МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт
з дисципліни **«СТВОРЕННЯ WEB-САЙТІВ»**

здобувачів освітнього ступеня «бакалавр»
за спеціальністю 121 – Інженерія програмного забезпечення,
освітня програма «Інженерія програмного забезпечення»
галузі знань 12 – Інформаційні технології
усіх форм навчання

Черкаси 2024

Затверджено вченою радою ФІТІС, протокол №____від _____2024 р., згідно з рішенням кафедри програмного забезпечення автоматизованих систем, протокол №____ від _____2024 р.

Упорядники Олексюк В.В., *к.т.н (PhD), ст. викладач*
Заспа Г.О., *к.т.н., доценнт*
Куницька С.Ю., *к.т.н., доцент*

Рецензент Лавданська О.В., *к.т.н., доцент*

Методичні рекомендації до виконання лабораторних робіт з дисципліни «СТВОРЕННЯ WEB-САЙТІВ» для здобувачів освітнього ступеня «бакалавр» за спеціальністю 121 – Інженерія програмного забезпечення, освітня програма «Інженерія програмного забезпечення» галузі знань 12 – Інформаційні технології усіх форм навчання [Електронний ресурс] / [упоряд. В.В. Олексюк, Г.О. Заспа, С.Ю. Куницька]; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2024. – 73 с. – Назва з титульного екрана.

Методичні рекомендації спрямовані на формування у здобувачів освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення» знань основ web-програмування, дизайну та розробки веб-сайтів, створення адаптивного та інтерактивного контенту завдяки HTML, CSS з використанням мови JavaScript, сучасних інтрументів та фреймворків.

Навчальне електронне видання
мережного використовування

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
до виконання лабораторних робіт
з дисципліни «**СТВОРЕННЯ WEB-САЙТІВ**»
здобувачів освітнього ступеня «бакалавр»
за спеціальністю 121 – Інженерія програмного забезпечення,
освітня програма «Інженерія програмного забезпечення»
галузі знань 12 – Інформаційні технології
усіх форм навчання

Упорядники: **Олексюк** Вадим Володимирович
Заспа Григорій Олександрович
Куницька Світлана Юріївна

В авторській редакції.

ЗМІСТ

ВСТУП

Методичні рекомендації до виконання лабораторних робіт з дисципліни «СТВОРЕННЯ WEB-САЙТІВ» розроблені з метою надання студентам інструментів та знань для практичного освоєння основних принципів і технологій веб-розробки. Дисципліна «СТВОРЕННЯ WEB-САЙТІВ» спрямована на формування у студентів компетентностей, необхідних для створення, редагування та підтримки веб-сайтів, використовуючи сучасні стандарти та технології.

Метою дисципліни є навчання студентів основам веб-програмування, дизайну та розробки веб-сайтів, включаючи створення адаптивного та інтерактивного контенту, а також публікацію веб-ресурсів в інтернеті. Протягом курсу студенти дізнаються про принципи структурної та семантичної розмітки HTML, стильове оформлення за допомогою CSS, базове використання JavaScript, а також про сучасні інструменти та фреймворки, які полегшують процес розробки.

Головні завдання лабораторних робіт включають:

– *Лабораторна робота №1* «Розмітка сторінки автобіографія» - навчити студентів основам HTML-розмітки та структуризації особистої інформації у вигляді веб-сторінки.

– *Лабораторна робота №2* «Розмітка price-таблиці digital-продукту» - ознайомити студентів з методами створення таблиць у HTML для представлення цінової інформації про цифрові продукти.

– *Лабораторна робота №3* «Створення лендінга» - сформувати навички створення односторінкових веб-сайтів (лендінгів), які ефективно представляють продукт або послугу.

– *Лабораторна робота №4* «Публікація сайту в інтернеті» - надати студентам знання та практичні навички щодо хостингу та доменів, а також методів публікації створених веб-сайтів в інтернеті.

– *Лабораторна робота №5* «Створення сайту порт фоліо» - навчити студентів розробці персонального портфоліо, що включає проекти, навички та досягнення, використовуючи HTML і CSS.

– *Лабораторна робота №6* «SASS. Препроцесори в CSS» - ознайомити студентів з можливостями CSS-препроцесорів, зокрема SASS, для більш ефективного та організованого написання стилів.

– *Лабораторна робота №7* «Адаптивна розмітка» - розвинути у студентів навички створення веб-сайтів з адаптивним дизайном, які правильно відображаються на різних пристроях і екранах.

Вище описані лабораторні роботи створені з метою надання студентам практичного досвіду, необхідного для професійного зростання у сфері веб-розробки.

Навчання студентів ведеться в формі лекцій, лабораторних та самостійних занять з використанням сучасних персональних комп'ютерів. Самостійні заняття зводяться до індивідуального вивчення додаткової літератури згідно навчального плану.

Лабораторна робота №1

Тема: «Розмітка сторінки автобіографія»

Мета: навчитись створювати прості веб-сторінки за допомогою мови HTML.

Звіт повинен містити:

1. Тему та мету роботи.
2. Постановку завдання.
3. Теоретичні відомості.
4. Результати виконання.
5. Висновок.

Завдання

1. Встановити редактор Visual Studio Code. Запустити, створити новий html-файл та скопіювати у нього текст HTML сторінки наведений нижче у завдані.
2. По бажанню скопіювати власне фото у папку з файлом, розкоментувавши блок <img> та вказати шлях до файлу замість посилання.
3. Відкрити html-файл за допомогою браузера.
4. Замінити всі [підказки] на відповідну інформацію про себе. Розділ «досвід роботи» може бути пропущений.
5. По бажанню встановити власний колір шрифту та фону сторінки за допомогою інструменту палітри кольору: https://www.w3schools.com/colors/colors_picker.asp

Теоретичні відомості

1. Історія та Призначення HTML

HTML (HyperText Markup Language) є основною мовою розмітки для створення веб-сторінок. Вона була створена Тімом Бернерсом-Лі в 1991 році. HTML використовується для створення структури та змісту веб-сторінок, визначення різних елементів, таких як заголовки, абзаци, списки, посилання, зображення і багато іншого.

Документи HTML є звичайними текстовими файлами, що містять спеціальні теги (або керуючі елементи що містяться в дужках <html>) розмітки. Теги розмітки вказують браузеру, як треба інтерпритувати сторінку.

Файли HTML зазвичай мають розширення .html (старіший варіант .htm). Їх можна створювати за допомогою будь-якого текстового редактора.

Є також велика кількість спеціалізованих редакторів для створення файлів HTML, таких як FrontPage, Macromedia Dreamweaver або Adobe Web Bundle, які володіють можливістю WYSIWYG (What You See Is What You Get – що бачиш, те й отримаєш). З їх допомогою можна легко створювати документи HTML, за допомогою кнопок та елементів меню, а не писати самому теги розмітки. Однак, тим, хто хоче стати технічно грамотним розробником, настійно рекомендується використовувати простий і водночас багатофункціональний текстовий редактор Visual Studio Code.

Окрім того корисним буде використання таких редакторів як: Notepad++, Atom, Sublime та ін.

2. Структура HTML-документа

HTML-документ має основну структуру, яка складається з тегів, атрибутів і текстового вмісту. Основні елементи включають тег <html>, який вказує початок та кінець HTML-документа, тег <head>, який містить мета-

інформацію про документ і тег <body>, який містить вміст сторінки, який буде відображений у браузері.

Вміст <head> частини не виводиться на екран користувача, за винятком заголовка, що позначається тегом <title>. Як правило в даному розділі за дпопомогою тегів <meta>, вказують ключові слова, авторів та іншу службову інформацію яку зчитують пошукові системи. Також в даному розділі підключають зовнішні таблиці стилів за допопогою тега <link> і JavaScript скрипти <script>.

У тегі <body> документа розміщують ту інформацію, яка буде виведена користувачеві.

3. Основні HTML-теги

HTML складається з багатьох тегів, кожен з яких має своє призначення. Деякі з основних тегів включають <h1>-<h6> для заголовків, <p> для абзаців тексту, <a> для посилань, <img> для вставки зображень, <ul> і <ol> для ненумерованих та нумерованих списків, відповідно.

4. Атрибути тегів

Атрибути тегів додають додаткову інформацію до елементів HTML. Наприклад, у тегу <a> атрибут *href* вказує URL-адресу, на яку буде переходити користувач при кліці на посилання, а у тегу <img> атрибут *src* вказує шлях до зображення.

5. Вкладеність тегів

HTML-теги можуть бути вкладені один в одного. Це означає, що один елемент може знаходитись всередині іншого. Наприклад, тег <p> може містити в собі текст або інші теги, такі як картинки <img> або гіперпосилання <a>.

Практична частина

1. Надрукуйте наступний текст:

```
<html>
  <head>
    <title>Це заголовок сторінки</title>
  </head>
  <body>
    <h1>Привіт!</h1>
    <p>Це моя перша сторінка HTML. <b>Цей текст виводиться жирним шрифтом.</b></p>
  </body>
</html>
```

2. Збережіть файл як index.html та відкрийте в браузері.

При збереженні файлу HTML можна використовувати розширення .htm, або .html. Розширення .htm було прийнято для старих версій операційних систем, які допускали трибуквенне розширення для файлів. В даний час практично всі операційні системи не мають подібного обмеження і можна використовувати розширення .html.

Ваш перший HTML-документ починається з тега <html>, який повідомляє браузеру про початок документа HTML і закінчується тегом </html>, який інформує браузер про досягнення кінця документа.

3. Текст між тегами <head> і </head> є інформацією заголовка документа. Ця інформація не виводиться у вікні браузера.

4. Текст «Це заголовок сторінки» між тегами <title> і </ title > є заголовком документа. Цей заголовок виводиться в рядку заголовка вікна браузера.

5. Текст між тегами <body> і </body> є текстом, який буде виведений у вікні браузера. Текст «**Привіт!**» між тегами <h1> і </h1> буде відображений стилем заголовка визначеним браузером, зазвичай жирним шрифтом більшого розміру.

6. Тег означає, що починається новий параграф, тег означає кінець параграфа.

Текст «**Цей текст виводиться жирним шрифтом.**» між тегами і буде виведений жирним шрифтом.

Текст HTML сторінки

```
<!DOCTYPE html>
<html lang=»en»>
<head>
    <meta charset=»UTF-8»>
       <meta   name=»viewport»   content=»width=device-width,   initial-
scale=1.0»>
    <title>Моя Автобіографія</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            background-color: #f4f4f4;
            color: #333;
        }
        .container {
            max-width: 800px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1, h2, h3 {
            color: #555;
```

```
        }
        p {
            line-height: 1.6;
        }
        img {
            width: 150px;
            height: auto;
            border-radius: 50%;
            margin-bottom: 20px;
        }
    </style>
</head>
<body>
    <div class=»container»>
        <header>
 <!-- <img src=»https://cdn.pixabay.com/photo/2016/08/08/09/17/avatar-1577909_1280.png» alt=»Ваше фото»> →
            <h1>Моя Автобіографія</h1>
                <p>Введіть ваші особисті дані та інформацію про вас тут.</p>
        </header>
        <section>
            <h2>Особисті дані</h2>
            <ul>
                <li><strong>Ім’я:</strong> [Ваше ім’я]</li>
                        <li><strong>Дата народження:</strong> [Дата народження]</li>
                    <li><strong>Місце проживання:</strong> [Місце проживання]</li>
                <li><strong>Електронна пошта:</strong> [Електронна пошта]</li>
                <li><strong>Телеграм:</strong> [Нікнейм]</li>
            </ul>
        </section>
        <section>
            <h2>Освіта</h2>
            <p>Опишіть вашу освіту тут.</p>
        </section>
        <section>
            <h2>Досвід роботи</h2>
```

```
            <p>Опишіть ваш досвід роботи тут.</p>
        </section>
        <section>
            <h2>Навички</h2>
            <p>Перерахуйте ваші навички тут.</p>
        </section>
        <section>
            <h2>Хобі та інтереси</h2>
            <p>Розкажіть про ваші хобі та інтереси тут.</p>
        </section>
        <footer>
            <p>© 2024 Моя Автобіографія</p>
        </footer>
    </div>
</body>
</html>
```

Наведемо приклад вигляду html сторінки в браузері на рисунку 1.

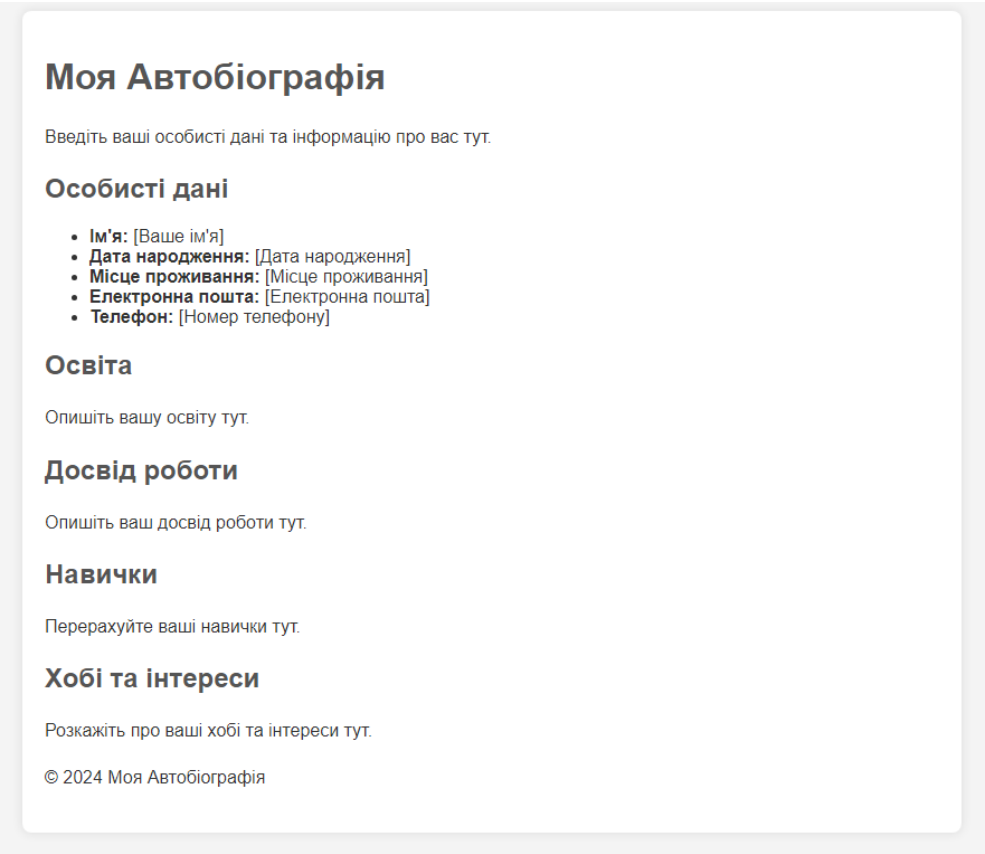

Рисунок 1 – Результат створення html-сторінкі у браузері

Лабораторна робота №2

Тема: «Розмітка price-таблиці digital-продукту»

Мета: отримати навички роботи з html-таблицями та CSS.

Звіт повинен містити:

1. Тему та мету роботи;
2. Постановку завдання;
3. Теоретичні відомості;
4. Результати виконання;
5. Висновок.

Завдання

1. Розробити структуру таблиці зображеної на рисунку 2 та заповнити її відповідними даними.

