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

Інститут електроенергетики
(навчально-науковий інститут)
факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра
(бакалавра, магістра)

Здобувача вищої освіти Лисун Юлії Ростиславівни
(ПІБ)

академічної групи 126-21-1
(шифр)

спеціальності 126 «Інформаційні системи та технології»
(код і назва спеціальності)

спеціалізації за освітньо-професійною (освітньо-науковою) програмою
(за наявності)
«Інформаційні системи та технології»
(офіційна назва)

на тему «Роброблка front-end лендінгу будівельної компанії з використанням сучасних вебтехнологій»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Соколова Н.О.			
розділів:				

Рецензент				

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій та комп'ютерної інженерії
(повна назва)

_____ Гнатушенко В.В.
(підпис) (ініціали та прізвище)

«_____» _____2025 року

ЗАВДАННЯ

на кваліфікаційну роботу
ступеня__бакалавра____
(бакалавра, магістра)

здобувача вищої освіти__Лисун Ю.Р._____**академічної групи**___126-21-1_____
(прізвище та ініціали) (шифр)

спеціальності___126 «Інформаційні системи та технології»_____

спеціалізації за освітньою-професійною програмою _____
(за наявності)

_____«Інформаційні системи та технології»_____

на тему _«Розробка front-end лендінгу будівельної компанії з використанням сучасних___
_____вебтехнологій»_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від __05.05.25__ № __336-с

Розділ	Зміст	Термін виконання
Розділ 1. Дослідження предметної області та постановка задачі розробки системи	Огляд сучасних тенденцій веброзробки, порівняльний аналіз існуючих лендінгів різних компаній, огляд текстових редакторів для розробки та обґрунтування вибору основного інструменту, вибір технологій розробки та висновки до розділу 1 та постановка задачі розробки.	03.02.2025 – 30.04.2025
Розділ 2. Проєктні рішення	1) Пошук привабливого та сучасного макету дизайну; 2) Розробка адаптивного інтерфейсу; 3) Реалізація функціональних елементів; 4) Хостинг сторінки; 5) Забезпечення базою даних; 6) Надсилання даних у Telegram-бот; 7) Перевірка працездатності; 8) Висновки до другого розділу.	03.02.2025 – 03.06.2025

Завдання видано _____ Соколова Н.О.
(підпис керівника) (ініціали та прізвище)

Дата видачі __03.02.2025_____

Дата подання до екзаменаційної комісії_____

Прийнято до виконання _____ Лисун Ю.Р.
(підпис здобувача вищої освіти) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 94 ст., 88 рис., 6 додатків, 25 джерел.

TELEGRAM, БЕМ, SCSS, АДАПТИВ, BREAKPOINTS, JQUERY, АНІМАЦІЇ, API, ХОСТИНГ, СУБД, БАЗА ДАНИХ, PHP, AJAX.

Об’єкт дослідження: вебінтерфейс компанії для представлення послуг, інформації, а також взаємодії з користувачем.

Предмет дослідження: застосування сучасних засобів front-end розробки для створення креативного вебресурсу.

Мета роботи: дослідити сучасні підходи до створення адаптивного вебінтерфейсу та розробка на їх основі функціонального та якісного лендінгу будівельної компанії CNST.

Методи дослідження – дослідження сучасних тенденцій веброзробки; дослідження існуючих лендінгів різних компаній; дослідження популярних текстових редакторів для розробки; дослідження технологій розробки на основі сучасних тенденцій.

Програмні засоби – текстовий редактор для створення та редагування сучасних вебзастосунків і програм Visual Studio Code, крос-платформний веб-сервіс для проектування та дизайну інтерфейсів Figma, набір інструментів для оптимізації роботи веб-сайту PageSpeed Insights (онлайн-сервіс), багатоплатформний месенджер Telegram, безкоштовний веб-браузер Google Chrome.

ABSTRACT

Explanatory note: 93 pages, 88 figures, 6 appendices, 25 sources.

TELEGRAM, BEM, SCSS, ADAPTIVE, BREAKPOINTS, JQUERY, ANIMATIONS, API, HOSTING, DBMS, DATABASE, PHP, AJAX.

Object of research: the company's web interface for presenting services, information, and interacting with users.

Subject of research: the use of modern front-end development tools to create a creative web resource.

Purpose of the work: to investigate modern approaches to creating an adaptive web interface and to develop a functional and high-quality landing page for the construction company CNST based on them.

Research methods: research of modern trends in web development; research of existing landing pages of various companies; research of popular text editors for development; research of development technologies based on modern trends.

Software tools: Visual Studio Code text editor for creating and editing modern web applications and programs, cross-platform web service for interface design and development Figma, set of tools for website optimization PageSpeed Insights (online service), multi-platform messenger Telegram, free web browser Google Chrome.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

JS - JavaScript

AI(ШІ) - Artificial Intelligence(Штучний інтелект)

CSS - Cascading Style Sheets

SCSS - Sassy Cascading Style Sheets

LESS - Leaner Style Sheets

LCP - Largest Contentful Paint

FID - Fist Input Delay

CLS - Cumulative Layout Shift

ДО - Дистанційна освіта

API - Application Programming Interface

SI - Speed Index

FCP - First Contentful Paint

LCP - Largest Contentful Paint

TBT - Total Blocking Time

HTML - HyperText Markup Language

PHP - Hypertext Preprocessor

SASS - Syntactically Awesome Stylesheets

ОС - Операційна система

UI - User Interface

UX - User Experience

SVG - Scalable Vector Graphics

СУБД - Система управління базою даних

БД - База даних

CDN - Content Delivery Network

БЕМ - Блок-Елемент-Модифікатор

F.A.Q. - Frequently Asked Questions

ВСТУП

У сучасному світі, люди віддають перевагу короткому, чіткому опису, в якому виділена лише необхідна інформація для знайомства із організацією, саме це повинні врахувати організації, великі та малі бізнеси, які хочуть більш поширитись у мережі Інтернет, зрозуміло представитись майбутнім клієнтам та зацікавити їх у використанні своїх послуг. Більшості користувачів набридає переходити по різним сторінкам для вивчення необхідної інформації зацікавленої їм компанії - це стосується особливо багатосторінкових веб-сайтів.

Сторінка-лендінг, яка містить найбільш важливу інформацію про компанію, свіжі новини, пропонуючі послуги компанії, приклади продукції або проєктів, які може запропонувати компанія, керуючий склад компанії, а також необхідні контактні дані для зв'язку, реалізує сучасний підхід до розробки корпоративних систем. Основними перевагами у створенні таких лендінгів «візиток» є виставлення короткої, необхідної інформації, зручність у користуванні та привабливий зовнішній вигляд, щоб, як кажуть, «не сіпалось око» при перегляді сторінки та не було бажання покинути її.

Саме лендінги полюбляють, як користувачі, так і розробники, через їх лаконічність та конкретизацію. Окрім реалізації короткого опису організації, ще однією головною перевагою лендінгів є об'єднання статичних та інтерактивних секцій, в яких можна, як і просто прочитати конкретну інформацію, так і взаємодіяти зі сторінкою (гортати слайдери, слайди яких може містити фото або інші картки (послуга чи продукт), перехід через посилання на іншу сторінку (натиснувши на кнопку), заповнити форму та відправити дані і т.і.). Саме інтерактивність і збільшує час перегляду сторінки, тому, що з’являється зацікавленість і можна «залипнути». Але треба подбати за тим, щоб все працювало та відображалось як потрібно, без жодних багів та ляпів – користувачі можуть змінити своє враження у погану сторону не лише

про сторінку, а і про саму компанію, яка не подбала про грамотне представлення себе, бо така сторінка стає обличчям компанії.

Також лендінги вважаються для розробників самим оптимізованим варіантом представлення веб-сайту. Системі буде легше завантажити одну сторінку, замість великої кількості сторінок, а отже не потрібно буде довго чекати на завантаження. Однак, не завжди лендінги можуть бути оптимізованими, якщо розробник не подбає про це, тому швидкість завантаження та в цілому навантаження на систему буде в першу чергу залежати від розробника.

Метою роботи є дослідження сучасних підходів до створення вебресурсів, а також розробка якісного та привабливого сайту-лендінгу умовної української будівельної компанії CNST, яка на ринку існує більше 10-ти років, враховуючи популярні вебтенденції та технології. Сайт повинен бути зручним у використанні та працювати без всяких проблем.

Основним завданням кваліфікаційної роботи є:

1. огляд популярних та сучасних тенденцій веброзробки, які найбільш поширені серед розробників;
2. порівняння існуючих прикладів та підбиття підсумків, які моменти потрібно врахувати для реалізації якісного інтерфейсу;
3. реалізація привабливого та достойного адаптивного вебінтерфейсу із елементами взаємодії з користувачем.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ СИСТЕМИ

1.1 Огляд сучасних тенденцій веброзробки

Кожен розробник повинен слідкувати за розвитком та зародженням сучасних тенденцій та технологій для створення прогресивних, сучасних та якісних веб-систем. Найпоширенішими серед розробників є наступні тенденції.

1.1.1 Мобільний перший підхід та адаптивний дизайн. Mobile-first – це підхід у розробці, де пріоритет має мобільний дизайн, тобто важливим першим етапом є те, як сторінка буде виглядати на мобільних пристроях. Нині, люди віддають більш перевагу у користуванні Інтернетом через телефон, тому важливо, щоб дизайн був підлаштований під такі пристрої. Це пояснюється зручністю або неможливістю зайти на сторінку через ПК. Взагалі, дуже важливо, щоб сторінка відображалась на будь-яких пристроях – від телефонів до великих моніторів (навіть можливо телевізорів), тому адаптивний дизайн є невід'ємною частиною веб-розробки [1].

1.1.2 Компонентні фреймворки. Найпрогресивнішим інструментом, який найбільш відокремлює, що саме ця людина є front-end розробником є компонентні або front-end фреймворки. Зазвичай, розробник обирає один із 3-х: Vue, React чи Angular. Їх основною перевагою є розбиття інтерфейсу на окремі блоки – компоненти, які окремо можна перевикористати та підтримати [2, 4]. На рисунку 1.1 зображено логотипи цих фреймворків.

Рисунок 1.1 – Фреймворки JavaScript

1.1.3 Інструменти AI та Low-Code/No-Code. Немає нічого постидного у використанні інструментів ШІ у веб-розробці. Технології ШІ стрімко розвиваються, тому можна сміливо використовувати такі технології у цілях навчання та розвитку, бо вони явно пришвидчать час, навчать якісному кодуванню та дійсно допоможуть у різних труднощах [3, 1].

Також, стало дійсним цікавим фактом, що, для того, щоб створити веб-сторінку, необов'язково мати великі знання у кодуванні або взагалі їх мати - забавно, чи не так? Інструменти Low-Code/No-Code (Webflow, Bubble і т. і.) дають можливість створенню сторінок з низьким кодом або взагалі без нього. Проте, це не означає, що не потрібно далі навчатись чомусь новому та взагалі розвиватись, тому цей підхід може підійти для тих, хто хоче починати поглинатись у веб-розробку та як найшвидше почати працювати у цій сфері [5].

1.1.4 SCSS та Utility-first CSS. Каскадна таблиця стилів CSS має свої препроцесори для більш зрозумілої логіки та чистого коду. Серед них є SCSS, LESS та Stylus, проте найбільш обирають SCSS через те, що він підтримує вкладеність, створення змін та міксинів.

При розробці веб-додатків на front-end фреймворці, для зручності використання стилів використовують утилітарні фреймфорки, найпоширенішим є Tailwind CSS. Він містить готові класи, які зручно можна використати при розробці.

1.1.5 Оптимізація продуктивності та Core Web Vitals. Сервіси Google оцінюють завантаження сторінки за такими метриками, як LCP, FID, CLS і т. і. (рис. 1.2). Для отримання високих балів, розробник повинен подбати про швидкість завантаження, оптимізацію ресурсів та можливого використання кешування. Додання лінивого завантаження зображень (lazy-loading) та скорочення коду є основними способи вирішення такого роду проблем.

Рисунок 1.2 – Оцінка ефективності завантаження сторінки платформи ДО Moodle

1.1.6 Анімації з Intersection Observer. Для створення анімацій, які не будуть навантажувати систему та працювати тоді, коли потрібно, використовують Intersection Observer API. Ця бібліотека дозволяє реалізувати «живі» мікро-анімації, коли анімований елемент з’являється на полі зору, і виконує логіку асинхронно. Також цю бібліотеку можна не тільки використати для анімацій, а і, наприклад, для завантаження контенту тоді, коли це буде потрібно.

1.2 Порівняльний аналіз існуючих лендінгів різних компаній

Для того, щоб зрозуміти, на що потрібно звернути увагу при розробці майбутнього проєкту, розглянуто та проаналізовано два дуже різних між собою приклади.

Перший сайт належить агенції «VIP SMM» [7] - агенції з просування бізнесу.

На рисунку 1.3 зображено головний екран сторінки, так можна подумати, що все добре, поки не почнеш скролити униз.

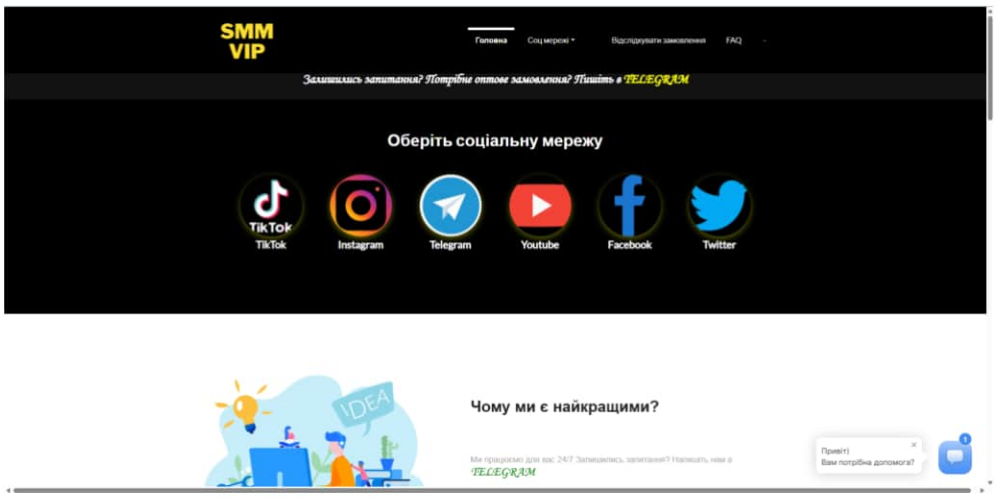


Рисунок 1.3 – Головний екран сторінки «VIP SMM»

Перше, що кидається на очі, це наявність горизонтального скроллу, якого взагалі не повинно бути, бо це дає одразу незручніть у користуванні сайтом, особливо у мобільній версії при гортанні сторінки, та і взагалі у ньому немає сенсу.

Вгорі екрану плавно з’являється меню навігації (рис. 1.4) – це добре, що воно є, для зручного перегляду сайту, але із зовнішнім виглядом все зовсім погано. Спочатку можна подумати, що воно майже пусте, але можна зрозуміти, що там є інші пункти меню, лише при наведенні курсору на місце поряд із активною сторінкою (рис.1.5).

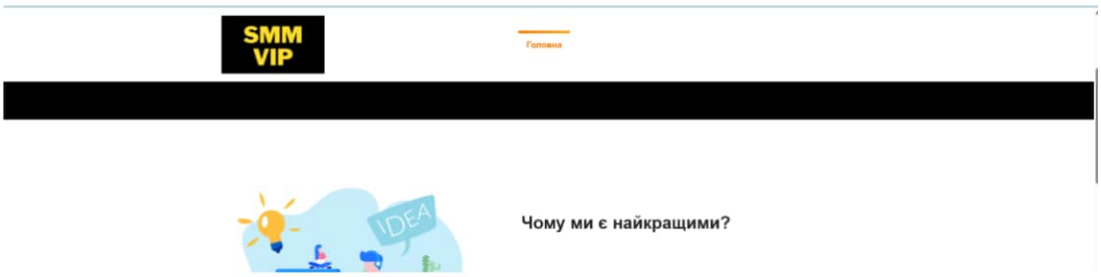


Рисунок 1.4 – Меню навігації

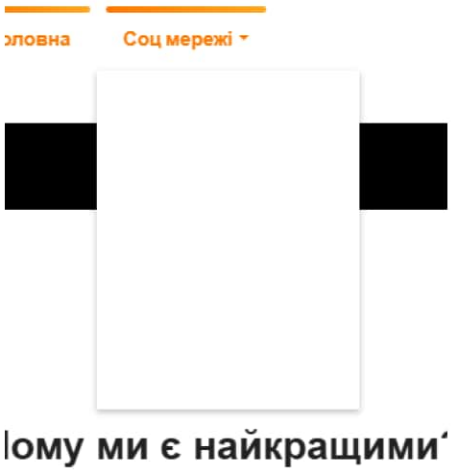


Рисунок 1.5 – Знайдений пункт меню

Роздивляючись повністю саме меню, та і в цілому сам сайт можна зробити висновок, що величезним недоліком є злиття тексту із фоном або іншим текстом: невдалий підбір кольору та розміру тексту, а також невдале його розташування (рис. 1.6). Такий текст вважається нечитабельним.

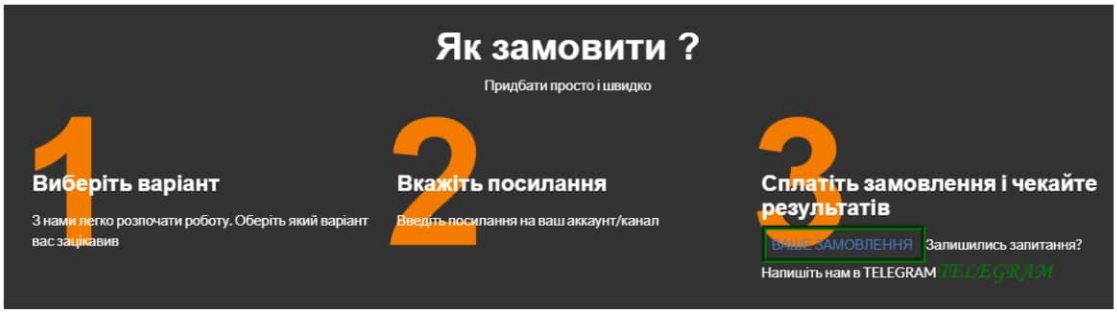


Рисунок 1.6 – Білий текст зливається із цифрами на фоні

Дуже важливо бути уважним у писанні свого коду, щоб не було такого роду негарностей, як на рисунку 1.7.

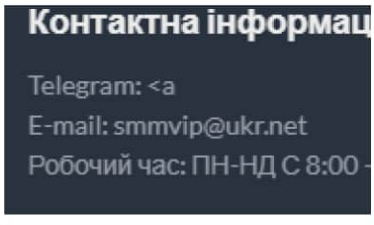


Рисунок 1.7 – Орфографічна помилка у коді

По даній ситуації можна зрозуміти, що тег <a> написаний некоректно і браузер не прочитав цей рядок як тег, а прочитав, як звичайний текст.

Якщо зайти у DevTools (інструменти розробника) та знайти цей елемент у коді, зафіксовано, що посилання взагалі недописано (рис. 1.8).

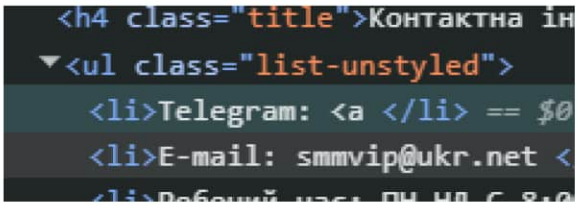


Рисунок 1.8 – Недописаний тег <a>

Мова сторінки – українська. Зовсім незрозуміла наявність таких селектів, як на рисунку 1.9.

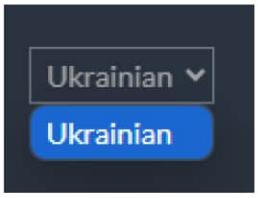


Рисунок 1.9 – Селект для обирання мови

Якщо сторінка має лише одну мову, то зовсім немає сенсу у додаванні селектів, користувач сам може зрозуміти, на якій мові сторінка.

Наявні проблеми із орфографічними помилками та великою кількістю недописаного тексту (рис. 1.10).

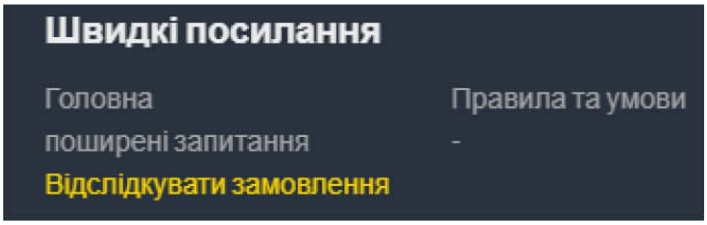


Рисунок 1.10 – Приклад орфографічної помилки (не всі тексти написані з великої літери) та недописаного тексту (незрозумілий символ -)

Для аналізу завантаження сторінки на комп'ютерах та мобільних пристроях використано сервіс від Google «Pagespeed Insights», який дуже простий та зручний у користуванні та показує всю детальну інформацію.

Швидкість завантаження сторінки на комп'ютері, судячи по значенням критерій, ще є у межах норми (рис. 1.11).

Рисунок 1.11 – Оцінка завантаження сторінки на комп'ютерах за критеріями

Максимально жахливий та незручний вигляд адаптивного дизайну. Побачивши вже головний екран, помічено, що шапка сайту перекриває частину тексту головного екрану, вона додає нечитабельності (рис. 1.12).

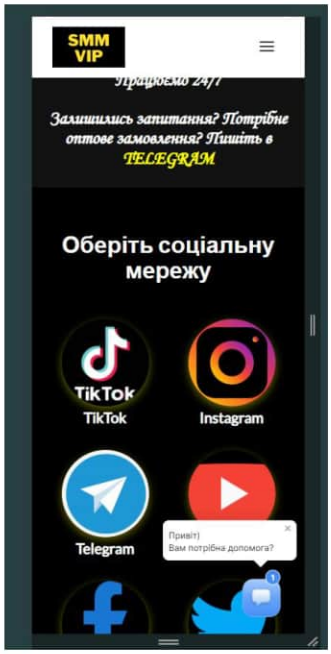


Рисунок 1.12 – Головний екран сайту на мобільних пристроях

В мобільній версії також спостережено картину із злиттям тексту із фоном, наприклад, у мобільному меню (рис.1.13).

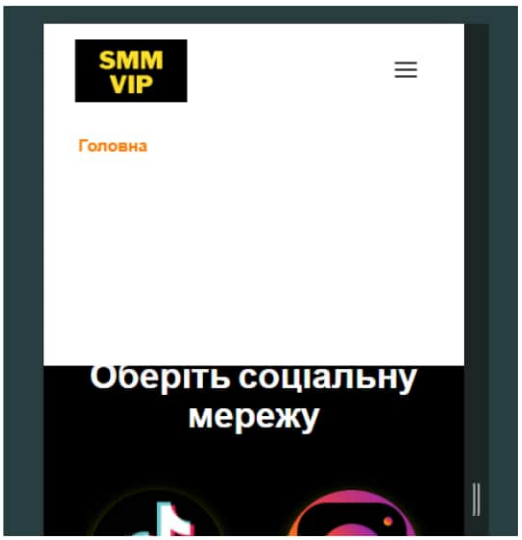


Рисунок 1.13 – Колір пунктів меню зливається із фоном, тому погіршена їх видимість

Швидкість завантаження на мобільних пристроях буде гірша, значення критеріїв дещо відрізняються від швидкості завантаження на комп'ютерах (рис. 1.14).

Рисунок 1.14 – Оцінка завантаження сторінки на мобільних пристроях за критеріями

Дуже погане значення продуктивності, низькі значення критеріїв: SI +3, FCP +0 та LCP +0 (рис. 1.15).

