Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка Progressive Web Apps

Виконав(ла): студент(ка) 4 курсу, групи СП-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

		Анастюк Д. Є.
	(підпис)	(прізвище та ініціали)
Керівник		Цуприк Г. Б.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Петрик М. Р.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра
(повна назва кафедри)

		ЗАТВЕРДЖУЮ		
		Завідувач кафедри		
		(підпис)		(прізвище та ініціали)
		«__» _ _ 2023 р.		

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Студенту Анастюку Даніілу Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка Progressive Web Apps

Керівник роботи Прізвище Ім'я По батькові, науковий ступінь, посада кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7-13Х

2. Термін подання студентом завершеної роботи Дата захисту 2023р.

3. Вихідні дані до роботи Вихідні дані

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Перелік елементів змісту (З номерами, через крапку. Примітка: Не достатньо вказати тільки назви розділів!). 3. Безпека життєдіяльності, основи хорони праці. Висновки. Перелік джерел. Додатки (якщо такі є).

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Повний перелік слайдів у презентації, включаючи перший та останній слайди (з номерами, через крапку).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	згідно наказу, уточнити при отриманні завдання	05.06.2023	08.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про …	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо …	01.02.2023-07.02.2023	*Виконано*
	Розроблення …		
5.	Оформлення розділу «	08.02.2023-09.02.2023	*Виконано*
	»		
6.	Оформлення розділу «	10.02.2023-12.02.2023	*Виконано*
	»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2023-06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	07.06.2023-08.06.2023	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2023	

Студент _____ Анастюк. Д.Є.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Цуприк Г.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Проект на тему «Розробка Progressive Web Apps».

Анастюк Даніїл Євгенович. Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СП–41, Тернопіль – 2023.С. – 77, рисунків – 25, додатків – 26.

Мета проекту: розробка Progressive Web Apps (PWA) – веб-додатків, які мають можливості мобільних додатків. Головною метою PWA є забезпечення швидкої та зручної роботи на будь-яких пристроях та платформах.

PWA – це веб-додатки, які можна використовувати на будь-яких пристроях з доступом до Інтернету, що мають можливості мобільних додатків. Їх головною перевагою є те, що вони можуть працювати в офлайні та забезпечувати швидку та зручну роботу.

Для досягнення мети проекту необхідно виконати наступні завдання:

- Ознайомитися з основними характеристиками PWA та їх перевагами;
- Дослідити можливості розробки PWA з використанням сучасних інструментів та технологій;
- Створити прототип PWA, який буде підтримувати роботу в офлайні та має відповідати вимогам до швидкості та продуктивності;
- Протестувати розроблений прототип та внести необхідні зміни для поліпшення його роботи та забезпечення відповідності вимогам.

Успішне виконання проекту дозволить розробити веб-додаток, який буде працювати на будь-яких платформах та пристроях, забезпечує швидку та зручну роботу, а також можливість роботи в офлайні.

ANNOTATION

Project on " Development progressive web app".

Anastiuk Daniil Yevhenovych. Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, SP – 41 Group, Ternopil – 2023.C. – 77, drawings – 25, applications – 26.

The purpose of the project: development of Progressive Web Apps (PWA) - web applications that have the capabilities of mobile applications. The main goal of PWA is to provide fast and convenient work on any devices and platforms.

PWAs are web apps that can be used on any internet-enabled device that has mobile app capabilities. Their main advantage is that they can work offline and provide fast and convenient work.

To achieve the goal of the project, the following tasks must be completed:

Familiarize yourself with the main characteristics of PWAs and their advantages;

Explore the possibilities of PWA development using modern tools and technologies;

Create a PWA prototype that will support offline work and must meet speed and performance requirements;

Test the developed prototype and make the necessary changes to improve its performance and ensure compliance with the requirements.

Successful implementation of the project will allow the development of a web application that will work on any platforms and devices, provides fast and convenient work, as well as the ability to work offline. Such a web application can be useful for various companies and organizations, as it allows for more convenient and faster access to information, and can also be a competitive advantage compared to a regular web application.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

GIT – це система контролю версій (VCS), яка дозволяє відстежувати та фіксувати зміни в коді: ви можете відновити код у разі збою або відкотити до попередніх версій.

SPA – це тип веб-застосунків, у яких завантаження необхідного коду відбувається на одну сторінку. Це дозволяє заощадити час на повторне завантаження тих самих елементів.

Framework – це програмне середовище, яке спрощує та прискорює створення програмного забезпечення. За використання фреймворків ви пишете лише код, який реалізує логіку, специфічну для вашого продукту. Вам не доводиться самостійно забезпечувати роботу з базою даних, автентифікацію, підтримку сеансів тощо. Все це реалізовано у фреймворках.

React – JavaScript-бібліотека, яка застосовується для розробки UI (призначених для користувача інтерфейсів), та відповідає за те, як ваш web-додаток буде виглядати.

Observer – це поведінковий патерн проектування, який створює механізм підписки, що дає змогу одним об’єктам стежити й реагувати на події, які відбуваються в інших об’єктах.

GraphQL – мова запитів даних та мова маніпулювання даними з відкритим вихідним кодом для побудови веб-орієнтованих програмних інтерфейсів. GraphQL був розроблений як внутрішній проект компанії Facebook у 2012 році, а пізніше у 2015 році був випущений публічно.

ЗМІСТ

ВСТУП

Останні роки відзначаються значним розширенням та розвитком веб-додатків, які забезпечують користувачів зручними та інноваційними функціями. Однією з важливих тенденцій у сфері веб-розробки є концепція Progressive Web Apps (PWA).

Progressive Web Apps поєднують в собі найкращі характеристики веб-сайтів та мобільних додатків. Основною ідеєю Progressive Web App є створення додатків, які можна встановити на домашній екран мобільного пристрою, подібно до звичайних мобільних додатків, але з використанням веб-технологій.

Розробка Progressive Web Apps відкриває широкі перспективи для підприємств та розробників. Вони дозволяють створювати масштабовані та інтерактивні веб-додатки з високою продуктивністю, які працюють на різних платформах та пристроях. Завдяки можливостям Progressive Web App, підприємства можуть поліпшити користувацький досвід, забезпечити офлайн-режим роботи, повідомлення та інші функції, які раніше були доступні лише у мобільних додатках. Основними принципами Progressive Web Apps є прогресивність, респонзивність, незалежність від мережі та апдейтів, можливість установки на домашній екран. Ці функції дозволяють додаткам працювати як звичайні веб-сторінки, але зберігати властивості традиційних мобільних додатків.

Однією з переваг Progressive Web Apps є їх швидкодія та висока продуктивність. Завдяки кешуванню та офлайн-режиму, користувачі можуть продовжувати використовувати додаток навіть при відсутності Інтернет-з'єднання. Крім того, PWA дозволяють використовувати сповіщення, доступ до камери та інших пристроїв, що дозволяє розширити можливості взаємодії з користувачем.

Метою даної дипломної роботи є дослідження та розробка Progressive Web Apps з використанням сучасних веб-технологій. Дослідження цієї теми дозволить

нам краще зрозуміти потенціал Progressive Web Apps і їх вплив на майбутнє веб-розробки та користувацького досвіду.

1. ПОГЛЯД ПЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНІВ

1.1. Огляд конкурентів

Огляд конкурентів на тему дипломної роботи "Розробка Progressive Web Apps"

У рамках моєї дипломної роботи з розробки Progressive Web Apps (Progressive Web App) я провів аналіз ринку та вибрав кілька прикладів конкурентів, які будуть розглянуті нижче.

Магазин спортивного спорядження "Athletics": Перший конкурент, який привернув мою увагу, - це магазин спортивного спорядження "Athletics". Інтерфейс користувача цього сайту є привабливим та інтуїтивно зрозумілим. Користувачам легко знаходити потрібні товари і здійснювати покупки. Магазин пропонує широкий асортимент спортивного спорядження для різних видів активності. Однак, відсутність підтримки різних мов та обмежені можливості вибору товарів, які не стосуються спорту, можуть бути недоліками цього конкурента.

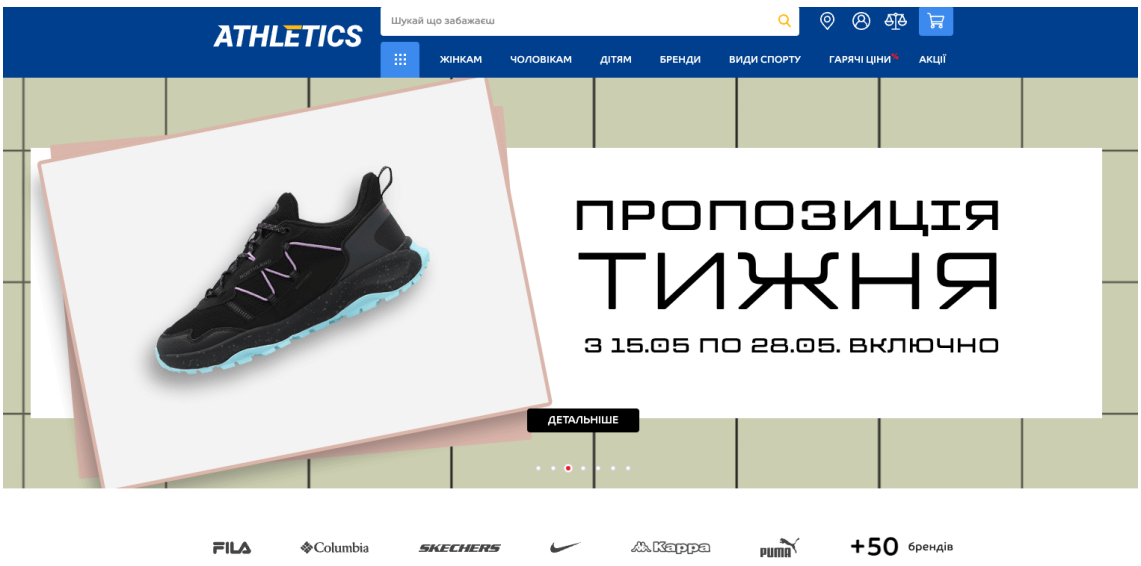


Рисунок 1.1 – Інтернет-магазин «Athletics»

Музичний магазин "SoundWave": Інший конкурент, який заслуговує уваги, - це музичний магазин "SoundWave". Цей сайт має стильний дизайн та пропонує широкий вибір музичних інструментів і записів. Він забезпечує зручну навігацію та можливість прослуховувати демо-версії треків перед покупкою. Однак, недоліком може бути обмежений функціонал магазину і відсутність можливості змінювати атрибути продуктів. Крім того, магазин не має підтримки для різних мов, що може обмежити його міжнародний потенціал.

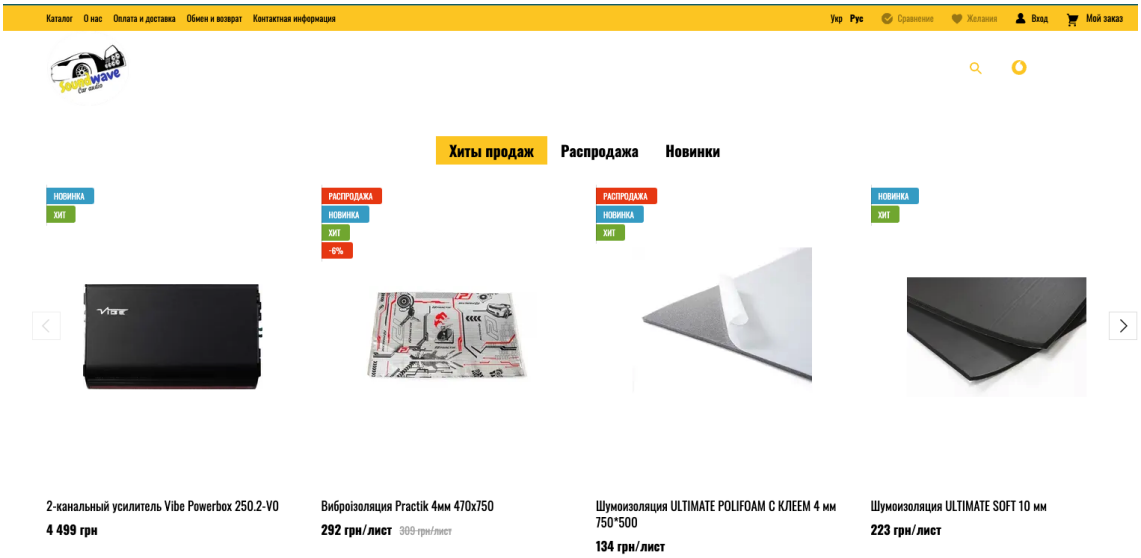


Рисунок 1.2 – Інтернет-магазин «SoundWave»

Маркетплейс рукоділля "CraftHub": Третім конкурентом, який варто розглянути, є маркетплейс рукоділля "CraftHub". Цей сайт спеціалізується на продажі унікальних ручних виробів та рукодільних матеріалів. Він надає можливість як покупцям, так і продавцям зареєструватися та створити власні магазини. Користувачі можуть легко переглядати і придбавати товари, а також спілкуватися з продавцями через систему повідомлень. Однак, недоліком може бути обмежена кількість товарів та відсутність розширених можливостей пошуку та фільтрації.

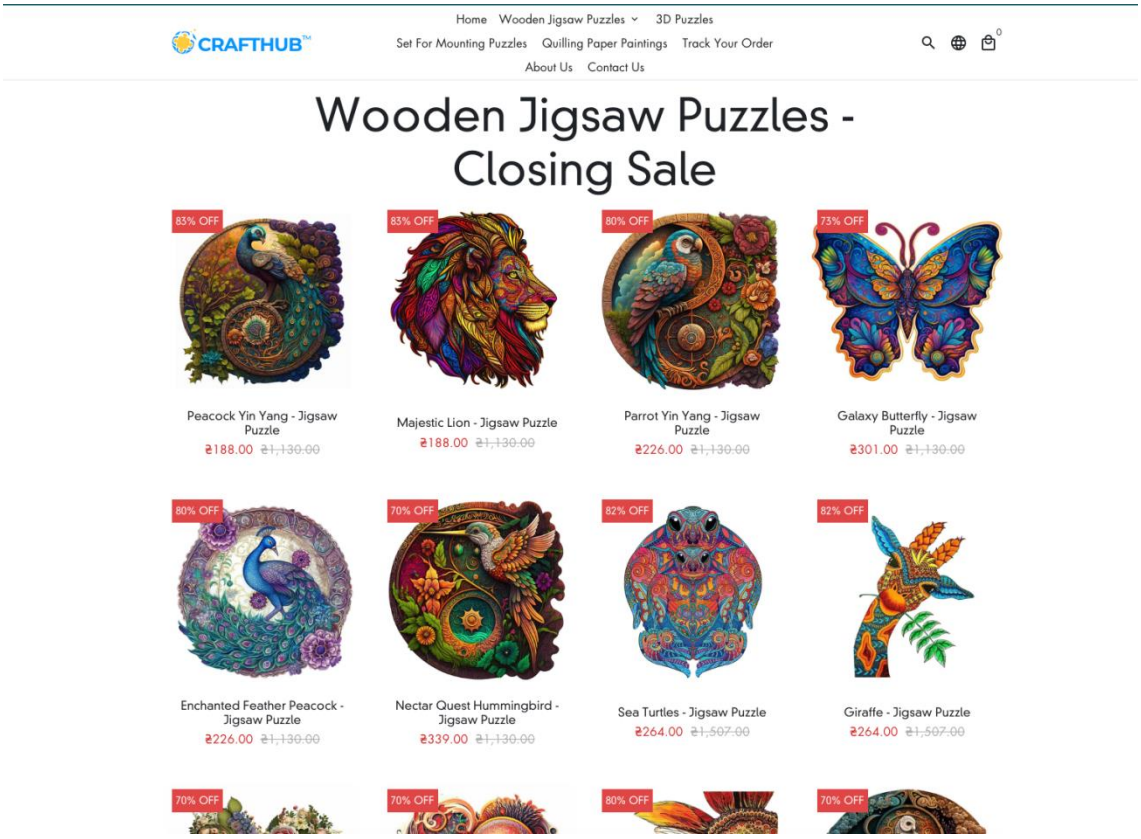


Рисунок 1.3 – Інтернет-магазин «CraftHub»

У своїй роботі я планую врахувати переваги та недоліки вищезгаданих конкурентів, а також розробити функціонал, який відповідатиме потребам користувачів та забезпечить покращену користувацьку досвід в моєму Progressive Web App. Буду працювати над створенням зручного інтерфейсу, розширеними можливостями пошуку і фільтрації товарів, а також підтримкою різних мов для максимальної доступності та зручності користувачів.

1.2. Обґрунтування вибору напрямку дослідження

У нашому сучасному цифровому світі, де мобільні пристрої відіграють важливу роль, розробка веб-додатків, які працюють швидко та ефективно на різних платформах, стає все більш актуальною. Тому, для мого дослідження, я вибрав напрямок розробки Progressive Web Apps (Progressive Web App).

