Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ІНФОРМАЦІЙНОЇ БІБЛІОТЕКИ ГОТОВИХ РІШЕНЬ CSS І HTML ДЛЯ РОЗРОБНИКІВ ІНТЕРФЕЙСІВ КОРИСТУВАЧА

DEVELOPMENT OF INFORMATION LIBRARY OF READY CSS AND HTML SOLUTIONS FOR USER INTERFACE DEVELOPERS

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗ-41
Сергійчук Валентин Петрович
_____
(підпис)

Керівник
д.т.н., професор
Гордєєв Олександр Олександрович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2024 р.

Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
______ Н. Лінник
«2» 02 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Сергійчука Валентина Петровича
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *розробка інформаційної бібліотеки готових рішень CSS і HTML для розробників інтерфейсів користувача*

Керівник роботи: *д.т.н. Професор Гордєєв О.О.*
затверджені наказом закладу вищої освіти від «*30*» *грудня 2023 р.* № *477/01-02*

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*

3. Вихідні дані до роботи: *Вимоги до розробки програмного забезпечення. Технології програмування клієнтської частини – HTML, CSS, JavaScript.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *Аналіз існуючих методів та засобів для розробки програмного забезпечення, та їх вибір, аналіз та опис функціонального наповнення для програмного забезпечення, детальний опис розробки та тестування програмного забезпечення, опис роботи програми*

5. Перелік графічного матеріалу: *18 рисунків; 1 таблиця*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		Завдання видав	Завдання прийняв
Аналіз предметної області	*Гордєєв О. О.*		
Специфікація вимог до розробленої системи	*Гордєєв О. О.*		
Розробка об'єкта проектування	*Гордєєв О. О.*		
Нормоконтроль	*Повстяна Ю .С*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	2,7 %		
Академічна доброчесність	*Яцук А. А.*		

7. Дата видачі завдання *« 2 » лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*11.02.2024 р.*	
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*26.02.2024 р.*	
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*07.03.2024 р.*	
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*19.04.2024 р.*	
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*15.05.2024 р.*	
6	*Тестування та налагодження об'єкта проектування*	*01.06.2023 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	

Здобувач вищої освіти _____ (підпис) Сергійчук В. П. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Гордєєв О. О. (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: *Сергійчук Валентин Петрович*

Тема: *«Розробка інформаційної бібліотеки готових рішень CSS і HTML для розробників інтерфейсів користувача».*

Коротка характеристика кваліфікаційної роботи та прийнятих рішень: *Кваліфікаційна робота бакалавра складається з трьох розділів, в яких проаналізовані проблематика та поставлені завдання за темою роботи, здійснено теоретичне дослідження та практичну реалізацію інформаційної бібліотеки готових рішень, розкрито суть експериментального дослідження результативності та функціонування додатку.*

Самостійні розробки і пропозицій автора: *Автор самостійно спроектував та розробив клієнтську частину інформаційної системи за допомогою використання веб-технологій та відповідного програмного забезпечення.*

Практичне значення роботи *полягає в теоретичному дослідженні методів розробки веб-додатків та створення інформаційної бібліотеки готових рішень для використання її в проектах інших розробників.*

Недоліки: *Істотних недоліків в роботі не виявлено.*

Загальний висновок: *У кваліфікаційній роботі бакалавра наведені результати вирішення актуального інженерного завдання з обґрунтування розробки інформаційної бібліотеки готових рішень CSS і HTML для розробників інтерфейсів користувача. Робота є результатом самостійної розробки студента. Істотних недоліків не виявлено. Кваліфікаційна робота здобувача освіти Сергійчука Валентина Петровича рекомендується до захисту екзаменаційній комісії та заслуговує позитивної оцінки.*

Рецензент:
Ухаль Анатолій Миколайович к.т.н. доцент каф. ЕІТК
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«07» 06 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет

Факультет *комп'ютерних та інформаційних технологій*
(назва)

Кафедра *інженерії програмного забезпечення*
(назва)

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти: *Сергійчук Валентин Петрович*
(ПІБ)

група *ІПЗ-41*
(шифр)

Тема кваліфікаційної роботи:

Розробка інформаційної бібліотеки готових рішень CSS і HTML для розробників інтерфейсів користувача
Development of Information Library of Ready CSS and HTML Solutions For User Interface Developers
(повна назва згідно наказу)

Керівник: *д.т.н професор Гордєєв О.О.*
(науковий ступінь, вчене звання, посада, ПІБ)

Актуальність теми:

Для ознайомлення з новими бібліотеками розробник може витрачати багато часу. Це пов'язано з відсутністю розробленого єдиного стандарту представленні інформації та обновлюваної документації до бібліотек та, як наслідок, відсутністю формалізованих інтерфейсів користувача.

Об'єкт дослідження:

UI компоненти бібліотеки, процеси їхньої розробки та впровадження.

Характеристика теоретичного рівня роботи, наявності самостійних розробок і практичної значущості роботи:

Кваліфікаційна робота містить усі необхідна розділи, обсяг та на змістовне наповнення. Дипломна робота відповідає усім вимогам, які висуваються для таких робот бакалаврського рівня.

Зауваження та недоліки:

Вимоги до інтерфейсу інформаційної бібліотеки готових рішень сформульовані не досить чітко та однозначно.

Загальний висновок:

Кваліфікаційна робота відповідає вимогам, студент був допущений до захисту. Рекомендована оцінка «добре».

Керівник _____ (*Гордєєв Олександр Олександрович*)
(підпис) (ПІБ)

«7» 06 2024 р.

ЗМІСТ

АНОТАЦІЯ

Сергійчук В. П. Інформаційна бібліотека готових рішень CSS і HTML для розробників інтерфейсів користувача. Рукопис.

Кваліфікаційна робота бакалавра. Кваліфікаційна робота за спеціальністю 121 Інженерія програмного забезпечення. Луцьк 2024. 47 с.

Кваліфікаційна робота присвячена розробці та реалізації інформаційної бібліотеки готових рішень для розробників інтерфейсів з використанням CSS,HTML і JavaScript.

Робота складається з трьох розділів, в яких описано проблематику веб-розробки та розв'язання завдань, пов'язаних з нею. Описано процес розробки бібліотеки, здійснено дослідження сучасних бібліотек та фреймворків та описано їхнє використання. Показано інструменти та методології для розробки інтерфейсів користувача.

Ключові слова: інформаційна бібліотека, інтерфейс, front-end, HTML, CSS, JavaScript.

ABSTRACT

Serhiichuk V. P. Information Library of Ready-Made Solutions in CSS and HTML for User Interface Developers. Manuscript.

Bachelor's Qualification Work. Qualification Work in the Specialty 121 Software Engineering. Lutsk 2024. 47 p.

The qualification work is dedicated to the development and implementation of an information library of ready-made solutions for user interface developers using CSS, HTML, and JavaScript.

The work consists of three sections, which describe the problems of web development and the solutions related to it. It outlines the development process of the library, conducts a study of modern libraries and frameworks, and describes their use. It also presents tools and methodologies for user interface development.

Keywords: information library, interface, front-end, HTML, CSS, JavaScript.

ВСТУП

В сучасних тенденціях розробка користувацького інтерфейсу займає все більше часу та вимагає більше ресурсів. Адже теперішній ринок вимагає швидких результатів, а для цього потрібно використовувати додаткові інструменти, такі як бібліотеки. Одним з найбільш важливих факторів, що впливають на ефективність розробки, є доступність і якість цих бібліотек. Адже, над їхнім створенням можуть працювати як і команда розробників якоїсь компанії, так і один розробник, що є сумнівним з точки зору якості та може призвести до проблем із недостатньою стандартизацією бібліотек. Через відсутність розробленого єдиного стандарту написання та оновлюваної документації, розробники стикаються з труднощами під час застосування бібліотек у своїх проектах. Також перешкодою для освоєння деяких сучасних бібліотек є їхня надмірна складність. Для того щоб їх вивчити витрачається багато часу, а такий підхід не відповідає вимогам ринку.

Напрям front-end розробки є досить поширеним в IT індустрії, через те що переважна більшість новачків в програмуванні починає свій шлях професіонального програміста саме із цієї галузі.

Таким чином вищезазначені проблеми у технологіях веб-розробки негативно впливають на якісь програмних продуктів та уповільнюють розвиток IT в цілому. Це потребує детального вивчення та пошуку шляхів для їх вирішення.

Метою даної роботи є розробка інформаційної бібліотеки, що буде зрозуміла для користувачів, які тільки починають вивчати подібні бібліотеки. Вона буде містити найпоширеніші компоненти, що використовуються в будь-яких інтерфейсах.

Об’єктом дослідження процеси розробки компонентів інформаційної бібліотеки їх впровадження та використання.

Предметом дослідження виступають самі готові рішення розроблені на CSS та HTML.

Завдання, які необхідно виконати:

– дослідити можливості веб-технологій для розробки інформаційної бібліотеки, порівняти їхні переваги та недоліки;

– визначення архітектури, потрібно провести детальний аналіз, які саме задачі буде виконувати бібліотека і дослідити, які компоненти та стилі потрібні розробникам;

– спроектувати макет за допомогою спеціальних програм, створити представлення зовнішнього вигляду бібліотеки, визначити перелік компонентів, створити UML-діаграму;

– верстка бібліотеки де буде описано основні стилі для базових елементів. Також для кожного компонента буде створено окремий файл зі стилями та додана інтерактивність;

– реалізувати компоненти інтерфейсу користувача з використанням HTML, CSS та JavaScript;

– тестування і виправлення помилок, де буде проводено перевірку функціональності всіх компонентів, щоб пересвідчитись у їхній коректній роботі, а також перевірятиму як вони працюють на різних браузерах.

– завантажити на GitHub готовий проект, де спільнота розробників зможе використовувати його стилі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Зі стрімким цифровим розвитком, більшість галузей діяльності людини та компаній потребують своєї веб-сторінки чи веб-сайту. Таким чином, веб-розробка залишається такою ж популярною та займає високе місце в IT індустрії.

Front-End – клієнтська частина, яка відповідає за взаємодію користувача з інтерфейсом.

Back-End – серверна частина, що відповідальна за зв’язок бази даних з клієнтською частиною, обробку та збереження даних.

Розробнику важливо мати можливість використовувати правильні інструменти, такі як бібліотеки, фреймворки, препроцесори, Git тощо. У контексті front-end напряму бібліотеки готових стилів та компонентів відіграють важливу роль допомагаючи оптимізувати процес написання програмного коду, запобігають появі типовим помилкам та покращують його якість.

Бібліотеки готових рішень – це набори попередньо протестованих компонентів коду: функцій, модулів та інструментів, які розробник може застосувати на своїй веб-сторінці для написання типових базових елементів.

Для початку розглянемо основні мови програмування, які можна використовувати при розробці інтерфейсів, а саме: HTML, CSS, JavaScript і TypeScript.

HTML (Hyper Text Markup Language) – мова гіпертекстової розмітки документу, за допомогою якої формується структура веб-додатку (навігація, заголовки, контейнери, header, footer тощо). Розробники надають перевагу останній версії HTML5, яка має удосконалену семантику, медіа-запити та загалом різноманітність інших функцій, що недоступні попереднім версіям.

Оскільки HTML використовується для формування структури веб-документу, його зовнішній вигляд не буде привабливий для звичайного користувача. Тому для стилізації інтерфейсу необхідно застосовувати CSS.