Choose your plan	starter $10 per month	econom $30 per month	standart $59 per month	professional $99 per month
Ampount of space	20GB	40GB	110GB	Unlimeted
Bandwidht per month	120GB	240GB	610GB	1300GB
Email accounts	1	15	60	Unlimeted
MySQL databases	1	15	60	Unlimeted
25h support	**No**	**Yes**	**Yes**	**Yes**
Suppport tickets per month	2	6	14	20
	Start	Start	Start	Start

Рисунок 2 – HTML Таблиця digital-продукту (вибір плану)

2. Створити в проєкті папку css в якій створити файл стилів style.css. Підключити його до html-сторінки за допомогою тегу <link>.

3. Задати стилі таблиці, відповідно вигляду (рисунок 2).

4. Використовувати відповідну кольорову схему (рисунок 2).

Теоретичні відомості

1. Основні теги для створення таблиць в HTML:

<table>: основний тег, який визначає початок і кінець таблиці;

<thead>: групує вміст заголовка таблиці;

<tbody>: групує основний вміст таблиці;

<tfoot>: групує вміст нижнього колонтитула таблиці;

<tr>: визначає рядок (**t**able **r**ow) таблиці;

<th>: визначення заголовкових (**t**able **h**eading) комірок у рядках таблиці;

<td>: визначає комірку (**t**able **d**ata) у рядках таблиці.

Таблиця 1 - Приклад опису таблиці

Header 1	**Header 2**	**Header 3**
Row 1 Data 1	Row 1 Data 2	Row 1 Data 3
Row 2 Data 1	Row 2 Data 2	Row 2 Data 3
Footer Content		

```
<table border="1">
        <thead>
            <tr>
                <th>Header 1</th>
                <th>Header 2</th>
                <th>Header 3</th>
            </tr>
        </thead>
```

```
        <tbody>
            <tr>
                <td>Row 1 Data 1</td>
                <td>Row 1 Data 2</td>
                <td>Row 1 Data 3</td>
            </tr>
            <tr>
                <td>Row 2 Data 1</td>
                <td>Row 2 Data 2</td>
                <td>Row 2 Data 3</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td colspan="3">Footer Content</td>
            </tr>
        </tfoot>
    </table>
```

Додатково можна використовувати різні атрибути для поліпшення вигляду таблиць, такі як border (задання рамки таблиці), colspan (об'єднання комірок по горизонталі), rowspan (об'єднання комірок по вертикалі) та ін.. Створення таблиць у HTML дозволяє вам ефективно представляти табличні дані на веб-сторінках, забезпечуючи чітку та організовану структуру для користувачів.

2. Каскадна таблиця стилів (Cascading Style Sheets, CSS)

CSS – це мова, яка використовується для опису зовнішнього вигляду веб-сторінок, наприклад, кольори, шрифти, розміри, розташування елементів та інше. У CSS визначаються правила стилю, які потім застосовуються до елементів HTML для того, щоб забезпечити конкретний дизайн та відображення вмісту веб-сторінки.

Основні цілі CSS:

– *розділення вмісту та представлення*: CSS дозволяє розділити веб-документ на дві частини: HTML, яка відповідає за вміст сторінки, і CSS, яка відповідає за стиль і візуальне оформлення сторінки. Це робить код більш структурованим і легким для редагування та підтримки.

– *консистентність та гнучкість*: CSS дозволяє встановлювати стиль для груп елементів, що робить його легко застосовувати до всієї сторінки в разі потреби. Він також дозволяє швидко змінювати вигляд всієї сторінки шляхом зміни лише одного файлу CSS.

– *підтримка адаптивного та відповідного дизайну*: CSS дозволяє створювати респонсивний дизайн, який адаптується до різних розмірів екранів та пристроїв, що полегшує веб-розробку для різних пристроїв.

Призначення CSS:

– визначення кольорів, шрифтів, розмірів, меж, відступів та інших властивостей елементів HTML;

– вирівнювання та позиціонування елементів на сторінці;

– визначення анімації, переходів та інших ефектів;

– створення адаптивного та відповідного дизайну для різних пристроїв;

– управління виглядом посилань, кнопок, форм та інших елементів інтерфейсу;

Загалом, CSS дозволяє веб-розробникам створювати привабливі, консистентні та адаптивні веб-сайти, забезпечуючи розділення вмісту та представлення та підвищуючи користувацький досвід.

3. Способи підключення CSS

Внутрішні стилі: безпосереднє задання стилю в атрибуті тега:

<h1 style="color: #cd66cc" **>Заголовок1</h1>**

Глобальні стилі, що описуються в розділі <head>

<style> h1 {color: #333366;} **</style>**

Пов'язані стилі, що імпортуються із зовнішнього файлу.

<link rel="stylesheet" href="style.css">

4. Базовий синтаксис CSS

Основним поняттям в CSS виступає **селектор** – це деяке ім'я стилю, для якого додаються параметри форматування. В якості селекторів можуть виступаюти теги, класи та ідентифікатори. *Примітка.* Кожен тег має наперед визначений стиль встановлений браузером, який CSS дає нам можливість перевизначати (рисунок 3).

Рисунок 3 - Загальний спосіб перевизначення стилів

Приведемо приклад опису стилів для селектора тега абзацу <p>:

```
p {
    text-align: justify; /*вирівнювання тесту по ширені сторінки*/
    color: #00FF00;      /*колір тексту зелений*/
    font-weight: 600;    /*визначає насиченість шрифту*/
    line-height: 1.2;    /*встановлює міжрядковий інтерава 1.2*/
    background: url("bg.png"); /*визначає фонове зображення*/
}
```

Кожен тег має наперед визначенні CSS властивості. Повертаючись до таблиць, ось кілька *основних CSS властивостей* для їх стилізації:

- *border-collapse*: дана властивість визначає, чи повинні бути злиті в одну подвійні рамки таблиці. Значення *collapse* об'єднує рамки, тоді як *separate* робить їх окремими;

- *border-spacing*: встановлює відстань між комірками таблиці при використанні значення *separate*;
- *text-align*: визначає вирівнювання тексту всередині комірок;
- *font-weight*: встановлює насиченість (товщину) шрифту для тексту в комірках таблиці;
- *background-color*: визначає колір фону комірок таблиці;
- *:hover* – псевдо-клас що встановлює стилі для елемента при наведенні миші;
- *overflow: hidden*; CSS-властивістю, яка визначає, як браузер повинен обробляти вміст, який виходить за межі області, встановленої для елемента.

Коли властивість overflow встановлена в hidden, будь-який зайвий контент, який не поміщається в задану область, буде обрізаний і прихований від виду.

Лабораторна робота №3

Тема: «Створення лендінга»

Мета: навчитись створювати простий сайт з використанням технологій HTML5 і CSS3 на основі графічного матеріалу макету.

Звіт повинен містити:

1. Тему та мету роботи;
2. Постановку завдання;
3. Теоретичні відомості;
4. Результати виконання;
5. Висновок.

Завдання

1. Розпакувати архів TaskTracker.zip у робочу папку з фамілією студента.
2. Відкритий файл index.html за допомогою браузера, має вигляд зображений на рисунку 4.

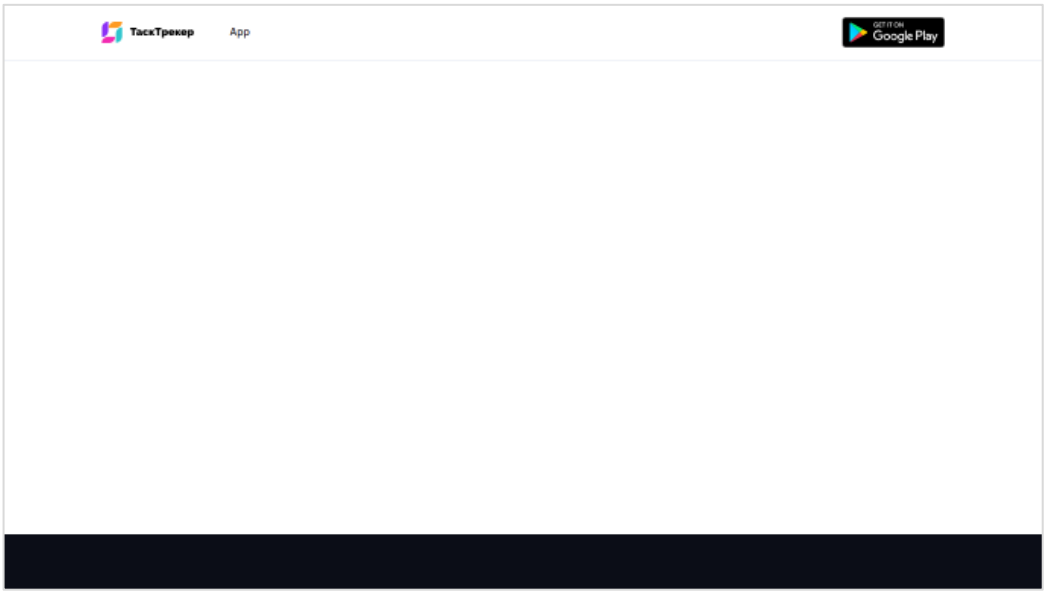

Рисунок 4 – Початковий вигляд лендінга

3. Відкрити файл index.html за допомогою редактора та заповнити його html – контентом (наведений нижче), додавши необхідні класи з файлу стилів. Графічний контент та файл стилів знаходяться у файлі site.zip:

```
<body>
    <header>
        <div>
            <div>
               <img width="50" src="./img/logo.png"> TaskTracker
            </div>

            <nav>
                <a href="#">App</a>
                <a href="#">Facilities</a>
                <a href="#">Price</a>
                <a href="#">Blog</a>
                <a href="#">Contacts</a>
            </nav>

            <div>
            <a href="https://play.google.com/"><img src=""></a>
            <a href="https://apps.apple.com/"><img src=""></a>
            </div>
        </div>
    </header>
    <main>
        <div>
            <div>
                <h1>Organiser app</h1>

                <div>
                    <p>Lorem ipsum dolor sit, amet consectetur
adipisicing elit. Obcaecati sequi cupiditate quo aut fuga
                        dolorem vitae suscipit libero quasi!</p>
                </div>
```

```
                <p>Lorem ipsum dolor sit amet consectetur,
adipisicing elit. Provident animi earum cupiditate quisquam,
                    veritatis?</p>

                <a href="#">Start using</a>
            </div>

            <div>
                <img src="" alt="">
            </div>
        </div>
    </main>

    <footer>
        <div>
            <div>
                <p>&copy;  2024  TaskTracker.  All  rights
reserved.</p>
            </div>

            <div>
     <a href="https://www.instagram.com/"><img src="./img/"></a>
     <a href="https://dribbble.com/"><img src=""></a>
     <a href="https://twitter.com/?lang=uk"><img src=" "></a>
     <a href="https://www.youtube.com/"><img src=" "></a>
            </div>
        </div>
    </footer>
</body>
```

4. Після оформлення, сторінка повинна мати вигляд, що зображено на рисунку 5 нижче.

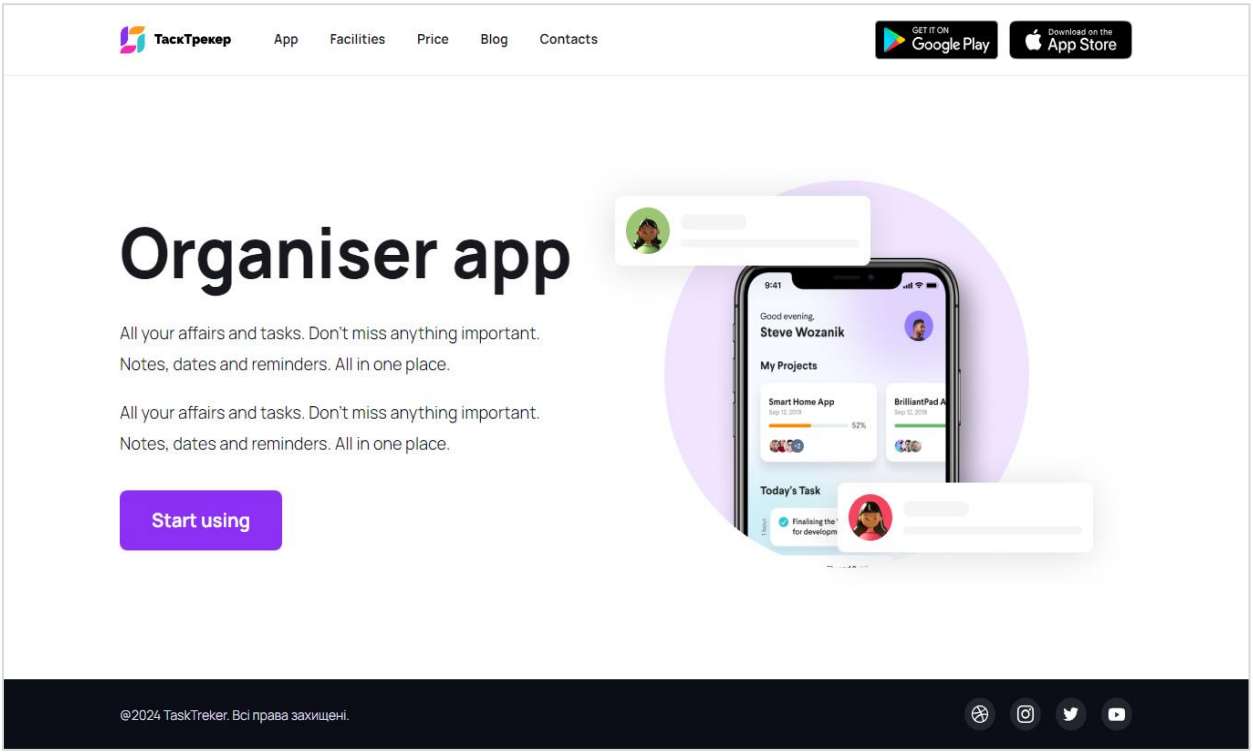

Рисунок 5 – Завершений вигляд лендінга

5. Додати до проєкту нову сторінку price.html (та посилання на неї в блоці nav) яка буде містити таблицю з попередньої лабораторної роботи, як показано на рисунку 6.

