомогою кукі та сесій.
- Обробка помилок: PHP надає засоби для ефективної обробки помилок, дозволяючи виводити корисні повідомлення та застосовувати заходи безпеки.
- Взаємодія з XML: PHP може взаємодіяти з XML-документами, здійснювати їх обробку та генерацію.

### 5.1 Змінні в PHP

У PHP використовуються змінні для збереження та опрацювання даних. Основні риси змінних в PHP включають:

**Іменування**: Назва для змінної у PHP починається із символу долара ($) та може включати літери, цифри та підкреслення. Потрібно враховувати регістр літер (на приклад, різниця між $var та $Var).

```
$example_variable;
```

Рисунок 5.1.1 – Іменування

**Присвоєння значень**: Значення призначаються змінним за допомогою оператора присвоєння (=).

```
$name = "John";
$age = 25;
```

Рисунок 5.1.2 – Присвоєння значення

**Типи даних**: У PHP визначається мова із слабкою типізацією, що означає, що тип даних змінної може змінюватися під час виконання програми. У цій мові типи даних можуть бути примітивними, такими як рядки (string), цілі числа (integer), числа з плаваючою комою (float), булеві значення (boolean), або складні, такі як масиви (array), об'єкти та інші.

```
$stringVar = "Hello, World!";
$intVar = 22;
$floatVar = 3.14;
$boolVar = true;
```

Рисунок 5.1.3 – Різні типи даних

**Динамічні змінні**: у PHP дають можливість використовувати їх динамічно для назв функцій та для отримання доступу до властивостей об'єктів.

**Глобальні та локальні змінні:** в PHP можуть бути оголошені на рівні глобального контексту (поза функціями та класами) або на рівні локального контексту (в межах функцій та блоків коду).

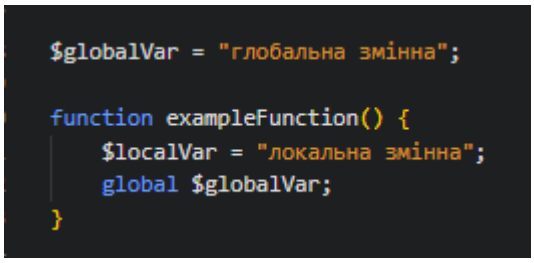



Рисунок 5.1.4 – Оголошення глобальних і локальних змінних

## 5.2 Функції PHP

У PHP функції представляють собою блоки коду, які виконують конкретні завдання та можуть бути викликані з інших частин програми. Основні аспекти функцій в PHP включають:

Оголошення функції:

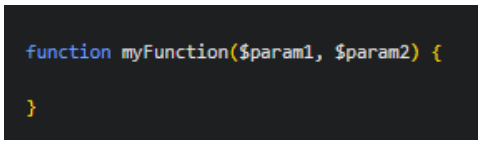



Рисунок 5.2.1 – Оголошення функції

Параметри функції:

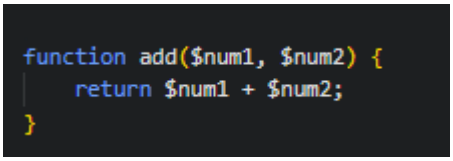



Рисунок 5.2.2 – Параметри функції

Повернення значень за допомогою return:

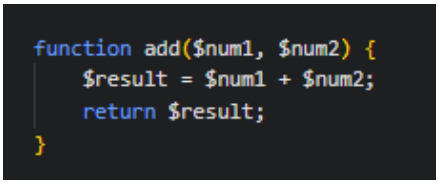



Рисунок 5.2.3 – Повернення значень

# 6 ЕТАПИ СТВОРЕННЯ ВЕБ-СТОРІНКИ

## 6.1 Створення макету

Першою задачею, стояв аналіз майбутньої інформаційної системи для сервісу по ремонту, для визначення необхідних елементів, та створення відповідного дизайну. Для створення макету був використаний додаток Figma.

Figma - програма для розробки інтерфейсу, із різними можливостями створення та відтворення різних ефектів, та корегування можливої спільної роботи в реального часу. Сервіс являться абсолютно безкоштовним, та легким в розуміні, завдяки простому інтерфейсу.

Після повного завантаження додатку із офіційного сайту та його встановлення, нас зустрів інтерфейс програми, який представлена на рисунку 6.1.1

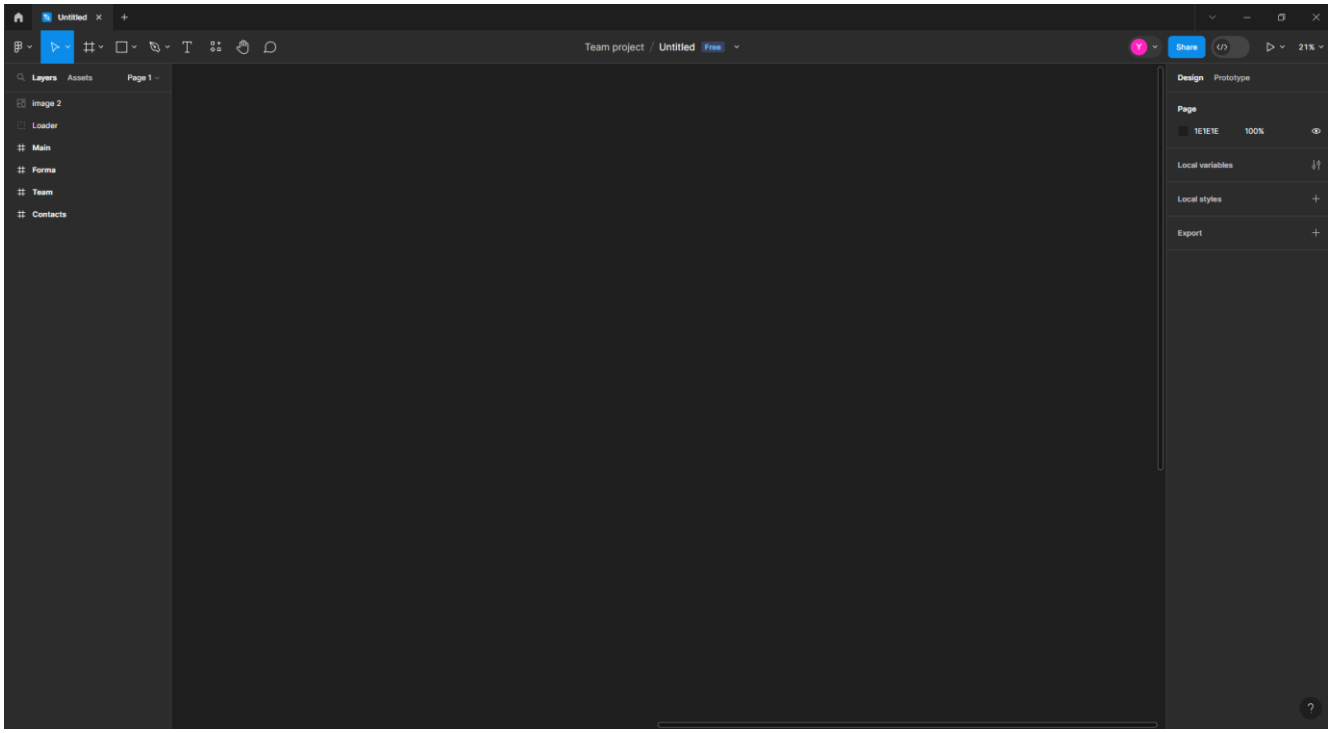



Рисунок 6.1.1 – Інтерфейс Figma

Після повного ознайомлення із інтерфейсом програми, було створено п’ять елементів майбутньої структури сайту, а саме:

Лоадер, який буде відповідати за повне завантаження всіх елементів сайту, і тільки після повного завантаження елементів, він буде зникати, і допускати до контенту сторінки.

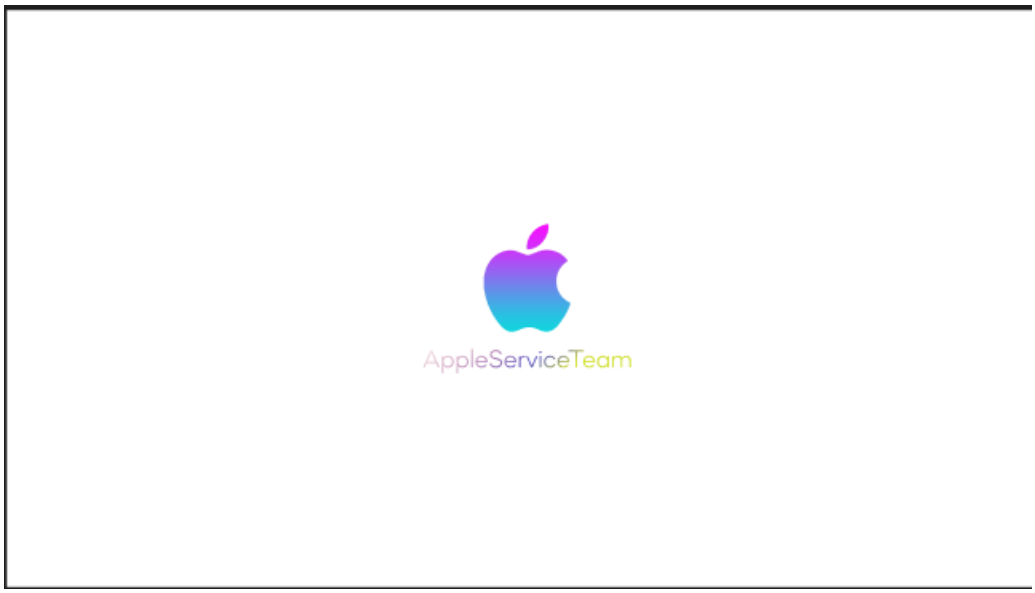



Рисунок 6.1.2 – Макет лоадера

Головна сторінка, є першою сторінкою, яку відвідувач бачитиме при переході за посиланням або введенні URL-адреси. Ця сторінка містить заголовок, логотип, основне меню навігації та інші важливі елементи, щоб користувачі могли швидко зорієнтуватися та отримати інформацію чи послуги, які їм потрібні.


Рисунок 6.1.3 – Головна сторінка

Сторінка із формою, яка буде відповідати за заявки користувачів на ремонт техніки, де розташовані елементи форми куди потрібно вносити ім’я, телефон, пошту та опис проблеми, після чого інформація буде передаватись на сервер.

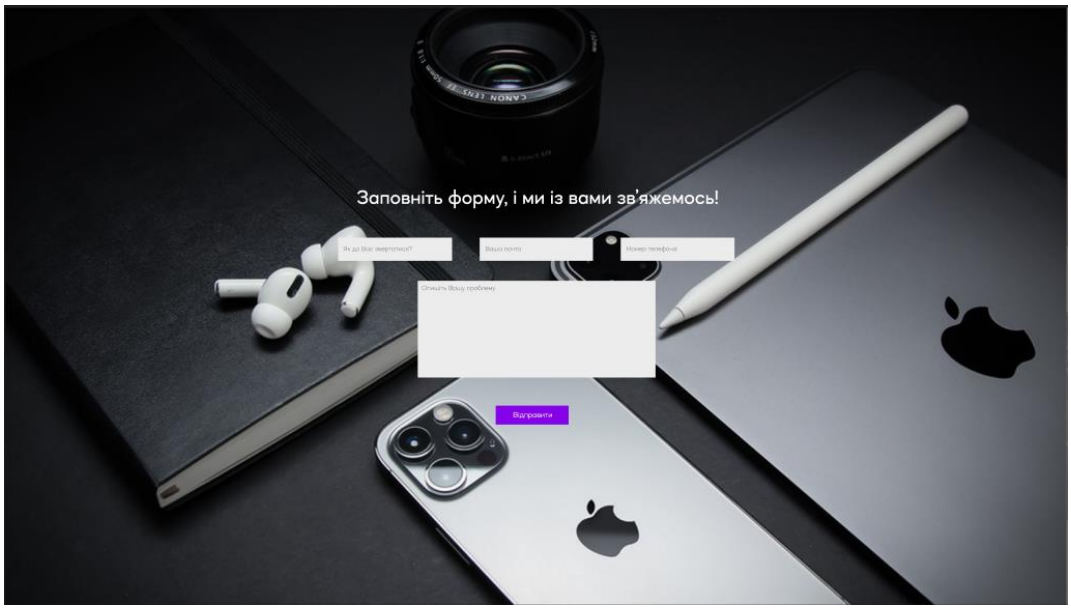



Рисунок 6.1.4 – Сторінка із формою

Наступний елемент, це сторінка із командою, де будуть розташовані керівники та спеціалісти сервісу.

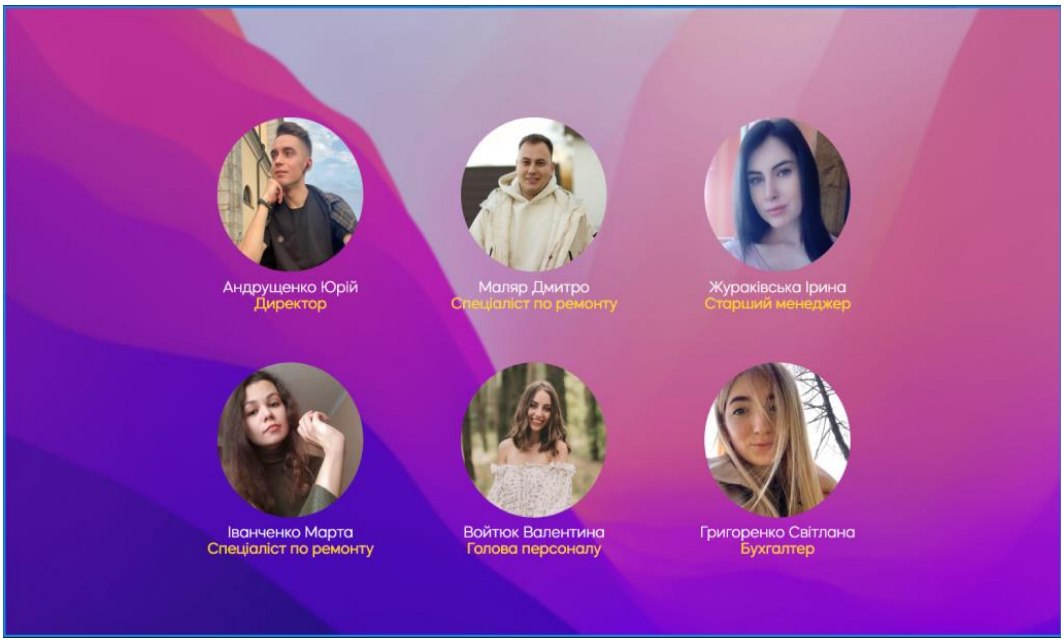



Рисунок 6.1.5 – Сторінка із командою

І останній елемент, це сторінка із контактною інформацію, де розташована карта із місцем знаходження організації, графік роботи та телефон.