Переваги Progressive Web Apps полягають у тому, що вони комбінують найкращі характеристики веб-сайтів і мобільних додатків. Основні фактори, які обумовили вибір даного напрямку дослідження, включають:

- Кросплатформеність: Progressive Web App можуть працювати на різних платформах та пристроях, таких як комп'ютери, планшети та смартфони, незалежно від операційної системи. Це дозволяє забезпечити однаковий функціонал та користувацький досвід на різних пристроях.
- Офлайн-режим: Progressive Web App можуть працювати в офлайн-режимі, забезпечуючи користувачам доступ до контенту та функціоналу, навіть коли вони не підключені до Інтернету. Це дуже важливо для поліпшення користувацького досвіду та забезпечення постійного доступу до інформації.
- Швидкодія: Progressive Web App використовують передові технології, такі як кешування та сервісні робітники, що дозволяють прискорити завантаження та відгук веб-додатків. Це особливо важливо для користувачів, які мають обмежений доступ до Інтернету або використовують повільні з'єднання.
- Налагодженість: Розробка Progressive Web App відбувається з використанням стандартних веб-технологій, таких як HTML, CSS та JavaScript. Це дозволяє розробникам використовувати вже існуючі навички та інструменти для створення і налагодження додатків.
- Масштабованість: Progressive Web App дозволяють поетапно розширювати функціонал додатків, не потребуючи завантаження та встановлення оновлень, як це відбувається з мобільними додатками. Користувачі автоматично отримують оновлення при наступному відкритті додатку. Обираючи розробку Progressive Web Apps, я бажаю створити ефективний та доступний веб-додаток, який забезпечить зручний користувацький досвід та

функціональні можливості на різних пристроях та в умовах з обмеженим Інтернет-з'єднанням. Такий додаток буде актуальним та цінним в сучасному цифровому світі.

1.3. Технічний аспект проблеми

У роботі "Розробка Progressive Web Apps" використовуються різні технології, які мають свої особливості та впливають на технічний аспект проекту. Нижче наведено унікальний аспект, пов'язаний з використанням конкретних технологій у проекті.

Використання Docker для контейнеризації:

Для розгортання та управління програмним середовищем у роботі використовується Docker. Docker є відкритою платформою, що дозволяє пакувати та запускати програми в контейнерах. Ця технологія дозволяє створювати легковажні, портативні та ізольовані середовища для різних компонентів проекту. З використанням Docker можна легко розгорнути та масштабувати додаток, забезпечуючи його стабільну та ефективну роботу [1].

Використання Node.js для серверної частини:

У роботі використовується Node.js – середовище виконання JavaScript на серверній стороні. Node.js дозволяє розробникам використовувати одну мову програмування (JavaScript) як на клієнтській, так і на серверній стороні. Це спрощує процес розробки та забезпечує зручну взаємодію між фронтендом та бекендом проекту [2]. Node.js також відомий своєю високою продуктивністю та можливістю обробки багатьох одночасних запитів, що є важливим аспектом для розробки сучасних веб-додатків.

Використання PHP для серверної частини:

PHP є ще однією технологією, яка використовується у роботі для серверної частини додатку. PHP є мовою програмування, спеціалізованою на розробці веб-додатків, та має широку підтримку у веб-серверах [3]. Використання PHP

дозволяє розробникам швидко створювати функціональність серверної частини та забезпечувати взаємодію з базою даних.

Використання React для фронтенду:

Фронтенд проекту розробляється з використанням бібліотеки React. React є потужним інструментом для розробки інтерактивних та швидкодіючих користувацьких інтерфейсів [4]. Він дозволяє розбити фронтенд на компоненти, що полегшує розробку, підтримку та масштабування коду. React також забезпечує високу продуктивність завдяки використанню віртуального DOM та оптимізованих алгоритмів оновлення.

Використання Linux для операційної системи:

Операційна система Linux обрана для розгортання проекту. Linux відомий своєю стабільністю, безпекою та широким спектром можливостей для розробки та розгортання веб-додатків [5]. Він є популярним вибором серед розробників і забезпечує надійну та ефективну роботу проекту.

Використання MySQL для системи управління базами даних:

У роботі використовується MySQL - система управління базами даних, яка забезпечує зберігання та доступ до даних проекту. MySQL є популярною та надійною базою даних з великим набором функцій та підтримкою мови SQL [6]. Вона дозволяє зберігати, організовувати та оптимізовувати дані проекту, що є важливим аспектом розробки веб-додатків.

Ці технології разом утворюють технічний стек для розробки Progressive Web Apps у проекті. Вони доповнюють один одного, забезпечуючи зручну та ефективну розробку, розгортання та функціонування додатку. Кожна з цих технологій має свої переваги та особливості, які допомагають досягти поставлених цілей та забезпечити якісну роботу проекту.

2.РОЗРОБКА МОДЕЛІ ТА ПРОГРАМНОГО КОМПЛЕКСУ

2.1. Розробка моделі предметної області та бізнес моделі

2.1.1. Розробка моделі предметної областІ

Для розробки програмного забезпечення було обрано технологію єдино сторінкового застосунку з фреймворком React. Перевагами такого клієнтського додатку є :

- Швидкість і відзивчивість: SPA-додатки завантажуються повністю при першому відвідуванні, і після цього взаємодіють з сервером лише для отримання або оновлення даних. Це дозволяє знизити час завантаження сторінки та поліпшити швидкість реакції на дії користувача, оскільки весь необхідний код і ресурси знаходяться на клієнтському боці.
- Зменшення трафіку: завдяки способу роботи SPA-додатків, лише необхідні дані передаються між сервером і клієнтом, а не весь HTML-код сторінки. Це дозволяє зменшити обсяг передачі даних і використовувати мережеві ресурси більш ефективно.
- Більш плавна навігація: у SPA-додатках, навігація відбувається без перезавантаження сторінки. Користувач може взаємодіяти з додатком, не чекаючи завантаження нових сторінок. Це створює більш плавний і безперервний досвід для користувачів.
- Розширені можливості взаємодії: SPA-додатки дозволяють використовувати багато інтерактивних елементів, таких як анімація, покажчики завантаження, перетягування елементів і багато іншого. Це робить користувальний досвід більш цікавим і залучаючим для користувачів.
- Легкість розробки та підтримки: завдяки використанню фреймворків і бібліотек, таких як Angular, React або Vue.js, розробка SPA-додатків може бути спрощеною. Ці інструменти надають

розширені можливості для організації коду, управління станом додатку і взаємодії з сервером. Крім того, оновлення і розширення SPA-додатків можуть бути більш простими завдяки розділенню логіки клієнта і сервера.

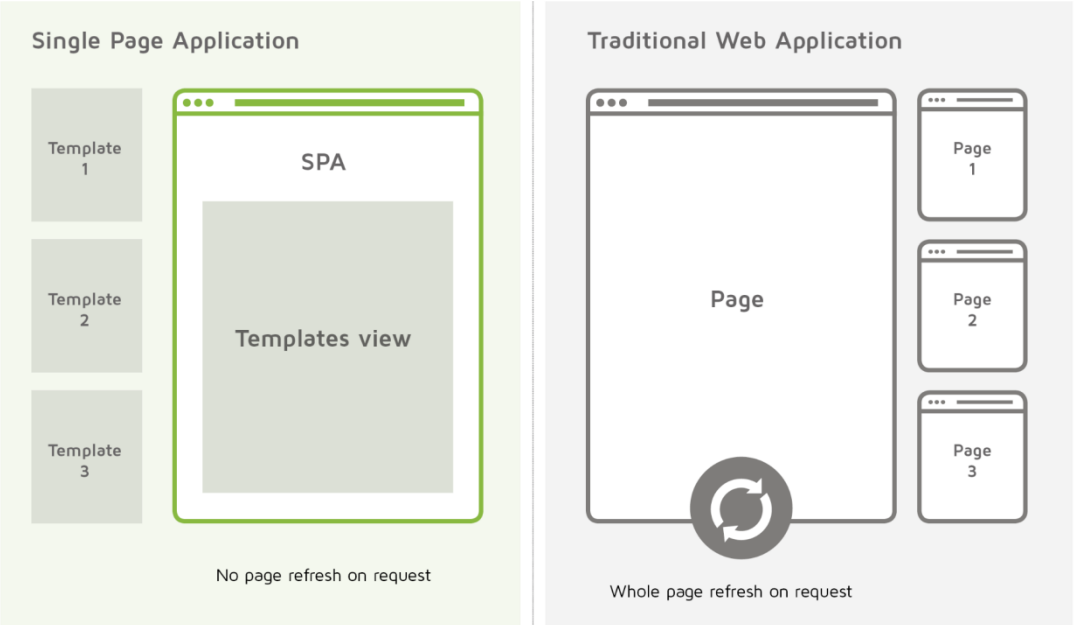


Рисунок 2.1 – Зображення відмін MVC від TWA

Загалом, SPA-додатки дозволяють створювати більш динамічні, швидкі та ефективні веб-додатки, які надають користувачам зручний і залучаючий досвід. Звичайні веб-сайти працюють за принципом, коли при кожному кліку на посилання або відправці форми веб-браузер звертається до сервера, який надсилає повний HTML-код нової сторінки. Це призводить до повторного завантаження всіх ресурсів (стилі, скрипти, зображення) і весь вміст сторінки, навіть якщо багато елементів залишаються незмінними. SPA-додатки, натомість, завантажуються повністю при першому відвідуванні. Після цього, вони взаємодіють з сервером лише для отримання або оновлення даних, а не для отримання повного HTML-коду. Це означає, що клієнтська частина додатка виконується на боці користувача, надсилаючи запити до сервера і обробляючи отримані дані без необхідності повного перезавантаження сторінки. Одна з основних переваг SPA-додатків - це швидкість і відзивчивість. Після

завантаження сторінки весь інтерфейс додатка вже доступний на клієнтській стороні. Це означає, що користувачі можуть без затримок взаємодіяти з додатком, не чекаючи завантаження нових сторінок. Всі подальші взаємодії з додатком відбуваються динамічно, без перезавантаження сторінки. Це дозволяє забезпечити плавний і безперервний досвід для користувачів. SPA-додатки також дозволяють зменшити обсяг трафіку.

При використанні традиційних веб-сайтів, кожне перезавантаження сторінки призводить до передачі повного HTML-коду, навіть якщо багато елементів залишаються незмінними. У SPA-додатках передаються лише необхідні дані між сервером і клієнтом, що дозволяє економити трафік і знижує навантаження на сервер. SPA-додатки також відкривають нові можливості для більш плавної навігації та взаємодії з користувачем. Оскільки весь інтерфейс додатка вже завантажений, перехід між сторінками відбувається миттєво і без затримок. Користувач може легко перемикатися між різними розділами додатка, без перезавантаження сторінки. Крім того, SPA-додатки можуть використовувати багато інтерактивних елементів, таких як анімація, покажчики завантаження, перетягування елементів і багато іншого, що створює більш цікавий та залучаючий користувальний досвід. Нарешті, розробка та підтримка SPA-додатків може бути спрощеною завдяки використанню фреймворків та бібліотек, таких як Angular, React або Vue.js. Ці інструменти надають розширені можливості для організації коду, управління станом додатку і взаємодії з сервером. Вони дозволяють розробникам швидко створювати складні SPA-додатки і спрощують процес розширення та підтримки додатків.

2.1.2. Розробка бізнес моделі взаємодії користувачів з магазином

При створенні бізнес-моделі важливо належним чином розуміти ролі користувачів системи та їх взаємодію з веб-сайтом. Необхідно провести аналіз всіх можливих сценаріїв використання магазину користувачами. Щоб краще зрозуміти модель, рекомендується розробити UML-діаграми, які потім можна

буде реалізувати у вигляді програми. Один з перших кроків у розробці бізнес-моделі – створення загальної UML-діаграми зовнішньої взаємодії. Тому перша створена діаграма буде діаграмою варіантів використання для інтернет-магазину [5] (див. рисунок 2.2).

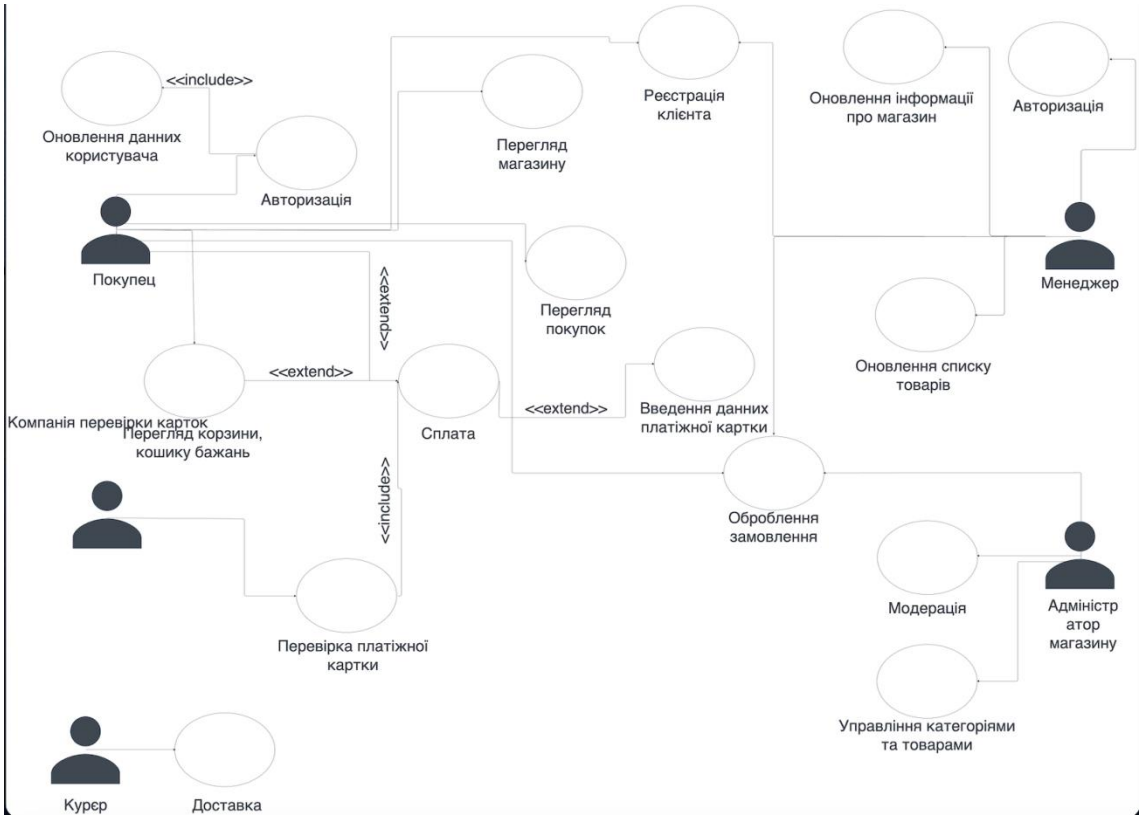


Рисунок 2.2 – Діаграма варіантів використання інтернет-магазину

Під час розробки бізнес-моделі були враховані три категорії користувачів: покупці, менеджери і адміністратори. Вибір такого набору осіб був обумовлений їхніми унікальними можливостями та потребами. Це надало основу для створення UML-діаграм активності, які використовуються для аналізу потоків подій. Результати цього аналізу можна знайти у таблицях потоків подій (див. табл. 2.1), які використовуються для побудови діаграм діяльності. Цей аналіз допомагає визначити основні процеси та взаємодії між різними діями та учасниками системи.

Таблиця 2.1 – Потоки подій

№	Потоки подій	Опис
1	Додавання товару у кошик	● Користувач переглядає сторінку зі списком товарів або окремий товар, який бажає додати до свого кошика. ● Користувач натискає на кнопку "Додати у кошик" або аналогічний елемент у інтерфейсі, що вказує на додавання товару до кошика. ● Система отримує дані про вибраний товар, його ідентифікатор, кількість тощо. ● Система перевіряє наявність товару в кошику. Якщо товар вже присутній, оновлюється його кількість, в іншому випадку, створюється новий запис про товар у кошику. ● Система зберігає оновлений стан кошика, оновлюючи його дані в базі даних або локальному сховищі. ● Користувач отримує підтвердження про успішне додавання товару у кошик. ● Користувач може продовжити переглядати і додавати інші товари у кошик або перейти до оформлення замовлення.
2	Фільтрація товарів	● Користувач активує функцію фільтрації, що доступна на сторінці інтернет-магазину. ● Користувач обирає критерії фільтрації, такі як ціновий діапазон, категорія товару, бренд тощо.

		● Система обробляє вибрані критерії та

Продовження таблиці 2.1

№	Потоки подій	Опис
		● виконує фільтрацію товарів відповідно до них. ● Результати фільтрації відображаються користувачу у вигляді списку товарів, які відповідають обраним критеріям. ● Користувач може переглянути деталі кожного товару, перейшовши на окрему сторінку товару.
3	Пошук товару	● Користувач активує пошукове поле, розташоване на сторінці інтернет-магазину, з метою знаходження потрібного товару. ● Користувач вводить унікальний код товару або ключові слова, що характеризують його, в поле пошуку. ● Система управління базами даних (СУБД) отримує запит із введеними даними і звертається до таблиці "Товари" для пошуку відповідних записів. ● СУБД виконує пошук, використовуючи унікальний код товару або шляхом співпадіння ключових слів з описом товару, назвою, категорією тощо. ● Результати пошуку передаються назад до інтерфейсу користувача.