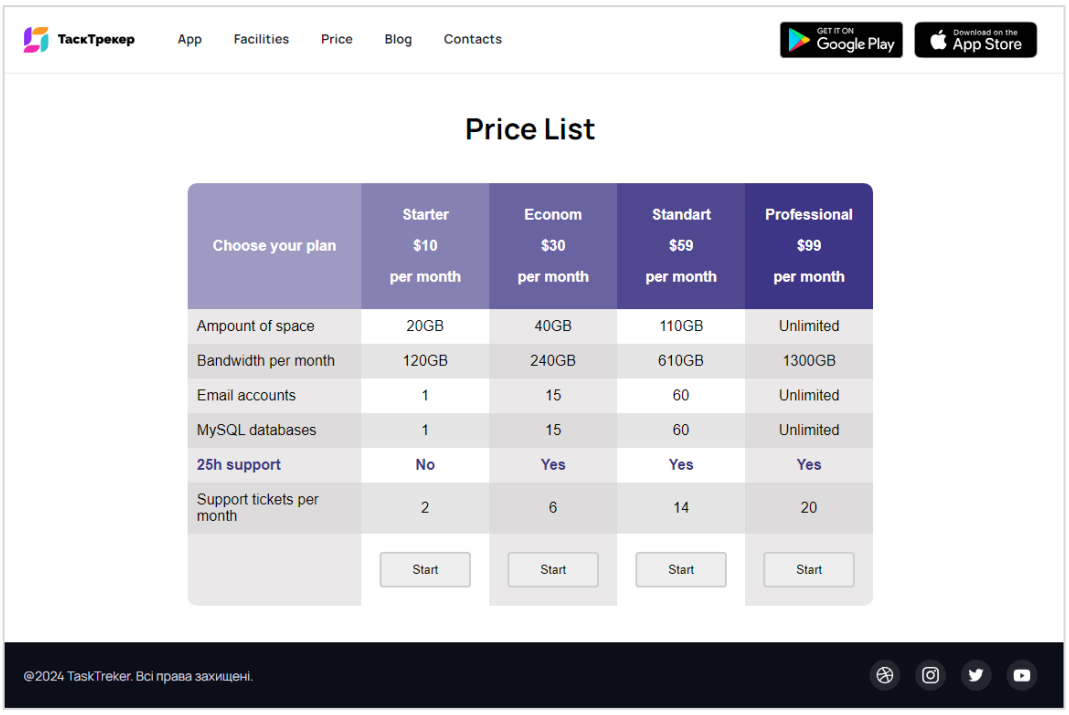

Рисунок 6 – Вигляд сторінки price.html

Теоретичні відомості

Box Model

У CSS кожен HTML-елемент розглядається як прямокутник (або «блок»). Box model складається з вмісту (контенту), внутрішнього відступу (padding), рамки (border) та зовнішнього відступу (margin). Відступи (padding, margin) та рамка (border) мають окремі сторони -top, -right, -bottom та -left.

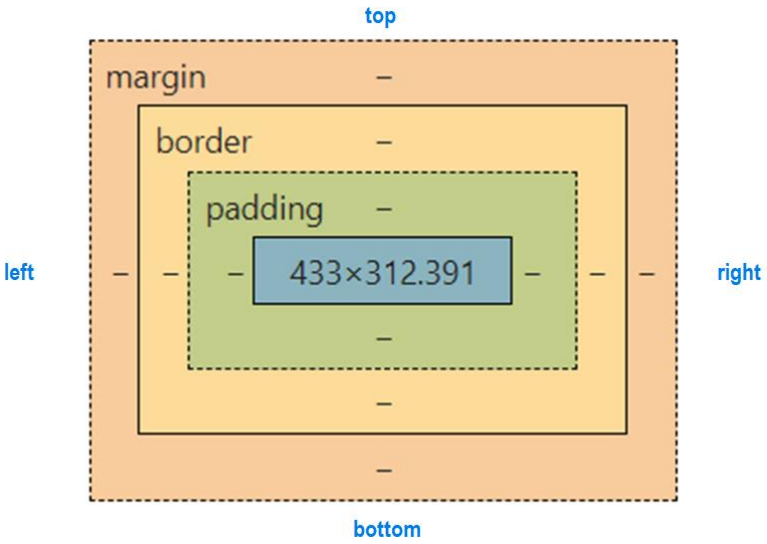

Рисунок 7 – Box model

Блок Content містить дві основні властивості:

- `width` – ширина контенту;
- `height` – висота контенту;

По замовчуванню border та padding впливають на розміри блоку на сторінці і обраховуються по принципу:

```
Total box width = padding-left + border-left + content width + border-right + padding-right;

Total box height = padding-top + border-top + content height + border-bottom + padding-bottom;
```

Відступи

CSS дає змогу задавати наступні варанти внутрішніх відступів:

- `padding-top: 10px;`
- `padding-right: 20px;`
- `padding-bottom: 30px;`
- `padding-left: 40px.`

Даний запис можна записати в скороченому вигляді:

```
padding: 10px 20px 30px 40px; /*top right bottom left*/
```

А коли відступи рівні:

- `padding-top: 10px;`
- `padding-right: 10px;`
- `padding-bottom: 10px;`
- `padding-left: 10px.`

Можна записати: `padding: 10px;`

Коли у нас повторюються відступи зверху та знизу а також зліва та справа:

- `padding-top: 10px;`
- `padding-bottom: 10px;`
- `padding-right: 20px;`
- `padding-left: 20px.`

Можемо скористатися записом:

```
padding: 10px 20px; /*top-bottom right-left*/
```

Користуючись відповідними правилами, ми можемо задавати зовнішній відступ (margin) та рамку (border) властивості елемента.

Приведемо приклад:

```
border-left: 3px solid red; /* 3px ліва суцільна червона рамка */
margin: 10px 20px 30px 40px; /* зовнішій відсту згори - 10px, зправа - 20px, знизу - 30px, зліва - 40px*/
```

Блочні елементи

Блочні елементи в HTML/CSS – це тип елементів, які займають всю доступну ширину батьківського контейнера та відображаються у відповідному блочному контексті.

Ключові характеристики блочних елементів:

– *займають всю доступну ширину:* блочні елементи, за замовчуванням, розтягуються горизонтально, від лівого до правого краю батьківського елемента;

– *займають новий рядок:* кожен блочний елемент розміщується на новому рядку, тобто наступний блок починається з нового рядка;

– *можуть містити інші блочні та строчні елементи:* блочні елементи можуть містити як інші блочні елементи, так і строчні елементи;

– *можуть мати зовнішні та внутрішні відступи, обмеження та рамки:* властивості CSS, такі як padding, margin, border, можуть бути застосовані до блочних елементів;

– *дозволяють контролювати розміщення елементів на сторінці:* блочні елементи можуть бути легко позиціоновані на сторінці за допомогою CSS властивостей, таких як position, float, display, тощо.

Приклади блочних елементів в HTML включають , <p>, <h1> - <h6>, <form>, <header>, <footer>, <section>, <nav>, <article>, тощо. Коли ви хочете, щоб елемент відображався як блок, ви можете використовувати CSS властивість display: block.

Рядкові (in-line) елементи

Інлайнові елементи в HTML/CSS – це тип елементів, які не розтягуються на всю доступну ширину а займають ширину контенту який вміщають та зазвичай відображаються поруч з іншими елементами в тому ж рядку.

Ось кілька ключових характеристик *інлайнових елементів*:

– *не розтягуються на всю доступну ширину:* інлайнові елементи займають лише стільки місця, скільки необхідно для вмісту всередині них;

– *не починають новий рядок:* інлайнові елементи можуть бути відображені поруч з іншими елементами в тому ж рядку, якщо вони мають достатньо місця;

– *можуть бути вміщені в блоки:* інлайнові елементи можуть бути вміщені всередині блокових елементів, але вони не можуть містити інші блокові елементи всередині себе;

– *не можуть мати зовнішніх відступів у всіх напрямках:* інлайнові елементи можуть мати зовнішні відступи тільки по горизонталі (left та right), а не по вертикалі (top та bottom). Однак вони можуть мати внутрішні відступи (padding) та рамки;

– *дозволяють контролювати розміщення елементів в межах рядка:* інлайнові елементи можуть бути легко позиціоновані у межах рядка за допомогою CSS властивостей, таких як vertical-align, line-height, тощо.

Приклади інлайнових елементів в HTML включають , <a>, <img>, , , <i>, <b>, <input>, <label>, тощо. Коли ви хочете, щоб елемент відображався як інлайн, ви можете використовувати CSS властивість display: inline; або display: inline-block; для комбінації інлайнового позиціонування з можливістю встановлення ширини та висоти.

Flex модель

Flex дозволяє використовувати більш ефективний спосіб розмітки, вирівнювання й розподілу вільного місця між елементами у контейнері.

Щоб почати працювати з CSS Flexbox потрібно зробити контейнер flex-контейнером:

```
#container {
  display: flex;
}
```

Рисунок 8 – flex-контейнер

У нього є дві осі (рисунок 9): головна та перпендикулярна їй.

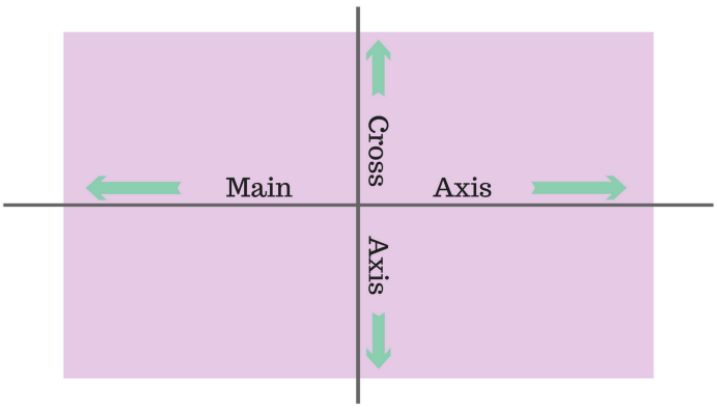

Рисунок 9 – Вісі flex-контейнера

За замовчуванням усі елемети розташовуються вздовж головної осі – зліва направо. Властивість **flex-direction: row | column | row-reverse | column-reverse** дозволяє змінювати кут головної осі відносно якої розміжуватимуться елементи flex-контейнера.

Вирівнювання елементів в середині контейнера відбувається за допомогою властивостей **justify-content** та **align-items.**

Властивість **justify-content** приймає наступні значення для вирівнювання елементів вздовж головної осі:

- `flex-start /*вирівнювання по лівому краю*/;`

```
- flex-end /* по правому краю*/;
- center  /*  по центру*/;
- space-between /* пропорційні відступи між елементами */;
-  space-around  /*  пропорційні  відступи  перед,  після  та  між
елементами */.
```

Властивість **align-items** приймає наступні значення для вирівнювання елементів вздовж перпендикулярної осі:

```
-  flex-start /*вирівнювання по верхньому краю*/;
- flex-end /* по нижньому краю*/;
- baseline /* вирівнювання відносно середини найвищого елемента*/;
- stretch /* розтягнення елементів вздовж перпендикулярної осі */.
```

Лабораторна робота №4

Тема: «Публікація сайту в інтернеті»

Мета: отримати навички роботи з хостингом.

Звіт повинен містити:

1. Тему та мету.
2. Постановка завдання.
3. Теоретичні відомості.
4. Результати виконання.
5. Висновок.

Завдання

1. Зареєструватися та авторизуватися на сайті з безкоштовним хостингом: https://dash.infinityfree.com/register

2. Розмістити розроблений та адаптований під мобільні пристрої сайт порт фоліо, на хостингу infinityfree.com або подібному безкоштовному аналозі.

3. Описати хід роботи.

4. Зробити висновки.

Теоретичні відомості

Для реєстрації на стайті, необхідно перейти по посиланню https://dash.infinityfree.com/register та заповнити просту форму (рисунок 10). Необхідно вказати свою поштову адресу та пароль (двічі для підтвердження). Після реєстрації на вказану пошту прийде лист з підтвердженням реєстрації та активації створеного аккаунта.

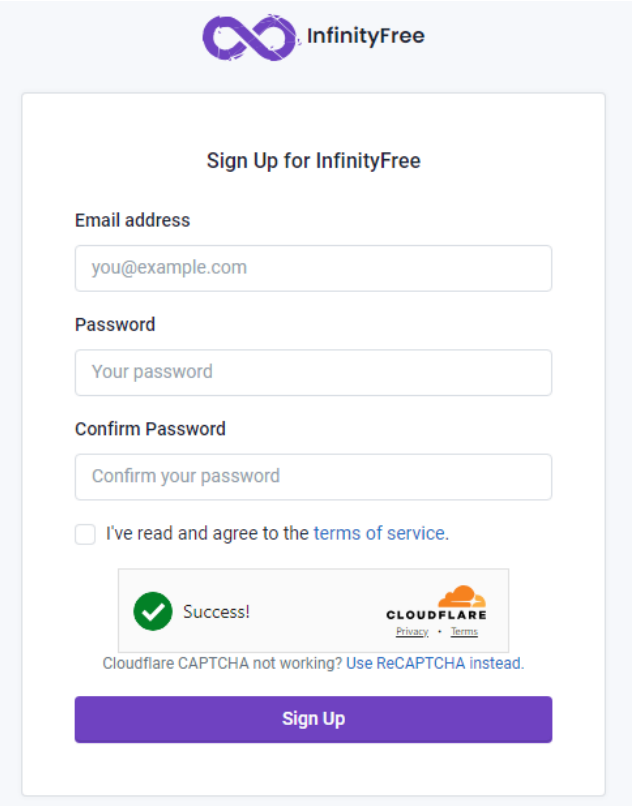

Рисунок 10 – Форма реєстрації

Після активації облікового запису необхідно авторизуватися на сайті перейшовши на головну сторінку та натиснувши кнопку «Login». Після чого необхідно заповнити форму авторизації та натиснути кнопку «Sign In».

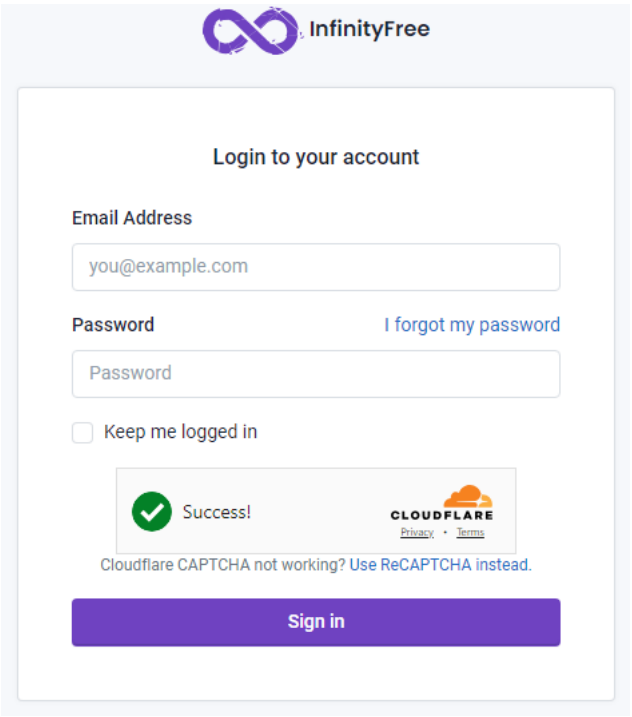

Рисунок 11 – Форма авторизації

Після успішної авторизації, ви побачите сторінку з хостинг-акаунтами, де одному хостинг-акаунту відповідає один сайт. Ви маєте змогу безкоштовно створити 3 хостинг-акаунти (рисунок 12).