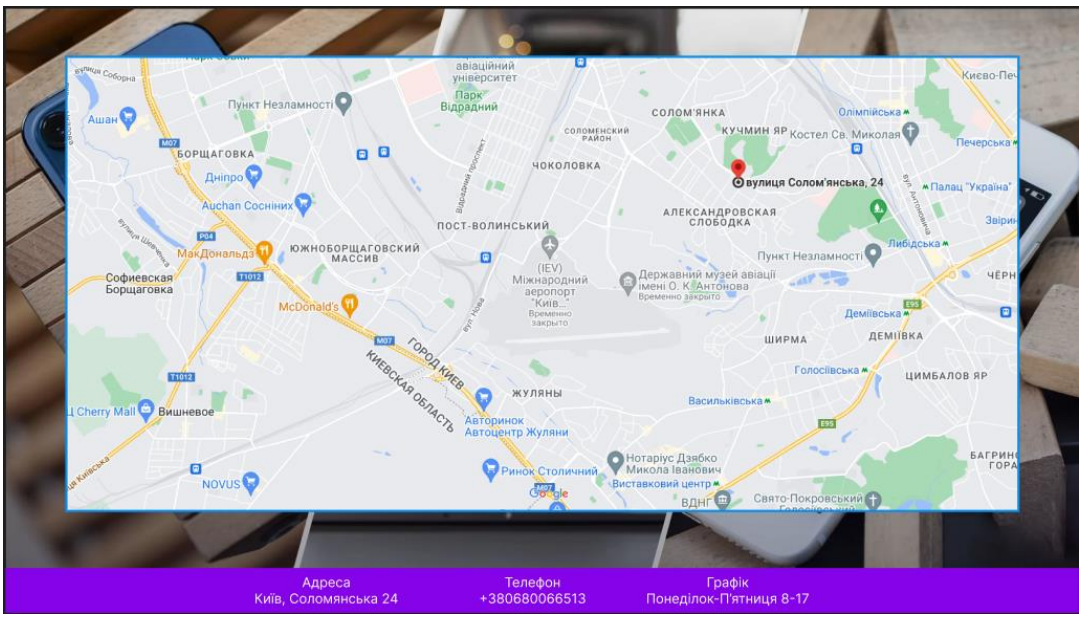



Рисунок 6.1.6 – Сторінка із контактною інформацією

## 6.2 Робота із редактором Visual Studio Code

Створивши макет, переходимо до відтворення структури нашої сторінки за допомогою мови HTML та редактору Visual Studio Code.

Visual Studio Code (VS Code) - це безкоштовний та відкритий текстовий редактор, розроблений Microsoft. Він має багатий функціонал, включаючи вбудовану підтримку для різноманітних мов програмування, розширень і інструментів, що полегшують написання коду, редагування і відлагодження.

Після завантаження додатку із офіційного сайту, та повного встановлення, розпочинаємо роботу із відтворенням структури. Інтерфейс програми представлений на рисунку 6.2.1.

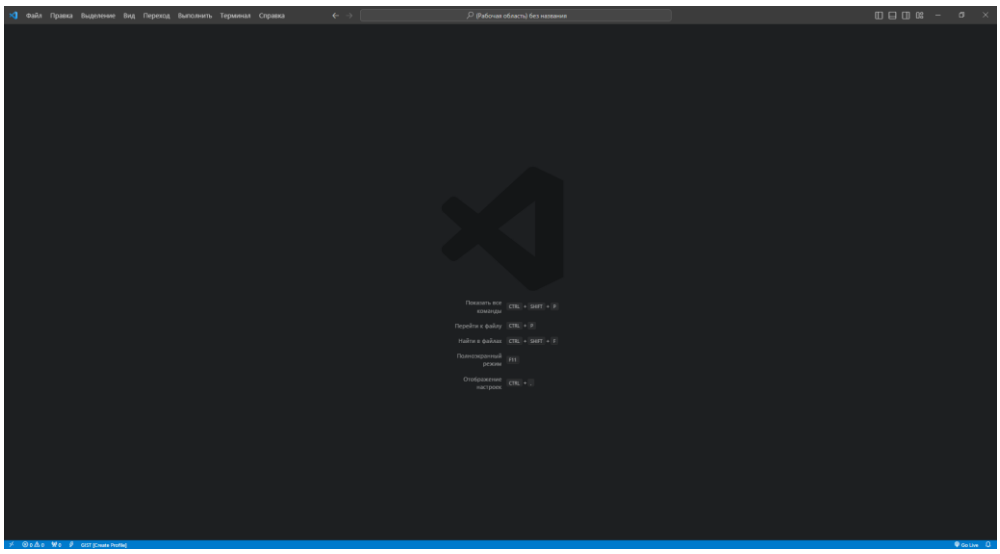


Рисунок 6.2.1 – Інтерфейс програми Visual Studio Code

Перший етап, це створення чотирьох файлів це:

- Index.html – який буде відповідати за структуру нашого елемента.
- Style.css – допоможе оформити нашу сторінку.
- App.js – до дасть динамічності та інтерактивності.
- Mail.php – допоможе реалізувати функцію відправки повідомлень на пошту.

### 6.2.1 Створення структури документа

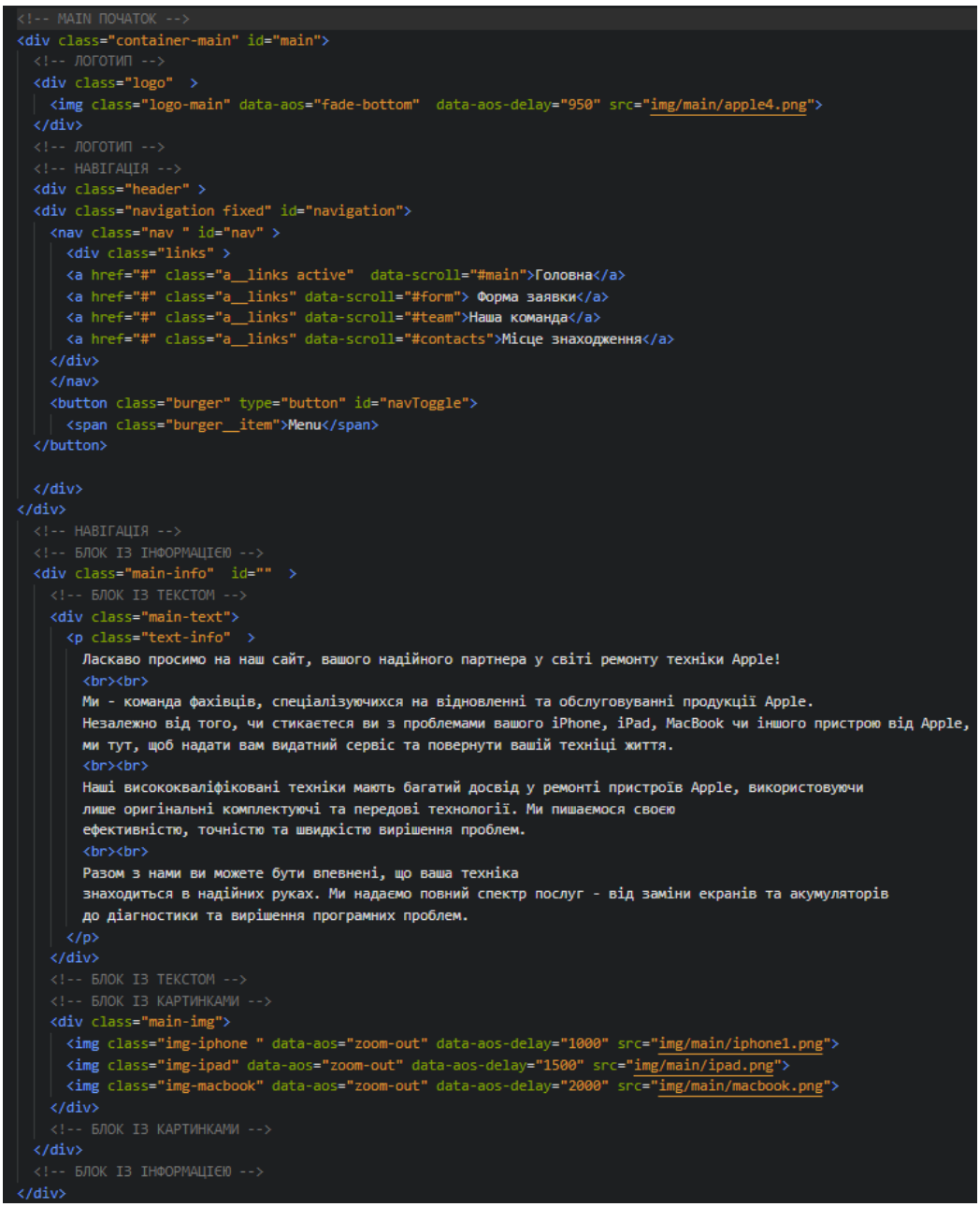
Розпочинаємо роботу із файлом Index.html, підключаємо зовнішній файл стилів «style.css», вказуємо кодування символів сторінки UTF-8 , задаємо мову сторінки, визначаємо заголовок сторінки, який буде відображатися в рядку заголовка браузера, встановлюємо іконку для веб-сайту, яка буде відображатися в рядку адреси браузера, під’єднуємо бібліотеку jquery, яка дозволить легко взаємодіяти із DOM та виконувати різні завдання за допомогою JavaScript. Визначаємо параметри відображення сторінки на мобільних пристроях, встановлюючи початковий розмір масштабу на 1.


Рисунок 6.2.1.1 – Інтерфейс програми Visual Studio Code

Після підключення всіх елементів, та задання стандартних параметрів, переходимо до створення структури лоадера, який буде з'являтися на екрані користувача під час завантаження веб-сторінки. Блоки ми будемо створювати за допомогою елементу div. В нашій структурі це буде картинка та назва сайту. Структура представлена на рисунку 6.2.1.2.


Рисунок 6.2.1.2 – Структура лоадера

Переходимо до головної сторінки, де помістимо вверху по центру логотип сайту, далі буде розташована наша динамічна  навігація сторінки, текст із описом організації та динамічні картинки. Код представлений на рисунку 6.2.1.3.



```
<!-- MAIN ПОЧАТОК -->
<div class="container-main" id="main">
  <!-- ЛОГОТИП -->
  <div class="logo"  >
    <img class="logo-main" data-aos="fade-bottom"  data-aos-delay="950" src="img/main/apple4.png">
  </div>
  <!-- ЛОГОТИП -->
  <!-- НАВІГАЦІЯ -->
  <div class="header" >
  <div class="navigation fixed" id="navigation">
    <nav class="nav " id="nav" >
      <div class="links" >
      <a href="#" class="a__links active"  data-scroll="#main">Головна</a>
      <a href="#" class="a__links" data-scroll="#form"> Форма заявки</a>
      <a href="#" class="a__links" data-scroll="#team">Наша команда</a>
      <a href="#" class="a__links" data-scroll="#contacts">Місце знаходження</a>
    </div>
    </nav>
    <button class="burger" type="button" id="navToggle">
      <span class="burger__item">Menu</span>
  </button>

  </div>
</div>
  <!-- НАВІГАЦІЯ -->
  <!-- БЛОК ІЗ ІНФОРМАЦІЄЮ -->
  <div class="main-info"  id=""  >
    <!-- БЛОК ІЗ ТЕКСТОМ -->
    <div class="main-text">
      <p class="text-info"  >
        Ласкаво просимо на наш сайт, вашого надійного партнера у світі ремонту техніки Apple!
        <br><br>
        Ми - команда фахівців, спеціалізуючихся на відновленні та обслуговуванні продукції Apple.
        Незалежно від того, чи стикаєтеся ви з проблемами вашого iPhone, iPad, MacBook чи іншого пристрою від Apple,
        ми тут, щоб надати вам видатний сервіс та повернути вашій техніці життя.
        <br><br>
        Наші висококваліфіковані техніки мають багатий досвід у ремонті пристроїв Apple, використовуючи
        лише оригінальні комплектуючі та передові технології. Ми пишаємося своєю
        ефективністю, точністю та швидкістю вирішення проблем.
        <br><br>
        Разом з нами ви можете бути впевнені, що ваша техніка
        знаходиться в надійних руках. Ми надаємо повний спектр послуг - від заміни екранів та акумуляторів
        до діагностики та вирішення програмних проблем.
      </p>
    </div>
    <!-- БЛОК ІЗ ТЕКСТОМ -->
    <!-- БЛОК ІЗ КАРТИНКАМИ -->
    <div class="main-img">
      <img class="img-iphone " data-aos="zoom-out" data-aos-delay="1000" src="img/main/iphone1.png">
      <img class="img-ipad" data-aos="zoom-out" data-aos-delay="1500" src="img/main/ipad.png">
      <img class="img-macbook" data-aos="zoom-out" data-aos-delay="2000" src="img/main/macbook.png">
    </div>
    <!-- БЛОК ІЗ КАРТИНКАМИ -->
  </div>
  <!-- БЛОК ІЗ ІНФОРМАЦІЄЮ -->
</div>
```

Рисунок 6.2.1.3 – Структура головної сторінки

Наступна сторінка це наша форма, де розмістимо саму форму, із текстовими блоками для заповнення імені, пошти користувача, номера телефону, та опису проблеми, із якою звертаються. Після буде розташована кнопка «відправити», щоб наша форма була адресована серверу. Код представлений на картинці 6.2.1.4.

```
<!-- КОНТЕЙРНЕР ФОРМА ПОЧАТОК -->
<div class="container-forma" id="form">
  <!-- ВНУТРШНІЙ КОНТЕЙНЕР ФОРМА -->
  <div class="forma"  data-aos="fade-up"  data-aos-delay="50">
    <!-- ЗАГОЛОВОК ФОРМИ -->
    <div class="tex-forma">Заповніть форму, і ми із вами зв'яжемось!</div>
      <!-- ЗАГОЛОВОК ФОРМИ -->
    <!-- <div class="main-form"> -->
      <!-- ОСНОВНА ФОРМА -->
      <form action="mail.php" method="POST" class="main-form">

      <input type="text" class="name-form" name="user_name" placeholder="Як до Вас звертатися?" required>
      <input type="text" class="email-form" name="user_email" placeholder="Ваша почта" required>
      <input type="text" class="phone-form" name="user_phone" placeholder="Номер телефона" required>

    <div class="textarea-button">
      <textarea class="textarea"  name="description" placeholder="Опишіть Вашу проблему"></textarea>
      <input type="submit" onclick="document.location='thank-you.html'" class="button-form" value="Відправити">
    </div>

    </form>
    <!-- ОСНОВНА ФОРМА -->
  </div>
    <!-- ВНУТРШНІЙ КОНТЕЙНЕР ФОРМА -->
</div>
<!-- КОНТЕЙРНЕР ФОРМА КІНЕЦЬ -->
```

