МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ

Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«_____» _____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНИЙ ПРОЄКТ, ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ **"БАКАЛАВР"**
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
"ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: «Web додаток "Європа для українців"»

Виконавець: студентка групи УС-412Б Брояк Вікторія Олегівна

Керівник: к.т.н., доцент Холявкіна Тетяна Володимирівна

Нормоконтролер: ст. викл. Олександр ШЕВЧЕНКО

Київ – 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *Комп'ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«____»_____2023р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студентки

Брояк Вікторії Олегівни

(прізвище, ім'я, по батькові)

1. Тема роботи: «Web додаток "Європа для українців"» затверджена наказом ректора від 01.05.2023р. № 623/ст.

2. Термін виконання роботи з 15.05.2023 р. по 25.06.2023 р.

3. Вихідні дані до роботи: web додаток «Європа для українців», який надає українським біженцям цінну інформацію для процесу прийняття рішень при розгляді країн-членів ЄС як потенційних місць для поселення.

4. Зміст пояснювальної записки: вибір мов програмування, аналіз важливих факторів, які мають вирішальне значення для українських біженців при виборі країни, комплексний аналіз різноманітних інтегрованих середовищ розробки програмного забезпечення, оцінка зручності використання та досвіду користувача.

5. Перелік обов'язкового графічного матеріалу: карта подорожі клієнта для Марії, каркасна конструкція сторінки «Країна», макет головної сторінки створений за допомогою Figma.

6. Календарний план-графік

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1.	Проаналізувати літературу та джерела за темою дипломного проекту.	15.05.2023 – 17.05.2023	
2.	Розроблення та затвердження плану дипломного проекту.	18.05.2023 – 20.05.2023	
3.	Провести консультації з науковим керівником щодо створення першого розділу.	21.05.2023– 23.05.2023	
4.	Розробка розділу 1	24.05.2023– 02.06.2023	
5.	Розробка розділу 2	03.06.2023– 09.06.2023	
6.	Розробка розділу 3	10.06.2023 – 13.06.2023	
7.	Висновки та оформлення пояснювальної записки дипломного проекту.	14.06.2023 – 16.05.2023	
8.	Підписання необхідних документів у встановленому порядку.	17.05.2023 – 19.05.2023	

7. Дата видачі завдання: «__15__» __травня____2023 р.

Керівник дипломного проекту _____ Тетяна ХОЛЯВКІНА
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання_____ Вікторія БРОЯК
(підпис студента) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Web додаток "Європа для українців"» складається зі вступу, трьох розділів, висновку, списку бібліографічних посилань містить 77 сторінок, 57 рисунків, 2 таблиці. Список бібліографічних посилань складається з 10 найменувань.

Об'єктом дослідження **є**: процес аналізу та розробки функцій web додатку «Європа для українців».

Предметом дослідження є: web додаток «Європа для українців».

Мета роботи: проектування та розробка web додатку «Європа для українців» з метою надання повної інформації та допомоги українським біженцям, які шукають тимчасового захисту в країнах Європейського Союзу.

Для досягнення мети проектування та розробки веб-додатку «Європа для українців» та вирішення поставлених завдань було застосувано декілька ***методів дослідження:***

- метод проведення комплексного огляду наявних джерел, інформації;
- метод проведення опитування та інтерв'ю;
- метод порівняння програмних продуктів і платформ;
- метод тестування зручності використання;
- метод створення прототипів.

Результатом дослідження є розроблений web-додаток «європа для українців», який сприяє поліпшенню доступу українських біженців до інформації та допомоги, спрощує їхні пошукові та адаптаційні процеси у країнах європейського союзу, а також сприяє підвищенню їхньої інтеграції та самостійності.

ЄВРОПЕЙСЬКИЙ СОЮЗ, WEB ДОДАТОК, МОВИ ПРОГРАМУВАННЯ, ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (IDE), РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API	Application Programming Interface – різновид програмного інтерфейсу, який пропонує послуги іншим частинам програмного забезпечення.
BEM	Block Element Modificator – методологія web-розробки
CMS	Content Management System – система керування контентом
CSS	Cascading Style Sheets – каскадні таблиці стилів
HTML	Hypertext Markup Language – це стандартна мова гіпертекстової розмітки
IDE	Integrated Development Environment – інтегроване середовище розробки
PHP	Personal Home Page – Інструмент для створення персональних web сторінок
Sass	Syntactically Awesome Style Sheets – скриптова метамова, яка інтерпретується в CSS
UCD	User Centered Design – проектування орієнтоване на користувача
UI	User Interface – користувацький інтерфейс
UX	User Experience – досвід користувавча
VS Code	Visual Studio Code – редактор коду
ЄС	Європейський Союз – економічний та політичний союз 28 держав-членів, що розташовані здебільшого у Європі.
ПП	Програмний продукт

ВСТУП

Більше року триває російське вторгнення в Україну. Це призвело до руйнівної гуманітарної кризи, через яку мільйони українців залишилися без домів. На жаль, з кожним днем все більше українців втрачають свої домівки та починають життя з чистого листа. Багато з них були змушені шукати притулку в сусідніх країнах і в Європейському Союзі [1]. Наразі ЄС надає тимчасовий захист українцям, які виїхали за кордон, та допомагає влаштуватися там. Однак, маючи таку кількість країн ЄС на вибір, може бути складно визначити, яка країна найкраще підходить для окремої людини чи сім'ї (див. Рис. 1.1.).

Для вирішення цієї проблеми розробляється web додаток під назвою «Європа для українців», який надає українцям вичерпну інформацію про країни-члени ЄС, допомагаючи їм приймати зважені рішення щодо того, де оселитися та на яку підтримку вони можуть розраховувати.

Основною метою цього дослідження є проектування та розробка web додатку, який збиратиме та представлятиме інформацію про різні аспекти країн-членів ЄС, які є найбільш актуальними для українських біженців, які шукають тимчасового захисту в цих країнах. Додаток буде надавати вичерпну актуальну інформацію про різні фактори, дозволяючи користувачам порівнювати різні країни та приймати обґрунтоване рішення про те, де оселитися.

Web додаток «Європа для українців» є **актуальним** як для українських біженців, так і для країн ЄС. Для українських біженців це стане цінним інструментом для прийняття обґрунтованих рішень щодо того, де оселитися в ЄС, на основі факторів, які для них найбільше стосуються. Додаток дозволить їм порівнювати різні країни та отримувати доступ до актуальної інформації з різних аспектів, забезпечуючи тим самим підтримку, необхідну для початку нове життя в новому домі.

Новизна проекту полягає в тому, що для країн-членів ЄС web додаток сприятиме інтеграції українських біженців у їхні суспільства, дозволяючи їм максимально використати можливості, які їм надаються.

РОЗДІЛ 1
ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ WEB-ДОДАТКІВ

1.1. Огляд програмного забезпечення для web-додатків, що використовуються для розробки подібних проектів.

Web програми стали важливим інструментом у сучасну цифрову епоху. Web додаток — це програма, яка працює на web сервері та доступна через web браузер. Ці програми можуть варіюватися від простих форм до складних соціальних мереж і сайтів електронної комерції. Існує безліч програмних інструментів, які допомагають розробникам створювати web програми, кожна з яких має власний набір функцій, переваг і недоліків. У цьому розділі ми надамо огляд програмного забезпечення, яке використовується для розробки подібних проектів, а також переваги та недоліки кожного.

Першим програмним інструментом, який зазвичай використовується для створення web додатків, є система керування контентом (CMS). CMS — це програма, яка дозволяє користувачам створювати, керувати та публікувати вміст на web сайті. Деякі з популярних варіантів CMS включають WordPress, Drupal і Joomla. Ці інструменти легко налаштовуються та надають низку функцій, включаючи спеціальні шаблони, плагіни та віджети. Однією з головних переваг використання CMS є те, що вона дає змогу нетехнічним користувачам керувати своїм web сайтом, не вимагаючи глибоких знань кодування. Однак CMS можуть бути складними та вимагати технічного досвіду для налаштування та обслуговування, особливо коли йдеться про складніші налаштування.

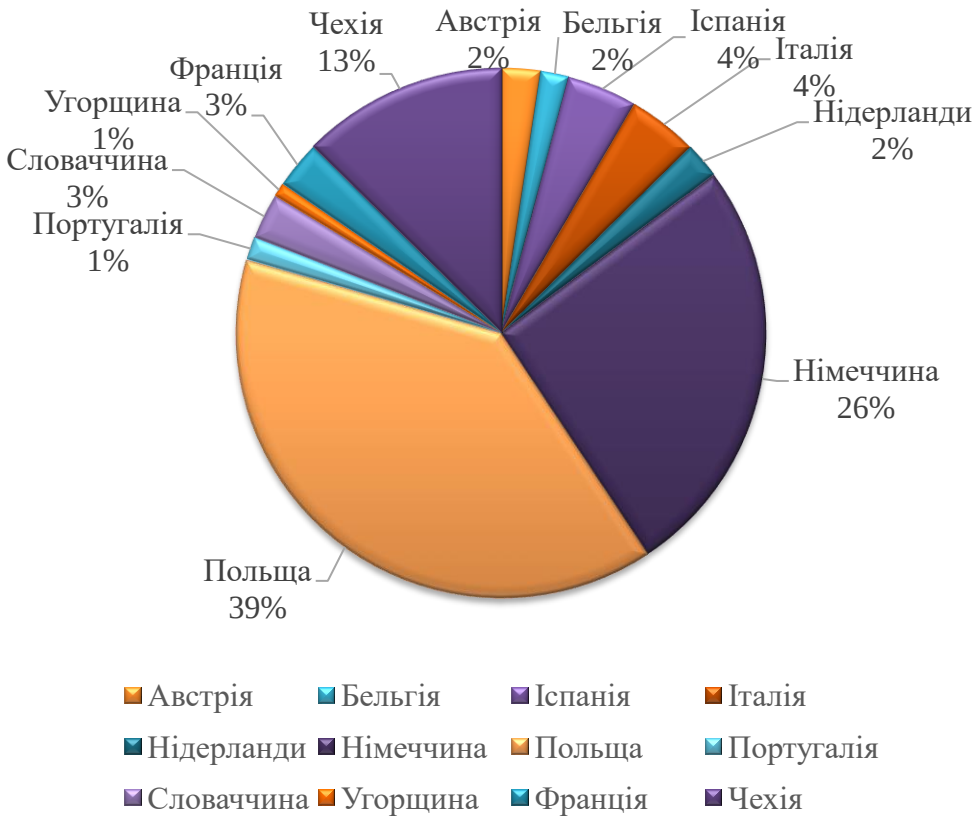

Рис. 1.1. Кількість українських біженців в країнах Європи

Іншим популярним інструментом для розробки web додатків є web фреймворк. Web фреймворк — це програмна основа, яка допомагає розробникам створювати web додатки, надаючи набір попередньо написаних бібліотек коду та модулів. Деякі з популярних web фреймворків включають Ruby on Rails, Django та Flask. Web платформи можуть пришвидшити процес розробки, надаючи попередньо написаний код для типових завдань web розробки, таких як обробка введених даних користувачами, керування сеансами та доступ до бази даних. Однак web фреймворки вимагають від розробників певного рівня технічних знань, і їх налаштування та обслуговування можуть бути складними.

Крім CMS і web фреймворків, існують інші програмні засоби, які розробники можуть використовувати для створення web додатків. До них належать інтегровані середовища розробки (IDE), редактори коду та бази даних. IDE — це програмна програма, яка надає розробникам комплексне середовище для написання, тестування та налагодження свого коду. Деякі популярні IDE включають Visual Studio,

Eclipse і NetBeans. Редактори коду — ще один тип програмного засобу, який використовується для розробки web додатків. Вони забезпечують спрощене середовище для редагування та керування кодом, а деякі популярні редактори коду включають Sublime Text, Atom і Notepad++. Бази даних необхідні для зберігання та керування даними у web додатках. Деякі з популярних систем керування базами даних включають MySQL, PostgreSQL і Microsoft SQL Server.

Загалом, не існує єдиного «найкращого» програмного засобу для розробки web додатків. Вибір програмного забезпечення залежить від ряду факторів, включаючи вимоги проекту, технічний досвід групи розробників і наявні ресурси. Розробники повинні враховувати ці фактори, обираючи програмні засоби для розробки своїх web додатків. У наступних розділах ми проаналізуємо функціональні вимоги та технічні характеристики web додатку «Європа для українців» та оцінимо програмні засоби, які найкраще відповідають цим вимогам.

1.2. Аналіз функціональних вимог до програмного забезпечення web додатків, які відповідають потребам проекту

Для визначення найбільш прийнятного програмного забезпечення для розробки web додатку «Європа для українців» важливо проаналізувати функціональні вимоги проекту. Функціональні вимоги стосуються специфічних функцій і функцій, які web додаток повинен мати, щоб задовольняти потреби своїх користувачів. Ці вимоги можуть варіюватися від основних функцій, таких як реєстрація користувача та вхід, до більш складних функцій, таких як інтеграція зі сторонніми API та платформами соціальних мереж.

Web додаток «Європа для українців» має на меті надати українцям платформу для знайомства з різними країнами Європи, зокрема їхнім кліматом, культурою праці, системою охорони здоров’я та податками. Для досягнення цієї мети web додаток повинен мати кілька важливих функцій. По-перше, він повинен забезпечувати інтуїтивно зрозумілий і зручний інтерфейс, який дозволяє користувачам переміщатися web сайтом і легко знаходити потрібну інформацію. По-друге, він повинен мати фу-

нкцію порівняння, яка дозволяє користувачам порівнювати конкретні країни, що цікавлять.

Окрім цих базових можливостей, web додаток «Європа для українців» має повинен мати ще кілька розширених функцій. Вони включають інтеграцію зі сторонніми API, такими як Карти Google і соціальні медіа-платформи, такі як Facebook і Twitter, щоб надавати користувачам актуальну інформацію та можливості обміну в соціальних мережах. Web програма також повинна мати адаптивний дизайн, який дозволяє отримати до неї доступ із різних пристроїв, таких як персональні комп'ютери, ноутбуки, планшети та смартфони. Нарешті, web додаток має бути безпечним і мати заходи для захисту даних користувачів і запобігання несанкціонованому доступу.

З огляду на функціональні вимоги web додатку «Європа для українців», необхідно оцінити варіанти програмного забезпечення, які можуть ефективно відповідати цим вимогам.

1.3. Порівняння програмних продуктів, опис переваг і недоліків кожного

Проаналізувавши функціональні вимоги web додатку «Європа для українців», важливо порівняти програмні продукти, які цим вимогам відповідають (табл. 1.1).

Хоча існує декілька програмних продуктів, доступних для розробки web додатків, кожен має свої переваги та недоліки. У випадку з web додатком «Європа для українців» VS Code було обрано через простоту використання, гнучкість і підтримку різних технологій.

Враховуючи ці технічні вимоги, VS Code, програмне забезпечення для розробки web додатку «Європа для українців», має ряд переваг. Він також підтримує системи контролю версій, такі як Git, що полегшує співпрацю з іншими розробниками та керування змінами коду.

Таблиця 1.1

Порівняльна характеристика ПП

ПП	**Visual Studio Code**	**Sublime Text**	**Atom**
Опис	Легкий, швидкий і ефективний редактор коду з підтримкою широкого діапазону мов програмування та фреймворків	Легкий редактор коду, який підтримує декілька мов програмування та фреймворків, відомий своєю швидкістю та продуктивністю.	Редактор відкритого вихідного коду з настроюваним інтерфейсом і широкою підтримкою мов, що підкреслює розширюваність і адаптивність.
Особливості	Вбудовані засоби налагодження та розширення, велика бібліотека розширень, що надає додаткові функції, такі як форматування коду, налагодження та тестування.	Велика бібліотека плагінів, що додають редактору додаткові функції.	Настроюваний інтерфейс користувача, велика бібліотека плагінів, що покращують функціональність
Переваги	Простота використання, гнучкість і підтримка різних технологій.	Швидкість і продуктивність, створені для швидкого реагування навіть із великою кодовою базою.	Розширюваність, налаштовується для різних робочих процесів і випадків використання.
Обмеження	Жодного	Відсутність вбудованих інструментів налагодження, менша бібліотека розширення порівняно з VS Code.	Може бути повільним і ресурсомістким, особливо з великою кодовою базою.