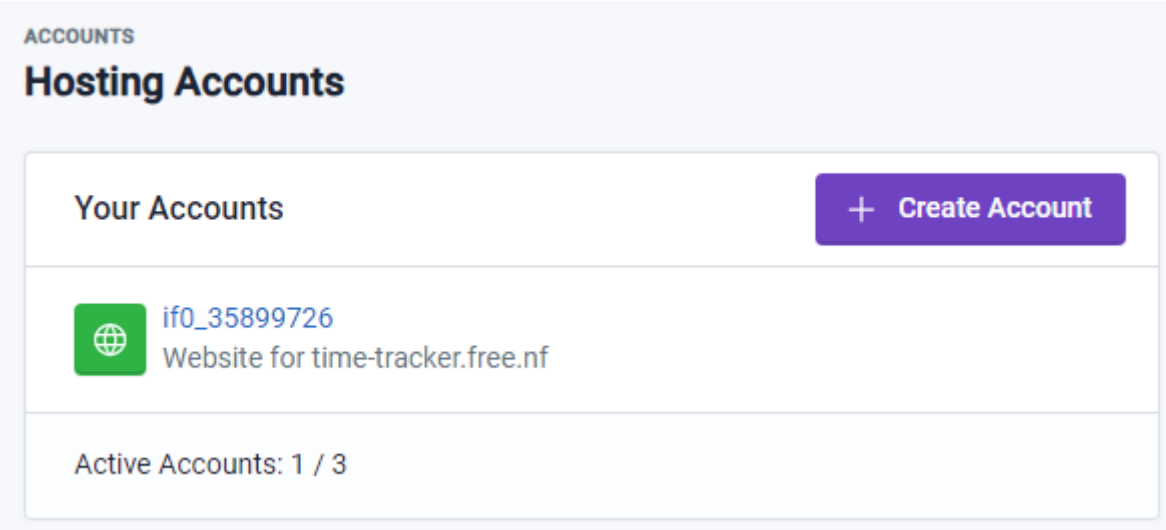

Рисунок 12 – Хостинг-аккаунти

Для створення нового хостинг-акаунту необхідно натиснути кнопку «Create Account». Після чого, необхідно вибрати безкоштовний хостинг-план як зображео на рисунку 13 нижче.

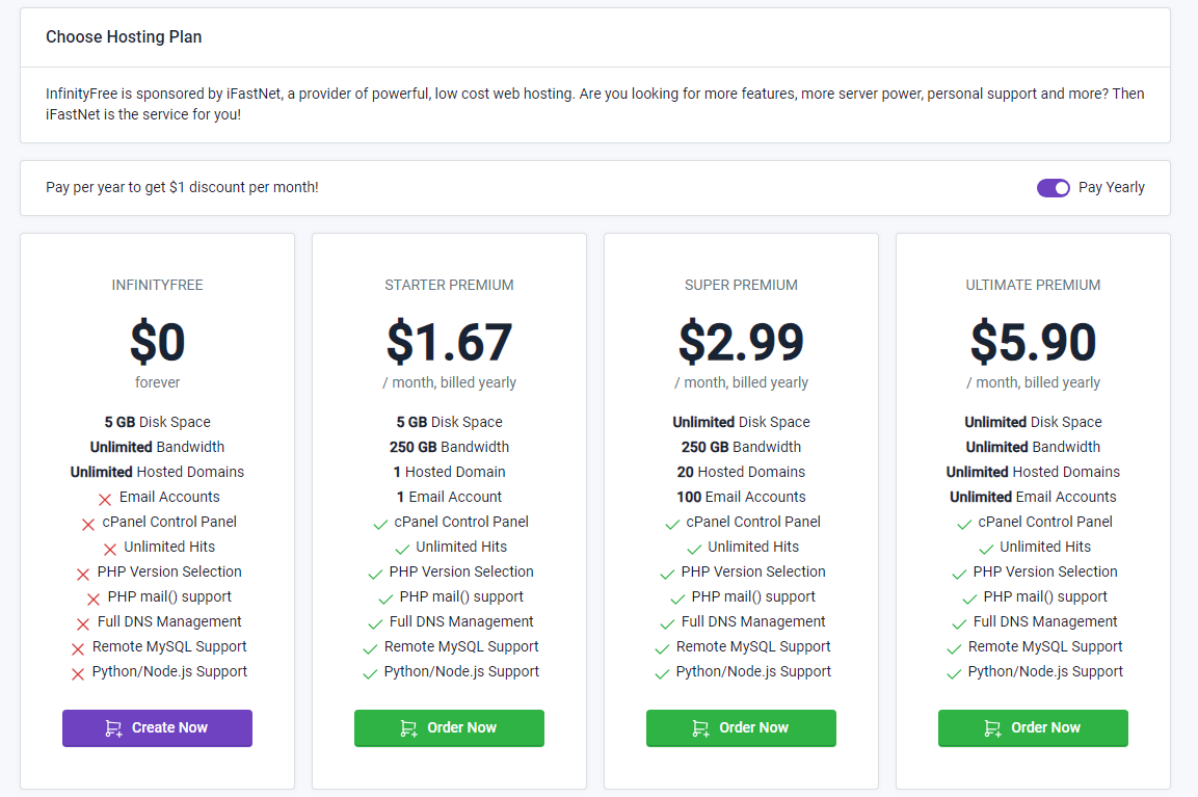

Рисунок 13 – Обрання хостинг плану

Другим кроком є обрання доменного імені (адреси) майбутнього сайту (рисунок 14), що складається із унікального під-домену (Subdomain) та префіксу (Domain extention) і який можна обирати із запропонованого списку.

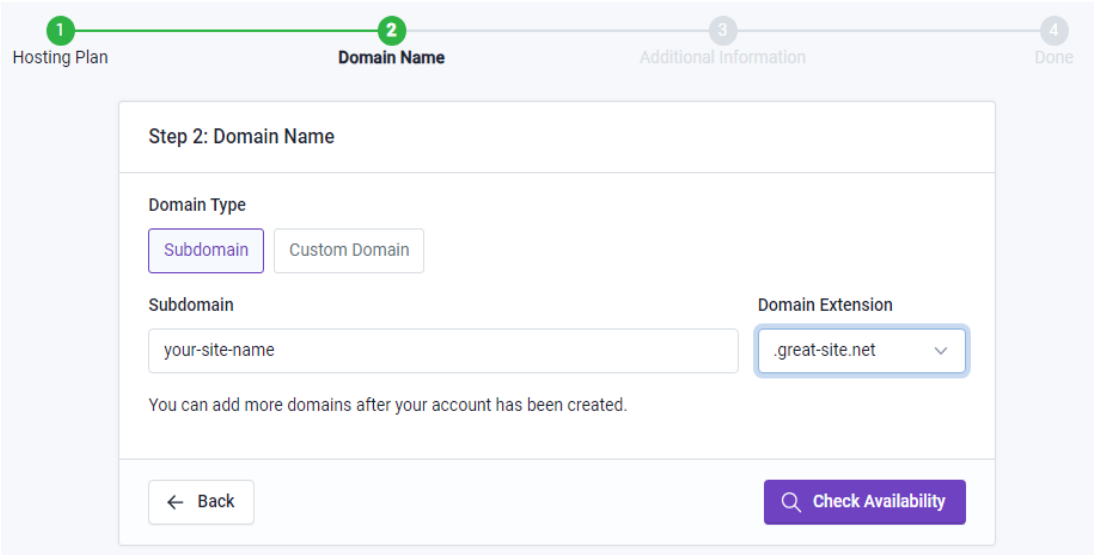

Рисунок 14 – Обрання доменного імені сайту

Після заповнення даних, необхідно перевірити чи такої адреси не було зареєстровано раніше, натиснувши відповідну кнопку «Check Avaliability».

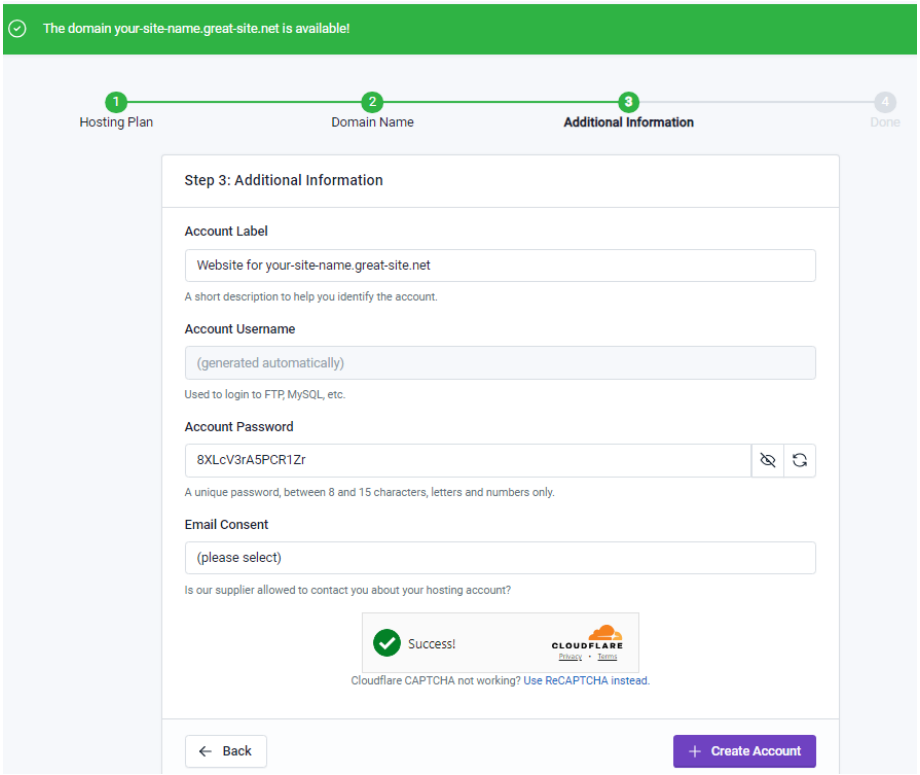

Рисунок 15 – Крок додаткова інформація

В разі доступності створеного імені, вам відобразиться вікно (рисунок 15) з додатковою інформацією про ваш акаунт. В даному вікні необхідно вибрати у полі *Email Consent* значення «I Approve», що надасть дозвіл постачальнику послуг зв'язатися з вами щодо питань вашого облікового запису. Після чого необхідно натиснути кнопку «Create Account».

Якщо все було виконанно згідно вимог, ви побачите вікно (рисунок 16).

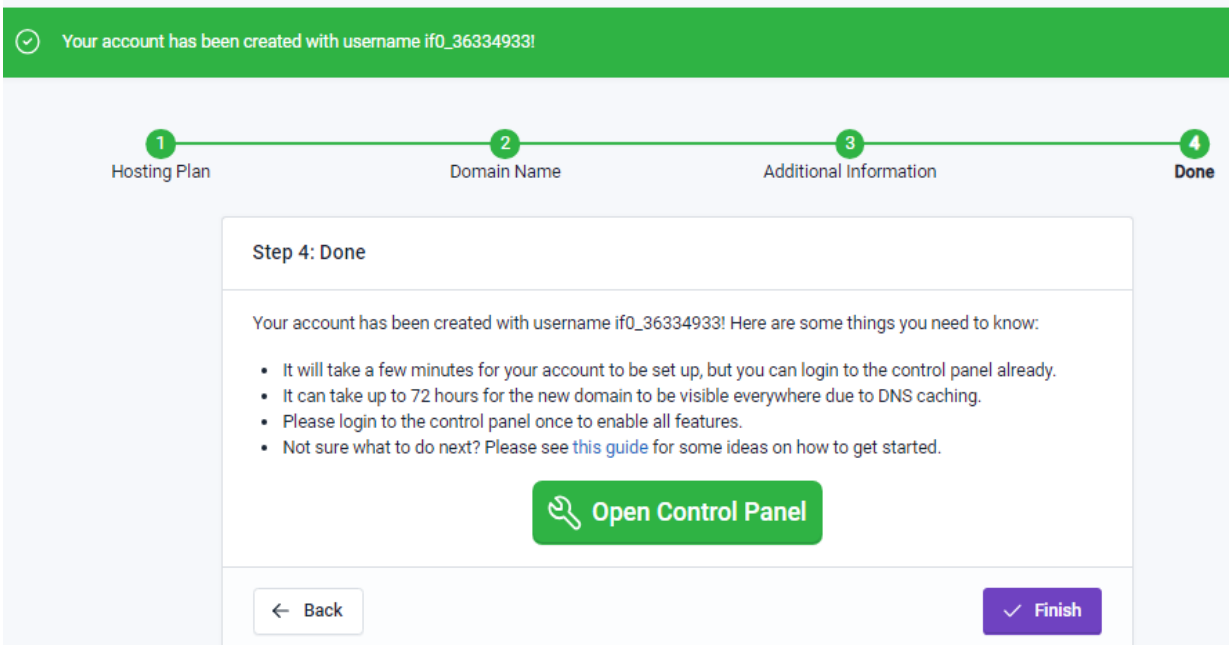

Рисунок 16 – Завершення створення акаунту

Далі необхідно натиснути кнопку «Open Control Panel», та підтвердити дозвіл на надсилання сповіщень про зміни (в сервісі і т.п.), що й зображено на рисунку 17 нижче.

Після підтвердження ви отримаєте доступ до адмін.-панелі, з детальною інформацією про ваш хостинг-акаунт з якою варто ознайомитися а потім перейти в клієнтську область натиснувши відповідку кнопку в головному меню під назвою «Client Area» (рисунок 18).

