Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для обміну речами. Front-end та мобільний застосунок.

(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-7

Кащенко Ю. Є.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доцент. кафедри ПІ Ворочек О. Г.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З. В. Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Кащенко Юхим Єгорович
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для обміну речами. Front-end та мобільний застосунок.
Затверджена наказом по університету від 20.05.2024 р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2024

3. Вихідні дані до роботи Розробити клієнтську частину програмної системи для обміну речами з використанням мови TypeScript та бібліотеки React.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	15.04.2024	*виконано*
2	Створення специфікації ПЗ	22.04.2024	*виконано*
3	Проектування ПЗ	30.04.2024	*виконано*
4	Розробка ПЗ	11.05.2024	*виконано*
5	Тестування ПЗ	01.06.2024	*виконано*
6	Оформлення пояснювальної записки	07.06.2024	*виконано*
7	Підготовка презентації та доповіді	10.06.2024	*виконано*
8	Нормоконтроль, рецензування	12.06.2024	*виконано*
9	Здача роботи у електронний архів	14.05.2024	*виконано*
10	Попередній захист	16.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	19.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка) _____ Кащенко Ю. Є.
(підпис)

Керівник роботи _____ доцент кафедри ПІ Ворочек О. Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка кваліфікаційної роботи бакалавра: 68 с., 49 рис., 0 табл., 4 додаток, 10 джерел.

БАРТЕР, КОРИСТУВАЧ, ОБМІН, ПРОГРАМНА СИСТЕМА, ТОВАР, УГОДА, TYPESCRIPT, REACT

Об’єкт розробки – програмна система для обміну речами.

Мета розробки – створення ефективної та зручної платформи, яка дає змогу користувачам шукати речі, які їм потрібні, та віддавати речі, які їм більше не потрібні, в обмін на іншу річ, яка не потрібна комусь іншому.

Метод рішення – середовище розробки та Visual Studio Code, мова програмування TypeScript, бібліотека React, бібліотека CSS класів для стилізації Tailwind, RTK (Redux Tool Kit) Query для запитів на сервер, бібліотека для керування станом складних форм Formik та yup для валідації цих форм, react-toastify для сповіщення юзера.

У результаті роботи було спроектовано програмну систему для обміну речей, яка допомагає людям знайти іншого учасника бартеру, для обміну речами, надаючи їм зручні інструменти для цього.

BARTER, USER, EXCHANGE, SOFTWARE SYSTEM, PRODUCT, TRANSACTION, TYPESCRIPT, REACT

The object of development is a software system for exchanging things.

The development goal is to create an efficient and convenient platform that allows users to search for things they need and give away things they no longer need in exchange for another thing that someone else does not need.

The solution method is a development environment and Visual Studio Code, TypeScript programming language, React library, CSS class library for styling Tailwind, RTK (Redux Tool Kit) Query for server requests, Formik library for managing the state

of complex forms and yup for validating these forms, react-toastify for user notifications.

As a result of the work, a software system for exchanging things was designed to help people find another barter participant to exchange things, providing them with convenient tools for this.

Я, Кащенко Юхим Єгорович, студент гр. ПЗПІ-20-7, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для обміну речами. Фронт-енд та мобільний застосунок.», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ 8
1.1 Опис предметної галузі 8
1.2 Аналіз існуючих аналогів 9
1.3 Цілі та ризики проєкту 12
1.4 Постановка задачі 13
2 ПЕРЕЛІК ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ 15
2.1 Загальний опис системи 15
2.2 Основна функціональність системи 16
2.3 Загальні обмеження системи 17
3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ 18
3.1 UML-проєктування програмної системи 18
3.2 Проєктування архітектури системи 19
3.3 Проєктування UI/UX системи або іншого дизайну системи 21
4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ 29
4.1 Опис архітектурного рішення 29
4.2 Обґрунтування вибору програмних рішень 33
5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 44
ВИСНОВКИ 48
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 49
ДОДАТОК А. Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ .. 50
ДОДАТОК Б. Слайди презентації 51
ДОДАТОК В. Специфікація програмного продукту 56
ДОДАТОК Г. Тези за темою атестаційної роботи 66

ВСТУП

Розвиток цифрових технологій кардинально змінив способи взаємодії між людьми. Результатом цього є поява програмних систем для обміну речами, які дозволяють користувачам обмінюватися товарами та послугами без необхідності використання традиційних платіжних засобів. Дані системи пропонують альтернативний підхід до споживання, який спрямований на зменшення витрат та свідоме використання ресурсів.