1.4. Вибір JavaScript для розробки web додатків

Коли йдеться про розробку web додатків, вибір правильної мови програмування має вирішальне значення. JavaScript є однією з найпопулярніших мов програмування для web розробки завдяки своїй універсальності, простоті використання та великій бібліотеці фреймворків і бібліотек. JavaScript — це мова сценаріїв, яка працює на стороні клієнта і може використовуватися для всього, від динамічної анімації сторінок до створення складних web додатків.

Однією з переваг використання JavaScript для розробки web додатків є його універсальність. JavaScript можна використовувати для широкого кола завдань, від простої перевірки форм до розробки складних web додатків. Його також можна використовувати для створення інтерактивних web інтерфейсів і анімації, що робить його чудовим вибором для створення привабливого досвіду для користувачів.

Ще однією перевагою використання JavaScript є простота використання. JavaScript — це зручна для початківців мова програмування, яку відносно легко вивчити, що робить її популярним вибором для розробників усіх рівнів кваліфікації. Крім того, JavaScript має величезну бібліотеку фреймворків і бібліотек, які можна використовувати для оптимізації процесу розробки та скорочення часу та зусиль, необхідних для створення web додатків.

Зрештою, JavaScript широко підтримується сучасними web браузерами, що робить його чудовим вибором для розробки кросплатформних web додатків. Це означає, що web додатки, створені за допомогою JavaScript, працюватимуть на широкому діапазоні пристроїв і платформ, від настільних комп’ютерів до мобільних пристроїв і всього між ними.

Вибір JavaScript було мудрим рішенням. Його універсальність, простота використання та крос-платформна сумісність роблять його ідеальним вибором для розробки web додатків.

1.5. Вибір технологій для розробки web додатків

Коли йдеться про розробку web додатку, вибір правильних технологій має важливе значення для забезпечення якості, продуктивності та зручності використання програми. Для web додатку «Європа для українців» використовували різноманітні технології, зокрема HTML, CSS, Sass, Bourbon та BEM.

1.5.1. Розуміння мови HTML у розробці web додатків

HTML або Hypertext Markup Language — це стандартна мова розмітки, яка використовується для створення web сторінок і web додатків. Він є основою кожної web сторінки та відповідає за структуру та вміст web сторінки. HTML використовує теги для визначення різних елементів web сторінки, таких як заголовки, абзаци, зображення, посилання та таблиці.

HTML — це декларативна мова, яка забезпечує чітку структуру web сторінок, що робить їх легкими для читання та розуміння. HTML5, остання версія HTML, представила нові елементи, які дозволяють розробникам створювати більш інтерактивні та динамічні web сторінки, такі як відео та аудіоплеєри, функцію перетягування та полотно для графіки.

HTML також дозволяє розробникам створювати web сторінки, доступні для всіх користувачів, у тому числі для людей з обмеженими можливостями. Мова підтримує використання допоміжних технологій, таких як програми зчитування з екрана, і надає розробникам можливість додавати альтернативний текст до зображень, підписи до відео та інші функції доступності.

У розробці web додатків HTML працює разом із CSS і JavaScript для створення адаптивного та інтерактивного інтерфейсу користувача. CSS використовується для стилізації елементів, визначених у HTML, тоді як JavaScript використовується для додавання динамічних функцій та інтерактивності користувача до web сторінки. У поєднанні з CSS і JavaScript HTML є потужним інструментом для створення сучасних і адаптивних web додатків.

1.5.2. CSS або каскадні таблиці стилів

CSS, або каскадні таблиці стилів, — це мова таблиць стилів, яка використовується для опису подання документа, написаного в HTML або XML. CSS використовується для керування макетом і зовнішнім виглядом web сайту, включаючи шрифти, кольори та інтервали між текстом, а також розташування та розмір зображень та інших елементів. Це важливий інструмент для створення візуально привабливих web сайтів, зручних для користувача та легких у навігації.

Однією з головних переваг використання CSS є те, що він дозволяє розробникам відокремити презентацію web сайту від його вмісту. Це означає, що HTML-код можна підтримувати чистим і вільним від непотрібного форматування, що полегшує читання та редагування. Використовуючи CSS для оформлення web сайту, можна швидко й легко змінити зовнішній вигляд сайту без необхідності змінювати базовий HTML-код.

CSS також пропонує широкий спектр функцій і параметрів для налаштування, що робить його універсальним інструментом для web розробників. За допомогою CSS розробники можуть створювати адаптивний дизайн, який адаптується до різних розмірів екрана та пристроїв, забезпечуючи чудовий вигляд їхніх web сайтів на настільних комп’ютерах, ноутбуках, планшетах і смартфонах.

Окрім стандартного CSS, існує також низка CSS-фреймворків і препроцесорів, які можуть допомогти розробникам оптимізувати процес розробки та створювати складніші проекти з меншою кількістю коду. Деякі популярні фреймворки CSS включають Bootstrap, Foundation і Materialize.

1.5.3. Переваги використання SCSS для стилізації web додатків

У той час як CSS був стандартом для оформлення web додатків, Sass (Syntactically Awesome Style Sheets) набув популярності завдяки своїм додатковим функціям і перевагам, у тому числі SCSS (Sassy CSS) – синтаксису Sass, який можна читати та підтримувати. SCSS — це мова препроцесора, яка дозволяє розробникам

писати більш ефективний і організований код CSS. SCSS має кілька переваг перед традиційним CSS.

По-перше, SCSS дозволяє створювати змінні, які можна використовувати для зберігання та повторного використання таких значень, як колірні коди та розмір шрифту, у всій програмі. Ця функція економить час і зусилля, оскільки усуває необхідність багаторазового запису тих самих значень у різних частинах програми.

По-друге, SCSS дозволяє створювати міксини, які є повторно використовуваними блоками коду. Ці міксини можна використовувати для загальних стилів, таких як певний макет або анімація, і їх можна легко інтегрувати в різні частини програми. Ця функція забезпечує послідовність і економить час, зменшуючи кількість коду, необхідного для написання.

Іншою перевагою SCSS є можливість вкладення селекторів CSS, що створює більш організовану структуру коду, яку можна підтримувати. Ця функція вкладеності полегшує розробникам розуміння зв’язку між різними частинами коду та забезпечує ефективніший і результативний процес кодування.

Крім того, SCSS пропонує функцію успадкування, яка дозволяє розробникам успадковувати властивості від одного класу до іншого. Ця функція особливо корисна під час роботи з повторюваними стилями та елементами в програмі. Завдяки успадкуванню розробники можуть зменшити кількість коду, який їм потрібно написати, заощадивши час і зусилля.

Зі зростанням складності web додатків використання SCSS стає все більш важливим для створення візуально привабливого та зручного досвіду для web користувачів.

1.5.4. Bourbon – бібліотека для Sass

Bourbon — це популярна бібліотека для Sass, яка надає набір міксинів і функцій, які полегшують написання ефективного та зручного для обслуговування коду CSS.

Однією з переваг використання Bourbon є те, що він дозволяє розробникам писати чистий і лаконічний код CSS, який легко читати та розуміти. Bourbon також допомагає зменшити кількість коду, необхідного для досягнення певних ефектів стилю, що може допомогти пришвидшити час розробки.

Крім того, модульна архітектура Bourbon дозволяє легко налаштувати та розширити його функціональність відповідно до потреб конкретного проекту. Також, Bourbon має активну спільноту розробників, які роблять внесок у його постійний розвиток і надають підтримку через форуми та інші ресурси.

Хоча існують інші бібліотеки для Sass, Bourbon виділяється своєю простотою, гнучкістю та надійними функціями, що робить його чудовим вибором для web розробників, які прагнуть покращити свій робочий процес CSS.

1.5.5. BEM (Block Element Modifier)

BEM (Block Element Modifier) — це методологія для написання коду CSS, яка має на меті зробити його більш читабельним, зручним для обслуговування та масштабованим. Він передбачає розбиття інтерфейсу користувача на невеликі, незалежні та багаторазово використовувані будівельні блоки, які називаються «блоками», які складаються з одного або кількох «елементів» і можуть бути вкладені один в одного. BEM також наголошує на використанні «модифікаторів» для зміни зовнішнього вигляду або поведінки блоків або елементів.

Однією з головних переваг використання BEM є його модульний підхід, який дозволяє спростити обслуговування та модифікацію коду. Розбиваючи інтерфейс користувача на більш дрібні компоненти, розробники можуть вносити зміни в окремі блоки або елементи, не впливаючи на решту кодової бази. Цей підхід також сприяє багаторазовому використанню коду, оскільки блоки можна повторно використовувати на кількох сторінках або проектах.

Ще однією перевагою BEM є його самодокументований характер. Правила іменування, які використовуються в BEM, полегшують розуміння мети та структури коду. Назви блоків зазвичай описують їх призначення, а елементи та модифікатори

називаються відповідно до їх зв’язку з блоком. Це полегшує новим розробникам приєднатися до проекту та швидко зрозуміти кодову базу.

BEM також допомагає уникнути конфліктів між стилями, використовуючи конкретні імена класів для кожного блоку, елемента та модифікатора. Ця специфіка гарантує, що стилі застосовуються лише до призначеного елемента та запобігають ненавмисним каскадним ефектам. Крім того, BEM забезпечує чітку ієрархію стилів, що полегшує організацію та структурування коду CSS.

Використання методології BEM у CSS може принести багато переваг проектам web розробки, включаючи покращену читабельність коду, зручність обслуговування, масштабованість, багаторазове використання, самодокументування та уникнення конфліктів стилів. Запровадження BEM може допомогти розробникам ефективніше керувати своїм кодом і створювати більш послідовні та надійні web додатки, що робить його корисним інструментом для проекту web додатків «Європа для українців».

1.6. Вибір мови PHP для обробки запитів

Під час розробки web додатку вибір мови програмування для серверної частини є вирішальним. Для проекту «Європа для українців» мовою обробки запитів контактної форми обрано PHP. PHP — це серверна мова сценаріїв, яка широко використовується для web розробки. Він відкритий, безкоштовний і має велику спільноту розробників.

PHP добре підходить для обробки даних форми, оскільки він має вбудовані функції для обробки введених даних. Це робить його ідеальним вибором для роботи контактної форми в проекті «Європа для українців». PHP також простий у вивченні та містить велику кількість навчальних посібників і ресурсів, доступних в Інтернеті, що робить його доступним для розробників з різним рівнем досвіду.

Порівняно з іншими популярними серверними мовами програмування, такими як Python, Ruby та Node.js, PHP має ряд переваг. Однією з головних переваг PHP є його швидкість. Скрипти PHP виконуються швидко, що важливо для web додатків,

яким потрібно обробляти великі обсяги даних у режимі реального часу. Ще однією перевагою PHP є його легкість інтеграції з такими популярними базами даних, як MySQL, що робить його популярним вибором для створення динамічних web додатків.

Хоча PHP має багато переваг, він не позбавлений і недоліків. Однією з потенційних проблем з PHP є його безпека. Оскільки це така широко використовувана мова, вона часто стає мішенню хакерів, які шукають уразливості у web додатках на основі PHP. Однак цей ризик можна зменшити, дотримуючись найкращих практик безпеки web додатків і оновлюючи PHP з останніми виправленнями безпеки.

Загалом PHP був найкращим вибором для обробки запитів через контактну форму в проекті «Європа для українців». Це швидка, проста у вивченні мова з великою спільнотою розробників і широким спектром доступних фреймворків і бібліотек. Хоча він має деякі потенційні ризики безпеці, їх можна вирішити, дотримуючись найкращих практик безпеки web додатків.

1.7. Вивчення переваг jQuery у розробці web додатку

У міру розвитку web додатку «Європа для українців» вирішальним є вибір відповідного програмного забезпечення для його впровадження.

jQuery — це швидка, невелика та багатофункціональна бібліотека JavaScript, яка спрощує розробку сценаріїв на стороні клієнта. Це потужний інструмент для web розробників, особливо тих, хто хоче створювати інтерактивні та динамічні web сторінки. За допомогою jQuery розробники можуть легко маніпулювати елементами HTML, створювати анімацію та керувати подіями. Крім того, це спрощує використання Ajax, дозволяючи легко оновлювати вміст web сторінки без перезавантаження всієї сторінки.

Однією із вагомих переваг використання jQuery у розробці web додатку «Європа для українців» є його кросбраузерність. jQuery дозволяє легко писати код, який бездоганно працює в кількох web браузерах, усуваючи необхідність писати окремий

код для кожного браузера. Ця функція економить розробникам значну кількість часу та зусиль, що робить її цінним інструментом для web розробки.

Рис. 1.2. Обрані технології розробки web додатків

Web додаток «Європа для українців» може отримати значну користь від використання jQuery у своїй розробці. Завдяки сумісності з різними браузерами, великій бібліотеці плагінів і простоті використання jQuery може спростити процес розробки та розширити функціональність web додатку. У результаті web додаток стане більш інтерактивним, динамічним і зручним для користувача, забезпечуючи чудовий досвід користувача для цільової аудиторії.

1.8. Висновок до розділу 1

У першому розділі було проведено ретельний розгляд та вибір програмного забезпечення для розробки web додатків.

Процес вибору програмного забезпечення включав оцінку різних варіантів з урахуванням конкретних вимог проекту, таких як ефективність, простота використання та гнучкість.

Для серверної мови сценаріїв було обрано PHP через його універсальність, сумісність і широке використання в спільноті web розробників. Вибір PHP дозволив ефективно обробляти запити та взаємодіяти з базою даних.

Для інтерфейсної розробки HTML використовувався для створення структури програми та макета, тоді як CSS і SCSS використовувалися для стилізації вмісту та компонування сторінок. Bourbon використовувався як платформа SCSS для спрощення та оптимізації процесу розробки. Методологія BEM була використана для

створення більш багаторазового та модульного коду CSS, що призвело до скорочення часу розробки та меншої кількості помилок.

JQuery використовувався як бібліотека JavaScript для оптимізації процесу розробки та сприяння складній інтерфейсній взаємодії.

Загалом результатом вибору та використання програмного забезпечення при розробці web додатку «Європа для українців» став ефективний, гнучкий та високофункціональний додаток. Вибір PHP, HTML, CSS, SCSS, Bourbon, BEM і JQuery виявився виграшною комбінацією, що дозволило команді розробників створити web програму найвищої якості, яка відповідала та перевершила очікування клієнтів.

РОЗДІЛ 2
РОЗРОБКА WEB-ДОДАТКУ «ЄВРОПА ДЛЯ УКРАЇНЦІВ»

2.1. Комплексний аналіз факторів, які враховують українські біженці при виборі країни для поселення в Європейському Союзі.

Рішення оселитися в чужій країні є вирішальним для українських біженців, які шукають тимчасового захисту в Європейському Союзі. Щоб полегшити їм процес прийняття рішень, важливо провести комплексний аналіз різних факторів, які вони враховують. У цьому підрозділі представлено скорочений аналіз клімату, вартості життя, умов працевлаштування, податків, системи охорони здоров'я та програм допомоги для українців у вибраних країнах-членах ЄС: Австрії, Бельгії, Іспанії, Італії, Нідерландах, Німеччині, Польщі, Португалії, Словаччині, Угорщина, Франція та Чехія. Вибір цих країн виправданий їх актуальністю як популярних напрямків для українських біженців на основі попередніх досліджень та міркувань. Обрані фактори були ретельно відібрані з огляду на їх доречність і значення в процесі прийняття рішень українськими біженцями, які шукають поселення в Європейському Союзі.

Клімат відіграє велику роль у визначенні придатності та комфортності країни для українських біженців. Це може вплинути на їхнє фізичне самопочуття, уподобання в способі життя та адаптацію до нового середовища. Враховуючи кліматичний фактор, web додаток може надати цінну інформацію, щоб допомогти українським біженцям вибрати країну, яка відповідає їхнім кліматичним уподобанням і підтримує їх загальний добробут.

Вартість життя є життєво важливою для українських біженців, оскільки вона безпосередньо впливає на їхню фінансову стабільність і якість життя в новій країні. Надаючи інформацію про вартість житла, товарів і послуг, web додаток може допомогти біженцям оцінити доступність і фінансові наслідки поселення в різних країнах-членах ЄС.