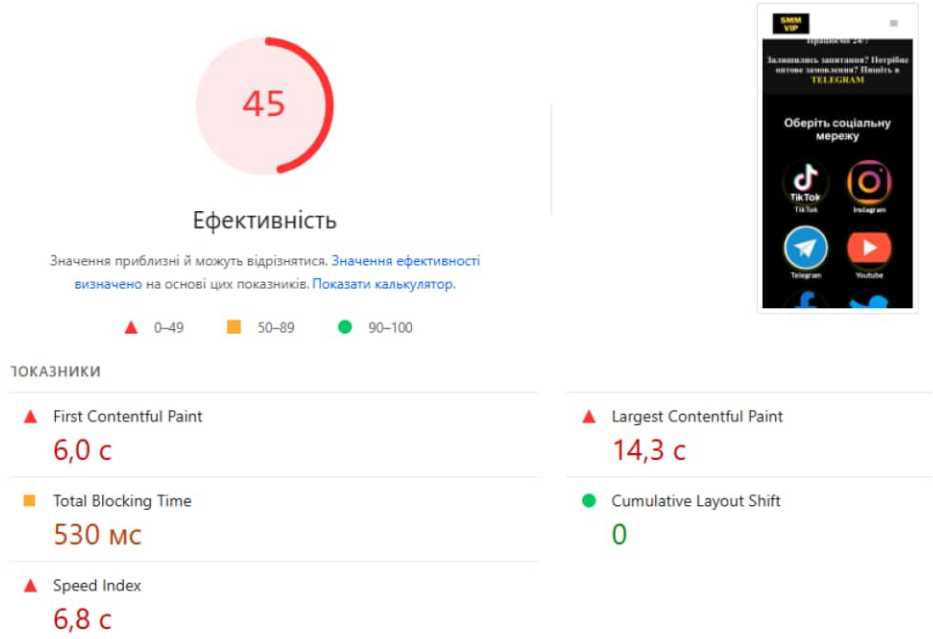


Рисунок 1.15 – Оцінка ефективності сторінки за показниками(мобільні пристрої)

Однією із переваг все ж таки можна виділити наявність анімацій при скролі сторінки, але з наступного прикладу зроблено висновок, що краще поставити у пріоритет прийнятний зовнішній вигляд, навіть, якщо сторінка буде статична.

Наступний лендінг належить студії «BeSocial» [8] – які допомагають людям, які потребують спілкування. Цей односторінковий сайт присвячений пошуку чат-менеджера, тому увага приділяється саме знайомству із студією. На рисунку 1.16 зображено головний екран сторінки.

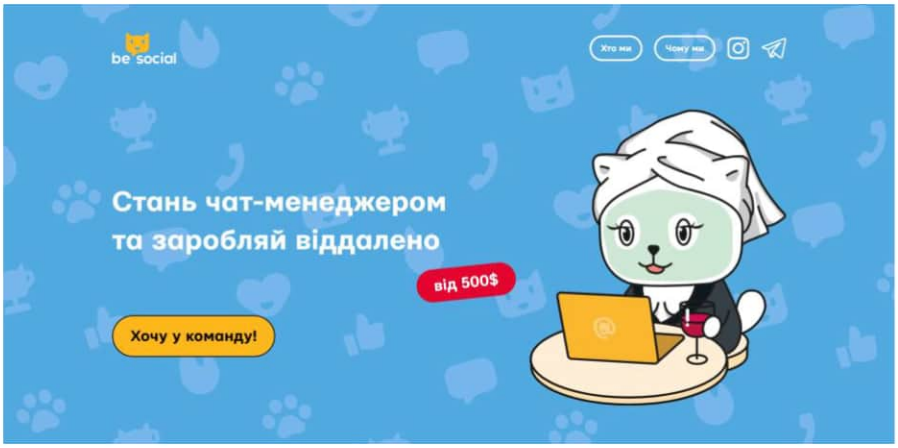


Рисунок 1.16 – Головний екран сторінки «BeSocial»

Переглядаючи сторінку, можна вже виділити такі переваги:

- привабливий, адаптивний та якісний дизайн;
- наявність меню навігації, що надає зручності у користуванні;
- чітка та необхідна інформація «без води»;
- зручне мобільне меню (рис. 1.17);
- швидкість завантаження на комп’ютері у межах норми (рис. 1.18).

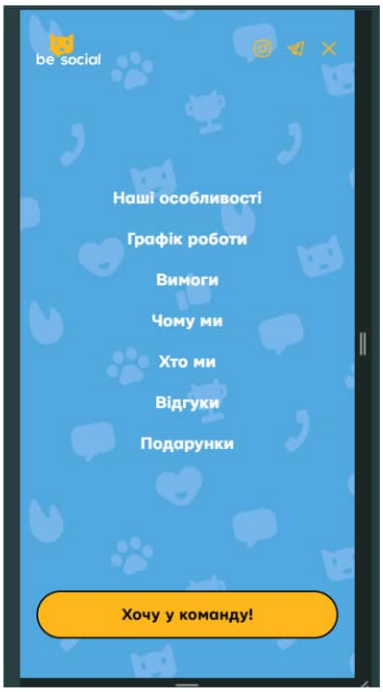


Рисунок 1.17 – Мобільне меню сторінки

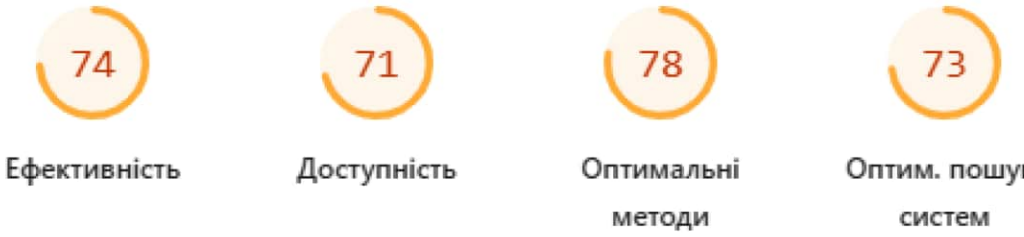


Рисунок 1.18 – Оцінка завантаження сторінки на комп’ютерах за критеріями

Для більш цікавого перегляду рекомендовано додати якоїсь інтерактивності, але порівнюючи із першим прикладом, то, як вже було зазначено, краще мати гарну та якісну сторінку зі статичними елементами, ніж жахливу, але з анімаціями!

Також, рекомендовано закріпити шапку сайту для зручності у навігації, але це не є обов’язковим, якщо сторінка і так має людяний вигляд.

Єдиним недоліком є низькі значення критеріїв завантаження на мобільних пристроях (рис.1.19), а саме значення ефективності: SI +0, FCP +0, LCP +0 та TBT +7 (рис. 1.20).

Рисунок 1.19 – Оцінка завантаження сторінки на мобільних пристроях за критеріями

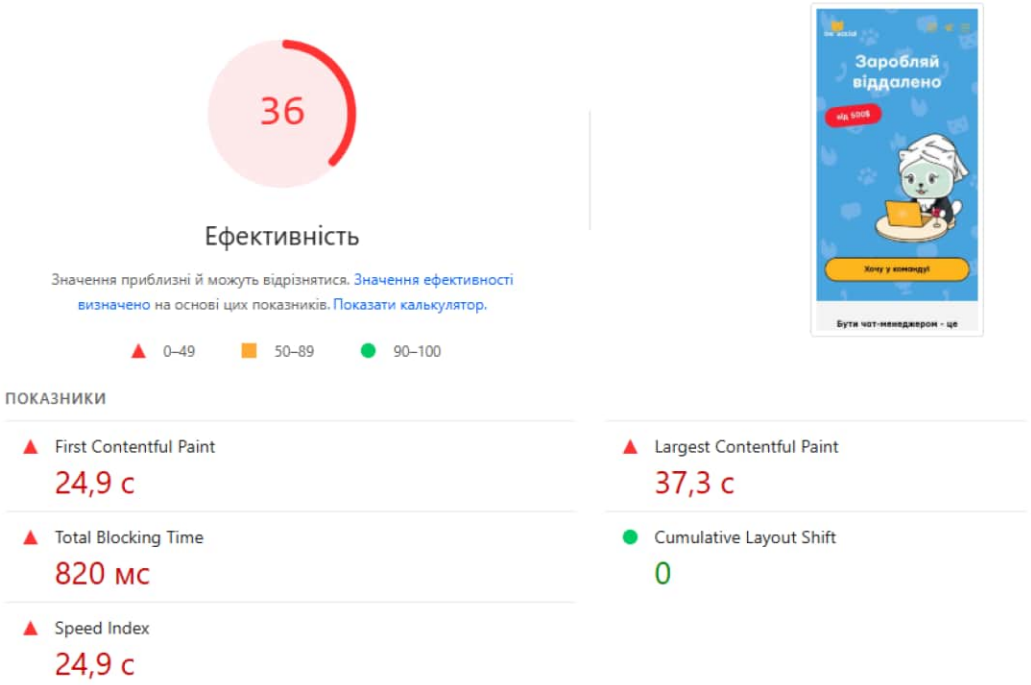


Рисунок 1.20 – Оцінка ефективності сторінки за показниками(мобільні пристрої)

Підсумовуючи всі переваги та недоліки обох прикладів, можна дати кожному прикладу по найголовнішій рекомендації:

• VIP SMM: подбати про приємний дизайн, у тому числі адаптивний, та читабельність тексту, перевірити тексти та код на помилки та пропуски;

• BeSocial: подбати про продуктивність мобільної версії сайту, проаналізувати завантаження сторінки та дотриматись наданих рекомендацій для запобігання низьких показників оцінки.

1.3 Огляд текстових редакторів для розробки та обґрунтування вибору основного інструменту

Розглянуто декілька прикладів текстових редакторів для веброзробки.

Високопродуктивним редактором коду, розмітки та тексту є редактор Sublime Text [9]. Це дуже легкий та швидкий редактор, який працює на Windows, MacOS та Linux. Він славиться мінімалістичним інтерфейсом без зайвих елементів, високою та надзвичайною продуктивністю навіть на слабких машинах та підтримкою множинних виділень тексту – підсвічування синтаксису та збірки для десятки мов [10, 11]. Проте, серед мінусів, чого розробники зазвичай не віддають перевагу цьому редактору, можна виділити те, що більшість розширень – це сторонні плагіни, які потребують налаштовування вручну. Середовище має платну ліцензію, тому ще одним недоліком є пропрієтарна ліцензія, тобто тимчасовий безстроковий демо-режим із періодичними нагадуваннями [12].

Легким редактором від компанії Adobe із акцентом на веброзробку є редактор Brackets [13]. Це безкоштовне та відкрите середовище, орієнтоване на вебдизайн та front-end розробку, написане на HTML, CSS та JavaScript і також доступне для Windows, MacOS та Linux. Основними перевагами є підтримка Live Preview, автоматичне відображення змін у браузері в реальному часі, та підтримка CSS-препроцесорів. Проте, підтримка Live Preview працює лише у браузері Google Chrome, а при відкритті DevTools з'єднання може перерватись та і взагалі, після 2021-го року, розвиток цього

редактора сповільнився та знизилась активність його розробників, тому його менше обирають для створення більш сучасних проєктів [14, 15].

Кожен розробник сам обирає для себе той текстовий редактор, який він вважає найзручнішим та спроможним. Середовищ розробки існує безліч і всі мають право на існування та використання. Проте, самим конкурентоспроможним є найпопулярніший текстовий редактор Visual Studio Code (VS Code), більшість розробників обирає саме його.

VS Code [16] – це безкоштовне, кросплатформне середовище від компанії Microsoft, яке поєднує в собі підтримку багатьох мов програмування та розширень. Це середовище поєднує в собі простоту використання та наявність потужних інструментів для розробника, що робить його зручним для створення різноманітних веб-проєктів, тому для розробки лендінгу було обрано саме його. Інтерфейс цього редактору зображено на рисунку 1.21.

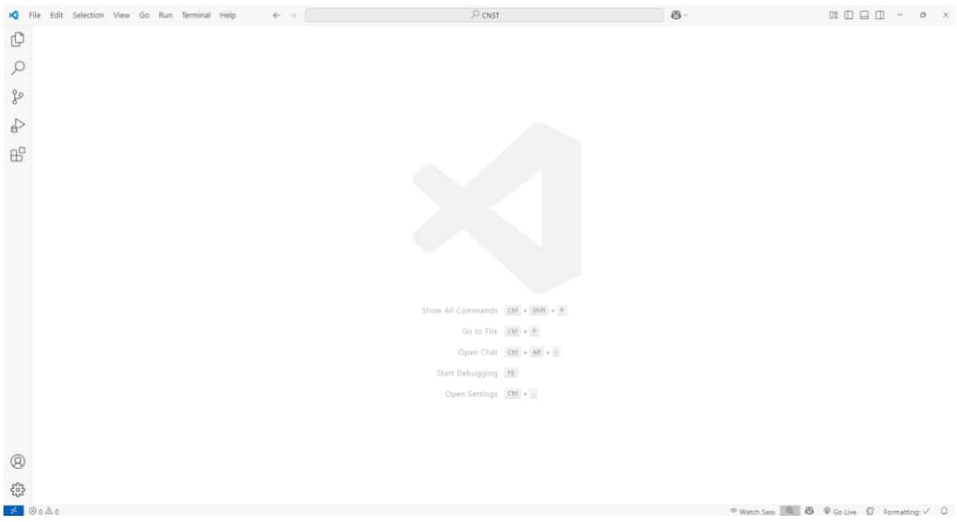


Рисунок 1.21 – Середовище розробки VS Code

Основні переваги редактору VS Code:

- підтримка понад 50-ти мов;
- автоматичне завершення коду, що пришвидшує написання та зменшує кількість помилок;
- наявність вбудованого терміналу;
- вбудована підтримка Git та взаємозв’язок із GitHub;

- наявність системи розширень;
- зручна навігація по проєкту.

Розглянуто одні із популярних розширень редактора, які найчастіше розробники використовують при створенні своїх проєктів:

1. Prettier – автоматичне форматування коду за обраним стилем для таких мов, як JavaScript, HTML, TypeScript, CSS та ін [17].
2. Live Server – запуск локального сервера із автоматичним перезавантаження сторінки для зручного перегляду результатів [17].
3. PHP Intelephense – підтримка синтаксису PHP – підсвічування, автодоповнення та діагностика коду.
4. Emmet – швидке написання коду завдяки спеціальним скороченням.
5. Autoprefixer – автоматичне доповнення префіксами у властивостях CSS для підтримки кросбраузерності.
6. eCSSctractor – конвертація скопійованого HTML-коду у шаблон CSS або SCSS для швидкого та зручного написання стилів.
7. Live Sass Compiler – запуск SASS-компілятора, який конвертує SASS код у CSS.

1.4 Вибір технологій розробки

Орієнтуючись на сучасні тенденції веброзробки (п. 1.1), буде розглянуто наступні технології для створення лендінгу.

1.4.1 HTML5. Мова гіпертексту HTML5 використано для написання базової розмітки сторінки. Ця мова має семантичні теги, такі, як <header>, <main>, <footer>, <nav> і т. п., що роблять код більш зрозумілим та читабельним, а також дружнішим до пошукових систем. HTML5 – це стандарт, який підтримують усі сучасні браузери, тому переживати про сумісність не доведеться.

1.4.2 SCSS. Препроцесор SCSS дозволяє писати CSS код зі змінними та вкладеннями. Завдяки цьому, стилі виглядатимуть більш структурованими та

легко їх буде підтримувати. Можливість створювання змінних дає змогу змінювати значення властивості в одному місці, що надає зручності у здійсненні правок, бо зміни застосовуватимуться скрізь, де була використана змінна.

1.4.3 JQuery (JavaScript). Ця бібліотека є чудовою альтернативою ванільного JS, що мінімізує код, маючи власні методи. Також завдяки JQuery можна швидко та легко маніпулювати DOM, а також створювати AJAX-запити для відправки даних, наприклад, з форми. З його допомогою легко можна зробити валідацію форми та підключати різні плагіни, які полегшують розробку і не писати довгі селектори власноруч.

1.4.4 Бібліотеки та API. Для реалізації функціональних елементів, такі як слайдери та анімації використовують різні бібліотеки та API, які полегшують та пришвидшують написання коду.

Для створення слайдерів використано дві популярні JS-бібліотеки, які використовують більшість розробників: Swiper (бібліотека JavaScript) та Owl Carousel (плагін JQuery). Ці бібліотеки надають можливості реалізації адаптивних слайдерів із підтримкою свайп-жестів для мобільних пристроїв.

Для додавання «лінивих» анімацій використано Intersection Observer API. Це сучасний JavaScript API, який дозволяє відстежувати появу елементів у зоні видимості екрану (viewport). Іншими словами, він повідомляє, коли певний елемент перетинає межі вікна браузера або контейнера.

Додатково використано бібліотеку Font Awesome для додавання іконок без створення власних SVG. Це іконки, які реалізовані за допомогою стилів та в загалом це називається іконочним шрифтом.

1.4.5 PHP + MySQL. Для реалізації серверної частини сайту використано серверну мову програмування PHP (версія 7+) та СУБД MySQL для роботою із базою даних. PHP виконує обробку даних форми, зберігає їх у БД та надсилає запит у Telegram-бот. MySQL – проста реляційна СУБД, яка добре справляється із невеликими обсягами даних заявок.

1.5 Висновки до розділу 1. Постановка задачі розробки

Отже, проведене дослідження предметної області дійсно підтвердило важливість якісних, привабливих та сучасних сайтів-візиток у будь якому бізнесі, навіть, якщо це розкручена компанія. Також, доведено, що дійсно лендінг є чудовим способом представити коротку, чітку та важливу інформацію «без води», як на 2-му прикладі порівняльного аналізу. Порівняльний аналіз нам показав, що головним є загальна якість, а не кількість використаного функціоналу, в якому не має сенсу, якщо на перший погляд вже виглядає жахливо.

В розділі було представлено огляд сучасних тенденцій веброзробки для подальшого вибору технологій для створення сайту. Проведено порівняльний аналіз двох абсолютно різних між собою прикладів лендінгів, який висвітлює проблеми та недоліки веброзробників та показує, які нюанси потрібно врахувати при розробці. Представлено огляд прикладів середовищ розробок, обрано основний інструмент розробки та обґрунтовано вибір. Обрано технології для розробки на основі огляду сучасних тенденцій. Представлені висновки розділу та поставлено задачі розробки системи.

Для досягнення поставленої мети, поставлено низку задач:

1) створення/пошук сучасного та привабливого дизайну, який відповідає фірмовому стилю та сучасним веб-трендам;
2) розробка адаптивного інтерфейсу користувача, що коректно відобразиться на різних пристроях;
3) реалізація функціональних елементів;
4) розміщення готового сайту на хостинг;
5) забезпечення базою даних, де будуть зберігатися дані з форми;
6) реалізація надсилання даних у Telegram-бот, який слугуватиме додатковим каналом для швидкого отримання нових заявок;
7) перевірка працездатності сторінки у реальному середовищі.

РОЗДІЛ 2
ПРОЄКТНІ РІШЕННЯ

Представлено умовну компанію CNST, яка спеціалізується на наданні якісних будівельних послуг, зосереджуючи свою діяльність на комплексному обслуговуванні будівельних проєктів: від планування і проєктування до реалізації та здачі об'єктів під ключ. CNST поєднує сучасні технології та професійний підхід, а також має досвід роботи близько 10-ти років.

З метою популяризації діяльності компанії та залучення нових клієнтів було створено вебсторінку типу «лендінг». Це односторінковий сайт, головне завдання якого — коротко та ефективно донести користувачеві найважливішу інформацію про компанію та її послуги. Вебсторінка має сучасний адаптивний дизайн, що дозволяє зручно переглядати її з різних пристроїв, включаючи смартфони, планшети та десктопи. Вона також оптимізована для швидкого завантаження.

Сторінка ознайомлює користувача з ключовими послугами компанії, її перевагами, етапами виконання робіт, відгуками клієнтів, а також відповідями на поширені запитання. Завдяки такій структурі, користувач отримує повну картину про діяльність компанії без необхідності переходити на інші сторінки. Кожна секція логічно доповнює попередню, плавно ведучи від ознайомлення до дії — заповнення форми зворотного зв'язку.

Окрему увагу було приділено реалізації форми зворотного зв'язку, що дозволяє відвідувачеві швидко залишити заявку на консультацію. Після заповнення форми дані користувача зберігаються у базі даних і відправляються адміністратору в Telegram, що дозволяє оперативно опрацювати звернення.

На рисунку 2.1 зображено схематично мапу сайту, яка чітко демонструє структуру та ієрархію сторінки — показує, з яких основних секцій вона складається та яку інформацію або дії дають користувачеві ці секції. Така

візуалізація є корисною як для розробника, так і для замовника, адже дозволяє з першого погляду оцінити логіку побудови сайту та виявити можливі недоліки або покращення.

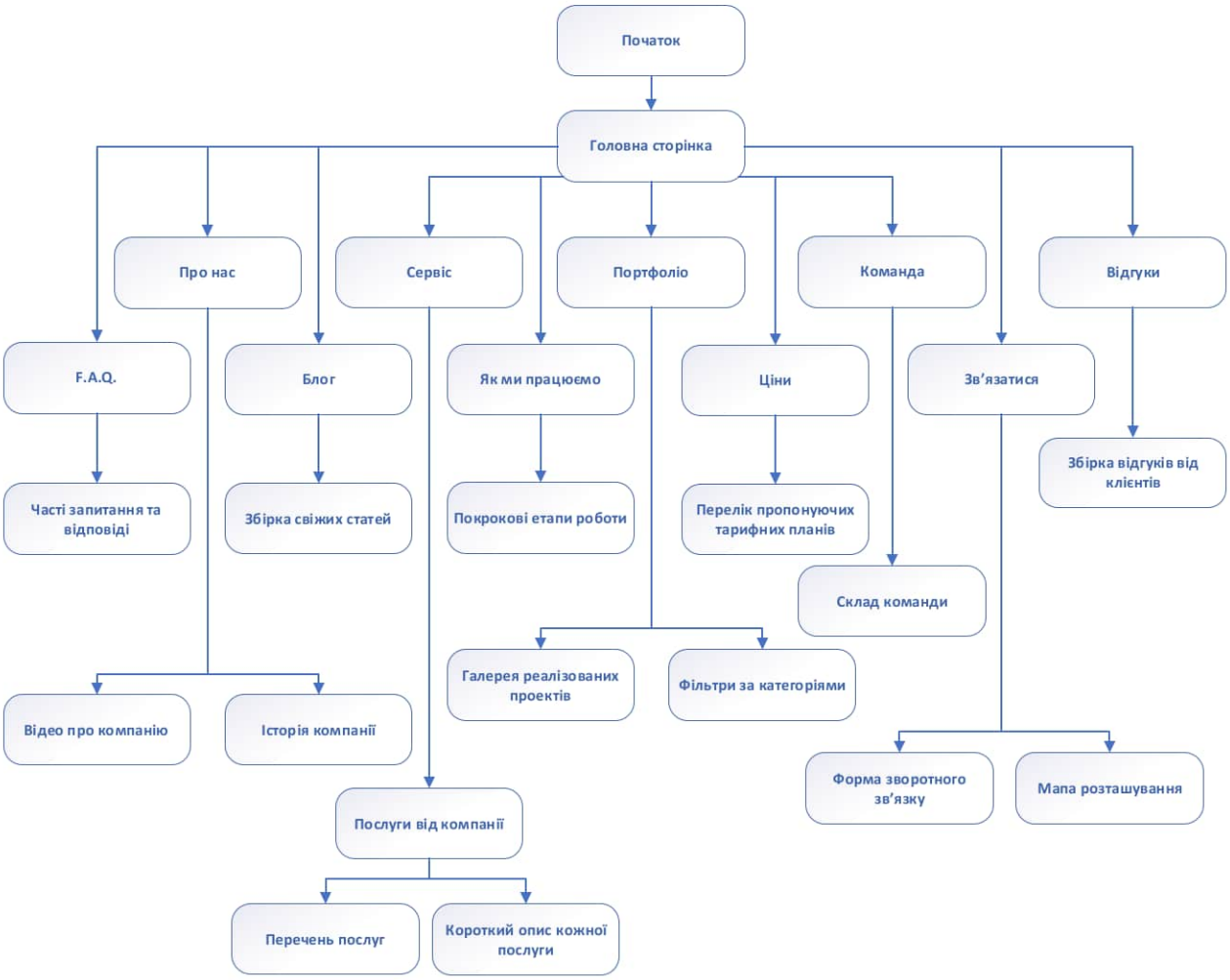


Рисунок 2.1 – Мапа сайту

Мапа сайту відображає логічну структуру вебресурсу компанії CNST, розділену на основні тематичні блоки. Центральним елементом є головна сторінка, яка веде до ключових секцій «Про нас», «Сервіс», «Портфоліо», «Блог» та «Зв'язок», а також до додаткових секцій «Команда», «Відгуки», «F.A.Q.», «Як ми працюємо» та «Ціни».