Рисунок 6.2.1.4 – Структура форми

Тепер, перейдемо до контактного блоку, де буде розташований виконавчий склад, директор, спеціалісти по ремонту, менеджера, бухгалтер та голова персоналу. Код представлений на рисунку 6.2.1.5.

```
<!-- КОНТЕЙНЕР КОМАНДА -->
<div class="container-team" id="team">
  <!-- ОСНОВНИЙ КОНТЕЙНЕР КОМАНДИ -->
  <div class="team">
    <!-- КОНТЕЙНЕР ДИРЕКТОРА -->
    <div class="person" data-aos="fade-up"
    data-aos-delay="50">
      <div class="img-person1 img_person"></div>
      <p class="name-person">Андрущенко Юрій</p>
      <p class="job-title">Директор</p>
    </div>
      <!-- КОНТЕЙНЕР ДИРЕКТОРА -->
      <!-- КОНТЕЙНЕР СПЕЦІАЛІСТА -->
    <div class="person" data-aos="fade-up"
    data-aos-delay="100">
      <div class="img-person2 img_person"></div>
      <p class="name-person">Маляр Дмитро</p>
      <p class="job-title">Спеціаліст по ремонту</p>
    </div>
      <!-- КОНТЕЙНЕР СПЕЦІАЛІСТА -->
      <!-- КОНТЕЙНЕР МЕНЕДЖЕРА -->
    <div class="person" data-aos="fade-up"
    data-aos-delay="150">
      <div class="img-person3 img_person"></div>
      <p class="name-person">Жураківська Ірина</p>
      <p class="job-title">Старший менеджер</p>
    </div>
    <!-- КОНТЕЙНЕР МЕНЕДЖЕРА -->
    <!-- КОНТЕЙНЕР ДРУГОГО СПЕЦІАЛІСТА -->
```

```
    <p class="name-person">Жураківська Ірина</p>
    <p class="job-title">Старший менеджер</p>
  </div>
  <!-- КОНТЕЙНЕР МЕНЕДЖЕРА -->
  <!-- КОНТЕЙНЕР ДРУГОГО СПЕЦІАЛІСТА -->
  <div class="person" data-aos="fade-up"
  data-aos-delay="200">
    <div class="img-person4 img_person"></div>
    <p class="name-person">Іванченко Марта</p>
    <p class="job-title">Спеціаліст по ремонту</p>
  </div>
    <!-- КОНТЕЙНЕР ДРУГОГО СПЕЦІАЛІСТА -->
    <!-- КОНТЕЙНЕР ГОЛОВИ ПЕРСОНАЛУ -->
  <div class="person" data-aos="fade-up"
  data-aos-delay="250">
    <div class="img-person5 img_person"></div>
    <p class="name-person">Войтюк Валентина</p>
    <p class="job-title">Голова персоналу</p>
  </div>
  <!-- КОНТЕЙНЕР ГОЛОВИ ПЕРСОНАЛУ -->
  <!-- КОНТЕЙНЕР БУХГАЛТЕРА -->
  <div class="person" data-aos="fade-up"
  data-aos-delay="300">
    <div class="img-person6 img_person"></div>
    <p class="name-person">Григоренко Світлана</p>
    <p class="job-title">Бухгалтер</p>
  </div>
</div>
<!-- КОНТЕЙНЕР БУХГАЛТЕРА -->
  <!-- ОСНОВНИЙ КОНТЕЙНЕР КОМАНДИ -->
</div>
<!-- КОНТЕЙНЕР КОМАНДА -->
```

Рисунок 6.2.1.5 – Структура командного блоку

І останній елемент нашої структури це контактний блок, де розташована мапа за допомогою iframe, графік роботи організації, адреса та телефон. Код представлений на рисунку 6.2.1.6.

```
<!-- КОНТЕЙНЕР МАПИ -->
<div class="container-maps" id="contacts">
  <!-- ОСНОВНИЙ КОНТЕЙНЕР МАПИ -->
  <div class="maps"  data-aos="fade-up"  data-aos-delay="100">
    <iframe class="google-maps" src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d2541.66
    width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="
  </div>
    <!-- ОСНОВНИЙ КОНТЕЙНЕР МАПИ -->
    <!-- КОНТЕЙНЕР ІНФО -->
  <div class="footer">
    <div class="adresa footer-text">Адреса<br>
      Київ, Соломянська 24 </div>
    <div class="telefon footer-text">Телефон<br>+380680066513</div>
    <div class="grafik footer-text">
      Графік <br>Понеділок-П’ятниця 8-17</div>
  </div>
    <!-- КОНТЕЙНЕР ІНФО -->
</div>
<!-- КОНТЕЙНЕР МАПИ -->
```

Рисунок 6.2.1.6 – Структура командного блоку

6.2.2 Оформлення веб-сайту

Після завершення створення структури нашого веб-сайту, переходимо до оформлення наших сторінок, за допомогою мови css, так як наш файл каскадних таблиць стилів підключений. Переходимо в файл style.css, та розпочинаємо задавати стилі.

Для початку, підключимо шрифти за допомогою «@font-face», вони будуть використовуватися на всіх сторінках нашого сайту:

```
@font-face {
  font-family: 'Mazzard Soft H';
  src: url('Fonts/MazzardSoftH-Thin.otf');
  font-weight: 100;
  font-style: normal;
 }

@font-face {
  font-family: 'Mazzard Soft H';
  src: url('Fonts//MazzardSoftH-ExtraLight.otf');
  font-weight: 200;
  font-style: normal;
 }
```

Рисунок 6.2.2.1 – Підключення шрифтів

Далі ми прибираємо смугу прокрутку із нашого сайту, щоб він виглядав більш естетично, видаляємо всі внутрішні і зовнішні відступи у нашого тіла документа, тобто «body», за допомогою margin та padding, задаємо шрифт який буде використовуватись на всіх сторінках завдяки «font-family», иакож забираємо відступи у заголовків «h» та абзаців «p». І встановимо «box-sizing: border-box», щоб обчислення розмірів елементів, а саме їх ширина та висота включали в себе границі та внутрішні відступи а не додавались до них. Код представлений на 6.2.2.2.

```css
/* Вимикаємо скролбар */
 ::-webkit-scrollbar {
  width: 0px;
  background: transparent;
}
/* Вимикаємо скролбар */

/* Забираємо відступи у body */
  body{
      margin: 0;
      padding: 0;
      font-family: 'Mazzard Soft H';
  }
/* Забираємо відступи у body */

/* Забираємо відступи у заголовків і абзаців */
h1,h2,h3,h4,h5,h6, p
{
    padding: 0;
    margin: 0;
}
/* Забираємо відступи у заголовків і абзаців */

/*Нормальний розрахунок ширини і висоти елементів */
  *,
  *:before,
  *:after{
      box-sizing: border-box;
  }
/*Нормальний розрахунок ширини і висоти елементів */
```

Рисунок 6.2.2.2 – Початкові налаштування в css

Переходимо до оформлення нашого лоадера, задаємо йому вирівнювання по центру по горизонталі та вертекалі, задаємо ширину «width» і висоту «height», встановлюємо колір фону «background-color: white;», вирівнюємо фон по центру, та забороняємо його повторення «background-repeat: no-repeat;», створюємо анімацію для нашої картинки завдяки «animation», робимо її безкінечне повторення. Форматуємо текст лоадера, а саме назву організації, задаємо їй розміри

«font-size: 32px;», виставляємо градієнтний колір, та анімацію для нашого тексту.

```
.mask {
  display: flex;
  justify-content: center;
  align-items: center;
  position: absolute;
  width: 100%;
  height: 100%;
  /* background-image: url('img/background/mask-background.jpg
  background-color: white;
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  top: 0;
  z-index: 6;

}
/* ЛОАДЕР, задаємо параметри сторінці лоадера */
/* Вирівнюємо внутрні елементи лоадера */
.loader {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;

}
/* Вирівнюємо внутрні елементи лоадера */
/* Задаємо параметри картинці лоадера, і анімації його */
.imgLoader {
  width: 125px;
  -webkit-animation: scale 2s linear infinite; /* Safari */
  animation: scale 2s linear infinite;
}
```

Рисунок 6.2.2.3 – Оформлення лоадера

```
/* Задаємо параметри тексту лоадера */
.textLoader{
  font-size: 32px;
  font-weight: 300;
  background: linear-gradient(330deg, #e05252 0%, #bf8c26
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}
/* Задаємо параметри тексту лоадера */

/* Анімація зменшення картинки */
@-webkit-keyframes spin {
0% { -webkit-transform:scale(1);  }
25% { -webkit-transform:scale(0.75); }
50% { -webkit-transform:scale(0.60); }
75% { -webkit-transform: scale(0.75);}
100% { -webkit-transform: scale(1); }
}
```

Рисунок 6.2.2.4 – Оформлення тексту лоадера

Прийшов час перейти до головної сторінки, знову задаємо вирівнювання по горизонталі та вертикалі, фіксуємо ширину та висоту, задаємо задній фон, відміняємо його повторення, задаємо зовнішні відступи для наших елементів, задаємо оформлення тексту для навігації та фіксуємо її «position:fixed;».

```css
/* КОНТЕЙНЕР МЕЙН */
.container-main {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  width: 100%;
  height: 100vh;
  background-image: url('img/background/main-bacground.jpg');
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
/* КОНТЕЙНЕР МЕЙН */
/* ОПИСУЄМО ПАРАМЕТРИ ДЛЯ КОНТЕЙНЕРА ЛОГО */
.logo {
  display: flex;
  justify-content: center;

  /* margin-top: 88px; */
  margin-bottom: 71px;
}
/* ОПИСУЄМО ПАРАМЕТРИ ДЛЯ КОНТЕЙНЕРА ЛОГО */
/* ЗАДАЄМО РОЗМІРИ КАРТИНЦІ */
.logo-main{
  width: 70px;
}
/* ЗАДАЄМО РОЗМІРИ КАРТИНЦІ */
/* ПАРАМЕТРИ КОНТЕЙНЕРА НАВІГАЦІЇ */
.navigation {
  display: flex;
  justify-content: center;
  margin-bottom: 71px;
}
/* ПАРАМЕТРИ КОНТЕЙНЕРА НАВІГАЦІЇ */
/* ПАРАМЕТРИ ПРИ ФІКСАЦІЇ НАВІГАЦІЇ */
```

```css
.navigation.fixed {
display: flex;
justify-content: center;
align-items: center;
position:fixed;
margin-top: 0px;
background-color: #8602E8;
left: 0;
top: 0;
width: 100%;
height: 40px;
z-index: 1000;

}
/* ПАРАМЕТРИ ПРИ ФІКСАЦІЇ НАВІГАЦІЇ */
/* ПАРАМЕТРИ ПРИ ФІКСАЦІЇ НАВІГАЦІЇ ПРИ КЛАСІ АКТИВ */
.navigation.fixed .active{
  font-weight: 400;
}
/* ПАРАМЕТРИ ПРИ ФІКСАЦІЇ НАВІГАЦІЇ ПРИ КЛАСІ АКТИВ */
/* ВНУТРІШНІЙ КОНТЕЙНЕР НАВІГАЦІЇ */
.links {
}
/* ВНУТРІШНІЙ КОНТЕЙНЕР НАВІГАЦІЇ */
/* ЗАДАЄМО ПАРАМЕТРИ ДЛЯ НАШИХ СИЛОК */
.a__links {
text-decoration: none;
color: white;
text-transform: uppercase;
font-weight: 200;
margin-right: 30px;
font-size: 18px;
opacity: 1;
}
```

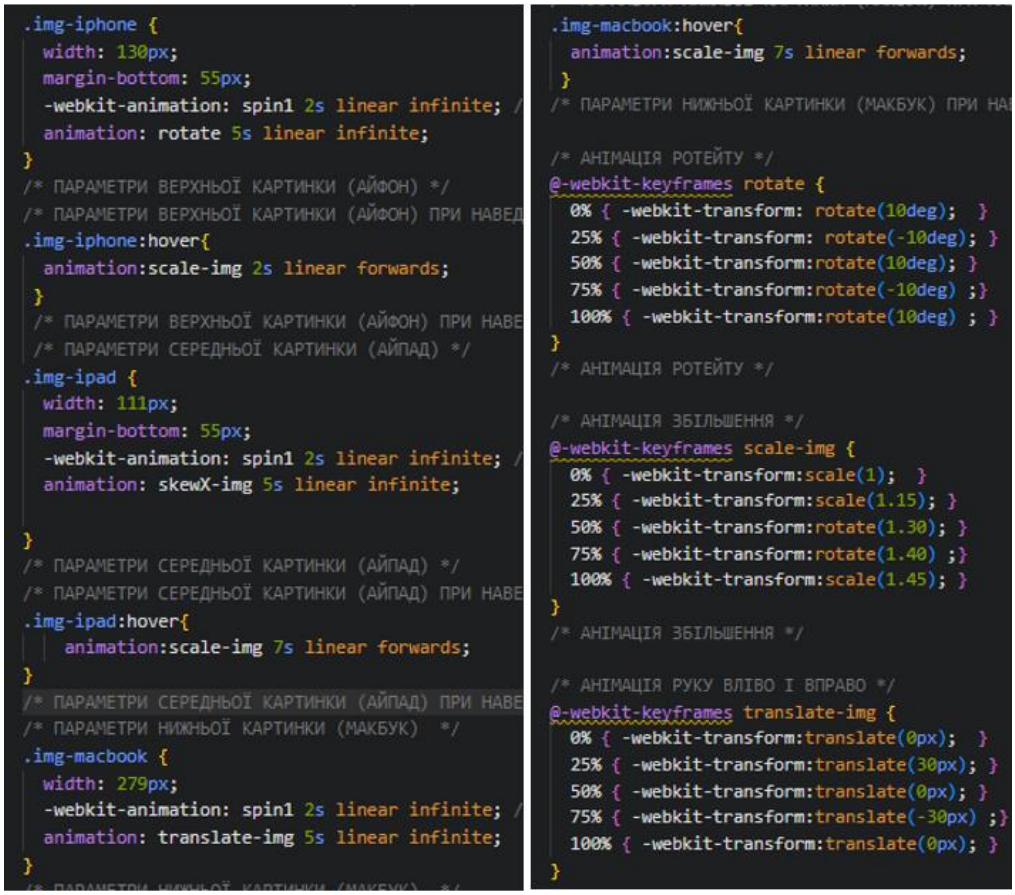
Рисунок 6.2.2.5 – Головна сторінка

Задаємо зміни при натисканні на нашу навігацію, також оформлюємо наш основний текст, задаючи колір, розмір, вимірювання. Оформляємо наші картинки, додаючи їм анімацію.

```css
.main-text {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 50%;
  color: white;
  font-weight:300;
  font-size: 20px;
}
```

```css
.main-img {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  width: 50%;
}
```