Можливості та **умови працевлаштування** є важливими факторами для українських біженців, які прагнуть осісти в новій країні. Інформація про ринки праці, галузі та трудове законодавство може допомогти їм оцінити потенційні перспективи кар'єри, рівень доходу та загальні умови працевлаштування. Враховуючи цей фактор, web додаток може підтримати біженців у пошуку відповідних можливостей працевлаштування та покращити їхні перспективи інтеграції.

Податкові системи мають значний вплив на фінансові аспекти поселення в конкретній країні. Враховуючи податкові ставки, розмір доходу та потенційні відрахування, web додаток може допомогти українським біженцям зрозуміти податкові наслідки та прийняти обґрунтовані рішення щодо свого фінансового планування та зобов'язань у вибраній країні-члені ЄС.

Доступ до якісних **медичних послу**г є вирішальним фактором для людей, які осідають у новій країні. Надаючи інформацію про системи охорони здоров'я, охоплення та послуги, web додаток може допомогти українським біженцям оцінити наявність та якість медичного обслуговування в різних країнах-членах ЄС. Цей фактор особливо важливий, оскільки він стосується їхнього здоров'я та благополуччя.

Програми **соціальної допомоги** та інтеграційної підтримки є важливими факторами для успішного поселення та інтеграції українських біженців. Включаючи інформацію про доступні програми, фінансову допомогу, мовне навчання та інтеграційні ініціативи, web додаток може допомогти біженцям орієнтуватися в системах підтримки в різних країнах-членах ЄС і отримати доступ до необхідних ресурсів для їх інтеграції.

Аналіз дає цінну інформацію для розвитку web додатку «Європа для українців», що дає змогу українським біженцям отримати доступ до вичерпної інформації та приймати зважені рішення щодо поселення в Європейському Союзі.

2.2. Проектування користувацького інтерфейсу та користувацького досвіду (UI/UX design).

Термін користувальницький інтерфейс (User Interface - UI) стосується візуального аспекту продукту, що охоплює такі елементи, як кольори, дизайн, анімація, вміст і різні компоненти, такі як кнопки, мітки та поля введення. З іншого боку, UX (користувальницький досвід) пов'язаний із загальною взаємодією та емоціями користувача під час використання інтерфейсу, будь то на комп'ютері чи мобільному пристрої. Він зосереджується на легкості або труднощах, з якими стикаються користувачі під час виконання запланованих завдань, таких як здійснення покупки, планування маршруту, перевірка цін або доступ до інформації про погоду.

По суті, головну відмінність між UI та UX можна порівняти з автомобілем: UI визначає його зовнішній вигляд, тоді як UX формує його функціональність. UX-дизайн передбачає створення продукту, орієнтованого на користувача, враховуючи подорож користувача через інтерфейс, його взаємодію та викликану емоційну реакцію. З іншого боку, UI-дизайн зосереджується на візуальній реалізації функцій інтерфейсу.

Підводячи підсумок, UX-дизайн спрямований на вирішення проблем користувачів, тоді як UI-дизайн передбачає створення інтуїтивно зрозумілих і візуально привабливих інтерфейсів, які полегшують взаємодію користувачів.

2.2.1. Орієнтований на користувача (User-Centered) підхід до проектування

В основі дизайну UI/UX web додатку «Європа для українців» було вирішено використовувати підхід, орієнтований на користувача. Цей підхід надасть нам пріоритет розумінню потреб, цілей і вподобань цільових користувачів і адаптації дизайну відповідно до їхніх вимог. Він передбачає проведення обширних досліджень, збирання думок користувачів і повторення дизайнерських рішень на основі відгуків користувачів. Розміщуючи користувачів у центрі процесу розробки, ми маємо на

меті забезпечити безперебійу і змістовну роботу додатку для українських біженців, які шукають інформацію та допомогу в переїзді до Європейського Союзу.

Першим кроком підходу до UCD є проведення комплексного дослідження та аналізу користувачів. Перш за все, зберемо інформацію про цільових користувачів. Щоб зрозуміти потреби та цілі українських біженців, використаємо різні методи дослідження, такі як інтерв'ю, опитування та спостереження. Розуміючи їхні унікальні обставини, культурне походження та конкретні проблеми, ми зможемо розробити більш чуйний та ефективний інтерфейс.

На основі зібраних даних було виявлено загальні закономірності (шаблони). Ці шаблони були пов'язані з уподобаннями, поведінкою або проблемами користувачів. Наприклад, можна помітити, що українські біженці віддають перевагу країнам із сприятливими можливостями працевлаштування та доступною вартістю життя. Виявлення цих закономірностей допомагає зрозуміти колективні потреби та очікування цільової групи користувачів, які потім можуть інформувати про дизайнерські рішення та визначати пріоритети особливостей і функцій web додатку «Європа для українців».

Персони користувача – це вигадане представлення архетипів користувачів у цільовій групі користувачів. Вони створені на основі інформації, отриманої в результаті досліджень і аналізу користувачів. Особи користувачів допомагають дизайнерам співпереживати користувачам і приймати дизайнерські рішення, які відповідають їхнім конкретним потребам і цілям.

У контексті web додатку «Європа для українців» особи користувача включають такі профілі, як *«Марія, молодий фахівець, який шукає можливості працевлаштування в країні з сильною економікою»,* або *«Андрій, орієнтована на сім'ю особа, яка шукає доступні варіанти житла та хороші можливості для навчання своїх дітей».*

Ці персонажі забезпечують орієнтовану на людину перспективу та керують процесом проектування, гарантуючи, що програма відповідає різноманітним потребам і очікуванням цільових користувачів.

Карти шляху користувача візуально зображують різні етапи та точки взаємодії користувача з web програмою. Вони ілюструють досвід користувача від початкової взаємодії з програмою до досягнення своїх цілей. Карти шляху користувача допомагають визначити можливості для покращення та сфери, де користувачеві може знадобитися допомога чи керівництво.

Карта подорожі клієнта для Марії – шукає можливості переїзду до іншої країни.

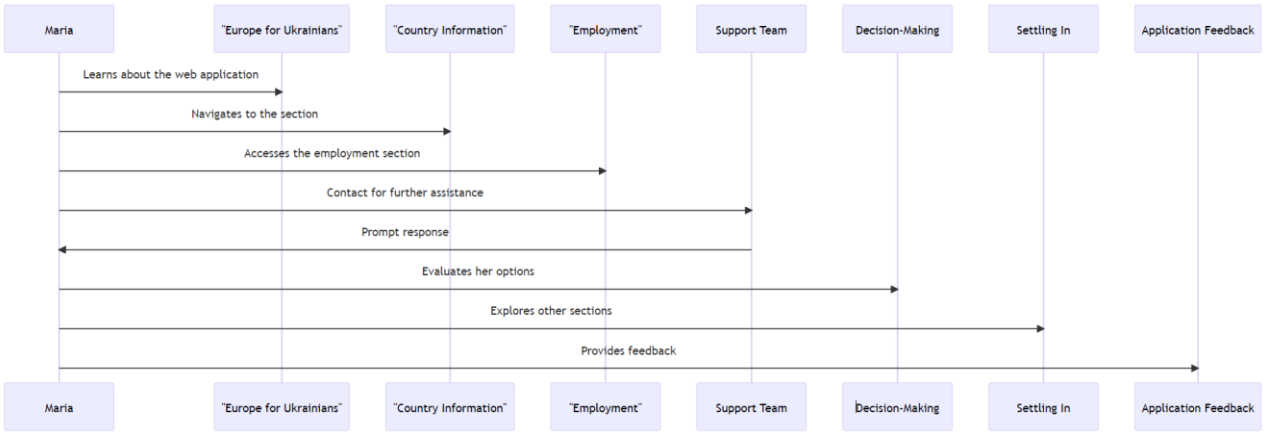

Рис. 2.1. Карта подорожі клієнта для Марії

Етап 1. Відкриття:

- Про web додаток «Європа для українців» Марія дізнається з уст в уста чи онлайн-досліджень;

- Вона відвідує web сайт і відразу помічає зручний інтерфейс і зрозумілу навігацію.

Етап 2. Вивчення інформації про країну:

- Марія переходить до розділу «Країни» в додатку, щоб ознайомитися з деталями про різні країни;

- Вона обирає країну з сильною економікою та можливостями працевлаштування на основі своїх уподобань.

Етап 3. Збір інформації про роботу:

- Марія переходить до розділу працевлаштування вибраної країни та знаходить вичерпну інформацію про перспективи роботи, галузі та дозволи на роботу;

- Вона вивчає діапазон середньої заробітної плати та переваги при працевлаштуванні, переконавшись, що вони відповідають її очікуванням.

Етап 4. Звернення за додатковою допомогою:

- Марія хоче отримати більш конкретну інформацію, пов'язану з її професією, і вирішує звернутися до служби підтримки;

- Вона заповнює контактну форму, вказуючи своє ім’я, електронну адресу та повідомлення зі своїм запитом.

Етап 5. Отримання підтримки та вказівок:

- Марія отримує швидку відповідь від команди підтримки, яка відповідає її запиту та надає додаткові ресурси чи контакти;

- Вона веде подальше листування, щоб з’ясувати будь-які сумніви або отримати додаткову допомогу.

Етап 6. Прийняття рішення:

- Озброєна відповідною інформацією та підтримкою, Марія оцінює свої можливості та порівнює можливості, доступні в різних країнах;

- Вона зважує такі фактори, як наявність роботи, зарплата, вартість та якість життя, щоб прийняти обґрунтоване рішення.

Етап 7. Влаштування:

- Коли Марія вибирає країну, вона вивчає інші розділи web додатку, щоб зібрати інформацію про клімат, середню вартість оренди житла, податки, системи охорони здоров’я та програми соціальної допомоги;

- Вона використовує цю інформацію, щоб спланувати свій переїзд і прийняти необхідні заходи для свого нового життя у вибраній країні.

Етап 8. Зворотній зв'язок із заявкою:

- Після успішного освоєння нової країни Марія може надати відгук про свій досвід використання програми;

- Вона може висловити своє задоволення інтерфейсом користувача, актуальністю інформації та отриманою підтримкою.

Ця карта подорожі клієнта ілюструє типові етапи та точки дотику, які проходить Марія під час використання web додатку «Європа для українців» для пошуку можливостей працевлаштування. Зручний дизайн web-додатку та повна інформація дозволяють Марії приймати зважені рішення та знаходити підтримку, необхідну для досягнення своїх кар’єрних цілей у країні з сильною економікою.

Створюючи карти подорожі користувача, ми отримуємо уявлення про те, як користувачі переміщуються в програмі, яку інформацію вони шукають і де вони можуть зіткнутися з труднощами чи розчаруваннями. Таке розуміння дає змогу нам оптимізувати потік користувачів, визначити сфери, які потрібно вдосконалити, і переконатися, що програма забезпечує безперебійний та інтуїтивно зрозумілий досвід для українських біженців, які шукають інформацію про поселення в Європейському Союзі.

Тож тепер, маючи чітке розуміння цільових користувачів, створимо ідею та концептуалізацію для створення дизайнерських рішень. Для цього розробимо ескізи, каркасні конструкції та створимо прототипи низької точності. Основну увагу приділимо дослідженню різноманітних можливостей дизайну та перетворенню потреб користувачів у конкретні елементи дизайну.

Далі згенеруємо ідеї та оцінимо їх здійсненність і відповідність вимогам користувачів. Мета полягає в тому, щоб створити основу дизайну, яка відповідає очікуванням користувачів і забезпечує основу для подальших ітерацій.

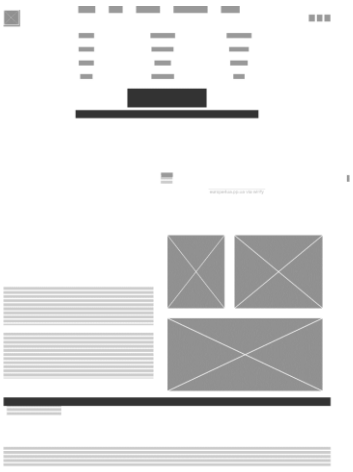

Рис. 2.2. Каркасна конструкція сторінки «Країна»

Рис. 2.3. Каркасна конструкція сторінки «Контакти»

2.2.2. Ітеративне проектування та прототипування

Підхід до проектування, орієнтований на користувача, наголошує на ітераційному процесі проектування, коли дизайн постійно вдосконалюється на основі відгуків користувачів і тестування зручності використання. Високоякісні прототипи створюються для імітації взаємодії з користувачем і збору уявлень про зручність використання та ефективність дизайнерських рішень.

Для високоякісного прототипування використаємо багатоплатформовий онлайн-сервіс для web дизайну під назвою Figma. За допомогою нього ми створимо дизайн сайту та різні елементи інтерфейсу.

Щоб створити прототип головної сторінки web додатку «Європа для українців» за допомогою Figma, виконаємо наступні дії.

Крок 1: Ознайомимося з Figma. Оскільки ми раніше не користувалися Figma, розпочнемо із ознайомлення з її інтерфейсом і основними функціями. Ознайомимося з різними інструментами та панелями дизайну, доступними для ефективного створення прототипу.

Крок 2: Визначимо структуру та макет. Почнемо із визначення структури та макета головної сторінки. Розглянемо ключові компоненти, які необхідно включити, наприклад панель навігації, представлені країни та відповідні інформаційні розділи.

Оскільки у нас уже є готовий каркас сторінки тож будемо орієнтуватись на нього. Далі використаємо інструменти векторного дизайну Figma, щоб створити та розташувати ці компоненти на полотні, забезпечуючи візуально привабливий і зручний макет.

Крок 3: Додаємо візуальні елементи та стиль Після створення макета додамо візуальні елементи, такі як зображення, піктограми та текст, щоб представити вміст головної сторінки. Розглянемо можливість використання відповідних зображень, щоб продемонструвати різноманітність європейських країн і привернути увагу користувачів. Застосуємо обраний стиль, включаючи шрифти, кольори та інтервали, щоб забезпечити послідовність і візуальну привабливість. Піч час одного із сеансів мозкового штурму було вирішено головну сторінку робити у кольорах українського прапору, а сторінки ріхних країн робити у кольоровій палітрі кожної країни.

Крок 4: Створимо інтерактивні елементи. Однією з переваг Figma є можливість створювати інтерактивні прототипи. Покращимо прототип головної сторінки, додаючи інтерактивні елементи, такі як кнопки, спадні меню та ефекти наведення курсора.

На даному етапі макет головної сторінки виглядає так.

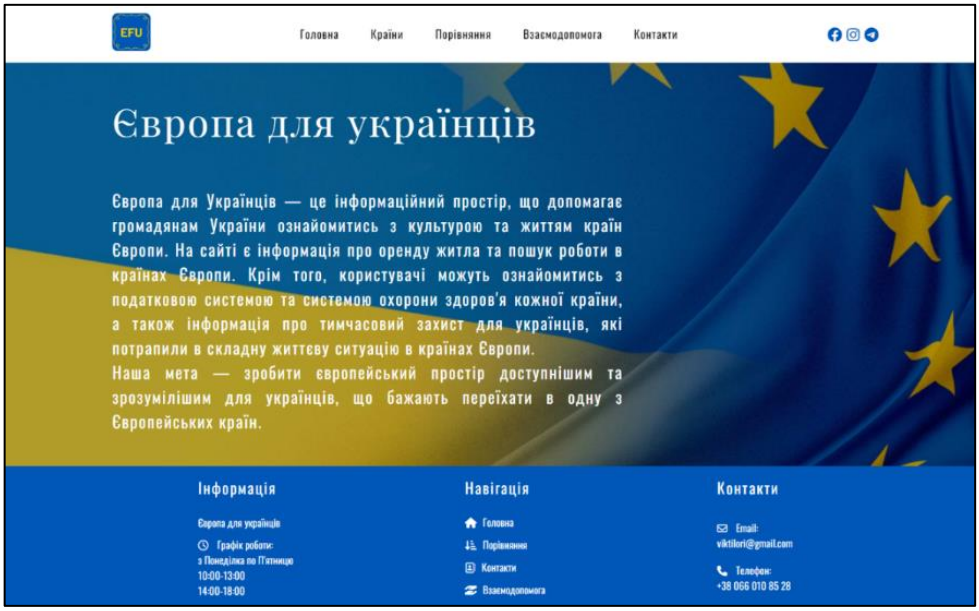

Рис 2.4. Макет головної сторінки створений за допомогою Figma

Рис 2.5. Прототип спливаючого меню

Визначимо поведінку цих елементів за допомогою функцій прототипування Figma. Наприклад, навігацію між сторінками або спливаючі режими для отримання додаткової інформації.