CSS (Cascading Style Sheets) – це мова стилів, які застосовуються до HTML-документів, форматує та змінює їх зовнішній вигляд. Найпоширеніші функції у CSS, такі: зміна кольору, розміру, шрифту, позиціонування, фону. На даний час останньою версією, що продовжує розвиватись, можна вважати CSS3. Нова версія додала до мови багато нових функцій, такі як: гнучкі контейнери, сітки, імпорт svg-зображень, що покращило адаптивність розроблюваних інтерфейсів.

Тепер за допомогою HTML і CSS можна створити привабливий для користувача веб-додаток. Проте відсутність функціоналу та статичність є проблемою для подальшого його використання. Щоб надати сторінці «живий» вигляд використовують JavaScript.

JavaScript – це високорівнева мова програмування, яка дозволяє додавати на сторінку інтерактивні елементи. Її головна функція полягає у створенні певного функціоналу, який би давав користувачам можливість взаємодіяти з серверною частиною веб-додатку, наприклад: реакція на кліки, позиціонування курсору, відправка форм, модальні вікна, випадаючі меню тощо.

Загальні технології для веб-розробки представлені на рисунку 1.1.

Рисунок 1.1 – Основні мови у Front-end технологіях [11]

TypeScript – ця мова програмування є надбудовою над JavaScript. Ця мова програмування стала популярною серед спільноти розробників через кілька причин, а саме:

– компанія Microsoft активно розвиває та підтримує TypeScript, що дозволяє отримувати розробникам постійні оновлення функціоналу;

– бібліотеки та фреймворки, які на даний час активно використовуються, надають типи для TypeScript, тим самим сприяючи їх використанню у проектах;

– головною перевагою застосування мови є можливість вчасно виявити помилку на етапі компіляці;

– використовує простори імен та класи для забезпечення кращої організації коду.

Вона також входить до списку основних технологій, проте є не надто розповсюдженою серед новачків у веб-розробці. Дана мова програмування зображена на рисунку 1.2.

Рисунок 1.2 – Логотип мови програмування TypeScript [6]

Отже, вищеописаних мов програмування достатньо щоб створити функціональний веб-додаток. Проте існуючі бібліотеки можуть значно спростити розробку та додати новий функціонал. Давайте тепер ширше розглянемо деякі із бібліотек для front-end розробника користувацьких інтерфейсів:

– jQuery – це досить популярна кросбраузена бібліотека JavaScript, яка користується попитом у розробників-початківців. Через свій простий

синтаксис опанувати дану бібліотеку досить просто і почати використовувати її для анімацій та маніпуляцій з HTML-документами(DOM). Основна функція бібліотеки у тому що вона надає API для роботи з елементами HTML, здійснює AJAX-запитів до сервера, надає можливість створити функції для обробки подій тощо. Проте в порівнянні із іншими вона вважається застарілою та програє за продуктивністю;

– Bootstrap або ж Twitter Bootstrap – створена командою розробників з компанії Twitter, вона є найвідомішою бібліотекою у спільноті front-end спеціалістів. Бібліотека має відкритий вихідний код та включає в себе потужний набір інструментів для стилізації веб-документів. Головна філософія Bootstrap базується на адаптивному підходу до написання коду, тобто оптимізації веб-сторінок до будь-якого розширення екрана. Основною її функцією є Grid System або ж Стікова система, що дозволяє створювати адаптивний дизайн на сторінці, розміщуючи елементи по горизонтальній та вертикальній осям. Також семантичний підхід спрощує читабельність коду, а у зв'язці з Grid System покращує SEO веб-сторінок. За сучасними тенденціями у дизайні, стилізація Bootstrap може вважатися застарілою, а через його великий розмір знижується оптимізація завантаження, тому краще використовувати більш сучасні бібліотеки;

– Bluma – це прогресивний CSS фреймворк з відкритим вихідним кодом побудованим на базі Flexbox. Розробники надають перевагу Bluma, тому що він не потребує багато часу для вивчення, а його простий синтаксис дозволяє зрозуміти код без досвіду використання. Також, фреймворк легко інтегрується з іншими бібліотеками, наприклад: Angular, React, Vue, Ember тощо. Широка документація Bluma з готовими прикладами може допомогти початківцям у веб-розробці стилізувати будь-які елементи. На мою думку, основною перевагою є побудова на технології Flexbox, що дозволити адаптувати проект під пізні екрани та наявність уже стилізованих компонентів, які відразу можна інтегрувати у свої проекти. Проте, не дивлячись на його сучасність Bootstrap має більшу кількість компонентів і

сторонніх плагінів, а велика популярність і спільнота переважає бібліотеку Bluma;

– React.js – це JavaScript-бібліотека створена командою розробників Facebook, яка допомагає розробникам створювати користувацькі інтерфейси. Дана бібліотека є досить популярною у спільноті розробників, тому у мережі можна знайти велику кількість інформації, яка допоможе вирішити проблему. Серед усі інших бібліотек JavaScript її першу часто обирають для вивчення початківці. React.js переважно фокусується саме на створенні інтерфейсу для користувачів. Головна функція цієї бібліотеки полягає у запобіганні виникненню помилок при розробці інтерфейсів, це досягається з допомогою автономних фрагментів коду. Зазвичай React.js використовується у зв'язці з React Router та Redux для створення одно сторінкових додатків (SPA), де React Router керує навігацією, а Redux – управляє даними. Через швидкий темп розвитку бібліотеки, виникають проблеми з відстеженням нових функцій, тому розробникам потрібно постійно вчити щось нове.

Статистика пошукових запитів перерахованих бібліотек буде показано на рисунку 1.3.

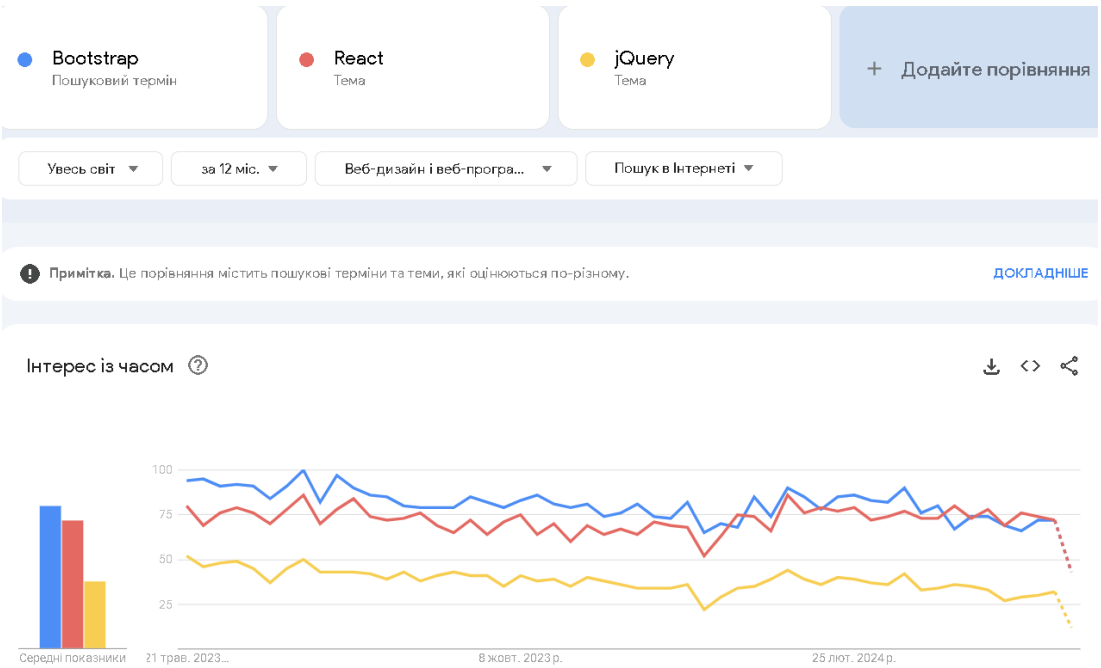

Рисунок 1.3 – Статистика front-end бібліотек [12]

Для створення якісного програмного продукту та й у цілому будь якого програмного забезпечення з його подальшим удосконаленням потрібно визначити архітектуру яку необхідно використовувати.

Для початку необхідно визначити мови, на яких спиратиметься процес створення клієнтської частини програмного продукту. Тому у корінній папці потрібно створити HTML-документ (index.html) в якому буде міститися уся структура проекту, під'єднані стилі, шрифти та скрипти. Наступним кроком буде створення трьох папок: для підключення скриптів (libs), для підключення стилів (css) та для додавання зображень та svg-елементів (img).

Враховуючи складність проекту, у папці (css) потрібно створити головний css-документ до якого будуть підключатися окремі документи зі стилями, які знаходитимуться у папці (parts). Під час розробки така структура допоможе оптимізувати процес пошуку помилок та в загальному підвищить читабельність коду і зручність його написання. Структура клієнтської частини буде показано (на прикладі створюваної інформаційної бібліотеки) в таблиці 1.1.

Таблиця 1.1 – Перелік складових частин архітектури клієнтської частини інформаційної бібліотеки

№	Назва папки або файлу	Призначення
1	index.html	Формує структуру веб-документу
2	easy-toggler.min.js	Обробник подій
3	style.css	Містить підключені файли
4	Ccs	Папка для зберігання головного css-файлу
5	Parts	Папка для зберігання окремих css-файлів
6	Img	Папка для зберігання зображень та svg-файлів
7	Libs	Папка для зберігання js-файлів

Оскільки темою кваліфікаційної робити бакалавра є створення інформаційної бібліотеки готових рішень CSS і HTML для розробників інтерфейсів користувача, доцільно було б використовувати репозиторії GitHab.

GitHab – є платформою де розробники можусь спільно створювати програмне забезпечення. Такий підхід буде корисним при виявленні можливих помилок чи збоїв, адже вручну знаходити неполадки дуже складно та займає багато часу. Крім того, що сервіс може відкочувати проект до попередньої версії, його функції дозволяють переглянути весь код не лише вам, а й іншим розробниками та надає їм можливість редагувати код і додавати до проекту нові функції. Підсумовуючи усе, платформа GitHab надає потужні інструменти для комфортного обміну кодом.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Протягом виконання кваліфікаційної роботи мають бути виконані наступні завдання:

– дослідити можливості веб-технологій для розробки інформаційної бібліотеки, порівняти їхні переваги та недоліки;

– визначення архітектури, потрібно провести детальний аналіз, які саме задачі буде виконувати бібліотека і дослідити, які компоненти та стилі потрібні розробникам;

– спроектувати макет за допомогою спеціальних програм, створити представлення зовнішнього вигляду бібліотеки, визначити перелік компонентів;

– верстка бібліотеки де буде описано основні стилі для базових елементів. Також для кожного компонента буде створено окремий файл зі стилями та додана інтерактивність;

– реалізувати компоненти інтерфейсу користувача з використанням HTML, CSS та JavaScript;

– тестування і виправлення помилок, де буде проводено перевірку функціональності всіх компонентів, щоб пересвідчитись у їхній коректній роботі, а також перевірятиму як вони працюють на різних браузерах.

– завантажити на GitHub готовий проект, де спільнота розробників зможе використовувати його стилі.

Висновки до розділу 1

В даному розділі були на ведені основні технології розробки веб-інтерфейсів. Описано принципи роботи front-end напряму та мови програмування і бібліотеки які можуть при цьому використовуватися. Було проведено порівняння сучасних бібліотек та розглянуто їхні переваги та недоліки. Також наведений приклад архітектури проекту та описано платформу GitHub на яку, в процесі розробки, буде завантажено готову бібліотеку. Постановлено завдання на кваліфікаційну роботу.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Спершу потрібно скласти UML діаграму, щоб зрозуміти як саме буде виконуватись проект. Використаємо для цього діаграму варіантів використання, де можна буде побачити функціонал бібліотеки та можливості для користувачів. Саму діаграму представлено на рисунку 2.1.