		● Користувачу відображається сторінка зі списком товарів, які відповідають заданим

Продовження таблиці 2.1

		● критеріям пошуку. ● Користувач може переглянути деталі кожного товару, перейшовши на окрему сторінку товару.
4	Купівля товару	● Користувач вибирає бажаний товар на сторінці інтернет-магазину та натискає кнопку "Додати до кошика" або "Купити". ● Система перевіряє наявність товару на складі та його ціну. ● Якщо товар є в наявності і його ціна відповідає умовам, система додає товар до кошика або переходить до процесу оформлення замовлення. ● Користувач переглядає вміст свого кошика, може внести зміни (додати/видалити товари, змінити кількість тощо). ● Після підтвердження замовлення, користувач вводить необхідні дані для доставки та оплати. ● Система обробляє замовлення, проводить операцію оплати та генерує підтвердження замовлення. ● Користувач отримує підтвердження про успішну купівлю, а також інформацію про

		спосіб оплати та очікуваний час доставки.

Продовження таблиці 2.1

№	Потоки подій	Опис
5	Додавання товару у лист побажань	● Користувач вибирає бажаний товар на сторінці інтернет-магазину. ● Користувач натискає кнопку або елемент, що вказує на можливість додати товар до свого списку побажань, наприклад, "Додати до списку побажань" або іконка серця. ● Система перевіряє, чи наявний товар у списку побажань користувача. Якщо так, система повідомляє користувача про це, а якщо ні, продовжує виконувати наступні кроки. ● Система додає товар до списку побажань користувача, зберігаючи його дані, такі як назва товару, зображення, ціна тощо. ● Користувач отримує підтвердження про успішне додавання товару до списку побажань. ● У майбутньому, коли користувач переглядає свій список побажань, він може побачити всі додані раніше товари та їх характеристики. ● Користувач може видалити товар зі списку побажань або змінити його статус.

З метою забезпечення більшої ефективності та розуміння функціональності системи було прийнято важливе рішення щодо побудови діаграм активності. Для цього було використано три ключові варіанти використання: "Вхід користувача в систему" (запропонований рисунок 2.3), "Пошук товарів у системі" та "Оцінка товару користувачем". Ці варіанти дозволили визначити основні дії та послідовність подій, які стосуються користувача та його взаємодії з системою.

Такий підхід до побудови діаграм активності не тільки сприяє розумінню та аналізу функцій системи, але й допомагає виявити потенційні можливості для поліпшення та оптимізації процесів. Це важливий крок у розробці бізнес-моделі, який дозволить забезпечити успішне впровадження та функціонування системи у реальному середовищі.

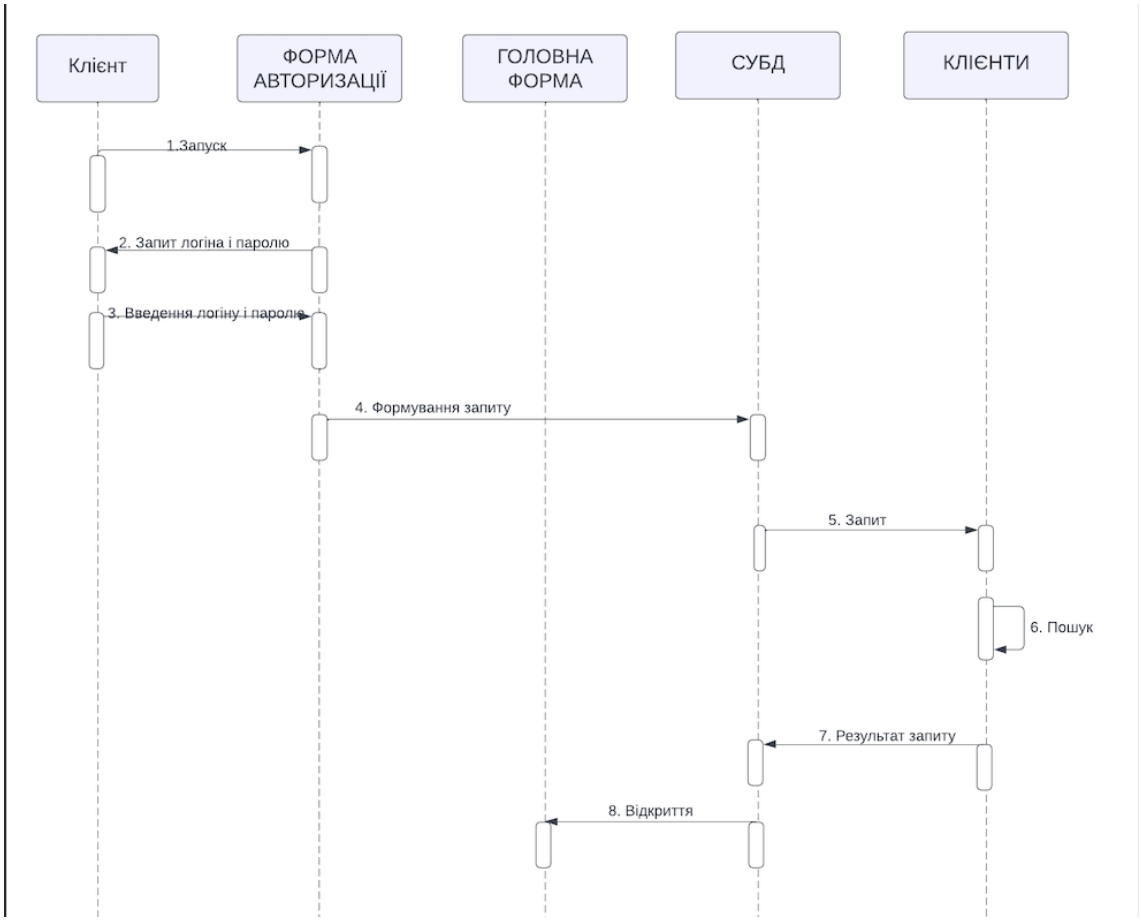


Рисунок 2.3 – Діаграма активності для варіанту використання «Авторизація користувача»

Необхідний аналіз альтернативних варіантів для визначення більшості подій, які виникають, коли основний потік не може бути успішним. У випадку з варіантом "Вхід користувача в систему", альтернативний потік буде використаний, якщо будь-яке з полів вводу заповнено неправильно (див. рисунок 2.4). Для використання варіанту "Пошук товарів у системі", альтернативний потік буде активований, якщо товар не буде знайдено в базі даних. У випадку використання варіанту "Оцінка товару користувачем", альтернативний потік буде використаний, якщо користувач спробує повторно оцінити вже оцінений товар. Діаграми діяльності, які були створені, наведені у додатку В. Ці діаграми дозволять краще розуміти послідовність подій та виявити потенційні проблеми, що можуть виникнути під час взаємодії з системою.

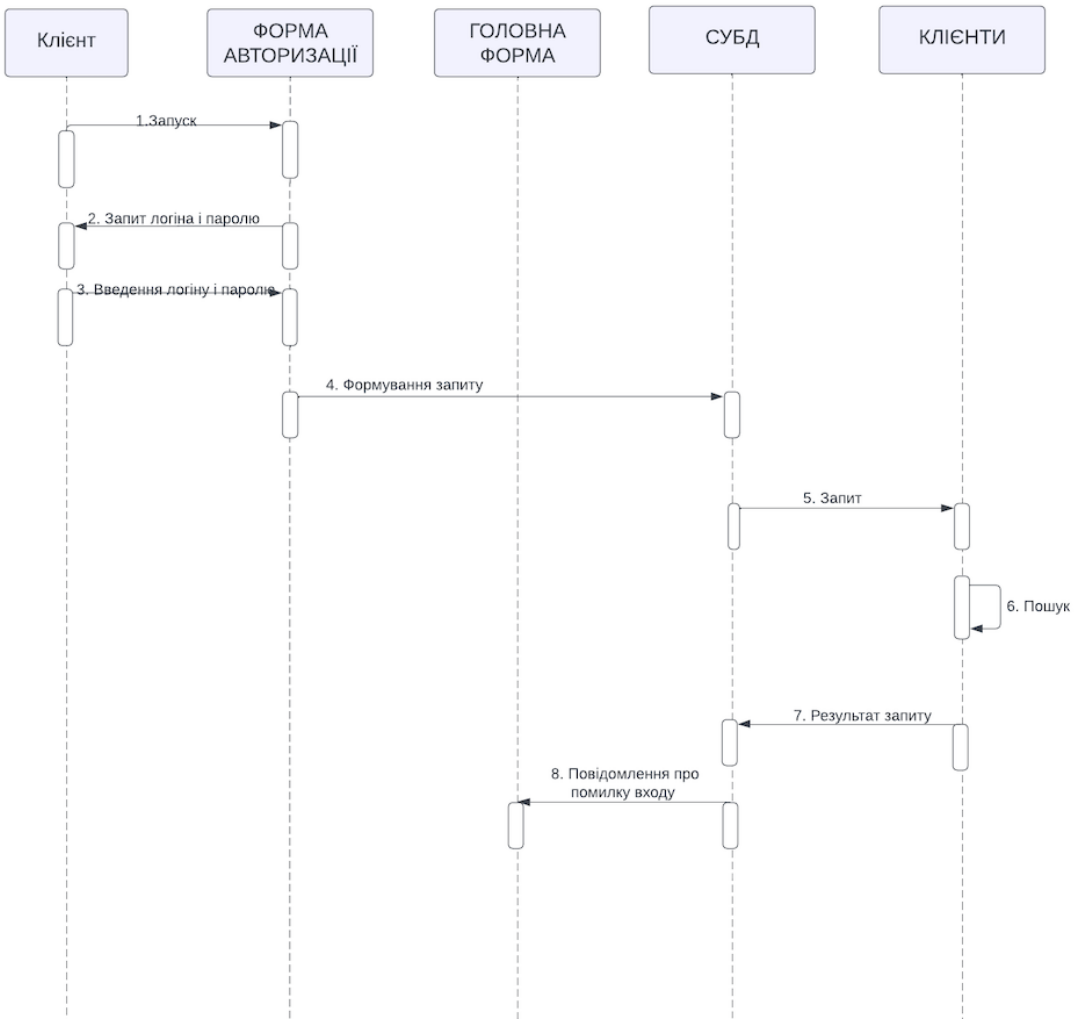


Рисунок 2.4 – Діаграма активності для альтернативного потоку варіанта використання «Вхід користувача в систему»

Таким чином в ході аналізу було розроблено UML діаграму варіантів використання інтернет-магазину, розглянуто потоки подій та побудовано діаграми діяльності для трьох варіантів використання, з урахуванням альтернативних потоків подій для кожного з варіантів використання.

2.2. Проектування клієнтського додатку інтернет-магазину з використанням технології Progressive Web App.

Для створення онлайн магазину з використанням технології Progressive Web App нам необхідна серверний додаток де будуть зберігатися та обробляються дані. Отже я прийняв рішення використовувати для серверного додатку CMS Magento 2. Тепер мені треба було вирішити як, яким способом мій клієнтський додаток буде обмінюватися даними з серверним додатком. Самі популярні способи обміну даними між REST API і GraphQl.

REST API (Representational State Transfer) та GraphQL є двома різними підходами до розробки API (Application Programming Interface), які використовуються для обміну даними між клієнтськими та серверними додатками. Основна різниця між ними полягає у способі, яким клієнти отримують дані від сервера та взаємодіють з ним [7].

REST API:

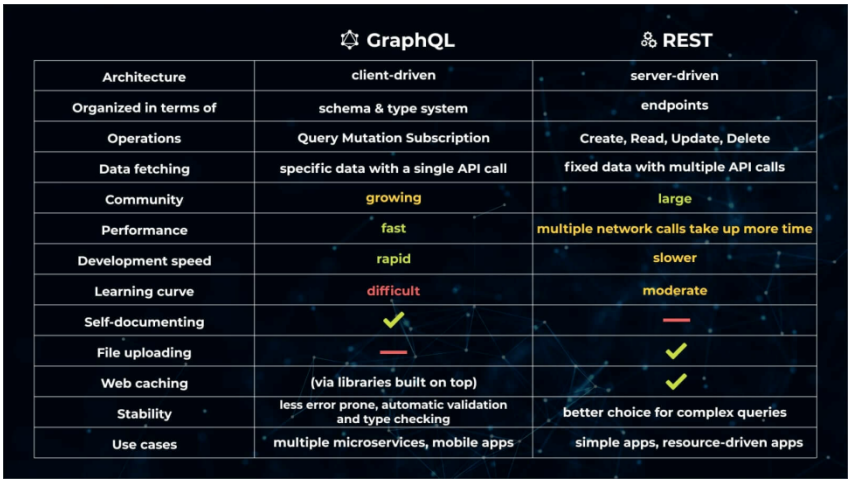
- REST API базується на принципах REST, які включають використання HTTP методів (GET, POST, PUT, DELETE) для виконання операцій над ресурсами.
- У REST API, для кожного типу ресурсів визначаються окремі URL-шляхи (endpoints), наприклад "/users" для отримання списку користувачів.
- Клієнти роблять запити до конкретних URL-шляхів та отримують повернені дані у форматі JSON або XML.

- REST API передає всі дані, пов'язані з ресурсом, навіть якщо клієнт потребує лише частину цих даних.
- Клієнти не контролюють формат та структуру повернутих даних, які визначаються сервером.

GraphQL:

- GraphQL є мовою запитів та виконавчим середовищем, що дозволяє клієнтам запитувати лише ті дані, які їм потрібні, та отримувати їх в одному запиті.
- У GraphQL немає фіксованих URL-шляхів. Клієнти звертаються до одного URL-шляху та використовують спеціальну запитову мову для визначення структури даних, які вони хочуть отримати [8].
- Клієнти можуть точно вказати, які поля та зв'язки їх цікавлять, та отримати дані з різних ресурсів у відповідності до своїх потреб.
- GraphQL повертає дані у форматі JSON.
- Сервер визначає доступні дані та можливі операції, а клієнти контролюють структуру запиту та отримують лише необхідні дані.

Основна перевага GraphQL полягає в тому, що воно дозволяє клієнтам отримувати точно ті дані, які вони потребують, у єдиному запиті, що поліпшує ефективність та продуктивність мобільних та веб-додатків. REST API, з іншого боку, є більш традиційним підходом та залишається популярним для багатьох веб-додатків, які працюють зі стандартними операціями CRUD (створення, отримання, оновлення, видалення) над ресурсами.



	GraphQL	REST
Architecture	client-driven	server-driven
Organized in terms of	schema & type system	endpoints
Operations	Query Mutation Subscription	Create, Read, Update, Delete
Data fetching	specific data with a single API call	fixed data with multiple API calls
Community	growing	large
Performance	fast	multiple network calls take up more time
Development speed	rapid	slower
Learning curve	difficult	moderate
Self-documenting	✓	—
File uploading	—	✓
Web caching	(via libraries built on top)	✓
Stability	less error prone, automatic validation and type checking	better choice for complex queries
Use cases	multiple microservices, mobile apps	simple apps, resource-driven apps

Рисунок 2.5 – Різниця між GraphQL та Rest

3.КОНСТРУЮВАННЯ ІНТЕРНЕТ-МАГАЗИНУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ PROGRESSIVE WEB APP

3.1. Реалізація Progressive Web Apps

Для створення Progressive Web Apps необхідно ініціалізувати проект та встановити всі необхідні пакети разом з їх залежностями. Наступним необхідним кроком буде створення serviceWorker (див. рисунок 3.1–3.2).

```
import { handleMessageFromClient } from './Utilities/messageHandler';
import setupWorkbox from './setupWorkbox';
import registerRoutes from './registerRoutes';
import registerMessageHandlers from './registerMessageHandlers';

setupWorkbox();

registerRoutes();

registerMessageHandlers();

self.addEventListener( type: 'message', listener: e : MessageEvent<any>  => {
    const { type, payload } = e.data;

    handleMessageFromClient(type, payload, e);
});
```

Рисунок 3.1 – Ініціалізація Сервіс-воркера

```
import {
    VALID_SERVICE_WORKER_ENVIRONMENT,
    handleMessageFromSW
} from '@magento/peregrine/lib/util/swUtils';

2 usages
export const registerSW = () : void  => {
    if (VALID_SERVICE_WORKER_ENVIRONMENT && globalThis.navigator) {
        window.navigator.serviceWorker
            .register( scriptURL: '/sw.js')
            .then(() : void  => {
                console.log('SW Registered');
            })
            .catch(() : void  => {
                window.console.warn( data: 'Failed to register SW.');
            });

        navigator.serviceWorker.addEventListener( type: 'message',  listener: e : MessageEvent<any>  => {
            const { type, payload } = e.data;
            handleMessageFromSW(type, payload, e);
        });
    }
};
```

Рисунок 3.2 – Реєстрація Сервіс-воркера

Сервіс-воркер (Service Worker) – це спеціальний скрипт JavaScript, який функціонує в браузері у фоновому режимі. Він має широкий спектр можливостей, які допомагають забезпечити високу якість та надійність розробленої Progressive Web App (Progressive Web App) [9]. Сервіс-воркери дозволяють виконувати різні завдання, включаючи кешування ресурсів, підтримку офлайн-режиму, взаємодію з пуш-сповіщеннями та багато іншого.