В умовах глобальної екологічної ситуації та економічної нестабільності важливо знати про способи ефективного використання ресурсів. Програмна система для обміну речами може стати інструментом, який допоможе зменшити надлишкове споживання та економити фінансові ресурси.

Метою даної бакалаврської роботи є розробка клієнтської частини та мобільного застосунку для програмної системи для обміну речами, що надасть зручний та інтуїтивно простий спосіб для взаємодії користувачів з системою.

Для досягнення поставленої мети в рамках роботи необхідно виконати наступні завдання:

- визначити мету та проаналізувати існуючі платформи для обміну речами;
- дослідити сучасні технології для розробки;
- розробити архітектуру системи, що включає тришарову структуру;
- реалізувати інтерфейс з використанням React TypeScript та Tailwind CSS;
- забезпечити можливість використання мобільного застосунку у вигляді прогресивного веб-застосунку (PWA).

Результати даної дипломної роботи можуть бути застосовані у різних сферах діяльності, де є потреба в організації обміну речами та послугами. Це можуть бути соціальні платформи, еко-ініціативи, благодійні організації, а також комерційні проекти, що пропонують альтернативні способи споживання та економії ресурсів. Використання розробленої системи дозволить значно спростити процес обміну речами, підвищити ефективність використання ресурсів та сприяти сталому розвитку суспільства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Опис предметної галузі

Обмін речами, або бартер [1], є однією з найдавніших форм економічних відносин, яка отримала нове призначення та у сучасних умовах завдяки розвитку цифрових технологій.

Системи обміну речами сприяють підвищенню соціальної згуртованості, дозволяючи користувачам знайомитися з новими людьми і встановлювати корисні зв’язки. З економічної точки зору, обмін речами дозволяє зменшити грошові витрати на придбання нових товарів або послуг, що особливо важливо для малозабезпечених верств населення. Ці системи можуть стати важливим ресурсом для тих, хто не має можливості купувати нові речі. З екологічної точки зору, обмін сприяє зменшенню відходів і повторному використанню товарів, що, в свою чергу, зменшує кількість сміття та зберігає природні ресурси. Використання вже існуючих речей знижує потребу у виробництві нових, що позитивно впливає на навколишнє середовище.

Сучасні технології відіграють важливу роль у розвитку цієї галузі. Цифрові платформи, такі як мобільні додатки, веб-сайти та соціальні мережі, значно полегшують процес обміну. Впровадження систем рейтингів, відгуків та верифікації користувачів підвищує довіру та безпеку угод, забезпечуючи надійність та прозорість процесу. Сфера застосування систем для обміну речами є доволі великою і вона включає як побутовий обмін одягом, технікою, меблями, іграшками та іншими речами, так і професійні послуги, такі як ремонт, навчання, консультації. Крім того, обмін інструментами, транспортом, житлом стає все більш популярним явищем.

Вплив систем обміну речами на суспільство є значним, адже вони сприяють розвитку культури спільного використання та сталого розвитку. Вони допомагають людям ефективніше використовувати ресурси, зменшувати екологічний слід та підтримувати один одного у важкі часи. Дані системи допомагають вирішувати соціальні проблеми, зокрема, через підтримку малозабезпечених та вразливих верств населення.

1.2 Аналіз існуючих аналогів

Для того, щоб аналіз предметної області програмної системи для обміну речами вийшов якісним та успішним важливо дослідити існуючі аналоги даної системи. Нижче наведено опис деяких відомих платформ, які успішно функціонують у цій сфері.