Рис. 2.6. та Рис. 2.7. Поведінка деяких елементів при наведенні

Крок 5: Визначимо потоки користувачів і взаємодію. Врахуємо потоки користувачів і взаємодію, в яку користувачі братимуть участь під час навігації головною сторінкою. Визначимо, як користувачі будуть отримувати доступ до різних розділів, вивчати інформацію про країни та взаємодіяти з функціями пошуку.

Для висвітлення інформації про країни було обрано використати спеціальну форму перемикачів Tabs.

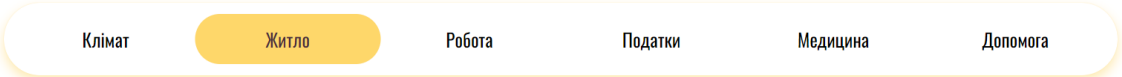

Рис. 2.8. Макет перемикачів між інформацією на сторінці «Країна»

Крок 6: Перевіримо та повторимо. Після створення прототипу переконаємось в зручності використання. Поділимося прототипом з потенційними користувачами або зацікавленими сторонами та проаналізуємо їхню взаємодію. Проаналізуємо

отримані відгуки та зробимо необхідні ітерації, щоб покращити взаємодію з користувачем і вирішимо будь-які проблеми зручності використання, виявлені під час перевірки.

Під час перевірки зручності прототипу було виявлено занадто маленький розмір шрифту, та не чітке формулювання розділів сайту. Дякуючи відгукам, ці питання було перевірено та виправлено.

Дотримуючись цих кроків, ми створили прототипи сторінок web додатку «Європа для українців» за допомогою Figma. Під час розробки інтерфейсу не забуваємо про підхід, орієнтований на користувача, і пам’ятаємо про потреби та вподобання українських біженців. Прототип служитиме візуальним представленням функціональності та макета сторінок, полегшуючи процес розробки та надаючи зацікавленим сторонам чітке бачення кінцевого продукту.

2.3. Розробка інтерфейсу web додатку

Розробка інтерфейсу відіграє вирішальну роль у тому, щоб додаток надавав бездоганний та інтуїтивно зрозумілий досвід користувача (UX) українським біженцям, які шукають інформацію про країни Європейського Союзу (ЄС) для поселення.

Інтерфейс користувача служить основною точкою взаємодії між користувачами та web програмою. Він охоплює різноманітні візуальні та інтерактивні елементи, такі як меню, кнопки, форми та макети вмісту, які дозволяють користувачам здійснювати навігацію, отримувати доступ до інформації та виконувати завдання в програмі. Ефективність дизайну інтерфейсу користувача безпосередньо впливає на залучення користувачів, задоволення та загальний успіх програми.

2.4. Налаштування Visual Studio Code для розробки web додатків

Перший етап розробки web додатку полягаю у завантаженні та налаштуванні середовища розробки. У розділі 1 було представлено обрані технології для розробки web додатку «Європа для українців». Обраним IDE є Visual Studio Code. Щоб забез-

печити плавну та продуктивну розробку web додатку «Європа для українців», важливо налаштувати VS Code за допомогою необхідних розширень, налаштувань та інструментів, адаптованих до вимог web розробки.

1. Встановимо Visual Studio Code. Почнемо із завантаження та встановлення Visual Studio Code з офіційного web сайту (https://code.visualstudio.com/Download). Виберемо відповідну версію для потрібної операційної системи.

2. Встановимо необхідні розширення. VS Code пропонує широкий спектр розширень, які покращують його функціональність для web розробки. Встановимо наступні необхідні розширення для підтримки розробки web додатку «Європа для українців», перейшовши до перегляду розширень у VS Code (клацнемо піктограму квадрата на лівій бічній панелі) і знайдемо потрібні розширення за назвою:

- GitLab Workflow, щоб легко керувати контролем версій у VS Code;
- Live Server, щоб запустити локальний сервер розробки та переглядати зміни в реальному часі під час кодування;
- Prettier - Code Formatter, щоб автоматично форматувати код і підтримувати постійний стиль;
- SCSS Formatter, спеціальне розширення коду Visual Studio для форматування SCSS;
- Ukrainian Code Spell Checker - розширення українського словника для VS Code;
- Auto Rename Tag та Auto Close Tag - Автоматично перейменовувати парні теги та додавати закриваючий тег HTML/XML, як це робить Visual Studio IDE.

3. Налаштуємо та розширимо. Visual Studio Code можна налаштувати, що дозволяє персоналізувати середовище розробки відповідно до ваших уподобань. Ознайомтимось з різними налаштуваннями, темами та налаштуваннями, доступними у VS Code. Змінимо кольорову тему, налаштуємо розмір шрифту та встановимо додаткові розширення для подальшого покращення робочого процесу та продуктивності.

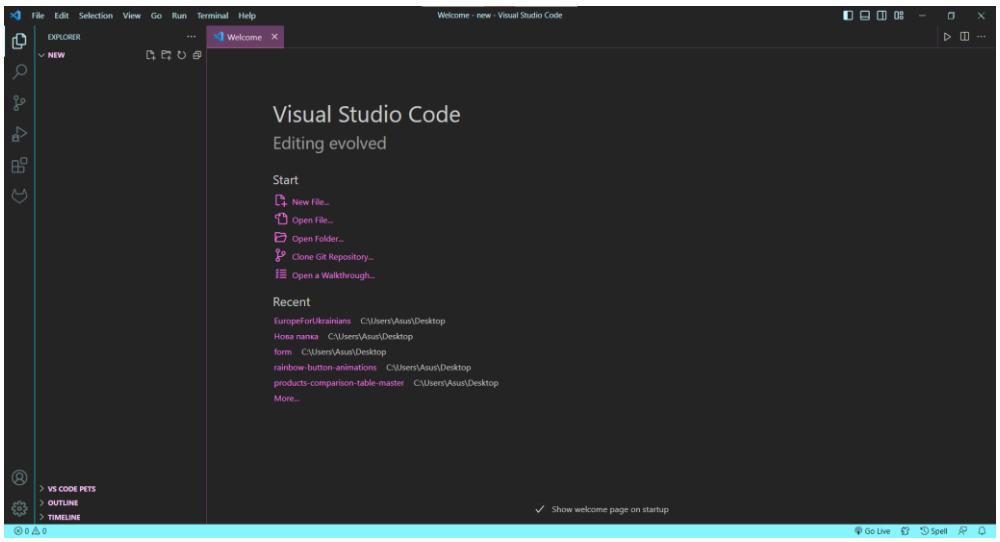

Рис. 2.9. Стартовий екран VS Code після встановлення усіх розширень

Після виконання цих кроків матимемо ось такий стартовий екран. (Рис. 9). Дотримуючись цих кроків, ми налаштували Visual Studio Code як ефективну та придатну IDE для розробки web додатку «Європа для українців». З належним чином налаштованим середовищем розробки ми будемо добре оснащені для написання чистого, ефективного та підтримуваного коду для web програми.

2.5. Реалізація елементів HTML та їхнє семантичне значення у web додатку

Використання належної структури HTML і семантичної розмітки має важливе значення для створення доступних web сайтів, які обслуговують різноманітне коло користувачів. Це також допомагає покращити видимість у пошуковій системі, рейтинг і взаємодію з користувачем, зрештою залучаючи більше органічного трафіку на ваш web сайт.

Продемонструємо HTML елементи, використані при створенні головної сторінки web-додатку. Давайте розглянемо код і виділимо семантичне значення використовуваних елементів.

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

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/css/france.css">
    <title>Франція для українців</title>
    <link rel="icon" type="image/x-icon" href="../img/favicon.ico">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Oswald&family=Playfair+Display&display=swap" rel="stylesheet">
    <script src="https://kit.fontawesome.com/b642661027.js" crossorigin="anonymous"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/meyer-reset/2.0/reset.min.css">
</head>
```

Рис. 2.10. Фрагмент коду з елементами «html» та «head»

Елемент `html` представляє кореневий елемент документа HTML. Атрибут `lang` визначає мову документа, що сприяє доступності та обробці для певної мови. Елемент `head` містить метадані та включає різні елементи, які покращують web програму. Наприклад:

- Елементи `meta` надають інформацію про кодування символів, сумісність і налаштування вікна перегляду.
- Елемент `title` встановлює назву web сторінки.
- Елементи `link` визначають зовнішні таблиці стилів, піктограму сайту та ресурси попереднього підключення для кращої продуктивності.
- Елемент `script` завантажує зовнішню бібліотеку JavaScript (у цьому випадку Font Awesome).

```html
<body>
    <div class="container">
        <div class="container__header">
            <header class="header">···
            </header>
        </div>
        <div class="container__main">···
        </div>
        <div class="container__footer">
            <footer>···
            </footer>
        </div>
    </div>
</body>
```

Рис. 2.11. Фрагмент коду с основними розділами

- Елемент `body` представляє вміст документа та містить основні розділи web програми.

- Елементи div з іменами класів забезпечують структурний поділ вмісту.
- Елемент `header` представляє вступний вміст або контейнер для навігації сайтом і брендингу.
- Елемент `footer` представляє розділ нижнього колонтитула, що містить інформацію про web програму.

```
<h1 class="container__mainTitle">Європа для українців</h1>
```

Рис. 2.12. Фрагмент коду з елементом «h1»

Елемент `h1` представляє головний заголовок сторінки, що передає назву «Європа для українців». Він позначає найважливіший заголовок на сторінці.

```
<nav>
    <ul class="links">
        <li class="header__item"><a href="index.html" class="active">Головна</a></li>
        <li class="header__item">
            <a href="#2">Країни</a>
            <div class="dropdown-content">···
            </div>
        </li>
        <li class="header__item"><a href="comparison-table/index.html">Порівняння</a></li>
        <li class="header__item"><a href="help/index.html">Взаємодопомога</a></li>
        <li class="header__item"><a href="contacts/index.html">Контакти</a></li>
    </ul>
</nav>
```

Рис. 2.13. Фрагмент коду з елементом «nav»

Елемент `nav` представляє розділ для навігаційних посилань. Він містить невпорядкований список (`ul`) з елементами списку (`li`). Елементи `a` всередині елементів списку представляють навігаційні посилання.

```
<a href="https://www.facebook.com/profile.php?id=100005502546434" target="_blank"
    rel="noopener noreferrer">
    <i class="fa-brands fa-facebook"></i>
</a>
```

Рис. 2.14. Фрагмент коду з гіперпосиланням

Елемент `a` представляє гіперпосилання. Він обгортає піктограму (елемент `i` з класами Font Awesome) і вказує призначення посилання за допомогою атрибута `href`. Атрибут `target="_blank"` відкриває посилання в новій вкладці, а атрибут `rel="noopener noreferrer"` використовується з міркувань безпеки.

```
<div class="container__mainDescription">···
</div>
```

Рис. 2.15. Фрагмент коду з контейнером «div»

Елемент `div` з класом `container__mainDescription` представляє контейнер для основного опису web програми.

```
<footer>
    <div class="links">···
    </div>
    <div class="bottom">···
    </div>
</footer>
```

Рис. 2.16. Фрагмент коду з структурою елементу «footer»

Елемент `footer` представляє розділ нижнього колонтитула сторінки. Він містить розділи для посилань і нижню частину колонтитула.

У коді різні інші елементи HTML, такі як `img`, `a`, `br`, і текстовий вміст, використовуються для структурування та представлення інформації web програми.

Ці елементи HTML сприяють структурі, доступності та семантиці web програми, дозволяючи браузерам, пошуковим системам і допоміжним технологіям правильно розуміти й обробляти вміст.

Також продемонструємо HTML елементи, використані при створенні сторінки країни, їх реалізацію та семантичне значення у web додатку. Давайте розберемо код і підкреслимо значення кожного елемента HTML.

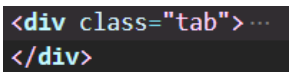

Рис. 2.17. Фрагмент коду елементу «div» з класом «tab»

Елемент `div` з класом `tab` представляє контейнер для навігації вкладками.

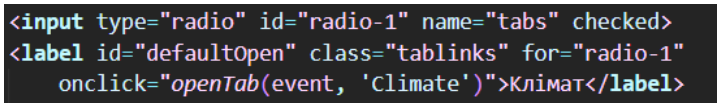

Рис. 2.18. Фрагмент коду елементу «input» та елементу «label»

Елемент `input` із `type="radio"` представляє перемикач, який відповідає певній вкладці. Елемент `label` пов'язано з перемикачем за допомогою атрибута `for` і надає текст мітки вкладки. Атрибут `onclick` визначає функцію JavaScript, яка буде виконуватись, коли клацнути мітку. Ця комбінація елементів представляє вкладку в навігації.

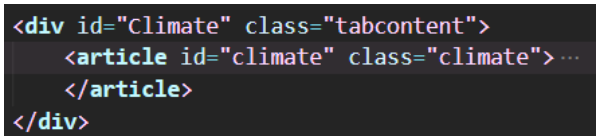

Рис. 2.19. Фрагмент коду елементу «div з ідентифікатором «Climate» і класом «tabcontent»

Елемент `div` з ідентифікатором `Climate` і класом `tabcontent` представляє контейнер для вмісту вкладки «Climate». Елемент `article` представляє самодостатню композицію в документі. Він містить інформацію про клімат у відповідній країні.

Решта розділів у коді дотримуються подібного шаблону, де елементи `div` представляють контейнери для вмісту вкладки, а елементи `article` представляють вміст окремої вкладки.

Використовуючи ці HTML-елементи та пов’язані з ними атрибути, код реалізує інтерфейс із вкладками, де користувачі можуть клацати мітки вкладок для перемикання між різними розділами вмісту. Це покращує взаємодію з користувачем, упорядковуючи інформацію та полегшуючи навігацію всередині web програми.

Наступним кроком продемонструємо використані HTML-елементи у розробці сторінки «Контакти». Ось пояснення елементів HTML та їхнього семантичного значення в наданому коді.

```
<div class="container__main">
    <div class="top">···
    </div>
    <div class="blocks">···
    </div>
</div>
```

Рис. 2.20. Фрагмент коду елементу «div» з класом «container__main»

Елемент `div` з класом `container__main` представляє основний контейнер для вмісту у web додатку. Він містить два підрозділи.

Елемент `div` з класом `top` представляє верхню частину контейнера.

Елемент `div` з класом `blocks` представляє розділ, що містить кілька блоків.

```
<form action="../php/sender.php" method="post" class="form">···
</form>
```

Рис. 2.21. Фрагмент коду елементу «form»

Елемент `form` представляє форму у web програмі. Він містить атрибут `action`, який визначає URL-адресу, куди будуть надсилатися дані форми. Для атрибута `method` встановлено значення «post», що вказує на те, що дані форми будуть надіслані за допомогою методу HTTP POST. Атрибут `class` має значення «form» для цілей стилю.

```
<h3>Форма зворотнього зв'язку</h3>
<h6>Надішліть, будь ласка, форму і ми з вами зв'яжемось</h6>
<input type="text" name="name" id="name" placeholder="Введіть своє ім'я" required>
```

Рис. 2.22.Фрагмент коду з елементами «h3», «h6», «input»

Елемент `h3` представляє заголовок у формі, що вказує на призначення форми, якою є «Форма зворотнього зв’язку» (Форма зворотного зв’язку).

Елемент `h6` представляє підзаголовок у формі, що містить інструкції або додаткову інформацію для користувача. У цьому випадку він радить користувачеві надіслати форму для зв’язку.

Елемент `input` представляє поле введення тексту у формі. Він має кілька атрибутів:

- Атрибут `type` має значення «text», що вказує на те, що це поле введення тексту.
- Для атрибута `name` встановлено значення «name», яке використовуватиметься як ідентифікатор цього введення під час надсилання форми.
- Атрибут `id` має значення «name», що надає унікальний ідентифікатор для поля введення.
- Атрибут `placeholder` надає підказку або приклад тексту, який допоможе користувачеві визначити, що вводити в поле.
- Атрибут `required` вказує, що введення є обов’язковим і має бути заповнене перед надсиланням форми.

Подібним чином існують елементи `введення` для електронної пошти, телефону та текстового поля для введення повідомлень.