Один з головних аспектів використання сервіс-воркера – це можливість кешування ресурсів, таких як HTML-сторінки, стилі, зображення, скрипти та інші. Це дозволяє зберігати копії цих ресурсів на стороні клієнта і використовувати їх при подальшій взаємодії з додатком, навіть якщо немає доступу до Інтернету.

Крім того, сервіс-воркери дозволяють впроваджувати пуш-сповіщення, що є потужним інструментом для залучення користувачів. За допомогою пуш-сповіщень можна надсилати повідомлення користувачам, навіть коли вони неактивні на веб-сторінці.

Таким чином, використання сервіс-воркерів в розробці Progressive Web App є необхідним кроком для забезпечення високої якості та надійності додатка. Вони надають можливість кешування ресурсів, підтримку офлайн-режиму, взаємодію з пуш-сповіщеннями та багато іншого, що покращує користувацький досвід та продуктивність. Тестування, зокрема використання Google Lighthouse, тестування дизайну, юніт-тестування та тестування у середовищі React, допомагають виявити та виправити потенційні проблеми, забезпечуючи стабільну та високоякісну роботу Progressive Web App.

Наступним кроком було налаштування маніфесту Progressive Web App (див. рисунок 3.3).

Рисунок 3.3 – Маніфест Progressive Web App

Файл manifest.json є ключовим компонентом Progressive Web App (PWA), оскільки він містить метадані та конфігураційні налаштування, необхідні для правильного відображення та функціонування додатка. Manifest.json надає браузеру інформацію про Progressive Web App, включаючи назву, опис, автора, іконки, тему кольорів та інші важливі візуальні атрибути.

Відповідно до стандартів Progressive Web App, файл manifest.json повинен бути розміщений у кореневій папці веб-додатка і бути доступним за допомогою посилання /manifest.json. Це дозволяє браузеру знайти і завантажити цей файл для отримання необхідної інформації про Progressive Web App.

У файлі manifest.json можна визначити різні налаштування, такі як theme_color – кольорова тема додатка, background_color – колір фону, display – режим відображення (наприклад, standalone або fullscreen), іконки для різних платформ та пристроїв, назву та опис додатка.

Важливо зазначити, що файл manifest.json повинен бути правильно сформований і відповідати структурі, визначеній стандартами Progressive Web App. Будь-які помилки або неправильні налаштування можуть призвести до некоректного відображення або непередбачуваної поведінки додатка.

В цілому, файл manifest.json є важливим компонентом Progressive Web App, який дозволяє визначити інформацію та налаштування додатка, що сприяє його правильному відображенню та функціонуванню на різних платформах та пристроях.

Також необхідним функціоналом мого магазину з технологією Progressive Web App було реалізовано Push-сповіщення.

Push-сповіщення (або пуш-нотифікації) є ефективним засобом спілкування з користувачами веб-додатків, особливо в контексті Progressive Web Apps (PWA). Вони дозволяють надсилати повідомлення користувачам навіть тоді, коли вони неактивні на веб-сайті або не використовують додаток. Push-сповіщення можуть містити текстове повідомлення, зображення, посилання та інші візуальні елементи, що привертають увагу користувача.

Для реалізації цього функціоналу було створено слухача Push-сповіщень (див. рисунок 3.4).

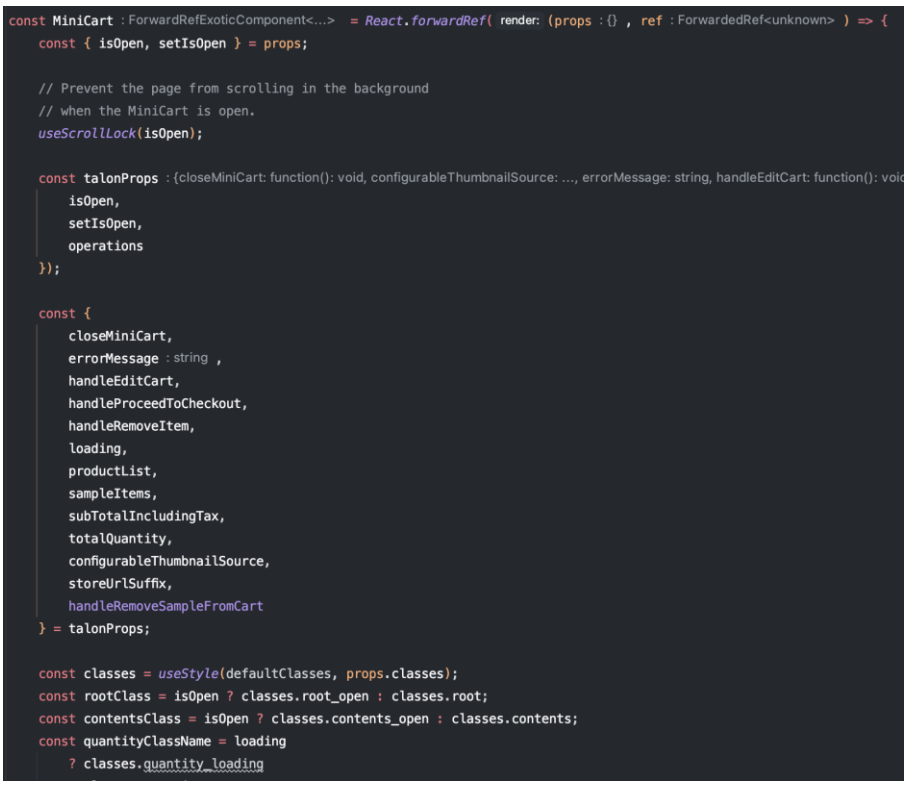
```
self.addEventListener( type: 'push', listener: (event : Event ) : void => {
    const options : {body: any} = {
        body: event.data.text(),
        // Додаткові параметри для відображення сповіщення
        // наприклад, заголовок, іконка тощо
    };

    event.waitUntil(
        self.registration.showNotification( title: 'Нове сповіщення', options)
    );
});

self.addEventListener( type: 'notificationclick', listener: (event : Event ) : void => {
    event.notification.close();
    // Логіка, яка виконується при натисканні на сповіщення
    // наприклад, відкриття вікна додатка або перехід на певну сторінку
});
```

Рисунок 3.4 – Слухач Push-сповіщень

3.2. Розробка функціоналу інтернет-магазину

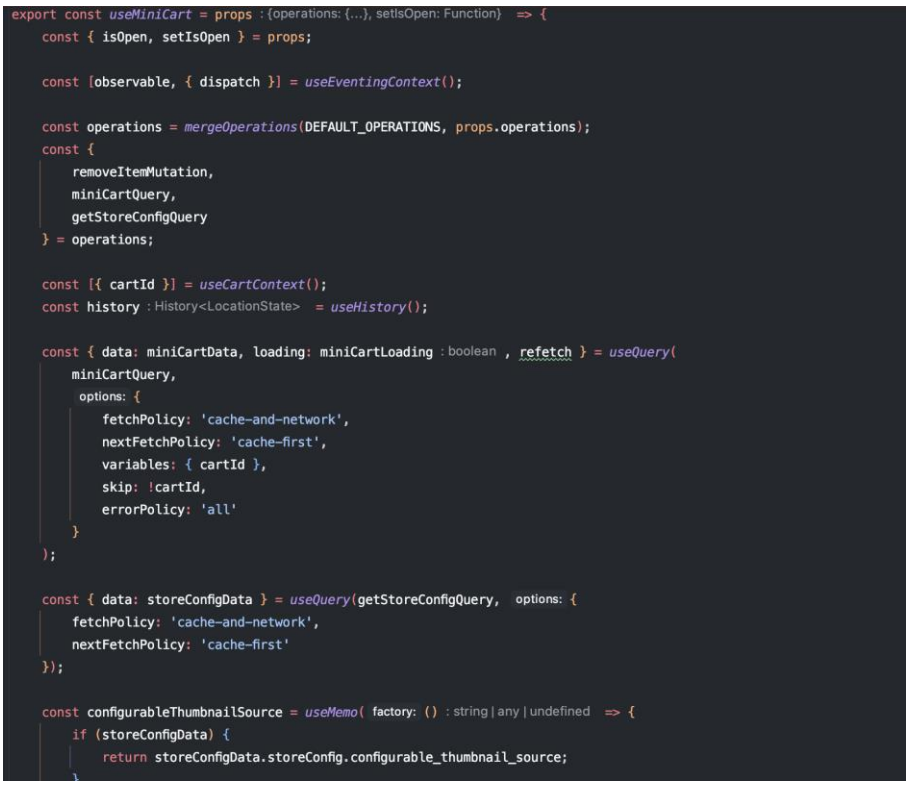
Першим найважливішим функціоналом інтернет магазину є корзина користувача. Під час розробки було використано метод декомпозування для розділення візуальної і логічної частини. Отже до кожної компоненти був створений свій талон. Талон це логічна частина компоненти створена для отримання і обробки даних з сервера.

Для візуалізації була розроблена компонента “MiniCart” (див. рисунок 3.5) а для написання логіки роботи з корзиною було створено талон “useMiniCart” (див. рисунок 3.6)



```
const MiniCart : ForwardRefExoticComponent<...>  = React.forwardRef( render: (props : {} , ref : ForwardedRef<unknown> ) => {
    const { isOpen, setIsOpen } = props;

    // Prevent the page from scrolling in the background
    // when the MiniCart is open.
    useScrollLock(isOpen);

    const talonProps : {closeMiniCart: function(): void, configurableThumbnailSource: ..., errorMessage: string, handleEditCart: function(): voi
        isOpen,
        setIsOpen,
        operations
    });

    const {
        closeMiniCart,
        errorMessage : string ,
        handleEditCart,
        handleProceedToCheckout,
        handleRemoveItem,
        loading,
        productList,
        sampleItems,
        subTotalIncludingTax,
        totalQuantity,
        configurableThumbnailSource,
        storeUrlSuffix,
        handleRemoveSampleFromCart
    } = talonProps;

    const classes = useStyle(defaultClasses, props.classes);
    const rootClass = isOpen ? classes.root_open : classes.root;
    const contentsClass = isOpen ? classes.contents_open : classes.contents;
    const quantityClassName = loading
        ? classes.quantity_loading
```

Рисунок 3.5 – Компонента “MiniCart”



```
export const useMiniCart = props : {operations: {...}, setIsOpen: Function}  => {
    const { isOpen, setIsOpen } = props;

    const [observable, { dispatch }] = useEventingContext();

    const operations = mergeOperations(DEFAULT_OPERATIONS, props.operations);
    const {
        removeItemMutation,
        miniCartQuery,
        getStoreConfigQuery
    } = operations;

    const [{ cartId }] = useCartContext();
    const history : History<LocationState>  = useHistory();

    const { data: miniCartData, loading: miniCartLoading : boolean , refetch } = useQuery(
        miniCartQuery,
         options: {
            fetchPolicy: 'cache-and-network',
            nextFetchPolicy: 'cache-first',
            variables: { cartId },
            skip: !cartId,
            errorPolicy: 'all'
        }
    );

    const { data: storeConfigData } = useQuery(getStoreConfigQuery,  options: {
        fetchPolicy: 'cache-and-network',
        nextFetchPolicy: 'cache-first'
    });

    const configurableThumbnailSource = useMemo( factory: () : string | any | undefined  => {
        if (storeConfigData) {
            return storeConfigData.storeConfig.configurable_thumbnail_source;
```

Рисунок 3.6 – Талон “useMiniCart”

Основними функціями роботи з корзиною це додавання, видалення товару. Цей функціонал був розроблений у середині компоненти “Product”(див. рисунок 3.7) та талона “useProduct” (див. рисунок 3.8).

```
            }
        ],
        ...(carouselCenterMode && { centerPadding }),
        ...{ infinite: items.length > slidesToShow && infinite }
    };

    const centerModeClass : any | null  = carouselCenterMode ? classes.centerMode : null;
    const centerModeSmallClass : String | any | null  = carouselSmallCenterMode
        ? classes.centerModeSmall
        : null;

    return (
        <div
            style={dynamicStyles}
            className={[
                classes.carousel,
                ...cssClasses,
                centerModeClass,
                centerModeSmallClass
            ].join(' ')}
        >
            <Carousel settings={carouselSettings} items={items} />
        </div>
    );
}

return (
    <div
        style={dynamicStyles}
        className={[classes.root, ...cssClasses].join(' ')}
    >
        <Gallery items={items} classes={{ items: classes.galleryItems }} />
    </div>
);
```

Рисунок 3.7 – Компонента “Product”

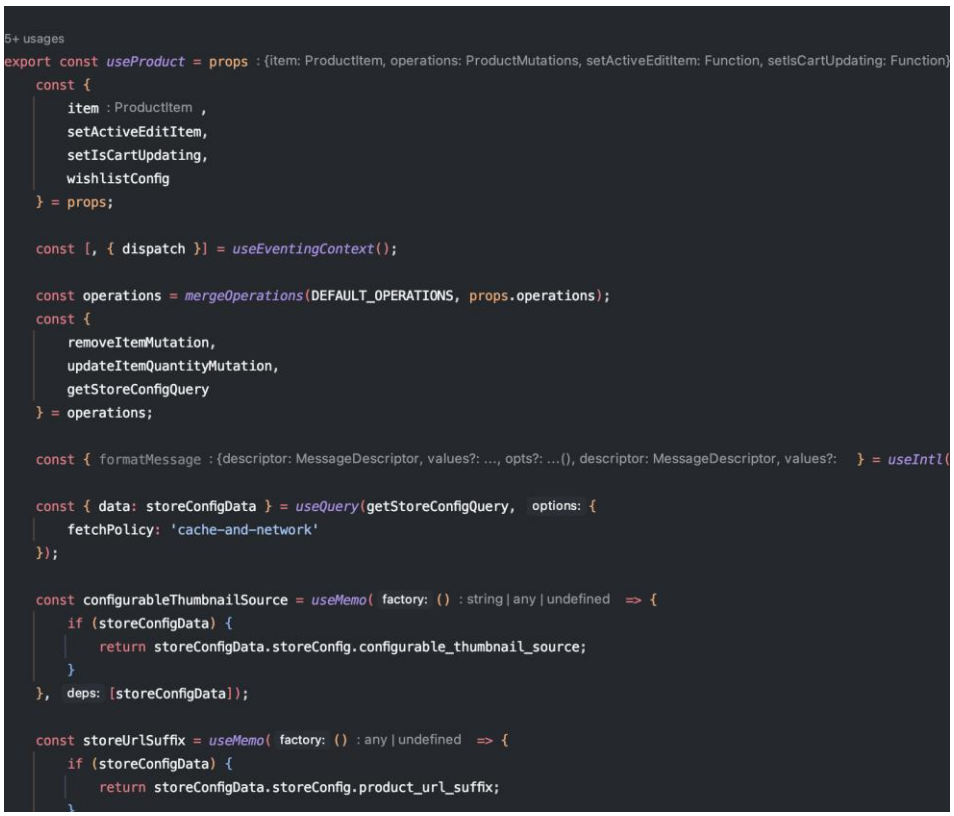
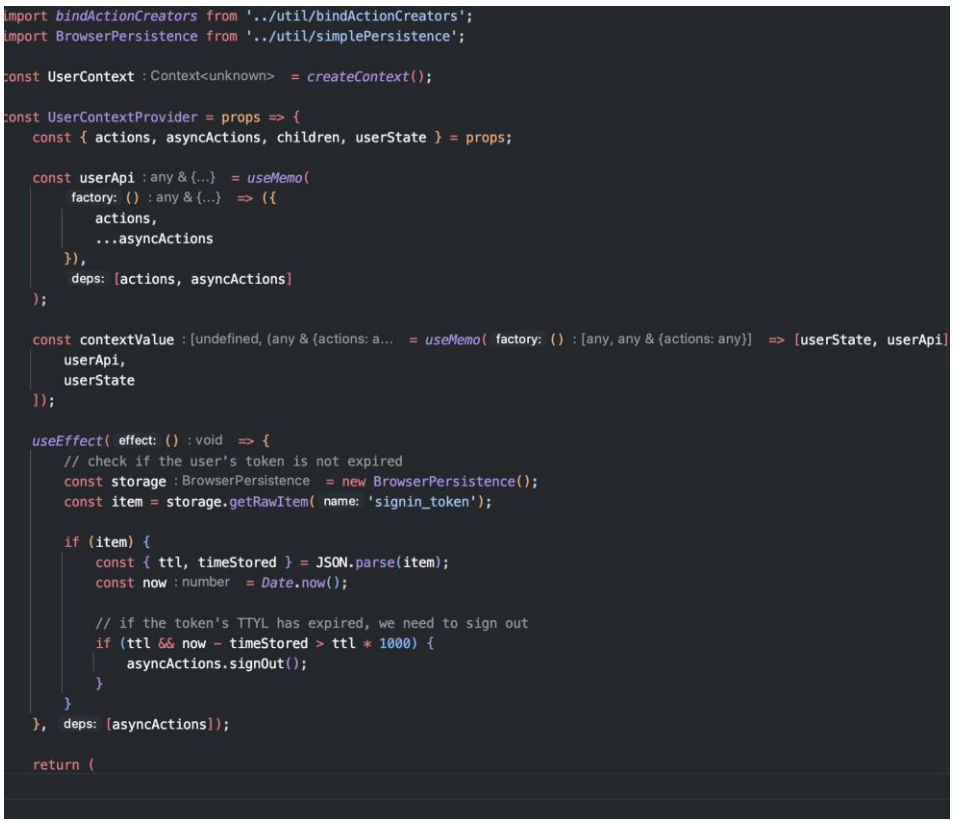