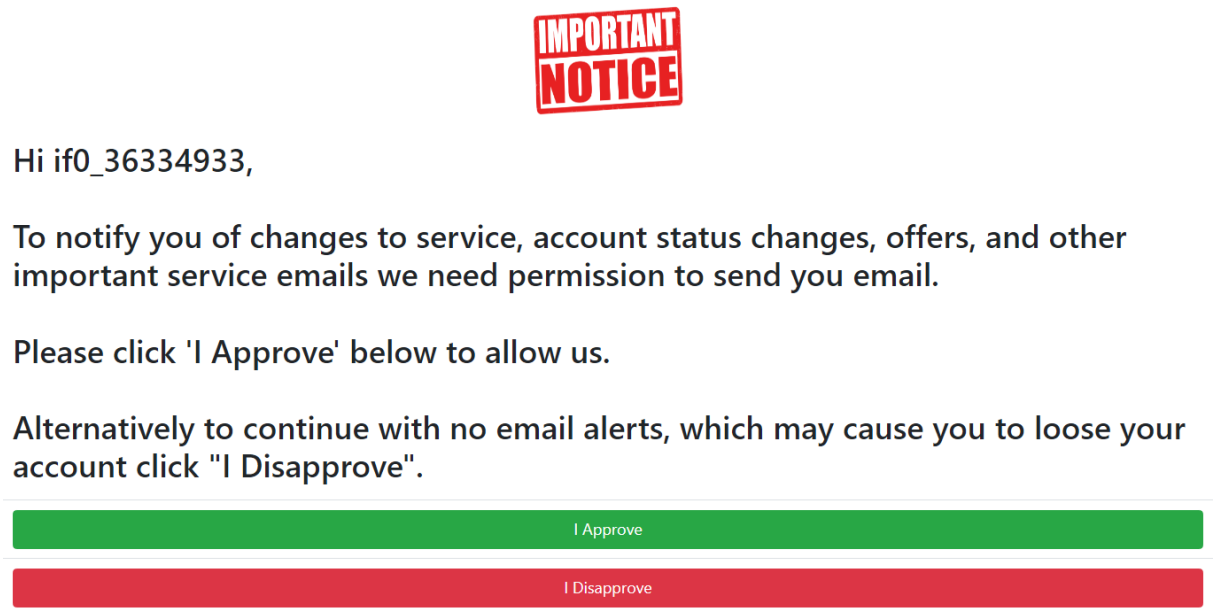

Рисунок 17 – Підтвердження дозволу на надсилання сповіщень

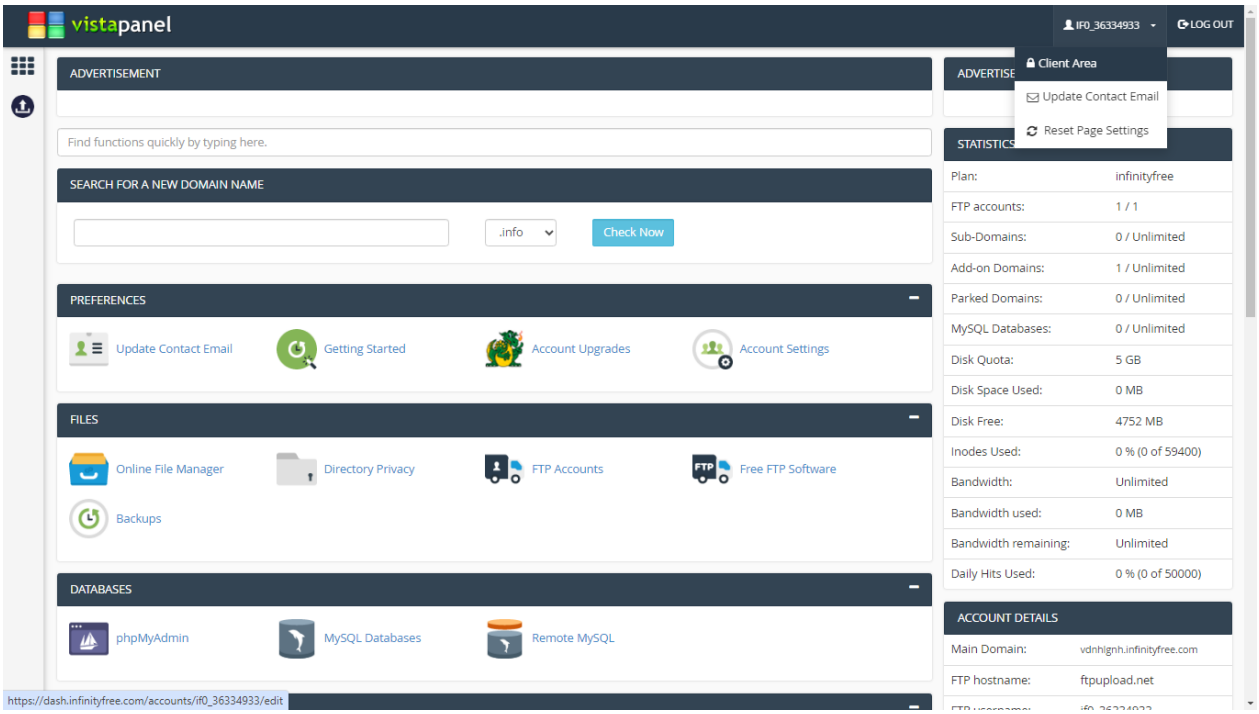

Рисунок 18 – Адміністративна панель

Перейшовши в клієнтську область, ми побачимо наступне вікно (рисунок 19).

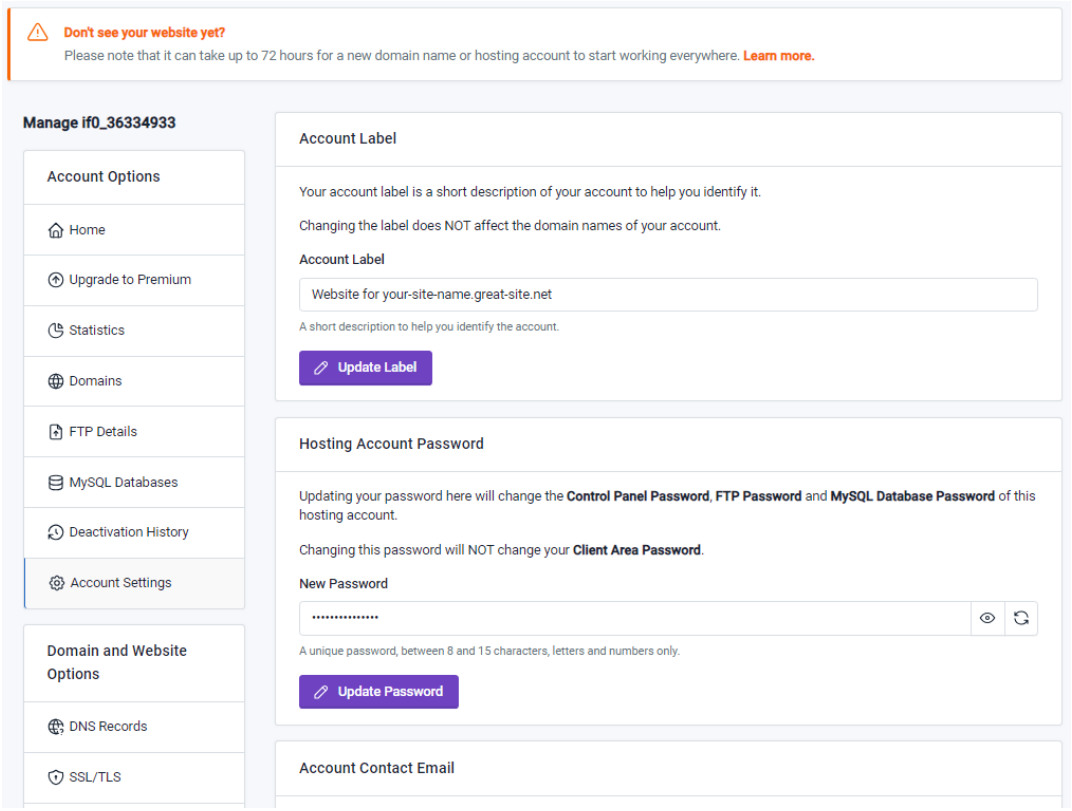

Рисунок 19 – Клієнстська область

Зліва на панелі необхідно натиснути на на посилання «Home» та на сторінці, що відкриється (рисунок 20), натиснути кнопку «File Manager».

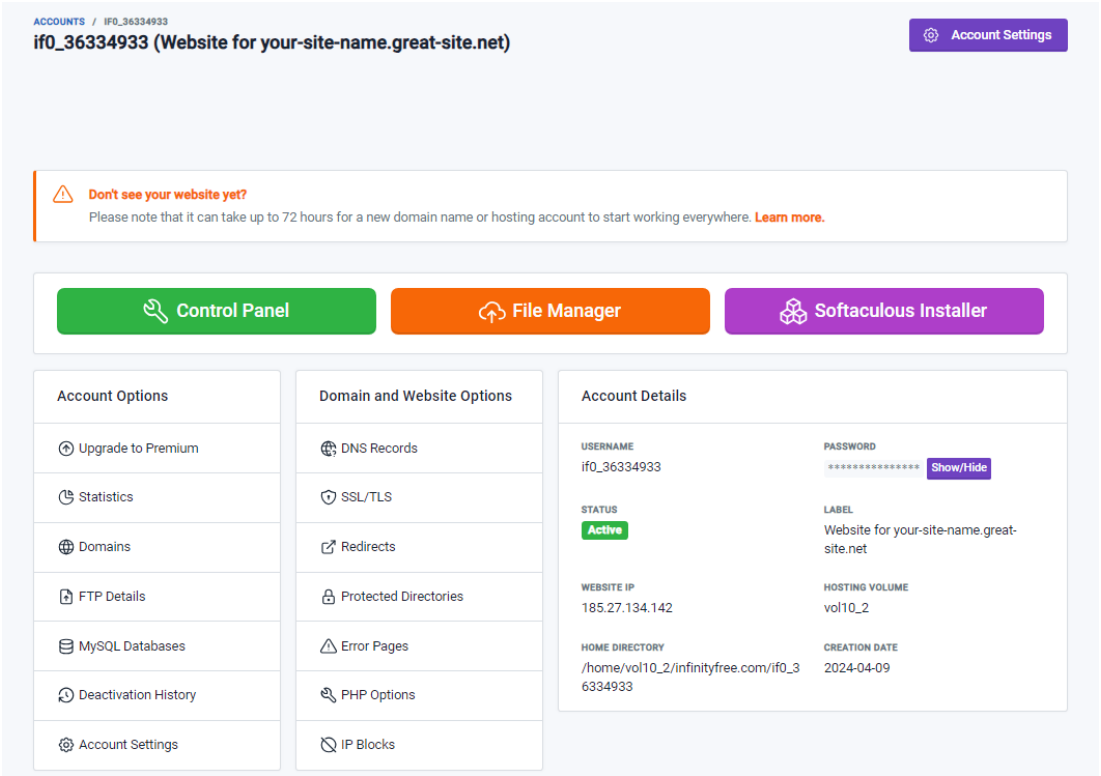

Рисунок 20 – Сторінка «Home»

У вікні файлового менеджера, що відкриється в новій вкладці, необхідно відкрити папку «htdots» в якій і будуть розміщуватися файли вашого сайту.

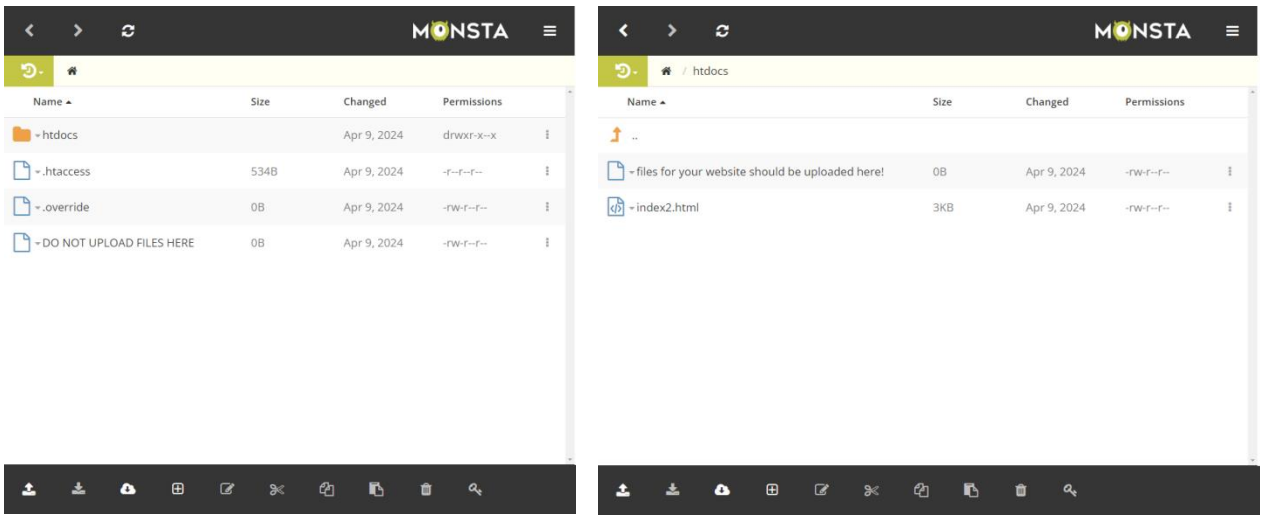

Рисунок 21 – Вікно файлового менеджера

Ми маємо змогу видалити всі файли з папки «htdots» та завантажити на їх місце файли нашого сайту. Це можна зробити двома способами, натиснувши на кнопку «Upload» із стрілкою «↑» на панелі, що знаходиться в низу сторінки, та вибрати, що саме ви завантажуєте, файл чи папку. Або за допомогою функції перетягування (Drag & Drop) необхідних файлів і папок у вікно менеджера.

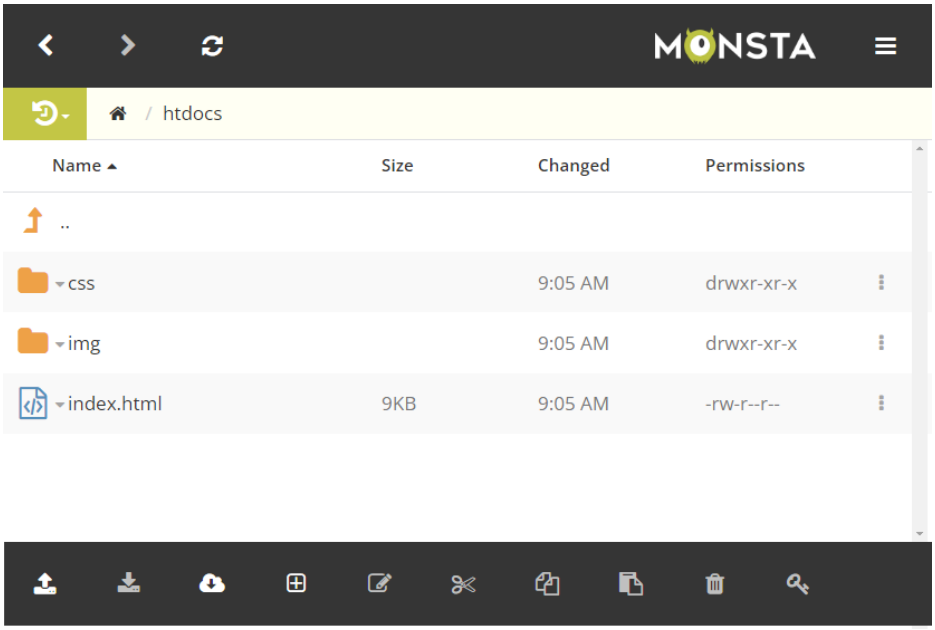

Рисунок 22 – Вміст папки «htdots» після завантаження файлів

Після завантаження файлів, на диск сервера, всіст папки «htdots» має виглядати так як вміст папки проекту на вашому локальному диску (рисунок 22).

Повертаючичь на вкладку із сторінкою «home» ми можемо побачити попередження, що на публікацію сайту може піти до 72 годин, хоча як показує практика сайт публікується значно швидше (менше години).

Для перевірки опублікованого сайту, необхідно натиснути на пункт меню «Domains» та перейти по посиланню вашого сайту, як показано на рисунку 23.

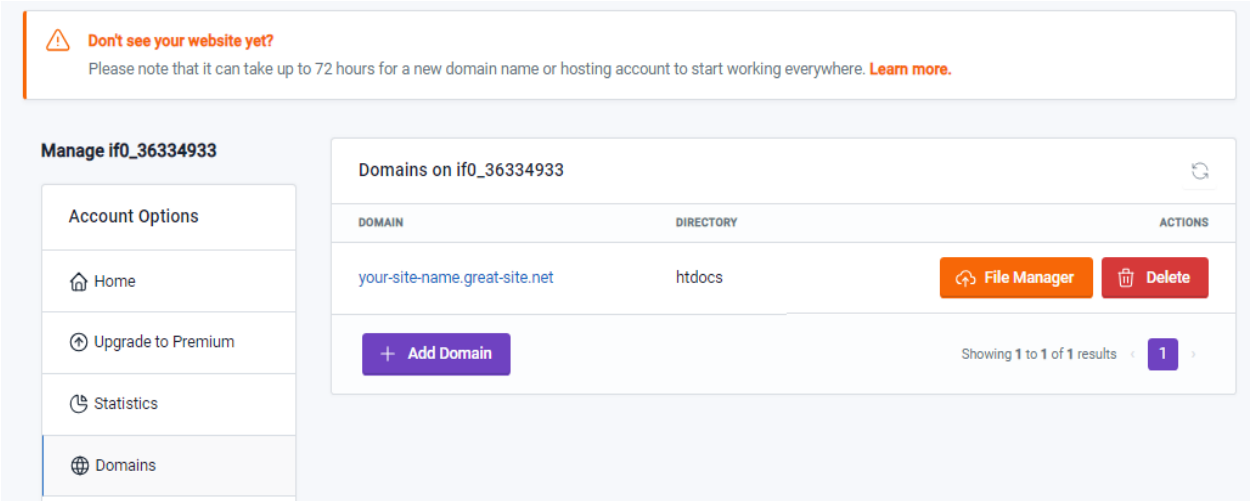

Рисунок 23 – Вкладка «Domains»

В разі успішної публікації сайту в мережі інтернет, ви побачите розроблену вами сторінку (рисунок 24) із глобальною адресою, яка доступна всім в інтернеті.

