МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна технологія візуалізації даних веб-портфоліо»

здобувача групи ІН.м-25 Ляшенко Дмитра Андрійовича

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

Дмитро Ляшенко
(підпис)

Науковий керівник,
кандидат фізико-математичних наук,
асистент кафедри комп'ютерних наук Олександр Власенко

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо-наукової програми «Інформатика»
здобувача групи ІН.м-25 Ляшенко Дмитра Андрійовича

1. Тема роботи: «Інформаційна технологія візуалізації даних веб-портфоліо»
затверджую наказом по СумДУ від «06» грудня 2023 р. № 1412-VI
2. Термін здачі здобувачем кваліфікаційної роботи до 22 грудня 2023 року
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз предметної області, аналіз аналогів та поставка задачі. 2) Огляд технологій створення веб-портфоліо . 3) Прототипування проекту. 4) Вибір інструментів для реалізації завдання 5) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз предметної області, аналіз аналогів та поставка задачі.*		
2	*Огляд технологій створення веб-портфоліо .*		
3	*Прототипування проекту*		
4	*Вибір інструментів для реалізації завдання*		
5	*Аналіз результатів.*		
4	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 64 стор., 31 рис., 3 додатки, 2 таблиці, 28 джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена інформаційним технологіям візуалізації даних які надають можливість донести до користувача інформацію оригінальним, цікавим та ефективний шляхом.

Об'єкт дослідження — інформаційні технології візуалізації даних.

Мета роботи — дослідження інформаційних технологій візуалізації даних та подальшої реалізації оформлення розділів веб-портфоліо з їх використанням

Методи дослідження — аналітичний метод дослідження, порівняння та аналогії, метод декларативного програмування, HTML, CSS, JavaScript.

Результати — в результаті кваліфікаційної роботи було досліджено інформаційні технології візуалізації даних на прикладі їх використання для оформлення розділів веб-портфоліо. Розроблено три розділи веб-портфоліо – шапка, основний контент та галерея з використанням різної візуалізації даних.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ВІЗУАЛІЗАЦІЯ ДАНИХ, ПОРТФОЛІО, PARALLAX, HTML, CSS, JAVASCRIPT

ЗМІСТ

ВСТУП

Актуальність роботи. В епоху, коли інформація панує над усіма, здатність ефективно передавати складні дані стала невід'ємним аспектом різних галузей. Оскільки цифровий ландшафт продовжує розвиватися, поєднання інформаційних технологій і веб-дизайну відкрило нові шляхи для представлення даних за допомогою візуально привабливих та інтерактивних засобів. Конвергенція цих дисциплін призвела до появи інноваційної техніки: паралакс веб-портфоліо — динамічний і захоплюючий підхід до візуалізації даних, який використовує потужність інформаційних технологій.

Дипломна робота спрямована на те, щоб заглибитися в сферу інформаційних технологій у контексті візуалізації даних через призму паралакса веб-портфоліо. А саме – дослідити складний зв'язок між технологіями та дизайном, як вони взаємодіють.

У цій кваліфікаційній роботі я досліджуватиму принципи та методології, що лежать в основі візуалізації даних, розкриваючи значення інформаційних технологій у формуванні способу представлення та розуміння даних. Крім того, дослідження заглибиться в нюанси паралакса веб-портфоліо, розбере його механізми та з'ясує потенціал у перетворенні статичних даних у динамічний, аби надати користувачам захоплюючий досвід в Інтернеті.

Крім того, аналізуючи тематичні дослідження та проводячи емпіричні дослідження, ця кваліфікаційна робота має на меті розкрити практичні наслідки та потенційні застосування використання інформаційних технологій для візуалізації даних через паралакс веб-портфоліо. Завдяки всебічному аналізу та критичній оцінці це дослідження намагається зробити внесок у розуміння того, як технологічний прогрес може революціонізувати представлення та розуміння даних у сучасних цифрових середовищах.

Зрештою, ця дипломна робота прагне дати цінну інформацію про інтеграцію інформаційних технологій і принципів веб-дизайну, пропонуючи розуміння потенціалу паралакса веб-портфоліо в зміні методів візуалізації даних і вдосконаленні стратегій візуальної комунікації в епоху цифрових технологій.

Метою кваліфікаційної роботи є дослідження інформаційних технологій візуалізації даних та подальшої реалізації оформлення розділів веб-портфоліо з їх використанням.

1.ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз предметної області

Основна мета створення веб-портфоліо – надання інформаційних ресурсів користувачам аби задовольнити їх потреби, а саме – ознайомлення з виконавцем, його працею, тощо. Розробка таких сайтів корисна для індивідуального бренду та зазвичай охоплює різні предметні області, кожна з яких сприяє загальній ефективності та впливу портфоліо на потенційних клієнтів.

Розглянемо області які безпосередньо залучені до створення веб-портфоліо:

- Дизайн та візуальна презентація

Створення привабливого та зручного дизайну, вибір кольорів та шрифтів, візуальні елементи, тощо.

- Створення та керування вмістом

Презентація ваших робіт, проектів та продуктів користувачу у візуально привабливій та інформативній формі; біографія виконавця.

- UX та навігація

Легка для розуміння та цілісна структура яка дає змогу користувачам комфортно взаємодіяти з сайтом.

- Технічна складова

Розробка інтерфейсу за допомогою мов програмування HTML, CSS, JavaScript і фреймворків; забезпечення функціональності веб-сайту.

- Аналітика та відстеження продуктивності

Інтеграція інструментів моніторингу поведінки відвідувачів, джерел трафіку та інших показників для аналізу; подальша оптимізація продуктивності.

- Інтеграція соціальних мереж

Поширення облікових записів у портфоліо для ширшого охоплення та фідбеку.

- Технічне обслуговування та оновлення матеріалів

Регулярні оновлення портфоліо новими проектами, навичками чи досягненнями виконавця; усунення помилок та підтримка продуктивності сайту.

Візуалізація даних у веб-дизайні ґрунтується на кількох принципах та методологіях, спрямованих на зручність сприйняття інформації та взаємодію з користувачем. [3] [20]

1) **Простота та зрозумілість**: Візуалізація даних має бути простою для розуміння. Використання чітких, легко сприйнятних елементів дозволяє користувачам швидше засвоювати інформацію.
2) **Контекстуалізація**: Подання даних в контексті допомагає зрозуміти їх значення. Візуалізація повинна враховувати обставини та умови, щоб користувач міг зробити відповідні висновки.
3) **Інтерактивність**: Можливість взаємодії з візуалізацією сприяє кращому розумінню даних. Веб-дизайн може використовувати різноманітні інтерактивні елементи, такі як фільтри, підказки тощо, для полегшення сприйняття інформації.
4) **Адаптивність та відповідність**: Візуалізація даних повинна бути адаптованою до різних пристроїв та розмірів екранів. Вона також

повинна відповідати потребам аудиторії та контексту використання.

5) **Графічна ефективність**: Використання графічних елементів, таких як діаграми, графіки, картографія, дозволяє швидше сприймати та розуміти дані.
6) **Кольорова та візуальна гармонія**: Вибір правильної кольорової палітри та візуальних елементів допомагає покращити сприйняття даних і забезпечити гармонійний вигляд.
7) **Відображення важливих відношень та шаблонів**: Важливі зв'язки та шаблони у даних мають бути виділені для полегшення їх розуміння та аналізу.

Методологія візуалізації даних у веб-дизайні базується на поєднанні цих принципів, щоб створити ефективні, зрозумілі та корисні візуальні елементи для користувачів в мережі Інтернет.

Одним з найпопулярніших інформаційних технологій візуалізації даних є ефект Parallax. Він використовується для створення враження глибини або руху шляхом різниці у швидкості руху різних шарів веб-сторінки під час прокручування. Це технічне рішення, яке створює враження псевдо-тривимірності, коли фонові елементи рухаються повільніше або швидше, ніж передні плани, при прокручуванні сторінки вгору або вниз.[13][14][15]

Ось кілька важливих аспектів які треба враховувати щодо використання ефекту паралаксу:

- ✓ Створення ефекту глибини

Це дозволяє розміщувати шари різної глибини на веб-сторінці, щоб створити враження просторовості.

- ✓ Візуальна привабливість

Паралакс може зробити веб-сторінку більш привабливою та естетично привабливою завдяки рухливим елементам.

- ✓ Залучення користувача

Рухливість може привернути увагу користувачів і зробити взаємодію з веб-сайтом більш захоплюючою.

- ✓ Бути обережним з експлуатацією

Важливо збалансувати використання паралаксу, оскільки надмірне застосування може вплинути на швидкість завантаження сторінки і ускладнити її взаємодію.

- ✓ Адаптивність

При використанні ефекту паралаксу важливо враховувати адаптивність для різних пристроїв та роздільної здатності екранів.

- ✓ Технічні аспекти

Існує багато бібліотек та інструментів, які допомагають реалізувати ефект паралаксу, такі як JavaScript бібліотеки (наприклад, ScrollMagic, Rellax, або Skrollr), CSS інструменти та фреймворки.

Ефект паралаксу дуже ефектний та здатний достукатися до серця майже кожного користувача, але треба розуміти що все потрібно робити в міру, надлишковість може відштовхнути відвідувачів та принести більше негативних наслідків ніж позитивних. До всього потрібно підходити з холодною головою. [13][14][15]

1.2 Аналіз аналогів

Згідно з завданням дипломної роботи, а саме дослідженням ефекту паралакс та технологій візуалізації даних, було проведено моніторинг різноманітних сайтів на яких вони реалізовані. Розглянемо декілька з них та проведемо їх аналіз:

- Delassus group (https://delassus.com/en/) — це марокканський виробник закусочних помідорів, цитрусових, винограду, авокадо та квітів. Він управляє 3000 га та експортує до 1000 000 т/рік. У ньому працює 6200 відданих співробітників. Крім головного слайдера, більшістю анімацій керує Gsap3. На додаток до Gsap вони використовують бібліотеку керування паралаксом, яку розробили власними силами. Приклад показано на рисунках 1.1-1.2.

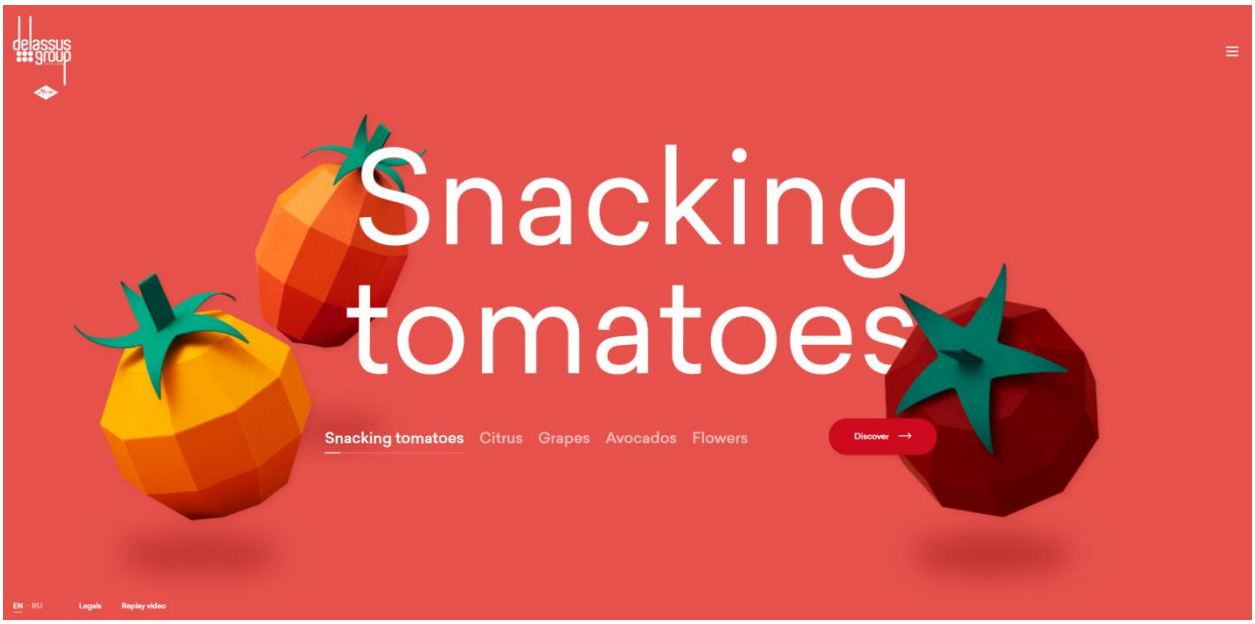


Рисунок 1.1 – Головна сторінка веб-сайту

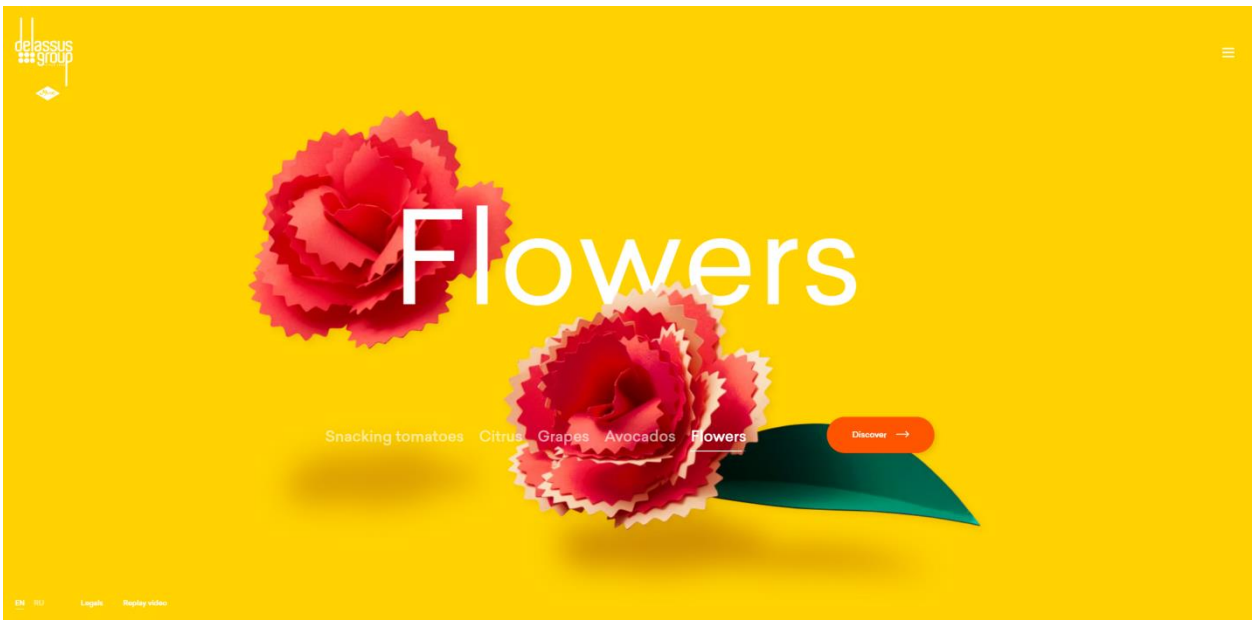


Рисунок 1.2 – Приклад вкладки «Flowers»

❖ На сайті Grab & Go (https://grabandgo.pt/#home) параллакс-ефект використовується в якості прикраси та оживлення картинок. При русі курсора можна спостерігати, як одночасно переміщаються будинки і дерева на задньому фоні. Начебто дрібниця, але за рахунок цього дизайнерського рішення є можливість що користувачам сайт запам’ятається краще. Приклад на рисунках 1.3-1.5.