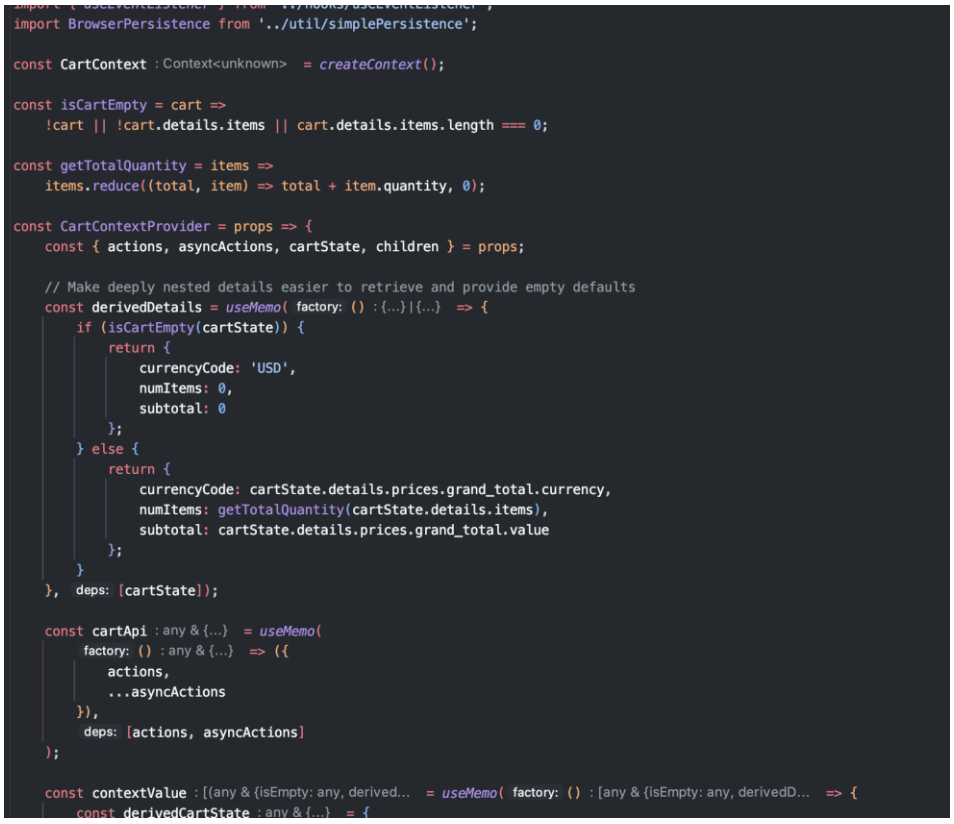
Рисунок 3.8 – Талон “useProduct”

Під час розробки було помічено проблему зберіганню стану компонентів та передача його від компоненти до компоненти незалежно від вкладеності і можливісті їх змінювати. Для вирішення цих проблем було вирішено використати хук “useContext”. Так було створено контекст до самих важливих сутностей таких як Користувач (див. рисунок 3.9) та Кошик (див. рисунок 3.10).



```
import bindActionCreators from '../util/bindActionCreators';
import BrowserPersistence from '../util/simplePersistence';

const UserContext : Context<unknown>  = createContext();

const UserContextProvider = props => {
    const { actions, asyncActions, children, userState } = props;

    const userApi : any & {...}  = useMemo(
        factory: () : any & {...}  => ({
            actions,
            ...asyncActions
        }),
        deps: [actions, asyncActions]
    );

    const contextValue : [undefined, (any & {actions: a...  = useMemo( factory: () : [any, any & {actions: any}]  => [userState, userApi]
        userApi,
        userState
    ]);

    useEffect( effect: () : void  => {
        // check if the user's token is not expired
        const storage : BrowserPersistence  = new BrowserPersistence();
        const item = storage.getRawItem( name: 'signin_token');

        if (item) {
            const { ttl, timeStored } = JSON.parse(item);
            const now : number  = Date.now();

            // if the token's TTYL has expired, we need to sign out
            if (ttl && now - timeStored > ttl * 1000) {
                asyncActions.signOut();
            }
        }
    },  deps: [asyncActions]);

    return (
```

Рисунок 3.9 – Контекст користувача



```
import BrowserPersistence from '../util/simplePersistence';

const CartContext : Context<unknown>  = createContext();

const isCartEmpty = cart =>
    !cart || !cart.details.items || cart.details.items.length === 0;

const getTotalQuantity = items =>
    items.reduce((total, item) => total + item.quantity, 0);

const CartContextProvider = props => {
    const { actions, asyncActions, cartState, children } = props;

    // Make deeply nested details easier to retrieve and provide empty defaults
    const derivedDetails = useMemo( factory: () : {...}|{...}  => {
        if (isCartEmpty(cartState)) {
            return {
                currencyCode: 'USD',
                numItems: 0,
                subtotal: 0
            };
        } else {
            return {
                currencyCode: cartState.details.prices.grand_total.currency,
                numItems: getTotalQuantity(cartState.details.items),
                subtotal: cartState.details.prices.grand_total.value
            };
        }
    },  deps: [cartState]);

    const cartApi : any & {...}  = useMemo(
        factory: () : any & {...}  => ({
            actions,
            ...asyncActions
        }),
        deps: [actions, asyncActions]
    );

    const contextValue : [(any & {isEmpty: any, derived...  = useMemo( factory: () : [any & {isEmpty: any, derivedD...  => {
        const derivedCartState : any & {...}  = {
```

Рисунок 3.10 – Контекст кошику

3.3. Тестування програмного забезпечення та оцінка якості

Тестування програмного забезпечення та оцінка якості є важливою складовою для забезпечення якості та надійності розробленого програмного забезпечення [10]. Для тестування свого додатку я використовував юніт тестування, Google lighthouse, тестування React компонент та тестування UI.

- Використання Google Lighthouse: Google Lighthouse є потужним інструментом, розробленим Google, який допомагає аналізувати та оцінювати різні аспекти якості та продуктивності веб-додатків. Під час розробки PWA, використання Google Lighthouse є важливим кроком для перевірки таких параметрів, як продуктивність завантаження, доступність, використання кешування, SEO та інші. Звіти, що надаються Google Lighthouse, містять рекомендації щодо покращення якості та продуктивності PWA.
- Тестування дизайну: Дизайн є важливим аспектом PWA, оскільки він безпосередньо впливає на користувацький досвід. Тестування дизайну включає перевірку розташування елементів, кольорової палітри, типографіки, взаємодії та інших аспектів дизайну. Забезпечення відповідності дизайну до вимог, які встановлені для PWA, допомагає забезпечити зручний та привабливий інтерфейс для користувачів.
- Юніт-тестування: Юніт-тестування є методом тестування, за яким окремі модулі та компоненти програмного забезпечення перевіряються на правильну роботу. У випадку розробки PWA, юніт-тестування включає перевірку функціональності окремих компонентів, реагування на події, валідацію введених даних та інші аспекти. Це допомагає виявити та виправити можливі помилки та недоліки на ранніх етапах розробки, забезпечуючи стабільність та надійність PWA.

- Тестування React component: У розробці PWA я використовував фреймворк React для побудови користувацького інтерфейсу. Тестування у середовищі React включає в себе перевірку правильності рендерингу компонентів, взаємодії між компонентами та перевірку функціональності, яка залежить від React-специфічних особливостей [11]. Для цього можна використовувати спеціальні інструменти тестування React, такі як Enzyme або React Testing Library, які допомагають забезпечити якісне тестування компонентів у середовищі React.

Шлях до успішної розробки PWA включає постійне тестування та оцінку якості. Використання Google Lighthouse допомагає перевірити різні аспекти додатка, включаючи його продуктивність, швидкість завантаження та доступність. Тестування дизайну дозволяє перевірити естетику, зручність використання та відповідність бренду. Юніт-тестування допомагає виявляти та виправляти помилки в окремих компонентах, а тестування у середовищі React забезпечує впевненість у правильності реалізації функціональності.

Використовуючи вищі наведені методи та інструменти тестування я зміг отримати великі показники оцінки Google Lighthouse (див. рисунок 3.11). Це допоможе мені краще просувати мій додаток у пошукових системах, та покращить користувачам використання.

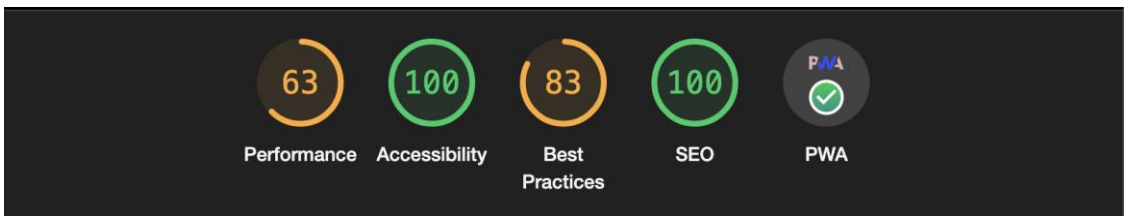


Рисунок 3.11 – Різниця між GraphQL та Rest

Використання цих методів тестування та оцінки якості дозволяє забезпечити високу якість та надійність розробленої PWA. Вони допомагають виявити та виправити потенційні проблеми, поліпшити користувацький досвід та

забезпечити ефективну роботу додатка на різних пристроях та браузерах. Налаштування тестування та оцінка якості повинні бути постійними етапами розробки PWA з метою забезпечення його успіху та задоволення потреб користувачів.

Отже перш за все треба протестувати це дизайн сайту. Дизайн сайту повинен бути виконаним у фіолетових і білих тонах(див. рисунок 3.12).

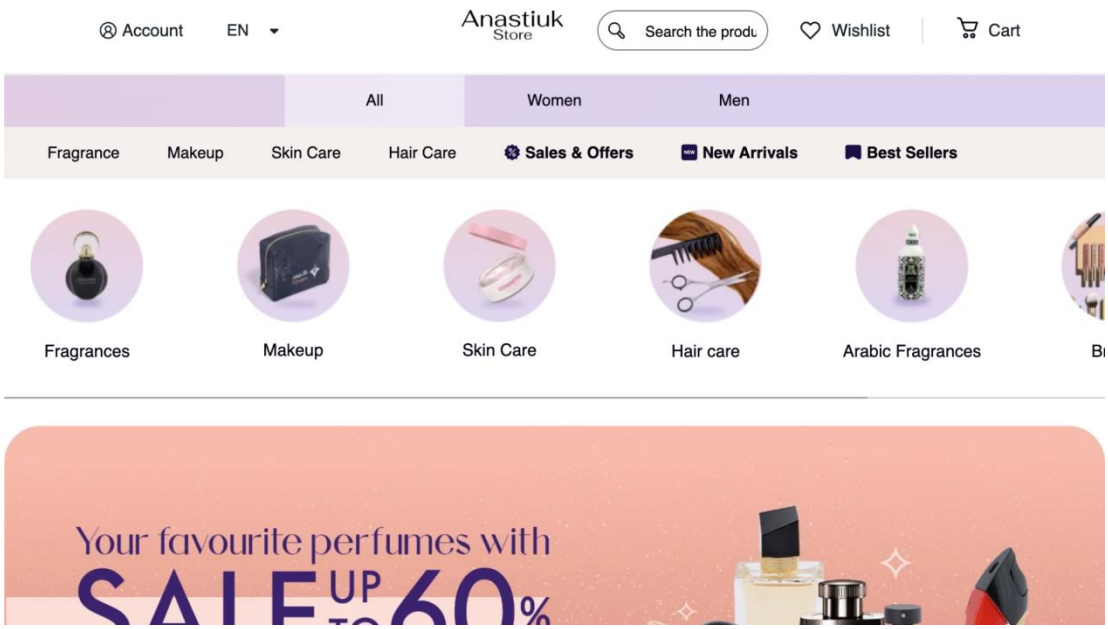


Рисунок 3.12 – Домашня сторінка сайту

Також у футері сайту були розміщені всі необхідні посилання, розроблений дизайн для прокруту продуктів(див. рисунок 3.14).

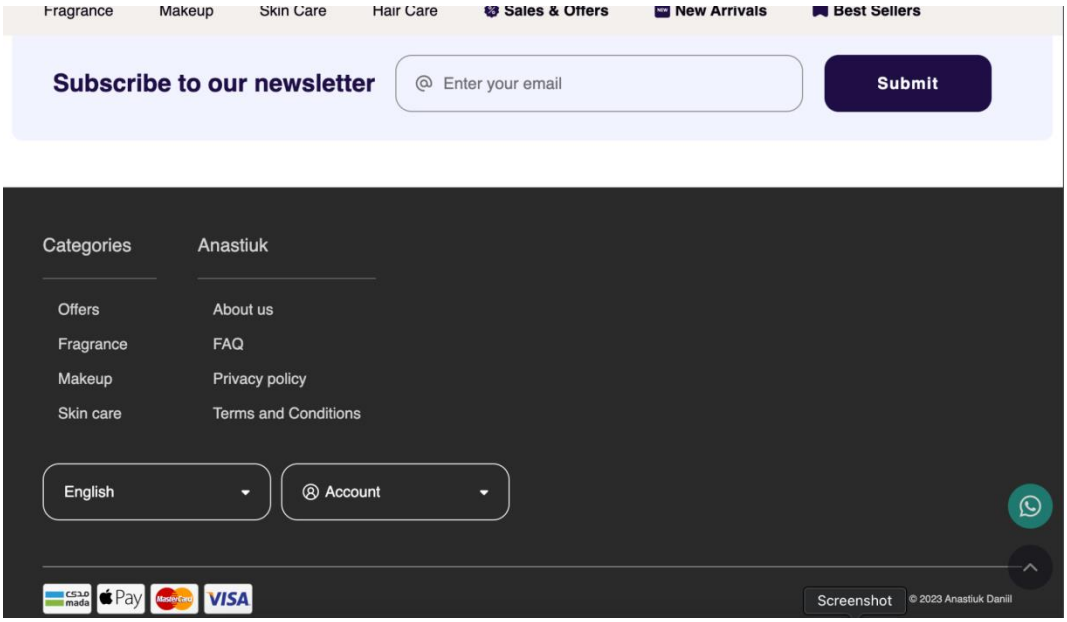


Рисунок 3.14 – Футер сайту

Наступним функціоналом який необхідно перевірити буде тестування способу оплати. Для цього необхідно додати товар у кошик і перейти на сторінку оплати. Вибрати спосіб оплати і натиснути кнопку “Place Order” (див. рисунок 3.15). Після виконаних усіх вище операцій ми отримаємо повідомлення про успішну оплату (див. рисунок 3.16).