Кожна з секцій має власний зміст. Наприклад, секція «Сервіс» можуть містити конкретні напрямки діяльності, такі як «Реконструкція будівлі» тощо. Секція «Портфоліо» повинна містити приклади виконаних робіт, а в секції

«Зв’язатись» доступна карта розташування офісу, а також форма зворотного зв’язку. Подібна структура полегшує навігацію користувачам та сприяє швидкому пошуку необхідної інформації.

2.1 Пошук привабливого та сучасного макету дизайну

Розробкою вебпроєктів займається ціла команда працівників, кожен з яких має свої задачі та обов’язки. Перший, хто починає та робить перший крок у реалізацію проєкту – це вебдизайнер.

Вебдизайнер – це людина, яка розробляє майбутній дизайн проєкту, враховуючи правила та нюанси UI та UX. Мета вебдизайнера – створення макету, який буде зрозумілим розробнику, для реалізації ідей замовника, підлаштовуючись під стиль організації та враховуючи зручність у користуванні клієнтською частиною сайту. Вебдизайнерам важливо прислуховуватись до порад та можливих правок від клієнта та розробника для покращення дизайну, бо зрозумілий дизайн – це важливий етап у реалізації проєкту без всяких проблем. Якщо немає можливості зв’язатись із дизайнером для вирішення деяких питань з приводу макету, обов’язок переходить до розробника, який має на свій власний розсуд викрутитись з таких проблем та не підлаштовуватись під недоліки дизайнера, бо, в такому випадку, головною умовою розробника є реалізація якісного дизайну, а не копіювання його помилок.

Оскільки, в даному випадку, немає людини, яка б змогла реалізувати ідеї розробки, було прийнято рішення надихнутись ідеями пошуку готового макету дизайну. Існує багато Інтернет-ресурсів, які мають безліч варіантів макетів на будь-які потреби та смак.

Найпопулярнішим методом пошуку необхідного матеріалу – спілкування із іншими фахівцями, які мають більший досвід роботи та можуть дати безліч порад, які стануть у нагоді. Для цієї мети використовують різні групи для спілкування у месенджерах, серед яких більш віддають перевагу

Telegram. Цей месенджер дуже стрімко розвивається та стає схожим на велику платформу, яка поєднує в собі і звичайне спілкування, і перегляд різного корисного, розважального контенту та матеріалу. Telegram дає можливість створення каналів, в яких люди публікують такий матеріал.

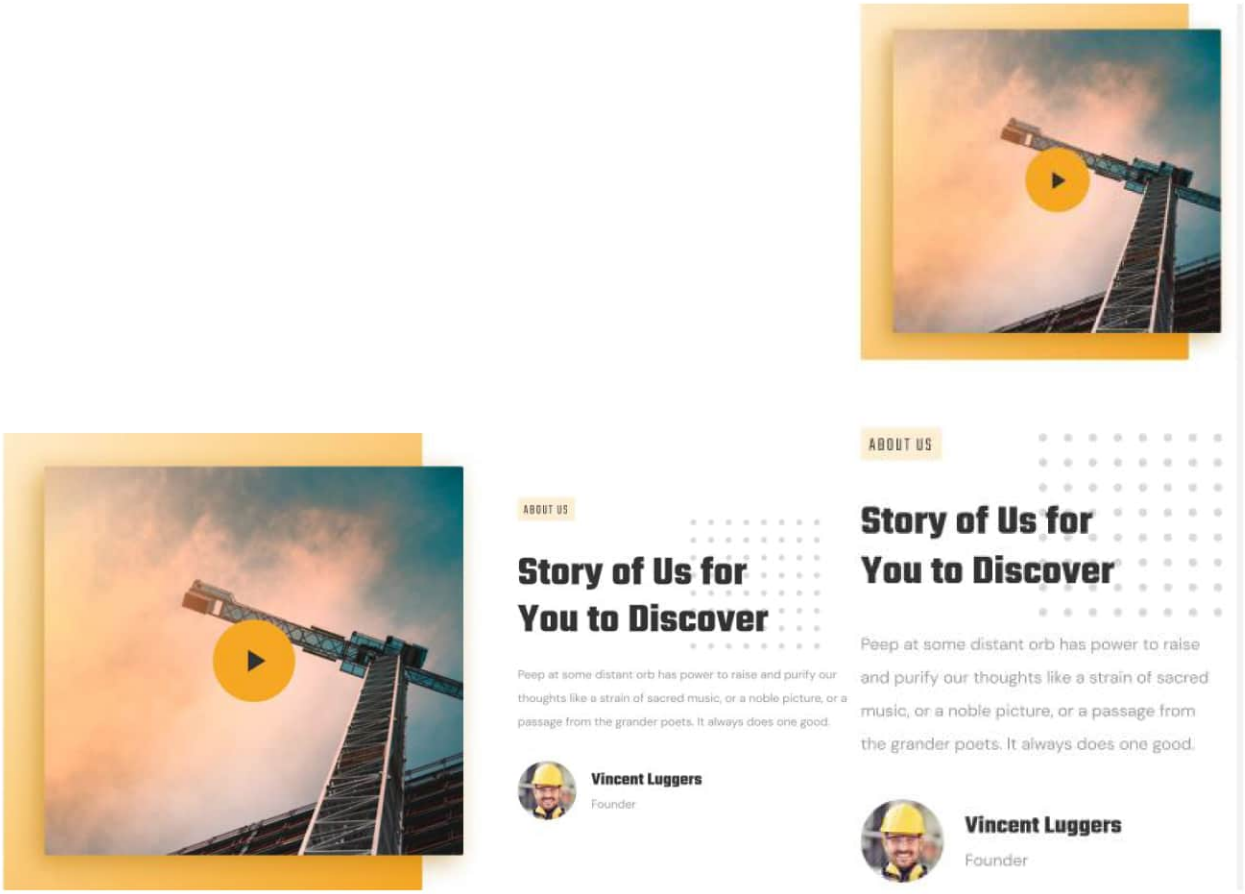
Завдяки такому спілкуванню із професійними веброзробниками, було знайдено відповідний макет у Telegram-групі «FIGMA Templates», а саме, було завантажено найбільш привабливий архів, який має 11 адаптивних дизайнів лендінгів. Для перегляду макету використано програму Figma.

Figma – це безкоштовний веб-інструмент для створення дизайнів інтерфейсу, який поєднує в собі малювання макетів, створення інтерактивних прототипів та передачу їх розробникам [18]. Ця програма доступна, як в онлайн режимі, так і для завантажування на різних ОС (Windows, MacOS, Linux). Розробники використовують цей інструмент для перегляду самого макету та стилів елементів дизайну. На рисунку 2.2 зображено обраний макет для реалізації ідей розробки, який підходить під всі вимоги чіткого та цікавого представлення будівельної компанії.

Рисунок 2.2 – Обраний макет лендінгу для розробки

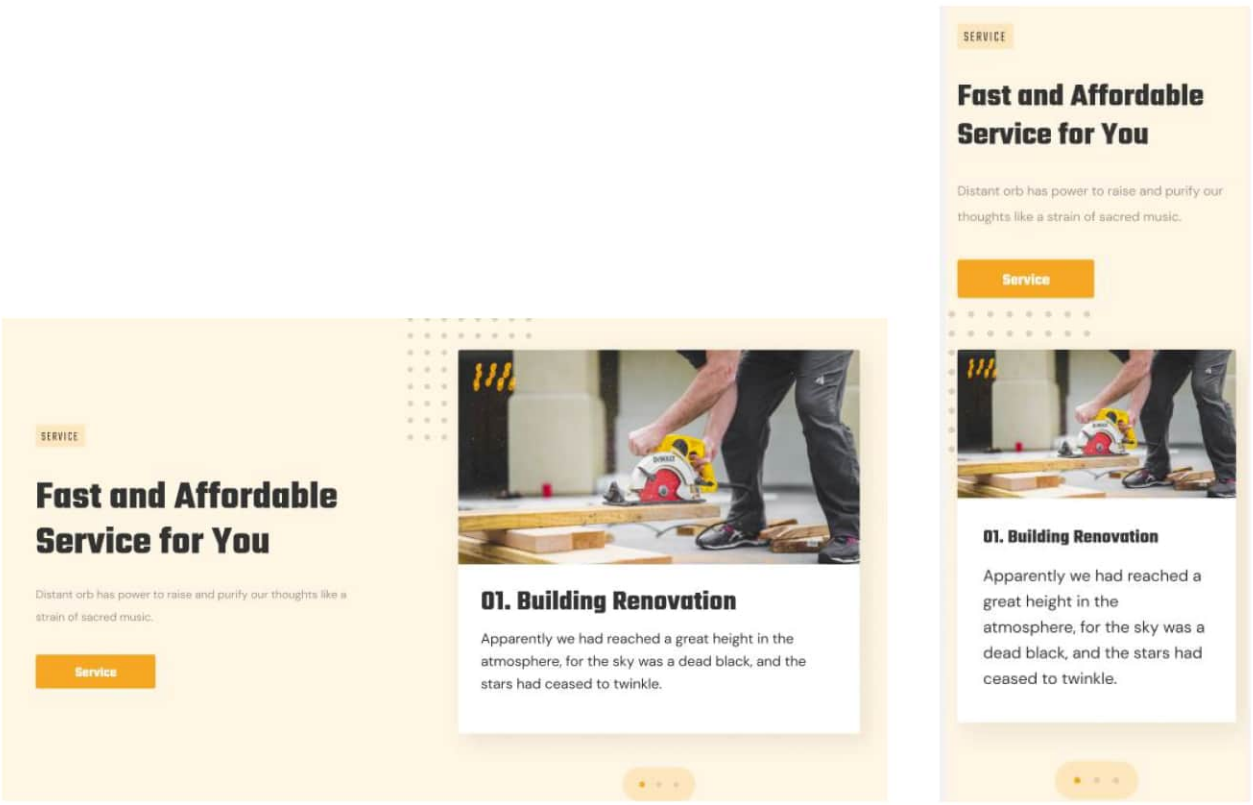
Представлено дизайн у двох виглядах: комп’ютерна та мобільна версія. Прототип сайту має меню навігації (рис. 2.3), різні секції із різною ознайомчою інформацію, такою, як про компанію (рис. 2.4), сервіс (рис. 2.5), портфоліо (рис. 2.6), новини (рис. 2.7), контактні дані і т. п..

Рисунок 2.3 – Меню навігації (шапка сайту)



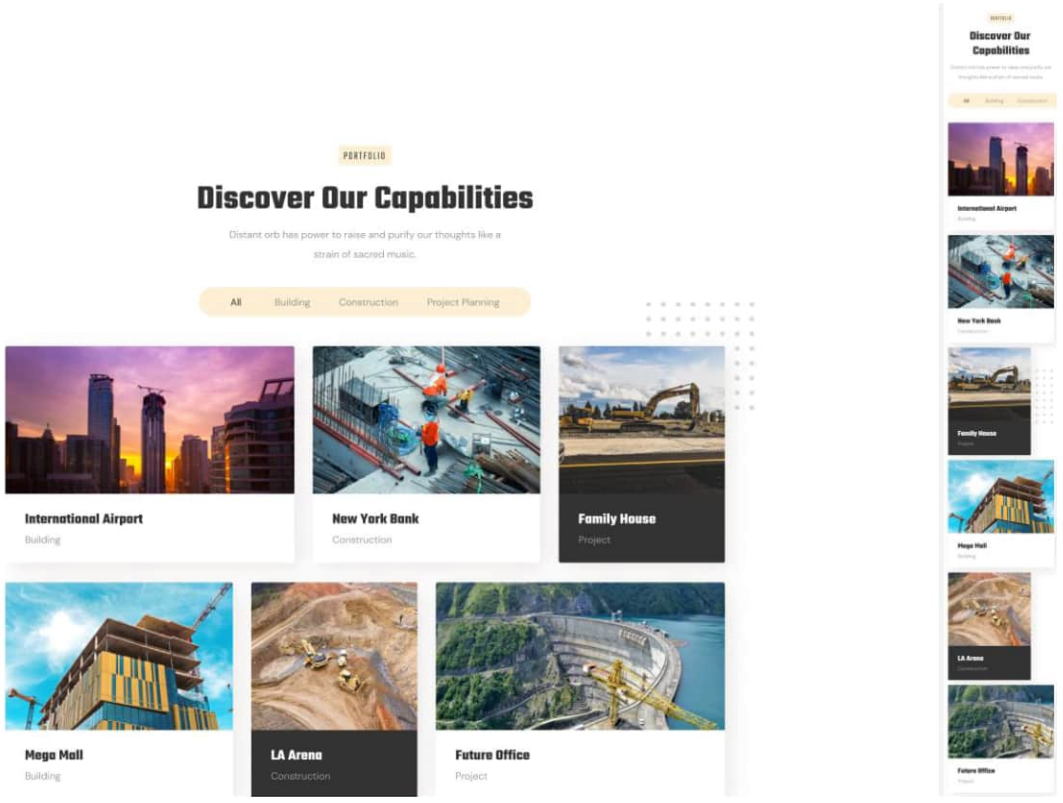
а) б)

Рисунок 2.4 – Секція «Про нас»: а) комп’ютерна версія; б) мобільна версія



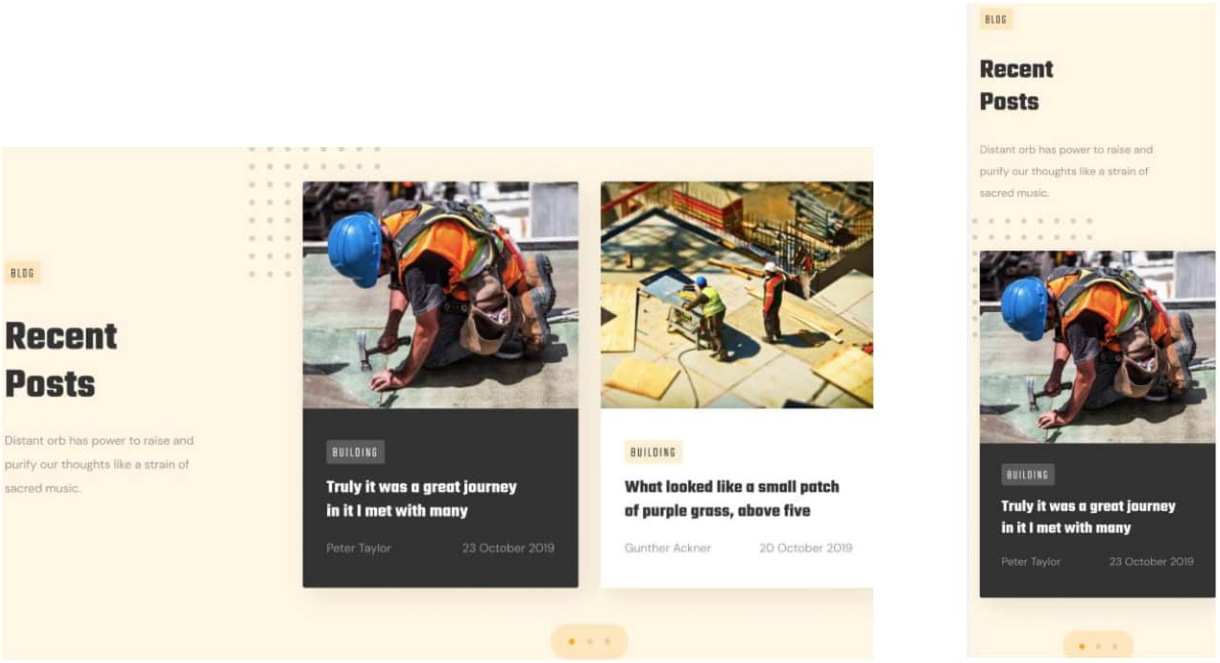
а) б)

Рисунок 2.5 – Секція «Сервіс»: а) комп’ютерна версія; б) мобільна версія



а) б)

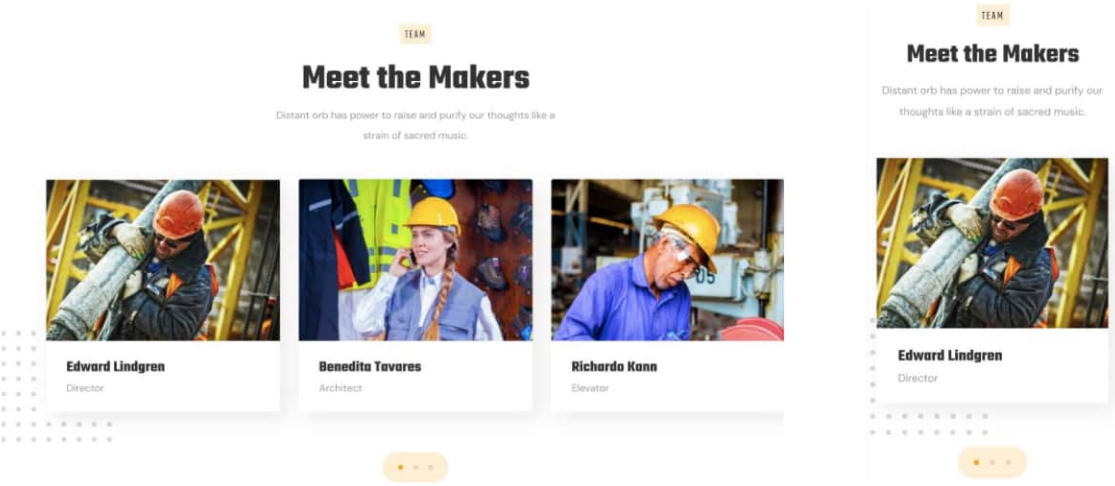
Рисунок 2.6 – Секція «Портфоліо»: а) комп’ютерна версія; б) мобільна версія



а) б)

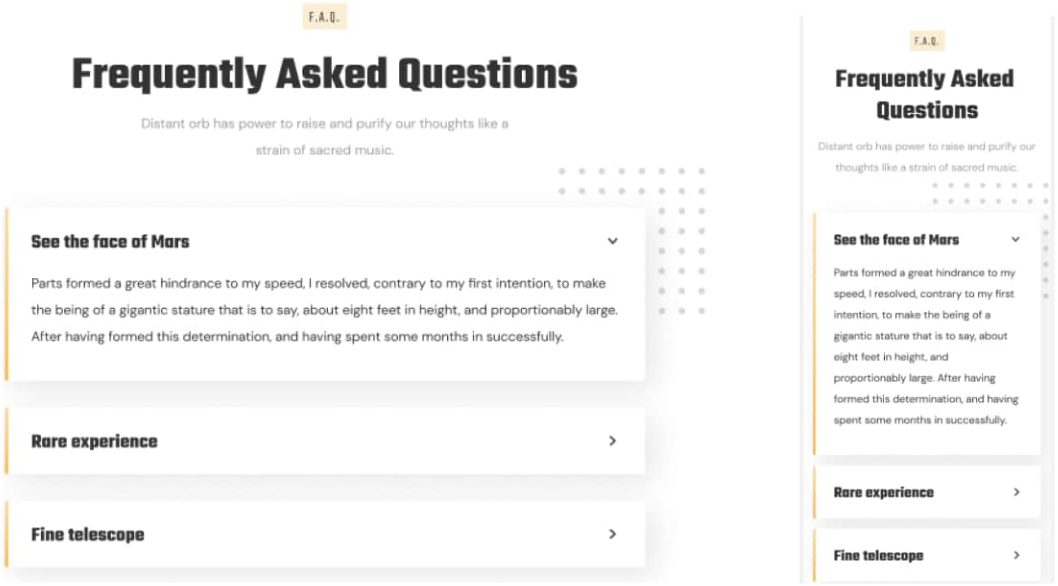
Рисунок 2.7 – Секція «Блог»: а) комп’ютерна версія; б) мобільна версія

Можна побачити, що сайт повинен мати функціональні елементи, такі, як відео, як на рисунку 2.4, слайдери (рис. 2.5 та рис. 2.8), деякі події при наведені на елемент курсором миші, як, наприклад на рисунку 2.7 (зміна кольорів слайду), акордеон (рис. 2.9), мапа місця розташування та форма зворотного зв’язку (рис. 2.10).



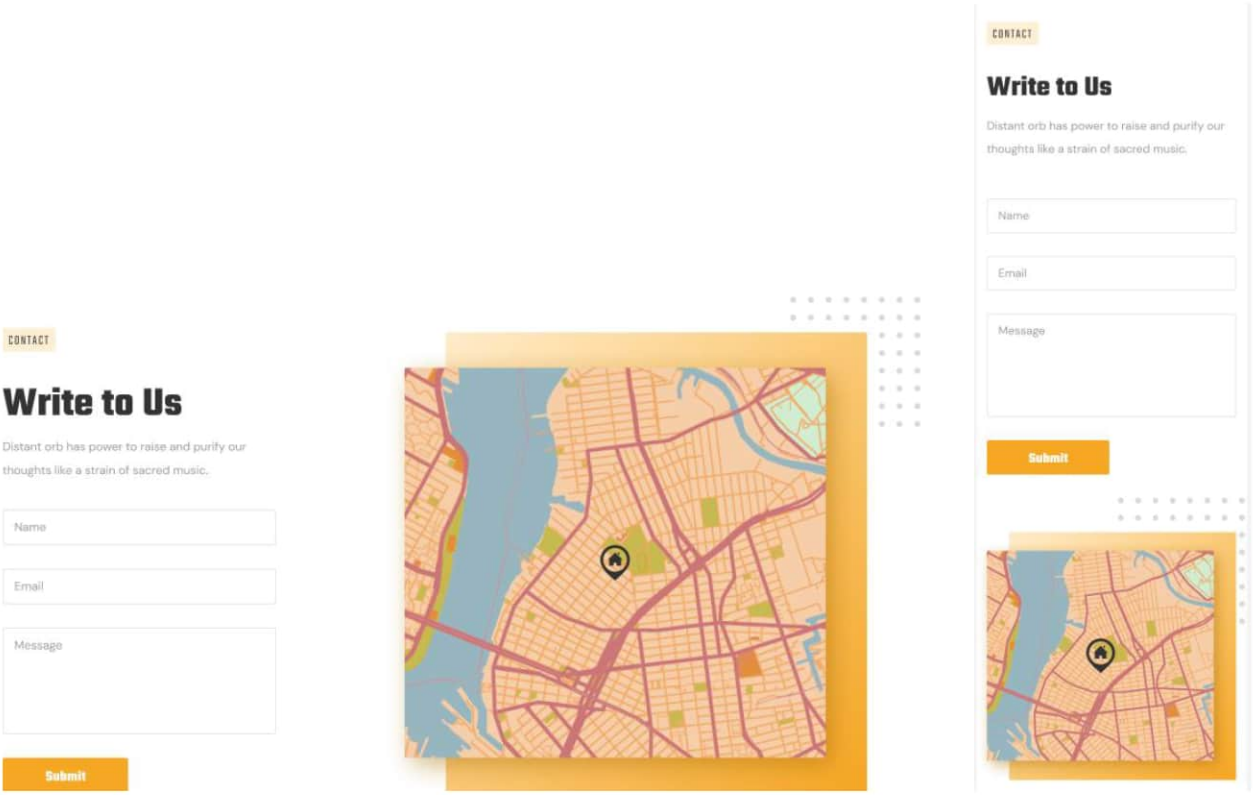
а) б)

Рисунок 2.8 – Слайдер у секції «Команда»: а) комп’ютерна версія; б) мобільна версія



а) б)

Рисунок 2.9 – Акордеон у секції «F.A.Q.»: а) комп’ютерна версія; б) мобільна версія



а) б)

Рисунок 2.10 – Форма зворотного зв’язку та мапа у секції «Зв’язок»: а) комп’ютерна версія; б) мобільна версія

Макет не позбавлений недоліків. По-перше, відсутній дизайн планшетної версії. Можна подумати, що на планшеті повинно виглядати так, як на телефоні, але потрібно врахувати той факт, що в деяких випадках, це може виглядати не так, як очікувалось і потрібно такі нюанси виправляти на власний розсуд, щоб все виглядало гармонічно та не ламано.