Рисунок 1.3 – Головна сторінка «Grab & Go»

Рисунок 1.4 – Приклад вкладки «Productos»

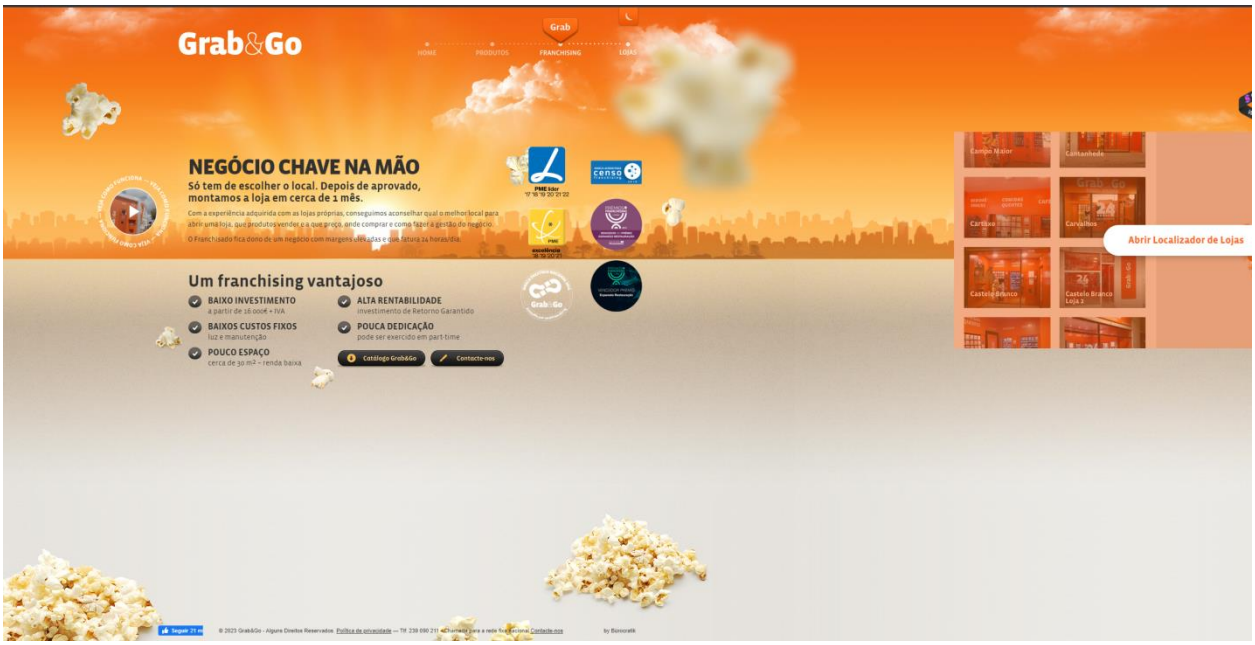
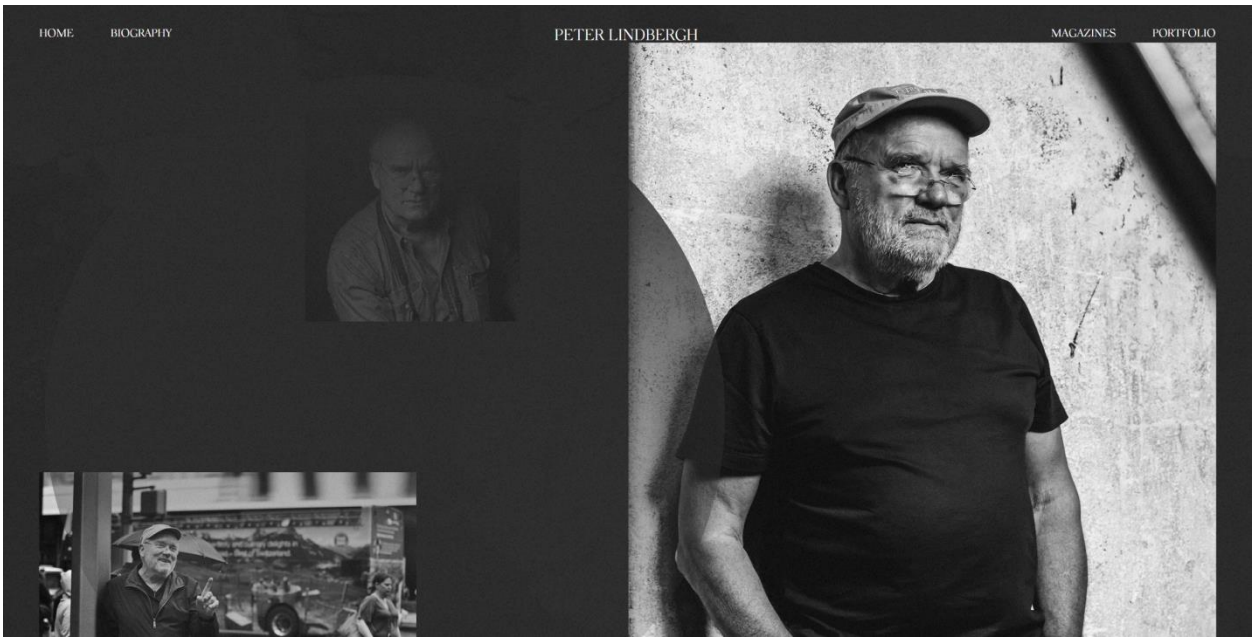
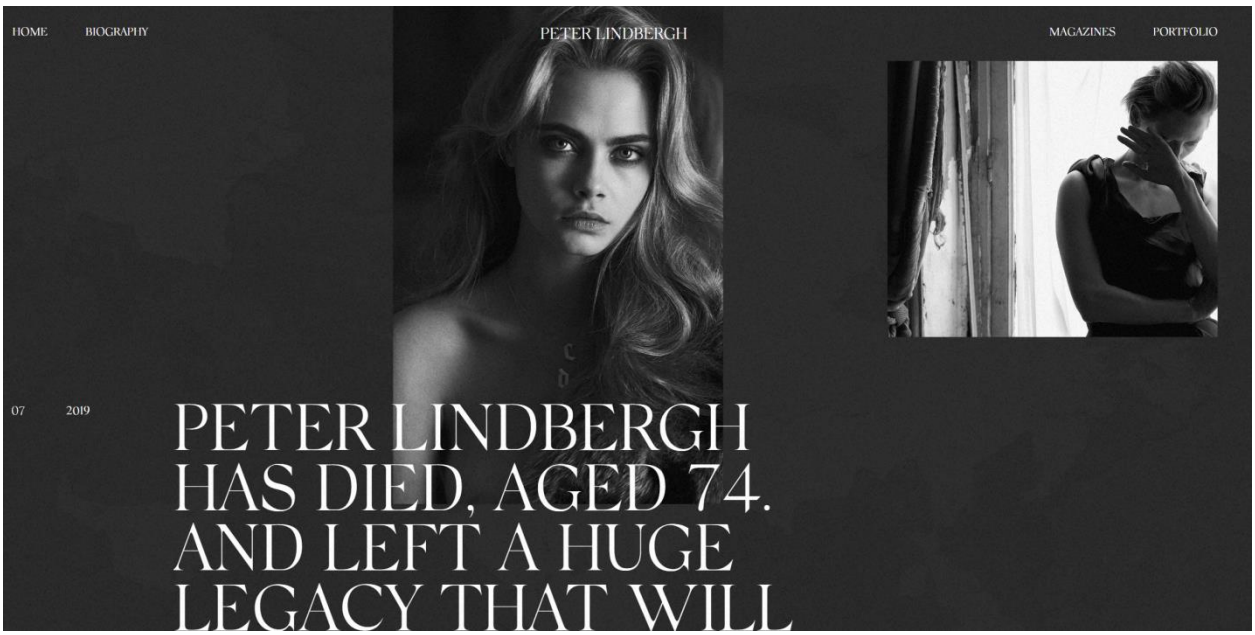


Рисунок 1.5 – Приклад вкладки «Franchising»

❖ Пітер Ліндберг (https://peterlindbergh.obys.agency/) – відомий фен-фотограф, який мав великий вплив у світі фешн-фотографії та був відомий своїм особливим стилем, який підкреслював природну красу та мінімальну ретуш. Сайт, який було створено в його честь,

виділяється з-поміж інших використанням різноманітних ефектів паралаксу. Приклад показано на рисунках 1.6-1.8.





Рисунки 1.6-1.8 – Сайт-портфоліо Пітера Ліндберга

1.3 Постановка задачі

Кваліфікаційна робота спрямована на вивчення документації, розробку та впровадження технології паралаксу та візуалізації даних у веб-розробці, метою яких є створення цікавого та привабливого веб-портфоліо.

Реалізація включає наступні задачі:

1. Огляд літератури та існуючих інформаційних технологій
    a. Провести огляд доступних технологій паралаксу та інструментів для візуалізації даних у веб-розробці.
    b. Проаналізувати приклади веб-сайтів з використанням технології паралакс та візуалізації даних у веб-портфоліо.
2. Проектування та розробка веб-портфоліо.
    a. Визначення структури.
    b. Розробка дизайну який використовує техніку паралакс та візуалізацію даних для кращого представлення інформації.
3. Імплементація
    a. Реалізація веб-портфоліо з подальшою інтеграцією ефекту паралакса та інструментів візуалізації даних.
4. Тестування
    a. Проведення тестування веб-портфоліо на адаптивність, перевірка його працездатності та коректності роботи.
    b. Демонстрація іншим людям з метою оцінки візуалізації даних через ефект паралаксу, зацікавленості.
5. Аналіз отриманого результату та висновки
    a. Зіставлення результатів з поставленими цілями.
    b. Формування висновків.

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАВДАНЬ

2.1 Вибір методології та мов програмування

Веб-програмування - це процес створення програмного забезпечення для роботи в Інтернеті або на веб-серверах. Він включає в себе розробку веб-сайтів, веб-додатків та інших програм, які працюють через браузери користувачів.

До основних складових веб-програмування відносять: мови програмування, фреймворки, мову розмітки та стилізації (HTML/CSS), бази даних, протоколи, розробку бек- та фронтенд частин, тестування. [20]

Для вирішення поставленої задачі необхідно опрацювати документацію та вибрати методологію програмування для створення продукту.

Існує кілька популярних методологій програмування, кожна з яких має свої унікальні особливості та підходи до розробки програмного забезпечення. Зробимо порівняльну характеристику для кожного з обраних, розберемося на чому саме фокусується та чи інша методологія, визначимо сильні та слабкі сторони.

Структурне програмування забезпечує фокус на підходах до побудови програми через послідовність інструкцій, уникання безумовних переходів. Це парадигма програмування, що базується на використанні послідовності ітерацій, умовних операторів та підпрограм для вирішення задачі. Його основна ідея полягає в тому, щоб розбити складні задачі на менші, більш управляються блоки або модулі. Має гарну структурованість коду, полегшення зрозуміння та управління програмами, але є складним у розумінні та використанні з великими програмами. Хоча структурне програмування є важливим етапом у розвитку програмування, багато сучасних мов програмування також підтримують інші парадигми, такі як

об'єктно-орієнтоване програмування та функціональне програмування, які надають додаткові засоби для створення програм.