Freecycle – це глобальна мережа, що складається з місцевих груп, де користувачі можуть дарувати і отримувати речі безкоштовно (рисунок 1.1). Мета платформи – зменшити кількість відходів, повторно використовуючи речі. Користувачі можуть створювати оголошення про речі, які вони хочуть віддати або отримати, спілкуватися з іншими учасниками для узгодження деталей обміну. Платформа працює через веб-сайт. Freecycle орієнтована виключно на безкоштовний обмін речами, що робить її популярною серед екологічно свідомих спільнот.

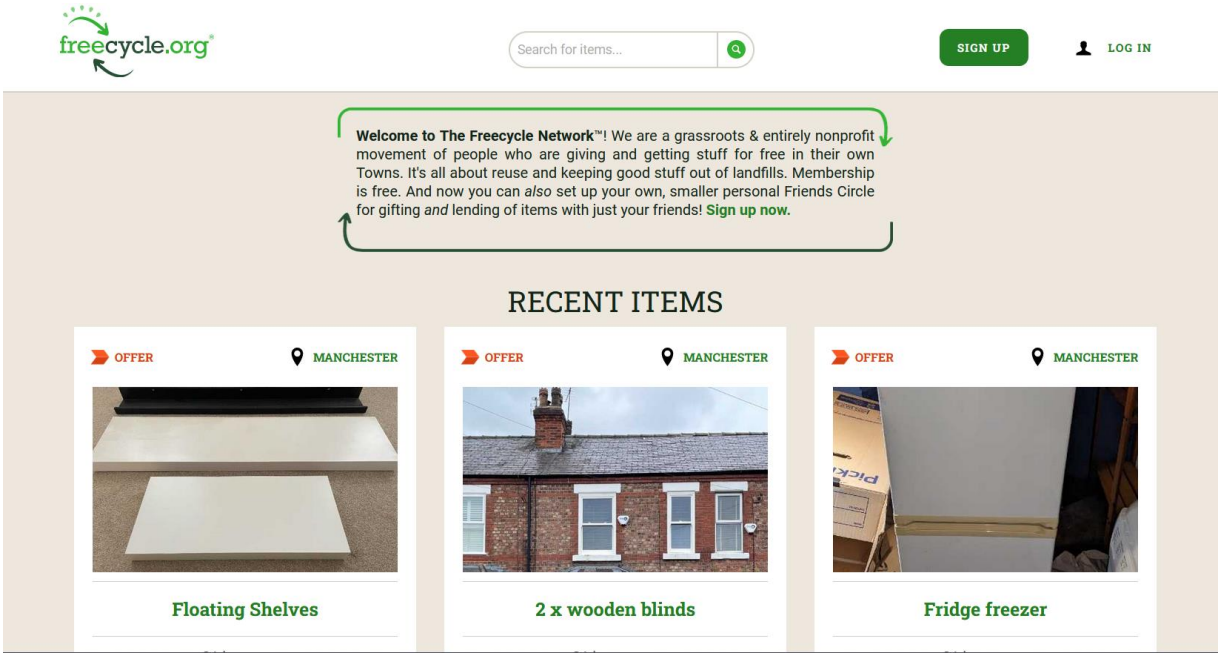

Рисунок 1.1 – Платформа «Freecycle»

OLIO – це платформа, що спеціалізується на обміні побутових товарів та їжі (рисунок 1.2). Користувачі можуть ділитися надлишками їжі або іншими предметами з сусідами. Додаток дозволяє користувачам розміщувати оголошення про надлишки їжі або інші речі, які вони хочуть віддати. Інші користувачі можуть

знаходити ці оголошення за допомогою геолокаційних сервісів та домовлятися про обмін. OLIO активно підтримує боротьбу з харчовими відходами і сприяє зменшенню кількості їжі, що потрапляє на смітник.