Мобільна версія повинна мати мобільне меню, про те дизайн цього меню відсутній. В такому випадку, потрібно увімкнути імпровізацію та, підлаштовуючись під стиль сайту, зробити таке меню, яке буде працювати, як навігація по сайту.

Якщо повернутись до рисунку 2.6(б), де зображено мобільну версію секції «Портфоліо», можна відзначити, що дизайнер не продумав, як такий адаптив буде виглядати на мобільних пристроях, які будуть шириною екрана більші, ніж в макеті (> 375 пікселів). Кращим рішенням буде розтягнути всі картки по ширині, бо такий підхід із деякими картками, які будуть залишати деяке пусте місце справа, виглядатиме насправді ламано та незручно для перегляду.

На рисунку 2.11 зображено секцію «Як ми працюємо», в якому показані етапи роботи компанії. Дизайнер також не продумав повністю адаптив цієї секції, не враховуючи принципи UX. Рішенням дизайнера стало зробити горизонтальний скрол цих кроків, але не врахував той факт, що користувач не зрозуміє, що потрібно скролити, щоб подивитися кроки, але основною метою є розробка сайту, який буде зрозумілим користувачу під час перегляду, тому, як вже було згадано, якщо немає можливості зв’язатись із дизайнером, то відповідальність перекладається на розробника, який не повинен реалізовувати помилки дизайнера та повинен вирішити поставлені проблеми.

а) б)

Рисунок 2.11 – Секція «Як ми працюємо»: а) комп'ютерна версія; б) мобільна версія

2.2 Розробка адаптивного інтерфейсу

Розробка починається зі створення та підготовки проєкту, проведення базових налаштувань.

2.2.1 Налаштування розширення Live Sass Compiler. Зміни налаштувань розширення Live Sass Compiler проведено у файлі settings.json (рис. 2.12) – конфігураційний файл основних налаштувань та налаштувань розширень.

```
"liveSassCompile.settings.formats": [
  {
    "format": "compressed",
    "extensionName": ".min.css",
    "savePath": "/css"
  }
```

Рисунок 2.12 – Налаштування Live Sass Compiler у файлі settings.json

Конвертується SASS-код у CSS, створюється мінімізований (стислий) CSS-файл, в який переноситься переведений файл із кодом та зберігається файл у каталозі CSS.

2.2.2 Створення змінних. Файл зі SASS-змінними (створено файл _variables.scss (рис. 2.13)) розміщують в окремому каталозі для файлів-утиліт(було створено каталог scss/utils). Створено кореневі змінні для кольорів, які використані в макеті, а також змінні із значеннями переломних точок (breakpoints) для реалізації адаптиву: мобільні пристрої, планшети та комп'ютери. Адаптив реалізовано підходом Mobile-first, тому реалізовано такі змінні, показані у коді нижче.

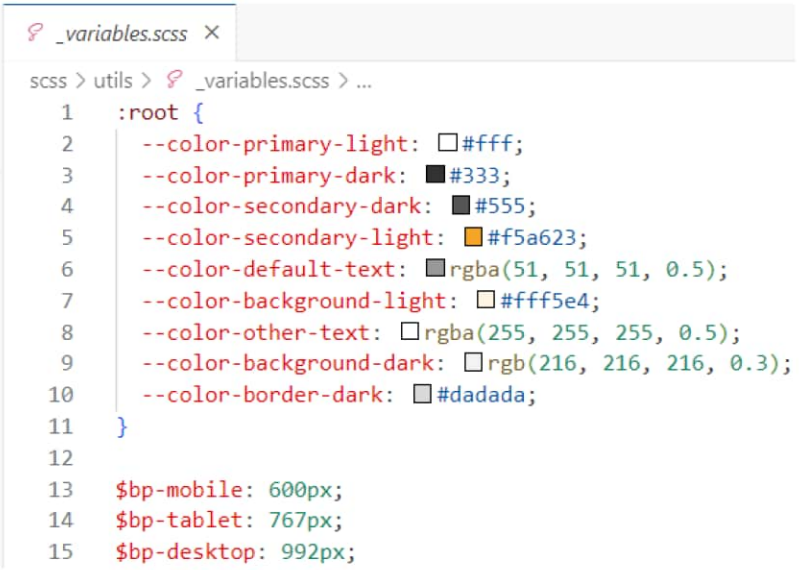


Рисунок 2.13 – Вміст файлу _variables.scss

2.2.3 Базові стилі. Для зручності та структурування стилів було створено окремий файл _base.scss, який міститься в каталозі scss. Цей файл відповідає за скидання стилів браузера за замовчуванням, базову типографіку, а також допоміжні стилі, які будуть використовуватись у всьому проєкті. Подібне структурування дозволяє уникати дублювання коду, зробити його більш читабельним та легким для підтримки.

Насамперед у файлі скидаються зовнішні та внутрішні відступи у елементів, таких як body, ul, input, td. Також усі елементи й псевдоелементи

налаштовуються на використання моделі блочного розрахунку розмірів (box-sizing: border-box). Це дозволяє краще контролювати розміри елементів на сторінці.

Встановлено базові стилі для елементів типу html та body, зокрема висота 100%, шрифт, кольори, розмір шрифту, фон, міжрядковий інтервал, відключення горизонтального прокручування (overflow-x: hidden) тощо. Для заголовків і абзаців (h1 - h6, p) обнулено верхній відступ, що дозволяє уникати небажаних прогалин між елементами.

Також у файлі вказано скидання стилів списків: ul має list-style-type: none, що прибирає маркери. Для address встановлено нормальний стиль шрифту (без курсиву). Елементам a і img задано блочне відображення, а зображенням — максимальна ширина 100% та автоматична висота з object-fit: cover, що допомагає адаптувати їх під різні блоки без втрати пропорцій.

Крім цього, поля введення (input, textarea, button) успадковують загальні шрифти, скидається їхня зовнішність у деяких браузерах, прибираються зайві стилі для пошуку та випадаючих списків, вимикається outline при фокусі, заборонено зміну розміру textarea, та додано плавну анімацію для a і button при зміні кольору чи фону.

Розглядаючи макет, можна зазначити стилі, які найбільш вживані і тому, краще задати значення цих стилів глобально тегу <body>, ніж писати це значення по декілька разів кожному елементу, таким чином дублювавши код. Це кольори та типографічні налаштування, які створюють єдину візуальну систему. Задано основний темний колір тексту (темно-сірий), обраний шрифт проєкту DM Sans, який використовується по всьому сайту, базовий розмір шрифту 16 пікселів, білий фон сайту, збільшений міжрядковий інтервал (використовується для звичайних текстів (2)) та стандартна товщина тексту 400.

Завдяки структурі цього файлу (див. додаток А), розробка наступних модулів значно спрощується, оскільки найпоширеніші стилі вже враховані і не потребують повторного оголошення.

2.2.4 Завантаження шрифту. Кращою та надійною практикою є завантаження локальних шрифтів, Для пошуку необхідного шрифту під назвою DM Sans, який належить до сімейства шрифтів sans-serif, використано сервіс google-webfonts-helper [19]. Обрані необхідні товщини шрифту, завантажено архів та отримано код підключення шрифту (його поміщено у відповідний файл _fonts.scss (рис. 2.14), у каталозі з утилітами utils). Завантажено файли шрифтів формату woff2.

Рисунок 2.14 – Вміст файлу _fonts.scss

2.2.5 Підключення необхідних файлів. Для подальшої роботи потрібно підключити необхідні файли. У тезі <head> додано фавікон (іконка сайту), а також підключено файли стилів, у тому числі файли стилів бібліотек Owl Carousel та Swiper (рис. 2.15).

```
<link rel="shortcut icon" href="./images/favicons/cnst-fav.ico" type="image/x-icon" />
<link rel="stylesheet" href="./owlcarousel/owl.carousel.min.css" />
<link rel="stylesheet" href="./owlcarousel/owl.theme.default.min.css" />
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.css" />
<link rel="stylesheet" href="./css/main.min.css" />
```

Рисунок 2.15 – Підключення необхідних файлів у тезі <head>

Фавікон (іконка сайту), як і інші зображення сайту, розміщений в окремому каталозі для зображень (в даному випадку images), окремо фавікон знаходиться у відповідному каталозі favicons.

Файли Owl Carousel [20] завантажені локально та перенесено до створеного каталогу owlcarousel для цих файлів, така практика є більш надійною.

Стилі Swiper [21] підключено через CDN-посилання, тобто без необхідності у завантаженні файлу, що надає зручності економить час підключення.

Кастомні стилі сайту беруться із згенерованого CSS-файлу завдяки SASS компілятору, а саме посилання йде до мінімізованого(стислого) файлу – це позитивно впливає на оптимізацію сторінки.

Оригіналом файлу main.min.css є файл main.scss, вміст якого наведено у додатку Б. Цей файл є поєднувачем усіх SCSS-файлів для зручної та оптимізованої компіляції, а також конвертування коду у CSS. У цьому файлі через директиву @import підключаються стилі з інших модулів, таких як базові стилі, компоненти, змінні, макети секцій тощо. Це дозволяє структуровано організувати код та легко ним керувати під час розробки.

Такий підхід відповідає патерну проєктування (типового, багаторазового рішення певної проблеми, яка часто виникає під час проєктування програмного забезпечення чи інтерфейсу) Facade (Фасад), оскільки main.scss виступає як єдина точка доступу до всієї системи стилів. Він приховує складну внутрішню структуру проєкту, забезпечуючи простий і зручний інтерфейс для подальшої компіляції та підтримки стилів.

У низу тега <body> підключено необхідні скрипти за допомогою посилань на них (рис. 2.16).

```
<script
  src="https://code.jquery.com/jquery-3.7.1.min.js"
  integrity="sha256-/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo="
  crossorigin="anonymous"
></script>
<script src="https://kit.fontawesome.com/7b20fac7cc.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.js"></script>
<script src="./owlcarousel/owl.carousel.min.js"></script>
<script src="./js/script.js"></script>
```

Рисунок 2.16 – Підключення необхідних скриптів внизу тега <body>

Здійснено підключення JQuery [22], Font Awesome [23] та скрипту Swiper через CDN, а також локальне підключення скрипту Owl Carousel.

2.2.6 Структура каталогу scss. Стилі розміщено в окремий каталог scss, на рисунку 2.17 зображено його структуру.

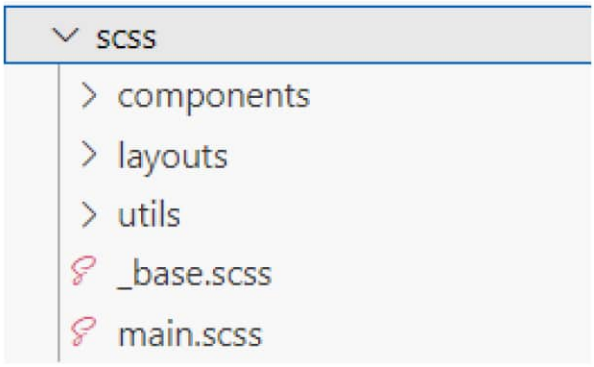


Рисунок 2.17 – Структура каталогу scss

В каталозі layouts створюють файли стилів для окремих частин сайту, наприклад, шапки, секцій, мобільного меню, підвалу. Також layout-структура має файли із багаторазовими класами (рис. 2.18).

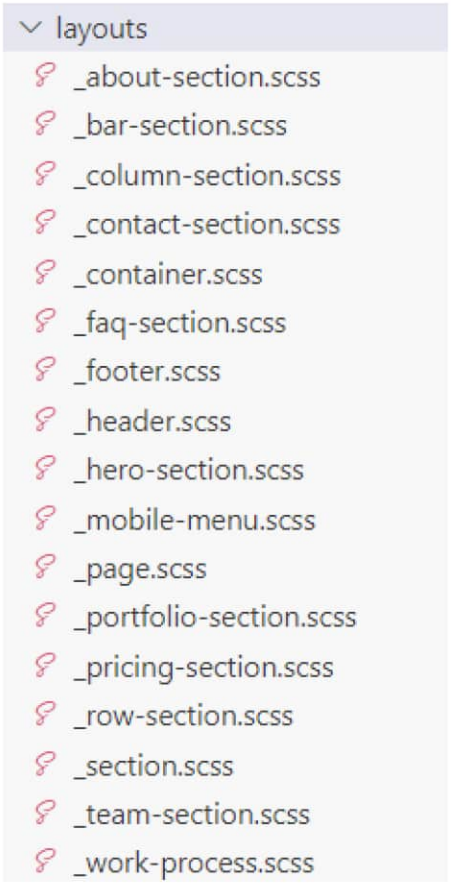
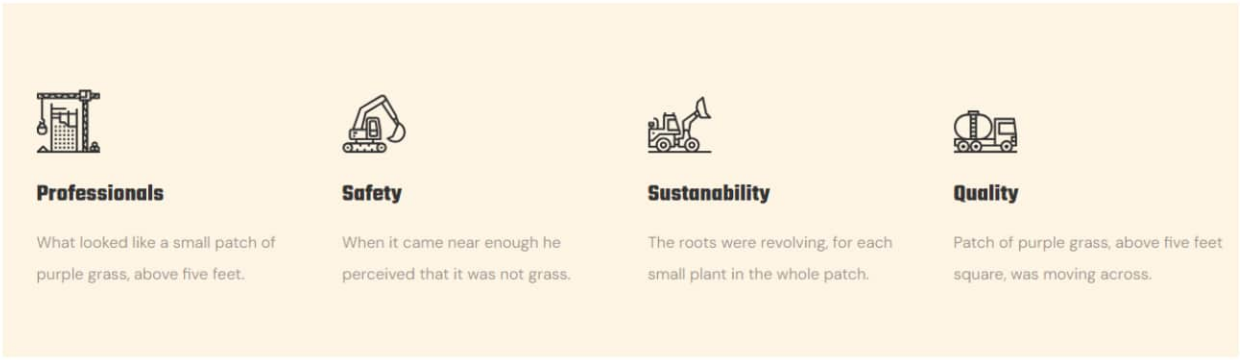
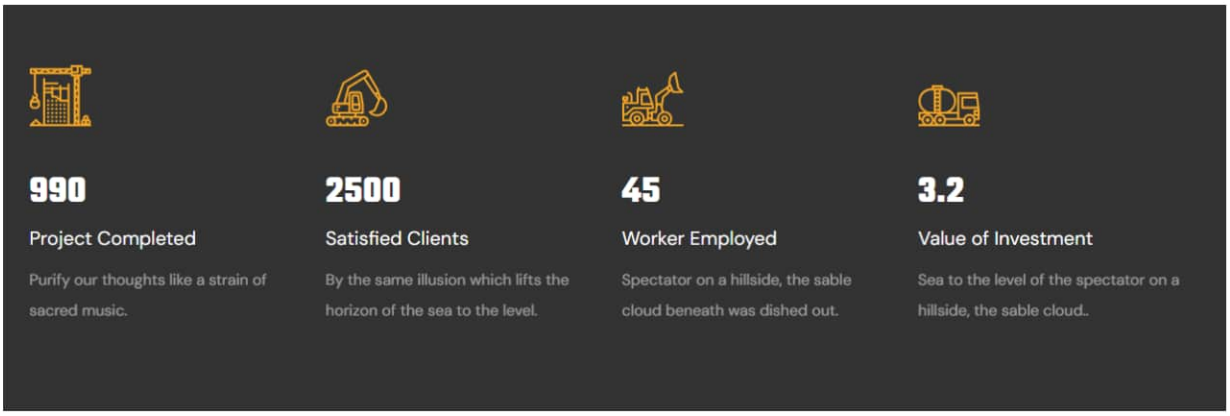


Рисунок 2.18 – Вміст каталогу layouts

Файли _bar-section.scss, _column-section.scss, _row-section.scss і _section.scss створені для зручності та пришвидшення стилізації секцій, бо наявні секції, які стилями між собою схожі, але із певними відмінностями (рис. 2.19). Тому, щоб уникнути дублювання одних і тих самих стилів багато разів для кожного блоку, створюється клас та його вкладені класи, а потім використовуються там, де це потрібно. Такий підхід має ознаки патерна Adapter (Адаптер), оскільки повторювані елементи «адаптуються» до різних контекстів використання без зміни їх структури.



а)



б)

Рисунок 2.19 – Схожі між собою секції, стилізовані класами файлу _bar-section.scss: а) переваги; б) статистика

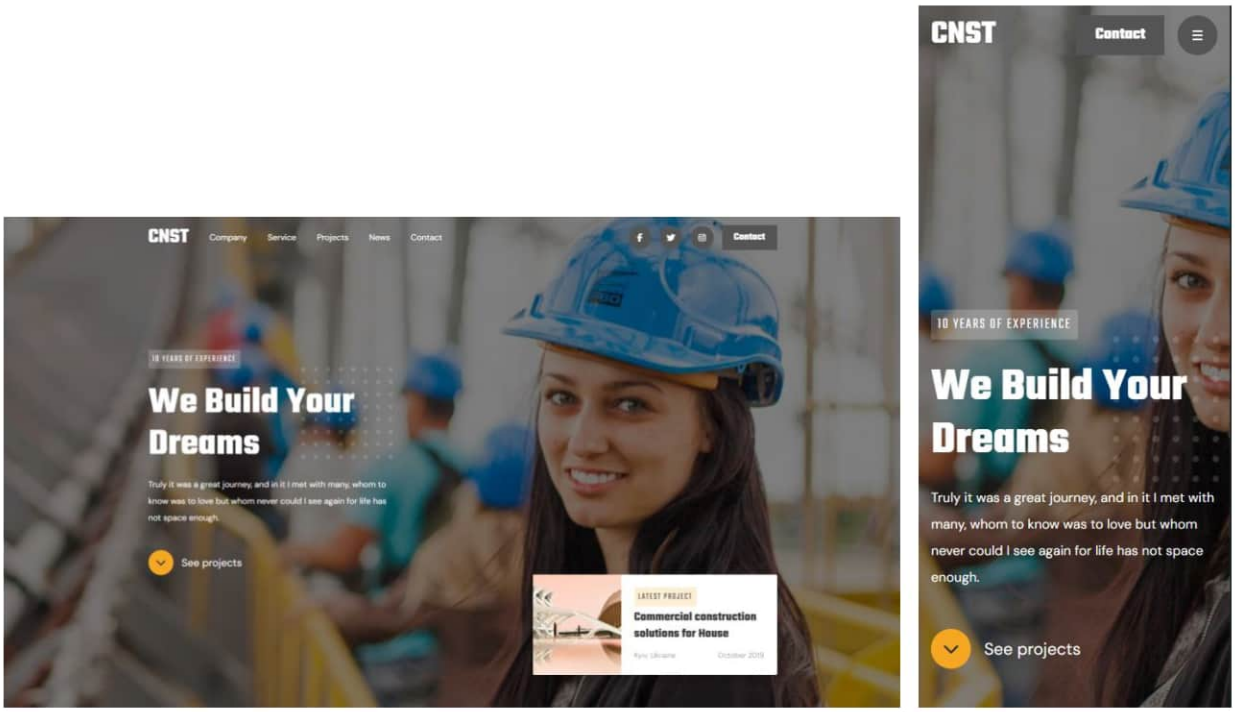
У файлі _container.scss (рис. 2.20) зберігаються стилі створеного обмежуючого контейнеру (.container). Обмежуючий контейнер створюється, щоб обмежити максимальну ширину контенту самої сторінки або конкретної секції, якщо за макетом сторінка не повинна займати всю ширину екрану на будь якому пристрої. За макетом, ширина обмежуючого контейнеру 1210 пікселів, але при встановлені максимальної ширини, потрібно також врахувати відступи зліва та справа – вони уникають «прилипання» до краю екрану та таким чином надають прийнятий вигляд (рис. 2.21). Нижче наведено код стилів цього контейнеру.

_container.scss

scss > layouts > _container.scss > .container

```scss
.container {
  width: 100%;
  max-width: 1270px;
  padding: 0 15px;
  margin: 0 auto;

  @media screen and (min-width: 768px) {
    padding: 0 30px;
  }

  &--hero {
    height: 100%;
    position: relative;
    padding-bottom: 47px;

    @media screen and (min-width: $bp-tablet) {
      padding-bottom: 0;
    }
  }

  &--overflow {
    overflow-x: hidden;
  }
}
```

Рисунок 2.20 – Вміст файлу _container.scss



а) б)

Рисунок 2.21 – Контент шапки та головної секції обмежені по ширині: а) у комп’ютерній версії контент не розтягується на всю ширину екрану; б) у мобільній версії використовуються відступи зліва та справа

Ефективною дією для реалізації вдалої адаптивної верстки є створення та використання технічних класів. Одним із таких класів є клас wrapper, що визначає обгортку сторінки. У блоці із таким класом пишуть усю структуру сторінки. Він чітко вишикує частини сторінки у колонку, що надає надійності при розробці. Така обгортка виконує роль Decorator (Декоратора), бо дозволяє накладати додаткові стилі та впливати на внутрішній вміст, не змінюючи його структури напряму.

Для того, щоб притиснути підвал сайту донизу (щоб він завжди був внизу сторінки, а не залишав білого поля, якщо контенту мало), створюють технічний клас (наприклад .page), який додають до основної частини сторінки (<main>). Ціль цього класу — зайняти все вільне місце між шапкою та підвалом.

Ці класи описані у файлі _page.scss (рис. 2.22).

_page.scss

scss > layouts > _page.scss > ...

```scss
.wrapper {
  min-height: 100%;
  display: flex;
  flex-direction: column;
}

.page {
  flex: 1 1 auto;
}
```

Рисунок 2.22 – Вміст файлу _page.scss

У ситуаціях, коли два елементи мають однакову структуру або вигляд, доцільно не дублювати класи, а винести спільні стилі в окремі блоки. Якщо такий елемент можна використовувати багаторазово, його можна оформити як компонент. Для стилів таких багаторазових блоків створено окремий каталог components (рис. 2.23). Це відповідає принципу патерна Facade (Фасад), де

загальні стилі компонента абстрагують складну структуру і забезпечують єдиний інтерфейс доступу до них.

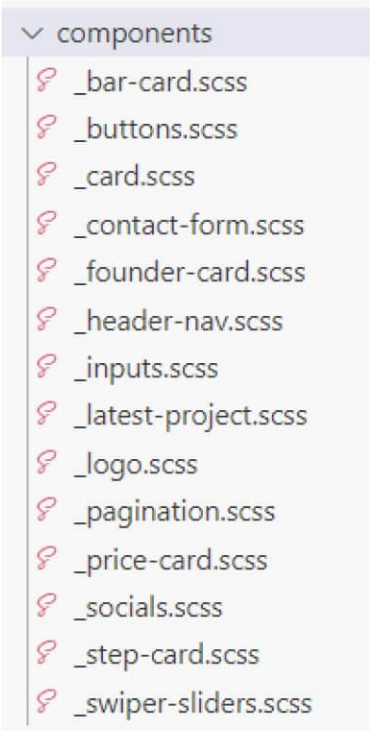


Рисунок 2.23 – Каталог components

2.2.7 Методологія БЕМ. БЕМ (Блок, Елемент, Модифікатор) – це зручна методологія іменування CSS-класів, яка дозволяє зробити код більш структурованим, зрозумілим та легким у підтримці. Головна ідея БЕМ полягає у поділі інтерфейсу на незалежні блоки, які можуть містити елементи та модифікатори.