Об'єктно-орієнтоване програмування представляє собою розгляд коду як набір об’єктів з властивостями та методами. Основними принципами ООП є: інкапсуляція, спадкування, класи та об’єкти, поліморфізм. Клас визначає структуру або шаблон для створення об’єктів, визначаючи їх характеристики та поведінку; об’єкт – екземпляр класу. За рахунок інкапсуляції можна обмежувати доступ та надавати його лише за необхідності. Спадкування дозволяє створювати ієрархії класів та узагальнити спільний функціонал. Поліморфізм надає можливість двояко реагувати на однакові повідомлення. Також, в ООП дуже цінується його реюзабельність за потреби, хоч і на вивчення та проектування може знадобитися більше часу, через що в деяких випадках не є раціональним рішенням.[18][19]

Функціональне програмування – це парадигма програмування, яка спрямована на створення програм за допомогою функцій. Основний акцент робиться на використанні функцій як основного засобу створення програм та обробки даних. Рекурсія, імутабельність даних, чисті функції, тощо – все це сприяє створенню більш безпечних, зручних для тестування та масштабованих програм, спрощує багатопотокові операції та полегшує розподілення роботи між багатьма процесами. Деякі мови програмування, такі як Haskell, Lisp, Scala, Clojure та інші, активно підтримують функціональну парадигму. Більшість сучасних мов програмування також мають підтримку функціональних концепцій.

Процедурне програмування – це парадигма програмування, в якій програма структурована за допомогою викликів процедур (або підпрограм). Основні відомості та код у процедурному програмуванні організовані через послідовність інструкцій і викликів підпрограм. Підходить для легких

завдань через свою простоту в реалізації, але для великих та складних програм – складне у використанні.

Подійно-орієнтоване програмування характеризується тим, що програма реагує на події котрі відбуваються в системі. Серед переваг можна виділити асинхронні застосування та можливість створення інтерактивних програм, але також є складним у відлагодженні та керуванні потоками подій.

Декларативне програмування представляє собою парадигму програмування, де програміст описує бажаний результат або поведінку, не визначаючи послідовність кроків для його досягнення. Основна ідея полягає в тому, щоб описати, що потрібно зробити, а не як саме це робити. Приклади декларативного програмування включають SQL (Structured Query Language) для роботи з базами даних, де ви описуєте, які дані ви хочете отримати, а не конкретні кроки для їх отримання, або CSS (Cascading Style Sheets) у веб-розробці, де ви описуєте вигляд та стиль елементів, не займаючись конкретним розташуванням кожного елементу на сторінці.[18]

Кожен з цих підходів має свої сильні та слабкі сторони. Вибір методології програмування залежить від типу проекту, його складності, вимог до продукту та власних вподобань команди розробників. Більшість сучасних проектів можуть використовувати комбінацію цих методологій, залежно від потреб проекту та специфіки завдань.

Також необхідно визначитися з мовою програмування, бібліотеками, інструментами та середовищем розробки.

Насамперед вибір мови програмування залежить від цілей, потреб та, звісно, вмінь виконавця. Найпопулярнішими варіантами розробки веб-додатків є:

1) HTML/CSS/Javascript:

a. HTML – структура веб-сторінки;
b. CSS – стилізація та вигляд сторінок;
c. JS – функціонал та динаміка сайту.

2) Javascript, фреймворки та бібліотеки
   a. React, Angular, Vue – прекрасні фреймворки для розробки веб-інтерфейсів.
   b. JQuery – старенький набір функцій, але все ще є попит.
   c. Redux + React.
3) Python, фреймворки та бібліотеки
   a. Django, Flask – фреймворки для розробки веб-додатків;
4) PHP

При виборі мови програмування для свого веб-портфоліо, важливо враховувати свої поточні навички програмування, знайомство з конкретною мовою або технологією, а також те, як це підходить для ваших цілей розробки та представлення вашої роботи. Також варто розглянути популярність та підтримку мови або фреймворку у веб-розробці, щоб забезпечити можливість подальшого розвитку та підтримки вашого портфоліо. Проаналізуємо кожен з представлених варіантів.[20]

Python пропонує кілька можливостей для розробки веб-додатків. Для створення веб-застосунків на Python зазвичай використовують фреймворки. Ось декілька найпопулярніших фреймворків для розробки веб-додатків на Python:

- Django

Django є одним з найпопулярніших та повнофункціональних веб-фреймворків Python. Він має вбудовану адміністративну панель, систему аутентифікації, ORM (Object-Relational Mapping) та інші корисні функції. Django спрощує швидку розробку веб-додатків і підходить для будь-якого розміру проекту.

- Flask

Flask є легким та розширюваним мікрофреймворком для Python. Він пропонує основні функції для розробки веб-додатків і дозволяє розробникам вибирати необхідні бібліотеки для свого проекту. Flask дає велику гнучкість у побудові додатку, але вимагає більшої ручної конфігурації порівняно з Django.

- FastAPI

FastAPI – це відносно новий веб-фреймворк, який набуває популярності завдяки своїй швидкості та простоті. Він дозволяє швидко створювати API за допомогою сучасних функцій Python і автоматичної документації.

Ці фреймворки можуть бути використані для створення різноманітних веб-додатків, від простих блогів до складних систем управління контентом чи електронних комерційних платформ.[8]

Для розробки веб-додатків на JavaScript існує безліч інструментів, бібліотек і фреймворків. JavaScript використовується для розробки клієнтської сторони веб-додатків, тобто для взаємодії з користувачем в браузері.

Ось декілька ключових технологій:

- React.js:

Це бібліотека, розроблена Facebook, яка використовується для створення інтерфейсів користувача. React використовує компонентний підхід для побудови веб-інтерфейсів, що робить його потужним і зручним для розробки SPA (односторінкові застосунки).

- Angular:

Це повноцінний фреймворк, розроблений командою Google, для розробки веб-додатків. Angular також використовує компонентний підхід і надає різноманітні інструменти для створення потужних веб-додатків.

- Vue.js:

Vue - це прогресивний JavaScript-фреймворк, який є легким у використанні та вивченні. Він дозволяє розробникам створювати високопродуктивні інтерфейси користувача та SPA.

- Node.js:

Node.js - це середовище виконання JavaScript, яке дозволяє запускати JavaScript на сервері. Воно забезпечує можливість писати серверний код на JavaScript, що робить його ідеальним для розробки повноцінних веб-додатків, включаючи серверну та клієнтську частини.

- Express.js:

Це один з найпопулярніших фреймворків для розробки серверної частини додатків на Node.js. Express дозволяє швидко створювати API та обробляти HTTP-запити.

- jQuery:

Це старіший, але все ще використовуваний набір інструментів JavaScript, який полегшує маніпулювання DOM, обробку подій та взаємодію з AJAX.[4][5]

Підведемо невеликі підсумки. JavaScript і Python - це дві популярні мови програмування, які використовуються для створення веб-додатків, проте вони відрізняються за своїми особливостями та призначенням.

JavaScript:

- Клієнтська мова:

JavaScript є мовою програмування, яка використовується переважно для розробки клієнтської частини веб-додатків. Вона працює у браузері користувача і дозволяє створювати динамічний контент, взаємодіючи з DOM (Document Object Model), а також виконувати асинхронні запити на сервер за допомогою AJAX.

- Широке використання:

JavaScript є фундаментальною мовою для фронтенд-розробки. Вона використовується для розробки веб-інтерфейсів, однак також може використовуватися у бекенді (за допомогою Node.js).

Python:

- Універсальність та чистота коду:

Python є загальним мовою програмування, яка використовується для різних цілей, таких як веб-розробка, штучний інтелект, наукові обчислення тощо. Він відомий своєю чистотою коду та простотою читання, що робить його привабливим для початківців.

- Фреймворки для веб-розробки:

Python має кілька популярних фреймворків для веб-розробки, таких як Django та Flask. Ці фреймворки допомагають прискорити процес розробки веб-додатків, надаючи готові рішення для рутинних завдань.

Обидві мови мають свої переваги та недоліки, і вибір між ними залежить від конкретних потреб проекту, зручності для розробника та специфіки задачі. JavaScript частіше використовується для фронтенду та взаємодії з клієнтом, в той час як Python застосовується для більш широкого спектру завдань, включаючи веб-розробку та інші області програмування.[4][5][8]

Розробка веб-додатків на PHP - це опція яку часто вибирають, оскільки PHP є однією з найпоширеніших мов програмування для створення веб-додатків. Вона використовується для створення динамічних веб-сторінок та різних систем, таких як управління вмістом (CMS), електронна комерція, соціальні мережі та інші.

Для розробки веб-додатків на PHP існує безліч фреймворків та інструментів, які спрощують та прискорюють процес розробки. Ці фреймворки мають різні переваги та підходять для різних видів проектів.[22]

Ось декілька найпопулярніших фреймворків PHP:

- Laravel:

Laravel - це потужний та добре документований фреймворк PHP. Він має чистий та елегантний синтаксис, вбудовану підтримку аутентифікації, маршрутизацію, ORM (Object-Relational Mapping) та багато інших корисних функцій. Laravel є дуже популярним серед розробників.

- Symfony:

Symfony - це високоякісний PHP-фреймворк, який дозволяє будувати різноманітні додатки, включаючи веб-сайти, веб-додатки, API і т.д. Він має велику кількість компонентів та бібліотек, що дозволяє розробникам вибирати тільки необхідні частини.

- CodeIgniter:

CodeIgniter - це легкий та простий у використанні фреймворк PHP, який підходить для швидкого створення веб-додатків. Він має мінімальну конфігурацію та вимоги до сервера, що робить його відмінним вибором для початківців.

- Yii:

Yii - це швидкий, ефективний і високопродуктивний фреймворк для розробки веб-додатків. Він має вбудовану підтримку кешування, безпеки та можливості розширення.

HTML – це основна мова кодування, яка створює і організовує веб-контент, щоб була можливість відображати його у браузері. Основні аспекти цієї мови:

1) HTML є стандартною мовою розмітки для веб-сторінок
2) HTML-елементи є будівельними блоками з яких складається HTML-сторінка
3) HTML-елементи представлені тегами < >
4) Мова не чутлива до регістру
5) Надає більш гнучкий спосіб створення веб-сторінок з текстом
6) Можливість відображати HTML-документи на будь-яких платформах, таких як Windows, Linux, Macintosh тощо.

HTML код забезпечує правильне форматування тексту і зображень для веб-браузеру. Ця мова розмітки створює базову структуру сторінки, на яку накладаються каскадні таблиці стилів, щоб змінити її зовнішній вигляд. Можна сказати, що HTML – це каркас веб-сторінки (структура), а CSS – її оболонка (зовнішній вигляд).

Каскадні таблиці стилів, або CSS, – це процес розробки, який використовується для того, щоб зробити веб-сторінку більш представницькою. Ця мова дозволяє встановлювати стилі для налаштувань вигляду сторінки у браузері, не залежачи від HTML-способу створення сторінки. Основна відмінність між мовою розмітки гіпертексту і каскадними таблицями стилів полягає в тому, що перше, як відомо, в основному забезпечує структурний спосіб ландшафту веб-сторінки, в той час як другий призначений для надання потужних методів колірного кодування і стилізації. CSS дає можливість вносити значні зміни в веб-макет всіх сторінок на одному веб-сайті, використовуючи тільки один файл. Це допомагає в

розробці якісного та креативного веб-сайту, який вражає аудиторію та привертає увагу. Таким чином, на сьогоднішній день це невід'ємна частина створення веб-сайтів, якою не слід нехтувати. [2]

Зважаючи на поставлене завдання, було обрано наступні мови програмування, а саме: HTML для розмітки сторінки, CSS – стилі, та JavaScript для динаміки.

Для реалізації нашого продукту обрано декларативну методологію програмування, адже саме вона представляє собою основу для нашого HTML, налаштування стилів CSS та взаємодію з динамікою JS.

2.2 Вибір графічного редактора та середовища розробки

Для обробки зображень використовувався Adobe Photoshop – багатофункціональний растровий графічний редактор, що розробляється та розповсюджується компанією Adobe Systems. Здебільшого працює з растровими зображеннями, проте має деякі векторні інструменти. Продукт є лідером ринку в галузі комерційних засобів редагування растрових зображень та найвідомішою програмою розробника.

Основні функції Adobe Photoshop включають:

- Редакція зображень:

Photoshop надає широкий спектр інструментів для ретушірування та редагування фотографій та зображень. Це включає роботу з кольором, яскравістю, контрастом, обрізку, розмиття, реставрацію, зміну розміру та багато іншого.

- Створення графічних елементів:

За допомогою Photoshop можна створювати графіку для веб-сайтів, рекламних матеріалів, логотипів, ілюстрацій та інших графічних елементів.

- Робота з шарами:

Photoshop працює з концепцією шарів, що дозволяє створювати складні композиції, розташовувати різні елементи окремо, накладати текстури, ефекти та фільтри.

- Робота з текстом:

В програмі є багато інструментів для роботи з текстом, таких як вибір шрифтів, налаштування розміру, колір, стиль тексту, а також можливість розміщення тексту на зображенні.

- Маски та фільтри:

Photoshop дозволяє застосовувати маски для точного контролю над частинами зображення. Також вона має різноманітні фільтри для створення різних ефектів та зміни вигляду зображення.

Photoshop є потужним інструментом для графічного дизайну, фотографії та роботи з графікою у всіх сферах, від творчих до професійних завдань. [12]

Середовищем розробки було обрано Visual Studio Code (VS Code) – це потужний та популярний безкоштовний редактор коду, розроблений компанією Microsoft. Це крос-платформений інструмент, який підтримує роботу на Windows, macOS і Linux. Ось кілька ключових особливостей середовища розробки VS Code:

- Розширюваність і плагіни:

VS Code надзвичайно гнучкий та розширюваний завдяки широкому спектру доступних плагінів. Редактор підтримує багато мов програмування, фреймворків і інструментів, які можна легко додати та налаштувати за допомогою плагінів.

- Швидкість та продуктивність:

VS Code працює досить швидко, має мінімальний споживання ресурсів системи та добре оптимізований для роботи з великими проектами.