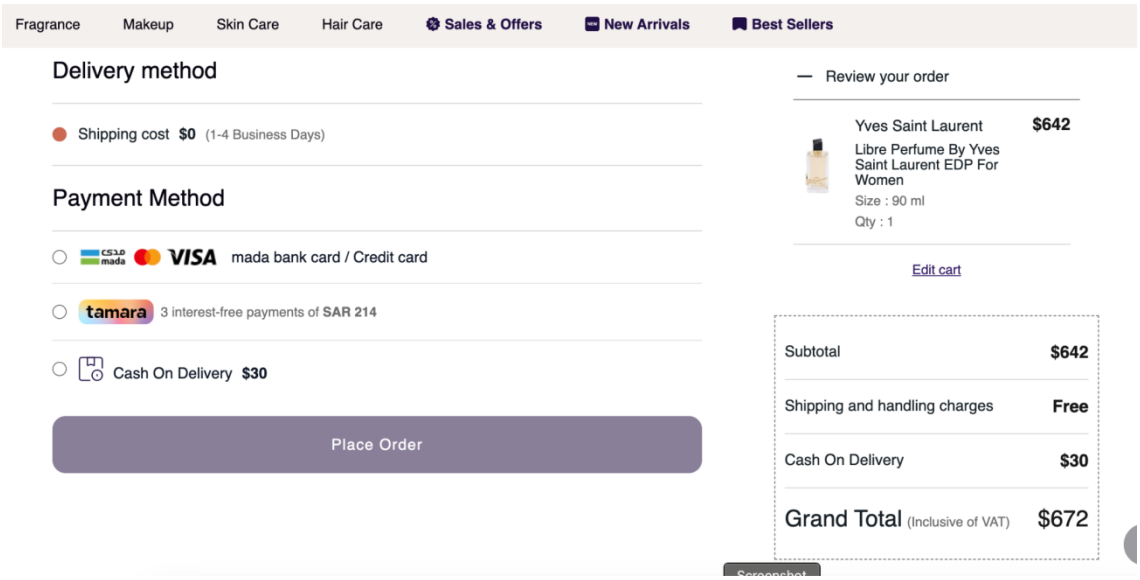


Рисунок 3.15 – Сторінка оплати

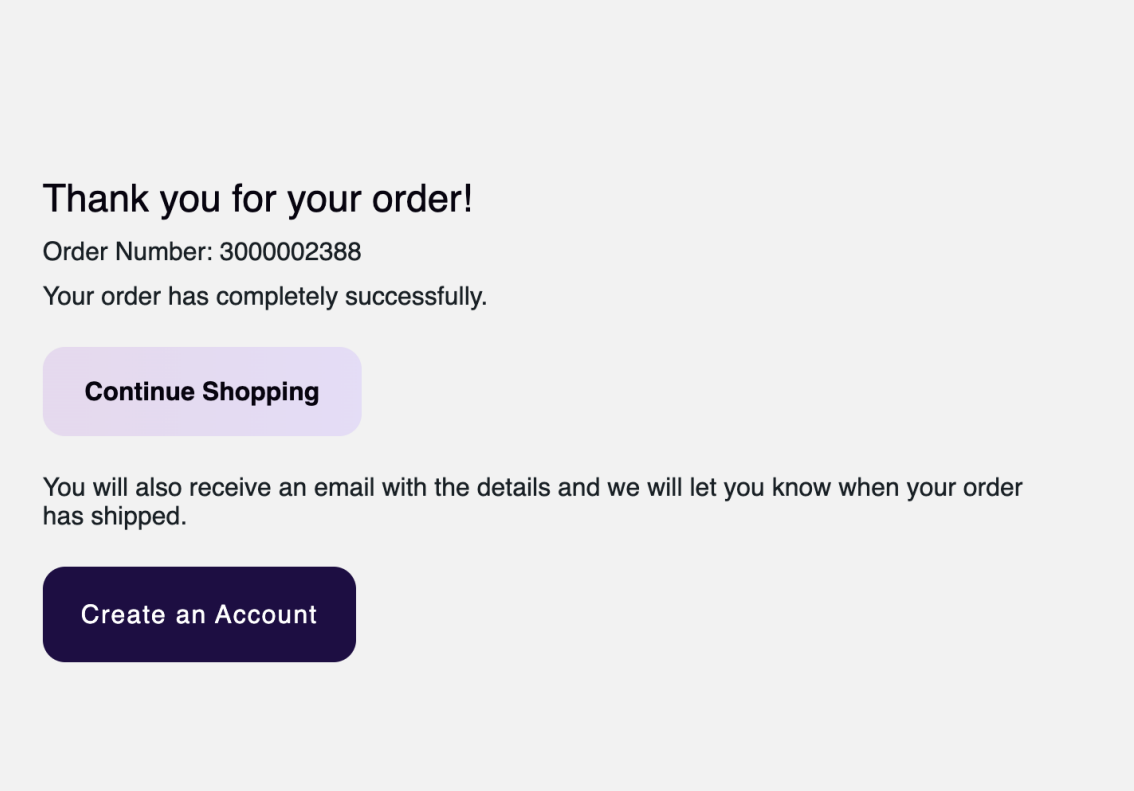


Рисунок 3.16 – Успішна оплата

Головним функціоналом який був доданий це тихнологія PWA. Для тестування необхідно завантажити додаток. Завантажується він за допомогою модального вікна яке демонструється при першому заході на сайт. Отже після встановлення PWA було додано до моїх програм (див. рисунок 3.17). І тепер я можу відкрити у вигляді нативної програми а не у браузері (див. рисунок 3.18).

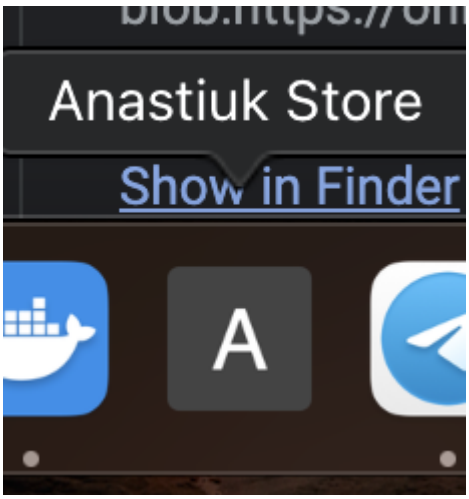


Рисунок 3.17 – Встановлений застосунок

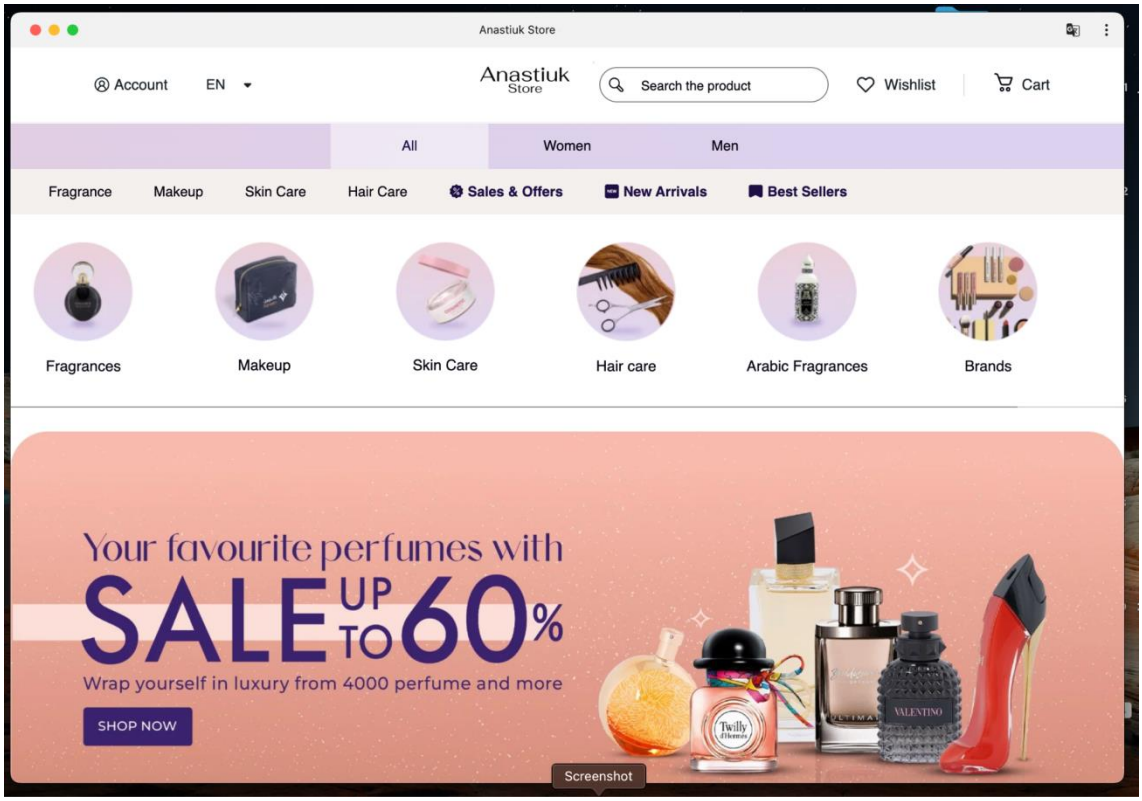


Рисунок 3.18 – Відкритий застосунок

Після проведення всебічного тестування інтернет-магазину, було підтверджено безперебійну роботу всіх його функцій і відсутність будь-яких проблем. Особливо слід відзначити високу якість модуля інтеграції з редактором продукту, який виконує свої функції бездоганно. Додатково, функціонал сповіщень про надходження нових продуктів працює на високому рівні, забезпечуючи вчасне та точне повідомлення користувачів. Також, завдяки ефективному кешуванню сторінок, швидкодія сайту вражає своєю швидкістю та плавністю. Під час тестування системи не було виявлено жодних проблем або недоліків, що підтверджує високу якість розробленого продукту.

4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1. Психологічні чинники небезпеки

На мозок і нервову систему людини безперервно діють різноманітні за кількістю і якістю подразники з внутрішнього і навколишнього середовищ. Виникнення несподіваної та напруженої ситуації призводить до порушення рівноваги між організмом і навколишнім середовищем. Наступає неспецифічна реакція організму у відповідь на цю ситуацію – стрес. Наявність мозку, нервових систем, ендокринних залоз дає можливість організму реагувати на внутрішні або зовнішні ситуації таким чином, щоб бути готовим до можливих змін.

Нервова система – це сукупність структур в організмі, яка об'єднує діяльність усіх органів і систем і забезпечує функціонування організму як єдиного ціло- го в його постійній взаємодії із зовнішнім середовищем. Вона сприймає зовнішні і внутрішні подразнення, аналізує, відбирає і перетворює сприйняту інформацію та координує функції організму.

Мала рухливість при розумовій роботі й одноманітність при фізичній праці призводять до стомлення нервової системи, що послаблює її регулюючу функцію і може спровокувати виникнення ряду хвороб Усунення фізичної утоми і нервової перевтоми настають при чергуванні одного виду діяльності з іншим, при цьому навантаження будуть відчувати по черзі різні групи нервових кліток. В умовах високої автоматизації виробництва профілактика перевтоми досягається особистою активністю працівника, його творчою зацікавленістю, регулярним чергуванням моментів праці і відпочинку.

Психіка людини – це здатність її мозку відображати об'єктивну дійсність у формі відчуттів, уявлень, думок та інших суб'єктивних образів об'єктивного світу. Вона проявляється у таких трьох видах психічних явищ: 1) психічні процеси – це короткочасні процеси отримання, переробки інформації та обміну нею (відчуття, сприйняття, пам'ять і мислення, емоції, воля); 2) психічні стани відображають порівняно тривалі душевні переживання, що впливають на життєдіяльність

людини (настрій, депресія, стрес); 3) психічні властивості – сталі душевні якості, що утворюються у процесі життєдіяльності людини і характеризують її здатність відповідати на певні дії адекватними психічними діями (темперамент, досвід, характер, здібності, інтелект). Психіка людини тісно пов'язана з безпекою її життєдіяльності. Небезпеки, які впливають на людину, не можна розцінювати ані як подію, яка породжена тільки зовнішньою стимулюючою ситуацією, ані як результат рефлекторної реакції організму людини на неї. Вплив цих небезпек зумовлюється психофізіологічними властивостями людини, які визначають її дії, вчинки, поведінку.

У процесі своєї діяльності людина використовує не тільки свої фізичні можливості, а й витрачає значні психологічні зусилля, такі як особливості характеру, волю, розумові здібності тощо. Психофізіологічними називаються небезпечні фа- ктори, зумовлені особливостями фізіології та психології людини.

Психофізіологічними факторами потенційної небезпеки постійної дії є: 1) недоліки органів відчуття (дефекти зору, слуху тощо); 2) порушення зв'язків між сенсорними та моторними центрами, внаслідок чого людина не здатна реагувати адекватно на ті чи інші зміни, що сприймаються органами відчуття; 3) дефекти координації рухів (особливо складних рухів та операцій, прийомів); 4) підвищена емоційність; 4) відсутність мотивації до трудової діяльності (незацікавленість в досягненні цілей, невдоволення оплатою праці, монотонність праці, відсутність пізнавального моменту, тобто нецікава робота, тощо).

Психофізіологічними факторами потенційної небезпеки тимчасової дії є: 1)недостатність досвіду (поява імовірної помилки, невірні дії, напруження нервово-психічної системи, побоювання припуститися помилки); 2) необережність (може призвести до ураження не лише окремої людини, а й всього колективу); 3) втома (розрізняють фізіологічне та психологічне втомлення); 4) емоційні явища (особливо конфліктні ситуації, душевні стреси, пов'язані з побутом, сім'єю, друзями, керівництвом).

На успіх діяльності особливо впливає стан людини. Будь-який вид діяльності викликає втому – зниження продуктивності діяльності через витрату

енергетич- них ресурсів організму людини. Цей стан виникає через певне ставлення людини до праці, звички до фізичного та розумового напруження. Якщо таких звичок не- має, то втома може настати ще до початку фізичного навантаження, на самому початку роботи. Втома після важкої, але потрібної людям праці, пов'язана з позитивним емоційним станом. Відпочинок, особливо активний, зміна виду діяльності поновлюють силу, створюють можливість продовження діяльності. Об'єктивним показником втомлення є уповільнення темпу роботи, а також зниження її якості. В перші дві години продуктивність праці зростає, досягаючи максимального рівня, а потім поступово знижується. Монотонна, нецікава робота призводить до того, що втома настає раніше, ніж у тих випадках, коли робота зацікавлює людину.

При перевтомі період оптимальної працездатності скорочується, а період нестійкої компенсації збільшується. Порушуються і відновні процеси в організмі. Прикмети втоми не щезають до початку роботи наступного дня. Посилюється роздратованість, реакції стають неадекватними. За наявності хронічної перевтоми: погіршується продуктивність праці; знижується опір організму до інфекції; зростає лабільність показників серцево-судинної системи; підвищується сухожилкові рефлекси, пітливість; часто зменшується маса тіла; збільшується кількість помилок, брак у роботі. Люди зі станом перевтоми характеризуються порушенням сну, відсутністю повного відновлення працездатності до наступного робочого дня, зниженням опору до дії несприятливих факторів довкілля, підвищенням нервово-емоційної збудливості.

Так як робота в іт , це розумова робота, яка вимагає багато часу проводити за комп'ютером, з постійним напруженням мозку і нервової системи, з постійною перевтомою, працівники цієї галузі в більше підвладні стресу, депресії, захворюванням серцево-судинними хворобами, нервовим зривам.

4.2. Вимоги електробезпеки при роботі на ПК

Умови й організацію праці при роботі з візуальними дисплейними терміналами усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок, що використовуються в електронно-обчислювальних машинах (ЕОМ*) колективного використання та персональних, визначають Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин (ДСанПІН 3.3.2.007-98), затверджені постановою головного державного санітарного лікаря України від 10 грудня 1998 р. № 7.

Мінімальні вимоги безпеки та захисту здоров’я під час роботи, пов’язаної з використанням екранних пристроїв незалежно від їхнього типу та моделі, визначають Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями, затверджені наказом Міністерства соціальної політики України від 14 лютого 2018 р. № 207 (далі — НПАОП 0.00-7.15-18).

Роботодавець повинен забезпечити навчання і перевірку знань з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними та проведення медичних оглядів працівників (п. 2, 6 розділу II НПАОП 0.00-7.15-18).

Національні стандарти України гармонізовані з європейськими нормами

Вимоги безпеки щодо використання комп’ютерної техніки визначають:

ДСТУ EN 41003:2014 «Обладнання, яке підключають до телекомунікаційних мереж та/або кабельних розподільчих систем. Додаткові вимоги щодо безпеки».

ДСТУ EN 60335-1:2015 «Прилади побутові та аналогічні електричні. Безпека. Частина 1. Загальні вимоги»

ДСТУ EN 60950-1:2015 «Обладнання інформаційних технологій. Безпека. Частина 1. Загальні вимоги» (далі — ДСТУ EN 60950-1:2015).

ДСТУ EN 61140:2015 «Захист проти ураження електричним струмом. Загальні аспекти щодо установок та обладнання» (далі — ДСТУ EN 61140:2015).

ДСТУ EN 62368-1:2017 «Обладнання аудіо-, відео-, інформаційних та комунікаційних технологій. Частина 1. Вимоги щодо безпеки».

Працівника, який використовує персональний комп’ютер, інструктують за цією інструкцією перед початком роботи, а потім через кожні 6 місяців. Результати інструктажу заносять до Журналу реєстрації інструктажів з питань охорони праці на робочому місці. До роботи на персональному комп’ютері допускають осіб, які пройшли інструктажі з питань охорони праці та пожежної безпеки.

Користувач зобов’язаний:

- виконувати правила внутрішнього трудового розпорядку;
- не допускати за своє робоче місце сторонніх осіб;
- не виконувати вказівок, які суперечать правилам охорони праці та пожежної безпеки;
- знати правила надання домедичної допомоги;
- знати розташування та вміти користуватись первинними засобами пожежогасіння;
- вміти працювати з ПК.

Вимоги безпеки після закінчення роботи

1. Зберегти інформацію.

2. Вимкнути ПК, монітор чи ноутбук.

3. Вимкнути стабілізатор, якщо комп’ютер під’єднаний до мережі через нього.

4. Прибрати робоче місце.