Методологія БЕМ базується на трьох складових:

- Блок – самостійна, функціонально завершена частина інтерфейсу;
- Елемент – складова частина блоку, яка не може існувати поза ним;
- Модифікатор – властивість або стан блоку чи елемента.

На рисунку 2.24 представлено HTML-структуру десктопної навігації, реалізованої за допомогою БЕМ. Блок .header-nav містить список (.header-nav__list) та посилання (.header-nav__link). Для того, щоб вказати на

десктопний варіант відображення, блоку додано модифікатор .header-nav--desktop.

```
<nav class="header-nav header-nav--desktop">
  <ul class="header-nav__list">
    <li>
      <a href="#about" class="header-nav__link">Company</a>
    </li>
    <li>
      <a href="#service" class="header-nav__link">Service</a>
    </li>
    <li>
      <a href="#portfolio" class="header-nav__link">Projects</a>
    </li>
    <li>
      <a href="#news" class="header-nav__link">News</a>
    </li>
    <li>
      <a href="#contact" class="header-nav__link">Contact</a>
    </li>
  </ul>
</nav>
```

Рисунок 2.24 – HTML-структура десктопної навігації

Використання БЕМ дозволяє легко зрозуміти структуру коду навіть без перегляду HTML. Це також сприяє повторному використанню стилів та зменшенню конфліктів між класами.

Перевагою методології БЕМ є її сумісність із препроцесорами на зразок SCSS, що підтримують вкладеність. Приклад SCSS-коду, в якому реалізовано стилі для навігації, наведено у додатку В. Завдяки вкладеній структурі стилів легко відстежувати зв'язок між блоком, його елементами та модифікаторами. Це значно спрощує підтримку коду та командну роботу над проєктом.

Блок .header-nav є основним контейнером, який займає доступний простір завдяки властивості flex-grow: 1. У межах цього блоку оголошено два модифікатори:

- &--desktop — стилі для десктопної версії, яка за замовчуванням прихована, але відображається при ширині екрану від $bp-desktop (breakpoint для десктопу);

- &--mobile — стилі для мобільної версії, які навпаки, приховуються на десктопних розмірах.

Елемент &__list описує структуру списку навігації. У мобільній версії елементи розташовуються вертикально (flex-direction: column), а на десктопі — горизонтально (flex-direction: row) з більшими відступами між пунктами (gap: 40px).

Елемент &__link відповідає за стилізацію посилань у навігації. Їм задано базовий колір, відсутність підкреслення (text-decoration: none) та плавна зміна кольору при наведенні (:hover).

Така структура SCSS-файлу дозволяє чітко бачити ієрархію елементів, пов'язаних із блоком .header-nav, а також забезпечує гнучкість при розширенні функціоналу або додаванні нових модифікацій.

2.2.8 Реалізація адаптиву сайту. Найпопулярніша технологія, яку використовують розробники для створення адаптивної верстки – це Flexbox. Flexbox – це сучасний спосіб вирівнювання елементів на сторінці. З його допомогою зручно робити адаптивні блоки, будувати сітки або центрувати вміст. Основна ідея цієї технології – це контейнер, в якого значення властивості display стоїть flex, і елементи цього контейнера автоматично вирівнюються в залежності від властивостей, які будуть встановлені. Flexbox сильно спрощує роботу з версткою, особливо коли треба зробити все красиво та рівно.

У п. 2.2.2 була мова про те, що створено SASS-зміні для значень breakpoints (переломні точки). Breakpoints – це точки, при яких змінюється вигляд сайту в залежності від ширин екрану. Їх використовують для створення адаптивної вертски, тобто, щоб сайт виглядав добре на різних пристроях: комп’ютерах, планшетах, телефонах. Для того, щоб застосувати ці breakpoints, потрібно створити медіа-запити (@media), які будуть застосовуватись до блоку/елементу, де потрібно додати адаптив. Взагалі, медіа-запити – це частина CSS, яка дозволяє змінювати елементи залежно від характеристик пристрою: ширини екрану, висоти, орієнтації, роздільної здатності, тощо.

Вони є основним інструментом для створення адаптивної верстки. Найчастіше використовують медіа-запити залежно від ширини екрану, тому майже всі створені запити такі. Як приклад, у лістингу додатка В, використовуються технологія Flexbox для розташування елементів навігації та медіазапити при мінімальній ширині екрану від значення $bp-desktop (опис коду є у п. 2.2.7).

Напрямок flex-елементів та відстань між ними змінюється від значення 992 пікселя.

2.2.9 Ретинізація фонового зображення головної секції. Ретинізація – це процес підготовки зображень для екранів з високою щільністю пікселів (наприклад Retina-дисплеїв), щоб вони виглядали більш чіткими та не розмитими. У чому суть: на екранах з високим PPI (пікселів на дюйм) звичайне зображення може виглядати розмитим, тому розробники використовують зображення в 2-3 рази більшої роздільності, але показують їх у стандартному розмірі.

Серед стилів присутній один медіа-запит, який змінює фонове зображення головної секції на зображення з більшою роздільністю в 2 рази, в залежності від роздільної здатності екрану. Приклад такого запиту наведено на рисунку 2.25.

```
@media screen and (min-device-pixel-ratio: 2),
  (min-resolution: 192dpi),
  (min-resolution: 2dppx) {
  background-image: linear-gradient(
      0deg,
      rgba(0, 0, 0, 0.5) 50%,
      rgba(0, 0, 0, 0.5) 50%
    ),
    url("../images/hero-bg@2x.jpg");
}
```

Рисунок 2.25 – Медіазапит для ретинізації фонового зображення головної секції (.hero-section)

Це медіа-запит, який активується на екранах із високою щільністю, тобто Retina-дисплеях для старих (наприклад Safari) та сучасних браузерів(вимірює роздільність у точках на дюйм та щільності пікселів(2x = Retina)).

2.3 Реалізація функціональних елементів

2.3.1 Меню навігації. На рисунку 2.24 було наведено код десктопного меню навігації у шапці сайту (рис. 2.26). Навігація має маркований список, в якому 5 пунктів списку. В кожному пункті списку знаходиться посилання, яке буде посилатись на відповідну секцію сторінки. Такі посилання називаються якорями.

Щоб спрацював якір, потрібно надати секції унікальний ідентифікатор(id) і у якості посилання ввести цей селектор ідентифікатору(#назва_ідентифікатору).

При натисканні на будь який пункт меню, відбувається скрол до секції, куди посилається цей пункт, а при наявності у тега <html> властивості scroll-behavior зі значенням smooth на комп'ютері (див. додаток А), цей скрол працює плавно.

Рисунок 2.26 – Шапка сайту

Навігація також присутня і у мобільному меню.

Виявлено проблему, що при скролі униз, шапка зливається із самим сайтом. Тому, було прийнято рішення надати колір фону шапки при скролі на деяку відстань униз(рис. 2.27).

Рисунок 2.27 – Шапка сайту із кольоровим фоном

Нижче наведено фрагмент скрипту із реалізацією цієї ідеї(рис. 2.28).

```
$(window).on("scroll", function () {
  if ($(window).scrollTop() > 100) {
    $(".header").addClass("scrolled");
  } else {
    $(".header").removeClass("scrolled");
  }
});
```

Рисунок 2.28 – Механіка роботи анімації

До вікна браузера додаємо слухача подій scroll та спрацьовує функція, яка має умову: якщо вікно браузера скролиться у верх на більше ніж 100 пікселів, то до шапки додається клас scrolled, який привласнює колір фону, в інакшому випадку видаляє цей клас. Таким чином використовується патерн Декоратор (Decorator), який відповідає за зміну кольору фону шапки. Фрагмент коду стилей цього класу (вкладено у клас header) зображено на рисунку 2.29.

```
&.scrolled {
  background-color: var(--color-default-text);
}
```

Рисунок 2.29 – Допоміжний клас scrolled

У цьому випадку використовується патерн Спостерігач (Observer): вікно браузера виступає у ролі суб’єкта, що генерує подію scroll, а функція-обробник є спостерігачем, який реагує на цю подію та змінює DOM. Такий підхід

дозволяє реалізувати гнучке реагування на дії користувача без жорсткого зв’язування між компонентами.

2.3.2 Вставка відео про компанію та мапи локації. На сторінці додано відео з YouTube (рис. 2.31) та мапа Google Maps (рис. 2.33), вставлене за допомогою тегу <iframe>. Це спеціальний HTML-тег, який дозволяє вбудувати сторонній вміст, такий, як відео, мапу чи іншу сторінку прямо на сайт. Отриманий код фрейму відео з YouTube виглядає наступним чином, як на рисунку 2.30.

```
<iframe
  class="about-section__iframe"
  src="https://www.youtube.com/embed/uTE5MKwUz0A?si=p3RUrnfk4aSLy7qt"
  title="YouTube video player"
  frameborder="0"
  allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"
  referrerpolicy="strict-origin-when-cross-origin"
  allowfullscreen
></iframe>
```

Рисунок 2.30 – Отриманий фрейм відео з YouTube

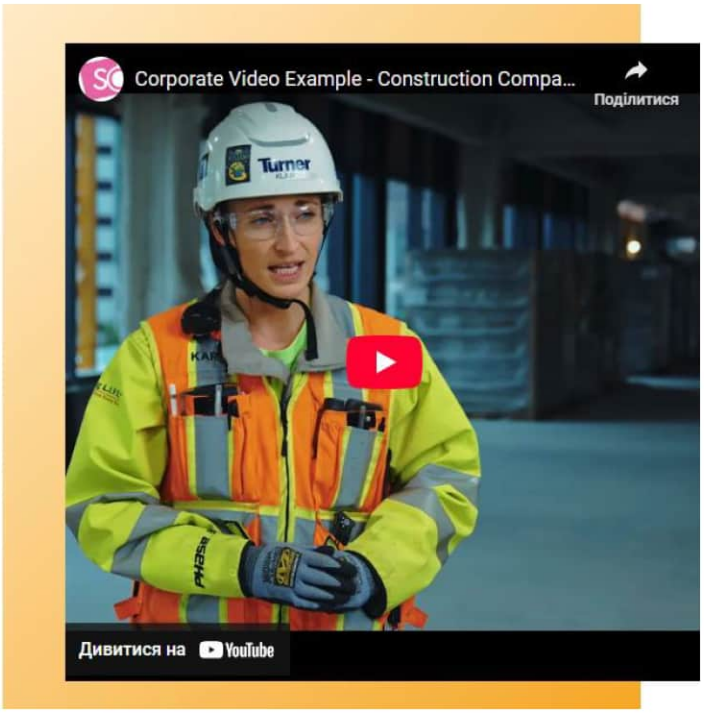


Рисунок 2.31 – Відео з YouTube, додане за допомогою iframe

Отриманий код фрейм мапи Google Maps із локацією компанії зображено на рисунку 2.32.

```
<iframe
  class="contact-section__map-iframe"
  src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d162735.93228222698!2d30.33043386
  allowfullscreen=""
  loading="lazy"
  referrerpolicy="no-referrer-when-downgrade"
></iframe>
```

Рисунок 2.32 – Отриманий фрейм відео з Google Maps

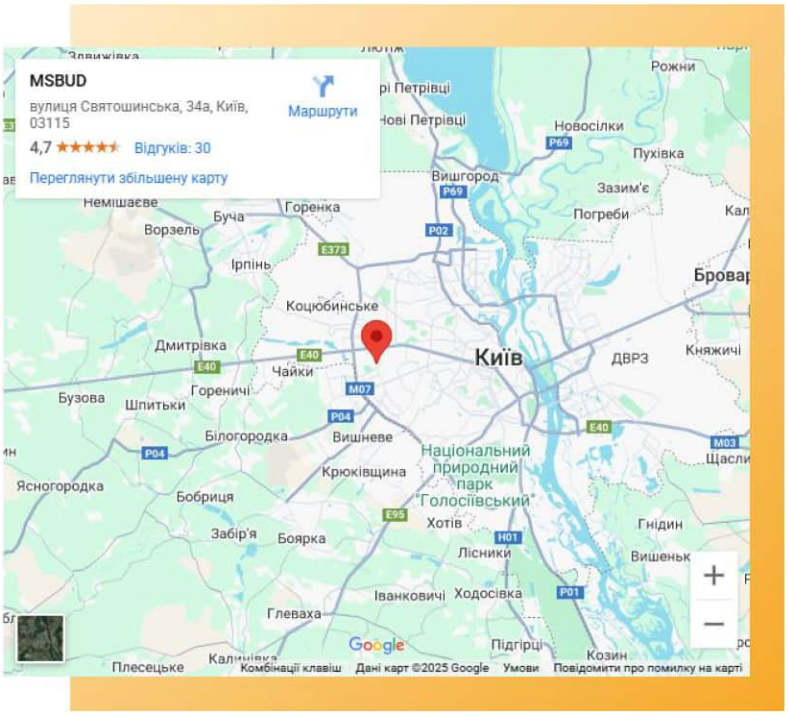


Рисунок 2.33 – Мапа з Google Maps, додана за допомогою iframe

2.3.3 Слайдери. Бібліотеки для створення слайдерів мають набір готових класів, які визначають блок та певні їх елементи, а також стилізують їх відповідно.

Для базової стилізації слайдера за допомогою Owl Carousel у HTML-файлі достатньо привласнити два класи основному блоку майбутнього слайдера: .owl-carousel (вказує, що саме цей блок є слайдером) та .owl-theme (додає базові стилі, які формують вигляд слайдера).

У додатку Г наведено HTML-код слайдера «Сервіс» (рис. 2.34), який реалізовано за допомогою бібліотеки Owl Carousel. Основний контейнер слайдера має класи row-slider, owl-carousel та owl-theme. Клас owl-carousel

активує бібліотеку, owl-theme додає базову стилізацію, а row-slider є кастомним класом для подальшої стилізації через SCSS.

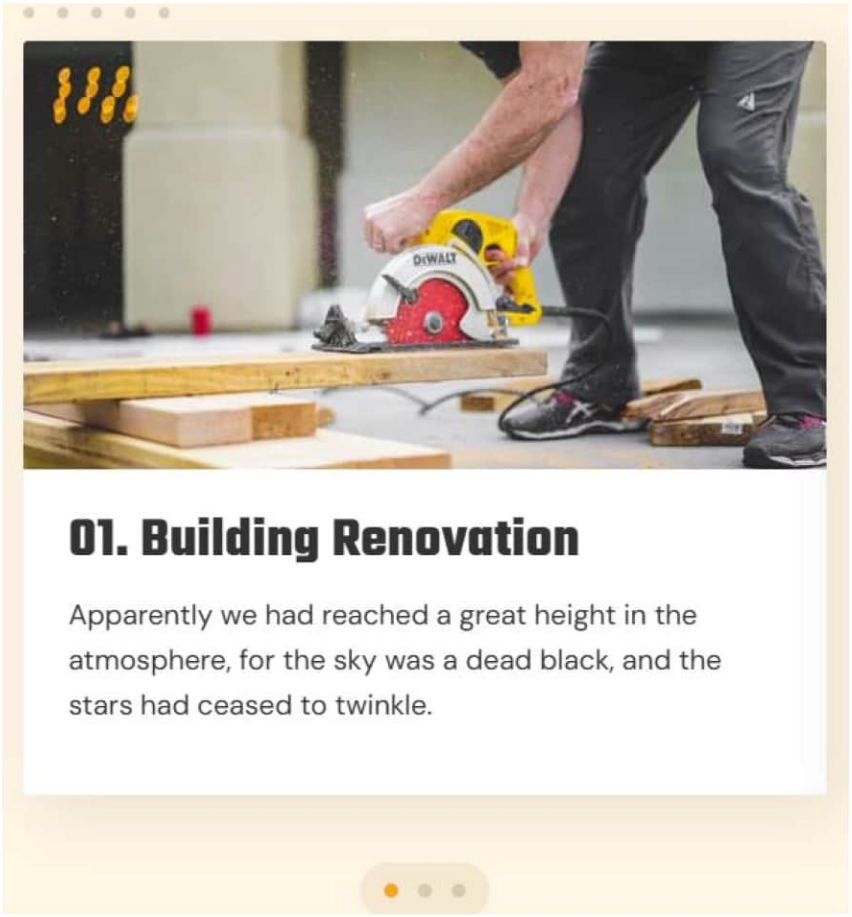


Рисунок 2.34 – Слайдер «Сервіс»

Слайдер складається з кількох блоків div з класом card, які є окремими слайдами. Кожна карточка містить зображення (img) та текстовий вміст у блоці card__content. Всередині цього блоку розміщені елементи p з класами card__title (із модифікатором card__title--service) та card__text.

Клас card__title--service слугує для стилізації заголовків саме в межах секції «Сервіс», що дозволяє відокремити їх від інших заголовків на сайті. Завдяки такій структурі можна легко налаштувати зовнішній вигляд кожного слайда, не дублюючи стилі для різних секцій.

Усі слайди мають однакову структуру, тому бібліотека коректно їх обробляє, створюючи адаптивний слайдер, що перемикається автоматично.

Зображення, текст та заголовки налаштовані так, щоб візуально гармонійно поєднуватись на різних розмірах екранів.

Проте цей блок не буде працювати, якщо слайдер не активувати за допомогою JavaScript. Для цього необхідно викликати метод owlCarousel(), передавши до нього об'єкт з налаштуваннями. Ключі цього об'єкта відповідають параметрам налаштування, а значення визначають поведінку слайдера.

На рисунку 2.35 представлений приклад ініціалізації слайдера класу .row-slider:

- items: 1 — одночасно показується один слайд;
- nav: false — навігаційні стрілки приховано;
- autoplay: true — слайди перемикаються автоматично;
- autoplayTimeout: 3000 — час між переключеннями слайдів становить 3 секунди;
- loop: true — після останнього слайду знову починається перший (циклічність).

```
$(".row-slider").owlCarousel({
  items: 1,
  nav: false,
  autoplay: true,
  autoplayTimeout: 3000,
  loop: true,
});
```

Рисунок 2.35 – Ініціалізація роботи слайдеру «Сервіс»

У цьому випадку використовується патерн проєктування «Стратегія» (Strategy): об'єкт з параметрами конфігурації передається у функцію, яка змінює поведінку слайдера без необхідності змінювати саму реалізацію плагіна.

Бібліотека Swiper має основні класи, які визначають головний блок (контейнер), кожен елемент слайдера (обгортка, слайд), а також додаткові елементи, такі як пагінація та/або навігація.

HTML-структура слайдера «Команда» (рис. 2.36) представлена у додатку Г. Головний контейнер має класи team-section__carousel та swiper. Всередині нього знаходиться обгортка swiper-wrapper, яка містить окремі слайди з класом swiper-slide. Кожен слайд є карткою (card), що включає зображення учасника команди та текстовий блок з ім'ям та посадою. Наприклад, картка включає зображення з класом card__image, заголовок з класом card__title та опис посади з класом card__type. Наприкінці контейнера розміщено елемент для пагінації з класом swiper-pagination.

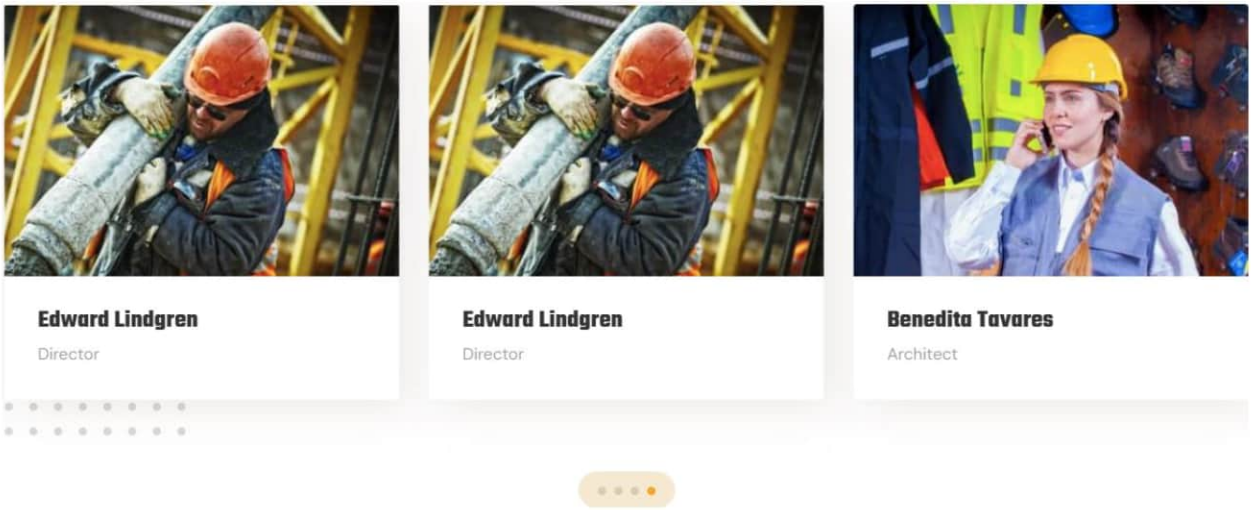


Рисунок 2.36 – Слайдер «Команда»

Таким чином, код HTML повністю відповідає вимогам ініціалізації слайдера, що дозволяє створити адаптивний, зручний у використанні та привабливий для користувача слайдер команди.

Для того, щоб активувати слайдер, потрібно оголосити змінну, значенням якої є новий об'єкт на основі класу Swiper, який надає сама бібліотека (рис. 2.37). Цей об'єкт повинен мати два аргументи:

- HTML-елемент, до якого потрібно під'єднатись (контейнер);
- Об'єкт, який містить властивості слайдера та їх значення.

```
const swiper = new Swiper(".swiper", {
  loop: true,
  slidesPerView: 1,
  spaceBetween: 30,
  breakpoints: {
    600: {
      slidesPerView: 2,
    },
    992: {
      slidesPerView: 3,
    },
  },
  autoplay: {
    delay: 3000,
  },
  pagination: {
    el: ".swiper-pagination",
    clickable: true,
  },
});
```

Рисунок 2.37 – Ініціалізація слайдеру «Команда»

Слайдер «Команда» пролистується автоматично з інтервалом у 3 секунди та циклічно. За замовчуванням, видима кількість слайдів становить 1. При ширині екрана від 600 пікселів – 2 слайди, від 992 пікселів – 3 слайди. Відстань між слайдами становить 30 пікселів. Також реалізована клікабельна пагінація, за яку відповідає елемент з класом .swiper-pagination (рис. 2.36).

У цьому випадку застосовується патерн проєктування «Фабричний метод» (Factory Method), оскільки створення об’єкта слайдера відбувається через виклик конструктора бібліотеки Swiper з певними параметрами. Таким чином, логіка створення об'єкта відокремлена від клієнтського коду, а конфігурація поведінки слайдера передається через аргументи, що забезпечує гнучкість та повторне використання.

2.3.4 Фільтрація проєктів портфоліо за категоріями. Секція «Портфоліо» є важливим елементом сайту будівельної компанії, оскільки дозволяє потенційним клієнтам ознайомитися із реалізованими проєктами та

оцінити професійні можливості компанії. Саме портфоліо часто є основою при прийнятті рішення клієнтом про співпрацю.

У секції «Портфоліо» (рис. 2.38) представлено натискні яскраві картки з зображеннями та описом об'єктів (структура цього блоку наведено у додатку Г). Кожен проєкт має свою категорію, а саме: будівництво, будівельні роботи або проєктне планування. Для кожної з цих категорій виведено кнопки фільтра, що дозволяють швидко перемикатись між проєктами за обраною категорією.