Рисунок 6.2.2.6 – Оформлення тексту та блоку картинок

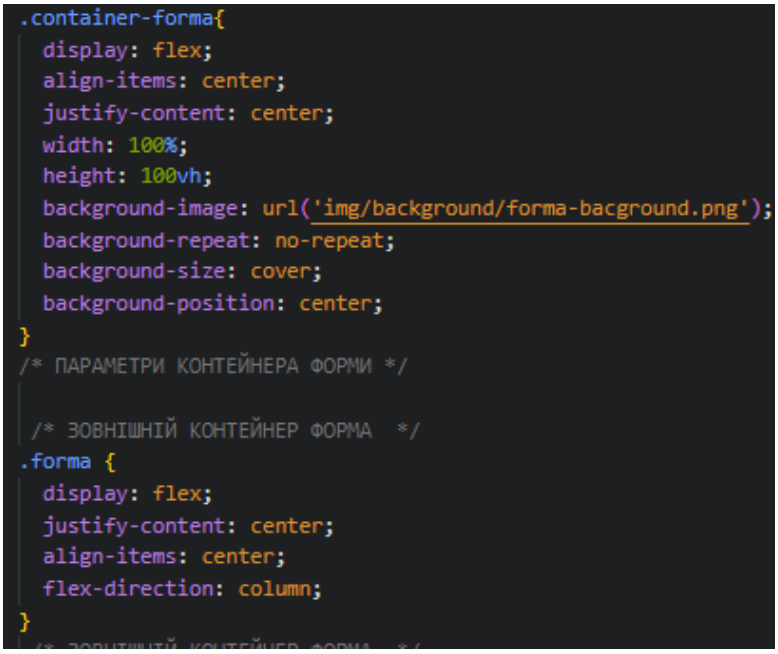


```
.img-iphone {
  width: 130px;
  margin-bottom: 55px;
  -webkit-animation: spin1 2s linear infinite; /
  animation: rotate 5s linear infinite;
}
/* ПАРАМЕТРИ ВЕРХНЬОЇ КАРТИНКИ (АЙФОН) */
/* ПАРАМЕТРИ ВЕРХНЬОЇ КАРТИНКИ (АЙФОН) ПРИ НАВЕД
.img-iphone:hover{
  animation:scale-img 2s linear forwards;
 }
 /* ПАРАМЕТРИ ВЕРХНЬОЇ КАРТИНКИ (АЙФОН) ПРИ НАВЕ
 /* ПАРАМЕТРИ СЕРЕДНЬОЇ КАРТИНКИ (АЙПАД) */
.img-ipad {
  width: 111px;
  margin-bottom: 55px;
  -webkit-animation: spin1 2s linear infinite; /
  animation: skewX-img 5s linear infinite;

}
/* ПАРАМЕТРИ СЕРЕДНЬОЇ КАРТИНКИ (АЙПАД) */
/* ПАРАМЕТРИ СЕРЕДНЬОЇ КАРТИНКИ (АЙПАД) ПРИ НАВЕ
.img-ipad:hover{
    animation:scale-img 7s linear forwards;
}
/* ПАРАМЕТРИ СЕРЕДНЬОЇ КАРТИНКИ (АЙПАД) ПРИ НАВЕ
/* ПАРАМЕТРИ НИЖНЬОЇ КАРТИНКИ (МАКБУК)  */
.img-macbook {
  width: 279px;
  -webkit-animation: spin1 2s linear infinite; /
  animation: translate-img 5s linear infinite;
}
```

```
.img-macbook:hover{
  animation:scale-img 7s linear forwards;
 }
/* ПАРАМЕТРИ НИЖНЬОЇ КАРТИНКИ (МАКБУК) ПРИ НА

/* АНІМАЦІЯ РОТЕЙТУ */
@-webkit-keyframes rotate {
  0% { -webkit-transform: rotate(10deg);  }
  25% { -webkit-transform: rotate(-10deg); }
  50% { -webkit-transform:rotate(10deg); }
  75% { -webkit-transform:rotate(-10deg) ;}
  100% { -webkit-transform:rotate(10deg) ; }
}
/* АНІМАЦІЯ РОТЕЙТУ */

/* АНІМАЦІЯ ЗБІЛЬШЕННЯ */
@-webkit-keyframes scale-img {
  0% { -webkit-transform:scale(1);  }
  25% { -webkit-transform:scale(1.15); }
  50% { -webkit-transform:rotate(1.30); }
  75% { -webkit-transform:rotate(1.40) ;}
  100% { -webkit-transform:scale(1.45); }
}
/* АНІМАЦІЯ ЗБІЛЬШЕННЯ */

/* АНІМАЦІЯ РУКУ ВЛІВО І ВПРАВО */
@-webkit-keyframes translate-img {
  0% { -webkit-transform:translate(0px);  }
  25% { -webkit-transform:translate(30px); }
  50% { -webkit-transform:translate(0px); }
  75% { -webkit-transform:translate(-30px) ;}
  100% { -webkit-transform:translate(0px); }
}
```

Рисунок 6.2.2.7 – Оформлення картинок та додавання анімації

На блок форми задаємо відповідні параметри, розташування по центру, задаємо фіксовану ширину і висоту, задній фон, вирівнювання всіх елементів по центру. Код представлений на рисунку 6.2.2.8.



```
.container-forma{
  display: flex;
  align-items: center;
  justify-content: center;
  width: 100%;
  height: 100vh;
  background-image: url('img/background/forma-bacground.png');
  background-repeat: no-repeat;
  background-size: cover;
  background-position: center;
}
/* ПАРАМЕТРИ КОНТЕЙНЕРА ФОРМИ */

 /* ЗОВНІШНІЙ КОНТЕЙНЕР ФОРМА  */
.forma {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
}
```

Рисунок 6.2.2.8 – Блок форми

Працюємо із тексом форми, задаючи йому розмір «font-size: 36px;» колір «color: white;» тип жирності «font-weight: 400;» та зовнішній нижній відступ «margin-bottom: 36px;». Оформлюємо самі блоки форми, задаючи їм ширину в 240 пікселів та висоту в 42 пікселі, забираємо «border: none;», щоб форми мали елегантний вигляд, та задаємо оформлення нашій кнопці, додаючи їй колір, колір тексту на ній та його розмір. Код представлений на рисунку 6.2.2.9.

```css
.tex-forma {
  color: white;
  font-size: 36px;
  font-weight: 400;
  margin-bottom: 36px;
}
 /* ЗАГОЛОВОК ФОРМИ */
 /* ОСНОВНА ФОРМА */
.main-form{
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: row;
  flex-wrap: wrap;
  width: 1000px;
}
 /* ОСНОВНА ФОРМА */
 /* ПАРАМЕТРИ ФОРМИ ІМ'Я */
.name-form {
  width: 240px;
  height: 42px;
  padding: 0px 10px;
  border: none;
}
 /* ПАРАМЕТРИ ФОРМИ ІМ'Я */

 /* ПАРАМЕТРИ ФОРМИ ПОШТИ */
.email-form {
  width: 206px;
  height: 42px;
  margin-left: 60px;
  margin-right: 60px;
  padding: 0px 10px;
  border: none;
}
 /* ПАРАМЕТРИ ФОРМИ ПОШТИ */

 /* ПАРАМЕТРИ ФОРМИ ТЕЛЕФОН */
.phone-form {
  width: 206px;
  height: 42px;
  padding: 0px 10px;
  border: none;
}
 /* ПАРАМЕТРИ ФОРМИ ТЕЛЕФОН */
```

```css
.textarea-button{
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  flex-wrap: wrap;
}


textarea {
  width: 430px;
  height: 175px;
  padding: 10px 10px;
  margin: 36px 0px;
  background-color: #f8f8f8;
  resize: none;
  font-size: 14px;
}
 /* ПАРАМЕТРИ ФОРМИ ОПИС ПРОБЛЕМИ *

 /* ПАРАМЕТРИ КНОПКИ ВІДПРАВКИ */
.button-form {
  width: 132px;
  height: 34px;
  font-size: 14px;
  font-weight: 300;
  background-color: #8602E8;
  color: white;
  cursor: pointer;
  border: NONE;
}
 /* ПАРАМЕТРИ КНОПКИ ВІДПРАВКИ */
 /*КІНЕЦЬ КОНТЕЙРНЕРА ФОРМА  */
```

Рисунок 6.2.2.9 – Оформлення тексту форми та кнопки

Блок із командним складом оформлюється аналогічно, задаючи центрування, розміри ширини та висоту, завантажуємо картинку на задній фон, вирівнюємо наш фон, задаємо внутрішні відступи «padding: 133px 430px;». Робимо блок стрічковий «flex-direction: row;» із можливість перенесення внутрішніх блоків, якщо вони не будуть коректно відображатися, через малий дисплей. Оформлюємо текст підписів, задаючи розмір шрифту, колір та завантажуємо картинки до наших елементів, встановлюючи їм позиціювання по центру, відміняємо по центру, та відміняємо розтягування. Код представлений на рисунку 6.2.2.10.

```
/* ПОЧАТОК КОНТЕЙНЕРА КОМАНДИ */
/* КОНТЕЙНЕР КОМАНДИ */
.container-team {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100vh;
  background-image: url('img/background/team-b
  background-position: center;
  background-size: cover;
  background-repeat: no-repeat;
  padding: 133px 430px;
}
/* КОНТЕЙНЕР КОМАНДИ */
/* ВНУТРІШНІЙ КОНТЕЙНЕР КОМАНДИ */
.team {
  display: flex;
  align-items: center;
  justify-content: center;
  flex-wrap: wrap;
  flex-direction: row;
  width: 100%;
  height: 100%;
}
/* ВНУТРІШНІЙ КОНТЕЙНЕР КОМАНДИ */
/* КОНТЕЙНЕР ПЕРСОНАЛУ */
.person {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  width: 33.333%;
}
/* КОНТЕЙНЕР ПЕРСОНАЛУ */
/* ФОТО ПЕРШОЇ ЛЮДИНИ */
.img-person1 {
  width: 243px;
  height: 243px;
  border-radius: 50%;
  background-image: url('img/team/yura.jpg');
  background-position: center;
  background-size: cover;
  background-repeat: no-repeat;
}
/* ФОТО ПЕРШОЇ ЛЮДИНИ */
```

```
/* ФОТО ЧЕТВЕРТОЇ ЛЮДИНИ */
.img-person4 {
  width: 243px;
  height: 243px;
  border-radius: 50%;
  background-image: url('img/team/marta.jpg');
  background-position: center;
  background-size: cover;
  background-repeat: no-repeat;
}
/* ФОТО ТЕРТЬОЇ ЛЮДИНИ */
/* ФОТО П'ЯТОЇ ЛЮДИНИ */
.img-person5 {
  width: 243px;
  height: 243px;
  border-radius: 50%;
  background-image: url('img/team/valya.jpg');
  background-position: center;
  background-size: cover;
  background-repeat: no-repeat;
}
/* ФОТО П'ЯТОЇ ЛЮДИНИ */
/* ФОТО ШОСТОЇ ЛЮДИНИ */
.img-person6 {
  width: 243px;
  height: 243px;
  border-radius: 50%;
  background-image: url('img/team/sveta.jpg');
  background-position: center;
  background-size: cover;
  background-repeat: no-repeat;
}
/* ФОТО ШОСТОЇ ЛЮДИНИ */

/* ПІБ ЛЮДИНИ */
.name-person {
  margin-top: 13px;
  color: white;
  font-weight: 300;
  font-size: 24px;
}
/* ПІБ ЛЮДИНИ */
/*ПОСАДА ЛЮДИНИ */
.job-title {
  color: #fdd610;
  font-weight: 400;
  font-size: 24px;
}
/*ПОСАДА ЛЮДИНИ */
```

Рисунок 6.2.2.10 – Задання параметрів для блоку із виконавчим складом

В контактний блок ми додали картку, прийшов час її оформити задаємо їй фіксовану ширину « width: 1700px;»  та висоту «height: 800px;», задаємо позицію нашому футеру із контактною інформацією, а саме абсолютну позицію «position: absolute;» прилягання до нижнього краю сторінки «bottom: 0px;»  ширин «width: 100%» висоту «height: 60px;», задаємо колір фону «background-color: #8602E8;». Задаємо стилі для тексту, білий колір, розмір шрифту та його вирівнювання по центру. Код представлений на рисунку 6.2.2.11.

```
/* ОСНОВНИЙ КОНТЕЙНЕР ФУТЕРА */
.container-maps {
  position: relative;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  height: 100vh;
  width: 100%;
  background-image: url('img/background/contacts-back
  background-position: center;
  background-size: cover;
  background-repeat: no-repeat;
}
/* ОСНОВНИЙ КОНТЕЙНЕР ФУТЕРА */
/* КОНТЕЙНЕР КАРТИ */
.maps {
}
/* КОНТЕЙНЕР КАРТИ */
/* НАЛАШТУВАННЯ СТИЛІВ ВІДОБРАЖЕНЕНЯ КАРТИ */
.google-maps{
  width: 1700px;
  height: 800px;
}
/* НАЛАШТУВАННЯ СТИЛІВ ВІДОБРАЖЕНЕНЯ КАРТИ */
/* ЗАДАННЯ ПОЗИЦІЙ КОНТАКТНІЙ ІНФО */
.footer {
  position: absolute;
  bottom: 0px;
  width: 100%;
  height: 60px;
  background-color: #8602E8;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: row;
}
/* ЗАДАННЯ ПОЗИЦІЙ КОНТАКТНІЙ ІНФО */
/* ТЕКСТ ФУТЕРА */
.footer-text{
  color: white;
  text-align: center;
  font-weight: 200;
}
/* ТЕКСТ ФУТЕРА */
/* ЗАДАННЯ ВІДСТУПІВ ДЛЯ  БЛОКУ ІЗ ТЕЛЕФОНОМ */
.telefon{
  margin: 0px 100px;
}
```

Рисунок 6.2.2.11 – Параметри контактного блоку

Тепер, можна спостерігати результат оформлення сторінок, за допомогою мвои css на рисунках 6.2.2.12-6.2.2.15. Після чого, перейдемо до створення динамічних та інтерактивних функцій для нашого проєкту.