- Інтеграція з Git і іншими інструментами розробки:

VS Code має вбудовану підтримку систем контролю версій, таких як Git. Також він інтегрується з різноманітними інструментами для розробки, що полегшує роботу з кодом та його відлагодження.

- Автоматичне завершення коду та підказки:

Редактор надає можливості автодоповнення коду, інтелектуальні підказки, вбудовану допомогу при написанні коду, що допомагає у швидкому та точному розробленні програм.

- Конфігурація та налаштування:

VS Code дозволяє розробникам налаштовувати інтерфейс, теми, клавіатурні скорочення та розширювати його функціональність через різні налаштування.

- Відлагодження та дебагінг:

Інтегроване середовище для відлагодження коду дозволяє вам встановлювати точки зупинки, аналізувати змінні та відстежувати виконання програми під час розробки.

VS Code - це популярний вибір серед розробників завдяки своїй потужності, гнучкості та багатофункціональності. Він надає розробникам інструменти для продуктивної роботи з будь-якими типами проектів та мовами програмування.[14]

Для швидкості та зручності написання проекту було задіяно Emmet –це потужний інструмент для швидкого написання HTML та CSS коду. Він є плагіном для багатьох редакторів коду, таких як Visual Studio Code, Sublime Text, Atom і інших, і дозволяє розробникам значно прискорити процес написання HTML та CSS.

Основні можливості Emmet:

- Скорочення коду:

Emmet використовує абревіатури для швидкого написання коду. Наприклад, ви можете використовувати html:5, щоб автоматично створити базову структуру HTML5 сторінки, або ul>li*10, щоб створити список з десятьма елементами <li>.

- Зручне генерування HTML та CSS:

Emmet дозволяє вбудовувати класи, ідентифікатори, атрибути, вкладені елементи та інші конструкції швидше за допомогою коротких синтаксичних правил.[10]

- Сприяє продуктивності:

За допомогою Emmet ви можете значно прискорити процес написання HTML та CSS, зменшуючи кількість введеного коду та час на написання повторюваних конструкцій.

Наприклад, команда div.container>ul#list>li.item$*10 з Emmet перетвориться на наступний код HTML:

Рисунок 2.1 – Вигляд виконаної команди через Emmet

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Алгоритм розробки сайту-портфоліо

До будь-якої розробки потрібно підходити серйозно та спершу скласти план роботи. Якщо в ході верстання виявиться яка-небудь помилка – це може призвести до суттєвих проблем, і чим пізніше вона виникне – тим гірше, та більше клопотів буде з її вирішенням. Гарно підготовлений план – не є рішенням всіх проблем, але він допоможе мінімізувати шанс їх виявлення.

Для створення сайту-портфоліо було сформовано наступний план дій:

Рисунок 3.1 – Алгоритм розробки

3.2 Технічне завдання

Створення технічного завдання є важливим етапом у розробці будь-якого програмного продукту. Технічне завдання (ТЗ) - це документ, який містить детальний опис вимог до продукту, його функціональність, архітектуру, технічні рішення та інші ключові аспекти.

Основним завданням роботи є демонстрація паралаксу та інших інформаційних технологій візуалізації даних при розробці веб-портфоліо. Було виділено три розділи на яких це буде реалізовано, а саме: шапка профілю, основний контент та галерея. Кожний з цих розділів має задовольняти наступним критеріям:

- Кольорова та візуальна гармонія;
- Оформлення контенту за допомогою передових інформаційних технологій візуалізації даних;
- Структурованість та логічність;
- Функціональність та динамічність;
- Адаптивність контенту;
- Мінімалістичність та зрозумілість.

Шапка профілю розробляється на тематику заповідника Асканія-Нова, особливістю оформлення її є реалізація паралаксу – 3D ефекту глибини за рахунок позиціонування.

Основний контент об’єднаний за темою путівника туристичної компанії, яка спеціалізується на визначних місцях, а саме – чудесах сучасного світу. В цьому розділі має бути реалізовано 3D анімація скрола з рухом слайдів углиб.

Розділ присвячений галереї має представляти достатню кількість контенту для ознайомлення. Тематикою обрано створення фото галереї відомого мандрівника та фотографа Марко Грассі.

3.3 Створення прототипу розділів за допомогою Axure RP

Axure RP — це програма, яка використовується для створення вайрфреймів, прототипів та специфікацій для веб-сайтів та мобільних програм. Цей інструмент популярний серед UX-дизайнерів, оскільки дозволяє створювати інтерактивні, клікабельні прототипи, які можна тестувати і представляти клієнтам.

Прототип – це попередня версія сайту, яка використовується для тестування та оцінки його дизайну та функціональності до повноцінної викладки в інтернеті. Як правило, це представлення кінцевого продукту з низькою точністю, тобто він може не мати всіх функцій чи елементів дизайну готового сайту. Прототипи є важливою частиною процесу проектування, оскільки вони дозволяють дизайнерам та розробникам тестувати та вдосконалювати користувальницький досвід, перш ніж інвестувати час та ресурси в кінцевий продукт. Наведемо етапи створення прототипів:

- ✓ Визначення мети та аудиторії:

Розуміння того, для кого призначений сайт і його основних цілей є ключовим етапом. Наприклад, якщо це портфоліо, основною метою може бути представлення робіт або послуг. Основна аудиторія - це потенційні клієнти або роботодавці.

- ✓ Створення скетчів:

Робота над концепцією сторінок за допомогою ручних малюнків або використання онлайн-інструментів для створення wireframe (макетів). Wireframe допоможе визначити розміщення елементів на сторінці без детального дизайну.

- ✓ Вибір інструментів для створення прототипу:

Використовування програм для створення прототипів, таких як Figma, Axure, Adobe XD, Sketch, або навіть простих онлайн-інструментів, які дозволяють створювати інтерактивні прототипи без необхідності писати код.

✓ Створення прототипу:

Використовуючи обрані інструменти, розмістіть елементи на сторінці згідно зі скетчами чи wireframe. Створення структури сторінок, включаючи меню, заголовки, текстовий та мультимедійний контент.

✓ Тестування та зворотний зв'язок:

Після створення прототипу треба протестувати на користувачах або отримати зворотний зв'язок від колег, друзів чи професіоналів. Це допоможе виявити можливі проблеми та поліпшити дизайн перед початком розробки.[17]

Було створено макет кожного з розділів з позначеними елементами інтерфейсу. Представимо скріншоти розділів сайту:

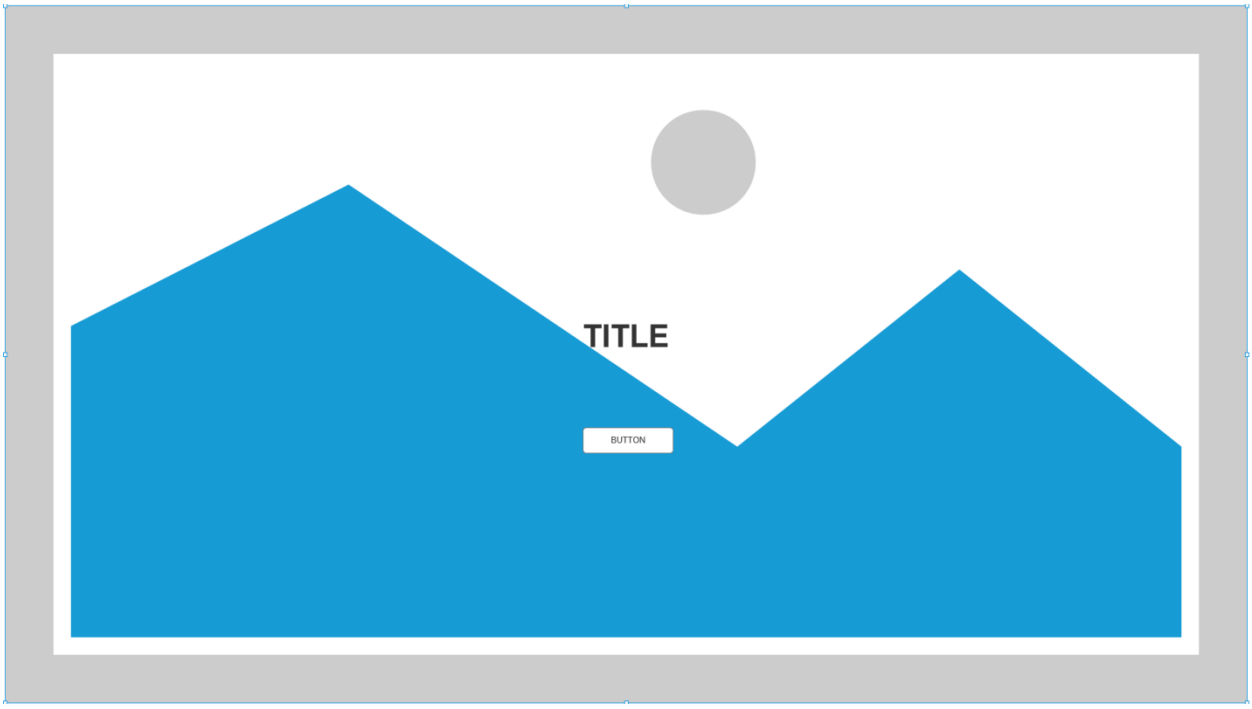


Рисунок 3.2 – Прототип шапки

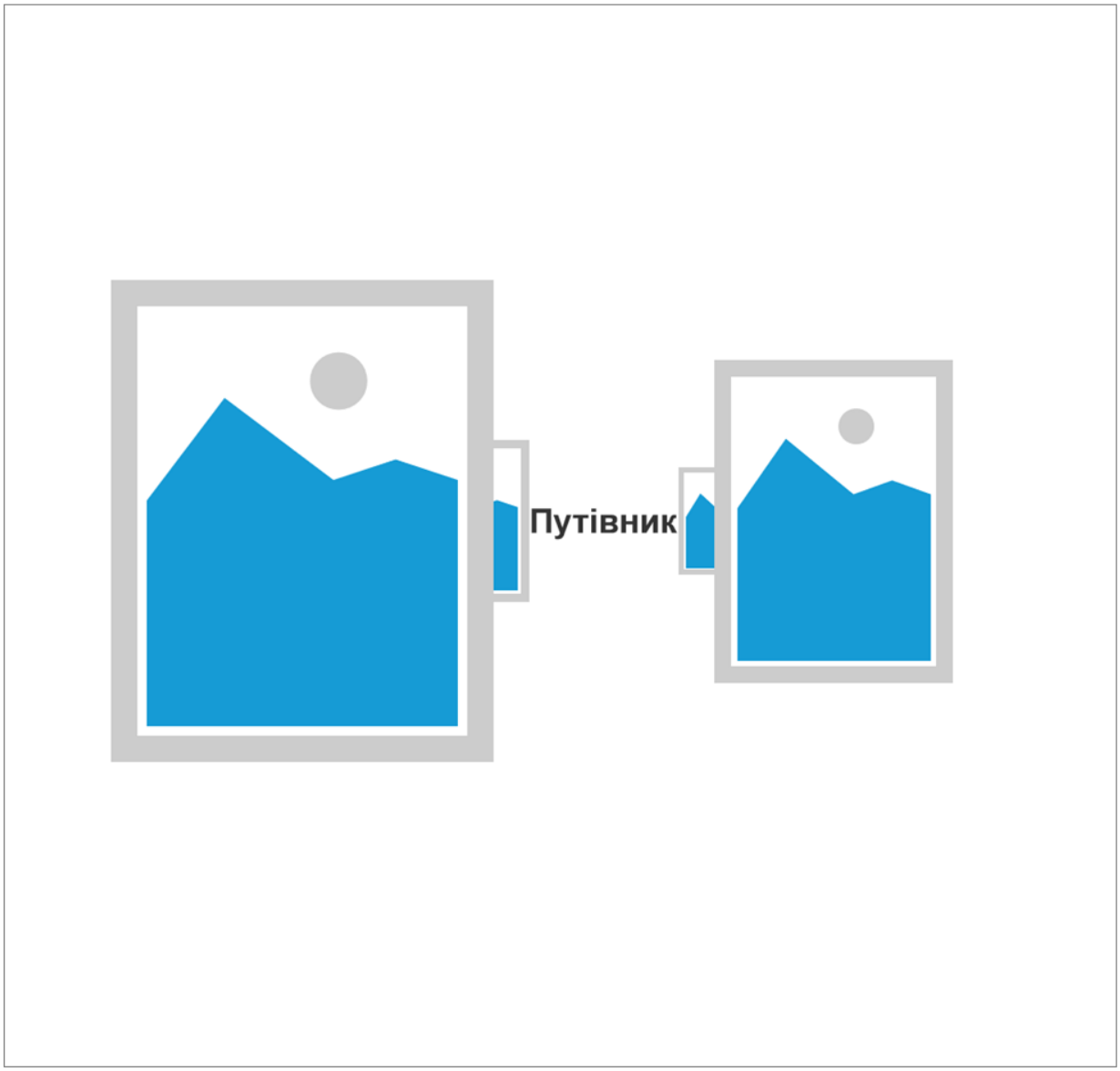


Рисунок 3.3 – Прототип основного контенту

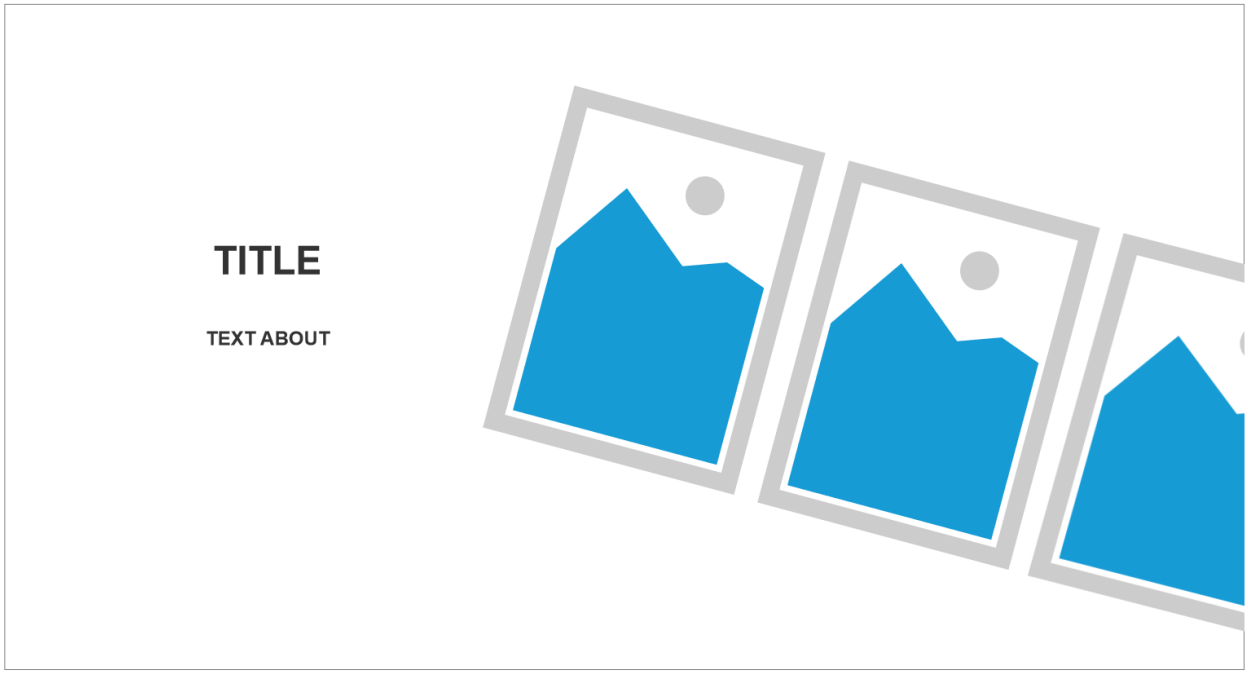


Рисунок 3.4 – Прототип галереї

3.4 Дизайн

Колірна палітра дуже важлива для створення сайту. Найголовніше, чого потрібно домагатися при підборі кольорів, – це гармонія. Гармонійне поєднання кольорів може бути одним з двох: м'яким або динамічним. Третього не дано, адже все, що за межею цих визначень, або хаотично, або монотонно.

Динамічна гармонія найчастіше спостерігається в роботах з використання яскравих, чистих кольорів без домішок і сильних контрастів. М'яка гармонія передбачає використання приглушених відтінків, м'яких переходів кольорів і розбавлення кольорів білим або чорним. Які фактори забезпечують гармонію? По-перше, ретельно підібрані комбінації кольорів; по-друге, контраст по яскравості, насиченості і відтінку.[3][20]

Можна виділити кілька важливих особливостей, що стосуються оформлення загального фону сайту і тексту. Виходячи з теорії соціальних угод фон сайт повинен бути світлим і однотипним. Бекграунд – це відмінний спосіб угруповання елементів по сенсу.

Фон повинен бути однотонним. Ні в якому разі не можна додавати квіточки або фотографії хмар в якості фонової картинки. Основна функція периферичного зору – це охорона від небезпеки. Якщо в зоні відповідальності периферичного зору находиться відволікаючий елемент, то потенційний покупець товару або послуги буде весь час забувати про мету візиту і не зможе сфокусуватися на вашій пропозиції.[20]

В розділі шапки використовувався бекграунд з різних слоїв, котрі накладалися одне на одного для візуалізації 3D ефекту.

Основний контент та галерея розроблялися на фоні відтінку чорного та тексту білого кольору:

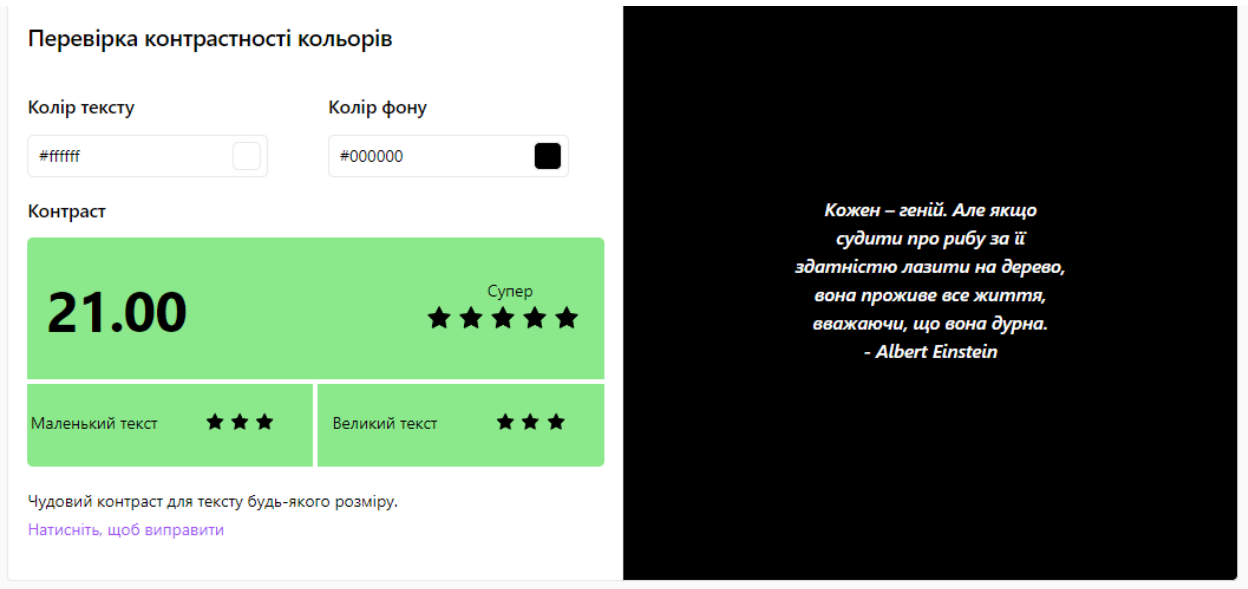
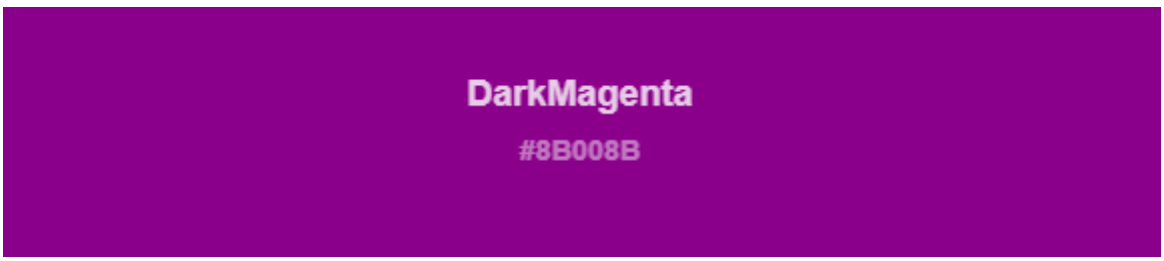


Рисунок 3.5 – Перевірка контрастності кольорів

Також, для титульного тексту в галереї був використана градієнтна заливка. Кольори які були задіяні наведено нижче:



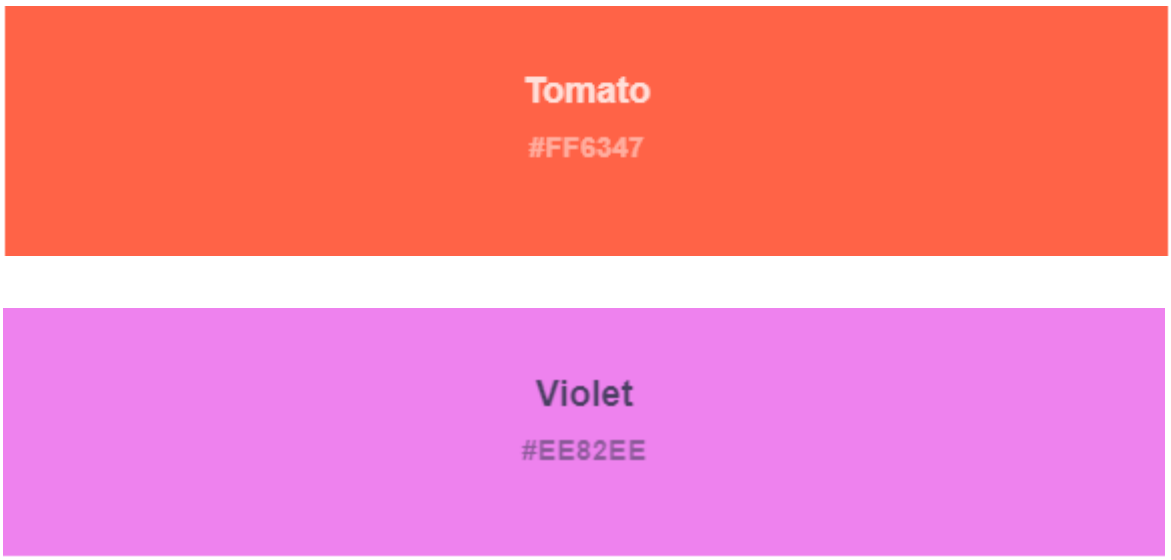


Рисунок 3.6 – Кольори градієнту

Результат міксування кольорів:

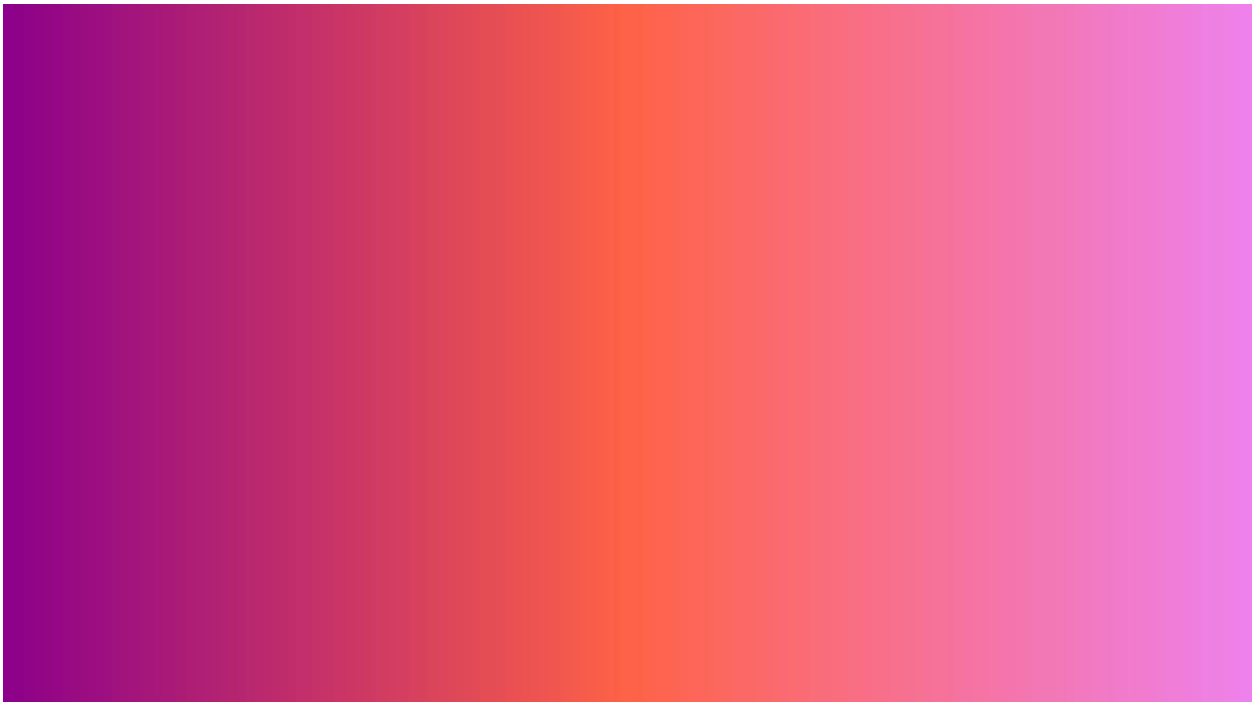


Рисунок 3.7 – Перевірка сумісності та вигляду

У медійному просторі існують свої правила оформлення тексу. У технічному оформленні не рекомендується для тексту використовувати червоний колір. Не слід також виділяти текст за допомогою Caps.

Людина легше сприймає короткі строки. Чим коротше рядок, тим з більшою охотою людина буде читати текст. Насправді короткі строки складніше для сприйняття інформації, але користувачеві здається, що короткий текст легше ніж інший, так як масиви тексту здаються менше.

Те, на чому в тексті необхідно сконцентрувати увагу, виділяють напівжирним шрифтом, але не більше трьох слів поспіль. Більш того, виділені слова повинні легко читатись і мати окреме смислове навантаження. Розмір від 12 до 18 px. Розмір шрифту на сайті повинен бути не менше 12 пікселів. Читати текст з екрану важче, ніж з паперу. До того ж у багатьох користувачів поганий зір і виходить, їм потрібно йти за окулярами, щоб розібрати рекламний текст. Тут вже не до покупки. Будь-які складності та незручності ведуть до втрати клієнтів.