Електробезпека при роботі з персональним комп'ютером

Основні шкідливі та небезпечні фактори, що можуть впливати на організм людини під час роботи з персональним комп'ютером (ПК), такі:

- підвищений рівень електромагнітних випромінювань;
- підвищений рівень іонізуючих випромінювань;

- підвищений рівень статичної електрики;
- підвищена напруженість електростатичного поля;
- підвищена чи знижена іонізація повітря;
- підвищена яскравість світла;
- пряма і відбита блискітливість;
- підвищене значення напруги в електромережі, замикання якої може статися крізь тіло людини;
- статичні перевантаження кістково-м'язового апарату та динамічні локальні перевантаження м'язів кистей рук;
- перенапруження зорового аналізатора;
- розумове перенапруження;
- емоційні перевантаження;
- монотонність праці.

Експлуатаційні заходи.

Необхідно дотримуватися правил техніки безпеки при роботі з високою напругою і наступних запобіжних засобів:

- монтаж, обслуговування, ремонт і наладка ЕОМ, заміна деталей, пристосувань, блоків повинна здійснюватися тільки при повному відключенні живлення;
- заземлені конструкції приміщення мають бути надійно захищені діелектричними щитками або сітками від випадкового дотику.

ВИСНОВКИ

Написання дипломної роботи на тему «Розробка Progressive Web Apps» дозволило провести дослідження цієї перспективної технології та виявити її переваги в порівнянні з традиційними сайтами. Прогресивні веб-програми є важливим кроком в еволюції веб-розробки, оскільки вони надають користувачам більше функцій, доступності та безпеки.

Однією з головних переваг Progressive Web Apps є можливість роботи в автономному режимі, що дозволяє користувачам використовувати програми незалежно від доступу до Інтернету. Це особливо корисно для тих, хто має обмежений доступ до мережі, але все одно хоче використовувати програми. Крім того, Progressive Web Apps мають більш високу продуктивність і швидкість завантаження, що забезпечує комфортні умови роботи з ними.

Технологія також забезпечує більший захист для користувачів завдяки використанню протоколу HTTPS, який забезпечує шифрування даних і захист від зловмисних атак. Це робить Progressive Web Apps надійними та безпечними у використанні.

Розробка Progressive Web Apps має великий потенціал для подальшого розвитку веб-технологій. Їх більш висока продуктивність, доступність і безпека забезпечують користувачам зручні умови для використання програм на різних пристроях. Вони є однією з головних тенденцій веб-розробки, і їх розвиток має важливе значення для подальшого розвитку сучасних веб-технологій.

У майбутньому можно глибше провести додаткові дослідження технології Progressive Web Apps і ознайомитися з її оновленнями та новими функціями. Використання Progressive Web Apps може стати стандартом для багатьох веб-додатків, забезпечуючи зручну та ефективну взаємодію користувача з веб-сервісами.

Завдяки своїм перевагам прогресивні веб-програми мають потенціал змінити спосіб розробки та використання веб-програм. Вони поєднують у собі

найкращі функції веб-сайтів із зручністю та функціональністю мобільних додатків. Користувачам не потрібно встановлювати додатки з магазинів, що спрощує процес доступу до необхідного функціоналу. Прогресивні веб-програми можна запускати безпосередньо з веб-браузера на різних пристроях, що робить їх більш універсальними та доступними.

Крім того, прогресивна розробка веб-додатків полегшує підтримку та оновлення програм. Оскільки вони мають загальну кодову базу, зміни й оновлення можна вносити централізовано й одразу бачити їхній вплив на всіх пристроях, які використовують програму. Це допомагає швидко впроваджувати нові функції та виправляти помилки.

Прогресивні веб-програми також є більш економічно ефективними для розробників. Замість того, щоб розробляти окремі програми для різних платформ (таких як Android та iOS), розробники можуть зосередитися на розробці єдиної програми, яка працюватиме на всіх пристроях, які підтримують веб-браузери. Це зменшує витрати на розробку та обслуговування програм, забезпечуючи більшу ефективність процесу.

Таким чином, прогресивна розробка веб-додатків має великий потенціал і забезпечує численні переваги як для користувачів, так і для розробників. Їх універсальність, офлайн-доступність, продуктивність і безпека роблять їх привабливими для використання на різних пристроях і платформах. Завдяки прогресу веб-технологій і зростанню популярності прогресивних веб-додатків вони можуть стати майбутнім стандартом розробки веб-додатків і відкрити нові можливості для покращення взаємодії користувача з цифровим середовищем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Використання Docker" – Едріен Моует, 2008 – 12 с.
2. "Node.js in Action" – Майкл Маккаллог, 2017 – 135 с.
3. "Програмування мовою PHP" – Олексій Васильєв, 2022 – 43 с.
4. "React і Redux: функціональна веброзробка" - Бенкс Алекс, 2018 – 12 с.
5. "Linux керівництво системного адміністратора" - Еві Немет, 2021 – 33 с.
6. "Вивчаємо SQL. Генерація та обробка даних" - Алан Больє, 2017 – 43 с.
7. "RESTful Web APIs" - Леонардо Рідчардсон, 2021 – 76 с.
8. "GraphQL. Мова запитів для сучасних додатків" - Бенкс Алекс, 2014 – 455 с.
9. "Progressive Web Application Development" - Кріс Лав, 2013 – 245 с.
10. "Learning Test-Driven Development" - Салем Седдік, 2020 – 211 с.
11. "Simplify Testing with React Testing Library" - Скот Крамп, 2018 – 65 с.
12. "Fullstack React Native: The Complete Guide to React Native and Friends" - Софія Шапшал, 2019 – 347 с.
13. "Mastering Node.js" - Сандро Пасцоне, Кевін Фаулер,2017 – 32 с.
14. "Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node" - Вамсі Кондепуді, 2017 – 674 с.
15. "JavaScript: The Definitive Guide" - Девід Фланаган, 2020 – 234 с.
16. Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці. – К.:Каравела, 2014. – 408 с.
17. Жидецький В.Ц. Охорона праці користувачів комп'ютерів. – Львів: Афіша, 2001.– 176 с.
18. Пасічник В.В, Пасічник О.В, Басюк Т.М., Думанський Н.О. Основи інформаційних технологій. – Львів: Новий світ-2000, 2020 – 390 с.

ДОДАТКИ

Додаток А

Фрагмент коду компоненти “Checkout”

```
import React, { Fragment, Suspense, useEffect, useState } from 'react';
import { shape, string } from 'prop-types';
import { FormattedMessage, useIntl } from 'react-intl';
import { AlertCircle as AlertCircleIcon } from 'react-feather';
import { useHistory, useLocation } from 'react-router-dom';
import { useWindowSize, useToasts } from '@magento/peregrine';
import {
    CHECKOUT_STEP,
    useCheckoutPage
} from '../../talons/CheckoutPage/useCheckoutPage';
import { useStyle } from '@lib/anastiuk/lib/classify';
import Button from '../Button';
import { StoreTitle } from '../Head';
import Icon from '@lib/anastiuk/lib/components/Icon';
import AddressBook from './AddressBook';
import OrderSummary from './OrderSummary';
import PaymentInformation from './PaymentInformation';
import ShippingMethod from './ShippingMethod';
import ShippingInformation from './ShippingInformation';
import OrderConfirmationPage from './OrderConfirmationPage';
import ItemsReview from './ItemsReview';
import defaultClasses from './checkoutPage.module.css';
import { Accordion } from '../Accordion';
import Section from './section';
import Price from '../Price';
import { useAuthModal } from '../AuthModal/useAuthModal';
import CheckoutAuth from './Auth/checkoutAuth';
import LoadingIndicator from '../LoadingIndicator';
import LoadingModal from './LoadingModal';
import { AddPaymentInfo } from '../GoogleTagManager';

const CouponCode = React.lazy(() =>
    import('../CartPage/PriceAdjustments/CouponCode')
);

const errorIcon = <Icon src={AlertCircleIcon} size={20} />;
```

```
const CheckoutPage = props => {
    const { classes: propClasses } = props;
    const { formatMessage } = useIntl();
    const location = useLocation();
    const one = '1';
    const two = '2';
    const defaultSelectAddressMessage = formatMessage({
        defaultMessage: 'Please select or add you address',
        id: 'checkoutPage.defaultSelectAddressMessage'
    });
    const { handleOpen: openAuthDialog } = useAuthModal();
    const [addressTitle, setAddressTitle] = useState(
        defaultSelectAddressMessage
    );
    const [openReview, setOpenReview] = useState(false);
    const [priceData, setPriceData] = useState({});
    const talonProps = useCheckoutPage(priceData);
    const [mapAddress, setMapAddress] = useState(null);
    const [paymentErrorMessage, setPaymentErrorMessage] = useState(null);
    const [payfortError, setPayfortError] = useState(null);

    const {
        /**
         * Enum, one of:
         * SHIPPING_ADDRESS, SHIPPING_METHOD, PAYMENT, REVIEW
         */
        activeContent,
        cartItems,
        checkoutStep,
        error,
        hasError,
        isGuestCheckout,
        isLoading,
        isUpdating,
        isSignedIn,
        orderDetailsData,
        orderDetailsLoading,
        orderNumber,
```

```
    placeOrderLoading,
    setCheckoutStep,
    setIsUpdating,
    setShippingInformationDone,
    scrollShippingInformationIntoView,
    setShippingMethodDone,
    scrollShippingMethodIntoView,
    setPaymentInformationDone,
    resetReviewOrderButtonClicked,
    handleReviewOrder,
    reviewOrderButtonClicked,
    payfortFormRef,
    payfortVaultRef,
    appleTransaction,
    appleStatus,
    handleApplePayClick,
    setPayfortFormData,
    setPaymentMethod,
    isTamaraPayment,
    isDisabledPlaceOrder,
    setIsDisabledPlaceOrder,
    payfortFormString,
    payfortFormData,
    setIsGuestCheckout,
    activeStep,
    setActiveStep,
    isOpenLoading,
    openLoadingModal,
    paymentMethod,
    setIsUseSavedCard,
    isUseSavedCard
} = talonProps;

const [, { addToast }] = useToasts();
const history = useHistory();

const handleClick = () => {
    //Show loading indicator
    //Start place order sequence and handle apple transaction
```

```
    AddPaymentInfo(paymentMethod, cartItems);
    if (paymentMethod === 'aps_apple') {
        handleReviewOrder();
        handleApplePayClick(priceData);
    } else if (paymentMethod === 'aps_fort_cc' && !isUseSavedCard) {
        if (payfortFormRef?.current) {
            payfortFormRef.current.submitForm();
        } else {
            setPayfortError(true);
        }
    } else if (paymentMethod === 'aps_fort_cc' && isUseSavedCard) {
        if (payfortVaultRef.current) {
            payfortVaultRef.current.submitForm();
        } else {
            setPayfortError(true);
        }
    } else {
        openLoadingModal('loading');
        handleReviewOrder();
    }
};

const saveAddress = address => {
    setMapAddress(address);
};

useEffect(() => {
    if (isUseSavedCard !== null) {
        setPayfortError(false);
    }
}, [isUseSavedCard]);

useEffect(() => {
    if (location?.state?.paymentError) {
        setPaymentErrorMessage(true);
    }
}, [location?.state?.paymentError]);

useEffect(() => {
```

```
        if (hasError) {
            const message =
                error && error.message
                    ? error.message
                    : formatMessage({
                          defaultMessage:
                              'Oops! An error occurred while submitting.
Please try again.',
                          id: 'checkoutPage.errorSubmit'
                      });
            addToast({
                type: 'error',
                icon: errorIcon,
                message,
                dismissable: true,
                timeout: 7000
            });

            if (process.env.NODE_ENV !== 'production') {
                console.error(error);
            }
        }
    }, [addToast, error, formatMessage, hasError]);

    useEffect(() => {
        if (paymentErrorMessage) {
            const message = formatMessage({
                defaultMessage:
                    'Oops! An error occurred. Please try again or choose
another payment method.',
                id: 'tamara.errorPayment'
            });
            addToast({
                type: 'error',
                icon: errorIcon,
                message,
                dismissable: true,
                timeout: 7000
            });
```

```
            history.replace({ state: {} });
            setPaymentErrorMessage(false);
        }
    }, [
        addToast,
        error,
        formatMessage,
        history,
        location,
        paymentErrorMessage
    ]);

    const classes = useStyle(defaultClasses, propClasses);

    const windowSize = useWindowSize();
    const isMobile = windowSize.innerWidth <= 768;

    let checkoutContent;

    if (
        (orderNumber &&
            orderDetailsData &&
            !isTamaraPayment &&
            !payfortFormData &&
            !appleTransaction &&
            appleStatus === null) ||
        (appleTransaction && appleStatus)
    ) {
        return (
            <OrderConfirmationPage
                data={orderDetailsData}
                orderNumber={orderNumber}
            />
        );
    } else if (payfortFormString) {
        const form =
            payfortFormString.createTransactionPayfort?.form ||
            payfortFormString.createTransactionPayfortVault?.form;
        const inputs = form?.inputs.map((item, ind) => {
```

```
        if (item !== null) {
            return (
                <input
                    key={ind}
                    id={item.name}
                    name={item.name}
                    value={item.value}
                />
            );
        }
    });

    const payfortForm = (
        <form
            style={{ display: 'none' }}
            id={form.id}
            action={form.action}
            method={form.method}
        >
            {inputs}
            <button id={'frm_payfort_fort_button'} type={'submit'} />
        </form>
    );
    return <div>{payfortForm}</div>;
} else if (isLoading) {
    return <LoadingIndicator />;
} else if (!isSignedIn && !isGuestCheckout) {
    return (
        <div className={classes.checkoutAuth}>
            <CheckoutAuth setIsGuestCheckout={setIsGuestCheckout} />
        </div>
    );
} else {
    const signInContainerElement =
        isGuestCheckout && activeStep === 1 ? (
            <div className={classes.signInWrapper}>
                <div className={classes.signInContainer}>
                    <Button
                        className={classes.signInButton}
```

```
                    data-cy="CheckoutPage-signInButton"
                    onClick={openAuthDialog}
                    priority="normal"
                >
                    <FormattedMessage
                        defaultMessage={'Sign in or Sign up'}
                        id={'checkoutPage.signInButton'}
                    />
                </Button>
            </div>
        </div>
    ) : null;

const shippingMethodSection =
    checkoutStep >= CHECKOUT_STEP.SHIPPING_METHOD ? (
        <ShippingMethod
            pageIsUpdating={isUpdating}
            onSave={setShippingMethodDone}
            onSuccess={scrollShippingMethodIntoView}
            setPageIsUpdating={setIsUpdating}
        />
    ) : null;

const paymentInformationSection =
    checkoutStep >= CHECKOUT_STEP.PAYMENT ? (
        <PaymentInformation
            onSave={setPaymentInformationDone}
            checkoutError={error}
            resetShouldSubmit={resetReviewOrderButtonClicked}
            setCheckoutStep={setCheckoutStep}
            shouldSubmit={reviewOrderButtonClicked}
            payfortFormRef={payfortFormRef}
            payfortVaultRef={payfortVaultRef}
            setPayfortFormData={setPayfortFormData}
            priceData={priceData}
            setPaymentMethod={setPaymentMethod}
            setPageIsUpdating={setIsUpdating}
            setIsDisabledPlaceOrder={setIsDisabledPlaceOrder}
            isOpenLoading={isOpenLoading}
```

```
                setIsUseSavedCard={setIsUseSavedCard}
                payfortError={payfortError}
            />
        ) : null;

    const itemsReview = (
        <Accordion classes={{ root: classes.itemsReview }}>
            <Section
                caller={'ItemsReview'}
                id={'ShippingInformation'}
                isMobile={isMobile}
                title={
                    !openReview ? (
                        <div className={classes.reviewTitleContainer}>
                            <div className={classes.reviewTitle}>
                                <FormattedMessage
                                    id={'checkoutPage.showOrderSummary'}
                                    defaultMessage={'Review your order'}
                                />
                            </div>
                            {priceData.total ? (
                                <div className={classes.totalPrice}>
                                    <Price
                                        value={priceData.total.value}
                                        currencyCode={
                                            priceData.total.currency
                                        }
                                    />
                                </div>
                            ) : null}
                        </div>
                    ) : (
                        <div className={classes.reviewTitleContainer}>
                            <div className={classes.reviewTitle}>
                                <FormattedMessage
                                    id={'checkoutPage.showOrderSummary'}
                                    defaultMessage={'Review your order'}
                                />
                            </div>
```

```
                        {priceData.total ? (
                            <div className={classes.totalPrice}>
                                <Price
                                    value={priceData.total.value}
                                    currencyCode={
                                        priceData.total.currency
                                    }
                                />
                            </div>
                        ) : null}
                    </div>
                )
            }
            setOpenReview={setOpenReview}
        >
            <div className={classes.items_review_container}>
                <ItemsReview
                    isMobile={isMobile}
                    isUpdating={isUpdating}
                    priceData={priceData}
                />
            </div>
        </Section>
    </Accordion>
);

const placeOrderButton = (
    <Button
        onClick={handleClick}
        priority="high"
        className={
            isUpdating ||
            placeOrderLoading ||
            orderDetailsLoading ||
            !paymentMethod ||
            checkoutStep < CHECKOUT_STEP.PAYMENT ||
            isDisabledPlaceOrder
                ? classes.placeOrderDisabled
                : classes.place_order_button
```

```
                }
                data-cy="CheckoutPage-placeOrderButton"
                disabled={
                    isUpdating ||
                    placeOrderLoading ||
                    orderDetailsLoading ||
                    !paymentMethod ||
                    checkoutStep < CHECKOUT_STEP.PAYMENT ||
                    isDisabledPlaceOrder
                }
            >
                <FormattedMessage
                    id={'checkoutPage.placeOrder'}
                    defaultMessage={'Place Order'}
                />
            </Button>
        );

        // If we're on mobile we should only render price summary in/after
review.
        const shouldRenderPriceSummary = !(
            isMobile && checkoutStep < CHECKOUT_STEP.REVIEW
        );

        const orderSummary = shouldRenderPriceSummary ? (
            <div className={classes.summaryContainer}>
                {itemsReview}
                <OrderSummary isUpdating={isUpdating} />
                <div className={classes.couponCode}>
                    <Suspense fallback={<LoadingIndicator />}>
                        <CouponCode setIsCartUpdating={() => {}} />
                    </Suspense>
                </div>
            </div>
        ) : null;

        const headerText = (
            <FormattedMessage
                defaultMessage={'Checkout'}
```

```
        id={'checkoutPage.checkout'}
    />
);

const checkoutContentClass =
    activeContent === 'checkout'
        ? classes.checkoutContent
        : classes.checkoutContent_hidden;

const addressBookElement = !isGuestCheckout ? (
    <AddressBook
        activeContent={activeContent}
        onSuccess={scrollShippingInformationIntoView}
        mapAddress={mapAddress}
        setMapAddress={setMapAddress}
        setIsDisabledPlaceOrder={setIsDisabledPlaceOrder}
        saveAddress={saveAddress}
        setActiveStep={setActiveStep}
        onSave={setShippingInformationDone}
        setAddressTitle={setAddressTitle}
        setCheckoutStep={setCheckoutStep}
    />
) : null;

const stepsNavigation = (
    <div className={classes.stepsNavigation}>
        <div
            className={
                activeStep >= 1
                    ? classes.stepsButtonActive
                    : classes.stepsButtonDisabled
            }
        >
            <div>
                <span className={classes.stepNumber}>{one}</span>
                <div className={classes.stepContainer}>
                    <div className={classes.stepTitle}>
                        <FormattedMessage
                            defaultMessage={'Shipping Address'}
```

```
                                id={'checkoutPage.shippingAddress'}
                            />
                        </div>
                        <div className={classes.stepContent}>
                            {addressTitle}
                        </div>
                    </div>
                </div>
                <div
                    className={
                        checkoutStep < CHECKOUT_STEP.SHIPPING_METHOD
                            ? classes.stepsButtonDisabled
                            : classes.stepsButtonActive
                    }
                >
                    <button
                        disabled={checkoutStep <
CHECKOUT_STEP.SHIPPING_METHOD}
                        onClick={() => {
                            setCheckoutStep(CHECKOUT_STEP.SHIPPING_METHOD);
                            setActiveStep(2);
                        }}
                    >
                        <span className={classes.stepNumber}>{two}</span>
                        <div className={classes.stepContainer}>
                            <div className={classes.stepTitle}>
                                <FormattedMessage
                                    defaultMessage={'Checkout'}
                                    id={'checkoutPage.titleCheckout'}
                                />
                            </div>
                            <div className={classes.stepContent}>
                                <FormattedMessage
                                    defaultMessage={
                                        'Card, ApplePay, Tamara or pay on
delivery'
                                    }
                                    id={'checkoutPage.step2Content'}
```

```
                                />
                            </div>
                        </div>
                    </button>
                </div>
            </div>
        );

        checkoutContent = (
            <Fragment>
                <div className={classes.heading_container}>
                    {!isMobile ? (
                        <h2 className={classes.heading}>{headerText}</h2>
                    ) : null}
                </div>
                {stepsNavigation}
                <div className={checkoutContentClass}>
                    {signInContainerElement}
                    {checkoutStep >= CHECKOUT_STEP.SHIPPING_ADDRESS &&
                    activeStep === 1 ? (
                        <div
    className={classes.shipping_information_container}>
                            {addressBookElement}
                            <ShippingInformation
                                onSave={setShippingInformationDone}
                                onSuccess={scrollShippingInformationIntoView}
                                setAddressTitle={setAddressTitle}
                                isMobile={isMobile}
                                mapAddress={mapAddress}
                                saveAddress={saveAddress}
                                setActiveStep={setActiveStep}
                            />
                        </div>
                    ) : null}
                    {activeStep > 1 ? (
                        <Fragment>
                            {isMobile ? (
                                <Fragment>
                                    {itemsReview}
```

```
                                    <Fragment>
                                        <div className={classes.couponCode}>
                                            <Suspense
                                                fallback={<LoadingIndicator
/>}
                                            >
                                                <CouponCode
                                                    setIsCartUpdating={() =>
{}}
                                                />
                                            </Suspense>
                                        </div>
                                    </Fragment>
                                </Fragment>
                            ) : null}
                            <div>
                                <div
                                    className={
                                        classes.shipping_method_container
                                    }
                                >
                                    {shippingMethodSection}
                                </div>
                                <div
                                    className={
                                        classes.payment_information_container
                                    }
                                >
                                    {paymentInformationSection}
                                </div>
                                {!isMobile ? (
                                    <div
classNameclasses.placeOrderButton}>
                                        {placeOrderButton}
                                    </div>
                                ) : null}
                            </div>
                        </Fragment>
                    ) : null}
```

```
                    {orderSummary}
                </div>
                <LoadingModal isOpen={isOpenLoading} />
            </Fragment>
        );
    }

    return (
        <div className={classes.container}>
            <div className={classes.root} data-cy="CheckoutPage-root">
                <StoreTitle>
                    {formatMessage({
                        id: 'checkoutPage.titleCheckout',
                        defaultMessage: 'Checkout'
                    })}
                </StoreTitle>
                {checkoutContent}
            </div>
            {checkoutStep >= CHECKOUT_STEP.SHIPPING_METHOD && activeStep > 1
? (
                <div className={classes.checkoutMobileFooter}>
                    <div className={classes.summaryContainer}>
                        <OrderSummary
                            isUpdating={isUpdating}
                            caller={'checkoutMobileFooter'}
                            setPriceData={setPriceData}
                        />
                    </div>
                    <div className={classes.footerPlaceOrder}>
                        <Button
                            onClick={handleClick}
                            priority="low"
                            className={
                                isUpdating ||
                                placeOrderLoading ||
                                orderDetailsLoading ||
                                !paymentMethod ||
                                checkoutStep < CHECKOUT_STEP.PAYMENT ||
                                isDisabledPlaceOrder
```

```
                                    ? classes.placeOrderDisabled
                                    : classes.place_order_button
                                }
                                data-cy="CheckoutPage-placeOrderButton"
                                disabled={
                                    isUpdating ||
                                    placeOrderLoading ||
                                    orderDetailsLoading ||
                                    !paymentMethod ||
                                    checkoutStep < CHECKOUT_STEP.PAYMENT ||
                                    isDisabledPlaceOrder
                                }
                            >
                                <FormattedMessage
                                    id={'checkoutPage.placeOrder'}
                                    defaultMessage={'Place Order'}
                                />
                            </Button>
                        </div>
                    </div>
                ) : null}
            </div>
        );
};

export default CheckoutPage;

CheckoutPage.propTypes = {
    classes: shape({
        root: string,
        checkoutContent: string,
        checkoutContent_hidden: string,
        heading_container: string,
        heading: string,
        cartLink: string,
        stepper_heading: string,
        shipping_method_heading: string,
        payment_information_heading: string,
        signInContainer: string,
```

```
        signInLabel: string,
        signInButton: string,
        empty_cart_container: string,
        shipping_information_container: string,
        shipping_method_container: string,
        payment_information_container: string,
        price_adjustments_container: string,
        items_review_container: string,
        summaryContainer: string,
        formErrors: string,
        review_order_button: string,
        place_order_button: string
    })
};
```