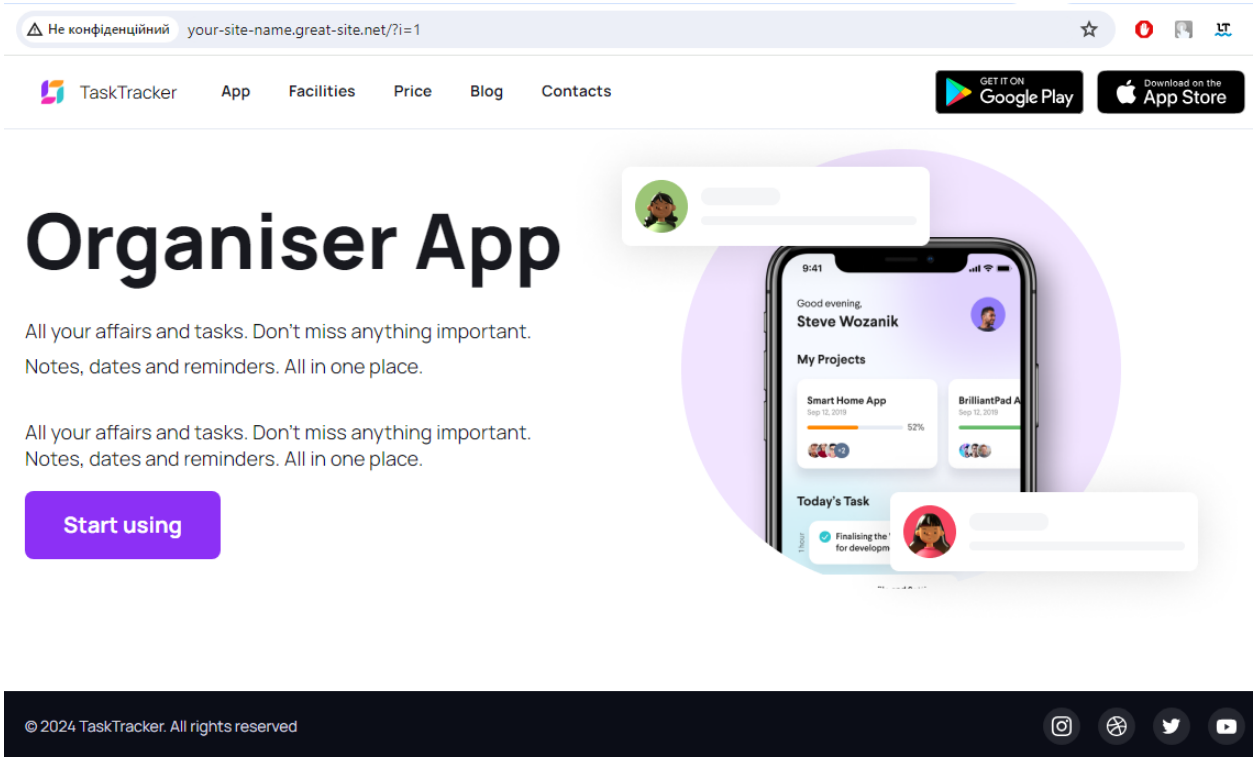

Рисунок 24 – Результат публікації сайту

Лабораторна робота №5

Тема: «Створення сайту портфоліо»

Мета: отримати навички роботи з багатосторінковими сайтами.

Звіт повинен містити:

1. Тему та мету;
2. Постановку завдання;
3. Теоретичні відомості;
4. Результати виконання;
5. Висновок.

Завдання

Завдання підвищеної складності

1. Створити розмітку макету по принципу desktop-first: https://www.figma.com/file/65jqO6WQdYUXxI0yOL4VfA/Porto?node-id=0%3A1

Завдання середньої складності

1. Знайти та виправити помилки в розмітці існуючого макету портфоліо (HTML та CSS код знаходитсь нижче), щоб його вигляд відповідав зображеному на рисунку 25.

Спільне завдання

1. Для обраного на попередньому кроці макету, створити html-сторінки (застосовуючи стилі обраного макету) для представлення кожної попередньої лабораторної роботи, включаючи її опис. Посиланя мають бути на роботи: автобіографія, таблиця digital-продукут, сайт task tracker[, власні роботи].

2. Додати посилання на кожну розроблену (п.2) сторінку в розділі потрфоліо з відповідним preview image. Приклад сторінки з роботою для макету середньої складності зображено на рисунку 26.

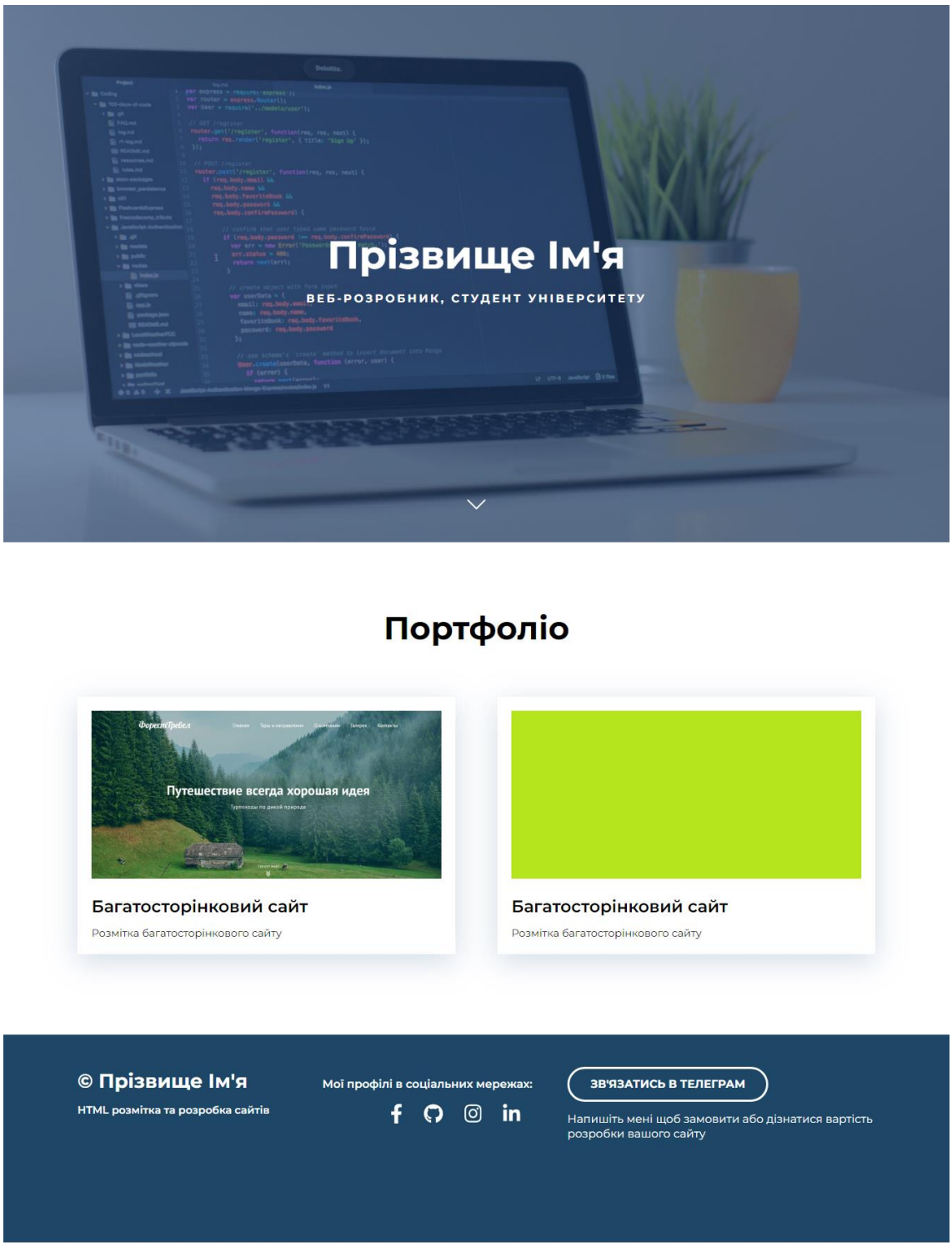

Рисунок 25 – Результати після виправлення

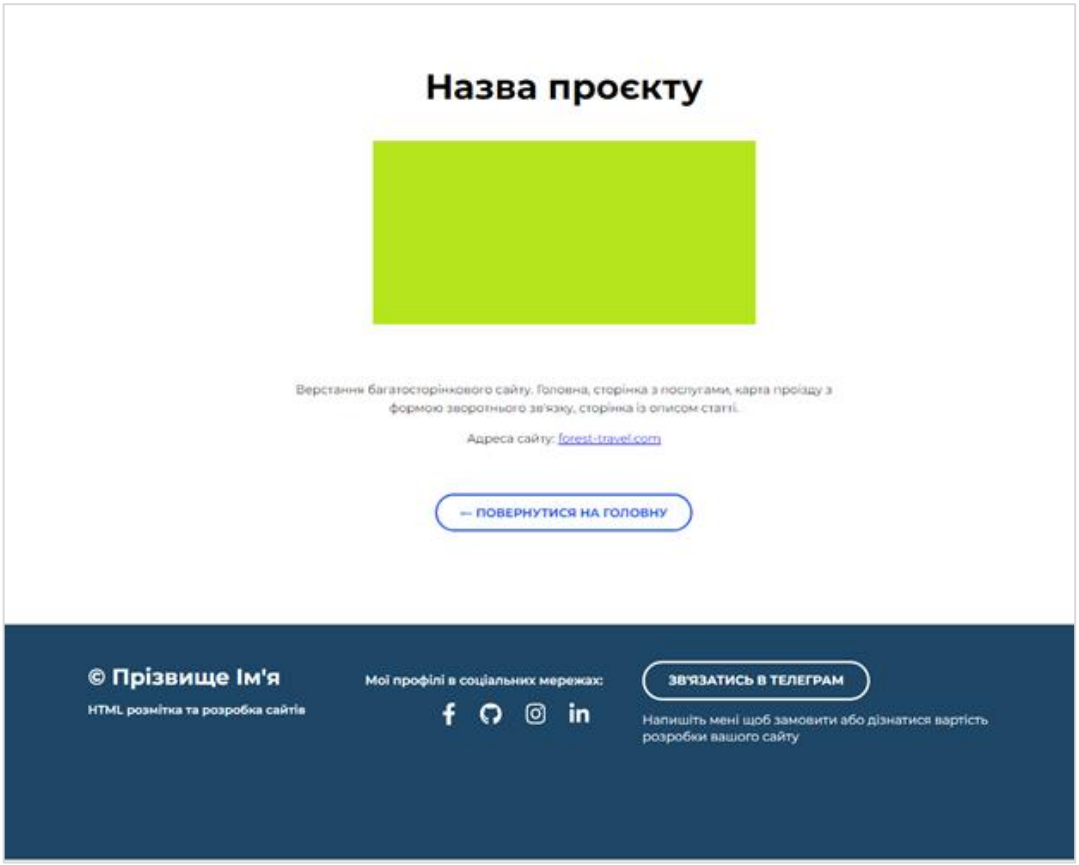

Рисунок 26 – Приклад сторінки проєкту середньої складності

HTML-код макету середньої складності, який містить помилки

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

<head>
    <meta charset="UTF-8">

    <link rel="stylesheet" href="./css/style.css">

    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com"
crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@
400;600;700&display=swap" rel="stylesheet">

    <title>Сайт портфоліо</title>
</head>
```

```
<body>

    <header class="header" id="header">
        <h1 class="header-title">Прізвище Ім'я</h1>
        <p class="header-subtitle">Веб-розробник, студент
університету</p>
        <div class="header-arrow">
            <a href="#portfolio">
                <img src="./img/header/chevron-down.svg">
            </a>
        </div>
    </header>

    <main class="portfolio" id="portfolio">
        <div class="container">
            <h2 class="-header">Портфоліо</h2>

            <div class="portfolio-cards-wraper">

                <div class="cart">
                    <a href="#" class="cart-link">
                        <img class="cart-img" src="...">
                        <h3 class="cart-
title">Багатосторінковий сайт</h3>
                        <p>Розмітка багатосторінкового
сайту</p>
                    </a>
                </div>

                <div class="cart">
                    <a href="#" class="card-link">
                        <img class="card-img"
src="./img/portfolio/2.png">
                        <h3 class="card-
title">Багатосторінковий сайт</h3>
                        <p>Розмітка багатосторінкового
сайту</p>
                    </a>
                </div>
```

```
            </div>
        </div>

    </main>

    <footer class="foter">
        <div class="container">
            <div class="foter-row">

                <div class="foter-copyright foter-col">
                    <div class="foter-copyright-name">&copy;
Прізвище Ім'я</div>
                    <p>HTML розмітка та розробка сайтів</p>
                </div>

                <div class="foter-icon footer-col">
                    <p>Мої профілі в соціальних мережах:</p>
                    <div class="foter-icons-row">
                        <a href="https://www.facebook.com/">
                            <img
src="./img/footer/facebook.svg">
                        </a>
                        <a href="https://github.com/">
                            <img src="./img/footer/github.svg">
                        </a>
                        <a href="#">
                            <img
src="./img/footer/instagram.svg">
                        </a>
                        <a href="https://www.linkedin.com/">
                            <img
src="./img/footer/linkedin.svg">
                        </a>
                    </div>
                </div>

                <div class="foter-contact footer-col">
                    <a href="" class="footer-buton">Зв'язатись
в телеграм</a>
```

```
                    <p>Напишіть мені щоб замовити або дізнатися
вартість розробки вашого сайту</p>
                </div>

            </div>
        </div>
    </footer>

</body>

</html>
```

CSS – код макету

```
* {
    box-sizing: border-box;
}

html {
    scroll-behavior: smooth;
}

body {
    margin: 0;
    font-family: 'Montserrat', sans-serif;
}

.none {
    display: none !important;
}

img {
    max-width: 100%;
}

.conteiner {
    max-width: 1140px;
    margin-left: auto;
    margin-right: auto;
}
```

```css
/* header */

.header {
    position: relative;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    height: 100vh;
    max-width: 100%;

    background-image: url("./../img/header/header-bg.jpg");
    background-color: #556983;
    background-size: cover;
    background-position: center;
}

.header-title {
    margin: 0;
    font-weight: 700;
    font-size: 56px;
    line-height: 1.3;
    color: white;
}

.header-subtitle {
    font-weight: 700;
    font-size: 16px;
    text-align: center;
    letter-spacing: 0.2em;
    text-transform: uppercase;
    line-height: 1.3;
    color: white;
}

.header-arrow {
    position: absolute;
    height: 48px;
    width: 48px;
```

```
    bottom: 30px;
    left: 50%;
    transform: translate(-50%,0);
}

/* Portfolio */

.portfolio {
    padding-top: 80px;
    padding-bottom: 115px;
}


.portfolio-header {
    font-size: 46px;
    font-weight: 700;
    margin-bottom: 70px;
    text-align: center;
}

.portfolio-cards-wrapper {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
}