Рисунок 2.1 – Діаграма варіантів використання компонентів бібліотеки

Для початку необхідно визначитись з моделлю життєвого циклу програмного забезпечення. Так як, проект який розробляється є інформаційною бібліотекою, логічно було б використати гучку модель. Яка б дозволила отримувати від користувачів зворотній зв’язок та швидко вносити зміни до проекту. Якщо розглядати сучасні бібліотеки, такі як, Bootstrap,

jQuery UI, Angular Material, можна виявити принципи моделей життєвого циклу.

Гнучка модель (Agile Model) – вважається найпопулярнішим методом для розробки бібліотек. Принципи гнучкого підходу спираються на цикли, тобто після кожного такого циклу проводиться огляд, де оцінюють чи був досягнутий бажаний результат, а після цього вносяться необхідні правки. Тому завдяки такій методології та великій спільноті React, Angular, Vue.js постійно оновлюються.

Модель швидкої розробки додатків (RAD) – ця методологія спирається на швидке створення прототипів та тестування, а вже після релізу програмного продукту розробник отримує від користувачів зворотній зв'язок та виправляє недоліки. За цією моделлю працюють і розробники бібліотеки Bootstrap.

Ітеративна модель (Iterative Model) – такий підхід до розробки, де проект реалізується на протязі циклів, чимось схожий на Agile Model, проте цей процес довший. Через те що кожен такий цикл включає в себе ретельну підготовку, розробку, тестування та аналіз, оновлення наприклад, бібліотеки jQuery виходять рідше.

В результаті проведеного дослідження, мною було обрано гнучка модель життєвого циклу ПЗ. Оскільки розроблювана інформаційна бібліотека відповідає принципам даної методології і не є відповідною для інших моделей через нестачу людського ресурсу та часу.

Наступним етапом при проектуванні та розробці веб-додатків є аналіз компонентів користувацького інтерфейсу, так як, їх визначення допоможе зрозуміти структуру та функціонал сторінки, що забезпечить користувачам зручне використання. Потрібно визначити пріоритетність для різних компонентів, а на основі їх значимості додавати у нашу бібліотеку.

Давайте широко розглянемо це питання та візьмемо до прикладу кілька сайтів для порівняння, див. на рисунку 2.2 та 2.3.

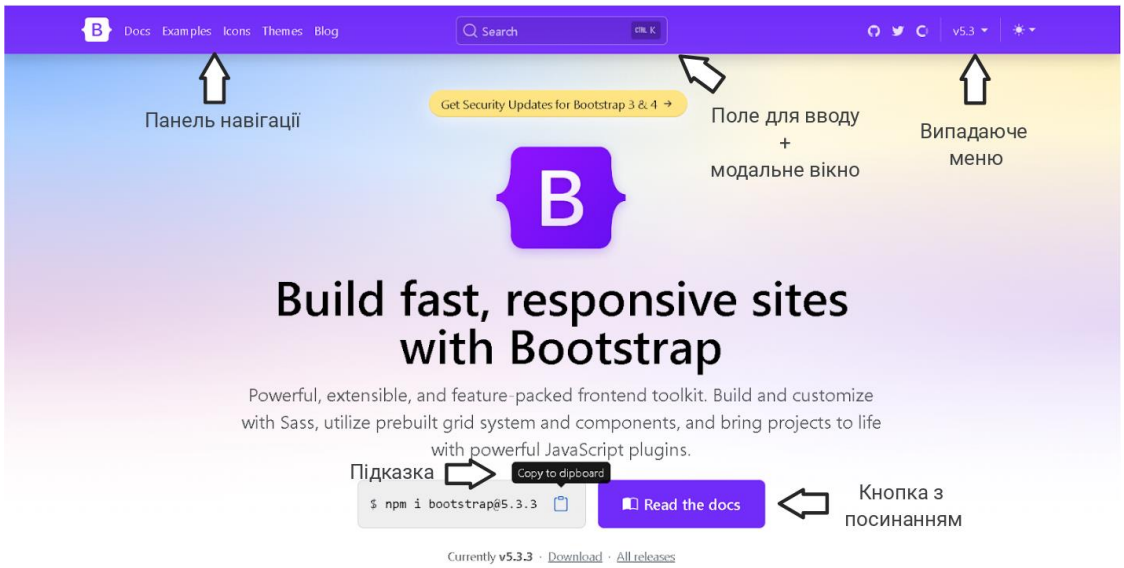

Рисунок 2.2 – Інтерфейс головної сторінки бібліотеки Bootstrap [7]

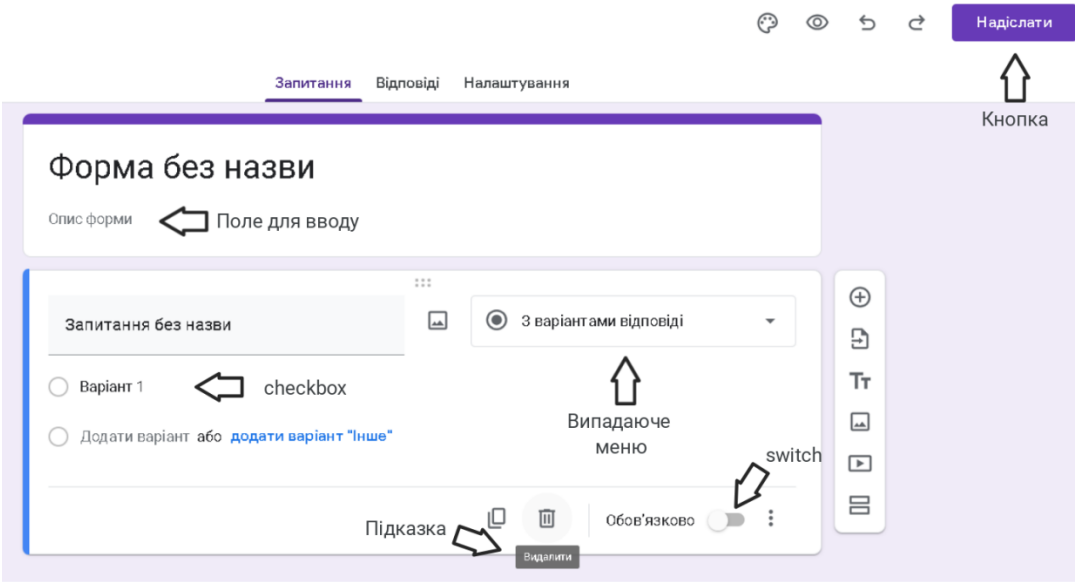

Рисунок 2.3 – Інтерфейс головної сторінки Google Forms [3]

При дослідженні сайтів з різними інтерфейсами були виявлені:

– панель навігації може використовуватись як і для переміщення по самому сайту, так і по окремому розділу. Зазвичай розробники використовують для навігації саме у розділах, щоб покращити читабельність і семантику коду;

– поле для вводу, тут усе зрозуміло, користувач може вводити текстові дані по типу, чисел, тексту, паролів тощо;

– підказки, основне їх завдання підказки в тому, щоб надати користувачеві якусь додаткову інформацію чи зрозуміти які функції у певних елементів;

– випадаюче меню, дозволяє зекономити місце та надати варіанти вибору або функції, доки користувач не активує його;

– checkbox та switch, використовується для визначення стану «обрано» або «не обрано» на формах, фільтрах у каталогах чи для підтвердження. А switch це перемикач який відповідає за ввімкнення чи вимкнення якихось функцій;

– меню сайту, працює схоже з панеллю навігації проте найчастіше використовується для переміщення по самому сайту, а ніж панель навігації, також для покращення читабельності коду;

– картки, їхнє основне призначення у тому щоб компактно і привабливо відображати інформацію для користувача, наприклад, товари в інтернет-магазинах чи якісь анонси новин. В картках часто можуть бути посилання на детальну інформацію у вигляді кнопки чи виділеного слова.

В процесі аналізу, багато інтерфейсів мали навігаційні панелі та картки з інформацією або товаром, тому було вирішено додати їх в основні компоненти.

Після того як було визначено головні компоненти інтерфейсу користувача, можна переходити до створення макету компонентів. Для того щоб побудувати макет компонентів можна використати онлайн-дошку Jamboard Google, приклад буде показано на рисунку 2.4 і 2.5.

Створювана бібліотека буде мати 1 сторінку, отож розглянемо її. Головна сторінка представляє готові стилізовані рішення компонентів інтерфейсу. Тут за допомогою навігації можна переміщатись до розділів, де знаходяться відповідні йому компоненти. На сторінці користувач зможе взаємодіяти з об’єктами, побачити їхню стилізацію та функції.

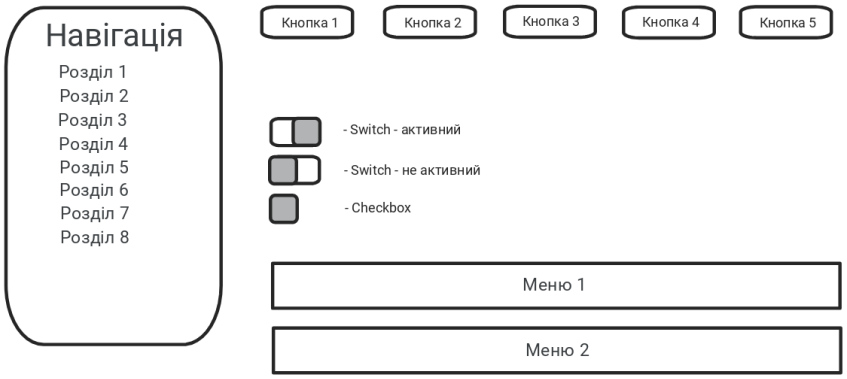

Рисунок 2.4 – Дизайн компонентів інформаційної бібліотеки ч.1

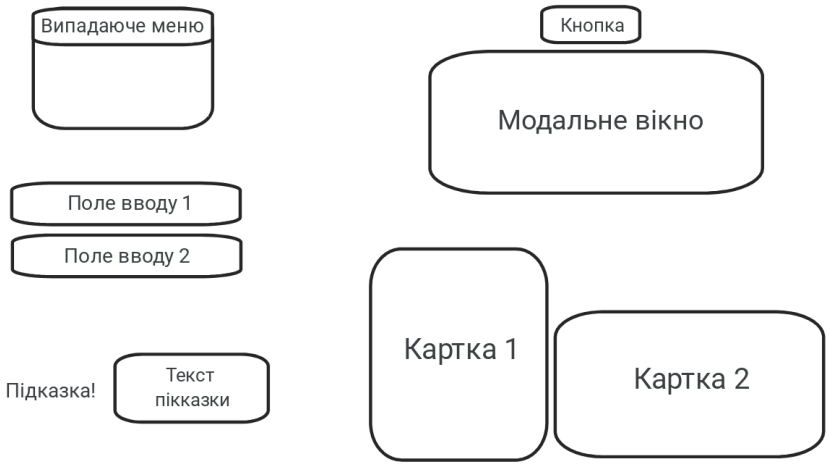

Рисунок 2.5 – Дизайн компонентів інформаційної бібліотеки ч.2

2.2 Вибір засобів, методів та алгоритмів вирішення поставленого завдання

Для початку, потрібно визначитись з мовами програмування. Отож, інтерфейс буде написаний на HTML та CSS з використанням JavaScript, так як реалізація обробки подій буде написана з її допомогою.