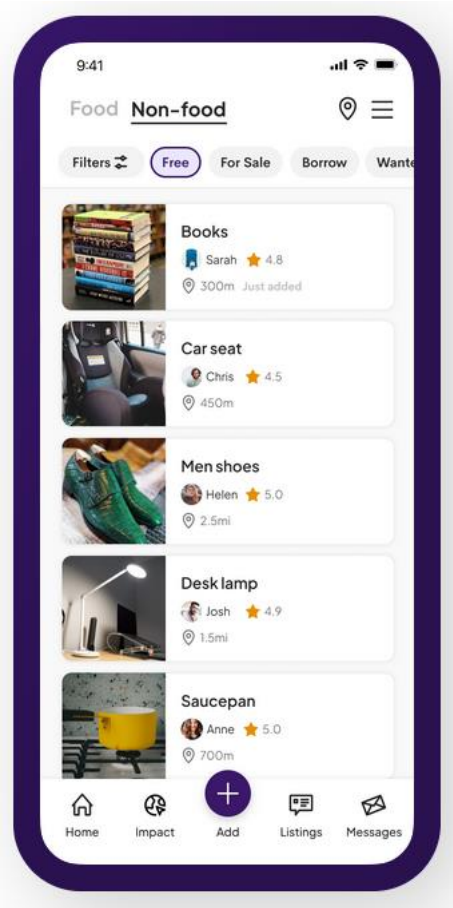

Рисунок 1.2 – Додаток «OLIO»

Rehash – це платформа для обміну одягом (рисунок 1.3). Користувачі можуть обмінюватися одягом безкоштовно, створюючи свої профілі і додаючи речі, які вони хочуть обміняти. Користувачі можуть переглядати профілі інших учасників, обирати речі для обміну та домовлятися про деталі угоди. Платформа включає систему рейтингу та відгуків для підвищення довіри між користувачами. Rehash фокусується на обміні одягом і створенні спільноти людей, зацікавлених у свідомому споживанні та екологічних практиках.

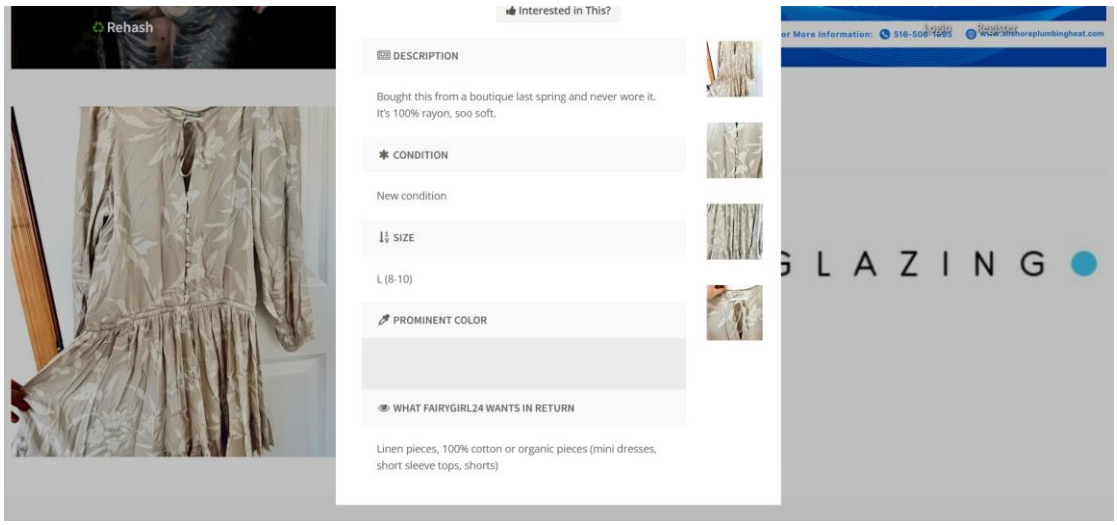

Рисунок 1.3 – Платформа «Rehash»

Letgo – це мобільний додаток, що дозволяє користувачам продавати, купувати та обмінюватися різними товарами, включаючи меблі, електроніку, автомобілі та інші речі (рисунок 1.4). Додаток підтримує розміщення оголошень із фотографіями товарів, інтеграцію з геолокаційними сервісами для пошуку найближчих пропозицій та можливість спілкування між користувачами для узгодження деталей угоди. Letgo поєднує в собі функції для продажу, купівлі та обміну товарів, що робить його універсальним інструментом для місцевих спільнот.