.cards {
    border: 1px solid;
}

.card {
    background-color: white;
    max-width: 540px;
    margin-bottom: 30px;
    box-shadow: 0px 10px 40px rgba(126, 155, 189, 0.35);
    transition: transform 0.2s ease-in;
}

.card:hover {
    transform: translateY(-15px);
```

```
}

.card-link {
    display: block;
    padding: 20px;
    text-decoration: none;
    color: #000;
    transition: color 0.2s ease-in;
}

.card-link:hover {
    color: rgb(39, 91, 236);
}

.card:nth-last-child(2), .card:last-child  {
    margin-bottom: 0;
}

.card-title {
    margin: 0;
    margin-bottom: 13px;
    font-weight: 600;
    font-size: 24px;
    line-height: 1.3;
}

.card-img {
    margin-bottom: 20px;
}

.card p {
    margin: 0;
    line-height: 1.3;
    font-size: 15px;
}

/* footer */

.footer {
    padding-top: 45px;
```

```
    padding-bottom: 130px;
    background-color: #1e4767;
    color: white;
}

.footer-row {
    display: flex;
    flex-direction: row;
    align-items: baseline;
    justify-content: space-between;
}

.footer-col {
    min-width: 350px;
    flex-grow: 1;
}

.footer-icons-row {
    display: flex;
    align-items: center;
    justify-content: center;
}

.footer-icons-row > * {
    margin-left: 30px;
    text-decoration: none;
}

.footer-copyright-name {
    font-size: 28px;
    font-weight: 700;
    margin-bottom: 15px;
}

.footer-copyright p {
    margin: 0;
    font-size: 15px;
    font-weight: 600;
    margin-bottom: 15px;
}
```

```
.footer-icons p {
    font-size: 16px;
    font-weight: 600;
    line-height: 1.3;
}

.footer-button {
    display: inline-block;
    height: 50px;
    padding-left: 30px;
    padding-right: 30px;
    padding-top: 12px;

    color: #fff;
    border: 3px solid #fff;
    border-radius: 50px;

    font-size: 16px;
    font-weight: 700;

    text-decoration: none;
    text-transform: uppercase;
    transition: all 0.2s ease-in;
}

.footer-button:hover {
    color: #1e4767;
    background-color: #fff;
}

.footer-contacts p {
    margin: 15px 0;
    font-size: 16px;
    line-height: 1.3;
}

/* Project */

.project {
```

```
    padding: 80px 0 120px 0;
    text-align: center;
}

.project-header {
    margin: 0;
    margin-bottom: 40px;
    font-weight: 700;
    font-size: 46px;
    line-height: 1.3;
}

.project-img {
    margin-bottom: 70px;
}

.project-description {
    margin: 0 auto 60px;
    max-width: 730px;
}

.project-description p {
    margin-top: 0;
    margin-bottom: 15px;
    font-size: 16px;
    line-height: 1.5;
}

.project-back-btn {
    display: inline-block;
    height: 50px;
    padding-left: 30px;
    padding-right: 30px;
    padding-top: 12px;

    color: #275BEC;
    border: 3px solid #275BEC;
    border-radius: 50px;

    font-size: 16px;
```

```
    font-weight: 700;

    text-decoration: none;
    text-transform: uppercase;
    transition: all 0.2s ease-in;
}

.project-back-btn:hover {
    background-color: #275BEC;
    color: #fff;
}
```

Теоретичні відомості

Анімація в CSS

Анімація в CSS – це процес створення рухомих ефектів на веб-сторінках за допомогою каскадних таблиць стилів. Основна мета анімації в CSS – зробити веб-сайт більш привабливим та взаємодієвим, надаючи різноманітних візуальних ефектів, які привертають увагу користувачів.

Основні завдання анімації в CSS включають:

– *залучення уваги користувачів*: анімація може використовуватися для виділення важливих елементів на сторінці, таких як кнопки, підказки або повідомлення;

– *покращення користувацького досвіду (UX/UI)*: анімація може зробити веб-сайт більш інтуїтивно зрозумілим та дружнім для користувача (user friendly), підкреслюючи важливі дії або зміни на сторінці;

– *створення візуальних ефектів*: CSS анімація дозволяє створювати різноманітні візуальні ефекти, такі як переходи кольорів, плавні рухи, пульсуючі або розширюючі об'єкти;

– *підвищення конверсії*: якщо анімація використовується мудро, вона може привертати увагу користувачів до ключових елементів, що може підвищити конверсію дій, таких як кліки на кнопки або заповнення форм;

– *додавання виразності до дизайну*: анімація дозволяє підкреслити особливості дизайну, зробити його більш виразним та привабливим для користувачів.

Усі ці завдання мають за мету поліпшити взаємодію користувачів з веб-сайтом, роблячи його більш цікавим, інформативним та привабливим.

Для роботи з анімацією в CSS слугують наступні властивості:

- `@keyframes`
- `animation-name`
- `animation-duration`
- `animation-delay`
- `animation-iteration-count`
- `animation-direction`
- `animation-timing-function`
- `animation-fill-mode`
- `animation`

Нижче преведена таблиця підтримки відповідних властивостей в різних версіях браузерів, що варто враховувати при розробці (таблиця 2).

Таблиця 2 - Підтримка властивостей в різних версіях браузерів

Property					
@keyframes	43.0	10.0	16.0	9.0	30.0
animation-name	43.0	10.0	16.0	9.0	30.0
animation-duration	43.0	10.0	16.0	9.0	30.0
animation-delay	43.0	10.0	16.0	9.0	30.0
animation-iteration-count	43.0	10.0	16.0	9.0	30.0
animation-direction	43.0	10.0	16.0	9.0	30.0
animation-timing-function	43.0	10.0	16.0	9.0	30.0
animation-fill-mode	43.0	10.0	16.0	9.0	30.0
animation	43.0	10.0	16.0	9.0	30.0

1. Властивість @keyframes

CSS @keyframes – це спосіб створення анімацій у CSS, який дозволяє визначити зміни стилів за допомогою визначення кадрів (frames) протягом певного проміжку часу. Основна структура @keyframes виглядає наступним чином:

```
@keyframes changeColor {
    0% {
      background-color: red; /* Початковий колір */
    }
    50% {
      background-color: green; /*Перехід до зеленого на половині анімації */
    }
    100% {
      background-color: blue; /* Синій кінцевий колір */
    }
  }
```

2. Властивість animation-name та animation-duration

Для того щоб застосувати описану вище анімацію до html-елементу, наприклад блоку div, необхідно вказати у CSS правилі властивість *animation-name* присвоївши їй значення *changeColor* (назви визначеної в *@keyframes*). Тривалість анімації визначається у властивості *animation-duration*. Дане значення розподіляється рівномірно відносно відсоткового співвідношення описаного в *@keyframes*. Весь код матиме вигляд:

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: changeColor; /* Назва оголошена в @keyframes */
  animation-duration: 4s; /* Тривалість анімації у секундах*/
}
```

3. Властивість animation-delay

Властивість *animation-delay* задає затримку початку анімації. У наступному прикладі задано затримку у 2 секунди перед початком анімації:

```
div {
     …
    animation-name: changeColor; /* Назва оголошена в @keyframes*/
    animation-duration: 4s; /* Тривалість анімації у секундах*/
    animation-delay: 2s; /*Затримка перед початком анімації*/
  }
```

4. Властивість animation-iteration-count

Властивість *animation-iteration-count* задає кількість повторів зазначеної в *@keyframes* анімації. Може приймати цілочисельні значення (1, 2, 3 …) та константу «infinite», що вказує на безперервну тривалість анімації. Приклад нижче демонструє використання двох варіантів:

Скінченна анімація

```
div {
     …
    animation-name: changeColor; /* Назва оголошена в @keyframes*/
    animation-duration: 4s; /* Тривалість анімації у секундах*/
    animation-delay: 2s; /*Затримка перед початком анімації*/
    animation-iteration-count: 3; /*Повторюємо анімацію тричі*/
  }
```

Нескінченна анімація

```
div {
     …
    animation-name: changeColor; /* Назва оголошена в @keyframes*/
    animation-duration: 4s; /* Тривалість анімації у секундах*/
    animation-delay: 2s; /*Затримка перед початком анімації*/
    animation-iteration-count: infinite; /*Повторюємо нескінченно*/
  }
```

5. Властивість animation-direction

Визначає в якій послідовності (напрямку) будуть виконуватися оголошені в *@keyframes* кадри анімації. Вона дозволяє контролювати, чи анімація буде рухатись вперед, назад, чи чергувати ці напрямки. Властивість може приймати наступні значення:

– **normal** (за замовчуванням): анімація відтворюється вперед, від початкового стану (кадру) до кінцевого, а потім повертається до початкового стану;

– **reverse**: анімація відтворюється в зворотньому напрямку, від кінцевого стану до початкового, а потім повертається до кінцевого стану;

– **alternate**: анімація відтворюється вперед (від початкового до кінцевого стану), потім назад (від кінцевого до початкового), і так далі, циклічно;

– **alternate-reverse**: анімація відтворюється назад (від кінцевого до початкового стану) на першому циклі, а потім вперед (від початкового до кінцевого) на наступних циклах.

Відповідно до оголошеної в *@keyframes* **changeColor** анімації:

```
@keyframes changeColor {
    from { /*ідентично значенню 0%*/
      background-color: red; /* Початковий колір */
    }
    to { /*ідентично значенню 100%*/
      background-color: blue; /* Синій кінцевий колір */
    }
  }
```

При різних властивостях будемо мати наступний порядок виконання анімації:

– *animation-direction:* ***normal****;* - елемент спочатку буде червоним, потім стане синім, а потім знову червоним;

– *animation-direction:* ***reverse***; - елемент спочатку буде синім, потім стане червоним, а потім знову синім;

– *animation-direction:* ***alternate***; - елемент буде змінювати колір між червоним і синім в обидва напрямки.

6. Властивість animation-timing-function

В CSS *animation-timing-function* - це властивість, яка контролює те, як анімація розгортається протягом своєї тривалості. Вона визначає швидкість анімації та те, як вона переходить між своїми початковим і кінцевим станами.

Анімацію можна представити як подорож від точки А (початковий стан) до точки В (кінцевий стан). Властивість *animation-timing-function* диктує темп анімації протягом цієї подорожі.

Поширені функції синхронізації:

- **ease (за замовчуванням):** анімація починається повільно, набирає обертів у середині, а потім сповільнюється ближче до кінця. Вона імітує рух у реальному світі, роблячи анімацію органічною;
- **ease-in:** анімація починається повільно і поступово прискорюється, доки не досягне стійкої швидкості. Це підходить для анімацій, які потребують набору динаміки або підкреслення початкового стану;
- **ease-out:** функція починає анімацію з постійною швидкістю, а потім поступово сповільнює її ближче до кінця. Це корисно для підкреслення кінцевого стану анімації або створення відчуття зупинки;
- **ease-in-out:** функція поєднує в собі як ease-in, так і ease-out, забезпечуючи збалансований потік. Анімація починається повільно, прискорюється в середині, а потім сповільнюється ближче до кінця;
- **linear:** Ця функція створює анімацію з постійною швидкістю. Анімація прогресує рівномірно від початку до кінця, що підходить для простих переходів користувацького інтерфейсу.

Розширене керування: cubic-bezier()

Для більш точного керування можна використовувати функцію cubic-bezier. Вона визначає криву Без'є за допомогою чотирьох значень (x1, y1, x2, y2), які діють як контрольні точки. Форма кривої визначає темп анімації.

```
.animation {
     animation-timing-function: cubic-bezier(0.2, 0.5, 0.8, 1);
}
```

7. *Властивість animation*

```
@keyframes transform {
  from { transform: translateX(0px); }
  to { transform: translateX(100px); }
}
```

Для скороченого запису всіх вище розглянутих властивостей слугує властивість **animation.** Запис відповідає правилу:

@keyframes duration | easing-function | delay | iteration-count | direction | fill-mode | play-state | name

Деякі елементи можуть бути пропущені, як в наведеному прикладі:

```
.animation-transform {
    /* duration | easing-function | delay | iteration-count | direction | fill-mode | play-state | name */
     animation: 1s ease-in-out 0s infinite alternate transform;
}
```

Лабораторна робота №6

Тема: «SASS. Препроцесори в CSS»

Мета: отримати навички роботи з препроцесором SAAS.

Звіт повинен містити:

1. Постановка завдання;
2. Теоретичні відомості по проекту;
3. Результати виконання завдання;
4. Висновок.

Завдання

1. Встановити пагін *SAAS Live Compiller* для *VS Code*.
2. Додати до проекту файл препроцесора з розширенням .scss
3. Сформувати на основі розробленого в попередній роботі css-файлу – файл препроцесора .scss.
4. Порівняти кількість рядків у файлах .scss та .css.
5. Скомпілювати scss-файл.
6. Описати хід роботи.
7. Зробити висновки.

Теоретичні відомості

SASS (Syntactically Awesome Stylesheet)

SASS є препроцесором CSS. Це означає, що SASS є розширенням CSS, яке додає додаткові функціональні можливості, такі як змінні, міксини (mixins), оператори та функції. Ці функціональні можливості роблять написання та підтримку CSS більш ефективними та організованими.

Переваги використання SASS:

- *змінні*: SASS дозволяє оголошувати та використовувати змінні, що робить ваш CSS більш динамічним та легким в підтримці. Наприклад, можна визначити одну змінну для основного кольору бренду вашого сайту та використовувати її в усьому вашому CSS-коді;
- *міксини (Mixins)*: міксини дозволяють створювати багаторазово використовувані блоки CSS-властивостей. Це допомагає підтримувати ваш код чистим, уникати повторень та спрощує внесення змін до стилів;
- *оператори та функції*: SASS пропонує різноманітні оператори та функції, які дозволяють виконувати математичні операції, маніпулювати рядками, змішувати кольори тощо. Це робить ваш код більш компактним та читабельним;
- *нестингова вкладеність*: SASS дозволяє використовувати вкладену структуру для організації стилів, що покращує читабельність коду.

Недоліки використання SASS:

- *необхідність додаткового етапу компіляції*: SASS-код потрібно компілювати у звичайний CSS-код перед тим, як браузер зможе його інтерпретувати. Це додатковий крок у робочому процесі. Але сучасний процес розробки, дозволяє це автоматизувати, шляхом встановлення відповідних плагінів (приклад SASS Live Compiler);
- *складність для початківців*: Якщо ви новачок в CSS, то SASS може здатися складнішим для освоєння, ніж звичайний CSS.

SASS Змінні

Змінні SASS є ключовим інструментом, який робить код CSS більш організованим, динамічним та гнучким. Вони дозволяють встановлювати централізовані значення дизайну та повторно використовувати їх у всьому коді, значно спрощуючи процес розробки та обслуговування.

Оголошення змінних

Оголошення змінних SASS використовує символ долара ($) перед ім'ям змінної, за яким йде значення. Ім'я змінної має бути описовим та відповідати стандартним правилам іменування CSS (починатися з літери або підкреслення, використовувати дефіси для розділення слів).

```
$variable-name: value;
```

Приклад:

```
$primary-color: #ff0000; /* Червоний колір */
$secondary-color: #00ffff; /* Блакитний колір */

body {
    background-color: $primary-color;
    color: $secondary-color;
}
```

У цьому прикладі ми оголошуємо дві змінні: `$primary-color` та `$secondary-color`. Потім використовуємо їх у селекторі `body`, щоб встановити фоновий та текстовий кольори.