Наступним кроком буде визначення методів, які будуть необхідні для успішної розробки бібліотеки. Важливо щоб готовий продукт був зрозумілий для інших користувачів, тобто структура повинна бути логічною та семантично підкріпленою. Також, потрібно розділити компоненти, зробити їх

незалежними від усього проекту. Давайте розглянемо наступні методології що відповідають нашим вимогам.

БЕМ (Блок-Елемент-Модифікатор) – це методологія, що використовується для написання структурованого і читабельного коду, яка особливо широко застосовується розробниками при створенні веб-інтерфейсів користувача. Основні принципи даної методології полягають у тому що:

– блок є незалежним компонентом з власною логікою та стилями, який може бути використаний у проекті повторно. Він може використовуватись незалежно від інших блоків, тому, при перенесенні або зміні, це ніяк не буде впливати на інші частини проекту;

– елемент, як частина блоку не може існувати окремо від нього. Стилі елементу знаходяться у межах певного блоку, тобто це запобігає конфліктам стилів інших блоків чи елементів;

– модифікатор використовується для зміни вигляду і стану блоку або елементу. Він може додаватись як додатковий клас, що буде впливати тільки на блок чи елемент у якому знаходиться. За філософією методології, при написанні модифікатору, потрібно вказати що саме буде змінюватись.

Далі буде показано як саме застосовується описана методологія на рисунку 2.6.

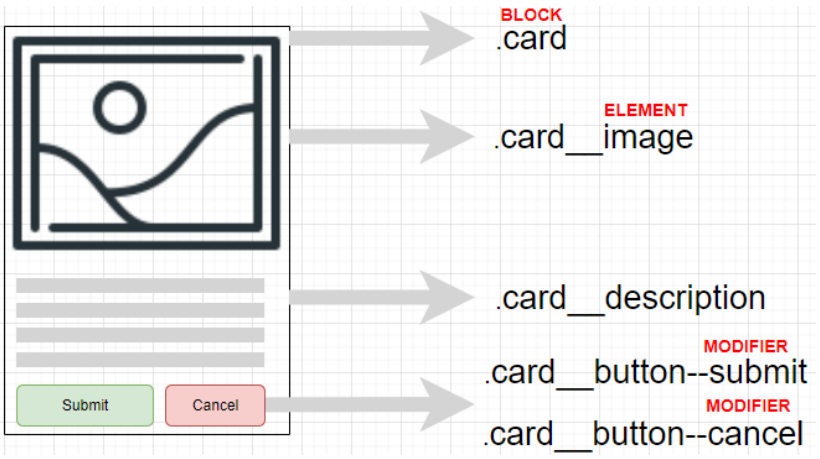

Рисунок 2.6 – Приклад застосування БЕМ методології [10]

Модульний підхід – дана методологія написання веб-додатків розбиває код на окремі частини, які називаються модулі. Кожен такий модуль містить в собі стилізацію та функції певного компоненту інтерфейсу користувача. При такому підході модулі можуться розроблятися окремо, що дозволяє розробникам працювати над великими проектами паралельно, а в разі виявлення помилок швидко їх виправити. Нижче на рисунку 2.7 наведено загальний приклад модульного підходу.

Директорія	Файл	Опис
src/		Коренева директорія проекту
src/components/		Директорія для компонентів
src/components/Button/		Директорія для компонента `Button`
src/components/Button/	Button.jsx	Реалізація компонента `Button`
src/components/Button/	Button.css	Стилі для компонента `Button`
src/components/Header/		Директорія для компонента `Header`
src/components/Header/	Header.jsx	Реалізація компонента `Header`
src/components/Header/	Header.css	Стилі для компонента `Header`
src/components/Footer/		Директорія для компонента `Footer`
src/components/Footer/	Footer.jsx	Реалізація компонента `Footer`
src/components/Footer/	Footer.css	Стилі для компонента `Footer`
src/utils/		Директорія для утиліт
src/utils/	math.js	Утиліта для математичних функцій
src/utils/	date.js	Утиліта для роботи з датами
src/	App.jsx	Основний компонент додатка
src/	index.js	Точка входу додатка

Рисунок 2.7 – Приклад структури модульного підходу

Тепер визначимось з інструментами для розробки. Для створення бібліотеки буде використовуватися редактор коду Visual Studio Code, який підтримується на операційних системах Windows, macOS, Linux. Однією з особливостей Visual Studio Code є відкритий код, таким чином розробники самі покращують даний програмний продукт. Редактор оптимізований для сучасної веб-розробки та постійно оновлюється. Він є безкоштовний, його можна завантажити з офіційного сайту Microsoft або магазину Microsoft Store.

В контексті front-end розробки хочу виділити основні його функції, а саме:

– повноцінне IDE з підтримкою налагодження, а інтеграція з Git забезпечує виконання всіх операцій з версіями коду у самому редакторі, таким чином це створить комфортне середовище для роботи з репозиторіями;

– інструменти для швидкого пошуку файлів та функцій, переходу між файлами, а також використання інших корисних функцій для навігації по проекту;

– VS Code підтримує підсвічування синтаксису та автозавершення для багатьох мов програмування, включаючи HTML, CSS, JavaScript, TypeScript і інші;

– роботу з HTML та CSS полегшують автодоповнення, перевірки на помилки, підказки та інші інструменти;

– Emmet дозволяє швидко генерувати HTML і CSS, використовуючи короткі абревіатури;

– редактор забезпечує можливість встановити розширення для підтримки різних фреймворків та бібліотек, таких як React, Vue, Angular і інші;

– плагін Live Server дозволяє запустити локальний розробницький сервер з «гарячим» перезавантаженням;

– VS Code надає зручні інструменти для спільної роботи над проектами, включаючи можливість обміну кодом у реальному часі, коментування та перегляд змін, що полегшує роботу команд див. на рисунок 2.8.

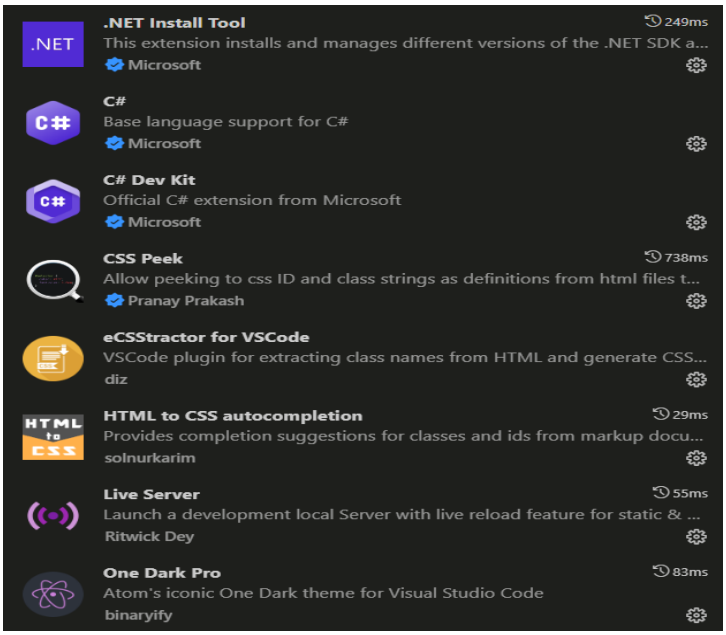

Рисунок 2.8 – браузер розширень VS Code

Сам Visual Studio Code має досить зручний інтерфейс, який дуже легко освоїти, його буде представлено на рисунку 2.9.

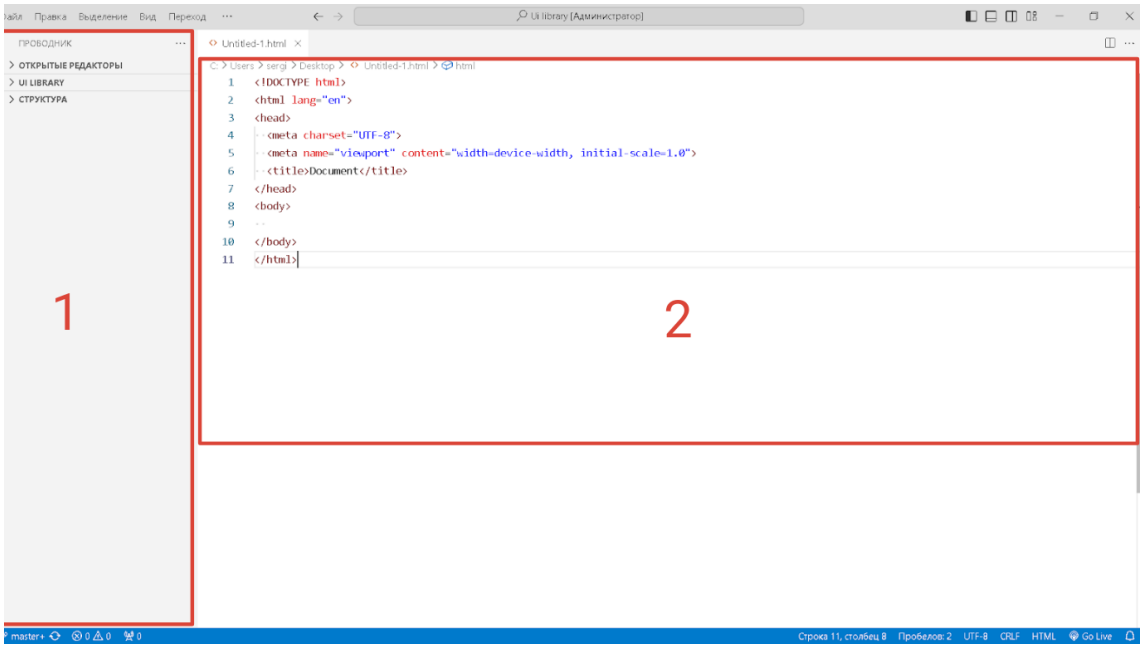

Рисунок 2.9 – Інтерфейс редактора Visual Studio Code

Далі буде розглянуто кожну частину інтерфейсу, що представлено на рисунку 2.9:

– частина 1 – це основна бокова панель, тут знаходиться провідник з допомогою якого можна переглядати папки та файли, а також керувати ними: додавати, редагувати, видаляти. Функція пошуку знаходить всі символи і текст у файлах або ж в одному конкретному файлі. Нижче показано вбудовану систему управління версіями та запуск і відладку для Visual Studio Code. Також є браузер розширень, які можна інтегрувати у свій проект, наприклад: додає підтримку мов програмування і інші допоміжні інструменти. Наступна функція дозволяє працювати з файлами, які знаходяться на сервері або на віддаленому комп’ютері;

– частина 2 – основна частина редактора, це область редагування коду. З її допомогою можна маніпулювати кодом, наприклад: переглядати історію змін, закоментовувати деякі частини коду, згортати код тощо.

Щоб ефективно керувати процесом розробки мною було обрано систему Git, так як вона надає усі необхідні інструменти управління, зараз ми це детально розглянемо.

Git – це система управління версіями, що допомагає розробникам відслідковувати усі зміни у проекті. Її можливості застосовують для створення бібліотек, тому що репозиторії є часто відкриті для інших користувачів, це дозволяє спільно працювати над ними. На мою думку переваго Git є гілкова система, що дозволить створити окремі гілки для написання нових функції, а основна гілка залишиться захищена від змін. Завдяки тому що Git інтегрований у VS Code, я легко завантажу готову бібліотеку на GitHub.

Висновок до розділу 2

У цьому розділі були досліджені головні компоненти інтерфейсу користувача та створені шаблони за якими їх буде розроблено. Розглянуто найбільш поширені методології написання. Були обрані інструменти для розробки інформаційної бібліотеки, такі як редактор коду VS Code та платформа GitHub(з детальним описом про кожен з них).