Серед шрифтів, які можна знайти на комп'ютерах більшості користувачів, слід звернути увагу на дві групи: шрифти із зарубками і без зарубок, або, інакше кажучи, серіф і санс-серіф. Дві ці групи, особливо остання, і використовуються на сайтах найчастіше. Відрізняються вони, як зрозуміло з назви, наявністю або відсутністю зарубок, а при найближчому розгляді - і іншими, менш помітними з першого погляду, ознаками. [3]

Було обрано шрифти: Raleway, Kamerik205 та Merriweather.

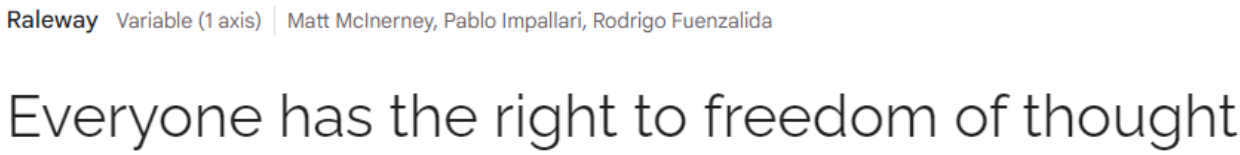


Рисунок 3.8 – Raleway

Рисунок 3.9 – Merriweather

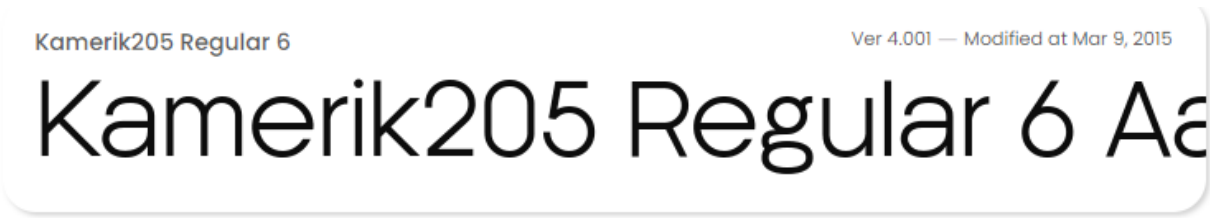


Рисунок 3.10 – Kamerik205

3.5 Обробка зображень

Підготовка зображень для проектів відбувалася в растровому графічному редакторі Adobe Photoshop – це програмне забезпечення для редагування та обробки зображень, яке розробляється компанією Adobe. Воно є одним з найпопулярніших інструментів для графічного дизайну, фотографії, ретуші, малювання та роботи зі зображеннями в цілому. За допомогою цього інструменту було нарізано слої для подальшої реалізації паралакс ефекту; «підігнали» зображення під потрібний нам формат, редагували бекграунд для шапки профілю аби надати проекту реалістичності та кращого 3D ефекту. [12]

Розглянемо процес редагування вихідного бекграунду:

Рисунок 3.11 – Вихідне зображення

Рисунок 3.12 – Деформація з подальшим розмиттям та корекцією кольору

Рисунок 3.13 – Редаговане зображення

3.6 Імплементація розділів

Основою для створення структури всіх трьох розділів є HTML(HyperText Markup Language) – це мова розмітки, що використовується для створення веб-сторінок та їх структури. Вона використовується для визначення структури, контенту та елементів на веб-сторінках, таких як текст, зображення, посилання, таблиці та інші елементи.[2]

Розглянемо на прикладі index.html першого проекту:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Project 1</title>

    <link rel="stylesheet" href="css/main.css">
    <script src="js/app.js" defer></script>

</head>
<body>

    <div class="container">
        <section class="gallery">

            <div class="frame">
                <div class="frame__content">
                    <h2>Путівник краєвидами</h2>
                </div>
            </div>

            <div class="frame">
                <div class="frame__content">
                    <div class="frame-media frame-media_left" style="background-image: url(images/1.jpg)"></div>
                </div>
            </div>

            <div class="frame frame_bg">
                <div class="frame__content">
                    <video class="frame-media frame-media_right" src="media/video1.mp4" autoplay loop muted></video>
                </div>
            </div>

            <div class="frame"></div>

            <div class="frame">
                <div class="frame__content text-right">
                    <h3>Колізей</h3>
                    <p>Амфітеатр Флавіїв або ж римський Колізей просто дивовижна пам'ятка архітектури, де за часів Римської ім
                </div>
            </div>

            <div class="frame frame_bg">
                <div class="frame__content">
                    <div class="frame-media frame-media_left" style="background-image: url(images/2.jpg)"></div>
                </div>
            </div>

            <div class="frame"></div>
```

Рисунок 3.14 – HTML код Project 1

<!DOCTYPE>	Задає тип документу, щоб браузер розумів з чим має справу
<html>	Контейнер, який включає в себе всю структуру веб-сторінки
<head>	Відображує заголовок сторінки та зберігання технічної інформації
<meta>	Визначає метатеги
<title>	Визначає заголовок документу
<link>	Зв’язок з зовнішнім документом – файлу зі стилями(main.css) чи шрифтами.
<script>	Використовується для опису скриптів, має посилання на ресурс
<body>	Цей тег призначений для зберігання контенту сторінки.
	Блоковий елемент, є контейнером для вмісту, вигляд якого надається за допомогою стилів.
<section>	Задає розділ документу.
<h1> <h2> <h3><h4>	Чотири з шести заголовків різних рівнів.
<p>	Визначає текстовий абзац.

Для визначення стилів застосовують CSS, або як їх ще називають каскадні таблиці стилів. Зазвичай його прописують як складовий елемент документу, але також є можливість використання у вигляді самостійного файлу. Саме за допомогою CSS та JavaScript реалізовувалися інформаційні технології візуалізації даних.

```css
body {
    background-color: #000;
    color: #fff;
    font-size: calc(var(--index) * .8);
    font-family: raleway_c, sans-serif;
    line-height: 1.75;
    height: var(--depth);
    font-weight: 300;
}
.container {
    width: 100%;
    height: 100%;
    position: fixed;
    perspective: 1500px;
}
.gallery {
    transform-style: preserve-3d;
    height: 100%;
}
.frame {
    width: 100%;
    height: 100%;
    position: absolute;
    display: flex;
    align-items: center;
    justify-content: center;
    transition: var(--transition), opacity .75s ease;
    will-change: transform;
    transform-style: preserve-3d;
}
h1, h2, h3, h4 {
    font-weight: 100;
    text-transform: uppercase;
    width: min-content;
    line-height: 1;
}
.frame h2 {
    text-align: center;
    font-size: calc(var(--index) * 3.3);
}
```

Рисунок 3.15 – CSS код Project 1

Принцип взаємодії заключається в тому, що CSS визначає конкретну властивість для елементу HTML. Приведемо визначення властивостям зображеним в прикладі:

background-color	Визначає колір фону для елементу
color	Визначає колір тексту.
font-size	Визначає розмір шрифта.

font-family	Встановлює сімейство шрифту
line-height	Встановлює міжрядковий інтервал тексту від базової лінії шрифту.
height	Визначає висоту елементу
font-weight	Встановлює насиченість шрифту.
width	Визначає ширину елементу
position	Визначає позиціонування елементу відносно вікна браузера чи інших елементів на сторінці.
perspective	Визначає відстань між площиною z=0 і користувачем для того щоб надати елементу що 3D позицінується ефекту перспективи.
transform-style	Визначає положення дочірнього елементу в 3D просторі або в тій же площині що й батьківський елемент.
display	Визначає вигляд елементу в документі.
align-items	Встановлює значення align-self для всіх безпосередніх дітей як групи.
justify-content	Визначає як браузеру розподіляти простір між і навколо елементів контенту по головній осі flex контейнера або рядковій grid.
transition	Універсальна властивість, яка дозволяє одночасно задати значення transition-property, transition-duration, transition-timing-function та transition-delay

will-change	Підказує браузеру яка зміна елементу очікується
text-transform	Керує перетворенням тексту елемента у великі або великі символи.
text-align	Визначає горизонтальне вирівнювання тексту у межах елементу.

Використовувався інструмент CSS Easing / Cubic-Bezier для налаштування анімацій проектів зі згладженими переходами.

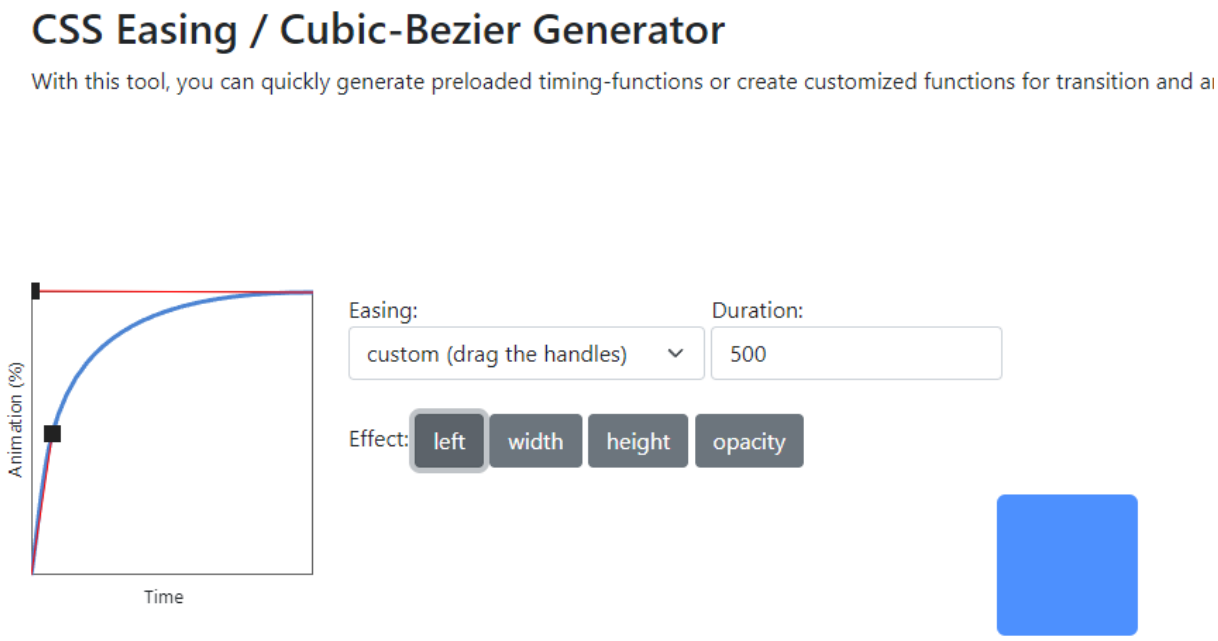


Рисунок 3.16 – CSS Easing / Cubic-Bezier

Для реалізації всіх задуманих ефектів звичайно не обійшлося без JavaScript, логіка якого була задіяна для створення динаміки. Приведемо приклад програмування 3D скролу.

```
let zSpacing = -1000,
    lastPos = zSpacing / 5,
    $frames = document.getElementsByClassName('frame'),
    frames = Array.from($frames),
    zVals = []

window.onscroll = function() {

  let top = document.documentElement.scrollTop,
      delta = lastPos - top

  lastPos = top

  frames.forEach(function(n, i) {
    zVals.push((i * zSpacing) + zSpacing)
    zVals[i] += delta * -5.5
    let frame = frames[i],
        transform = `translateZ(${zVals[i]}px)`,
        opacity = zVals[i] < Math.abs(zSpacing) / 1.8 ? 1 : 0
    frame.setAttribute('style', `transform: ${transform}; opacity: ${opacity}`)
  })