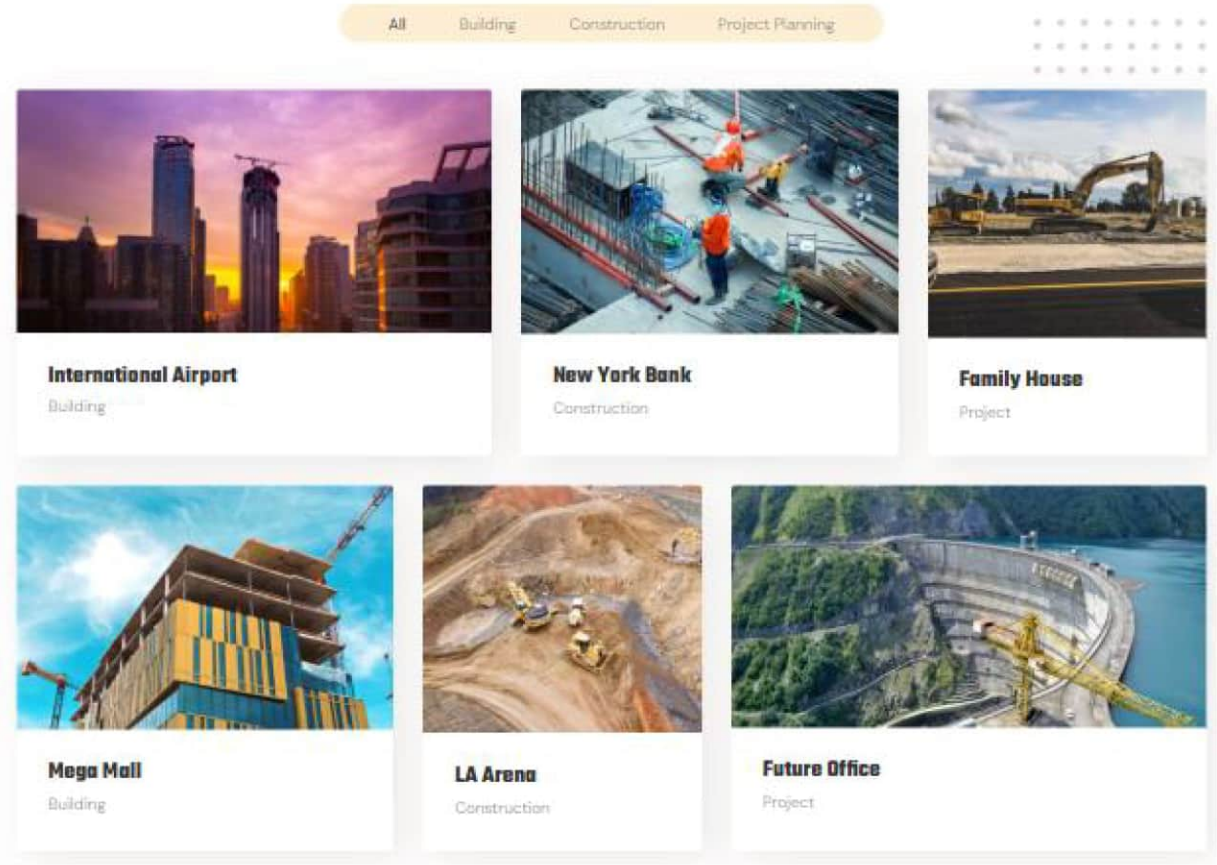


Рисунок 2.38 – Портфоліо компанії

При натисканні на кнопку, спрочиняється скрипт (рис. 2.39), що обробляє значення вибраної категорії. Відбувається функція, де створюється зміна зі значенням дата-атрибуту filter натиснутої кнопки. Зі всіх кнопок видаляється клас button-active, який змінює колір тексту кнопки, а до натиснутої цей клас додається. Якщо значення дата-атрибуту filter є all, то до

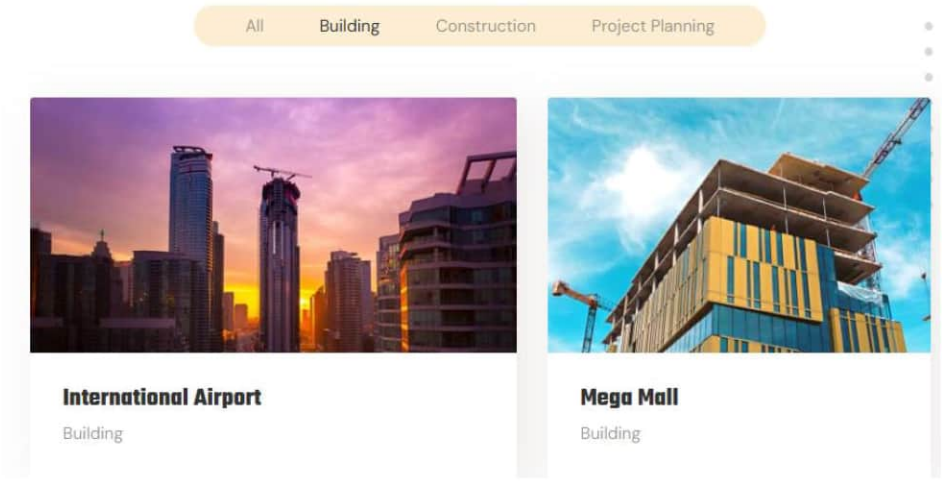
всіх карток проєктів з класом card видаляється клас hidden, який ховає картку. В інакшому випадку, тобто, якщо обрано фільтр, йде перебір всіх карток. Якщо картка має відповідний клас(він називається так, як називається дата-атрибут), то вона залишається видимою, якщо не має, додається клас hidden.

У цьому випадку використано патерн проєктування «Стратегія» (Strategy). Залежно від обраного варіанта фільтрації, динамічно змінюється логіка відображення елементів. Вибір фільтра визначає, яка стратегія (показати всі або показати вибрані) застосовується до колекції елементів без зміни самої структури обробника.

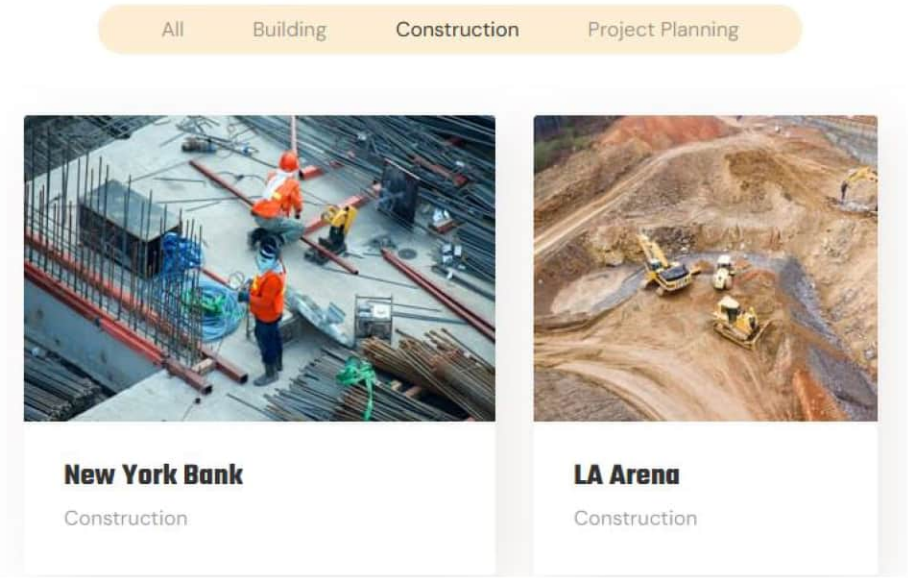
```
$(".portfolio-section__button").on("click", function () {
  const filter = $(this).data("filter");

  $(".portfolio-section__button").removeClass("button-active");
  $(this).addClass("button-active");

  if (filter === "all") {
    $(".card").removeClass("hidden");
  } else {
    $(".card").each(function () {
      $(this).toggleClass("hidden", !$(this).hasClass(filter));
    });
  }
});
```

Рисунок 2.39 – Слухач події click для кнопок фільтру

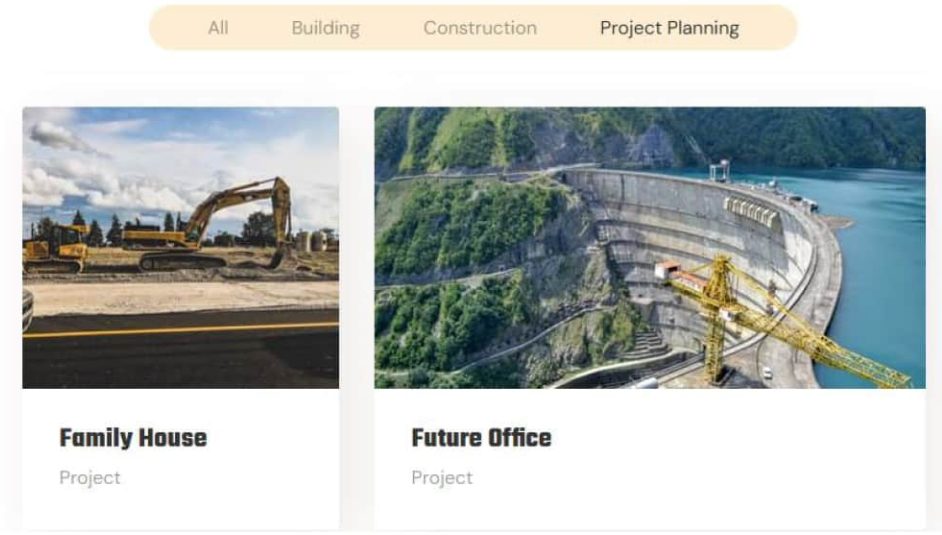
Такий підхід дозволяє зробити інтерфейс сайту більш зручним та інтуїтивним для користувача, дозволяє швидко знайти потрібний проєкт та скласти враження про реалізовані об’єкти компанії. Результат роботи фільтру показано на рисунках 2.40.



а)



б)



в)

Рисунок 2.40 – Робота фільтру по категоріям: а) Будівництво; б) Будівельні роботи; в) Проєктне планування

2.3.5 Акордеон F.A.Q. Акордеон – це елемент інтерфейсу, який дозволяє відкривати та закривати блоки з текстом. Найчастіше акордеон використовують для секцій із частими запитаннями («F.A.Q.»), списками інструкцій або для мобільного меню. У цьому проєкті акордеон використано в секції «F.A.Q.», де кожне питання відкриває приховану відповідь після натискання на заголовок (рис. 2.41).

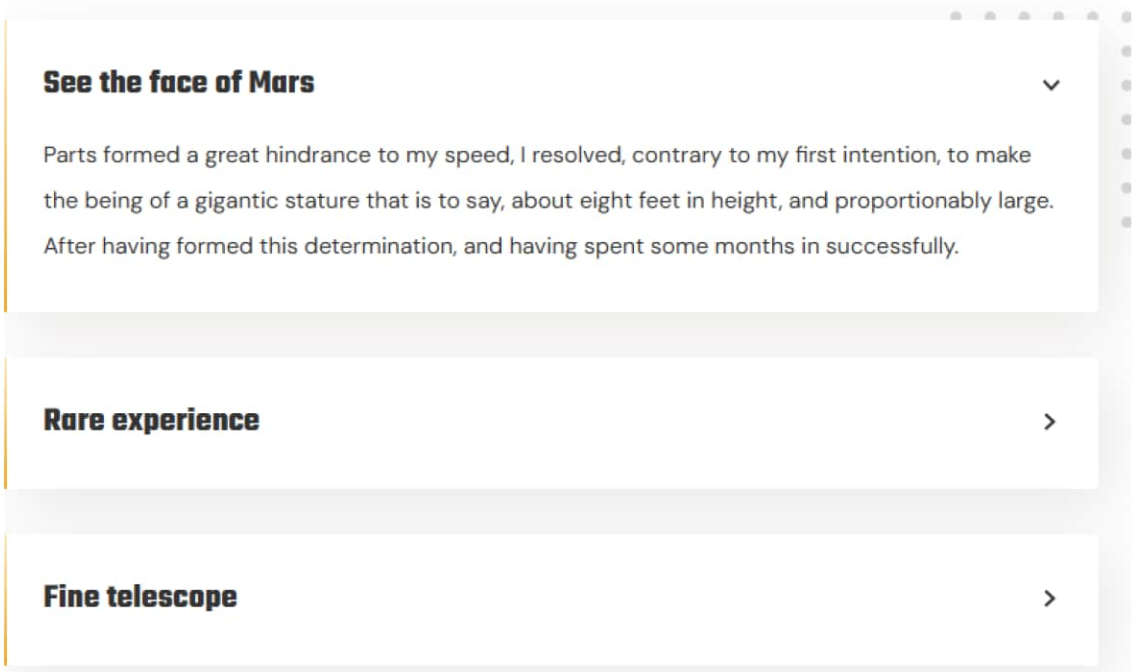


Рисунок 2.41 – Акордеон «*F.A.Q.*»

Якщо подивитися на структуру HTML-коду (див. додаток Г), то можна побачити, що весь акордеон складається з кількох блоків з однаковою структурою. Кожен з цих блоків називається «спойлер». Усередині кожного спойлера є заголовок – це частина, на яку потрібно натискати, та прихований текст з відповіддю. Заголовок включає не лише текст, а й маленьку стрілочку (іконку SVG), яка повертається на 90 градусів, коли спойлер відкривається.

Увесь текст відповіді за замовчуванням прихований, тобто користувач його не бачить при першому завантаженні сторінки. При натисканні на заголовок вмикається скрипт (рис. 2.42), який додає або забирає клас "rotate" у блоку зі стрілочкою. Цей клас відповідає за обертання стрілки, завдяки чому

вона красиво повертається вправо. Поворот відбувається плавно, тому що в стилях додано анімацію через властивість transition зі значенням 400 мс (рис. 2.43). Далі скрипт знаходить текст спойлера (елемент, який йде після заголовка) і за допомогою методу slideToggle() показує або ховає його з плавною анімацією, тривалістю за замовчуванням у 300 мс.

У цьому випадку реалізується патерн проєктування «Стан» (State). Залежно від поточного стану елемента (відкритий або закритий), скрипт змінює його вигляд та поведінку — обертає стрілку і розгортає або приховує текст. Клас rotate та функція slideToggle() динамічно відображають зміну стану, що дозволяє користувачеві зручно взаємодіяти з інтерфейсом.

```
$(".faq-section__text").hide();

$(".faq-section__spoller-title").click(function () {
  $(this).find(".faq-section__spoller-btn").toggleClass("rotate");
  $(this).siblings(".faq-section__text").slideToggle();
});
```

Рисунок 2.42 – Скрипт роботи акордеону

```
&__spoller-btn {
  transition: transform 0.4s;

  &.rotate {
    transform: rotate(90deg);
  }
}
```

Рисунок 2.43 – Стилі для анімації стрілочки «спойлеру»

При натисканні на заголовок, текст розкривається вниз, а стрілочка обертається, щоб показати, що блок відкритий. Якщо знову натиснути – все закривається. Таким чином реалізовано зручний і зрозумілий для користувача інтерфейс взаємодії з частими питаннями.

Це рішення дозволяє заощадити місце на сторінці, не показуючи одразу всі відповіді, а лише ті, які цікавлять користувача. Така реалізація часто зустрічається на сучасних сайтах, адже вона проста, зручна та ефективна в користуванні.

2.3.6 Форма зворотного зв'язку. Для того, щоб зв’язатись із компанією, потрібно залишити заявку, заповнивши дані відповідної форми (рис. 2.44). Форма зворотного зв'язку додається до сторінки, щоб компанія змогла отримати дані клієнта та зв’язатись із ним для подальшого діалогу.

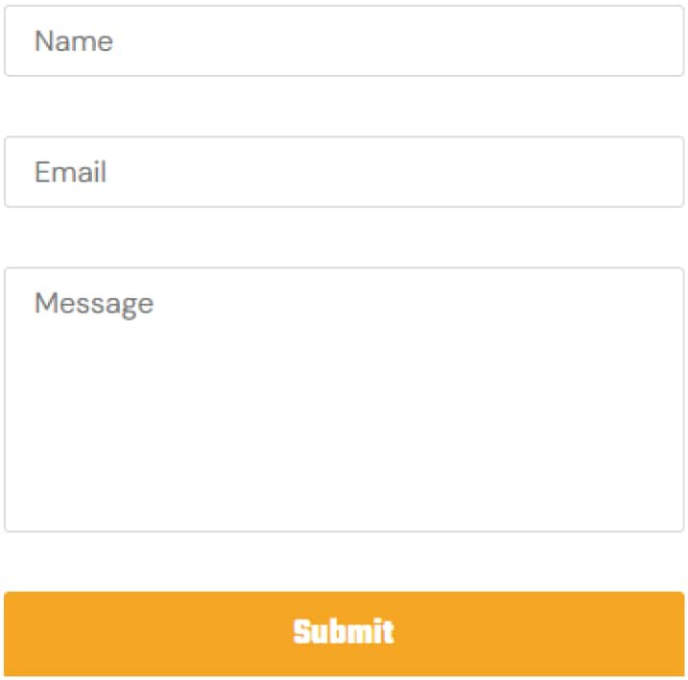


Рисунок 2.44 – Форма зворотного зв’язку секції «Зв’язатись»

На рисунку 2.45 наведено структуру цієї форми.

```
<form name="contact-form" autocomplete="off" class="contact-section__form" id="contact-form">
  <!-- prettier-ignore -->
  <input type="text" name="user-name" id="user-name" placeholder="Name" class="control control--input"/>
  <div class="form-message" id="form-name"></div>
  <!-- prettier-ignore -->
  <input type="email" name="user-email" id="user-email" placeholder="Email" class="control control--input"/>
  <div class="form-message" id="form-email"></div>
  <!-- prettier-ignore -->
  <textarea name="user-message" id="user-message" class="control control--textarea" placeholder="Message"></textarea>
  <div class="form-message" id="form-message"></div>

  <button class="contact-section__btn btn" type="submit">Submit</button>
</form>
```

Рисунок 2.45 – Структура форми

HTML-код форми складається з трьох основних полів для введення: поля для імені, електронної пошти та текстового повідомлення. Під кожним полем знаходиться блок з класом form-message, який використовується для виведення повідомлення про помилку у випадку некоректного заповнення. Також у формі присутня кнопка для підтвердження та надсилання даних.

Форма проходить валідацію перед надсиланням. Валідація — це процес перевірки введених користувачем даних на відповідність певним правилам. Це важливо, адже коректні дані дозволяють менеджеру компанії швидше обробити заявку та зв'язатися з клієнтом.

Усі перевірки виконуються на стороні клієнта за допомогою JavaScript. Коли користувач натискає кнопку надсилання, стандартна дія форми скасовується за допомогою preventDefault(). Далі скрипт отримує значення з усіх трьох полів, очищує їх від зайвих пробілів за допомогою trim() і перевіряє за умовами:

- поле з іменем не повинно бути порожнім та має містити щонайменше 2 символи.
- поле з email не повинно бути порожнім та має відповідати правильному формату адреси. Це перевіряється функцією validateEmail(), яка використовує регулярний вираз для визначення правильності написання електронної пошти.
- поле з повідомленням не повинно бути порожнім та повинно містити щонайменше 10 символів.

Якщо якесь із полів заповнено неправильно, відповідний блок form-message виводить повідомлення про помилку. Якщо ж усі поля заповнено вірно, ці повідомлення очищуються, і форма готова до подальшої обробки.

Щоб запобігти повторному натисканню кнопки відправки під час обробки даних, кнопка тимчасово блокується за допомогою disabled, а її текст змінюється на "Sending...". Це дозволяє уникнути дублювання заявок і покращує досвід взаємодії користувача з формою.

Валідація форми реалізована у вигляді обробника події submit на формі з ідентифікатором #contact-form. Повний лістинг коду наведено у додатку Д.

2.3.7 Анімації. На сайті реалізовано два види анімацій: за допомогою стилів та за допомогою скриптів.

До простих анімацій належать ті, які реалізуються через зміну властивостей стилю. Найрозповсюдженіший приклад — це зміна зовнішнього вигляду елементів при наведенні миші. Наприклад, на сайті реалізовано зміну кольору кнопок за допомогою псевдокласу :hover. Щоб ці зміни виглядали плавно, до кнопок і посилань у базових стилях (див. додаток А) додано властивість transition. Вона забезпечує плавну зміну кольору фону та тексту тривалістю 200 мс з ефектом ease-in.

Динамічніші анімації на сайті реалізовані за допомогою API Intersection Observer у поєднанні з бібліотекою jQuery. Це дозволяє відслідковувати, коли елемент потрапляє в область видимості екрана користувача. Приклад такої анімації — поява контейнерів у кожній секції та анімація чисел у секції «Статистика» (рис. 2.46).

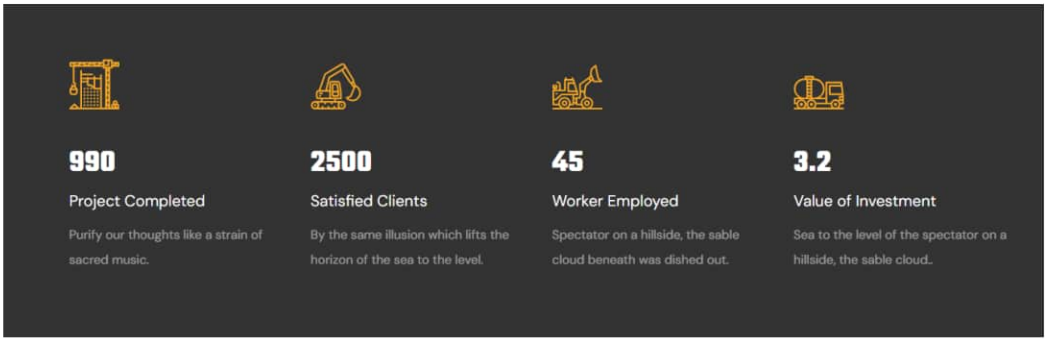


Рисунок 2.46 – Секція «Статистика»

Створюється функція animateCounters() (див. додаток Д), яка знаходить усі елементи з класом .count і анімує їх від значення 0 до числа, яке вказано в атрибуті data-target. В ній створюється JQuery-анімація числа – поступове оновлення тексту елемента .count від 0 до data-target.

За допомогою Intersection Observer запускається анімація лічильників, а потім відключається спостереження після показу. Тобто, API спостерігає за

певними елементами і коли хоча б 10% елемента видно на екрані, запускається функція, в якій:

1. до елемента, за яким спостерігається, додається клас visible (рис. 2.47);
2. викликається функція animateCounters();
3. вимикається спостереження, щоб воно не повторювалось.

При готовому DOM, до контейнера .container додається клас animate (рис. 2.47), тим самим помічається, що він буде анімуватись.

```
.animate {
  opacity: 0;
  transform: translateY(40px);
  transition: all 0.8s ease-out;
}

.animate.visible {
  opacity: 1;
  transform: translateY(0);
}
```

Рисунок 2.47 – Кастомний клас animate та додатковий клас visible

Клас має властивості такі, як прозорість 0%, зміщення по осі Y на 40 пікселів та зміни всіх властивостей триватимуть 800 мс.

Про те, робиться виключення для контейнера головної секції. Додається цей клас до нього через 300 мс і потім відбувається спостереження за всіма контейнерами, окрім контейнера, який має модифікатор container—hero, та спостереження за секцією Статистика.

У цьому випадку використовується патерн проєктування «Спостерігач» (Observer). Intersection Observer виступає в ролі механізму підписки на появу елемента в зоні видимості, після чого повідомляє про це відповідну функцію, яка запускає анімацію. Такий підхід дозволяє відокремити логіку виявлення видимості елементів від їх анімації, роблячи код більш гнучким і модульним.

Також існує підхід створення анімацій у JQuery за допомогою асинхронного методу setInterval(), який виконує певну функцію за поставлений інтервал часу. Реалізовано декоративну анімацію у секції «Як ми працюємо»(рис. 2.48), де цятки, які помічають певний етап роботи змінюють по черзі свій колір та додається, так би мовити, «ефект підсвічування», що надає більший акцент на кожному етапі.