РОЗДІЛ 3
РОЗРОБКА ІНФОРМАЦІЙНОЇ БІБЛІОТЕКИ ГОТОВИХ РІШЕНЬ CSS І HTML ДЛЯ РОЗРОБНИКІВ ІНТЕРФЕЙСУ КОРИСТУВАЧА

3.1 Практична реалізація об’єкта проектування

Для початку необхідно завантажити редактор коду Visual Studio Code. Після цього налаштуємо його для комфортної веб-розробки, встановимо плагіни Emmet (за допомогою скорочень можна писати розгорнутий HTML і CSS код) та Live Server (запускає локальний сервер з функцією миттєвого перезавантаження).

Тепер потрібно створити кореневу папку з усіма необхідними файлами. Структуру забезпечуватиме файл Index.html, в ньому міститимуться шаблони компонентів, які в подальшому можна використовувати. Далі створимо папку під назвою Css, де зберігатиметься файл style.css, який забезпечить підключення стилів із папки Parts (куди в процесі розробки я буду додавати окремі css-файли зі стилізацією компонентів) до головного html-документу. Також створимо папку Img для зберігання зображень та папку Libs для скрипта easy-toggler.min.js, який буде реагувати на відкриття і закриття вікон чи меню. У програмі Figma, створимо повноцінний макет бібліотеки, що зображений на рисунку 3.1.

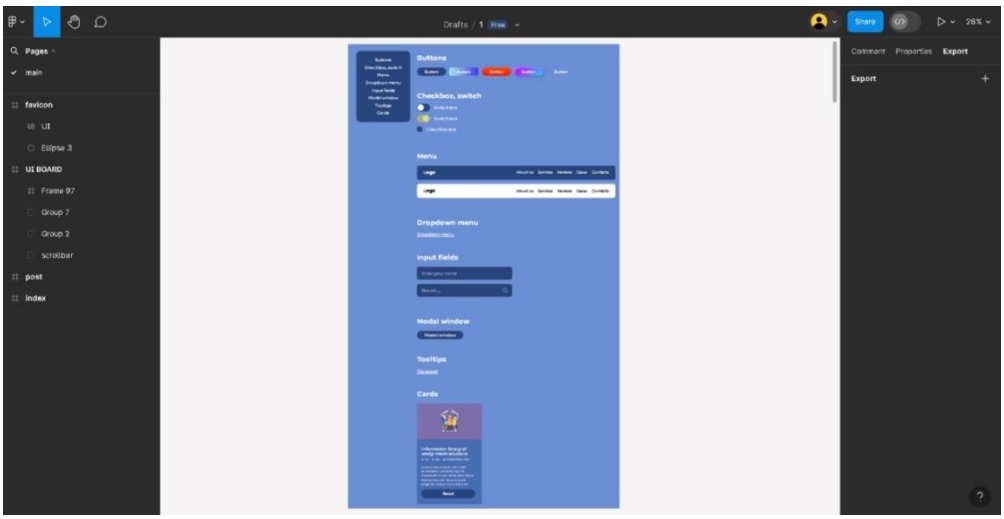

Рисунок 3.1 – Готовий дизайн у програмі Figma

Після встановлення середовища розробки, завантаження необхідних розширень для розробки та налаштування папок можна переходити безпосередньо до кодування.

Отож, спочатку потрібно реалізувати навігацію для швидкого переміщення до наших компонентів. Так як в інформаційній бібліотеці все буде знаходитись на одній сторінці, я використав семантичний тег «nav» у який помістив «посилання-якорі» (наприклад, #text), які при натисканні на них, притягуватимуся до ідентифікатора (id) елементів навігації, як це буде показано на лістингу 3.1. Такий підхід був застосований до кожного посилання у навігації.

Лістинг 3.1 – Реалізація навігації до компонента

```
<nav class="navbar">
  <a href="#button" class="navbar_snare">Buttons</a>
</nav>
<section id="button" class="section">
  <h2 class="title">Buttons</h2>
</section>
```

Кінець лістингу 3.1

Після підключення css-файлу до нашого html-документу, створюємо окремий файл _navbar.css, де будуть знаходитись стилі для навігації. Тепер підключимо їх до головного style.css, що показано на лістингу 3.2. Так само будуть підключатись усі файли.

Лістинг 3.2 – Підключення стилів компонентів до головного файлу стилів

```
@import url('parts/_main.css');
@import url('parts/_section.css');
@import url('parts/_navbar.css');
@import url('parts/_reset.css');
@import url('parts/_container.css');
@import url('parts/_title.css');
@import url('parts/_btn.css');
@import url('parts/_tooltip.css');
@import url('parts/_link.css');
@import url('parts/_dropdown.css');
```

```
@import url('parts/_checkbox.css');
@import url('parts/_switch.css');
@import url('parts/_field.css');
@import url('parts/_menubar.css');
@import url('parts/_modal.css');
@import url('parts/_card.css');
@import url('parts/_card-v2.css');
```

Кінець лістингу 3.2

Перш за все створимо контейнер в середину якого помістимо навігацію. Окрім переміщення за ідентифікатором, ще однією особливістю буде проста анімація зміни кольору. Вона реалізується за допомогою селектора «:hover», ця функція буде відповідати за зміну кольору при наведені на посилання.

Також для того, щоб наша панель не зникала разом з прокрутом, я застосував властивість «position» зі значенням «sticky», щоб навігаційна панель прокручувалась разом з вмістом сторінки.

Далі я буду по-порядку коротко описувати головні функції усіх стилізованих компонентів.

Переходимо до розділу Buttons, створимо новий файл із базовими стилями, які ще будуть використовуватись у проекті окрім даного розділу. Далі будуть представлені пояснення та основні стилі кнопки у лістингу 3.3.

Лістинг 3.3 – Базові стилі кнопки

```
/* Клас для користувацької кнопки */
.custom-btn {
  display: inline-block; /* Відображення кнопки як рядковий блок */
  font-weight: 700; /* Жирний шрифт */
  margin-right: 10px; /* Відступ праворуч */
  padding: 7px 34px; /* Внутрішні відступи */
  background-color: #29467F; /* Колір фону */
  color: #fff; /* Колір тексту */
  border-radius: 30px; /* Закруглені кути */
  line-height: 25px; /* Висота рядка */
  text-align: center; /* Вирівнювання тексту по центру */
  cursor: pointer; /* Зміна курсору при наведенні */
  transition: all 200ms ease-in-out; /* Плавний перехід для всіх
властивостей */
  user-select: none; /* Відключення виділення тексту */
```

```
}
/* Стиль для кнопки при наведенні */
.custom-btn:hover {
  background-color: #3967c3; /* Зміна кольору фону */
}
/* Стиль для кнопки при активному стані */
.custom-btn:active {
  transform: scale(.95); /* Зменшення розміру */
  box-shadow: inset 0 0 7px rgba(0, 0, 0, .4); /* Внутрішня тінь */
}
```

Кінець лістингу 3.3

Опис використання стилів, що показані на лістингу 3.3:

– новий селектор «active» визначає стилі для активного посилання, тобто при кліку;

– transform: scale (.95), ця властивість масштабує елемент до 70% від початкового розміру;

– box-shadow: inset 0 0 7px rgba (0, 0, 0, .4), застосовує внутрішню тінь до кнопки.

Перейдемо до стилізації перемикачів для розділу Checkbox, switch . У мене буде два ідентичних елемента перемикача, кожен з яких має div контейнер з класом «switch», «input» з елементом типу «checkbox» та «label» елемент. Елемент «label» асоціюється з «input» елементом за допомогою «for» атрибута. Далі на лістингу 3.4 буде показано структуру даних перемикачів.

Лістинг 3.4 – Структура перемикачів

```
<div>
  <div class="switch">
    <input type="checkbox" name="switch1" id="switch1">
    <label for="switch1" class="switch__label">Switch text</label>
  </div>
</div>
<div>
  <div class="switch">
    <input type="checkbox" name="switch2" id="switch2" checked
disabled>
    <label for="switch2" class="switch__label">Switch text</label>
  </div>
```

```
</div>
```

Кінець лістингу 3.4

Властивості перемикачів які представленоо на лістингу 3.5:

– «.switch input:checked ~ .switch__label:before» і «.switch input:checked ~ .switch__label:after» ці стилі використовуються для оновлення дизайну, коли перемикач позначено. Колір фону перемикача змінюється, а кнопка перемикання переміщується вправо;

– «.switch input:active ~ .switch__label:after» стиль що використовується для створення візуального ефекту, коли перемикач активний (тобто, коли користувач натискає на нього);

– «.switch input:checked:active ~ .switch__label:after» оновлює візуальний ефект, коли перемикач позначено й активний;

– «.switch input:disabled ~ .switch__label» показує вимкнення подій покажчика на мітці, коли введення вимкнено;

– «.switch input:disabled ~ .switch__label:before» та «.switch input:disabled ~ .switch__label:after» ці стилі оновлюють дизайн, коли введення вимкнено, через що перемикач виглядає сірим. Дані властивості представлені на лістингу 3.5.

Лістинг 3.5 – Стилізація перемикачів

```
/* Стиль для перемикача при вибраному стані */
.switch input:checked ~ .switch__label:before {
  background-color: #f7d00e; /* Колір фону переднього елементу при
вибраному стані */
}
/* Стиль для перемикача при вибраному стані */
.switch input:checked ~ .switch__label:after {
  left: 28px; /* Позиція рухомого елементу при вибраному стані */
}
/* Стиль для перемикача при активному стані */
.switch input:active ~ .switch__label:after {
  width: 40px; /* Ширина рухомого елементу при активному стані */
}

/* Стиль для перемикача при вибраному та активному стані */
```

```
.switch input:checked:active ~ .switch__label:after {
  left: 17px; /* Позиція рухомого елементу при вибраному та активному
стані */
}
/* Стиль для перемикача при вимкненому стані */
.switch input:disabled ~ .switch__label {
  pointer-events: none; /* Відключення взаємодії з користувачем */
}
/* Стиль для переднього елементу при вимкненому стані */
.switch input:disabled ~ .switch__label:before {
  opacity: .5; /* Прозорість переднього елементу */
}
/* Стиль для рухомого елементу при вимкненому стані */
.switch input:disabled ~ .switch__label:after {
  opacity: .5; /* Прозорість рухомого елементу */
}
```

Кінець лістингу 3.5

Розділ Menu, міститиме 2 види навігаційного меню. В обох випадках будуть однакові стилі, окрім кольору. Стилі написано в 1 файлі компонентів і щоб не створювати ще один, просто до стандартного меню додамо додатковий клас «menubar--light». Приклад реалізації показано на лістингу 3.6.

Лістинг 3.6 – Приклад зміни стилів з допомогою наслідування

```
.menubar--light {
  background-color: #fff;
}
.menubar--light .menubar__brand {
  color: #090a1a;
}
.menubar--light .menubar__nav-link {
  color: #090a1a;
}
```

Кінець лістингу 3.6

Наступний розділ Dropdown menu, в якому буде реалізована функція відкриття меню. Для початку підключимо до нашого документу скрипт та використаємо його класи у написанні спадного меню, як показано на лістингах 3.7-3.9.

Лістинг 3.7 – Підключення js-скрипта

```
<script src="https://cdn.jsdelivr.net/npm/easy-
toggler@2.2.7"></script>
```

Кінець лістингу 3.7

Після успішного підключення скрипта, необхідно інтегрувати його класи до нашого html-документу, це дозволить звертатися до стану компоненту та змінювати його властивості.