```
<button type="button" class="btn btn--blue">
    <span class="btn__txt">Відправити</span>
    <i class="btn__bg" aria-hidden="true"></i>
    <i class="btn__bg" aria-hidden="true"></i>
    <i class="btn__bg" aria-hidden="true"></i>
    <i class="btn__bg" aria-hidden="true"></i>
</button>
```

Рис. 2.23. Фрагмент коду елементу «button»

Елемент `button` представляє кнопку, яку можна натиснути у формі. Він має такі атрибути:

- Для атрибута `type` встановлено значення «button», що вказує на те, що це загальна кнопка, а не кнопка надсилання форми.
- Атрибут `class` має значення «btn btn—blue» для цілей стилю.

Вміст кнопки складається з елемента `span` з текстом «Відправити» (Надіслати) та чотирьох елементів `i` з класом «btn__bg», який, представляє фонову графіку.

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

Рис. 2.24. Фрагмент коду елементу «div» з класом «status»

Елемент `div` з класом «status» є порожнім контейнером. Він використовується для відображення результату надсилання форми, можливо, через AJAX.

```
<div class="map">
    <iframe
        src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d2538.5884799447003!2d30.63282097650469!3d50.
        486004371599016!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.
        1!3m3!1m2!1s0x40d4d06ef902e7af%3A0x8037d0eb2add20b4!2zMjRBLCDQstGD0LvQuNGG0Y8g0JrRg9Cx0LDQvdGB0YzQutC-0Zcg0KPQu
        tGA0LDRl9C90LgsIDI00JAsINCa0LjRl9CyLCDQo9C60YDQsNGX0L3QsCwgMDIwMDA!5e0!3m2!1suk!2spt!4v1683041140906!5m2!1suk!2
        spt"
        width="500" height="300" style="border:0;" allowfullscreen="" loading="lazy"
        referrerpolicy="no-referrer-when-downgrade"></iframe>
</div>
```

Рис. 2.25. Фрагмент коду елементу «iframe»

Елемент `div` з класом «map» представляє контейнер для вбудовування розташування Google Maps. Елемент `iframe` використовується для вбудовування карти, вказуючи URL-адресу джерела, розміри та різні атрибути, пов’язані з її поведінкою та безпекою.

```
<div class="block">
    <h5><i class="fa-solid fa-phone"></i>Зателефонуйте нам</h5>
    <p><a href="tel:+380660108528">+38 066 010 85 28</a></p>
</div>
```

Рис. 2.26. Фрагмент коду елементу «div» з класом «block»

Елемент `div` із класом «block» представляє блок у розділі «blocks». Це містить:

- Елемент `h5` для заголовка «Зателефонуйте нам», який містить елемент `i` з класами Font Awesome, що представляє значок телефону.
- Елемент `p`, що містить гіперпосилання (елемент `a`) із номером телефону «+38 066 010 85 28». Атрибут `href` вказує, що це телефонне посилання.

Так само є ще два блоки з різними заголовками, іконками та вмістом.

Ці HTML-елементи та їхня структура сприяють організації, представленню та функціональності вмісту web програми, дозволяючи користувачам взаємодіяти з формою, переглядати вбудовану карту та отримувати доступ до контактної інформації семантичним і доступним способом.

І нарешті структура сторінки «Порівняння».

Структура HTML складається з section.cd-products-comparison-table, що містить <header> і div.cd-products-table. <header> містить кнопки дій (фільтр і скидання), тоді як div.cd-products-table використовується для обгортання div.features (список функцій продукту) і div.cd-products-wrapper. Останній містить невпорядкований список (ul.cd-products-columns) для елементів списку продуктів.

```
<section class="cd-products-comparison-table">
    <header>
        <h2>Тут ви знайдете порівняльну характеристику країн з різних аспектів,<br> к
        </h2>
        <div class="actions">
            <a href="#0" class="reset">Скинути</a>
            <a href="#0" class="filter">Відфільтрувати</a>
        </div>
    </header>

    <div class="cd-products-table">
        <div class="features">
            <div class="top-info">Країна</div>
            <ul class="cd-features-list">···
            </ul>
        </div> <!-- .features -->

        <div class="cd-products-wrapper">
            <ul class="cd-products-columns">
                <li class="product">
                    <div class="top-info">
                        <div class="check"></div>
                        <img src="img/Flag_of_Austria.svg.png" alt="Flag of Austria">
                        <h3>Австрія</h3>
                    </div> <!-- .top-info -->

                    <ul class="cd-features-list">···
                    </ul>
                </li> <!-- .product -->
```

Рис. 2.27. Фрагмент коду елементу «section»

2.6. Використання функцій SCSS

Загалом, SCSS як препроцесор CSS розширює можливості CSS і забезпечує більш ефективний і зручний спосіб написання коду CSS. Він покращує організацію коду, можливість повторного використання та читабельність, що робить його популярним вибором для розробників, які працюють над складними чи великомасштабними web додатками.

Продемонструємо використання таких функцій SCSS, як змінні, міксини, вкладення та частини в процесі розробки інтерфейсу користувача.

У цьому фрагменті коду ми визначили кілька змінних за допомогою синтаксису SCSS. Ці змінні використовуються для зберігання значень кольорів, шрифтів і розмірів, які можна повторно використовувати в таблиці стилів.

```scss
// colors

$color-1: #404042; // Ship Gray
$color-2: #9dc997; // Spring Rain
$color-3: #ffffff; // White
$border: shade($color-3, 10%);

// fonts

$primary-font: "Source Sans Pro", sans-serif;

// Table

$products-number: 12;
$products-column-width-mobile: 150px;
$products-column-width: 310px;
$features-column-width-mobile: 120px;
$features-column-width: 210px;
```

Рис. 2.28. Фрагмент коду змінних у SCSS

Ось розбивка того, що відбувається в коді.

Кольори:

- `$color-1` присвоєно значення `#404042`, яке представляє колір «Сірий корабель»;

- `$color-2` присвоєно значення `#9dc997`, яке представляє колір «Весняний дощ» ;
- `$color-3` присвоєно значення `#ffffff`, яке представляє колір «Білий» ;
- `$border` присвоюється результат функції `shade()`, яка приймає значення кольору `$color-3` і затемнює його на 10%.

Шрифти:

- `$primary-font` присвоєно значення `»Source Sans Pro», sans-serif`. Він представляє кращий стек шрифтів для основного шрифту, де «Source Sans Pro» є першим вибором, а потім загальні шрифти без зарубок.

Розміри таблиці:

- `$products-number` присвоєно значення `12`. Він представляє кількість продуктів;
- `$products-column-width-mobile` присвоюється значення `150px`, яке представляє ширину стовпців таблиці для мобільних пристроїв;
- `$products-column-width` присвоюється значення `310px`, яке представляє ширину стовпців таблиці для великих екранів;
- `$features-column-width-mobile` присвоюється значення `120px`, яке представляє ширину стовпців функцій для мобільних пристроїв;
- `$features-column-width` присвоюється значення `210px`, яке представляє ширину стовпців функцій для великих екранів.

```
@import "../partials/variables"; // colors, fonts etc...
```

Рис. 2.29. Фрагмент коду імпорту файлу

Код `@import «../partials/variables»;` у SCSS — це оператор імпорту, який включає вміст іншого файлу в поточну таблицю стилів. У цьому випадку він імпортує файл під назвою «variables» з каталогу «partials».

Метою цього коду є централізація та систематизація загальних значень, таких як кольори, шрифти та інші змінні, які використовуються в таблиці стилів. Розмі-

щуючи ці змінні в окремому файлі, це сприяє багаторазовому використанню коду, послідовності та полегшенню обслуговування.

Коли зустрічається оператор імпорту, процесор SCSS отримує вміст зазначеного файлу («../partials/variables») і включає його замість оператора імпорту. Процесор по суті поєднує вміст обох файлів в одну таблицю стилів, ніби вміст імпортованого файлу було безпосередньо записане в поточний файл.

І він імпортується за допомогою `@import «../partials/variables»;`, змінні, визначені в цьому файлі, будуть доступні в поточній таблиці стилів. Це дозволяє використовувати ці змінні у всьому коді CSS, забезпечуючи центральне місце для їх зміни, якщо це необхідно.

Використання цього підходу може значно спростити процес оновлення стилів для кількох елементів або сторінок. Замість того, щоб шукати та змінювати окремі входження значень (таких як кольори чи шрифти), ви можете вносити зміни у файл «змінних», і ці зміни автоматично поширюватимуться на всю таблицю стилів.

Наступний фрагмент коду демонструє використання SCSS для визначення контрольних точок і створення медіа-запитів, а також реалізацію спрощеної сіткової системи за допомогою міксинів.

Код починається з оголошення трьох змінних: $S, $M і $L, які представляють контрольні точки для малого, середнього та великого екранів відповідно. Ці контрольні точки визначають мінімальну ширину, до якої будуть застосовані певні стилі.

Далі код визначає міксин під назвою «MQ» (скорочення від media query), який приймає параметр під назвою $canvas. Цей міксин використовується для створення медіа-запитів на основі наданого значення $canvas. Усередині mixin умовні оператори використовуються для перевірки значення $canvas і створення відповідного медіа-запиту за допомогою правила @media. Директива @content використовується для включення стилів у блок медіа-запиту.

Після міксину медіа-запиту визначається ще один міксин під назвою «column». Цей міксин використовується для створення адаптивних стовпців таблиці. Він приймає два параметри: $percentage і $float-direction. Параметр $percentage визначає ширину стовпця у відсотках, а параметр $float-direction визначає напрямок

стовпця з плаваючою точкою (за замовчуванням — «left», якщо не вказано). Всередині mixin властивість width встановлюється на обчислену ширину на основі наданого відсотка, а властивість float встановлюється на основі параметра $float-direction.

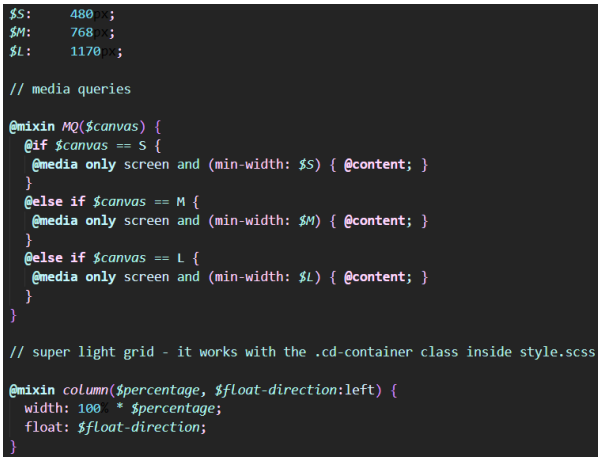

Рис. 2.30. Фрагмент коду змінних та міксину

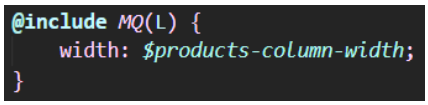

Рис. 2.31. Фрагмент коду використання директиви «include»

Директива @include використовується для включення стилів у певний медіа-запит. У цьому випадку він націлений на точку зупинки L (великі екрани). У блоці медіа-запитів для властивості ширини встановлено значення $products-column-width. Передбачається, що змінна $products-column-width визначена в іншому місці коду.

Загалом цей код демонструє потужність і гнучкість SCSS у визначенні точок зупину та створенні медіа-запитів на основі цих точок зупину. Він також демонструє використання міксинів для створення багаторазово використовуваних і конфігурованих стилів, таких як міксин стовпців таблиці. Ці концепції важливі для створення адаптивних і підтримуваних інтерфейсів користувача у web розробці.

Наступний фрагмент коду представляє стилі CSS для кнопки з ідентифікатором «myBtn». При застосуванні до елемента HTML ці стилі змінять вигляд і поведінку цього елемента.

```css
#myBtn {
    width: 50px;
    height: 50px;
    background-color: #306fbe;
    position: fixed;
    bottom: 50px;
    right: 50px;
    border-radius: 100%;
    font-family: "Playfair Display", serif;
    z-index: 1000;
    border-color: #0057b8 #0057b8;
    padding: 13px 20px;
    text-decoration: none;
    color: white;
    &::after {
        position: absolute;
        top: 50%;
        left: 50%;
        content: "";
        width: 16px;
        height: 16px;
        margin: -3px 0 0 -8px;
        border-left: 2px solid #fff;
        border-bottom: 2px solid #fff;
        -webkit-transform: rotate(135deg);
        transform: rotate(135deg);
        box-sizing: border-box;
    }
}
```

Рис. 2.32. Фрагмент коду кнопки з ідентифікатором «myBtn»

Ось розбивка того, що робить кожна властивість:

- `width: 50px;` і `height: 50px;` - встановлює ширину та висоту кнопки на 50 пікселів, створюючи квадратну форму.

- `background-color: #306fbe;` - визначає колір фону кнопки як відтінок синього (#306fbe).

- `position: fixed;` - розміщує кнопку відносно вікна перегляду, дозволяючи їй залишатися на місці навіть під час прокручування сторінки.

- `bottom: 50px;` і `right: 50px;` - розміщує кнопку на 50 пікселів від нижнього та правого країв вікна перегляду, створюючи фіксоване положення в нижньому правому куті.

- `border-radius: 100%;` - застосовує круговий радіус межі до кнопки, завдяки чому вона виглядає як коло через однакову ширину та висоту.

- `font-family: «Playfair Display», serif;` - визначає сімейство шрифтів для тексту всередині кнопки. У цьому випадку він налаштований на шрифт «Playfair Display», повертаючись до загального шрифту із засічками, якщо він недоступний.

- `z-index: 1000;` - встановлює порядок розміщення кнопки, визначаючи її позицію у вертикальному порядку розміщення елементів на сторінці. Більше значення, наприклад 1000, гарантує, що кнопка з'явиться поверх інших елементів.

- `border-color: #0057b8 #0057b8;` - встановлює колір рамки кнопки на відтінок синього (#0057b8) на верхньому та нижньому краях, створюючи двоколірну рамку.

- `padding: 13px 20px;` - додає 13 пікселів відступу вгорі та внизу та 20 пікселів відступу ліворуч і праворуч, створюючи деякий простір між вмістом кнопки та її межею.

- `text-decoration: none;` - видаляє будь-яке оформлення тексту, наприклад підкреслення, з тексту кнопки.

- `color: white;` - встановлює білий колір тексту кнопки, щоб він виділявся на тлі.

- `&::after { … }` - цей селектор і наступні властивості визначають стилі для псевдоелемента (`::after`), який додається після вмісту кнопки. Псевдоелемент використовується для створення форми стрілки. Властивості всередині цього блоку розміщують і стилізують стрілку, повертаючи її на 135 градусів, щоб вказувати по діагоналі вгору ліворуч.

Рис. 2.33. Кнопка «нагору»

Використовуючи ці стилі CSS, отримана кнопка матиме фіксоване положення в нижньому правому куті вікна перегляду, круглу форму з двокольоровою рамкою, колір фону та білу стрілку, спрямовану по діагоналі вгору ліворуч. Кнопка також матиме певний відступ, сімейство шрифтів і колір тексту.

У наданому коді вкладення використовується в синтаксисі SCSS. Це дозволяє вам вкладати селектори CSS один в одного, створюючи ієрархічну структуру, яка відображає структуру розмітки HTML.

У наведеному коді вкладення використовується в такій частині.

Рис. 2.34. Фрагмент коду використання &

Тут символ «&» є заповнювачем, який посилається на батьківський селектор, який у цьому випадку є «#myBtn». Використовуючи символ `&` у вкладеному селекторі, він посилається на батьківський селектор і поєднує його з вкладеним селектором, створюючи більш конкретний селектор.

Селектор `&::after` вкладено в селектор `#myBtn`. Це дозволяє стилям, указаним у вкладеному блоці, застосовуватися лише до псевдоелемента `::after`, який є дочірнім елементом `#myBtn`. Вкладені властивості всередині блоку `&::after` застосовуються до псевдоелемента, встановлюючи його позицію, розміри, стилі рамок і обертання.

Використовуючи вкладення, стає легше візуалізувати та впорядкувати стилі, які застосовуються до конкретних елементів та їхніх нащадків. Це може покращити читабельність коду та обслуговування, особливо у великих проектах, де можуть існувати складні вимоги до стилю для різних елементів.

У наступному фрагменті код представляє стиль кнопки за допомогою SCSS (Sass). Давайте розберемо код і опишемо його функціональність:

1. Кнопка визначається за допомогою класу `.btn` і додаткового класу-модифікатора (наприклад, `.btn—blue`) для визначення різних варіантів кольору для кнопки.

2. Спеціальна властивість CSS `--hue` використовується для визначення значення відтінку для кольору фону кнопки. Різні класи модифікаторів (наприклад, `.btn—blue`) встановлюють спеціальні значення для `--hue`, що призводить до різних кольорів кнопок.

3. Псевдокласи `:active` і `:focus` використовуються для визначення стилів, коли кнопка натискається або фокусується. У цьому випадку контур застосовується з дещо іншим значенням відтінку для створення інтерактивного ефекту.

4. Кнопки, які з'являються після іншої кнопки (`.btn + .btn`), мають верхнє поле для створення вертикального інтервалу між ними.

5. Текст кнопки стилізовано за допомогою класу `.btn__txt`, який встановлює шрифт, розмір і взаємне розташування.

6. Фон кнопки стилізовано за допомогою класу `.btn__bg`. Він використовує спеціальну властивість CSS `--hueBg` для визначення значення відтінку для кольору фону. Цикл `for` використовується для створення та стилізації чотирьох фонових шарів, кожен з яких має різне значення відтінку на основі значення `--hue` кнопки.