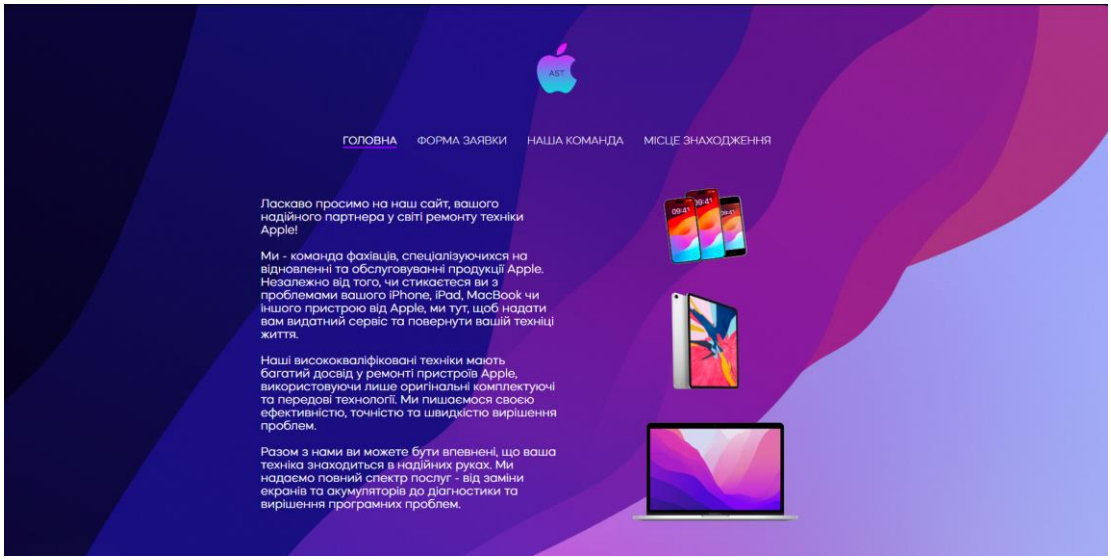



Рисунок 6.2.2.12 – Головна сторінка

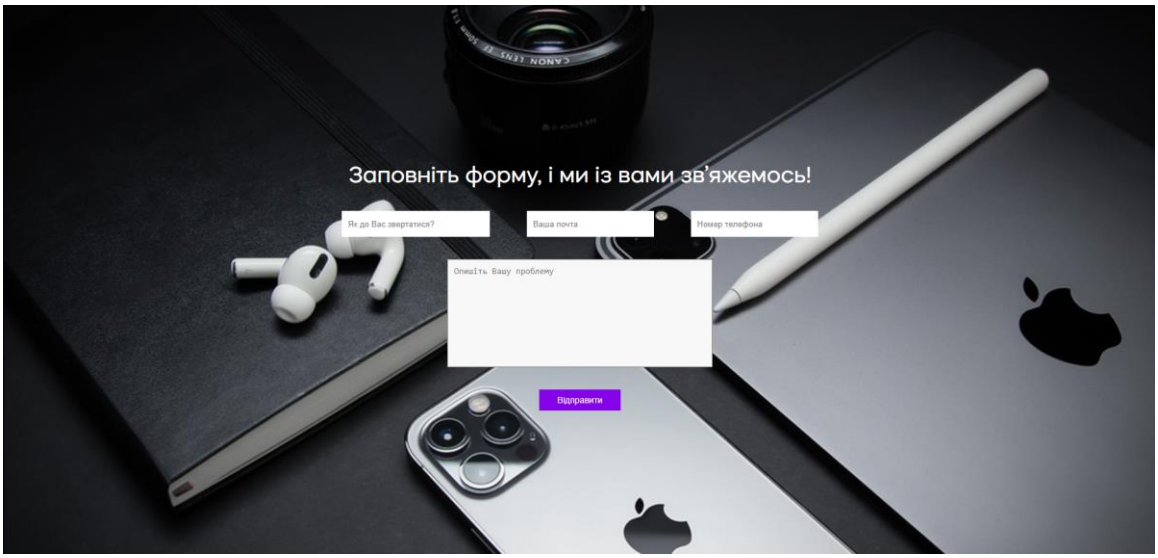



Рисунок 6.2.2.13 – Сторінка із формою

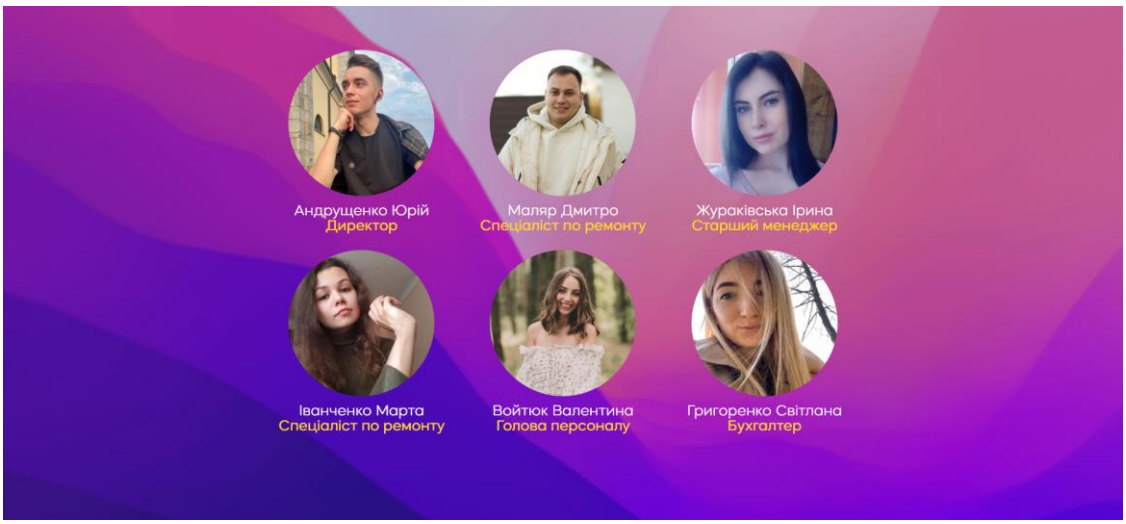



Рисунок 6.2.2.14 – Сторінка із основним складом

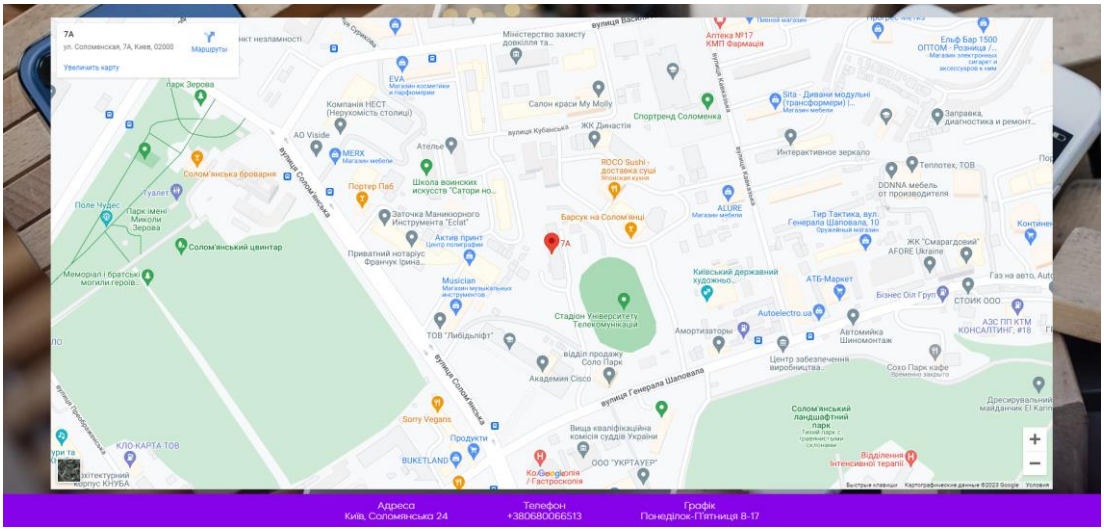



Рисунок 6.2.2.15 – Сторінка із контактним блоком

### 6.2.3 Додавання динамічності та інтерактивності

Переходимо до додавання динамічності нашому веб-сайту за допомогою JS. Для початку зробимо так, щоб наш лоадер зникав, після повного завантаження всіх елементів на сторінці. Для початку створимо зміну, яка буде отримувати посилання на елемент із класом «mask» в документі, додаємо подію «load», яка спрацьовує коли вся сторінка повністю завантажена, включаючи всі ресурси (зображення, стилі і т.д). І коли сторінка буде повністю завантажена, нашій змінні буде присвоюватись клас «hide», який приховує вигляд цього елемента. Далі у нас

встановлений таймер, який через 1 секунду виконає функцію, видаляючи елемент з класом «mask»  із документа. Код представлений на рисунку 6.2.3.

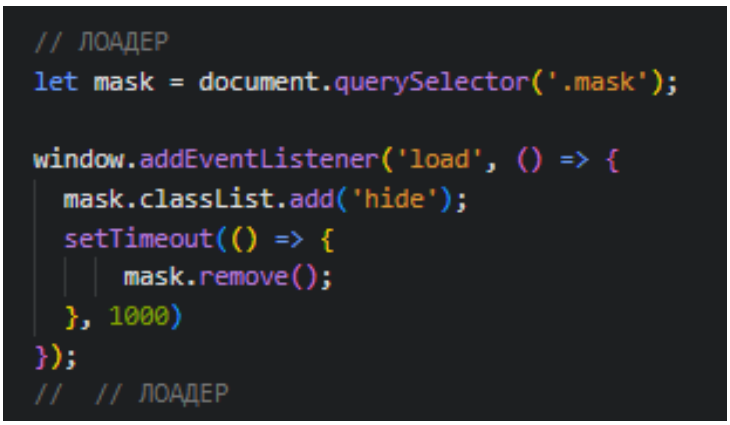



Рисунок 6.2.3.1 – Функція лоадера

Далі додаємо нашій навігацію подію, при якій при натисканні на розділі навігації їй буде додаватися клас, при якому буде додаватися певне оформлення. Для цього створюється змінна link, яка отримує колекцію  елементів з класом "a__links" ( клас який присвоєний нашій навігації), запускаємо цикл for, який перебирає всі елементи, зібрані в link, додаємо обробник подій "click" для кожного елемента. Також, так як по стандарту наш клас вже присвоєний першому елементу навігації, то потрібно створити змінну standart, яка отримує колекцію елементів з класом "active". Це передбачає, що на сторінці є елементи, які можуть мати клас "active". І нам потрібно видаляти клас, який був у попереднього елемента, за допомогою виразу: «standart[0].className = standart[0].className.replace(" active", "")», який буде знімати клас "active" з першого елемента в колекції standart. І тепер коли користувач буде натискати на елементи навігації, то клас буде видалятися із елемента у якого він вже стоїть, і буде додавати його поточному елементу навігації. Код представлений на рисунку 6.2.3.2.

```
//ЗНАХОДИМО НАШУ НАВІГАЦІЮ, І ЗМІНЮЄМО СТАН ACTIVE ПРИ НАТИСНЕННІ
 let link = document.getElementsByClassName("a__links");
for (let i = 0; i < link.length; i++) {
  link[i].addEventListener("click", function() {
    let standart = document.getElementsByClassName("active");
    standart[0].className = standart[0].className.replace(" active", "");
    this.className += " active";
  });
}
//ЗНАХОДИМО НАШУ НАВІГАЦІЮ, І ЗМІНЮЄМО СТАН ACTIVE ПРИ НАТИСНЕННІ
```

Рисунок 6.2.3.2 – Додавання динамічності навігації

Далі нам потрібно зробити нашу навігацію динамічною, щоб вона завжди була із нами. Для початку викликаємо функцію, оголошуємо наші змінні які будуть відповідати за навігацію, за поточку позицію користувача на екрані, та висоту навігації, і при події «scroll», змінні будуть оновлюватися і порівнюватись між собою, так коли поточна позиція буде більша за висоту навігації, то навігація буде фіксуватися, а якщо навпаки, то клас буде видалятися із навігації, і вона буде повертатися до поточного вигляду.

```
$(function() {

  let header = $("#navigation");
  let nav = $("#nav");
  let navH = nav.innerHeight();
  let scrollPos = $(window).scrollTop();
  let navToggle = $("#navToggle");

  checkScroll(scrollPos, navH);

  $(window).on("scroll  resize", function(){

      navH = nav.innerHeight();
      scrollPos = $(this).scrollTop();

      checkScroll(scrollPos, navH);

  });
  function checkScroll(scrollPos, navH){
      if( scrollPos > navH ){
          header.addClass("fixed");
      } else{
          header.removeClass("fixed");
      }

  }
```

Рисунок 6.2.3.3 – Додавання динамічності навігації

Тепер до дамо код, який буде відповідати за плавну анімацію до наших елементів. Після кліку на навігацію визначається ідентифікатор цільового елемента та відбувається плавна анімація прокрутки. Також він видаляє клас "show" з елементу навігації, щоб відмінити попереднє форматування навігації. Код представлений на рисунку 6.2.3.4.


Рисунок 6.2.3.4 – Анімація прокрутки сторінки

Прийшов час додати функціоналу нашій формі, щоб була змога відправки повідомлень на пошту, для цього встановлюємо обробник подій для віправки форми, зберігаємо посилання на форму в зміну «var th = $(this);», викликаємо AJAX запит, вказуємо тип запиту POST, адресу на яку відправляється запит, це наш файл mail.php, в якому продовжимо відправку, та за допомогою «data: th.serialize()» обираємо елементи форми які будуть відправлятися. Тепер ще додаємо умову, що коли наш запит буде виконано, ставиться затримка на 1 секунду, та скидаються всі значення полів у формі завдяки «th.trigger("reset");» , які заповнював користувач. Код представлений на рисунку 6.2.3.5.