Функція changeStepColor() (див. додаток Д) додає клас active до поточного елементу, починаючи з першого, а в інших елементах цей клас видаляє, і переходить до наступного. Запускаємо функцію через кожні 2 секунди.

Рисунок 2.48 – Секція «Як ми працюємо»

У цій реалізації реалізовано патерн "Ітератор" (Iterator). Кожен крок (етап) обробляється по черзі у циклічному порядку, і setInterval() виступає як механізм керування ітерацією — послідовно змінюючи стан кожного елементу у встановленому ритмі.

2.3.8 Мобільне меню. Планшетна та мобільна версії сайту мають мобільне меню. Це один із підходів до адаптиву, який застосовується як вирішення проблеми, коли пункти меню у шапці починають прилипати один до одного при зменшенні екрану. Тому в такому випадку пункти меню

ховаються та з’являється кнопка для відкриття такого меню (рис. 2.49), його ще називають «бургером» через те, що зображення типових кнопок схожі на бургер.

Рисунок 2.49 – Шапка сайту(мобільна версія)

Мобільне меню сайту (рис. 2.52) також має навігацію з такими самими пунктами, як і шапка, і посилання на соц мережі(в шапці вони також сховані). На рисунку 2.50 реалізовано механіку відкриття та закриття меню.

```
$(".header__burger-btn").on("click", function () {
  $(".mobile-menu").addClass("is-open");
  $("body").css("overflow", "hidden");
});
```

```
$(".mobile-menu__btn-close, .header-nav__link").on("click", function () {
  $(".mobile-menu").removeClass("is-open");
  $("body").css("overflow", "");
});
```

Рисунок 2.50 – Відкриття та закриття мобільного меню

При кліці на кнопку «бургер»-меню, до мобільного меню додається клас is-open (рис. 2.51), який відкриває меню, а також при уникненні скролу сторінки, до тіла HTML-документу, додається властивість overflow зі значенням hidden, що блокує скрол як по вертикалі, так і по горизонталі.

```
&.is-open {
  transform: translateX(0);
}
```

Рисунок 2.51 – Кастомний клас is-open

Мобільному меню привласнено зміщення по осі X на 100%, тому його не видно, а при додаванні до нього класу is-open, меню повертається на своє місце.

При кліці на кнопку закриття меню, яка знаходиться у самому меню, або по пункту меню, з мобільного меню видаляється клас is-open та прибирається значення властивості overflow з тіла документу.

У цьому випадку реалізується патерн «Команда» (Command). Кожна дія — відкриття або закриття меню — виконується як окрема команда, яку можна викликати у відповідь на подію (натискання кнопки). Такий підхід дозволяє централізовано керувати поведінкою інтерфейсу через чітко визначені дії.

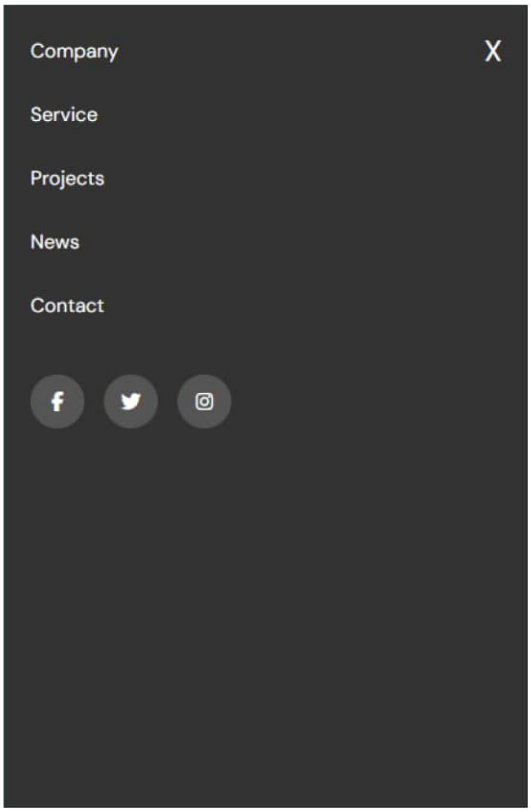


Рисунок 2.52 – Мобільне меню

Оскільки дизайну для мобільного меню на макеті не було, то реалізація була виконана інтуїтивно, показавши максимальний сенс цього надзвичай важливого елементу у мобільній версії сторінки.

2.4 Хостинг сторінки

Хостинг – це послуга з розміщення файлів і даних вебсайту на віддаленому сервері, який постійно підключений до Інтернету. Завдяки хостингу будь-який користувач світу може відкрити сайт за його URL-адресою.

Для розміщення готового сайту використано хостинг Україна [25], в якому вже зареєстровано окремий домен firemoretest.space. Надано власником домену доступ до його панелі управління ADM.TOOLS та створено окремий сайт (рис. 2.53), в який можна завантажувати самі файли.

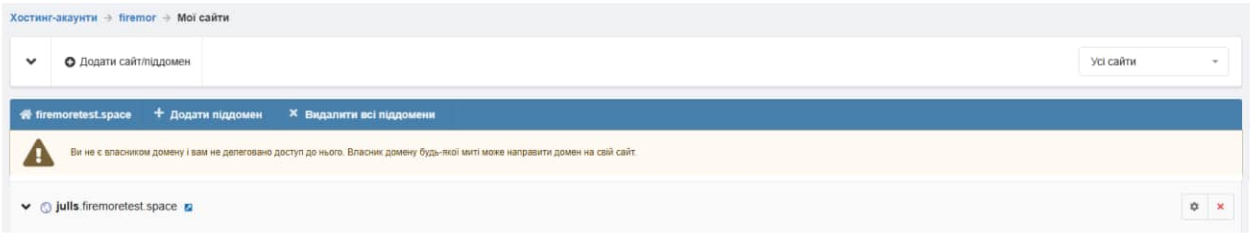


Рисунок 2.53 – Створений сайт

Для перенесення файлів проєкту використано FTP – це стандартний мережевий протокол, який дозволяє передавати файли між комп'ютерами через Інтернет або локальну мережу. Він працює за моделлю клієнт-сервер, де клієнт підключається до FTP-сервера для завантаження або вивантаження файлів.

Отже, для підключення до FTP-серверу спочатку потрібно додати FTP-користувача (рис. 2.54), це можна зробити за допомогою панелі керування. Тепер користувач має такі дані як хост, власні ім'я користувача та пароль – вони знадобляться для з'єднання.

Рисунок 2.54 – Додано FTP-користувача

Для під’єднання до FTP-серверу використано спеціальний програмний клієнт FileZilla – це найпоширеніша та надійна програма для успішного FTP-з’єднання. Вона надає можливість безпроблемного перенесення та викачування файлів, а також можливість не тільки читання, а і редагування файлів. Як раз сюди потрібно вводити хост, ім’я користувача та пароль, порт можна не писати, якщо він за замовчуванням (21).

Перевагою цього програмного забезпечення є те, що інтерфейс поділений на дві панелі: локальний сайт та віддалений сайт. Локальний сайт - це каталог проєкту, який знаходиться локально на комп’ютері. Віддалений сайт – це FTP-каталоги домену. За наявності водночас цих 2-х панелей процес перенесення файлів на FTP-сервер відбувається легко та швидко. Файли потрібно переносити в створений каталог сайту (рис. 2.55).

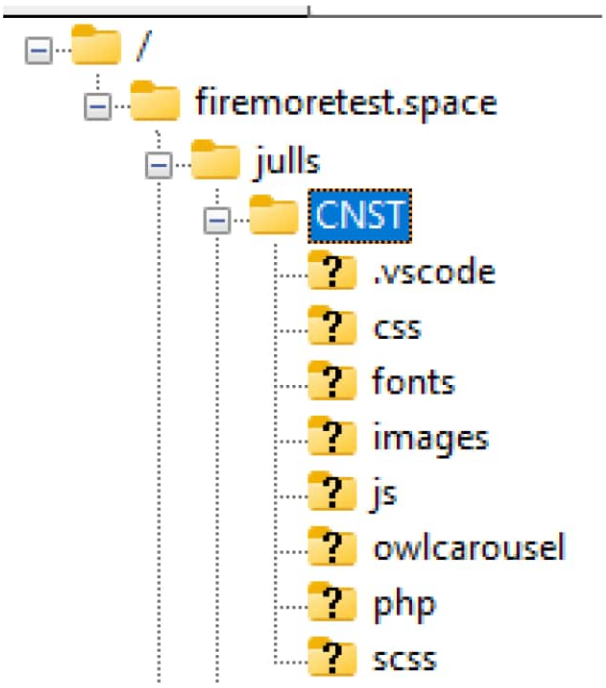


Рисунок 2.55 – Ієрархія сторінки

Після успішного перенесення можна спробувати зайти на сторінку (рис. 2.56). Зайти можна за заключним посиланням https://julls.firemoretest.space/CNST/.

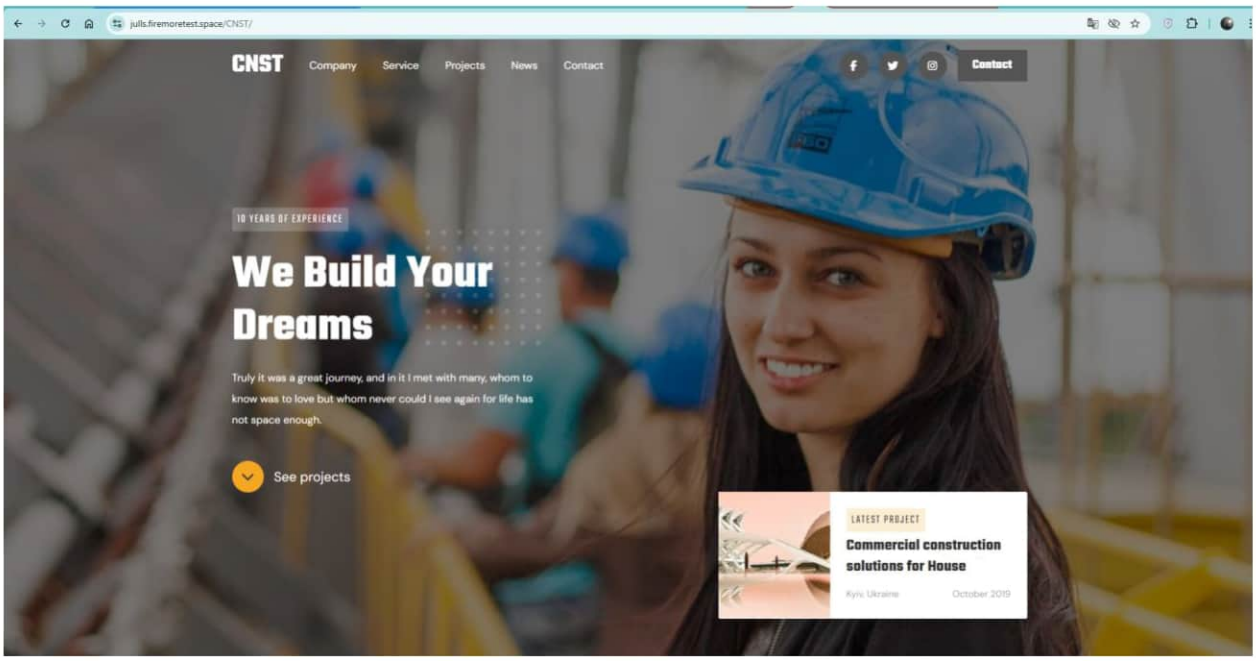


Рисунок 2.56 – Результат успішного перенесення файлів

2.5 Забезпечення базою даних

Після того, як клієнт заповнив дані, вони повинні десь зберігатись, щоб компанія змогла з ним надалі зв’язатись. Для цього створюється база даних, яка є надійною скринькою для зберігання даних, які надсилаються туди.

Хостинг використовує у якості СУБД MySQL – найбільш популярна СУБД, що є чудовим варіантом для створення сучасних БД. У панелі керування створено базу даних для сайту (рис. 2.57), відомі такі дані, як хост, логін та пароль(вони знадобляться), а також прямий доступ до phpMyAdmin – це інструмент для зручного керування БД MySQL через браузер. Принцип роботи, як з Excel – також є можливість у створенні, зміненні, видаленні таблиць та додавання записів. Також цей веб-застосунок дозволяє писати SQL-запити, не заходячи в термінал.

Рисунок 2.57 – Створено БД firemor_julls

За допомогою phpMyAdmin створено нову таблицю cnst-db (рис. 2.58) у БД firemor_julls. Таблицю створено за допомогою графічного інтерфейсу, але також можна використати SQL-запит, який матиме такий вигляд:

```
CREATE TABLE cnst-db (
      id INT(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
      name VARCHAR(30) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
      email VARCHAR(30) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
      message TEXT CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
);
```

Рисунок 2.58 – Створено таблицю cnst-db

Створено 4 поля таблиці:

- id – ціле число, яке автоматично збільшується при кожному новому записі та є первинним ключем;
- name та email – символьні значення, які обмежені до 30-ти символів та не можуть бути порожніми;
- message – текст (тип TEXT дає дозвіл вводити довгі посилання).

Використовується кодування utf8_general_ci для підтримки українських символів. Обрано стандартний тип таблиці MyISAM (рис. 2.59).

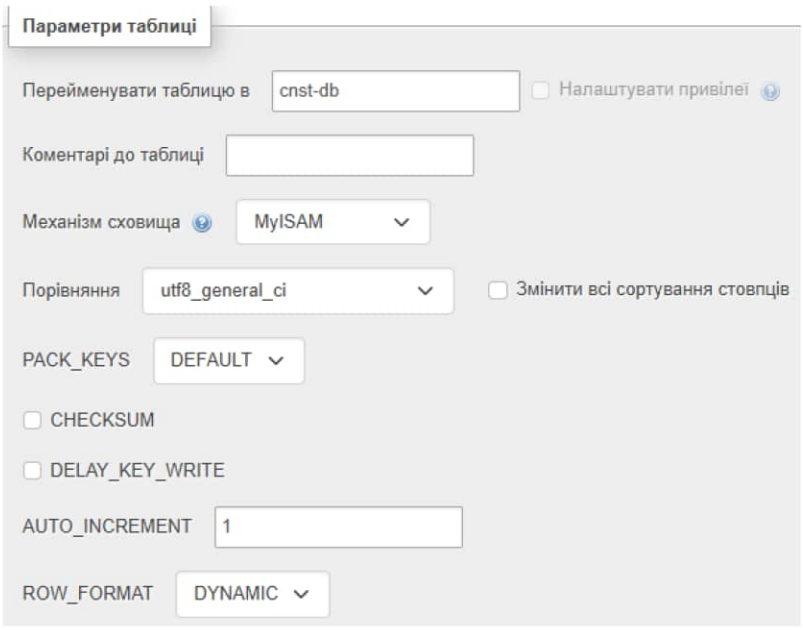


Рисунок 2.59 – Параметри таблиці cnst-db

Щоб таблиця отримувала дані, потрібно до неї підключитись та відправити запит.

У лістингу додатку Е реалізовано роботу з БД. Отримано та профільтровано дані: метод htmlspecialchars() із додатковими значеннями 2-х останніх аргументів ENT_QUOTES та 'UTF-8' є сучасним методом для захисту дані від XSS-атак, замінюючи спецсимволи на безпечні. Першим аргументом є саме значення с поля введення, яке зберігатиметься після відправки у глобальному масиві $_POST (запит буде надіслано за допомогою методу POST, тобто у сервер). Перед обробкою прибираються зі значення зайві пробіли спочатку та в кінці за допомогою методу trim().

Для того, щоб відправити правильні дані, перед роботою із БД потрібно провести валідацію даних. Йде така сама перевірка даних, як і у скрипті JQuery (див. додаток Д). Якщо перевірку пройдено – виконання скрипту продовжується, якщо ні, то завершується.

Створюється об’єкт $mysqli на основі класу mysqli для підключення до БД. Вона приймає такі параметри: назва хосту, логін, пароль та назва бази даних. Якщо не вдалося підключення – показується повідомлення про помилку.

Створюється зміна $stmt, яка зберігатиме підготовлений SQL-запит, який потрібно виконати. Йде підготовка SQL-запиту до таблиці, щоб вставити дані, але без вставлення конкретних значень – це захищає від SQL-ін'єкцій. Далі, підставляються зміні до запиту, кожна змінна типу string. Виконується запит і закривається запит та з'єднання із БД.

Відправка запиту реалізована за допомогою технології AJAX, який надає можливість у відправлені запитів без перезавантаження сторінки у фоновому режимі.

Запит (див. додаток Д) відправляється до PHP-обробника методом POST, тобто, їх не буде видно у URL сторінки. Всі дані з форми серіалізуються, тобто, перетворюються у формат key=value (ключ=значення). Якщо запит вдався, то форма ховається із анімацією за 300 мс та виводиться відповідь (текстове значення відповіді у PHP-обробнику) сервера в підготовлений для цього елемент під формою #response-message.

Якщо сталася помилка при запиті, показується червоне повідомлення про помилку.

Після завершення запиту, незалежно від результату, кнопка відправки знову стає активною та повертає своє минуле текстове значення(«Submit»).

2.6 Надсилання даних у Telegram-бот

Для того, аби компанія миттєво отримувала повідомлення про запити від клієнтів, було вирішено створити бота у месенджері Telegram, куди будуть приходити заповнені дані та повідомляти про це.

Створено бота «CNST заявки» у цьому месенджері за допомогою офіційного бота BotFather. Після введення його назви та нікнейму, бот відправляє посилання до створеного бота, а також токен для доступу до HTTP API (рис. 2.60). Токен – це унікальний ключ, який видано для автентифікації. Саме він дозволить надсилати повідомлення боту через Telegram API.

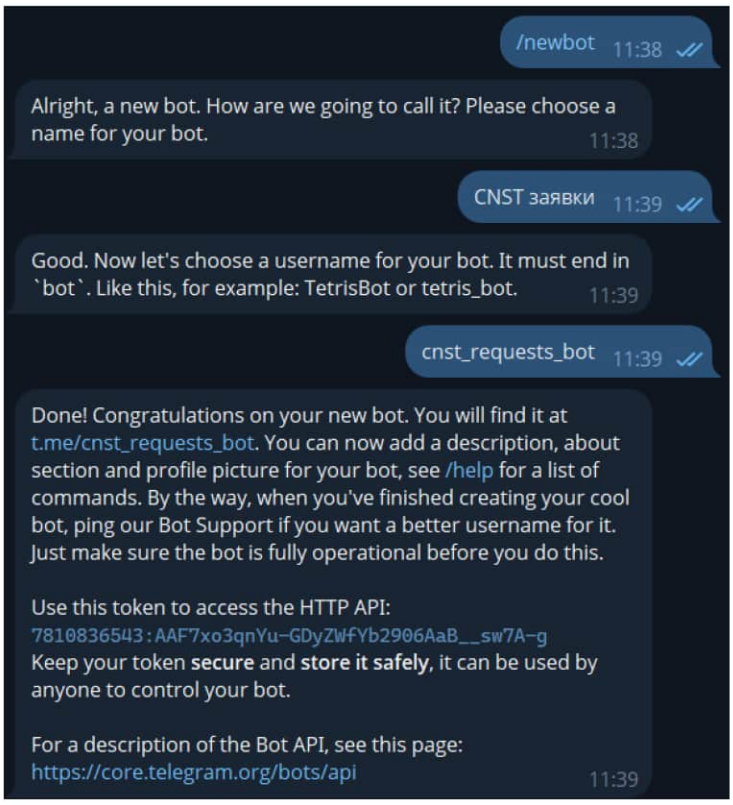


Рисунок 2.60 – Процес створення Telegram-бота «CNST заявки»

Також для підключення, потрібно дізнатись id користувача, якому бот буде відправляти запит. Для цього існують боти, які автоматично визначають та відправляють інформацію про користувача, який запустив цього бота. Використано бот userinfobot, який надав інформацію про поточного користувача, у тому числі id (рис. 2.61).

Рисунок 2.61 – Визначення id користувача

Щоб бот запрацював, його потрібно запустити.

У лістингу додатку Е також реалізовано роботу із Telegram-ботом. В цьому коді формується та надсилається саме повідомлення: виконується GET-

запит до Telegram API (за відповідним шляхом до методу надсилання + правильно сформований URL, який має параметри: куди надіслати (id користувача), що саме надіслати(сформоване повідомлення) та дозвіл на використання HTML у тексті (для роботи тегу <b>)).

У кінці PHP-обробника реалізовано умову, яка визначає, яку відповідь(response, див. додаток Д) надати після успішної або невдалої відправки. Відправка даних йде водночас до БД та Telegram-бот, тому реалізовано перевірку відправлення у бот (див. додаток Е).

2.7 Перевірка працездатності

На рисунках 2.62 та 2.63 зображено результати завантаження сторінки на мобільних пристроях та комп'ютерах відповідно – вони є у межах норми, а це свідчить про те, про проблеми із завантаженням сторінки не виникнуть.

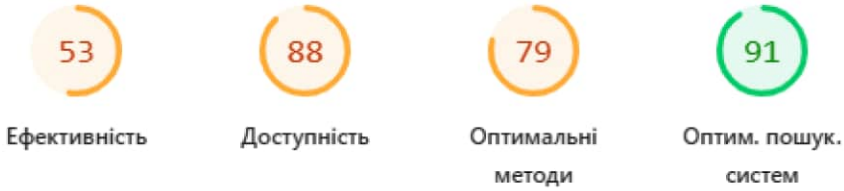


Рисунок 2.62 – Оцінка завантаження сторінки на мобільних пристроях за критеріями

Рисунок 2.63 - Оцінка завантаження сторінки на комп'ютерах за критеріями

Якщо ввести неправильно дані у форму, з'являються відповідні повідомлення про помилку введення, що означає, що валідація полів працює коректно (рис. 2.64).

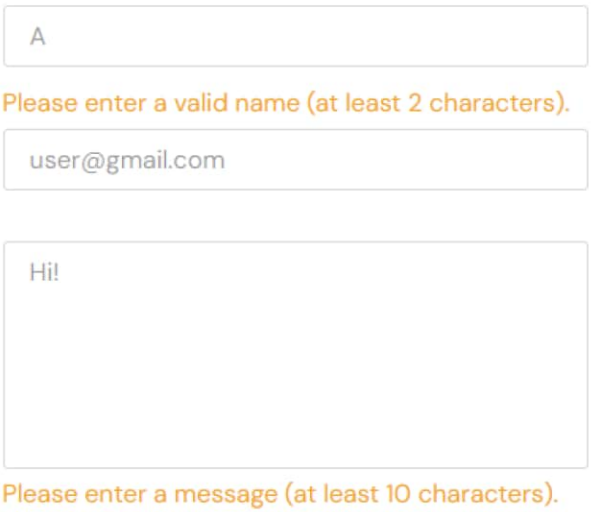


Рисунок 2.64 – Робота валідації форми

Після введення правильних даних, форма ховається та виводиться відповідне повідомлення (рис. 2.65), форма відправилась майже миттєво, але з невеличкою затримкою десь у секунду, тому ідея із блокуванням подій із кнопкою під час відправки форми стала у нагоді.

Рисунок 2.65 – Повідомлення про успішну відправку даних

Отримано майже одразу повідомлення від Telegram-боту «CNST заявки», яке має всі відправлені дані (рис. 2.66).