Лістинг 3.8 – Використання класів у HTML

```
<section id="dropdown" class="section">
      <h2 class="title">Dropdown menu</h2>
      <div class="dropdown">
          <a href="#" class="link" easy-toggle="#drop_def" easy-
          class="dropdown__drop--active" easy-rcoe>Dropdown menu</a>
        <div id="drop_def" class="dropdown__drop">
          <a href="#" class="dropdown__drop-item">First menu item</a>
          <a href="#" class="dropdown__drop-item">Second menu item</a>
          <a href="#" class="dropdown__drop-item">Third menu item</a>
        </div>
      </div>
     </section>
```

Кінець лістингу 3.8

Лістинг 3.9 – Стилізація класів у CSS

```
/* Стиль для випадаючого меню */
.dropdown__drop {
  display: flex; /* Використання флексбоксу для контейнера */
  flex-flow: column nowrap; /* Відображення елементів у колонку без
переносу */
  padding: 5px 0; /* Внутрішні відступи */
  position: absolute; /* Абсолютне позиціонування */
  width: 220px; /* Ширина контейнера */
  background: #082154; /* Колір фону */
  border-radius: 10px; /* Закруглені кути */
  top: 100%; /* Позиція зверху */
  transition: all 200ms ease-in-out; /* Плавний перехід для всіх
властивостей */
  z-index: -1; /* Рівень по осі Z */
  visibility: hidden; /* Невидимість контейнера */

  opacity: 0; /* Прозорість */
}
```

```
/* Стиль для активного стану випадаючого меню */
.dropdown__drop--active {
  visibility: visible; /* Видимість контейнера */
  opacity: 1; /* Непрозорість */
  z-index: 99; /* Рівень по осі Z */
  top: calc(100% + 7px); /* Позиція зверху з додатковим відступом */
}
```

Кінець лістингу 3.9

Для лістингу 3.9 буде описано стилізацію атрибуту:

– атрибути easy-toggle і easy-class, використовуються для перемикання видимості спадного меню, з допомогою властивостей «visibility», «opacity», «z-index» та додавання і видалення класів стилізації меню.

У розділі Input fields буде застосований генератор іконок Iconify, ми його приєднаємо та надамо просту стилізацію полям вводу, що показано на лістингу 3.10 та 3.11.

Лістинг 3.10 – Застосування атрибутів Iconify в HTML

```
<div class="field">
     <span class="iconify field__icon" data-icon="akar-
     icons:search"></span>
     <input type="text" class="field__control"
     placeholder="Search....">
</div>
```

Кінець лістингу 3.10

Лістинг 3.11 – приклад анімації при взаємодії з полями вводу

```
.field__control:focus {
  border-color: #001aff;
  background-color: #23243e;
  box-shadow: 0px 0px 13px rgba(0, 26, 255, 0.25);
}
```

Кінець лістингу 3.11

На лістингу 3.11 описані наступні властивості:

– якщо використати селектор «:focus» до класу «field__control» стилі будуть надаватись при фокусі на цей елемент;

– data-icon є атрибутом, який використовується бібліотекою iconify для визначення піктограми для відтворення.

Для створення модального вікна необхідно поєднати стилі кнопки та спадного меню, тобто принцип той самий, після кліку на кнопку вікно стає видимим. Дану стилізацію представлено на лістингу 3.12.

Лістинг 3.12 – використання атрибутів EasyToggler для модального вікна

```
<section id="modal" class="section">
      <h2 class="title">Modal window</h2>
      <button class="custom-btn" easy-toggle="#modalWin" easy-
class="modal--active">Modal window</button>
      <div id="modalWin" class="modal">
        <div class="modal__window">
          <h2 class="modal__window-title">Title windows</h2>
          <button class="moodal__window-close" easy-remove="#modalWin"
easy-class="modal--active">
            <span class="iconify" data-icon="eva:close-
outline"></span>
          </button>
          <div class="modal__body">
            <span class="modal__body-label">Any content</span>
          </div>
        </div>
        <div class="modal__overlay" remove="#modalWin" easy-
class="modal--active"></div>
      </div>
     </section>
```

Кінець лістингу 3.12

Далі розробимо найпростіший розділ Tooltip. Для цього помістимо у «span» текст з класом «link» та атрибутома «data-tooltip». Стилі першого відповідатимуть за текст, а «data-tooltip» міститиме стилі контейнера, який буде з’являтись при наводі на текст. Написання компонента-підказки показано на лістингу 3.13.

Лістинг 3.13 – Структура компонента-підказки

```
<section id="tooltip" class="section">
      <h2 class="title">Tooltips</h2>
```

```
      <span class="link" data-tooltip="Lorem ipsum dolor sit amet
consectetur adipisicing elit.">Підказка!</span>
     </section>
```

Кінець лістингу 3.13

Поясненя до лістингу 3.14:

– псевдоелемент «:before» буде використаний щоб створити віртуальний елемент, який міститиме вміст підказки. А «:hover» при наведенні зробить елемент з підказкою видимим, це видно на лістингу 3.14.

Лістинг 3.14 – Анімація появи компонента-підказки

```
/* Стиль для підказки при наведенні */
[data-tooltip]:hover:before {
  z-index: 100; /* Рівень по осі Z для переднього плану */
  opacity: 1; /* Непрозорість підказки */
  visibility: visible; /* Видимість підказки */
  left: calc(100% + 17px); /* Позиція зліва з додатковим відступом */
}
```

Кінець лістингу 3.14

В останньому розділі бібліотеки будуть знаходитись картки з інформацією. У контейнер помістимо заголовок, зображення, текст, кнопку і список, а з допомогою бібліотеки Iconify додамо і конки до нашого тексту. Далі буде написана структура у лістингу 3.15 виконана стилізація, результат якої видно на рисунку 3.2.

Лістинг 3.15 – структура картки

```
     <a href="#">
        <article class="card">
          <img src="img/front-end-840.jpg" alt="" class="card__image">
          <div class="card__body">
           <h3 class="card__title">Information library of ready-made
           solutions</h3>
           <ul class="card__parameters">
                <li><span class="iconify" data-icon="ant-design:eye-
                filled"></span> 7,4</li>
                <li><span class="iconify" data-icon="fluent:comment-
                12-filled"></span> 195</li>
```

```
            <li><span class="iconify" data-icon="ant-
            design:calendar-filled"></span>
            <time>24 July 2023, 14:00</time></li>
          </ul>
            <p class="card__text">Lorem ipsum dolor sit amet
            consectetur  adipisicing elit.
            Praesentium vel reiciendis atque accusamus. Ad harum,
            quod eligendi ullam illum est quo!
            </p>
          <span class="custom-btn">Read</span>
        </div>
      </article>
    </a>
```

Кінець лістингу 3.15

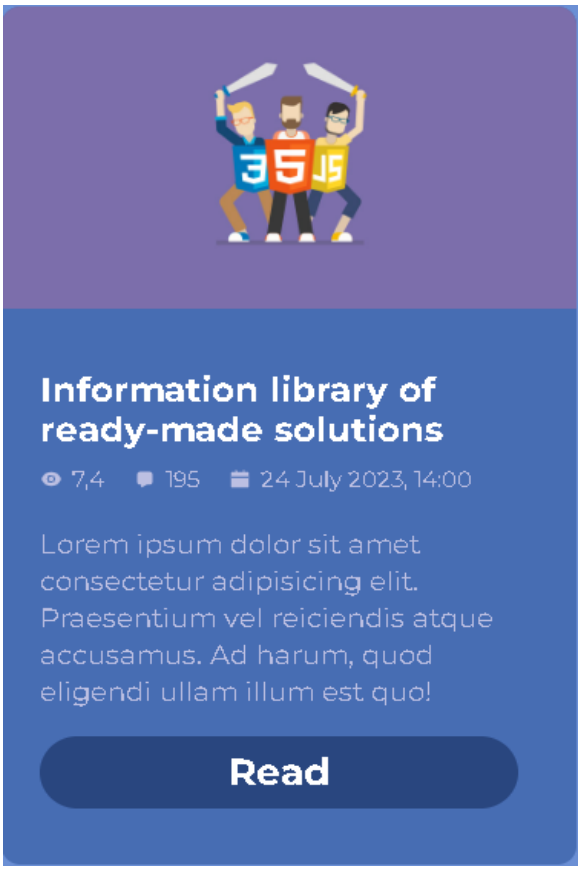

Рисунок 3.2 – Стилізована картка першого виду

Також у наступної картки змінімо стилі (зробимо її горизонтальною) та класи з «card» на «card-v2», та додамо збільшення розміру при наведенні властивістю «background-size: 120%». Результат видно на рисунку 3.3.

Рисунок 3.3 – Стилізована картка другого виду

3.2 Розробка документації для програмного продукту

Наступним що потрібно розробити є документація використання інформаційної бібліотеки, тобто файл README.md. Далі я буду описувати загальну структуру HTML та використання класів CSS розроблюваної бібліотеки. Отже, html-документ містить структуру з наступними розділами:

– <head>: Містить посилання на CSS, шрифти Google, іконки Iconify та скрипти;

– <body>: Включає контейнер з навігаційним меню <nav class="navbar"> та основним вмістом <main class="main">;

– <main>: Основний вміст сторінки, розділений на секції з унікальними ідентифікаторами, такими як <section id="button">, <section id="checkbox">, тощо.

Використання основних CSS класів:

– container – основний контейнер для всього вмісту;

– navbar – горизонтальне навігаційне меню;

– navbar_snare – посилання в навігаційному меню;

– main – контейнер для основного вмісту сторінки;

– section – розділ сторінки;

– title – заголовок розділу;

a) класи, які використовуються для кнопок:
   – custom-btn – базовий клас для кнопки;
   – btn-1, btn-2, btn-3, btn-4 – варіанти стилів кнопок;
b) класи, які використовуються для перемикачів і чекбоксів:
   – switch – контейнер для перемикача;
   – switch__label – мітка для перемикача;
   – checkbox – контейнер для чекбокса;
   – checkbox__label – мітка для чекбокса;
c) класи, які використовуються для меню:
   – menubar – горизонтальне меню;
   – menubar__brand – логотип або назва бренду в меню;
   – menubar__nav – список навігаційних елементів;
   – menubar__nav-link – посилання в навігаційному меню;
   – menubar—light – варіант світлого стилю меню;
d) класи, які використовуються для спадного меню:
   – dropdown – контейнер для спадного меню;
   – dropdown__drop – спадне меню;
   – dropdown__drop-item – елемент спадного меню.
e) класи, які використовуються для полів вводу:
   – field – контейнер для поля вводу;
   – field__control – поле для вводу;
   – field__icon – іконка всередині поля вводу;
f) класи, які використовуються для модальних вікон:
   – modal – контейнер для модального вікна;
   – modal__window –модальне вікно;
   – modal__window-title – заголовок модального вікна;
   – modal__window-close – кнопка закриття модального вікна;
   – modal__body – тіло модального вікна;
   – modal__body-label – мітка в тілі модального вікна.