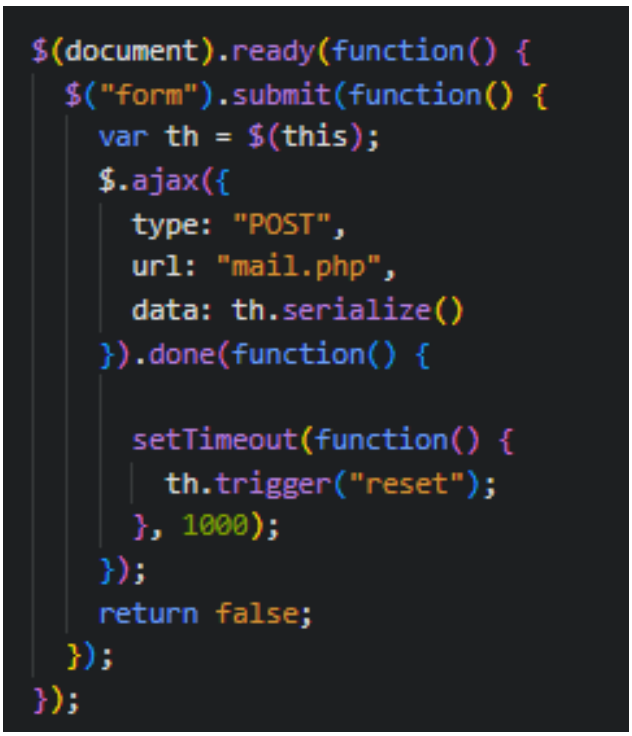



Рисунок 6.2.3.5 – Відправка даних форми

6.2.4 Відправа форми за допомогою PHP

Для цього був створений файл mail.php, в якому ми підключаємо бібліотеку PHPMailer та створюємо два об’єкти даної бібліотеки «$mail = new PHPMailer;» «$mail->CharSet = 'utf-8';». Тепер описуємо отримання даних із POST-запиту, а саме ['user_name'], ['user_phone'], ['user_email'], ['description']. Після чого налаштовуємо відправлення через SMTP, вказуючи наш хост, тобто gmail, логін нашої пошти із якої буде відправка, та секретний пароль, підключаємо ssl і вказуємо порт 465. Встановлюємо адреси відправника та отримувача, та формат листа це буде HTML. Оформлюємо тіло листа, встановлюючи тему, та додаємо наші елементи із форми. Тепер потрібно додати умову, що якщо наш лист через якісь причини не буде відправлений то буде виводитися помилка, а якщо буде успішне відправлення то перехід на іншу сторінку. Код представлений на рисунку 6.2.4.1

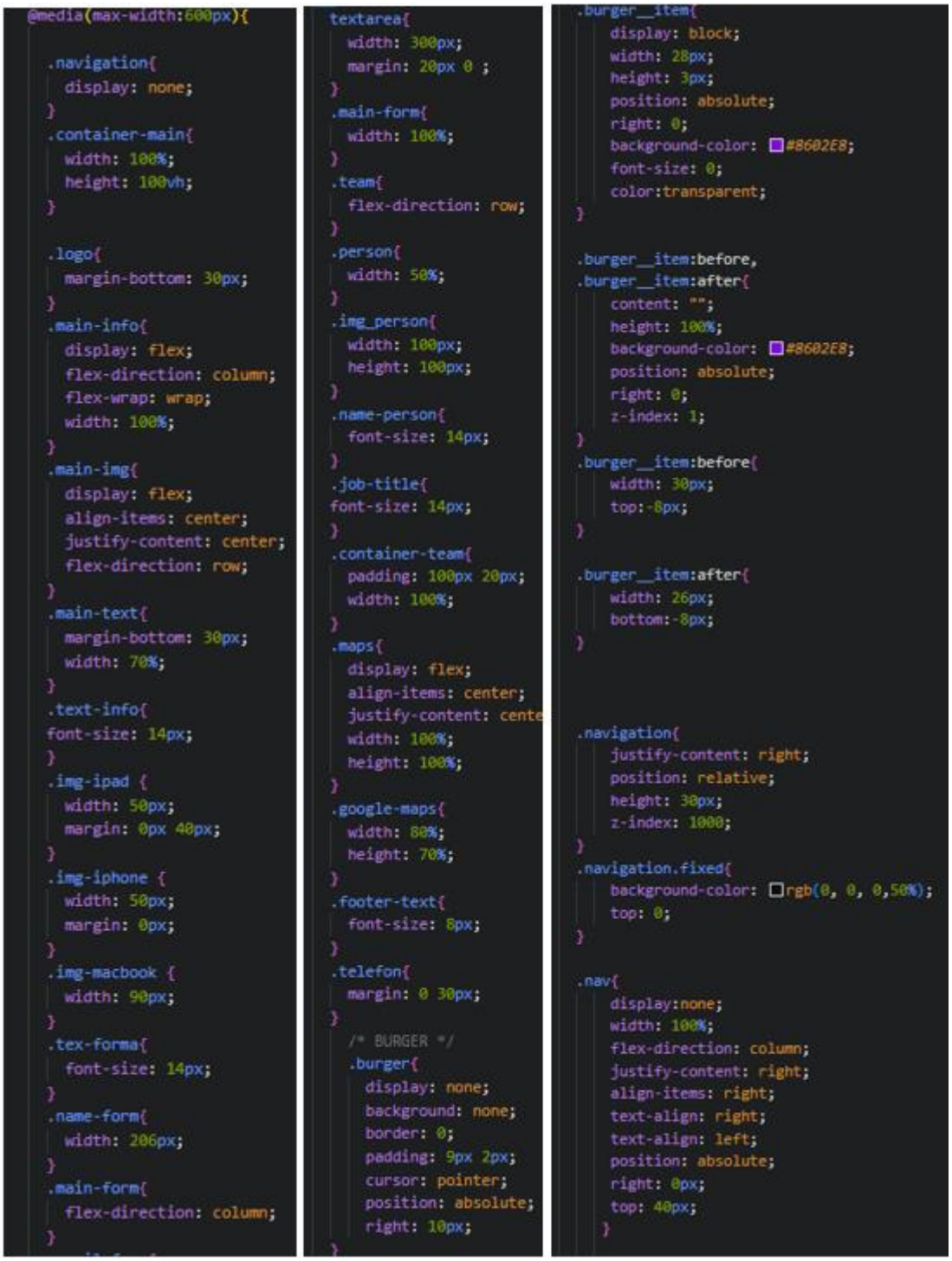
```php
<?php

require_once('phpmailer/PHPMailerAutoload.php');
$mail = new PHPMailer;
$mail->CharSet = 'utf-8';

$name = $_POST['user_name'];
$phone = $_POST['user_phone'];
$email = $_POST['user_email'];
$description = $_POST['description'];


$mail->isSMTP();
$mail->Host = 'smtp.gmail.com';
$mail->SMTPAuth = true;
$mail->Username = 'yuraandrushenk@gmail.com'; // Логін почти із якої буде ві
$mail->Password = 'xvitkprrjhayqsjp'; // Секретний код від вошої пошти
$mail->SMTPSecure = 'ssl';                            // Вмикаємо TLS encryp
$mail->Port = 465;

$mail->setFrom('yuraandrushenk@gmail.com');
$mail->addAddress('yuraandrushenk@gmail.com');
$mail->isHTML(true);

$mail->Subject = 'Заявка із AppleServiceTeam ';
$mail->Body    = '' .$name . ' залишив заявку, ось його телефон ' .$phone. '
$mail->AltBody = '';

if(!$mail->send()) {
    echo 'Error';
} else {
    header('location: thank-you.html ');
}

?>
```

Рисунок 6.2.4.1 – Відправка даних на пошту

### 6.2.5 Налаштування відображення сайту на телефонах

Так як веб-сайти мають велику популярність, потрібно оптимізувати їх роботу на інших девайсах, таких як телефони. Для цього ми застосуємо @media - це атрибут, який дозволяє визначити стилі, що застосовуються до елементів сторінки в залежності від різних умов екрану чи пристрою, на якому відображається сторінка. Тому ми створимо правило, якщо екран відповідає розмірам телефона, то будуть застосовуватися інші стилі оформлення, буде

змінюватися позиція елементів, їх розміри, кольори, розміри шрифту, та додання іншої навігації для телефону, для коректного відображення. Код представлений на рисунку 6.2.5.1.



```
@media(max-width:600px){

  .navigation{
    display: none;
  }
  .container-main{
    width: 100%;
    height: 100vh;
  }

  .logo{
    margin-bottom: 30px;
  }
  .main-info{
    display: flex;
    flex-direction: column;
    flex-wrap: wrap;
    width: 100%;
  }
  .main-img{
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: row;
  }
  .main-text{
    margin-bottom: 30px;
    width: 70%;
  }
  .text-info{
  font-size: 14px;
  }
  .img-ipad {
    width: 50px;
    margin: 0px 40px;
  }
  .img-iphone {
    width: 50px;
    margin: 0px;
  }
  .img-macbook {
    width: 90px;
  }
  .tex-forma{
    font-size: 14px;
  }
  .name-form{
    width: 206px;
  }
  .main-form{
    flex-direction: column;
  }
```

```
  textarea{
    width: 300px;
    margin: 20px 0 ;
  }
  .main-form{
    width: 100%;
  }
  .team{
    flex-direction: row;
  }
  .person{
    width: 50%;
  }
  .img_person{
    width: 100px;
    height: 100px;
  }
  .name-person{
    font-size: 14px;
  }
  .job-title{
  font-size: 14px;
  }
  .container-team{
    padding: 100px 20px;
    width: 100%;
  }
  .maps{
    display: flex;
    align-items: center;
    justify-content: cente
    width: 100%;
    height: 100%;
  }
  .google-maps{
    width: 80%;
    height: 70%;
  }
  .footer-text{
    font-size: 8px;
  }
  .telefon{
    margin: 0 30px;
  }
    /* BURGER */
    .burger{
      display: none;
      background: none;
      border: 0;
      padding: 9px 2px;
      cursor: pointer;
      position: absolute;
      right: 10px;
  }
```

```
  .burger__item{
      display: block;
      width: 28px;
      height: 3px;
      position: absolute;
      right: 0;
      background-color: #8602E8;
      font-size: 0;
      color:transparent;
  }

  .burger__item:before,
  .burger__item:after{
      content: "";
      height: 100%;
      background-color: #8602E8;
      position: absolute;
      right: 0;
      z-index: 1;
  }
  .burger__item:before{
      width: 30px;
      top:-8px;
  }

  .burger__item:after{
      width: 26px;
      bottom:-8px;
  }



  .navigation{
      justify-content: right;
      position: relative;
      height: 30px;
      z-index: 1000;
  }
  .navigation.fixed{
      background-color: rgb(0, 0, 0,50%);
      top: 0;
  }

  .nav{
      display:none;
      width: 100%;
      flex-direction: column;
      justify-content: right;
      align-items: right;
      text-align: right;
      text-align: left;
      position: absolute;
      right: 0px;
      top: 40px;
    }
```

Рисунок 6.2.5.1 – Оптимізація веб-сайту на телефонах

Демонстрація вигляду проєкту на телефонах, представлена на рисунках 6.2.5.2-6.2.5.3.

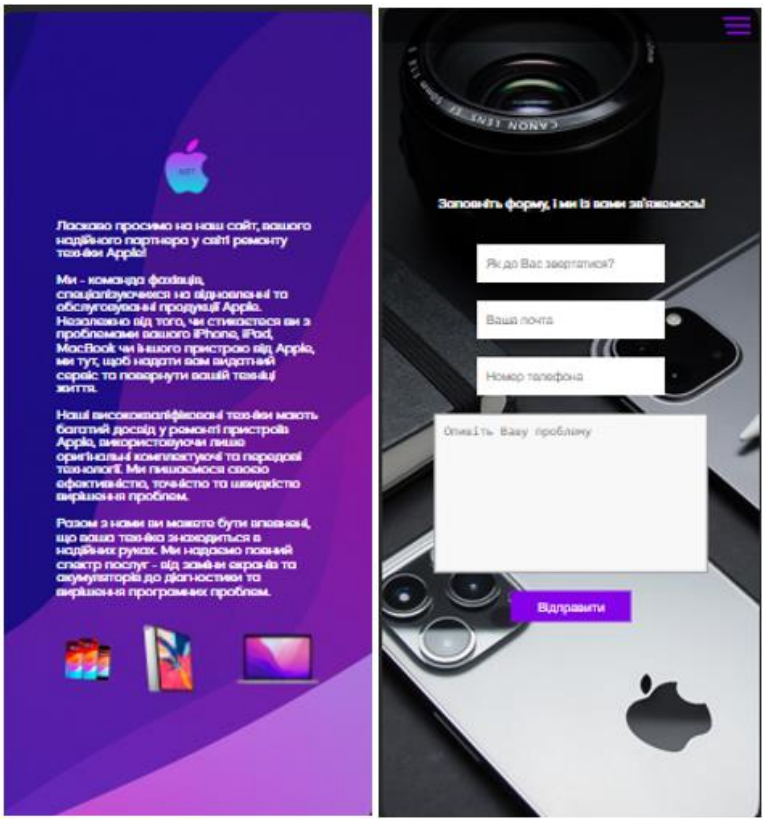



Рисунок 6.2.5.2 Вигляд веб-сайту на телефонах

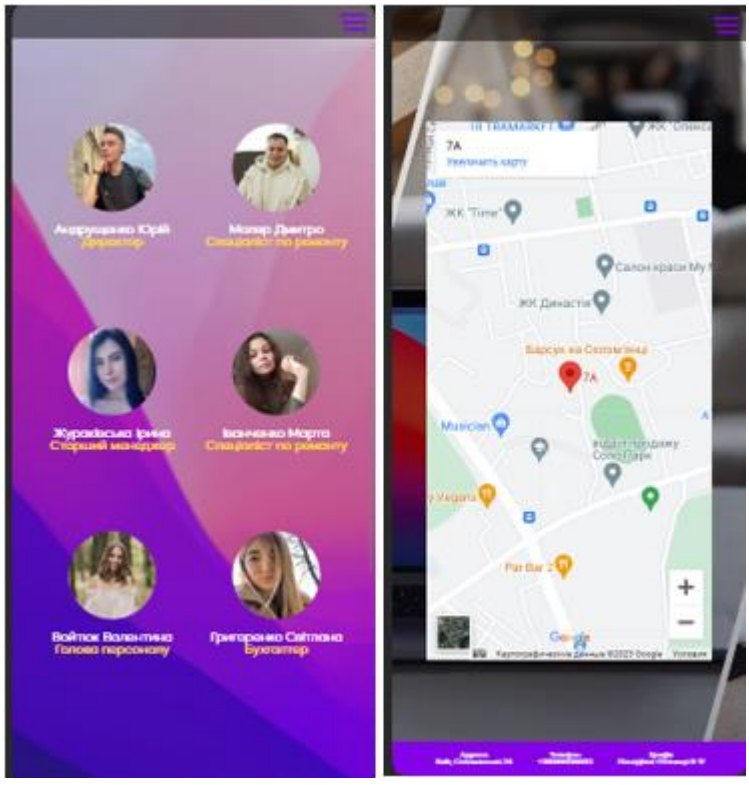



Рисунок 6.2.5.3 Вигляд веб-сайту на телефонах

6.2.6 Виставлення на хостинг

Хостинг є сервісом, який надає послуги компанією, яка дозволяє розміщувати ваш веб-сайт або додаток в Інтернеті. Придбавши пакет хостингу, ви отримуєте можливість користуватися частиною сервера або віртуальним сервером, де можна зберігати ваші файли, бази даних та інші ресурси, необхідні для оптимальної роботи вашого веб-сайту чи додатку.