Рисунок 1.4 – Додаток «Letgo»

Аналіз аналогів показав, що існуючі програмні системи для обміну речами успішно вирішують ряд завдань, пов’язаних з організацією процесу обміну товарів і послуг. Вони забезпечують основні функціональні можливості, такі як реєстрація користувачів, додавання та пошук речей, комунікація між користувачами, а також система рейтингів і відгуків.

Однак, існуючі рішення мають певні обмеження і недоліки, які можуть бути покращені. Деякі мають обмежені можливості для налаштування інтерфейсу та не забезпечують підтримку багатомовності, що обмежує їх використання у міжнародному масштабі. Крім того, не всі системи адаптовані під мобільні пристрої і не підтримують прогресивні веб-застосунки (PWA), а деякі навпаки не мають веб-сайт реалізації, що знижує їх зручність та доступність для користувачів.

1.3 Цілі та ризики проєкту

Метою даної роботи є розробка і впровадження клієнтської частини програмної системи для обміну речами, яка забезпечить користувачам зручний спосіб обміну товарів. Основні цілі проєкту включають:

- розробка інтуїтивно зрозумілого та зручного інтерфейсу, який забезпечить зручний доступ до основних функцій обміну речами для користувачів;
- врахування побажань та очікувань користувачів під час розробки інтерфейсу для забезпечення позитивного користувацького досвіду;
- впровадження технологій та методів оптимізації для забезпечення швидкості завантаження сторінок та покращення продуктивності веб-додатка;
- розробка інтерфейсу, який буде працювати на різних пристроях (комп’ютерах, планшетах, мобільних пристроях) та в різних веб-браузерах.

Попри очевидні успіхи, проєкт все одно має ризики, які можуть вплинути на його успішне виконання. Деякі з найважливіших ризиків включають:

- можливість появи технічних труднощів під час розробки та впровадження програмної системи, таких як проблеми з інтеграцією компонентів чи

несподівані технічні обмеження;

- ризик порушення безпеки даних користувачів через атаки з боку зловмисників або вразливості в програмному забезпеченні;
- ризик обмеженої активності користувачів, що може призвести до низької популярності та ефективності системи;
- можливість виникнення проблем з масштабуванням системи в разі збільшення обсягів обміну товарами та послугами.

Для успішної реалізації проєкту необхідно ретельно аналізувати та управляти цими ризиками на кожному етапі розробки та впровадженні програмної системи.

1.4 Постановка задачі

Розробка клієнтської частини програмної системи для обміну речами включає створення сучасного, інтуїтивно зрозумілого та функціонального інтерфейсу, який забезпечить ефективну взаємодію користувачів з системою. Основні задачі, які необхідно вирішити під час розробки клієнтської частини, включають розробку інтерфейсу користувача, адаптивний дизайн, реалізацію функціональних можливостей, оптимізацію продуктивності, забезпечення безпеки, впровадження мультимовності, інтеграцію з серверною частиною та проведення тестування.

Першочерговим завданням є створення привабливого та зручного дизайну, який враховує принципи UX/UI [2], а також забезпечення інтуїтивно зрозумілої навігації та зручного доступу до всіх функцій системи. Інтерфейс має коректно відображатися на різних пристроях, включаючи комп’ютери, планшети та мобільні пристрої, і працювати у всіх популярних браузерах.

Функціональні можливості клієнтської частини включають механізми реєстрації та автентифікації користувачів, можливість додавання, редагування та видалення оголошень про обмін речами, системи пошуку та фільтрації товарів за різними параметрами, функціонал чату для комунікації між користувачами, а також інтеграцію системи рейтингів та відгуків для підвищення довіри та безпеки обмінів.

Оптимізація продуктивності є важливим аспектом розробки, тому необхідно використовувати сучасні технології та методи оптимізації для забезпечення швидкого завантаження сторінок, а також оптимізацію коду та ресурсів для мінімізації часу відгуку інтерфейсу. Безпека користувацьких даних є пріоритетною, тому слід впровадити заходи для захисту особистих даних користувачів та забезпечити безпечне зберігання і передачу даних між клієнтом і сервером.