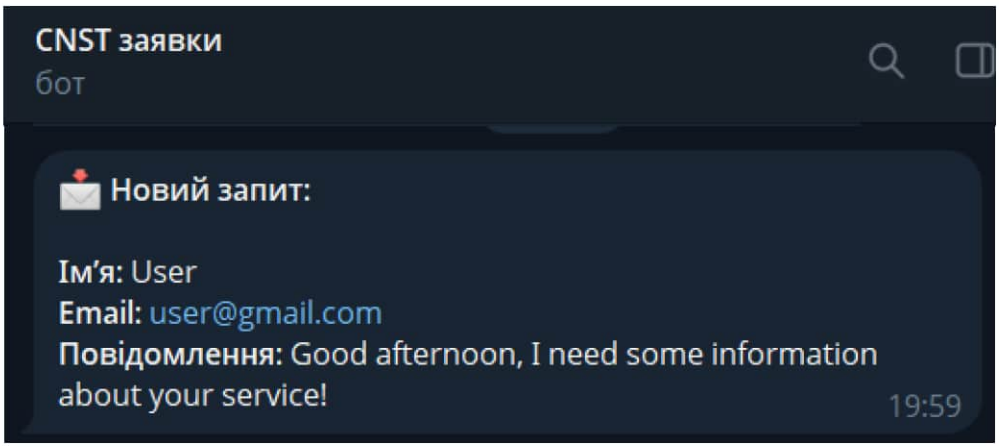


Рисунок 2.66 – Отримане повідомлення від боту

Також, в таблиці cnst-db бази даних firemor_julls додано новий запис із цими самим даними (рис. 2.67).

id	name	email	message
1	User	user@gmail.com	Good afternoon, I need some information about your...

Рисунок 2.67 – Доданий новий запис до таблиці

Користувач User отримай свій id зі значенням 1, якщо до таблиці буде додано новий користувач(новий запис), то автоматично він отримає свій id на 1 більше (2).

Результати перевірки роботи форми зворотного зв’язку свідчать про коректну роботу скриптів – PHP-обробника та AJAX-запиту.

2.8 Висновки до другого розділу

В другому розділі було описано пошук привабливого та сучасного дизайну, який відповідає стилю компанії та сучасним вебтрендам. Обрано технології, враховуючи сучасні тенденції веброзробки. Описано основні моменти у розробці адаптивного інтерфейсу користувача, що коретно відобразиться на всіх пристроях. Розкрито реалізацію функціональних елементів. Розглянуто процес розміщення сайту на хостинг. Описано створення БД та роботу із ним. Розглянуто реалізацію надсилання даних у Telegram-бот, який служить додатковим кагалом для швидкого отримання нових заявок. Проведено перевірку на працездатність у реальному середовищі.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто поставлену мету – досліджено сучасні технології front-end розробки та створено адаптивний вебсайт для будівельної компанії CNST.

На основі порівняльного аналізу існуючих лендінгів було визначено типові помилки та переваги, які допомогли уникнути подібних недоліків у власній розробці. У роботі було обґрунтовано вибір середовища розробки (VS Code) та технологій (HTML5, SCSS, JQuery, PHP, MySQL, API), що забезпечили ефективну реалізацію поставлених завдань.

Результатом стала повноцінна адаптивна сторінка з інтерактивними елементами, включно з формами зворотного зв’язку, анімаціями, слайдерами, інтеграцією з Telegram-ботом та збереженням даних у базу даних. Також було проведено перевірку працездатності сайту в реальному середовищі після його розміщення на хостинг.

Сайт успішно проходить тестування на адаптивність та має прийнятні показники швидкості завантаження, що підтверджує правильність обраних рішень. Таким чином, створений лендінг не тільки виконує поставлені функції, але й відповідає сучасним вимогам як до дизайну, так і до технічної реалізації.

У подальшому сайт можна доповнити додатковими сторінками, багатомовною підтримкою або системою керування контентом (CMS), що розширить його функціонал і зробить зручнішим для адміністрування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Top 15 web development trends to expect in 2025. *Digital Silk*. URL: https://www.digitalsilk.com/digital-trends/web-development-trends/ (дата звернення: 02.05.2025).
2. Future of web development trends & latest technology (2024). *Frogslayer*. URL: https://frogslayer.com/blog/the-future-of-web-development-emerging-trends-and-technologies/ (дата звернення: 02.05.2025).
3. 8 web development trends for 2025: insights for developers and wordpress users. *WP Engine*. URL: https://wpengine.com/blog/web-development-trends/ (дата звернення: 02.05.2025).
4. 8 of the hottest web development trends in 2025. *CareerFoundry*. URL: https://careerfoundry.com/en/blog/web-development/8-biggest-trends-in-web-development-trends/ (дата звернення: 02.05.2025).
5. PageSpeed insights. URL: https://pagespeed.web.dev/ (дата звернення: 02.05.2025).
6. SMM VIP. URL: https://smmvip.com.ua/ (дата звернення: 02.05.2025).
7. Besocial. URL: https://besocial.studio/ (дата звернення: 02.05.2025).
8. Sublime Text - the sophisticated text editor for code, markup and prose. *Sublime Text - Text Editing, Done Right*. URL: https://www.sublimetext.com/ (дата звернення: 04.05.2025).
9. Introduction | sublime text community documentation. *Sublime Text Community Documentation*. URL: https://docs.sublimetext.io/guide/ (дата звернення: 04.05.2025).
10. Unpopular personal opinion: Sublime Text is still (one of) the most well-made popular editor. URL: https://www.reddit.com/r/webdev/comments/12ln8gj/unpopular_personal_opinion_sublime_text_is_still/ (дата звернення: 04.05.202).
11. Учасники проєктів Вікімедіа. Sublime text – вікіпедія. *Вікіпедія*. 2014. URL: https://uk.wikipedia.org/wiki/Sublime_Text (дата звернення: 04.05.2025).

12. A modern, open source code editor that understands web design. *Brackets*. URL: https://brackets.io/ (дата звернення: 04.05.2025).
13. The 10 best JavaScript IDEs (and code editors) for 2024. URL: https://www.educative.io/blog/best-javascript-ides-code-editors (дата звернення: 04.05.2025).
14. Учасники проєктів Вікімедіа. Adobe brackets – вікіпедія. *Вікіпедія*. 2015. URL: https://uk.wikipedia.org/wiki/Adobe_Brackets (дата звернення: 04.05.2025).
15. Microsoft. Visual studio code - code editing. redefined. URL: https://code.visualstudio.com/ (дата звернення: 02.05.2025).
16. Ibrahim M. Best VS code extensions to boost your productivity 🚀. *DEV Community*. URL: https://dev.to/moibra/best-vs-code-extensions-to-boost-your-productivity-1n86 (дата звернення: 02.05.2025).
17. Free online UI design tool & software for teams | figma. *Figma*. URL: https://www.figma.com/ui-design-tool/ (дата звернення: 13.05.2025).
18. Google webfonts helper. *google webfonts helper*. URL: https://gwfh.mranftl.com/fonts (дата звернення: 13.05.2025).
19. Home | owl carousel | 2.3.4. *Site not found · GitHub Pages*. URL: https://owlcarousel2.github.io/OwlCarousel2/ (дата звернення: 20.05.2025).
20. Swiper - the most modern mobile touch slider. *Swiper*. URL: https://swiperjs.com/ (дата звернення: 20.05.2025).
21. JQuery. *jQuery*. URL: https://jquery.com/ (дата звернення: 20.05.2025).
22. Font awesome. *Font Awesome*. URL: https://fontawesome.com/ (дата звернення: 20.05.2025).
23. Intersection Observer API - Web APIs | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API (дата звернення: 22.05.2025).

24. Хостинг україна - купити хостинг в україні | хостинг україна. *Хостинг Україна - купити хостинг в Україні | Хостинг Україна*. URL: https://www.ukraine.com.ua/ (дата звернення: 23.05.2025).

ДОДАТОК А
ВМІСТ ФАЙЛУ _BASE.SCSS

```
*,
*::before,
*::after {
      box-sizing: border-box;
}

html,
body {
      height: 100%;
}

html {
      @media screen and (min-width: 991px) {
            scroll-behavior: smooth;
      }
}

body,
ul {
      margin: 0;
}

body {
      color: var(--color-primary-dark);
      font-family: "DM Sans", sans-serif;
      font-size: 16px;
      background-color: var(--color-primary-light);
      line-height: 2;
      font-weight: 400;
      overflow-x: hidden;
}

h1,
h2,
h3,
h4,
h5,
h6,
p {
      margin-top: 0;
}

ul,
input,
td {
      padding: 0;
}

ul {
      list-style-type: none;
}

address {
      font-style: normal;
}

a {
```

```
      display: inline-block;
}

img {
      display: block;
      max-width: 100%;
      height: auto;
      object-fit: cover;
      user-select: none;
}

textarea,
input,
button {
      font-family: inherit;
      font-size: inherit;
}

input,
button {
      user-select: none;
}

input [type="search"]::-webkit-search-decoration,
input [type="search"]::-webkit-search-cancel-button,
input [type="search"]::-webkit-search-results-button,
input [type="search"]::-webkit-search-results-decoration,
select {
      -webkit-appearance: none;
}

input:focus,
select:focus,
textarea:focus {
      outline: none;
}

input [type="checkbox"],
button {
      cursor: pointer;
}

textarea {
      resize: none;
}

select {
      -moz-appearance: none;
      appearance: none;
      -o-appearance: none;
      cursor: pointer;
}

select::-ms-expand {
      display: none;
}

a,
button {
      transition: background-color 0.2s ease-in, color 0.2s ease-in;
}
```

ДОДАТОК Б
ВМІСТ ФАЙЛУ MAIN.SCSS

```
@import "./utils/fonts";
@import "./utils/variables";

@import "./base";

@import "./utils/utils";

@import "./components/bar-card";
@import "./components/buttons";
@import "./components/card";
@import "./components/founder-card";
@import "./components/header-nav";
@import "./components/inputs";
@import "./components/latest-project";
@import "./components/logo";
@import "./components/pagination";
@import "./components/price-card";
@import "./components/socials";
@import "./components/step-card";
@import "./components/swiper-sliders";

@import "./layouts/about-section";
@import "./layouts/bar-section";
@import "./layouts/column-section";
@import "./layouts/contact-section";
@import "./layouts/container";
@import "./layouts/faq-section";
@import "./layouts/footer";
@import "./layouts/header";
@import "./layouts/hero-section";
@import "./layouts/mobile-menu";
@import "./layouts/page";
@import "./layouts/portfolio-section";
@import "./layouts/pricing-section";
@import "./layouts/row-section";
@import "./layouts/section";
@import "./layouts/team-section";
@import "./layouts/work-process";
```

ДОДАТОК В
ВМІСТ ФАЙЛУ _HEADER-NAV.SCSS

```
.header-nav {
	flex-grow: 1;

	&--desktop {
		display: none;

		@media screen and (min-width: $bp-desktop) {
			display: block;
		}
	}

	&--mobile {
		@media screen and (min-width: $bp-desktop) {
			display: none;
		}
	}

	&__list {
		display: flex;
		gap: 20px;
		flex-direction: column;

		@media screen and (min-width: $bp-desktop) {
			flex-direction: row;
			gap: 40px;
		}
	}

	&__link {
		text-decoration: none;
		color: var(--color-primary-light);

		&:hover {
			color: var(--color-secondary-light);
		}
	}
}
```

ДОДАТОК Г
ФРАГМЕНТИ ФАЙЛУ INDEX.HTML

Г.1 Структура слайдеру «Сервіс»

```
<div class="row-slider owl-carousel owl-theme">
	<div class="card">
		<img src="./images/service-slide-image.jpg" alt="Building Renovation" />

		<div class="card__content">
			<p class="card__title card__title--service">01. Building
		Renovation</p>
			<p class="card__text">
				Apparently we had reached a great height in the atmosphere,
				for the sky was a dead black, and
				the stars had ceased to twinkle.
			</p>
		</div>
	</div>
	<div class="card">
		<img src="./images/service-slide-image.jpg" alt="Building Renovation" />

		<div class="card__content">
			<p class="card__title card__title--service">01. Building
		Renovation</p>
			<p class="card__text">
				Apparently we had reached a great height in the atmosphere,
				for the sky was a dead black, and
				the stars had ceased to twinkle.
			</p>
		</div>
	</div>
	<div class="card">
		<img src="./images/service-slide-image.jpg" alt="Building Renovation" />

		<div class="card__content">
			<p class="card__title card__title--service">01. Building
		Renovation</p>
			<p class="card__text">
				Apparently we had reached a great height in the atmosphere,
				for the sky was a dead black, and
				the stars had ceased to twinkle.
			</p>
		</div>
	</div>
</div>
```

Г.2 Структура слайдеру «Команда»

```
<div class="team-section__carousel swiper">
	<div class="swiper-wrapper">
		<div class="card swiper-slide">
			<img src="./images/director.jpg" alt="Edward Lindgren"
			class="card__image" />

			<div class="card__content">
				<p class="card__title card__title--section">Edward
				Lindgren</p>
				<p class="card__type">Director</p>
			</div>
```

```
            </div>
            <div class="card swiper-slide">
                  <img src="./images/architect.jpg" alt="Benedita Tavares"
                  class="card__image" />

                <div class="card__content">
                        <p class="card__title card__title--section">Benedita
                        Tavares</p>
                        <p class="card__type">Architect</p>
                </div>
            </div>
            <div class="card swiper-slide">
                  <img src="./images/elevator.jpg" alt="Richardo Kann"
                  class="card__image" />

                <div class="card__content">
                        <p class="card__title card__title--section">Richardo
                        Kann</p>
                        <p class="card__type">Elevator</p>
                </div>
            </div>
            <div class="card swiper-slide">
                  <img src="./images/director.jpg" alt="Edward Lindgren"
                  class="card__image" />

                <div class="card__content">
                        <p class="card__title card__title--section">Edward
                        Lindgren</p>
                        <p class="card__type">Director</p>
                </div>
            </div>
      </div>

      <div class="swiper-pagination"></div>
</div>
```

Г.3 Структура галереї з фільтрацією секції «Портфоліо»

```
<div class="portfolio-section__buttons">
      <div class="portfolio-section__buttons-container">
            <button class="portfolio-section__button button-active" type="button"
            data-filter="all">All</button>
            <button class="portfolio-section__button" type="button" data-
            filter="building">Building</button>
            <button class="portfolio-section__button" type="button" data-
            filter="construction">
                  Construction
            </button>
            <button class="portfolio-section__button" type="button" data-
            filter="project-planning">
                  Project Planning
            </button>
      </div>
</div>

<div class="portfolio-section__cards">
      <a href="#" class="card card--section building">
            <img src="./images/international-airport.jpg" alt="International Airport"
            />

            <div class="card__content">
                  <p class="card__title card__title--section">International
                  Airport</p>
                  <p class="card__type">Building</p>
            </div>
      </a>
```

```
      <a href="#" class="card card--section construction">
            <img src="./images/new-york-bank.jpg" alt="New York Bank" />

            <div class="card__content">
                  <p class="card__title card__title--section">New York Bank</p>
                  <p class="card__type">Construction</p>
            </div>
      </a>
      <a href="#" class="card card--section project-planning">
            <img src="./images/family-house.jpg" alt="Family House" />

            <div class="card__content">
                  <p class="card__title card__title--section">Family House</p>
                  <p class="card__type">Project</p>
            </div>
      </a>
      <a href="#" class="card card--section building">
            <img src="./images/mega-mell.jpg" alt="Mega Mall" />

            <div class="card__content">
                  <p class="card__title card__title--section">Mega Mall</p>
                  <p class="card__type">Building</p>
            </div>
      </a>
      <a href="#" class="card card--section construction">
            <img src="./images/la-arena.jpg" alt="LA Arena" />

            <div class="card__content">
                  <p class="card__title card__title--section">LA Arena</p>
                  <p class="card__type">Construction</p>
            </div>
      </a>
      <a href="#" class="card card--section project-planning">
            <img src="./images/future-office.jpg" alt="Future Office" />

            <div class="card__content">
                  <p class="card__title card__title--section">Future Office</p>
                  <p class="card__type">Project</p>
            </div>
      </a>
</div>
```

Г.4 Структура акордеону «F.A.Q»

```
<div class="faq-section__accordion">
      <div class="faq-section__spoller">
            <div class="faq-section__spoller-title">
                  See the face of Mars
                  <div class="faq-section__spoller-btn">
                        <svg width="8" height="12" viewBox="0 0 8 12" fill="none"
                        xmlns="http://www.w3.org/2000/svg">
                              <!-- prettier-ignore -->
                              <path fill-rule="evenodd" clip-rule="evenodd"
                              d="M1.50933 12L0 10.5855L4.9808 5.998L0
                              1.4145L1.50933 0L8 5.998L1.50933 12Z"
                              fill="#333333"/>
                        </svg>
                  </div>
            </div>

            <p class="faq-section__text">
                  Parts formed a great hindrance to my speed, I resolved, contrary
                  to my first intention, to make the
                  being of a gigantic stature that is to say, about eight feet in
                  height, and proportionably large.
                  After having formed this determination, and having spent some
                  months in successfully.
```

```
            </p>
        </div>
        <div class="faq-section__spoller">
            <div class="faq-section__spoller-title">
                Rare experience
                <div class="faq-section__spoller-btn">
                    <svg width="8" height="12" viewBox="0 0 8 12" fill="none"
                    xmlns="http://www.w3.org/2000/svg">
                        <!-- prettier-ignore -->
                        <path fill-rule="evenodd" clip-rule="evenodd"
                        d="M1.50933 12L0 10.5855L4.9808 5.998L0
                        1.4145L1.50933 0L8 5.998L1.50933 12Z"
                        fill="#333333"/>
                    </svg>
                </div>
            </div>

            <p class="faq-section__text">
                Lorem ipsum dolor sit amet consectetur adipisicing elit. Libero
                porro eius omnis quis corporis
                praesentium inventore maxime necessitatibus, dolore voluptatibus
                rerum animi sunt magni sit
                aspernatur officia alias molestias voluptatum.
            </p
        </div>
        <div class="faq-section__spoller">
            <div class="faq-section__spoller-title">
                Fine telescope
                <div class="faq-section__spoller-btn">
                    <svg width="8" height="12" viewBox="0 0 8 12" fill="none"
                    xmlns="http://www.w3.org/2000/svg">
                        <!-- prettier-ignore -->
                        <path fill-rule="evenodd" clip-rule="evenodd"
                        d="M1.50933 12L0 10.5855L4.9808 5.998L0
                        1.4145L1.50933 0L8 5.998L1.50933 12Z"
                        fill="#333333"/>
                    </svg>
                </div>
            </div>

            <p class="faq-section__text">
                Lorem ipsum dolor sit amet consectetur adipisicing elit.
                Exercitationem magnam architecto explicabo
                delectus vitae repudiandae enim veniam, facere, earum facilis
                laudantium praesentium in voluptatem
                ab eaque quas reprehenderit et repellat.
            </p>
        </div>
</div>
```

ДОДАТОК Д
ФРАГМЕНТИ ФАЙЛУ SCRIPT.JS

Д.1 Реалізація відправки форми зворотного зв'язку

```
$(document).ready(function () {
      $("#contact-form").on("submit", function (e) {
            e.preventDefault();

            const name = $("#user-name").val().trim();
            const email = $("#user-email").val().trim();
            const message = $("#user-message").val().trim();
            const $submitBtn = $(".contact-section__btn");

            if (name === "" || name.length < 2) {
                  $("#form-name").html(
                  "<p>Please enter a valid name (at least 2 characters).</p>"
                  );
                  return;
            } else $("#form-name").html("");

            if (email === "" || !validateEmail(email)) {
                  $("#form-email").html("<p>Please enter a valid email
                  address.</p>");
                  return;
            } else $("#form-email").html("");

            if (message === "" || message.length < 10) {
                  $("#form-message").html(
                  "<p>Please enter a message (at least 10 characters).</p>"
                  );
                  return;
            } else $("#form-message").html("");

            $submitBtn.prop("disabled", true).text("Sending...");

            $.ajax({
                  url: "php/check.php",
                  type: "POST",
                  data: $(this).serialize(),
                  success: function (response) {
                        $("#contact-form").fadeOut(300, function () {
                              $("#response-message")
                              .html("<p>" + response + "</p>")
                              .fadeIn(300);
                        });
                  },
                  error: function () {
                        $("#response-message")
                        .html('<p style="color: red;">Error sending
                        data.</p>')
                        .fadeIn(300);
                  },
                  complete: function () {
                        $submitBtn.prop("disabled", false).text("Submit");
                  },
            });
      });
});
```

```
function validateEmail(email) {
      const re = /^ [^\s@]+@ [^\s@]+\. [^\s@]+$/;
      return re.test(email);
}
```

Д.2 Реалізація анімацій

```
function animateCounters() {
      $(".count").each(function () {
            const $this = $(this);
            const target = parseFloat($this.attr("data-target"));

            $({ countNum: 0 }).animate(
                        { countNum: target },
                        {
                        duration: 2000,
                        easing: "swing",
                        step: function () {
                              const value = Number.isInteger(target)
                                    ? Math.floor(this.countNum)
                                    : this.countNum.toFixed(1);
                              $this.text(value);
                        },
                        complete: function () {
                              const value = Number.isInteger(target) ? target
                              target.toFixed(1);
                              $this.text(value);
                        },
                        }
            );
      });
}

const observer = new IntersectionObserver(
      (entries, observer) => {
            entries.forEach((entry) => {
                        if (entry.isIntersecting) {
                        $(entry.target).addClass("visible");
                        animateCounters();
                        observer.unobserve(entry.target);
                        }
            });
      },
      { threshold: 0.1 }
);

$(document).ready(function () {
      $(".container").addClass("animate");

      setTimeout(() => {
            $(".hero-section .container").addClass("visible");
      }, 300);

      $(".container").each(function () {
            const $container = $(this);
            if (!$container.hasClass("container--hero")) {
                        observer.observe(this);
            }
      });

      const statsSection = document.querySelector("#statistics");
      if (statsSection) {
            observer.observe(statsSection);
```

```
      }
});

let stepIndex = 0;
const steps = $(".step-card");

function changeStepColor() {
      steps.each(function (index) {
            if (index === stepIndex) {
                  $(this).addClass("active");
            } else {
                  $(this).removeClass("active");
            }
      });

      stepIndex = (stepIndex + 1) % steps.length;
}

setInterval(changeStepColor, 2000);
```

ДОДАТОК Е
ВМІСТ ФАЙЛУ CHECK.PHP

```
<?php
      $name = htmlspecialchars(trim($_POST ['user-name']), ENT_QUOTES, 'UTF-
      8');
      $email = htmlspecialchars(trim($_POST ['user-email']), ENT_QUOTES, 'UTF-
      8');
      $message = htmlspecialchars(trim($_POST ['user-message']), ENT_QUOTES,
      'UTF-8');

      if (mb_strlen($name) < 2) {
            exit('Invalid name. Name must have at least 2 characters.');
      }
      if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            exit('Invalid email format.');
      }
      if (mb_strlen($message) < 10) {
            exit('Message is too short. It must have at least 10
            characters.');
      }

      $mysqli = new mysqli('firemor.mysql.tools', 'firemor_julls',
      '8+S_kE6s4m', 'firemor_julls');
      if ($mysqli->connect_error) {
            exit('Connection failed: ' . $mysqli->connect_error);
      }

      $stmt = $mysqli->prepare("INSERT INTO `cnst-db` (`name`, `email`,
      `message`) VALUES (?, ?, ?)");
      $stmt->bind_param('sss', $name, $email, $message);
      $stmt->execute();
      $stmt->close();
      $mysqli->close();

      $token = '7810836543:AAF7xo3qnYu-GDyZWfYb2906AaB__sw7A-g';
      $chat_id = '573622100';

      $telegram_message = "📩 <b>Новий запит:</b>\n\n";
      $telegram_message .= "<b>Ім’я:</b> $name\n";
      $telegram_message .= "<b>Email:</b> $email\n";
      $telegram_message .= "<b>Повідомлення:</b> $message";

      $sendToTelegram =
      file_get_contents("https://api.telegram.org/bot$token/sendMessage?" .
      http_build_query( [
            'chat_id' => $chat_id,
            'text' => $telegram_message,
            'parse_mode' => 'HTML'
      ]));

      if ($sendToTelegram) {
            echo 'Your message was successfully sent!';
      } else {
            echo 'Message sent to database, but failed to send in Telegram.';
      }
?>
```