}

window.scrollTo(0, 1)
```

Рисунок 3.17 – Реалізація 3D скролу

Змінна zSpacing, як вже зрозуміло по назві, визначає відстань по осі Z. В нашому випадку це 1000 пікселів. Також реалізовано зберігання останньої позиції щоб при оновленні сторінки не скролити та шукати те місце на якому зупинилися, значення zSpacing напряму впливає на цей параметр.

Використання getElementsByClassName або getElementsById дуже важливо, щоб можна було працювати як з масивом. Усі елементи які знайде наш метод ми перетворюємо в масив. Також визначається ще один порожній масив zVals, в який ми пушимо оновлені значення по осі Z. Плавне зникнення та поява слайдів реалізовано за допомогою атрибутів transform і opacity, динамічні значення котрих визначено та передано.

Розглянемо як працює створений скрол на прикладі одного зі слайдів:

Рисунок 3.18 – Приклад слайду з Project 1

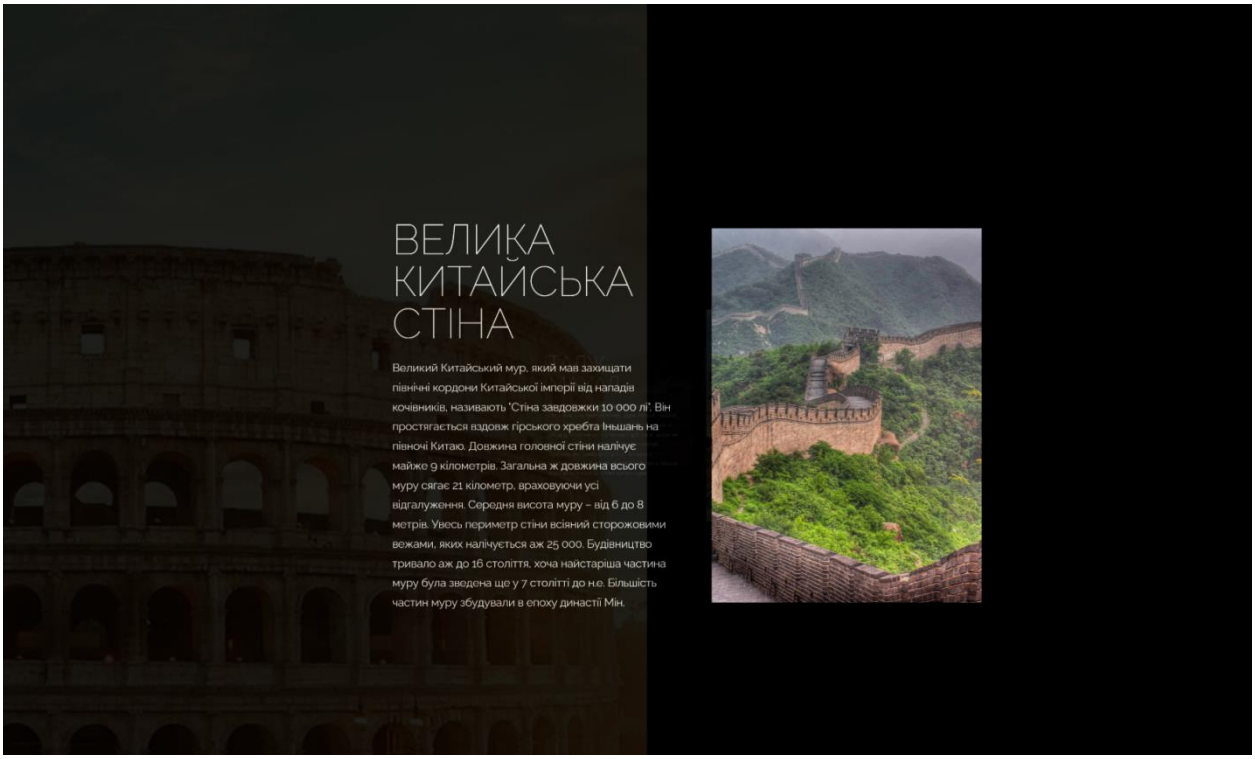


Рисунок 3.19 – Плавне зникнення та поява елементів при скролі

Сторінка не потребує написання медіа запитів для налаштування адаптивності, адже розміри були вказані у форматах vh, vw, %, тому весь контент сам підлаштовується під піксельну роздільну здатність пристрою для комфортного перегляду. Приклад зображення на iPad та iPhone SE зображено на рисунках :

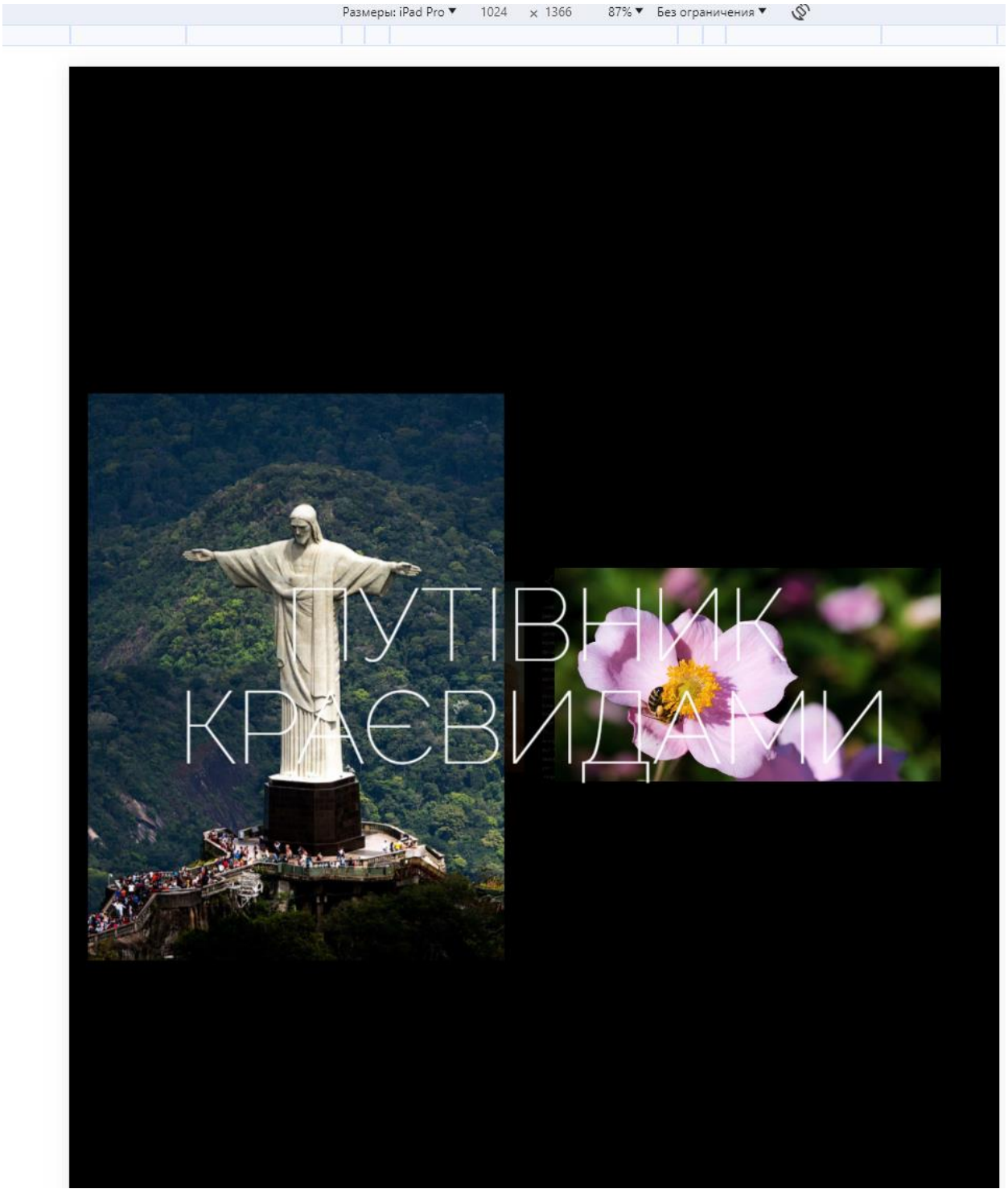


Рисунок 3.20 – Адаптивність на IPad Pro

Рисунок 3.21 – Адаптивність на iPhone SE

ВИСНОВКИ

Метою кваліфікаційної роботи було дослідження 3D ефекту Parallax та інших інформаційних технологій візуалізації даних для подальшої реалізації оформлення розділів веб-портфоліо. Для цього було виділено три основні розділи: шапка, основний контент та приклад галереї.

Реалізація відбувалася в середовищі розробки Visual Studio Code з налаштування його конфігурації для зручності та ефективності написання коду. Для підвищення ефективності був задіяний Emmet – потужний інструмент в розробці веб-додатків.

Підготовка зображень для проектів відбувалася в растровому графічному редакторі Adobe Photoshop. За допомогою цього інструменту було нарізано слої для подальшої реалізації паралакс ефекту; «підігнали» зображення під потрібний нам формат, редагували бекграунд для шапки профілю аби надати проекту реалістичності та кращого 3D ефекту.

Для того щоб отримати бажаний результат, були зроблені такі кроки: в першій частині було проаналізовано предметну область, моніторинг аналогів та постановку задачі відповідно до завдання. У другій частині проведено опис та вибір методології та мов програмування, інструментів які використовувалися впродовж розробки. В третій частині було представлено алгоритм розробки, технічне завдання та етапи програмної реалізації

В результаті кваліфікаційної роботи було досліджено інформаційні технології візуалізації даних на прикладі їх використання для оформлення розділів веб-портфоліо. Розроблено три розділи веб-портфоліо – шапка, основний контент та галерея з використанням різної візуалізації даних. Вплив 3D технологій дуже значущий, бо це можливість донести до користувача інформацію оригінальним, цікавим та ефективний шляхом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tufte, Edward R. "The Visual Display of Quantitative Information." Graphics Press, 2001.
2. Jon Duckett "HTML and CSS: Design and Build Websites", 2011
3. Cairo, Alberto. "The Functional Art: An Introduction to Information Graphics and Visualization." New Riders, 2012
4. Marijn Haverbeke "Eloquent JavaScript: A Modern Introduction to Programming", 2018
5. Nicholas C. Zakas "Professional JavaScript for Web Developers" (4th Edition), 2012
6. Few, Stephen. "Information Dashboard Design: Displaying Data for At-a-Glance Monitoring." Analytics Press, 2013
7. Steele, Julie, et al. "Beautiful Data: The Stories Behind Elegant Data Solutions." O'Reilly Media, 2009
8. Luciano Ramalho "Fluent Python: Clear, Concise, and Effective Programming", 2015
9. Krum, Randy. "Cool Infographics: Effective Communication with Data Visualization and Design." John Wiley & Sons, 2013
10. Zhenyao Mo "Mastering Emmet: Pro Tips and Tricks", 2014
11. Heer, Jeffrey, et al. "Interactive Data Visualization: Foundations, Techniques, and Applications." A K Peters/CRC Press, 2015.
12. Simon Peyton Jones "The Implementation of Functional Programming Languages", 1987
13. Scott Kelby "The Adobe Photoshop Book for Digital Photographers", 2023
14. Bruce Walton "Mastering Visual Studio Code: Build enterprise-ready, industrial-strength web applications using TypeScript, JavaScript, and modern JavaScript libraries", 2019

15. Porter, Joshua. "Designing for the Web with Parallax." Smashing Magazine, 2017

16. Harrower, Mark. "Cartography with Parallax Scrolling: Designing for Interactivity." Cartographic Perspectives, no. 77, 2014, pp. 67-72

17. Tolia, Nirav. "The Web Designer's Guide to Parallax Scrolling." A List Apart, 2012

18. Wattenberg, Martin, and Viégas, Fernanda B. "Data-Driven Documents." IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 12, 2011, pp. 2301-2309

19. Jessica Enders "Designing UX: Prototyping", 2017

20. Donald E. Knuth "The Art of Computer Programming" (Volumes 1-4A), 2015

21. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (The Gang of Four) "Design Patterns: Elements of Reusable Object-Oriented Software", 1994

22. Ware, Colin. "Information Visualization: Perception for Design." Morgan Kaufmann, 2012

23. Bertin, Jacques. "Semiology of Graphics: Diagrams, Networks, Maps." Esri Press, 2010

24. David Powers "PHP Solutions: Dynamic Web Design Made Easy" (3rd Edition), 2014

25. Kirk, Andy. "Data Visualisation: A Handbook for Data Driven Design." SAGE Publications Ltd, 2016

26. Kosara, Robert. "Storytelling: The Next Step for Visualization." IEEE Computer Society, 2016

27. Murray, Scott. "Interactive Data Visualization for the Web." O'Reilly Media, 2013

28. Wood, Keith. "The Data Visualization Handbook." Wiley, 2020.

ДОДАТКИ

Додаток А

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Project 1</title>

    <link rel="stylesheet" href="css/main.css">
    <script src="js/app.js" defer></script>

</head>
<body>

    <div class="container">
        <section class="gallery">

            <div class="frame">
                <div class="frame__content">
                    <h2>Путівник краєвидами</h2>
                </div>
            </div>

            <div class="frame">
                <div class="frame__content">
                    <div class="frame-media frame-media_left" style="background-image: url(images/1.jpg)"></div>
                </div>
            </div>

            <div class="frame frame_bg">
                <div class="frame__content">
                    <video class="frame-media frame-media_right" src="media/video1.mp4" autoplay loop muted></video>
                </div>
            </div>

            <div class="frame"></div>

            <div class="frame">
                <div class="frame__content text-right">
                    <h3>Колізей</h3>
                    <p>Амфітеатр Флавіїв або ж римський Колізей просто дивовижна пам'ятка архітектури, де за часів Римської імперії п
                </div>
            </div>

            <div class="frame frame_bg">
                <div class="frame__content">
                    <div class="frame-media frame-media_left" style="background-image: url(images/2.jpg)"></div>
                </div>
            </div>

            <div class="frame"></div>

            <div class="frame">
                <div class="frame__content text-left">
                    <h3>Велика Китайська стіна</h3>
                    <p>Великий Китайський мур, який мав захищати північні кордони Китайської імперії від нападів кочівників, називают
                </div>
            </div>


            <div class="frame frame_bg">
                <div class="frame__content">
                    <div class="frame-media frame-media_right" style="background-image: url(images/3.jpg)"></div>
                </div>
            </div>

            <div class="frame"></div>

            <div class="frame">
                <div class="frame__content text-left">
                    <h3>Тадж-Махал</h3>
                    <p>Тадж-Махал є однієї з найгарніших споруджень світу, яке знаходиться в Індії. Тадж-Махал наказав звести монголь
                </div>
            </div>