Додаток Б

Фрагменти коду компоненти “PaymentMethods”

```
import React from 'react';
import { shape, string, bool, func } from 'prop-types';
import { FormattedMessage, useIntl } from 'react-intl';
import { usePaymentMethods } from
'../../../talons/CheckoutPage/PaymentInformation/usePaymentMethods';
import { useStyle } from '@lib/anastiuk/lib/classify';
import RadioGroup from '@lib/anastiuk/lib/components/RadioGroup';
import Radio from '@lib/anastiuk/lib/components/RadioGroup/radio';
import defaultClasses from './paymentMethods.module.css';
import payments from './paymentMethodCollection';
import Image from '@lib/anastiuk/lib/components/Image';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faApple } from '@fortawesome/free-brands-svg-icons';
import Price from '../../Price';
import CmsBlockGroup from '@lib/anastiuk/lib/components/CmsBlock';
import arTamaraLogo from '../../../images/tamaraImages/tamara_ar_logo.png';
import enTamaraLogo from '../../../images/tamaraImages/tamara_en_logo.png';
import LoadingIndicator from '../../LoadingIndicator';

const PaymentMethods = props => {
    const {
        classes: propClasses,
        onPaymentSuccess,
        resetShouldSubmit,
        shouldSubmit,
        payfortFormRef,
        payfortVaultRef,
        setPayfortFormData,
        handlePaymentMethodSave,
        priceData,
        setPaymentMethod,
        setIsUseSavedCard,
        payfortError
    } = props;

    const { formatMessage, locale } = useIntl();
```

```
const classes = useStyle(defaultClasses, propClasses);
const talonProps = usePaymentMethods({
    handlePaymentMethodSave,
    total: priceData?.total?.value,
    setPaymentMethod
});

const numberOfInstalments = 3;
const grandTotal = priceData?.payment?.value
    ? (priceData?.total?.value - priceData?.payment?.value) * 100
    : priceData?.total?.value * 100;
const mod = grandTotal % numberOfInstalments;
const payForEachMonth = (grandTotal - mod) / numberOfInstalments / 100;
const imageSrc = isArLocale ? arTamaraLogo : enTamaraLogo;
const tamaraHref = isArLocale
    ? 'https://tamara.co'
    : 'https://tamara.co/en/index.html';

const {
    availablePaymentMethods,
    currentSelectedPaymentMethod,
    isLoading,
    handleChangePaymentMethod,
    tammaraLimit
} = talonProps;
console.log(tammaraLimit)
if (isLoading) {
    return null;
}

        let MethodTitle;
        if (code === 'tamara_pay_by_instalments') {
            MethodTitle = (
                <div className={classes.tamaraTitle}>
                    <div className={classes.tamaraLink}>
                        <a href={tamaraHref} target={'_blank'}>
                            <Image
                                alt={formatMessage({
                                    defaultMessage: 'Shop',
```

```
                                id: 'footer.shop'
                            })}
                            height={22}
                            src={imageSrc}
                            width={100}
                        />
                    </a>
                </div>
                <div className={classes.tamaraLabel}>
                    <FormattedMessage
                        defaultMessage={'3 دفعات بدون فوائد '}
                        id={'tamara.label'}
                    />
                    <span className={classes.price}>
                        <Price
                            currencyCode={priceData?.total?.currency}
                            value={payForEachMonth}
                        />
                    </span>
                </div>
            </div>
        );
    } else if (code === 'aps_apple') {
        MethodTitle = (
            <div className={classes.applePayTitle}>
                <FontAwesomeIcon icon={faApple} />
                {formatMessage({
                    defaultMessage: 'Pay',
                    id: 'applePay.paymentTitle'
                })}
                <div className={classes.applePayName}>
                    <FormattedMessage
                        defaultMessage={'ApplePay'}
                        id={'applePay.applePay'}
                    />
                </div>
            </div>
        );
    } else if (code === 'cashondelivery') {
```

```
                MethodTitle = (
                    <div className={classes.cashOnDeliveryTitle}>
                        <svg
                            width="27"
                            height="27"
                            viewBox="0 0 29 29"
                            fill="none"
                            xmlns="http://www.w3.org/2000/svg"
                        >
                            <path
                                fillRule="evenodd"
                                clipRule="evenodd"
                                d="M0.9375 4.16667C0.9375 2.38325 2.38325 2.38325
0.9375 4.16667 0.9375H24.8333C26.6167 0.9375 28.0625 2.38325 28.0625
4.16667V13.8542H26.7708V4.16667C26.7708 3.09661 25.9033 2.22917 24.8333
2.22917H4.16667C3.09661 2.22917 2.22917 3.09661 2.22917
4.16667V24.8333C2.22917 25.9033 3.09661 26.7708 4.16667
26.7708H13.8542V28.0625H4.16667C2.38325 28.0625 0.9375 26.6167 0.9375
24.8333V4.16667Z"
                                fill="#200D45"
                            />
                            <path
                                fillRule="evenodd"
                                clipRule="evenodd"
                                d="M9.97916 1.58333C9.97916 1.22665 10.2683
0.9375 10.625 0.9375H18.375C18.7316 0.9375 19.0208 1.22665 19.0208
1.58333V10.625C19.0208 10.8326 18.921 11.0275 18.7527 11.1489C18.5842 11.2703
18.3678 11.3033 18.1708 11.2377L14.5 10.0141L10.8292 11.2377C10.6323 11.3033
10.4158 11.2703 10.2474 11.1489C10.079 11.0275 9.97916 10.8326 9.97916
10.625V1.58333ZM11.2708 2.22917V9.72896L14.2958 8.72064C14.4283 8.67646
14.5717 8.67646 14.7042 8.72064L17.7292 9.72896V2.22917H11.2708Z"
                                fill="#200D45"
                            />
                            <path
                                fillRule="evenodd"
                                clipRule="evenodd"
                                d="M20.9583
22.8958V20.3125H22.25V22.8958H20.9583Z"
                                fill="#200D45"
```

```
                            />
                            <path
                                fillRule="evenodd"
                                clipRule="evenodd"
                                d="M21.6042 16.4375C18.7508 16.4375 16.4375
18.7507 16.4375 21.6041C16.4375 24.4576 18.7508 26.7708 21.6042
26.7708C24.4576 26.7708 26.7708 24.4576 26.7708 21.6041C26.7708 18.7507
24.4576 16.4375 21.6042 16.4375ZM15.1458 21.6041C15.1458 18.0373 18.0374
15.1458 21.6042 15.1458C25.171 15.1458 28.0625 18.0373 28.0625
21.6041C28.0625 25.171 25.171 28.0625 21.6042 28.0625C18.0374 28.0625 15.1458
25.171 15.1458 21.6041Z"
                                fill="#200D45"
                            />
                        </svg>
                        {title}
                        {payment_fee.value ? (
                            <span className={classes.price}>
                                <Price
                                    currencyCode={payment_fee?.currency}
                                    value={payment_fee.value}
                                />
                            </span>
                        ) : null}
                    </div>
                );
            } else if (code === 'aps_fort_cc') {
                MethodTitle = (
                    <div className={classes.payFortTitle}>
                        <div className={classes.paymentMethodsContainer}>
                            <CmsBlockGroup
                                identifiers={['payment-methods-checkout']}
                            />
                        </div>
                        {formatMessage({
                            defaultMessage: 'بطاقة مدى البنكية/ بطاقة ائتمان',
                            id: 'payFort.paymentTitle'
                        })}
                    </div>
                );
```

```
} else {
    MethodTitle = title;
}

const id = `paymentMethod--${code}`;
const isSelected = currentSelectedPaymentMethod === code;

const PaymentMethodComponent = payments[code];

const renderedComponent = isSelected ? (
    <PaymentMethodComponent
        tammaraLimit={tammaraLimit}
        onPaymentSuccess={onPaymentSuccess}
        resetShouldSubmit={resetShouldSubmit}
        shouldSubmit={shouldSubmit}
        payfortFormRef={payfortFormRef}
        payfortVaultRef={payfortVaultRef}
        setPayfortFormData={setPayfortFormData}
        code={code}
        priceData={priceData}
        setIsUseSavedCard={setIsUseSavedCard}
        payfortError={payfortError}
    />
) : null;

return (
    <div key={code} className={classes.payment_method}>
        <Radio
            checked={isSelected}
            classes={{
                label: classes.radio_label
            }}
            id={id}
            label={MethodTitle}
            value={code}
        />
        {renderedComponent}
    </div>
);
```

```
        })
        .filter(paymentMethod => !!paymentMethod);

    const noPaymentMethodMessage = !radios.length ? <LoadingIndicator /> :
null;

    return (
        <div className={classes.root}>
            <RadioGroup
                classes={{ root: classes.radio_group }}
                field={'selectedPaymentMethod'}
                initialValue={''}
                onChange={event =>
                    handleChangePaymentMethod(event.target.value)
                }
            >
                {radios}
            </RadioGroup>
            {noPaymentMethodMessage}
        </div>
    );
};

PaymentMethods.propTypes = {
    classes: shape({
        payment_method: string,
        radio_label: string,
        root: string
    }),
    onPaymentError: func,
    onPaymentSuccess: func,
    resetShouldSubmit: func,
    selectedPaymentMethod: string,
    shouldSubmit: bool
};

export default PaymentMethods;
```

Додаток В

Диск з матеріалами бакалаврської