```scss
// 90 = 360/no of layers (4 in my case)
$hueStep: 90;
$delayStep: 0.115s;
.btn {
    background: hsl(var(--hue), 98%, 80%);
    border: none;
    border-radius: 7px;
    cursor: pointer;
    color: black;
    text-decoration: none;
    letter-spacing: 0.05em;
    overflow: hidden;
    padding: 1.15em 3.5em;
    min-height: 3.3em;
    position: relative;

    &--blue {
        --hue: 210;
    }

    &:active,
    &:focus {
        outline: 3px solid hsl(calc(var(--hue) + #{$hueStep}), 98%, 80%);
    }

    & + & {
        margin-top: 2.5em;
    }

    &__txt {
        position: relative;
        z-index: 2;
        font-family: "Oswald", sans-serif;
        font-size: 20px;
    }

    &__bg {
        background: hsl(var(--hueBg), 98%, 80%);
        border-radius: 50%;
        display: block;
        height: 0;
        left: 50%;
        margin: -50% 0 0 -50%;
        padding-top: 100%;
        position: absolute;
        top: 50%;
        width: 100%;
        transform: scale(0);
        transform-origin: 50% 50%;
        transition: transform 0.175s cubic-bezier(0.5, 1, 0.89, 1);
        z-index: 1;

        @for $i from 1 through 4 {
            &:nth-of-type(#{$i}) {
                --hueBg: calc(var(--hue) - #{$i * $hueStep});
                transition-delay: $delayStep / 2 * (4 - $i);
            }
        }

        .btn:hover &,
        .btn:focus &,
        .btn:active & {
            transform: scale(1.5);
            transition: transform 0.35s cubic-bezier(0.11, 0, 0.5, 0);
            @for $i from 1 through 4 {
                &:nth-of-type(#{$i}) {
                    transition-delay: $delayStep * $i;
                }
            }
        }
    }
}
```

Рис. 2.35. Фрагмент коду стилів кнопки «Відправити»

7. Під час наведення курсора, фокусування або клацання фонові шари збільшуються, створюючи ефект розширення. Властивість `transition-delay` використовується для розподілу анімації кожного шару, створюючи ефект послідовної трансформації. Функція `cubic-bezier` використовується для визначення функції синхронізації для переходів.

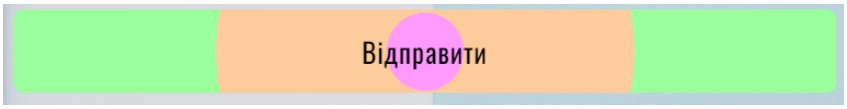

Рис. 2.36. Вигляд анімації кнопки «Відправити» на сторінці «Контакти»

Рис. 2.37. Вигляд анімації кнопки «Відправити» на сторінці «Контакти»

Загалом, цей код використовує такі функції SCSS, як змінні, вкладення та цикли, для генерації CSS-коду з динамічними та багаторазово використовуваними стилями для кнопки та її фонових шарів. Результатом є візуально приваблива та інтерактивна кнопка з настроюваними колірними варіаціями та ефектами анімації.

У наступному фрагменті коду ми маємо CSS-анімацію, визначену за допомогою правила @keyframes. Анімація називається «fadeEffect». Він визначає перехід між двома ключовими кадрами: `від` і `до`.

```
@keyframes fadeEffect {
    from {
        opacity: 0;
    }
    to {
        opacity: 1;
    }
}
```

Рис. 2.38. Фрагмент коду використання опису анімації

Ключовий кадр «from» представляє початковий стан анімації та встановлює властивість непрозорості на 0. Це означає, що елемент, до якого застосована ця анімація, буде повністю прозорим або невидимим.

Ключовий кадр «to» представляє кінцевий стан анімації та встановлює властивість opacity на 1. Це означає, що елемент перейде в повністю непрозорий або видимий стан.

Коли ця анімація застосовується до елемента за допомогою CSS, вона поступово змінює властивість непрозорості з 0 (початковий стан) до 1 (кінцевий стан) протягом певного часу, створюючи ефект поступового згасання.

Цей код використовується для застосування ефекту поступового згасання до елементів web сторінки. Потім анімація застосовується до елемента за допомогою властивості animation. У цьому випадку анімація має назву «fadeEffect», а тривалість анімації становить 600 мілісекунд (або 0,6 секунди).

```
animation: fadeEffect 600ms; /* Fading effect takes 1 second */
```

Рис. 2.39. Фрагмент коду використання ефекту

Наступний фрагмент коду використовує бібліотеку Bourbon, яка є набором міксинів і функцій Sass, які спрощують розробку CSS.

```
    @include clearfix;
    @include transition(background-color 0.3s);
    @include transform(scale(0));
    @include transition(opacity 0.3s, visibility 0.3s, transform 0.3s);
    @include backface-visibility(hidden);
    @include linear-gradient(to bottom, rgba(#000, 0.06), rgba(#000, 0),
$fallback: rgba(#000, 0));
```

Рис. 2.40. Фрагмент коду використання бібліотеки Bourbon

Давайте розберемо кожен рядок коду та пояснимо, що відбувається:

1. `@include clearfix;` - цей міксин використовується для очищення float елемента, гарантуючи, що його контейнер правильно обертається навколо нього. Він

додає необхідні властивості CSS, щоб очистити будь-які плаваючі елементи всередині цільового елемента.

2. `@include conversion(background-color 0,3s);` - цей міксин використовується для застосування ефекту переходу CSS до властивості `background-color` елемента. У цьому випадку тривалість переходу становить 0,3 секунди, тобто будь-які зміни властивості `background-color` будуть анімовані протягом цього часу.

3. `@include transform(scale(0));` - цей міксин використовується для застосування перетворення CSS до елемента, зокрема зменшуючи його до розміру 0. У цьому випадку функція `scale` використовується з значення 0, що фактично робить елемент невидимим і не займає місця.

4. `@include conversion(opacity 0.3s, visibility 0.3s, transform 0.3s);` - цей mixin використовується для застосування переходів CSS до кількох властивостей одночасно. Він визначає три властивості: `opacity`, `visibility` і `transform`. Властивості `opacity` і `visibility` змінюватимуться протягом 0,3 секунди, тоді як властивість `transform` також переходитиме протягом тієї ж тривалості. Це означає, що зміни цих властивостей будуть анімовані плавно протягом 0,3 секунди.

5. `@include backface-visibility(hidden);` - цей міксин використовується для застосування властивості `backface-visibility` до елемента, встановлюючи для нього значення `hidden`. Ця властивість визначає, чи буде видима задня грань 3D-трансформованого елемента, якщо дивитися від глядача. Встановлення для нього значення `hidden` гарантує, що задня грань не буде видно.

6. `@include linear-gradient(донизу, rgba(#000, 0,06), rgba(#000, 0), $fallback: rgba(#000, 0));` - цей міксин використовується для створення лінійного градієнтний фон для елемента. Він використовує CSS-функцію linear-gradient для визначення градієнта. У цьому випадку градієнт починається зверху і йде вниз («донизу»), і він переходить від значення кольору RGBA зі значенням альфа 0,06 до значення кольору RGBA зі значенням альфа 0. ` Параметр $fallback` дозволяє вказати запасний колір, якщо браузер не підтримує градієнти CSS.

Використовуючи ці міксини Bourbon, ми легко застосовували загальні стилі та ефекти CSS до своїх елементів. Міксин `clearfix` допомагає з компонуванням і пози-

ціонуванням, міксин `transition` додає плавні переходи до вказаних властивостей, а міксин `transform` дозволяє такі перетворення, як масштабування, обертання або переміщення елементів. Міксин `backface-visibility` допомагає контролювати видимість тильної сторони трансформованих елементів, тоді як міксин `linear-gradient` дозволяє створювати градієнтні фони з настроюваними кольорами та напрямками. Ці міксини забезпечують зручний і зручний спосіб написання CSS-коду, що полегшує створення модульних таблиць стилів для багаторазового використання.

2.7. Впровадження методології BEM

Продемонструємо застосування принципів БЕМ до структури та стилю компонентів UI у web додатку «Європа для українців».

У наступному фрагменті коду ми застосуємо принципи BEM (Block, Element, Modifier) до структури та стилю компонентів UI у web додатку «Європа для українців». BEM — це популярна методологія іменування, яка допомагає створювати модульний код CSS, який можна багаторазово використовувати та підтримувати. Давайте розберемо код і продемонструємо, як застосовуються принципи БЕМ.

```
<div class="container">
    <div class="container__header">
        <header class="header">
```

Рис. 2.41. Фрагмент коду використання технології BEM

Тут у нас є блок «container», який служить основним елементом контейнера. Він представлений класом CSS `.container`.

Усередині контейнера ми маємо елемент «container__header», який є підкомпонентом блоку контейнера. Він представлений класом CSS `.container__header`. Подвійне підкреслення (`__`) означає, що це елемент блоку «контейнер».

У заголовку контейнера ми маємо компонент «заголовок», який є іншим підкомпонентом. Він представлений класом CSS `.header`. Цей компонент можна використовувати незалежно в інших частинах програми.

```
<li class="header__item"><a href="index.html" class="active">Головна</a></li>
```

Рис. 2.42. Фрагмент коду використання технології BEM

У компоненті заголовка ми маємо елемент списку з класом `.header__item`. Це елемент компонента заголовка, на що вказує подвійне підкреслення.

```
<div class="container__main">
    <h1 class="container__mainTitle">Європа для українців</h1>
    <div class="container__mainDescription">
```

Рис. 2.43. Фрагмент коду використання технології BEM

Далі у нас є елемент «container__main», який є підкомпонентом блоку контейнера. Він представлений класом CSS `.container__main`.

Усередині основного контейнера ми маємо елемент «container__mainName», який є елементом блоку «container__main». Він представлений класом CSS `.container__mainName`.

Далі ми маємо елемент «container__mainTabs», який є ще одним елементом блоку «container__main». Він може мати наприклад додатковий клас-модифікатор «container__mainTabs—active», щоб вказати певний стан або варіацію елемента. Модифікатори надають спосіб змінити стиль або поведінку елемента або блоку. У цьому випадку клас модифікатора `.container__mainTabs—active` представляє активний стан.

Дотримуючись угоди про іменування BEM, ми створюємо чітку та структуровану ієрархію для наших компонентів інтерфейсу користувача, що полегшить розуміння та підтримку коду. Підхід BEM сприяє багаторазовому використанню та міні-

мізує вплив змін стилю на інші частини програми, підвищуючи масштабованість і зручність обслуговування кодової бази.

2.8. Функціональність JavaScript

Давайте розглянемо приклади фрагментів коду JavaScript для реалізації певних функцій або поведінки у web програмі.

Даний код представляє функцію JavaScript під назвою `openTab`, яка використовується для реалізації функцій навігації з вкладками.

Давайте розберемо код і опишемо, що робить кожна частина:

1. Функція openTab приймає два параметри: evt (об'єкт події) і tabName (ім'я вкладки, яку потрібно відкрити).

2. Він оголошує три змінні: `i`, `tabcontent` і `tablinks`. Ці змінні будуть використані пізніше у функції.

3. Потім він отримує всі елементи з назвою класу «tabcontent» за допомогою методу `getElementsByClassName` і призначає їх змінній `tabcontent`.

```
function openTab(evt, tabName) {
  var i, tabcontent, tablinks;

  tabcontent = document.getElementsByClassName("tabcontent");
  for (i = 0; i < tabcontent.length; i++) {
    tabcontent[i].style.display = "none";
  }

  tablinks = document.getElementsByClassName("tablinks");
  for (i = 0; i < tablinks.length; i++) {
    tablinks[i].className = tablinks[i].className.replace(" active", "");
  }

  document.getElementById(tabName).style.display = "block";
  evt.currentTarget.className += " active";
}
document.getElementById("defaultOpen").click();
```

Рис. 2.44. Фрагмент коду функції «openTab»

4. Він проходить по кожному елементу колекції `tabcontent` за допомогою циклу for і встановлює для властивості стилю `display` кожного елемента значення «none». Це фактично приховує всі елементи вмісту вкладки.

5. Він отримує всі елементи з назвою класу «tablinks» за допомогою методу `getElementsByClassName` і призначає їх змінній `tablinks`.

6. Він проходить по кожному елементу колекції `tablinks` за допомогою циклу for і видаляє назву класу «active» із властивості `className` кожного елемента. Це видаляє «активний» клас з усіх посилань вкладок, скасовуючи їх вибір.

7. Показує поточну вкладку, встановлюючи властивість стилю `display` елемента з ідентифікатором, який відповідає параметру `tabName`, на значення «block». Це робить вміст відповідної вкладки видимим.

8. Він додає назву класу «active» до властивості `className` кнопки, яка викликала подію відкриття вкладки (`evt.currentTarget`). Це підсвічує поточне активне посилання на вкладку.

9. Нарешті, поза функцією він вибирає елемент з ідентифікатором «defaultOpen» за допомогою `document.getElementById` і запускає подію клацання на ньому за допомогою методу `click()`. Це імітує натискання вкладки за замовчуванням під час завантаження сторінки.

Загалом, цей код реалізує функцію навігації з вкладками, приховуючи та показуючи відповідний вміст вкладки на основі взаємодії користувача. Це гарантує, що одночасно буде видно лише одну вкладку, і підсвічує посилання на активну вкладку. Останній рядок коду встановлює вкладку за замовчуванням, яка відкривається під час завантаження сторінки.

Наданий фрагмент коду демонструє використання бібліотеки jQuery для обробки надсилання форми та виконання запитів AJAX.

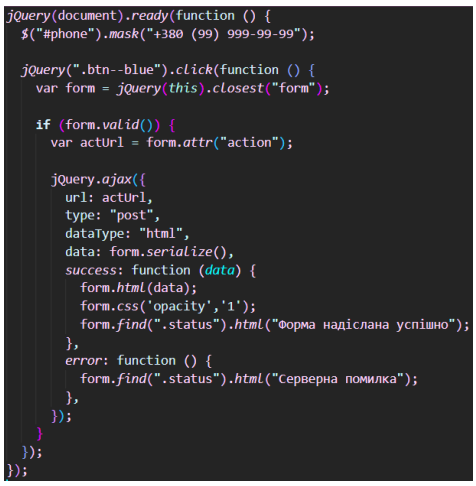

Рис. 2.45. Фрагмент коду функції використання Jquery

Давайте розберемо код і опишемо, що він робить:

1. `jQuery(document).ready(function () { … });` - цей код гарантує, що вкладений JavaScript буде виконано після завершення завантаження DOM (об’єктної моделі документа). Він забезпечує функцію зворотного виклику, яка виконується, коли документ готовий.

2. `$(«#phone»).mask(«+380 (99) 999-99-99»);` - цей рядок застосовує маску до елемента з ідентифікатором `phone` за допомогою плагіна маски jQuery. Зазначений тут формат маски дозволяє вводити номери у форматі `+380 (99) 999-99-99`, забезпечуючи візуальне керівництво для введення телефонних номерів.

3. `jQuery(«.btn—blue»).click(function () { … });` - цей код приєднує обробник події клацання до елементів із класом `btn—blue`. Коли клацнути будь-який елемент із цим класом, буде виконана вкладена функція.

4. `var form = jQuery(this).closest(«form»);` - цей рядок знаходить найближчий елемент `<form>` відносно натиснутої кнопки та призначає його змінній `form`. Це дозволяє наступному коду працювати з конкретною формою, пов’язаною з натиснутою кнопкою.

5. `if (form.valid()) { … }` - ця умова перевіряє, чи дійсна форма. Припускається, що існує певна логіка перевірки форми, а метод `valid()` використовується для перевірки дійсності вхідних даних форми. Якщо форма дійсна, виконується код в умові.

6. `var actUrl = form.attr(«action»);` - цей рядок отримує значення атрибута `action` з елемента форми. Припускається, що форма має визначений атрибут `action`, що вказує URL-адресу, куди мають бути надіслані дані форми.

7. `jQuery.ajax({ … });` - цей блок коду виконує запит AJAX до вказаної URL-адреси (`actUrl`) за допомогою методу `ajax` jQuery. Він надсилає дані форми, серіалізовані за допомогою `form.serialize()` як корисне навантаження запиту.

8. Усередині методу `ajax` функції `success` і `error` визначені як зворотні виклики для обробки відповідних результатів запиту AJAX:

- Якщо запит виконано успішно («успішно»), отримані «дані» обробляються. У наданому коді функція успіху встановлює повідомлення «Форма надіслана успішно» (Форма надіслана успішно) всередині елемента з класом `.status`.

- Якщо під час запиту AJAX виникає помилка (`error`), функція помилки встановлює повідомлення «Серверна помилка» (Помилка сервера) всередині елемента з класом `.status`.

Загалом, фрагмент коду демонструє, як jQuery можна використовувати для покращення функціональності форми, наприклад застосування масок до полів введення, перевірки форм і асинхронного надсилання даних форми за допомогою AJAX.

Давайте розберемо код сторінки Порівняння і опишемо, що відбувається:

1. Код вкладено в блок `jQuery(document).ready(function ($) {…});`, який гарантує, що код буде виконано, коли документ повністю завантажено та готовий до обробки за допомогою jQuery.