– modal__overlay – накладений шар, який затемнює фоновий вміст під час відкритого модального вікна;

g) класи, які використовуються для підказок:

– [data-tooltip] – атрибут для елементів, які мають підказки;

h) класи, які використовуються для карток:

– card – картка з зображенням і текстовим вмістом;

– card__image – зображення в картці;

– card__body – тіло картки з текстом і кнопкою;

– card__title – заголовок картки;

– card__parameters – список параметрів картки;

– card__text – описний текст картки;

– card-v2 – альтернативний варіант картки з фоновим зображенням;

– card-v2__body – тіло альтернативної картки;

Інструкції з використання класів компонентів бібліотеки:

1) кнопки – використовуйте клас .custom-btn для створення кнопки, а класи btn-1, btn-2, btn-3, btn-4 для різних стилів кнопок;
2) перемикачі і чекбокси – для створення перемикача використовуйте структуру HTML з класом switch, а для чекбокса – клас checkbox;
3) меню: – створіть горизонтальне меню, використовуючи клас menubar. Для світлого варіанту додайте модифікатор menubar--light;
4) спадне меню – створити спадне меню можна використовуючи клас dropdown для контейнера і dropdown__drop для випадаючого списку;
5) поля вводу – для полів вводу використовуйте клас field як контейнер і field__control для самого поля вводу;
6) модальні вікна – створюйте модальні вікна, використовуючи клас modal як контейнер, modal__window для вікна і modal__overlay для затемнення фону;
7) підказки – додайте атрибут data-tooltip з текстом підказки до будь-якого елемента, щоб відображати підказку при наведенні курсора;

8) картки – створити картки, використовуючи клас card або card-v2 для альтернативного варіанту.

3.3 Тестування та налагодження інформаційно-комп'ютерної системи

Щоб наш програмний продукт працював належним чином необхідно протестувати його за допомогою спеціальних інструментів. Для початку розглянемо найпоширеніші помилки, а саме:

– проблеми з адаптацією до різних браузерів;

– помилки валідації форм;

– несумісність макета сторінки.

Для тестування я обрав наступний інфтрумент. BrowserStack – це платформа, що надає доступ до великої кількості пристроїв і браузерів для тестування, що дозволяє розробникам переконатися в коректному функціонуванні веб-сайтів та мобільних додатків на різних платформах.

Спершу я перевірю як відображаються стилі на різних браузерах у застосунку Browser Stack. Для цього я виберу, наприклад, операційну систему Windows та браузер Google, далі у полі введення URL напишу локальну адресу. Приклад показано на рисунку 3.4.

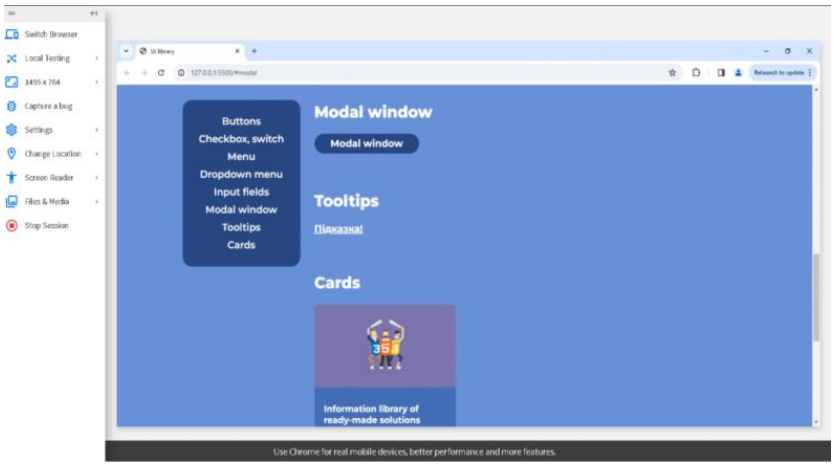

Рисунок 3.4 – Тестування у застосунку Browser Stack

На рисунку 3.5 показано:

– панель для вибору операційних систем та їх версії;

– панель для вибору браузерів та їх версії.

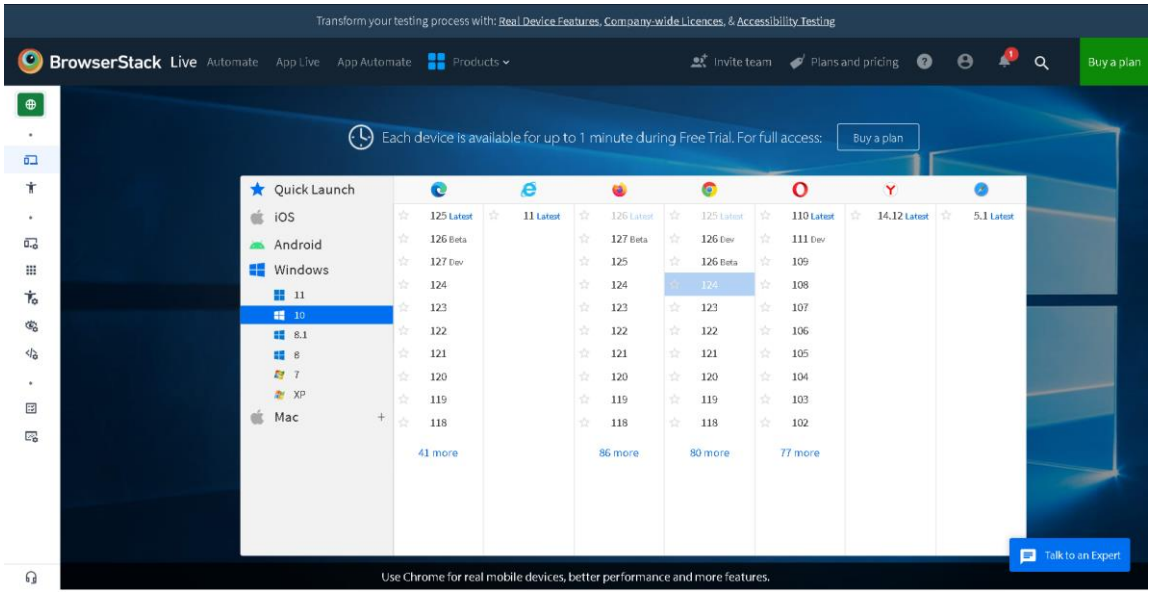

Рисунок 3.5 – Середовище тестування Browser Stack

Також, у самому браузері було протестована сторінка з допомогою стандартного інструмента Inspect Element. Далі описано функції які будуть застосовані для тестування на рисунку 3.6.

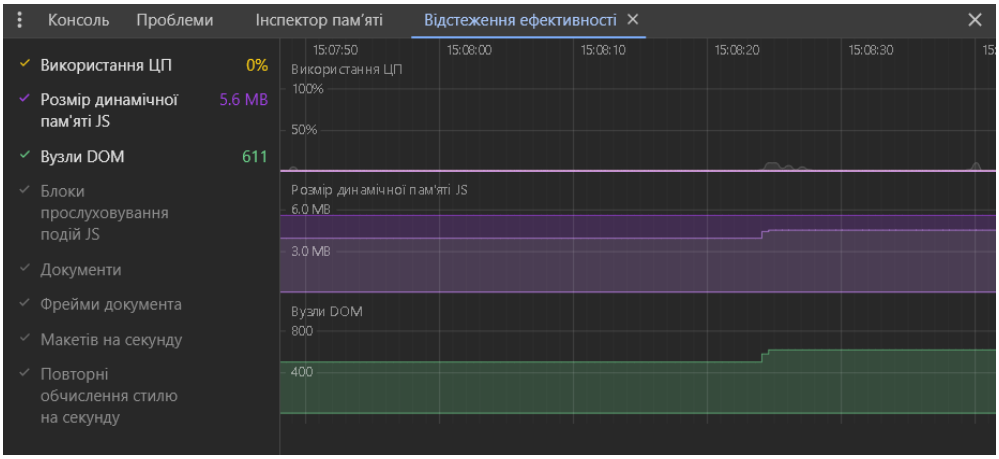

Рисунок 3.6 – Інструмент тестування Inspect Element

Inspect Element дозволяє відслідковувати та аналізувати ефективність виконання коду. У лівій частині показано елементи, що використовуються для моніторингу, а саме:

– використання ЦП – показує відсоток використання процесорного часу програмою;

– розмір динамічної пам’яті JS – показує обсяг пам‘яті, що використовується для виконання коду JavaScript;

– вузли DOM – показує кількість елементів, що складають веб-сторінку, які створюються і управляються JavaScript кодом.

У правій частині розташований графік, що відображає динаміку використання пам’яті та інших показників за певний проміжок часу. Нижче розташовані лінії, що показують динаміку інших параметрів, таких як кількість вузлів DOM, кількість подій, що обробляються, та інші. Це допомагає зрозуміти, як змінюються ці показники з часом, і визначити, які саме компоненти програми можуть викликати проблеми з ефективністю.

3.4 Завантаження на GitHub

Останнім кроком у розробці нашої інформаційної бібліотеки стане публікація в репозиторію GitHub. Переходимо до його створення, ось кроки які потрібно виконати:

– увійти до свого облікового запису;

– щоб створити новий репозиторій, потрібно натиснути кнопку «New repository»;

– ввести назву репозиторію у полі «Repository name»;

– коротко описати що містить репозиторій;

– обрати публічний чи закритий доступ;

– додати файл README;

– підтвердити створення репозиторію.

Далі, після створення репозиторію на GitHub потрібно ініціалізувати локальний репозиторій на комп'ютері та пов'язати його з репозиторієм на GitHub. Для цього відкриємо термінал та введомо команди що показані на лістингу 3.16.

Лістинг 3.16 – Підключення до репозиторію

```
git init
git add .
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/aomine228/nn.git
git push -u origin main
```

Кінець лістингу 3.16

На лістингу 3.16 показано команд, такі як:

– git init – ініціалізує новий локальний репозиторій Git у поточній директорії. Це створює піддиректорію .git, де зберігаються всі необхідні файли для контролю за версіями;

– git add – додає всі зміни, внесені у файли в поточній робочій директорії, до індексу, що означає, що всі нові, змінені файли будуть включені в наступний коміт;

– git commit -m "first commit" – створює новий коміт, що фіксує зміни, додані до індексу, у локальний репозиторій;

– git branch -M main – перейменовує поточну гілку з ім'ям master на main. Параметр -M використовується для примусового перейменування, навіть якщо гілка з ім'ям main вже існує;

– git remote add origin https://github.com/aomine228/nn.git – додає новий віддалений репозиторій з ім'ям origin із зазначеним URL. Це повідомляє Git, де знаходиться віддалений репозиторій, з яким ви синхронізуєте свій локальний репозиторій;

– git push -u origin main – відправляє зміни з локальної гілки main на віддалений репозиторій origin. Параметр «u» встановлює віддалений

репозиторій і гілку для поточного локального репозиторію, що дозволить у майбутньому використати команду git push без необхідності вказувати віддалений репозиторій і гілку.

Висновок до розділу 3

У цьому розділі було описано процес розробки інформаційної бібліотеки та показано яким чином встановлюються необхідні інструменти, які потрібні для подальшої розробки застосунку.

Показано принцип роботи кожного компонента. Представлено дизайн бібліотеки та описано головні елементи. Було реалізовано та протестовано верстку компонентів інтерфейсу, а також було показано процес завантаження готової бібліотеки на платформу GitHub.

ВИСНОВКИ

В даній роботі було розроблено інформаційну бібліотеку готових рішень CSS і HTML для розробників інтерфейсів користувача.

У поставленому завданні було детально розглянуто веб-технології, проаналізовано популярні бібліотеки та описано їхні переваги та недоліки. Для реалізації інформаційної бібліотеки обрано HTML, CSS та бібліотеку на JavaScript та обґрунтовано вибір середовища для розробки Visual Studio Code

В результаті дослідження моделей життєвого циклу було обрано гнучку модель через її перевагу у швидкості розробки. Вибрані методології (БЕМ та модульний підхід) написання коду були ретельно досліджені та успішно інтегровані у процес розробки бібліотеки. Їхні методи оптимізації піходять для написання інформаційної бібліотеки. Спроектована UML-діаграма показує можливі варіанти взаємодії з даною бібліотекою, як користувачу так і розробнику. Також були досліджені основні компоненти веб-додатків на основі сторінок Google Forms та бібліотеки Bootstrap, розроблені макети компонентів та бібліотеки вцілому.