Переваги використання змінних:

– *централізований контроль*: зберігаючи основні значення дизайну (кольори, шрифти, розміри) у змінних, ви можете легко змінити весь вигляд та відчуття вашого сайту, просто оновивши визначення змінних;

– *злагодженість*: використання змінних гарантує послідовне застосування кольорів, шрифтів та інших елементів дизайну у ваших таблицях стилів, зменшуючи помилки та підтримуючи єдиний візуальний стиль;

– *читабельність*: код, який використовує змінні, стає більш читабельним і зрозумілим, особливо для великих проектів зі складними стилями.

Використання змінних у розрахунках

SASS дозволяє виконувати базові математичні операції над змінними. Це дає можливість динамічно розраховувати значення у ваших стилях. Ось деякі підтримувані оператори:

- `+`: Додавання;
- `-`: Віднімання;
- `*`: Множення;
- `/`: Ділення.

Приклад:

```
$base-font-size: 16px; /* Розмір базового шрифту */
$heading-size: $base-font-size * 1.5; /* Розмір заголовків */

h1 {
  font-size: $heading-size;
}
```

Тут ми розраховуємо розмір шрифту для заголовків (`h1`), помножуючи змінну розміру базового шрифту на 1.5.

Також варто знати:

– за замовчуванням змінні мають локальну область дії. Ви можете визначити їх у межах певного блоку стилів або зробити їх глобальними за допомогою міксинів або спеціальної директиви ***global***;

– SASS підтримує різні типи даних для змінних, включаючи рядки, числа, логічні значення, списки та карти (більш просунуті).

SASS Mixin

Міксини SASS - це потужна функціональність, яка дозволяє створювати багаторазово використовувані блоки CSS-властивостей. Що дозволяє економить час, робить код більш організованим та спрощує внесення змін до стилів.

Синтаксис міксинів

Міксини визначаються за допомогою ключового слова `@mixin` за яким йде ім'я міксину та опціональні аргументи. Далі йдуть CSS-властивості, які складають блок міксину.

```
@mixin mixin-name($arg1, $arg2...) {
    /* CSS-властивості */
}
```

Використання міксинів

Для використання міксину, необхідно викликати його прописавши: `@include` ім'я міксину та аргументи (якщо вони були визначеними) в будь-якому місці вашого CSS-коду.

```
@include mixin-name($value1, $value2...);
```

Приклад:

```
@mixin button-style($color, $border-width) {
    background-color: $color;
    border: $border-width solid #ccc;
    padding: 10px 20px;
    text-decoration: none;
    cursor: pointer;
  }

.primary-button {
    @include button-style(#007bff, 2px);
}

.secondary-button {
    @include button-style(#6c757d, 1px);
}
```

У цьому прикладі ми створили міксин `button-style`, який приймає два аргументи: `$color` та `$border-width`. Міксин визначає стилі для кнопки, включаючи фоновий колір, рамку, відступ та курсор.

Потім ми використовуємо міксин двічі, щоб створити дві кнопки з різними стилями. Перша кнопка (`primary-button`) буде мати синій фон і товсту рамку, а друга кнопка (`secondary-button`) - сірий фон і тонку рамку.

Переваги використання міксинів:

- *повторне використання*: міксини дозволяють створювати та повторно використовувати блоки CSS-властивостей, роблячи ваш код більш компактним та організованим;
- *зменшення дублювання*: замість повторного написання однакових стилів у різних місцях, ви можете використовувати міксин, щоб визначити їх один раз і використовувати де завгодно;
- *легкі зміни*: якщо вам потрібно змінити стилі в декількох місцях, вам просто потрібно оновити визначення міксину, і всі екземпляри, які його використовують, автоматично оновляться.

Ви можете визначити значеннями по замовчуванню для аргументів міксину, що робить його більш гнучким.

```
@mixin button-style($color: #007bff, $border-width: 2px) {
    background-color: $color;
    border: $border-width solid #ccc;
    padding: 10px 20px;
    text-decoration: none;
    cursor: pointer;
}
```

У цьому прикладі міксин `button-style` має два аргументи: `$color` та `$border-width`. `$color` має по замовчуванню значення `#007bff`, а `$border-width` - `2px`.

Міксини можуть включати інші міксини, передаючи їм аргументи. Це дозволяє створювати складні та ієрархічні стилі.

```
@mixin primary-button {
    @include button-style(#007bff);
}

@mixin secondary-button {
    @include button-style(#6c757d, 1px);
}
```

У цьому прикладі міксини `primary-button` та `secondary-button` просто включають міксин `button-style` з відповідними значеннями аргументів.

Нестингова вкладеність в SASS

Нестингова вкладеність (nested indentation) - це зручний спосіб організації CSS-стилів в SASS. Вона дозволяє вам використовувати відступи для визначення вкладених селекторів, що створює візуально ієрархічну структуру вашого коду.

Переваги використання нестингової вкладеності:

- *покращена читабельність*: відступи допомагають візуально відокремити стилі, пов'язані з певним елементом, покращуючи читабельність коду, особливо для складних селекторів;
- *організація коду*: вкладеність робить ваш код більш організованим, оскільки стилі для дочірніх елементів чітко відступають від батьківських селекторів;
- *логічна структура*: структура відступів відображає ієрархію елементів на сторінці, що полегшує розуміння того, як стилі застосовуються до вкладених елементів.

Синтаксис нестингової вкладеності

Нестингова вкладеність працює шляхом відступу селекторів всередині інших селекторів. Кожен рівень вкладеності повинен мати додатковий рівень відступу.

```
nav {
    ul {
      margin: 0;
      padding: 0;
      list-style: none;
    }

    li {
      display: inline-block;
    }

    a {
      display: block;
      padding: 6px 12px;
      text-decoration: none;
    }
}
```

В цьому прикладі стилі для елементів `<ul>`, `<li>` та `<a>` вкладені всередину стилів для елемента `<nav>`. Кожен рівень вкладеності відсунутий на один TAB.

Застереження щодо нестингової вкладеності

– *складність при глибокій вкладеності*: при дуже глибокій вкладеності код може стати громіздким та важко читабельним;

– *специфічність селекторів*: нестингова вкладеність може випадково збільшувати специфічність селекторів, що може призвести до проблем із перезаписом стилів.

Рекомендації щодо використання

– використовуйте нестингову вкладеність для логічної організації стилів для дочірніх і онуків елементів;

– уникайте надмірної вкладеності, яка може ускладнити код.

Лабораторна робота №7

Тема: «Адаптивна розмітка»

Мета: отримати навички роботи з медіа запитами.

Звіт повинен містити:

1. Постановка завдання;
2. Теоретичні відомості по проекту;
3. Результати виконання;
4. Висновок.

Завдання

1. Додати адаптивність до розробленого сайту портфоліо:

1.1 Створити media.css файл, підключити його до html-сторінок, скопіювати в нього CSS текст нижче та заповнити його відповідними правилами адаптації.

2. Опублікувати адаптований веб-сайт в мережі Інтернет.
3. Провести тестування на мобільних пристоях.
4. Описати хід роботи.
5. Зробити висновки.

Файл *media.css*

```
@media (max-width: 1140px) { /*для екранів з шириною до 1140px*/
    .card {
        width: calc(50% - 15px);
    }
    .footer {
    }
}

@media (max-width: 760px) { /* Для екранів з шириною до 760px*/

}
```

Теоретичні відомості

Media queries

Медіа-запити (media queries) – це правила CSS, які дозволяють керувати стилями елементів залежно від значень технічних параметрів пристроїв.

Нижче на рисунку 27, преведена таблиця з версіями браузерів, починаючи з яких вони почали підтримувати медіа-запити.

Internet Explorer	Chrome	Opera	Safari	Firefox	Android	iOS
✔ 9.0+	✔ 1.0+	✔ 10.0+	✔ 4.0+	✔ 3.6+	✔ 2.0+	✔ 2.0+

Рисунок 27 – Таблиця версій браузерів

Всі запити починаються з ключового слова **@media**, після чого слідує **умова**, в якій використовуються:

- типи носіїв;
- логічні оператори;
- медіа-функції.

Типи носіїв

У CSS медіа-запити використовуються для визначення типу пристрою, на якому відображається веб-сторінка, та застосування різних стилів CSS залежно від цього типу (таблиця 3).

Приклад використання

```
@media screen, projection, tv {
     body { font-size: 16px; }
     h1 { font-size: 24px; }
}
```

Таблиця 3 – Відповідність типу пристрою та стилів CSS

№	Тип	Опис
1	**all**	Всі типи (Default)
2	**braille**	Пристрої, що базуються на системі Брайля
3	**embossed**	Принтери, які використовують для друку систему Брайля
4	**handheld**	Цей тип застарів, але все ще використовується деякими браузерами для позначення мобільних пристроїв
5	**print**	Принтери
6	**projection**	Проектор
7	**screen**	Екран монітора
8	**speech**	Синтезатори речі
9	**tty**	Телетайпи, термінали
10	**tv**	Телевізори (SMART-TV)

Логічні оператори

Логічні вирази дозволяють комбінувати декілька умов у медіа-запитах CSS, роблячи їх більш гнучкими та потужними. Ви можете використовувати логічні оператори **`and`** та **`not`** для визначення того, які умови повинні виконуватися одночасно, щоб медіа-запит застосовувався.

```
@media all and (color) { ... } /* всі кольорові пристої */

@media all and (not handheld) { ... } /* всі пристрої окрім мобільних */
```

Оператор OR реалізовано за допомогою оператора позначеного комою «,»:

```
@media screen, projection, tv { ... } /* один із пристоїв */
```

Медіа функції

Медіа функції CSS, також відомі як "медіа властивості", є ключовим компонентом медіа запитів CSS. Вони дозволяють вам описувати характеристики пристроїв, на яких відображається веб-сторінка, та застосовувати різні стилі CSS залежно від цих характеристик. Це робить CSS потужним інструментом для створення адаптивних веб-сайтів, які добре виглядають та функціонують на різних пристроях, таких як комп'ютери, планшети та смартфони.

Існує широкий спектр медіа функцій, які можна використовувати в медіа-запитах. Даний список властивостей можна представити в наступній таблиці, із зазначенням назви властивості які використовуються в CSS:

Таблиця 4 - Медіа функцій, які можна використовувати в медіа-запитах

№	Назва	Опис
1	**height**	*Перевіряє висоту області перегляду.*
2	**width**	*Перевіряє ширину області перегляду.*
3	**aspect-ratio**	*Визначає співвідношення ширини і висоти області, що відображається (16/9)*
4	**orientation**	*Визначає орієнтацію області перегляду*
5	**resolution**	*Визначає роздільну здатність пристрою*
6	**color**	*Визначає кількість бітів на канал кольору.*
7	**color-index**	*Визначає кількість кольорів, яку підтримує пристрій.*

Дані таблиці можна систематизувати за наступними розділами:

– *розмір екрана*: дозволяє перевірити ширину (***width***), висоту (***height***) або співвідношення (***aspect-ratio***) сторін екрана пристрою;

– *орієнтація екрана*: визначає, чи є екран пристрою в портретному (***portrait***) чи альбомному (***landscape***) режимі;

– *роздільна здатність екрана:* ви можете перевірити роздільну здатність (***resolution***) екрана пристрою, наприклад, HD, Retina або 4K;

– *кольорова гама:* ви можете визначити, чи підтримує екран пристрою широку (***wide***) або вузьку кольорову гаму (***color-gamut***);

– *вказівний пристрій*: дозволяє визначати, чи використовується для взаємодії з пристроєм миша, сенсорний екран або інший вказівний пристрій.

Преставки min-, та max-

Для числових властивостей можна використовувати преставки ***min-***, та ***max-*** для визначення мінімальної та максимальної величини властивості елемента, відповідно. Наприклад:

– *min-width* та *max-width;*

– *min-height* та *max-height;*

– *min-resolution* та *max-resolution;*

– *min-color* та *max-color.*

Приведемо приклад використання медіа-функцій:

```
/*Медіа запит для екранів з шириною до 1140px*/
@media (max-width: 1140px) {
    .card {
        width: calc(50% - 15px);
    }
}

/*Медіа запит для екранів з альбомною орієнтацією*/
@media screen and (orientation: landscape) {
    #logo { background: url(logo1.png) no-repeat; }
}

/*Медіа запит для екранів з портретною орієнтацією*/
@media screen and (orientation: portrait) {
    #logo { background: url(logo2.png) no-repeat; }
}
```

КОНТРОЛЬНІ ПИТАННЯ

КРИТЕРІЇ ОЦІНЮВАННЯ ДИСЦИПЛІНИ

- Оцінка «5» (*за шкалою ECTS – A*, тобто 90-100 балів) виставляється студенту, який володіє фактичним матеріалом теми, розділу, курсу. Відповідь повна, побудована послідовно, логічно і підтверджується прикладами або програмами.

- Оцінка «4» (*за шкалою ECTS – B*, тобто 85-89 балів, а також *за шкалою ECTS – C*, тобто 75-84 бали) виставляється студенту, який добре знає фактичний матеріал теми, розділу, курсу, але відповідь не достатньо повна, потребує додаткових питань викладача.

- Оцінка «3» (*за шкалою ECTS –D*, тобто 67-74 бали, а також *за шкалою ECTS – E*, тобто 60-66 бали) виставляється студенту, який володіє лише основними поняттями теми, розділу, курсу, знає основні алгоритми. Відповідає лише на деякі додаткові питання викладача.

- Оцінка «2» (*за шкалою ECTS – FX*, тобто 34-59 – це можливість повторного захисту роботи після доопрацювання, а також *за шкалою ECTS – F*, тобто 1-34 бали – з обов'язковим повторним курсом навчання) виставляється студенту, який не володіє теоретичним матеріалом теми, розділу, курсу, не знає відповідних алгоритмів, не вміє їх застосовувати.

Узагальнена система критеріїв оцінювання

<table>
<tr><td>Рейтинговий показник</td><td colspan="2">Оцінка у національній шкалі</td><td>Оцінка ECTS</td></tr>
<tr><td>90-100</td><td rowspan="4">Зараховано</td><td>5 (відмінно)</td><td>A (відмінно)</td></tr>
<tr><td>82-89</td><td rowspan="2">4 (добре)</td><td>B(добре)</td></tr>
<tr><td>75-81</td><td>C(добре)</td></tr>
<tr><td>68-74</td><td rowspan="2">3 (задовільно)</td><td>D (задовільно)</td></tr>
<tr><td>60-67</td><td>E (задовільно)</td></tr>
<tr><td>35-59</td><td rowspan="2">Незараховано</td><td>2 (незадовільно)</td><td>FX (незадовільно) з можливістю повторного</td></tr>
<tr><td>1-34</td><td>-</td><td>F (незадовільно) з обов'язковим повторним</td></tr>
</table>

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ben Frain. Responsive Web Design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques, 3rd Edition. 2020. – 410 с.
2. Молчанов В.П. Технології WEB-дизайну : конспект лекцій / В.П. Молчанов. – Харків : Вид. ХНЕУ, 2011. – 212 с.
3. Молчанов В. П. Основи проектування WEB-видань. Конспект лекцій. – Харків: Вид. ХНЕУ, 2008. – 168 с.
4. HTML Tutorial w3schools. [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/html/default.asp
5. CSS Tutorial w3schools. [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/css/default.asp
6. HTML Конструювання. [Електронний ресурс]. – Режим доступу: http://htmlbook.in.ua/
7. Середовище розробки Visual Studio Code. [Електронний ресурс] Режим доступу: https://code.visualstudio.com/