```
jQuery(document).ready(function ($) {
  function productsTable(element) {
    this.element = element;
    this.table = this.element.children(".cd-products-table");
    this.tableHeight = this.table.height();
    this.productsWrapper = this.table.children(".cd-products-wrapper");
    this.tableColumns = this.productsWrapper.children(".cd-products-columns");
```

Рис. 2.46. Фрагмент коду використання Jquery для таблиці Порівняння

2. Код визначає функцію конструктора під назвою `productsTable`, яка ініціалізує та керує таблицею продуктів. Він приймає параметр `element` і призначає йому різні властивості, наприклад `table`, `tableHeight`, `productsWrapper`, `tableColumns` тощо. Ці властивості представляють різні елементи в структурі таблиці продуктів.

3. Функція `bindEvents` визначена як метод-прототип конструктора `productsTable`. Він приєднує обробники подій до різних елементів у таблиці, таких як подія прокручування на `productsWrapper`, події клацання на `products`, `filterBtn` і `resetBtn`, а також події клацання на навігаційних посиланнях.

```
productsTable.prototype.bindEvents = function () {
  var self = this;
  //detect scroll left inside product table
  self.productsWrapper.on("scroll", function () {···
  });
  //select single product to filter
  self.products.on("click", ".top-info", function () {···
  });
  //filter products
  self.filterBtn.on("click", function (event) {···
  });
```

Рис. 2.47. Фрагмент коду використання Jquery для таблиці Порівняння

4. Функція upadteFilterBtn є прототипом методу, який оновлює стан кнопки фільтра на основі кількості вибраних продуктів. Він додає або видаляє клас CSS «активний» на кнопці фільтра залежно від того, чи вибрано принаймні два продукти.

```
productsTable.prototype.upadteFilterBtn = function () {
  //show/hide filter btn
  if (this.selectedproductsNumber >= 2) {
    this.filterActive = true;
    this.filterBtn.addClass("active");
  } else {
    this.filterActive = false;
    this.filterBtn.removeClass("active");
  }
};
```

Рис. 2.48. Фрагмент коду використання Jquery для таблиці Порівняння

5. Функція `updateLeftScrolling` є прототипом методу, який обробляє поведінку прокручування в таблиці продуктів. Він виявляє подію прокручування в `productsWrapper` і оновлює позицію прокручування, додає або видаляє клас CSS "прокручування" в елементі `table` і оновлює видимість кнопок навігації на основі позиції прокручування.

```
productsTable.prototype.updateLeftScrolling = function () {
  var totalTableWidth = parseInt(this.tableColumns.eq(0).outerWidth(true)),
    tableViewport = parseInt(this.element.width()),
    scrollLeft = this.productsWrapper.scrollLeft();

  scrollLeft > 0
    ? this.table.addClass("scrolling")
    : this.table.removeClass("scrolling");

  if (this.table.hasClass("top-fixed") && checkMQ() == "desktop") {
    setTranformX(this.productsTopInfo, "-" + scrollLeft);
    setTranformX(this.featuresTopInfo, "0");
  }

  this.leftScrolling = false;

  this.updateNavigationVisibility(scrollLeft);
};
```

Рис. 2.49. Фрагмент коду використання JQuery для таблиці Порівняння

6. Функція `updateNavigationVisibility` є прототипом методу, який оновлює видимість кнопок навігації на основі позиції прокручування. Він додає або видаляє клас CSS «неактивний» на попередній і наступній кнопках навігації залежно від положення прокручування.

```
productsTable.prototype.updateNavigationVisibility = function (scrollLeft) {
  scrollLeft > 0
    ? this.navigation.find(".prev").removeClass("inactive")
    : this.navigation.find(".prev").addClass("inactive");
  scrollLeft <
    this.tableColumns.outerWidth(true) - this.productsWrapper.width() &&
  this.tableColumns.outerWidth(true) > this.productsWrapper.width()
    ? this.navigation.find(".next").removeClass("inactive")
    : this.navigation.find(".next").addClass("inactive");
};
```

Рис. 2.50. Фрагмент коду використання JQuery для таблиці Порівняння

7. Код містить кілька інших методів-прототипів, таких як `updateTopScrolling`, `updateProperties`, `showSelection`, `resetSelection`, `filterProducts`, `resetProductsVisibility` і `updateSlider`. Ці методи обробляють різні

аспекти функціональності таблиці продуктів, такі як оновлення поведінки верхнього прокручування, оновлення властивостей таблиці, фільтрація та скидання вибору продуктів і оновлення поведінки повзунка.

8. Код також ініціалізує екземпляри конструктора `productsTable` для кожного елемента з класом "cd-products-comparison-table", знайденим у документі. Ці екземпляри зберігаються в масиві `comparisonTables`.

```
var comparisonTables = [];
$(".cd-products-comparison-table").each(function () {
  //create a productsTable object for each .cd-products-comparison-table
  comparisonTables.push(new productsTable($(this)));
});
```

Рис. 2.51. Фрагмент коду використання JQuery для таблиці Порівняння

9. Слухачі подій налаштовані для подій прокручування вікна та зміни розміру, щоб запускати відповідні функції (`checkScrolling` і `checkResize`), які обробляють відповідну поведінку для таблиць продуктів.

```
function checkScrolling() {
  var scrollTop = $(window).scrollTop();
  comparisonTables.forEach(function (element) {
    element.updateTopScrolling(scrollTop);
  });

  windowScrolling = false;
}

function checkResize() {
  comparisonTables.forEach(function (element) {
    element.updateProperties();
  });