Був описаний процес розробки програмного забезпечення, де пояснюється кожна функція окремого компонента інформаційної бібліотеки. Для полегшення розробки функціональних компонентів було інтегровано бібліотеку Easy-toggler, що написана на JavaScript. Вибір досліджуваних інструментів тестування Inspect Element та Browser Stack дозволив перевірити клієнтську частину на коректне відображення компонентів у браузері та їхні модифікації на різних операційних системах. Також було реалізовано документаціяю, де пояснюється використання кожного компонента та опис усіх класів. Процес вивантаження інформаційної бібліотеки реалізовано за допомогою платформи GitHub тим самим це надає розробникам доступ до коду, його редагування та впровадження.

Отриманий програмний продукт є простим в освоєнні, а його зрозуміла та чітко визначена структура допоможе недосвідченим розробникам покращити свої навички у веб-розробці.

Даний проект може бути розвинений до повноцінної бібліотеки, так як за допомогою користувачів, які можуть безпосередньо впливати на процес розробки та самостійно покращувати його.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Codropes. URL: https://tympanus.net/codrops/category/tutorials/ (дата звернення: 04.03.2024).

2. CSS Styling the web. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS (дата звернення: 01.03.2024).

3. Google Form. URL: https://docs.google.com/forms/d/1evzYwU9mpLQK0RARWylATVmCvzQwGwG1yjkuF-QHAhk/edit?hl=uk (дата звернення: 20.05.2024).

4. Structuring the web with HTML. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 01.03.2024).

5. The Fundamental Elements of Web Development. URL: https://www.linkedin.com/pulse/fundamental-elements-web-development-html-css-anass-essadikine-byn7e (дата звернення: 05.03.2024).

6. TypeScript. URL: https://en.wikipedia.org/wiki/TypeScript (дата звернення: 17.04.2024).

7. Веб-сайт бібліотеки Bootstrap. URL: https://getbootstrap.com (дата звернення: 20.04.2024).

8. Документація GitHub та його використання для спільного розроблення проектів. URL: docs.github.com (дата звернення: 20.05.2024).

9. Документація Visual Studio Code. URL: code.visualstudio.com/docs (дата звернення: 27.02.2024).

10. Методологія БЕМ в дії. URL: https://avivi.pro/ua/blog/metodologiya-bem-v-deystvii/ (дата звернення: 22.04.2024).

11. Основні вимоги до Junior Frontend Developer. URL: https://foxminded.ua/shcho-maie-znaty-junior-frontend/ (дата звернення: 16.02.2024).

12. Статистика бібліотек. URL: https://trends. google.com.ua /trends/explore?geo=UA&q=%2Fm%2F0j671ln,React%20js,%2Fm%2F0268gyp&hl=uk (дата звернення: 17.04.2024).

ДОДАТКИ

Додаток А

Лістинг 1 – Структура компонентів бібліотеки

```
<!DOCTYPE html>
<html lang="en">
<head>
<!-- Встановлення кодування символів для документа -->
<meta charset="UTF-8">
<!-- Встановлення параметрів для адаптивного дизайну -->
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<!-- Підключення зовнішнього файлу стилів -->
<link rel="stylesheet" href="./css/style.css">
<!-- Підключення до сервісу шрифтів Google -->
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com"
crossorigin>
<!-- Підключення шрифту Montserrat з Google Fonts -->
<link
href="https://fonts.googleapis.com/css2?family=Montserrat:ital,wg
ht@0,100..900;1,100..900&display=swap" rel="stylesheet">
<!-- Підключення бібліотеки Easy Toggler для керування класами
CSS -->
<script src="https://cdn.jsdelivr.net/npm/easy-
toggler@2.2.7"></script>
<!-- Підключення бібліотеки Iconify для використання іконок -->
<script
src="https://code.iconify.design/2/2.2.1/iconify.min.js"></script
>
<!-- Встановлення заголовка документа -->
<title>Ui library</title>
</head>
<body>

<div class="container">

<div>
<!-- Навігаційна панель -->
<nav class="navbar">
<!-- Посилання на розділи сторінки -->
<a href="#button" class="navbar_snare">Buttons</a>
<a href="#checkbox" class="navbar_snare">Checkbox, switch</a>
<a href="#menu" class="navbar_snare">Menu</a>
<a href="#dropdown" class="navbar_snare">Dropdown menu</a>
<a href="#input" class="navbar_snare">Input fields</a>
<a href="#modal" class="navbar_snare">Modal window</a>
<a href="#tooltip" class="navbar_snare">Tooltips</a>
<a href="#card" class="navbar_snare">Cards</a>
</nav>
</div>
```

```
<main class="main">

<!-- Розділ з кнопками -->
<section id="button" class="section">
<!-- Заголовок розділу -->
<h2 class="title">Buttons</h2>
<!-- Кнопки з різними стилями -->
<button class="custom-btn">Button</button>
<button class="custom-btn btn-1">Button</button>
<button class="custom-btn btn-2"><span>Button</span></button>
<button class="custom-btn btn-3">Button</button>
<button class="custom-btn btn-4">Button</button>
</section>

<!-- Розділ з чекбоксами та перемикачами -->
<section id="checkbox" class="section">
<!-- Заголовок розділу -->
<h2 class="title">Checkbox, switch</h2>
<!-- Перемикач -->
<div>
<div class="switch">
<input type="checkbox" name="switch1" id="switch1">
<label for="switch1" class="switch__label">Switch text</label>
</div>
</div>

<!-- Перемикач, заблокований і відзначений -->
<div>
<div class="switch">
<input type="checkbox" name="switch2" id="switch2" checked
disabled>
<label for="switch2" class="switch__label">Switch text</label>
</div>
</div>
<!-- Чекбокс -->
<div>
<div class="checkbox">
<input type="checkbox" name="check1" id="check1">
<label for="check1" class="checkbox__label">Checkbox text</label>
</div>
</div>
</section>

<!-- Розділ з меню -->
<section id="menu" class="section">
<!-- Заголовок розділу -->
<h2 class="title">Menu</h2>
<!-- Темне меню -->
<nav class="menubar">
<!-- Логотип -->
```

```
<a href="#" class="menubar__brand">Logo</a>
<!-- Список посилань у меню -->
<ul class="menubar__nav">
<li><a href="#" class="menubar__nav-link">About us</a></li>
<li><a href="#" class="menubar__nav-link">Services</a></li>
<li><a href="#" class="menubar__nav-link">Reviews</a></li>
<li><a href="#" class="menubar__nav-link">Cases</a></li>
<li><a href="#" class="menubar__nav-link">Contacts</a></li>
</ul>
</nav>

<!-- Світле меню -->
<nav class="menubar menubar--light">
<!-- Логотип -->
<a href="#" class="menubar__brand">Logo</a>
<!-- Список посилань у меню -->
<ul class="menubar__nav">
<li><a href="#" class="menubar__nav-link">About us</a></li>
<li><a href="#" class="menubar__nav-link">Services</a></li>
<li><a href="#" class="menubar__nav-link">Reviews</a></li>
<li><a href="#" class="menubar__nav-link">Cases</a></li>
<li><a href="#" class="menubar__nav-link">Contacts</a></li>
</ul>
</nav>
</section>

<!-- Розділ з випадаючим меню -->
<section id="dropdown" class="section">
<!-- Заголовок розділу -->
<h2 class="title">Dropdown menu</h2>
<!-- Випадаюче меню -->
<div class="dropdown">
<a href="#" class="link" easy-toggle="#drop_def" easy-
class="dropdown__drop--active" easy-rcoe>Dropdown menu</a>
<div id="drop_def" class="dropdown__drop">
<a href="#" class="dropdown__drop-item">First menu item</a>
<a href="#" class="dropdown__drop-item">Second menu item</a>
<a href="#" class="dropdown__drop-item">Third menu item</a>
</div>
</div>
</section>

<!-- Розділ з полями вводу -->
<section id="input" class="section">
<!-- Заголовок розділу -->
<h2 class="title">Input fields</h2>
<!-- Поля вводу -->
<div style="width: 440px;">
<div class="field">
<input type="text" class="field__control" placeholder="Enter your
name">
```

```
</div>
<div class="field">
<span class="iconify field__icon" data-icon="akar-
icons:search"></span>
<input type="text" class="field__control"
placeholder="Search....">
</div>
</div>
</section>

<!-- Розділ з модальними вікнами -->
<section id="modal" class="section">
<!-- Заголовок розділу -->
<h2 class="title">Modal window</h2>
<!-- Кнопка для виклику модального вікна -->
<button class="custom-btn" easy-toggle="#modalWin" easy-
class="modal--active">Modal window</button>
<!-- Модальне вікно -->
<div id="modalWin" class="modal">
<div class="modal__window">
<h2 class="modal__window-title">Title windows</h2>
<!-- Кнопка для закриття модального вікна -->
<button class="moodal__window-close" easy-remove="#modalWin"
easy-class="modal--active">
<span class="iconify" data-icon="eva:close-outline"></span>
</button>
<div class="modal__body">
<span class="modal__body-label">Any content</span>
</div>
</div>
<div class="modal__overlay" remove="#modalWin" easy-class="modal-
-active"></div>
</div>
</section>

<!-- Розділ з підказками -->
<section id="tooltip" class="section">
<!-- Заголовок розділу -->
<h2 class="title">Tooltips</h2>
<!-- Підказка -->
<span class="link" data-tooltip="Lorem ipsum dolor sit amet
consectetur adipisicing elit.">Підказка!</span>
</section>

<!-- Розділ з картками -->
<section id="card" class="section">
<!-- Заголовок розділу -->
<h2 class="title">Cards</h2>
<!-- Картка першого типу -->
<a href="#">
<article class="card">
```

```
<img src="img/front-end-840.jpg" alt="" class="card__image">
<div class="card__body">
<h3 class="card__title">Information library of ready-made
solutions</h3>
<ul class="card__parameters">
<li><span class="iconify" data-icon="ant-design:eye-
filled"></span> 7,4</li>
<li><span class="iconify" data-icon="fluent:comment-12-
filled"></span> 195</li>
<li>
<span class="iconify" data-icon="ant-design:calendar-
filled"></span>
<time>24 July 2023, 14:00</time>
</li>
</ul>
<p class="card__text">Lorem ipsum dolor sit amet consectetur
adipisicing elit.
Praesentium vel reiciendis atque accusamus. Ad harum, quod
eligendi ullam illum est quo!
</p>
<span class="custom-btn">Read</span>
</div>
</article>
</a>
<br></br>

<!-- Картка другого типу -->
<a href="">
<article class="card-v2" style="background-image: url(/img/front-
end-840.jpg);">
<div class="card-v2__body">
<h3 class="card-v2__title">Information library of ready-made
solutions</h3>
<ul class="card-v2__parameters">
<li><span class="iconify" data-icon="ant-design:eye-
filled"></span> 7,4k</li>
<li><span class="iconify" data-icon="fluent:comment-12-
filled"></span> 195</li>
<li>
<span class="iconify" data-icon="ant-design:calendar-
filled"></span>
<time>24 July 2023, 14:00</time>
</li>
</ul>
</div>
</article>
</a>
</section>
</main>
</div>
```

```
<!-- Підключення бібліотеки Easy Toggler -->
<script src="libs/easy-toggler.min.js"></script>
</body>
</html>
```

Кінець лістингу 1