Був використаний хостинг «www.zzz.com.ua/uk», для початку нам потрібно було зареєструватися:

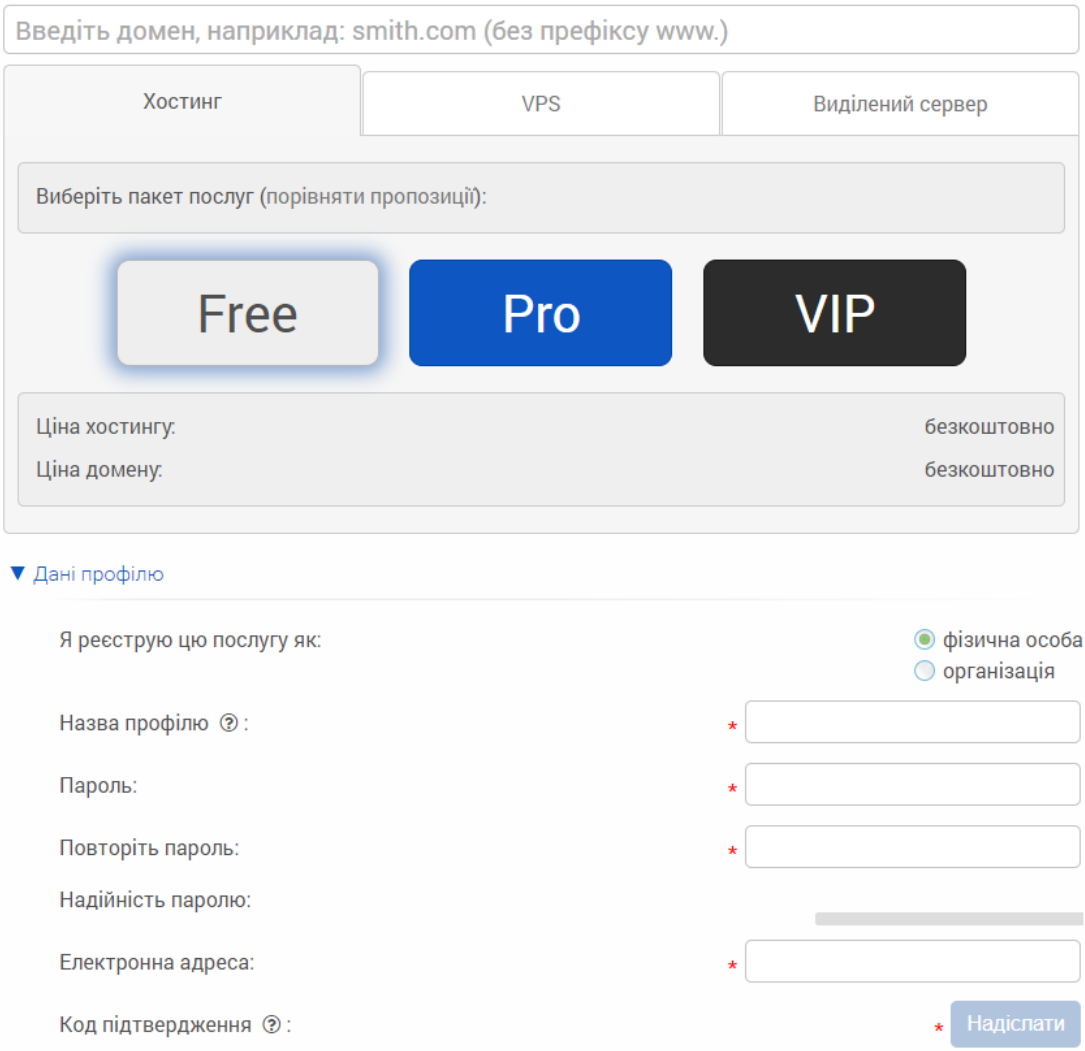



Рисунок 6.2.6.1 Реєстрування на хостингу

Після чого, нам потрібно створити FTP акаунт :

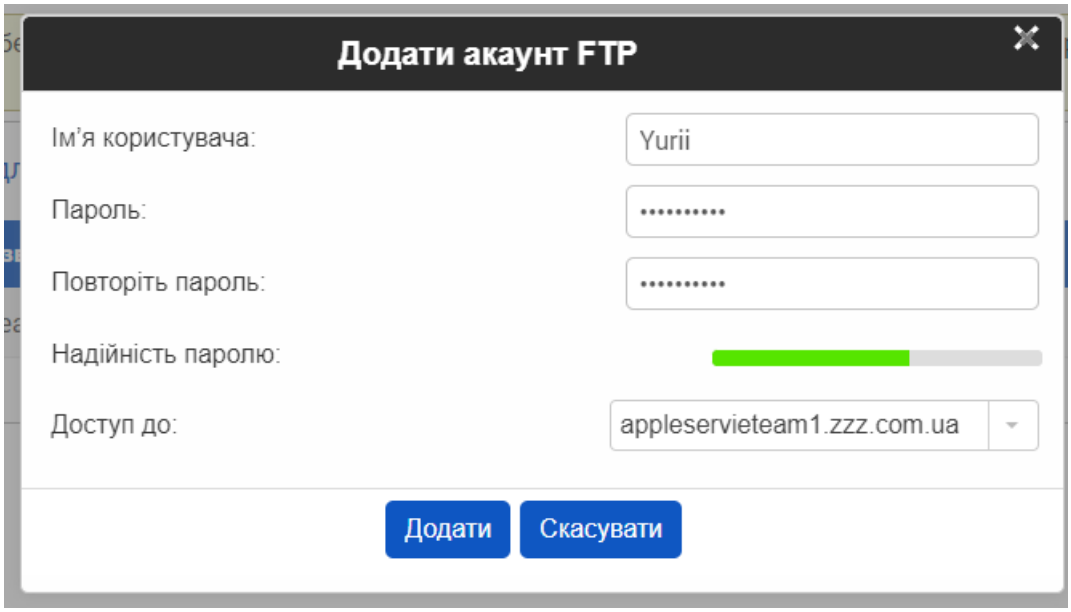



Рисунок 6.2.6.2 Створення FTP акаунта

Далі ми виконуємо вхід в наш створений FTP акаунт, і завантажуємо всі необхідні файли для робити веб-сайту:

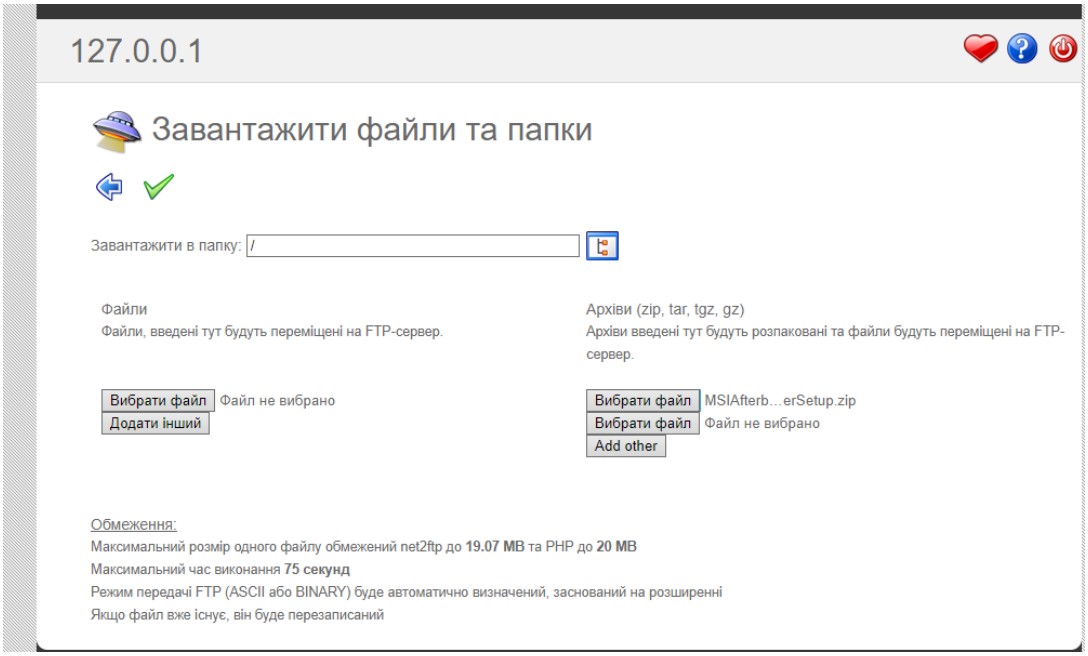



Рисунок 6.2.6.3 Завантаження всіх необхідних файлів

Демонстрація роботи веб-сайту на створеному домені appleservieteam1.zzz.com.ua, представлена на рисунку 6.2.6.4.

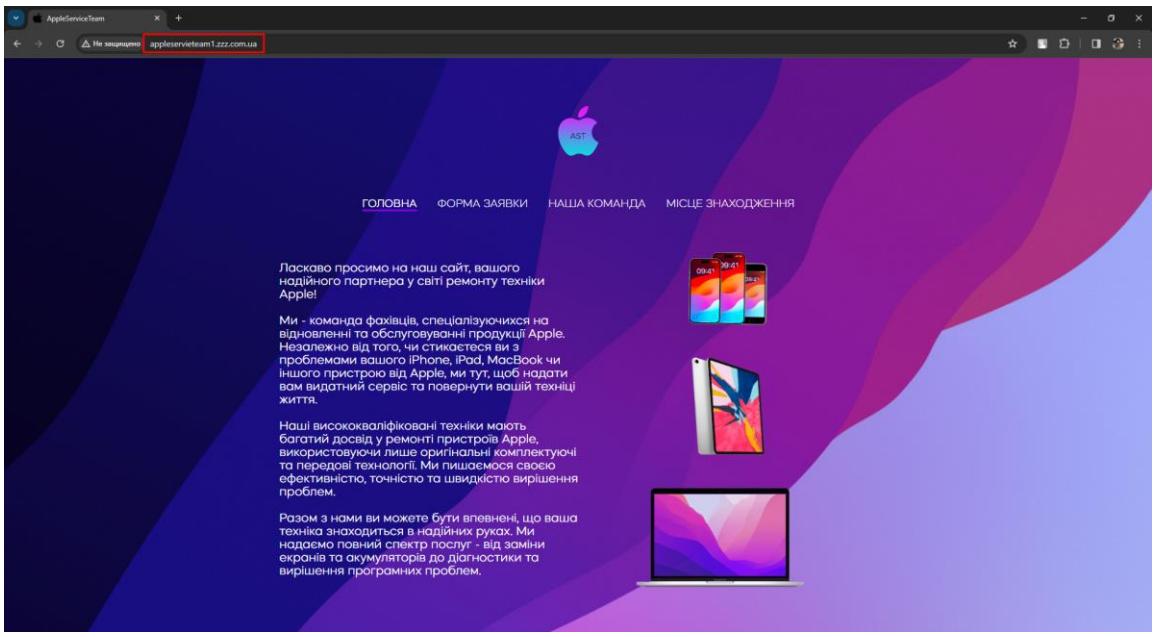



Рисунок 6.2.6.4 Робота сайту на створеному домені

# ВИСНОВОК

Процес розвитку мов програмування для веб-розробки відкриває безліч можливостей для створення різноманітних веб-сайтів. Це можуть бути сайти для торгівлі, особистих портфоліо, зберігання інформації, інформаційні платформи або розважальні ресурси.

Однією з ключових мов для веб-розробки є HTML, яка відповідає за створення основної структури документа. Її використання дозволяє розбивати контент на блоки та ефективно розміщувати елементи на сторінці. CSS відіграє важливу роль у стилізації контенту, надаючи сторінці візуальну привабливість та зручність для користувачів. Мови програмування JS та JQuery надають можливість динамічно змінювати поведінку об'єктів на веб-сторінці, роблячи її більш інтерактивною та унікальною.

У процесі вивчення мов програмування були закріплені базові навички, вивчені різноманітні редактори та програми для дизайну. Результатом цього дослідження став створений повноцінний веб-сайт з гнучким та адаптивним меню, різноманітними стилями шрифтів, інтерактивною мапою місцезнаходження, інформаційним блоком, списком учасників та контактними даними. Сайт також має адаптивний дизайн для коректного відображення на різних пристроях. Та форму для клієнтів, завдяки якій можна отримувати замовлення та виконувати їх, що допомагає зберігати зворотній зв’язок із клієнтами.

Створений веб-сайт є інформаційною платформою з можливістю подальшого розширення та додавання різноманітних функціональностей та стилів оформлення. Він був успішно завантажений на хостинг для загального доступу за визначеним доменом.

## ПЕРЕЛІК ПОСИЛАНЬ


1. Книга, Ніксон Робін Чуйний дизайн на HTML5 та CSS3 для будь-яких пристроїв. 3-те вид. – 250с.
2. Створюємо динамічні веб-сайти за допомогою PHP, MySQL, JavaScript, CSS та HTML5. 5-те видання. – 320с.
3. Книга, Ніксон Робін Чуйний дизайн на HTML5 та CSS3 для будь-яких пристроїв. 3-те вид. – 250с.
4. HTML5 та CSS3. Веб-розробка за стандартами нового покоління. 2-ге вид. – 166с.
5. Книга, Елізабет Фрімен та Ерік Фріман – 365с.

# ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)


**Об’єкт дослідження** - процес проектування, та створення інформаційної системи для роботи сервісного центру під виглядом веб-сторінки.

**Предмет дослідження** – інформаційна система для роботи сервісного центру.

**Мета роботи** - проектування та розробка веб-сайту для сервісного центру.

**Завдання:**

1. Визначення ключових функцій та структури системи.
2. Створення макету системи для візуалізації її структури та компонентів.
3. Впровадження HTML-коду для створення основної структури системи.
4. Розробка заданих позицій та візуального оформлення основного контенту за допомогою CSS .
5. Розробка PHP-скрипта для обробки даних, отриманих з форми.
6. Використання JavaScript для динамічного додавання та керування класами елементів.
7. Виведення створеної системи на хостинг для доступу та використання через Інтернет.