```

```
            <div class="frame frame_bg">
                <div class="frame__content">
                    <div class="frame-media frame-media_right" style="background-image: url(images/4.jpg)"></div>
                </div>
            </div>

            <div class="frame">
                <div class="frame__content">
                    <video class="frame-media frame-media_left" src="media/video2.mp4" autoplay loop muted></video>
                </div>
            </div>


            <div class="frame"></div>

            <div class="frame">
                <div class="frame__content text-left">
                    <h3>Мачу-Пікчу</h3>
                    <p>"Місто серед хмар", так ще називають Мачу-Пікчу, розташоване в Андах. Місто-символ Імперії інків зн
                </div>
            </div>

            <div class="frame frame_bg">
                <div class="frame__content">
                    <div class="frame-media frame-media_right" style="background-image: url(images/5.jpg)"></div>
                </div>
            </div>


            <div class="frame frame_bg">
                <div class="frame__content">
                    <video class="frame-media" src="media/video3.mp4" autoplay loop muted></video>
                </div>
            </div>


            <div class="frame"></div>

            <div class="frame">
                <div class="frame__content">© Dmytro Lyashenko, ІН.м-25</div>
            </div>

        </section>
    </div>

    <img class="soundbutton paused" src="images/sound.gif" alt="Alt">
    <audio class="audio" src="media/audio.mp3" loop></audio>

</body>
</html>
```

main.css

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
:root {
    --index: calc(1vw + 1vh);
    --gutter: 30px;
    --side-small: 26;
    --side-big: 36;
    --depth: 4000px;
    --transition: .75s cubic-bezier(.075, .5, 0, 1)
}


body {
    scrollbar-width: none; /* Firefox */
}
body::-webkit-scrollbar {
    display: none; /* Safari and Chrome */
}

@font-face {
    font-family: raleway_c;
    src: url(../fonts/raleway-v22-cyrillic-300.woff2);
    font-weight: 300;
}
@font-face {
    font-family: raleway_c;
    src: url(../fonts/raleway-v22-cyrillic-100.woff2);
    font-weight: 100;
}
body {
    background-color: #000;
    color: #fff;
    font-size: calc(var(--index) * .8);
    font-family: raleway_c, sans-serif;
    line-height: 1.75;
    height: var(--depth);
    font-weight: 300;
}
.container {
    width: 100%;
    height: 100%;
    position: fixed;
    perspective: 1500px;
}
.gallery {
    transform-style: preserve-3d;
    height: 100%;
}
```

```css
.frame {
  width: 100%;
  height: 100%;
  position: absolute;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: var(--transition), opacity .75s ease;
  will-change: transform;
  transform-style: preserve-3d;
}
h1, h2, h3, h4 {
  font-weight: 100;
  text-transform: uppercase;
  width: min-content;
  line-height: 1;
}
.frame h2 {
  text-align: center;
  font-size: calc(var(--index) * 3.3);
}
.frame-media {
  position: relative;
  width: calc(var(--index) * var(--side-small));
  height: calc(var(--index) * var(--side-big));
  background-position: center;
  background-size: cover;
}
.frame-media_left {
  right: calc(var(--side-small) / 2 * var(--index) + var(--gutter));
}
.frame-media_right {
  left: calc(var(--side-small) / 2 * var(--index) + var(--gutter));
}
.frame_bg {
  background-color: rgb(0 0 0 / .87);
}
video.frame-media {
  width: calc(var(--index) * var(--side-big));
  height: calc(var(--index) * var(--side-small));
}
video.frame-media_right {
  left: calc(var(--side-big) / 2 * var(--index) + var(--gutter));
}
video.frame-media_left {
  right: calc(var(--side-big) / 2 * var(--index) + var(--gutter));
}
.text-right > * {
  position: relative;
  left: 18vw;
}
.text-left > * {
  position: relative;
  right: 18vw;
}
.frame h3 {
  font-size: calc(var(--index) * 3);
}
.frame p {
  max-width: 30vw;
  margin-top: 3vh;
}
.soundbutton {
  position: fixed;
  bottom: 5vh;
  right: 5vw;
  cursor: pointer;
```

```
.frame p {
    max-width: 30vw;
    margin-top: 3vh;
}
.soundbutton {
    position: fixed;
    bottom: 5vh;
    right: 5vw;
    cursor: pointer;
    width: 24px;
    transition: .25s ease;
}
.soundbutton.paused {
    opacity: .25;
}
```

app.js

```
let zSpacing = -1000,
    lastPos = zSpacing / 5,
    $frames = document.getElementsByClassName('frame'),
    frames = Array.from($frames),
    zVals = []

window.onscroll = function() {

    let top = document.documentElement.scrollTop,
        delta = lastPos - top

    lastPos = top

    frames.forEach(function(n, i) {
        zVals.push((i * zSpacing) + zSpacing)
        zVals[i] += delta * -5.5
        let frame = frames[i],
            transform = `translateZ(${zVals[i]}px)`,
            opacity = zVals[i] < Math.abs(zSpacing) / 1.8 ? 1 : 0
        frame.setAttribute('style', `transform: ${transform}; opacity: ${opacity}`)
    })

}

window.scrollTo(0, 1)


let soundButton = document.querySelector('.soundbutton'),
    audio = document.querySelector('.audio')

soundButton.addEventListener('click', e => {
    soundButton.classList.toggle('paused')
    audio.paused ? audio.play() : audio.pause()
})

window.onfocus = function() {
    soundButton.classList.contains('paused') ? audio.pause() : audio.play()
}

window.onblur = function() {
    audio.pause()
}
```

Додаток Б

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="libs/swiper/swiper-bundle.min.css">
    <link rel="stylesheet" href="css/main.css">
    <title>Project 2</title>
</head>
<body>

    <div class="description">
        <div class="logo">Marco Grassi Gallery</div>
        <p>I am very pleased to welcome you to my gallery, I hope that you will receive the same delight from these beau
    </div>

    <div class="swiper slider slider_main">
        <div class="swiper-wrapper slier__wrapper">

            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/1.jpg);"></div>
            </div>
            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/2.jpg);"></div>
            </div>
            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/3.jpg);"></div>
            </div>
            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/4.jpg);"></div>
            </div>
            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/5.jpg);"></div>
            </div>
            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/6.jpg);"></div>
            </div>
            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/7.jpg);"></div>
            </div>
            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/8.jpg);"></div>
            </div>
            <div class="swiper-slide slider__item">
                <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/9.jpg);"></div>
            </div>

        </div>
    </div>

```

```
<div class="swiper slider slider_bg">
    <div class="swiper-wrapper slier__wrapper">

        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/1.jpg);"></div>
        </div>
        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/2.jpg);"></div>
        </div>
        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/3.jpg);"></div>
        </div>
        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/4.jpg);"></div>
        </div>
        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/5.jpg);"></div>
        </div>
        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/6.jpg);"></div>
        </div>
        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/7.jpg);"></div>
        </div>
        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="30%" style="background-image: url(images/8.jpg);"></div>
        </div>
        <div class="swiper-slide slider__item">
            <div class="slider__img" data-swiper-parallax="20%" style="background-image: url(images/9.jpg);"></div>
        </div>

    </div>
</div>

<script src="libs/swiper/swiper-bundle.min.js"></script>
<script src="js/app.js"></script>

</body>
</html>
```

main.css

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
:root {
    --dark: #070707;
    --sizeindex: calc(1vw + 1vh);
    --transition: .75s cubic-bezier(.2, .6, 0, 1);
}
@font-face {
    font-family: raleway_m;
    src: url(../fonts/raleway-v22-cyrillic-300.woff2);
    font-weight: 300;
}
@font-face {
    font-family: raleway_m;
    src: url(../fonts/raleway-v22-cyrillic-900.woff2);
    font-weight: 900;
}
body, html {
    width: 100%;
    height: 100%;
    background-color: var(--dark);
    color: #fff;
    font-family: raleway_m, sans-serif;
    overflow: hidden;
    line-height: 1.8;
}
.slider {
    height: 100%;
    transform: rotate(15deg);
    overflow: visible;
    top: 10vh;
}
.slier__wrapper {
    transition: var(--transition)!important;
    will-change: transform;
}
.slider__item {
    max-height: 75vh;
    overflow: hidden;
    cursor: pointer;
    transition: transform var(--transition), box-shadow var(--transition);
}
.slider__img {
    width: 300%;
    height: 100%;
    background-size: cover;
    position: absolute;
    left: -100%;
    transition: var(--transition)!important;
    will-change: transform;
}
.slider_bg {
    z-index: 0;
    transform: rotate(-20deg)!important;
    top: -90vh;
    left: -10vh;
    opacity: .15;
    filter: blur(120px) saturate(10);
}
```

```css
.slider_bg .slider__item {
    max-height: 100vh;
}
.slider__item.opened {
    z-index: 2;
    transform: rotate(-15deg) scale(1.45);
    box-shadow: rgb(0 0 0 / .75) 0 0 0 10000px;
}
.description {
    position: absolute;
    font-size: calc(var(--sizeindex) * .8);
    top: 20vh;
    left: 8vw;
    max-width: 24vw;
    transition: opacity var(--transition), transform var(--transition);
}
.logo {
    font-size: calc(var(--sizeindex) * 1.6);
    font-weight: 900;
    display: inline-block;
    background: linear-gradient(45deg, DarkMagenta, Tomato, Violet);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
}
.description p {
    opacity: .9;
    transition: transform var(--transition);
    transition-duration: 3s;
}
.description.hidden {
    opacity: 0;
    transform: translateY(5vh);
}
.description.hidden p {
    transform: translateY(2vh);
}

```

app.js

```
const sliderMain = new Swiper('.slider_main', {
    freeMode: true,
    centeredSlides: true,
    mousewheel: true,
    parallax: true,
    breakpoints: {
        0: {
            slidesPerView: 2.5,
            spaceBetween: 20
        },
        680: {
            slidesPerView: 3.5,
            spaceBetween: 60
        }
    }
})
const sliderBg = new Swiper('.slider_bg', {
    centeredSlides: true,
    parallax: true,
    spaceBetween: 60,
    slidesPerView: 3.5
})
sliderMain.controller.control = sliderBg

document.querySelectorAll('.slider__item').forEach(item => {
    item.addEventListener('click', event => {
        item.classList.toggle('opened')
    })
})

let desc = document.querySelector('.description')
sliderMain.on('slideChange', e => {
    sliderMain.activeIndex > 0 ? desc.classList.add('hidden') : desc.classList.remove('hidden')
})

```

Додаток В

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Project 3</title>
    <link rel="stylesheet" href="css/main.css">
    <script src="js/app.js" defer></script>
    <script src="libs/rain.js" defer></script>
</head>
<body>

    <div class="logo" style="background-image: url(img/logo.svg);"></div>

    <section class="layers">
        <div class="layers__container">
            <div class="layers__item layer-1" style="background-image: url(img/layer-1.jpg);"></div>
            <div class="layers__item layer-2">
                <div class="hero-content">
                    <h1> Welcome<span>to Ascania Nova </span></h1>
                    <div class="hero-content__p"></div>
                    <button class="button-start">Start Journey</button>
                </div>
            </div>
            <div class="layers__item layer-3">
                <canvas class="rain"></canvas>
            </div>
            <div class="layers__item layer-4" style="background-image: url(img/layer-4.png);"></div>
            <div class="layers__item layer-5" style="background-image: url(img/layer-5.png);"></div>
        </div>
    </section>

</body>
</html>
```

main.css

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
:root {
  --index: calc(1vw + 1vh);
  --transition: 1.5s cubic-bezier(.05, .5, 0, 1);
}
@font-face {
  font-family: kamerik-3d;
  src: url(../fonts/kamerik205-heavy.woff2);
  font-weight: 900;
}
@font-face {
  font-family: merriweather-italic-3d;
  src: url(../fonts/merriweather-regular-italic.woff2);
}
body {
  background-color: #000;
  color: #fff;
  font-family: kamerik-3d;
}
.logo {
  --logo-size: calc(var(--index) * 7.8);
  width: var(--logo-size);
  height: var(--logo-size);
  background-repeat: no-repeat;
  position: absolute;
  left: calc(51% - calc(var(--logo-size) / 2));
  top: calc(var(--index) * 2.8);
  z-index: 1;
}
.layers {
  perspective: 1000px;
  overflow: hidden;
}
.layers__container {
  height: 100vh;
  min-height: 500px;
  transform-style: preserve-3d;
  transform: rotateX(var(--move-y)) rotateY(var(--move-x));
  will-change: transform;
  transition: transform var(--transition);
}
.layers__item {
  position: absolute;
  inset: -5vw;
  background-size: cover;
  background-position: center;
  display: flex;
  align-items: center;
  justify-content: center;
}
.layer-1 {
  transform: translateZ(-55px) scale(1.06);
}
```

```css
.layer-2 {
    transform: translateZ(80px) scale(.88);
}
.layer-2 {
    transform: translateZ(180px) scale(.8);
}
.layer-3 {
    transform: translateZ(190px) scale(.9);
}
.layer-4 {
    transform: translateZ(300px) scale(.9);
}
.layer-5 {
    transform: translateZ(380px);
}
.hero-content {
    font-size: calc(var(--index) * 2.9);
    text-align: center;
    text-transform: uppercase;
    letter-spacing: calc(var(--index) * -.15);
    line-height: 1.35em;
    margin-top: calc(var(--index) * 5.5);
}
.hero-content span {
    display: block;
}
.hero-content__p {
    text-transform: none;
    font-family: merriweather-italic-3d;
    letter-spacing: normal;
    font-size: calc(var(--index) * .73);
    line-height: 3;
}
.button-start {
    font-family: Arial;
    font-weight: 600;
    text-transform: uppercase;
    font-size: calc(var(--index) * .71);
    letter-spacing: -.02vw;
    padding: calc(var(--index) * .7) calc(var(--index) * 1.25);
    background-color: transparent;
    color: #fff;
    border-radius: 10em;
    border: rgb(255 255 255 / .4) 3px solid;
    outline: none;
    cursor: pointer;
    margin-top: calc(var(--index) * 2.5);
}
.layer-3, .layer-4, .layer-5 {
    pointer-events: none;
}

```

app.js

```
document.addEventListener('mousemove', e => {
    Object.assign(document.documentElement, {
        style: `
        --move-x: ${(e.clientX - window.innerWidth / 2) * -.005}deg;
        --move-y: ${(e.clientY - window.innerHeight / 2) * .01}deg;
        `
    })
})

```