  windowResize = false;
}
```

Рис. 2.52. Фрагмент коду використання JQuery для таблиці Порівняння

10. Визначено допоміжні функції, такі як `checkMQ` і `setTranformX`, які використовуються в методах `productsTable` для перевірки статусу медіа-запиту та застосування перетворень CSS відповідно.

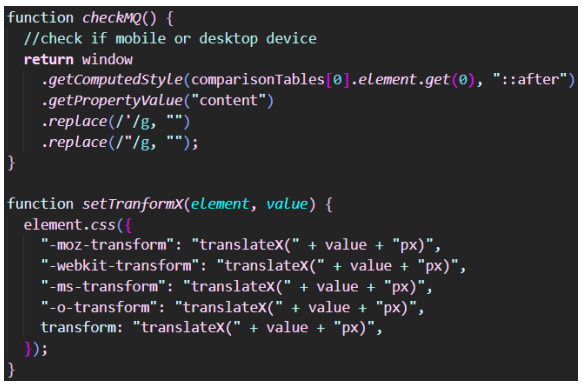

Рис. 2.53. Фрагмент коду використання JQuery для таблиці Порівняння

2.9. Інтеграція PHP

Продемонструємо інтеграцію PHP у розробку інтерфейсу користувача для отримання та відображення відповідної інформації з сервера.

В нашому проекту PHP використовується для обробки надсилання форми та надсилання електронного листа з даними форми.

Розглянемо код крок за кроком:

1. Код починається з тегів відкриття та закриття PHP (`<?php` і `?>`), щоб вказати, що він містить код PHP.

2. Змінним `$name`, `$email`, `$phone` і `$message` присвоюються значення, надіслані через метод HTTP POST із форми. Значення отримують із суперглобального масиву `$_POST` за допомогою відповідних назв полів введення (`name`, `email`, `phone` і `message`).

3. Адреса електронної пошти, на яку будуть надіслані дані форми, зберігається в змінній `$to`.

4. Змінні `$date` і `$time` використовуються для збереження поточної дати та часу відправлення електронного листа відповідно. Функція `date()` використовується разом із специфікаторами формату для отримання поточної дати й часу.

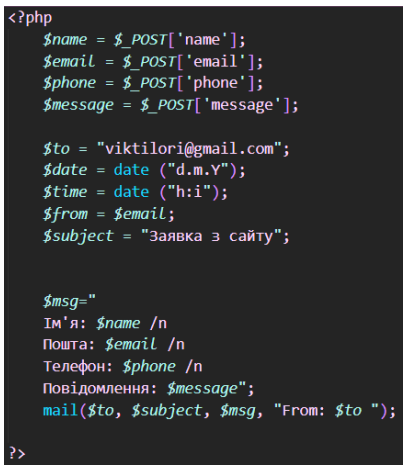

Рис. 2.54. Фрагмент коду використання PHP для контактної форми

5. Змінній `$from` присвоюється значення змінної `$email`, яка представляє адресу електронної пошти відправника.

6. Змінній `$subject` присвоюється рядок «Заявка з сайту». Цей рядок використовуватиметься як тема електронного листа.

7. Змінна `$msg` створена за допомогою багаторядкового рядка. Він містить дані форми (ім’я, електронну пошту, телефон і повідомлення), які будуть включені в тіло електронного листа. Змінні інтерполюються в рядку за допомогою символу `$`, за яким слідує назва змінної. Символи `/n` представляють новий рядок.

8. Функція `mail()` використовується для надсилання електронного листа. Він приймає чотири параметри: адресу електронної пошти одержувача (`$to`), тему електронного листа (`$subject`), тіло електронного листа (`$msg`) і необов’язкове додаткове поле заголовка (у цьому випадку «Від: $до"). Функція надсилає електронний лист зазначеному одержувачу.

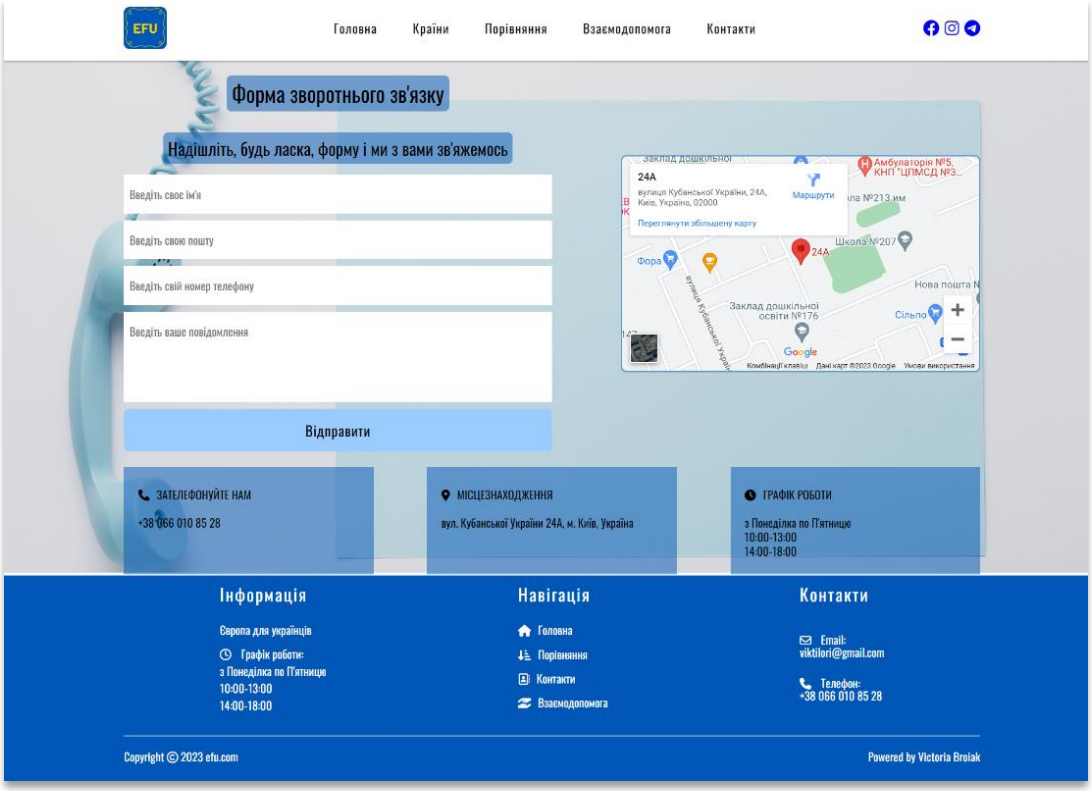

Рис. 2.55. Сторінка «Контакти»

2.10. Висновок до розділу 2

У другому розділі представлений комплексний аналіз факторів, які враховують українські біженці при виборі країни для поселення в Європейському Союзі.

Обговорено принципи дизайну інтерфейсу користувача, включаючи візуальну ієрархію та особливості макета. Презентовано каркаси та макети, що демонструють запланований дизайн інтерфейсу web додатку «Європа для українців».

Продемонстровано реалізацію елементів HTML та їхнє семантичне значення у web додатку. Продемонстровано використання таких функцій SCSS, як змінні, міксини, вкладення та частини в процесі розробки інтерфейсу користувача. Продемонстровано застосування принципів БЕМ до структури та стилю компонентів UI у web додатку «Європа для українців».

Надано приклади фрагментів коду JavaScript для реалізації певних функцій або поведінки у web програмі. Продемонстровано інтеграцію PHP у розробку інтерфейсу користувача для отримання та відображення відповідної інформації з сервера.

РОЗДІЛ 3
ЗБІР ДАНИХ ТА ОЦІНКА ЗВУЧНОСТІ ВИКОРИСТАННЯ

3.1. Збір даних і керування ними

Успіх web додатку «Європа для українців» значною мірою залежить від наявності та точності даних про клімат, вартість життя, умови працевлаштування, податки, системи охорони здоров'я та програми соціальної допомоги в різних країнах-членах Європейського Союзу (ЄС). У цьому підрозділі буде розглянуто важливість збору та управління даними в контексті web додатку, обговорено методи, джерела та стратегії, які використовуються для забезпечення надання надійної та актуальної інформації для українських біженців, які шукають можливості поселення в ЄС.

3.1.1. Методи збору даних

Збір даних для web додатку «Європа для українців» – це багатогранний процес, який включає різні методи та джерела. Ефективний збір даних має важливе значення для забезпечення точності та надійності інформації, представленої у web додатку. Для збору даних про клімат, вартість життя, умови працевлаштування, податки, системи охорони здоров'я та програми соціальної допомоги використовуються різні методи. Ці методи включають:

1. Онлайн-дослідження. Для отримання найточнішої та найновішої інформації проводяться масштабні дослідження з використанням авторитетних онлайн-джерел, урядових web сайтів, офіційної статистики та звітів.

Проведемо одне з онлайн досліджень для пошуку інформації про систему охорони здоров’я Австрії. Ось інформація про австрійську систему охорони здоров’я, зібрана з авторитетних онлайн-джерел, урядових web сайтів, офіційної статистики та звітів.

Австрійська система охорони здоров'я відома своїм високим рівнем обслуговування та повним охопленням. В основному він фінансується за рахунок поєднання внесків на соціальне медичне страхування, податків і платежів із власної кишені. Система створена для того, щоб усі жителі мали доступ до необхідних медичних послуг, незалежно від їхнього доходу чи статусу зайнятості.

Система соціального медичного страхування в Австрії є обов'язковою і забезпечує охоплення більшості населення. Він фінансується за рахунок внесків найманих працівників, роботодавців і самозайнятих осіб. Найбільшим постачальником соціального медичного страхування в Австрії є Фонд загального медичного страхування (Allgemeine Unfallversicherungsanstalt, AUVA), який охоплює працевлаштованих осіб.

Австрія має добре розвинену мережу постачальників медичних послуг, включаючи лікарні, лікарів загальної практики, спеціалістів та амбулаторні клініки. Система охорони здоров’я приділяє значну увагу первинній медичній допомозі, а лікарі загальної практики є першим пунктом контакту для пацієнтів.

Австрійська система охорони здоров’я гарантує всім жителям рівний доступ до медичних послуг. Невідкладна медична допомога доступна цілодобово без вихідних, і немає списків очікування на необхідне лікування. Пацієнти мають свободу вибору постачальників медичних послуг, у тому числі спеціалістів, у системі соціального медичного страхування.

Соціальне медичне страхування в Австрії включає широкий спектр медичних послуг, таких як консультації з медичними працівниками, стаціонарне лікування, ліки, профілактичні огляди та реабілітаційні послуги. Однак деякі послуги можуть вимагати співоплати або додаткового приватного страхового покриття.

Австрія підтримує високі стандарти якості та безпеки охорони здоров’я. У країні є добре підготовлені медичні працівники та сучасні медичні заклади, оснаще-

ні передовими технологіями. Заходи забезпечення якості, такі як акредитація та ліцензування, застосовуються для забезпечення надання високоякісної медичної допомоги.

Австрійська система охорони здоров'я покриває значну частину витрат на призначені ліки. Система відшкодування базується на списку схвалених ліків, при цьому пацієнти сплачують невелику доплату. Австрійське агентство з лікарських засобів (AGES Medizinmarktaufsicht) регулює безпеку та ефективність фармацевтичних препаратів у країні.

Австрійська система охорони здоров’я надає пріоритет зміцненню здоров’я та профілактиці захворювань. Він пропонує різноманітні профілактичні послуги, такі як вакцинація, скринінг раку та програми медичної освіти, щоб допомогти людям зберегти гарне здоров’я та запобігти виникненню захворювань.

Джерела які були використані для проведення даного аналізу:

1. Федеральне міністерство праці, соціальних справ, охорони здоров’я та захисту прав споживачів (2022). Система охорони здоров'я в Австрії. Отримано з https://www.sozialministerium.at/Themen/Gesundheit/Gesundheitssystem/Gesundheitssystem-und-Qualitaetssicherung.html

2. Європейська обсерваторія систем та політики охорони здоров’я (2019). Австрія: огляд системи охорони здоров'я. Отримано з https://apps.who.int/iris/handle/10665/327980

3. Євростат. Діяльність у сфері охорони здоров'я - основні показники. Отримано з https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Health

2. Опитування та інтерв’ю. Опитування та інтерв’ю можна проводити з відповідними зацікавленими сторонами, такими як урядовці, експерти в цій галузі та українські біженці, які вже оселилися в країнах ЄС. Ці основні методи збору даних дають цінну інформацію та досвід з перших вуст.

Надамо приклад одного з опитувань (табл. 3.1) з українським біженцем, який оселився у Австрії та надав відповіді на питання про тимчасовий захист.

Таблиця 3.1

Опитування з українським біженцем з Австрії

1. Наскільки ви знайомі з програмами соціальної допомоги, доступними українським біженцям під тимчасовим захистом в Австрії?	Добре ознайомлений
2. Чи користувалися Ви особисто будь-якими програмами соціальної допомоги під час перебування в Австрії?	Так
3. Вкажіть, будь ласка, якими програмами соціальної допомоги ви користувалися, і коротко опишіть свій досвід.	Я отримав доступ до програми Рік інтеграції, яка передбачала мовні курси та підтримку в пошуку роботи. Програма дуже допомогла мені в процесі інтеграції. Однак отримати допомогу на житло було складно через обмежену доступність.
4. Як би Ви оцінили ефективність і доступність програм соціальної допомоги українським біженцям в Австрії?	Дуже ефективні і доступні
5. З якими проблемами, якщо такі були, ви стикалися під час доступу до програм соціальної допомоги в Австрії?	Головною проблемою, з якою я зіткнувся, була обмежена доступність житлової допомоги. Важко було знайти відповідне та доступне житло в рамках програми.
6. Чи є якісь покращення чи додаткові заходи підтримки, які, на вашу думку, слід запровадити для кращої допомоги українським біженцям під тимчасовим захистом?	**Я вважаю, що розширення варіантів житлової допомоги та надання більшої інформації про доступні послуги соціальної підтримки принесе велику користь українським біженцям. Крім того, створення можливостей для спілкування з місцевими громадами та організаціями могло б посилити зусилля з інтеграції.**
7. Чи отримали ви достатньо інформації про програми соціальної допомоги, коли прибули до Австрії?	Частково деяка інформація була відсутня або незрозуміла

Продовження таблиці 3.1.

8. Наскільки ви задоволені загальною підтримкою, яку надають програми соціальної допомоги в Австрії?	**Певною мірою задоволений**
9. Чи рекомендували б ви програми соціальної допомоги в Австрії іншим українським біженцям, які шукають тимчасового захисту?	Так
10. Поділіться будь-якими додатковими коментарями чи пропозиціями щодо програм соціальної допомоги для українських біженців в Австрії.	Було б корисно мати спеціалізований допоміжний персонал, який міг би супроводжувати біженців через процес подання заявки на соціальну допомогу та оперативно вирішувати будь-які проблеми чи запитання. Крім того, періодичні інформаційні сесії про доступні програми та оновлення були б доречними.

3. Агрегація даних. Дані мають бути зведені з надійних джерел, таких як міжнародні організації, дослідницькі установи та авторитетні бази даних, що забезпечує комплексний і цілісний підхід до збору інформації.

Враховуючи динамічний характер факторів, що впливають на процес прийняття рішень українськими біженцями, регулярне оновлення даних є важливим. Частота оновлення даних визначається на основі наявності нової інформації та значущості змін у таких факторах, як умови працевлаштування, податкова політика та системи охорони здоров'я.

3.2. Оцінка зручності використання та досвіду користувача

Зручність використання та оцінка досвіду користувача є невід'ємними частинами орієнтованого на користувача підходу до проектування. Для оцінки зручності використання та ефективності дизайну використаємо різні методи оцінки, такі як евристичні оцінки, когнітивні покрокові інструкції та тестування користувача.

Евристичні оцінки включають експертів-оцінювачів, які систематично оцінюють дизайн на основі набору принципів зручності використання або евристик. Ця оцінка допомагає виявити потенційні проблеми з зручністю використання та області для покращення.

Когнітивні покрокові інструкції імітують процес прийняття рішень користувачем і оцінюють простоту використання інтерфейсу та можливість навчання.

Тестування користувачів передбачає спостереження за реальними користувачами, які взаємодіють із програмою, а також збір їхніх відгуків і думок. Цей метод оцінки надає цінну інформацію про те, наскільки добре інтерфейс підтримує користувачів у виконанні їхніх завдань і досягненні цілей. Відгуки користувачів збираються за допомогою інтерв'ю, опитувань і спостережень і ретельно аналізуються, щоб повідомити про подальші вдосконалення дизайну.

Відгуки користувачів для оцінки зручності використання та ефективності дизайну (користувачі виявили бажання залишатися анонімними):

1. Зворотний зв'язок евристичної оцінки:

– Користувач A: "Навігація інтуїтивно зрозуміла та проста для розуміння. Однак розмір шрифту в розділі інформації про країни дещо маленький, тому його трохи важко читати.".

– Користувач B: «Загалом дизайн дотримується послідовних візуальних шаблонів і відповідає галузевим стандартам. Я натрапив на кілька непрацюючих посилань, що вплинуло на взаємодію з користувачем».

2. Когнітивні покрокові інструкції. Зворотній зв'язок:

– Користувач C: «Інструкції щодо заповнення контактної форми були чіткими та лаконічними. До речі форму було знайти не складно, адже вона розміщена на видимій частині сайту».

3. Відгуки користувачів про тестування:

– Користувач D: «Функція порівняння працювала добре й дозволяла мені фільтрувати країни за певними критеріями. Це допомогло мені звузити вибір і ефективно порівняти потрібну інформацію».

3.3. Врахування відгуків користувачів

Врахування цих відгуків в процесі проектування може допомогти визначити сфери для вдосконалення та забезпечити ефективну web програму, орієнтовану на користувача, для українських біженців, які шукають інформацію про поселення в Європейському Союзі.

Завдяки активному залученню користувачів до процесу розробки web додаток «Європа для українців» гарантує, що кінцевий дизайн буде зручним, відповідатиме очікуванням користувачів, а також забезпечуватиме безперебійну та змістовну взаємодію з користувачем. Постійне залучення користувачів і зворотний зв'язок дозволяють команді дизайнерів приймати обґрунтовані дизайнерські рішення та створювати інтерфейс, який ефективно підтримує українських біженців у їхньому поселенні в межах Європейського Союзу.

3.4. Висновок до розділу 3

У третьому розділі було представлено методи збору даних таких як онлайн дослідження, опитування та агрегація даних.

Також було проведено оцінювання зручності використання на основі відгуків користувачів, які використовували певні методи оцінки, такі як евристичні оцінки, когнітивні покрокові інструкції та тестування користувача.

Підсумовуючи, орієнтований на користувача підхід до дизайну web додатку «Європа для українців» передбачає також тестування зручності використання та врахування відгуків користувачів. Розуміючи потреби, цілі та вподобання українських біженців, ми можемо створити користувальницький інтерфейс, який відповідатиме їхнім конкретним вимогам, полегшить використання та забезпечить бездоганну та привабливу взаємодію з користувачем.

ВИСНОВКИ

Підсумовуючи вищевикладене, можна зробити висновок, що всі поставлені перед дипломним проектом завдання були успішно виконані, що призвело до досягнення мети проекту.

Зроблено вибір програмного забезпечення, який включав оцінку різних варіантів з урахуванням конкретних вимог проекту, таких як ефективність, простота використання та гнучкість.

Загалом результатом вибору та використання програмного забезпечення при розробці web додатку «Європа для українців» став ефективний, гнучкий та високофункціональний додаток. Вибір PHP, HTML, CSS, SCSS, Bourbon, BEM і JQuery виявився виграшною комбінацією, що дозволило команді розробників створити web програму найвищої якості, яка відповідала та перевершила очікування клієнтів.

Було представлено комплексний аналіз факторів, які враховують українські біженці при виборі країни для поселення в Європейському Союзі.

Обговорено принципи дизайну інтерфейсу користувача, включаючи візуальну ієрархію та особливості макета. Презентовано каркаси та макети, що демонструють запланований дизайн інтерфейсу web додатку «Європа для українців».

Продемонстровано реалізацію елементів HTML та їхнє семантичне значення у web додатку. Продемонстровано використання таких функцій SCSS, як змінні, міксини, вкладення та частини в процесі розробки інтерфейсу користувача. Продемонстровано застосування принципів БЕМ до структури та стилю компонентів UI у web додатку «Європа для українців».

Надано приклади фрагментів коду JavaScript для реалізації певних функцій або поведінки у web програмі. Продемонстровано інтеграцію PHP у розробку інтерфейсу користувача для отримання та відображення відповідної інформації з сервера.

Було представлено методи збору даних таких як онлайн дослідження, опитування та агрегація даних.

Також було проведено оцінювання зручності використання на основі відгуків користувачів, які використовували певні методи оцінки, такі як евристичні оцінки, когнітивні покрокові інструкції та тестування користувача.

В цілому, створення web додатку для збору та порівняння інформації про різні країни може допомогти українцям, які шукають тимчасовий захист та можливість переїхати до інших країн. Такий додаток може зробити процес пошуку інформації більш зручним та ефективним, тим самим допомогти українським переселенцям у новому етапі життя.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The European Union – what it is what it does. [Electronic resourse]. – Access mode: https://op.europa.eu/webpub/com/eu-what-it-is/en/ (lastaccess: 20.04.2023). – Title from the screen.

2. E. A. Meyer gitlabCSS: The Definitive Guide / E. A. Meyer, E. Weyl; O'Reilly Media, Inc. - 1005 Gravenstein Highway North, Sebastopol, CA : 2017. – 1088 с. - Електрон. аналог друк. вид.: режим доступу: http://projanco.com/Library/CSS%20The%20De%C2%80finitive%20Guide.pdf (дата звернення: 29.04.2023.) – Назва з екрану.

3. D. Cederholm Sass for web designers / D. Cederholm, C. Coyier; A Book Apart - New York : 2013. – 98 с.

4. Методологія БЕМ [Електронний ресурс]. – Режим доступу: https://en.bem.info/methodology/ (дата звернення: 02.05.2023.) – Назва з екрана.

5. J. Chaffer Learning jQuery / J. Chaffer, K. Swedberg; Packt Publishing Ltd. - Livery Place 35 Livery Street Birmingham B3 2PB, UK : 2013. – 444 с. - Електрон. аналог друк. вид.: режим доступу: https://it.dru.ac.th/o-bookcs/pdfs/28.pdf (дата звернення: 07.05.2023) – Назва з екрана.

6. Томсон Л. Web розробка PHP і MySQL / Томсон Л., Веллінг. Л.; Sams Publishing – United States of America : 2013. – 893 с. - Електрон. аналог друк. вид.: режим доступу: https://repository.unikom.ac.id/32751/1/php%20and%20mysql%20web%20dev.pdf (дата звернення: 09.05.2023)

7. Робсон Е. Head First HTML and CSS / Робсон, Е., Фрімен, Е.; – O'Reilly Media: Sebastopol, CA, 2014. – 720 с. – Електрон. аналог друк. вид.: режим доступу: http://artsites.ucsc.edu/sdaniel/170a_2014/Head_First_HTML_CSS_XHTML.pdf (дата звернення 01.05.2023) – Назва з екрана.

8. Шклар Л. Архітектура web додатків: принципи, протоколи та практики / Шклар, Л., Розен, Р.; - John Wiley & Sons Ltd. – England : 2013. – 374 с. Електрон. аналог друк. вид.: режим доступу: http://bedford-computing.co.uk/learning/wp-

content/uploads/2016/07/Web-Application-Architecture-Principles-Protocols-and-Practices.pdf (дата звернення: 25.04.2023) – Назва з екрана.

9. Дакетт Дж. Web дизайн із набором HTML, CSS, JavaScript і jQuery / Дж. Дакетт; John Wiley & Sons, Inc., - Indianapolis: 2017 – 514 с. Електрон. аналог друк. вид.: режим доступу: https://d-pdf.com/book/2539/read (дата звернення: 03.05.2023) – Назва з екрана.

10. Маркотт, І. Адаптивний web дизайн / І. Маркотт. - Нью Йорк : A Book Apart, 2014. - 150 с.