Рис. 3.1 – Макет головної сторінки, і сторінки форми

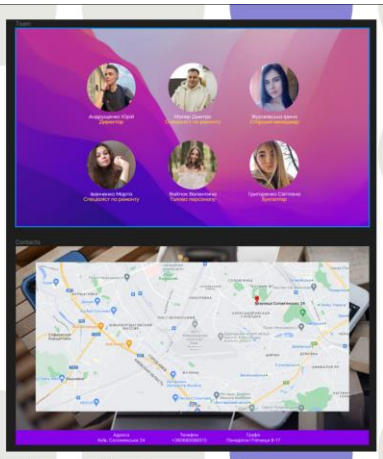


Рис. 3.2 – Макет сторінки із командою, і контактною інформацією

Рис. 3.3 – Макет лоадера

Рис. 4.1 – Підключення основних файлів, та опис перших елементів структури

Рис. 4.2 – Опис структури навігації

Рис. 4.3 – Опис структури сторінки із формаю

Рис. 4.3 – Опис структури сторінки із контактною інформацією

```
/* Підключення шрифтів */
@font-face {
  font-family: 'Mazzard Soft H';
  src: url('Fonts/MazzardSoftH-Thin.otf');
  font-weight: 100;
  font-style: normal;
 }

@font-face {
  font-family: 'Mazzard Soft H';
  src: url('Fonts//MazzardSoftH-ExtraLight.otf');
  font-weight: 200;
  font-style: normal;
 }

@font-face {
  font-family: 'Mazzard Soft H';
  src: url('Fonts/MazzardSoftH-Light.otf');
  font-weight: 300;
  font-style: normal;
 }

@font-face {
  font-family: 'Mazzard Soft H';
  src: url('Fonts/MazzardSoftH-Regular.otf');
  font-weight: 400;
  font-style: normal;
 }

@font-face {
  font-family: 'Mazzard Soft H';
  src: url('Fonts/MazzardSoftH-Medium.otf');
  font-weight: 500;
  font-style: normal;
 }

@font-face {
  font-family: 'Mazzard Soft H';
  src: url('Fonts/MazzardSoftH-SemiBold.otf');
  font-weight: 600;
  font-style: normal;
 }
```

Рис. 5.1 – Підключення шрифтів

Відео 5.1 – Лоадер

```
/* Вимикаємо скролбар */
::-webkit-scrollbar {
  width: 0px;
  background: transparent;
}
/* Вимикаємо скролбар */

/* Забираємо відступи у body */
 body{
     margin: 0;
     padding: 0;
     font-family: 'Mazzard Soft H';
 }
/* Забираємо відступи у body */
```

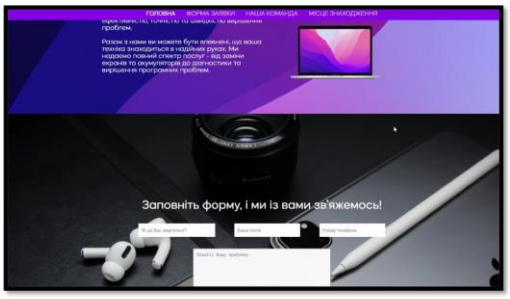
Рис. 5.1 – Відключення скролбару

```
/* Забираємо відступи у заголовків і абзаців */
h1,h2,h3,h4,h5,h6, p
{
    padding: 0;
    margin: 0;
}
/* Забираємо відступи у заголовків і абзаців */

/*Нормальний розрахунок ширини і висоти елементів */
  *,
  *:before,
  *:after{
      box-sizing: border-box;
  }
/*Нормальний розрахунок ширини і висоти елементів */
```

Рис. 5.1 – Оформлення заголовків

```
/* ЛОАДЕР ПОЧАТОК */
/* ЛОАДЕР, задаємо параметри сторінці лоадера */
.mask {
  display: flex;
  justify-content: center;
  align-items: center;
  position: absolute;
  width: 100%;
  height: 100%;
  /* background-image: url('img/background/mask-background.jpg'); */
  background-color: white;
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  top: 0;
  z-index: 6;

 }
/* ЛОАДЕР, задаємо параметри сторінці лоадера */

/* Вирівнюємо внутрішні елементи лоадера */
 .loader {
   display: flex;
   justify-content: center;
   align-items: center;
   flex-direction: column;

 }
/* Вирівнюємо внутрішні елементи лоадера */

/* Задаємо параметри картинці лоадера, і анімації його */
 .imgLoader {
   width: 125px;
   -webkit-animation: scale 2s linear infinite; /* Safari */
   animation: scale 2s linear infinite;
 }
/* Задаємо параметри картинці лоадера, і анімації його */

/* Анімація картинки */
 @-webkit-keyframes scale {
   0% { -webkit-transform:scale(1); }
   25% { -webkit-transform:scale(0.75); }
   50% { -webkit-transform:scale(0.60); }
   75% { -webkit-transform: scale(0.75);}
   100% { -webkit-transform: scale(1); }
}
/* Анімація картинки */
```

Рис. 5.1 – Оформлення лоадера




Відео 7.1 – Фіксація навігації

Відео 7.2 – Логіка навігації

```
$(function() {

  let header = $("#navigation");
  let nav = $("#nav");
  let navH = nav.innerHeight();
  let scrollPos = $(window).scrollTop();
  let navToggle = $("#navToggle");

  checkScroll(scrollPos, navH);

  $(window).on("scroll  resize", function(){

      navH = nav.innerHeight();
      scrollPos = $(this).scrollTop();

      checkScroll(scrollPos, navH);

  });
```

Рис. 7.1 – Додавання подій навігації

```
function checkScroll(scrollPos, navH){
    if( scrollPos > navH ){
        header.addClass("fixed");
    } else{
        header.removeClass("fixed");
    }

}
```

Рис. 7.2 – Додавання подій навігації

```
$("[data-scroll]").on("click", function(event) {
    event.preventDefault();

    let elementId = $(this).data('scroll');
    let elementOffset = $(elementId).offset().top;

    nav.removeClass("show");

    $("html, body").animate({
        scrollTop: elementOffset - 0
    }, 700);

});
```

Рис. 7.3 – Додавання подій навігації

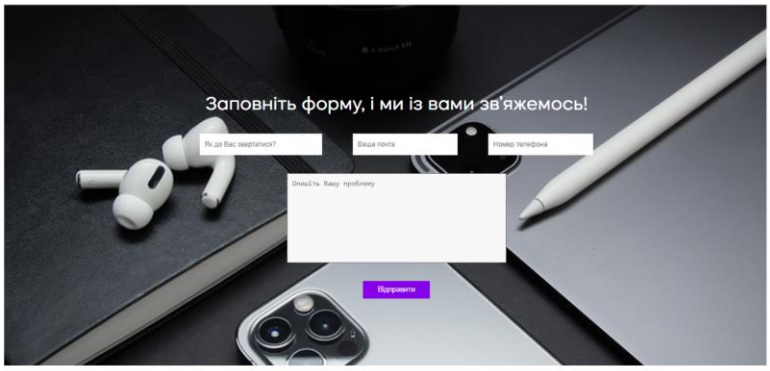



**Рис. 8.1 – Форма заповнення**

```
<!-- КОНТЕЙНЕР ФОРМА ПОЧАТОК -->
<div class="container-forma" id="form">
  <!-- ВНУТРІШНІЙ КОНТЕЙНЕР ФОРМА -->
  <div class="forma"  data-aos="fade-up"  data-aos-delay="50">
    <!-- ЗАГОЛОВОК ФОРМИ -->
    <div class="tex-forma">Заповніть форму, і ми із вами зв'яжемось!</div>
    <!-- ЗАГОЛОВОК ФОРМИ -->
    <!-- <div class="main-form"> -->
      <!-- ОСНОВНА ФОРМА -->
      <form action="mail.php" method="POST" class="main-form">

        <input type="text" class="name-form" name="user_name" placeholder="Як до Вас звертатися?" required>
        <input type="text" class="email-form" name="user_email" placeholder="Ваша почта" required>
        <input type="text" class="phone-form" name="user_phone" placeholder="Номер телефона" required>

      <div class="textarea-button">
        <textarea class="textarea"  name="description" placeholder="Опишіть Вашу проблему"></textarea>
        <input type="submit" onclick="document.location='thank-you.html'" class="button-form" value="Відправити">
      </div>

      </form>
      <!-- ОСНОВНА ФОРМА -->
    </div>
    <!-- ВНУТРІШНІЙ КОНТЕЙНЕР ФОРМА -->
</div>
<!-- КОНТЕЙНЕР ФОРМА КІНЕЦЬ -->
```

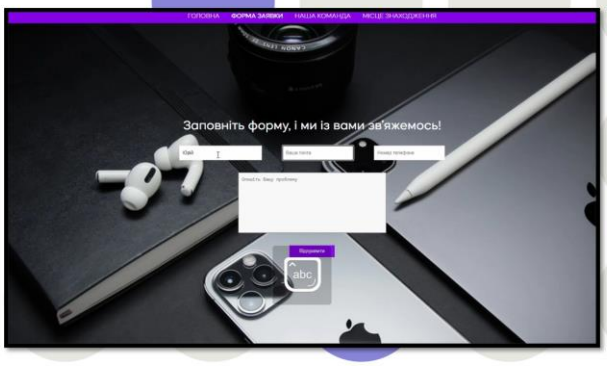
**Рис. 8.2 – Структура форми**

```
$(document).ready(function() {
  $("form").submit(function() {
    var th = $(this);
    $.ajax({
      type: "POST",
      url: "mail.php",
      data: th.serialize()
    }).done(function() {
      alert("Thank you!");
      setTimeout(function() {

        th.trigger("reset");
      }, 1000);
    });
    return false;
  });
});
```

**Рис. 8.2 – Логіка форми**

```
require_once('phpmailer/PHPMailerAutoload.php');
$mail = new PHPMailer;
$mail->CharSet = 'utf-8';

$name = $_POST['user_name'];
$phone = $_POST['user_phone'];
$email = $_POST['user_email'];
$description = $_POST['description'];


$mail->isSMTP();
$mail->Host = 'smtp.gmail.com';
$mail->SMTPAuth = true;
$mail->Username = 'yuraandrushenk@gmail.com'; // Логін почти із якої буде відправлятися лист
$mail->Password = 'xvitkprrjhayqsjp'; // Секретний код від вашої пошти
$mail->SMTPSecure = 'ssl';                            // Вмикаємо TLS encryption
$mail->Port = 465;

$mail->setFrom('yuraandrushenk@gmail.com');
$mail->addAddress('yuraandrushenk@gmail.com');
$mail->isHTML(true);

$mail->Subject = 'Заявка із AppleServiceTeam ';
$mail->Body    = '' .$name . ' залишив заявку, ось його телефон ' .$phone. '<br>Пошта цього користувача:
$mail->AltBody = '';

if(!$mail->send()) {
    echo 'Error';
} else {
    header('location: thank-you.html ');
}
?>
```

**Рис. 9.1 – Логіка роботи SMTP**




**Відео 9.1 – Робота SMTP**

```
@media(max-width:600px){
  .navigation{
    display: none;
  }
  .container-main{
    width: 100%;
    height: 100vh;
  }
  .logo{
    margin-bottom: 30px;
  }
  .main-info{
    display: flex;
    flex-direction: column;
    flex-wrap: wrap;
    width: 100%;
  }
  .main-img{
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: row;
  }
  .main-text{
    margin-bottom: 30px;
    width: 70%;
  }
  .text-info{

  font-size: 14px;
  }
  .img-ipad {
    width: 50px;
    margin: 0px 40px;
  }
  .img-iphone {
    width: 50px;
    margin: 0px;
  }
  .img-macbook {
    width: 90px;
  }
```

```
  .tex-forma{
    font-size: 14px;
  }
  .name-form{
    width: 206px;
  }
  .main-form{
    flex-direction: colum
  }
  .email-form{
    margin: 20px 0;
  }
  textarea{
    width: 300px;
    margin: 20px 0 ;
  }
  .main-form{
    width: 100%;
  }
  .team{
    flex-direction: row;
  }
  .person{
    width: 50%;
  }
  .img_person{
    width: 100px;
    height: 100px;
  }
  .name-person{
    font-size: 14px;
  }
  .job-title{
  font-size: 14px;
  }
  .container-team{
    padding: 100px 20px;
    width: 100%;
  }
}
```

```
.nav{
    display:none;
    width: 100%;
    flex-direction: column;
    justify-content: right;
    align-items: right;
    text-align: right;

    text-align: left;

    position: absolute;
    right: 0px;
    top: 40px;
  }
  .links{
    display: flex;
      width: 100%;
      flex-direction: column;

      align-items: flex-end;
  }
.active{
  border-bottom: 0px;
}
.nav.show{
    display: flex;
    justify-content: right;
    background-color: rgb(0,


  }

.burger{
    display: block;
    padding-right: 20px;
}
.a__links{
  text-align: right;
    font-size: 14px;
    margin-right: 0;
    margin: 3px ;
    color: #fff;
    padding-right: 10PX;
}
```

```
.nav{
    display:none;
    width: 100%;
    flex-direction: column;
    justify-content: right;
    align-items: right;
    text-align: right;

    text-align: left;

    position: absolute;
    right: 0px;
    top: 40px;
  }
  .links{
    display: flex;
      width: 100%;
      flex-direction: column;

      align-items: flex-end;
  }
.active{
  border-bottom: 0px;
}
.nav.show{
    display: flex;
    justify-content: right;
    background-color: rgb(0, 0, 0,50%);


  }

.burger{
    display: block;
    padding-right: 20px;
}
.a__links{
  text-align: right;
    font-size: 14px;
    margin-right: 0;
    margin: 3px ;
    color: #fff;
    padding-right: 10PX;
}
```

```
s{
splay: flex;
ign-items: center;
stify-content: center;
dth: 100%;
ight: 100%;

gle-maps{
dth: 80%;
ight: 70%;



nter-text{
nt-size: 8px;


efon{
rgin: 0 30px;


  BURGER */


urger{
display: none;
background: none;
border: 0;
padding: 9px 2px;
cursor: pointer;
position: absolute;
right: 10px;


ger__item{
display: block;
width: 28px;
height: 3px;
position: absolute;
right: 0;
background-color: #8602E8;

font-size: 0;
color:transparent;
```

**Рис. 10.1 – Адаптація веб-сайту для телефонів**


**Відео 9.1 – Робота адаптації**

# ВИСНОВКИ

Було створено інформаційну систем для роботи сервісного центру. Сайт оснащений гнучким та адаптивним меню, різними стилями оформлення шрифтів, інтегрованою картою для відображення місцезнаходження організації, головною сторінкою, формою для введення даних, списком учасників та контактним блоком. Крім того, сайт має адаптивний дизайн, що забезпечує коректне відображення на різних пристроях, незалежно від їх розміру екрану. Також була додана можливість заповнення форми клієнтами та подальше її відправлення. Завершивши розробку, сайт був розміщений на хостингу і став доступним для клієнтів за відповідним доменом.

# ДЯКУЮ ЗА УВАГУ!

## Апробація результатів дослідження:

Андрущенко Ю.М «МІСЦЕ ІНТЕРНЕТ РЕЧЕЙ В ПОБУТІ»

Тези доповіді на Всеукраїнську науково-технічну конференцію “Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і Світу” 28 листопада 2023 року.