Мультимовність є важливою складовою для забезпечення доступності системи для ширшої аудиторії, тому необхідно реалізувати підтримку кількох мов інтерфейсу. Інтеграція з серверною частиною забезпечується через коректну взаємодію з сервером для обміну даними, використовуючи REST API. Проведення детального тестування всіх компонентів інтерфейсу дозволить виявити та усунути помилки, забезпечуючи стабільну роботу системи під різними навантаженнями.

2 ПЕРЕЛІК ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальний опис системи

Програмна система для обміну речами розробляється з метою створення зручної, функціональної та безпечної платформи, яка дозволить користувачам обмінюватися різноманітними товарами. Основною метою системи є надання користувачам платформи, де вони зможуть легко знаходити потрібні товари, обмінюватися ними з іншими користувачами, підтримувати комунікацію та отримувати зворотний зв’язок. Це включає в себе реєстрацію та авторизацію користувачів, створення та управління оголошеннями, системи пошуку та фільтрації, а також механізми обміну повідомленнями та рейтингової системи.

Для розробки клієнтської частини системи плануються використовуватись наступні технології:

- Visual Studio Code, як основне середовище розробки;
- React з TypeScript для створення компонентів інтерфейсу користувача;
- Vite, як інструмент для швидкої збірки проєкту;
- Tailwind CSS для стилізації інтерфейсу;
- i18n для забезпечення мультимовності;
- Formik для роботи з формами та валідацією даних;
- PWA (Progressive Web App) для забезпечення мобільної версії додатку.

У якості архітектури планується використання тришарової моделі, яка включає:

- презентаційний шар, який відповідає за взаємодію з користувачем, включаючи інтерфейс користувача та логіку відображення;
- шар логіки, який містить логіку застосунку, що обробляє дані та приймає рішення;
- дані та інтеграції, який взаємодіє з серверною частиною, включає API запити та обробку даних.

Клієнтська частина системи є веб-додатком, який буде побудований на React з використанням TypeScript для забезпечення строгого типізування та підвищення

надійності коду. Tailwind CSS буде використовується для швидкої та ефективної стилізації інтерфейсу, що забезпечує сучасний та адаптивний дизайн. PWA дозволить використовувати веб-додаток як мобільний застосунок, забезпечуючи зручність та доступність на різних пристроях без необхідності встановлення з окремих магазинів додатків.

Загалом, клієнтська частина системи забезпечить інтуїтивно зрозумілий, швидкий та сучасний інтерфейс для користувачів, дозволяючи їм легко обмінюватися товарами та послугами через зручну та надійну платформу.

2.2 Основна функціональність системи

Програмна система для обміну речами має на меті забезпечити користувачів зручним та функціональним інструментом для обміну товарами та послугами. Основна функціональність системи включатиме наступні компоненти.

Реєстрація та авторизація користувачів. Система забезпечить користувачів можливістю створювати облікові записи за допомогою реєстрації, що включає введення основних даних, таких як ім’я, електронна пошта та пароль. Після реєстрації користувачі можуть увійти в систему через процес авторизації.

Управління профілем користувача. Користувачі зможуть редагувати свій профіль, додаючи або змінюючи особисту інформацію, фотографії та контактні дані.

Створення та управління оголошеннями. Система надасть можливість користувачам створювати оголошення про товари, які вони бажають обміняти. В процесі створення оголошення користувачі можуть додавати опис, фотографії, категорії та інші важливі деталі. Користувачі також зможуть редагувати або видаляти свої оголошення в будь-який час.

Пошук та фільтрація оголошень. Для зручності користувачів система забезпечить механізми пошуку та фільтрації оголошень за різними параметрами, такими як категорія, колір, бажані категорії. Це допоможе швидко знаходити потрібні товари серед великої кількості пропозицій.

Процес обміну. Процес обміну почнеться з вибору користувачем цікавого

оголошення та ініціювання запиту на обмін. Після досягнення згоди щодо умов обміну, користувачі можуть підтвердити обмін у системі.

Взаємодія між користувачами. Користувачі зможуть взаємодіяти між собою через вбудований чат, що допоможе обговорити деталі обміну безпосередньо в системі. Це забезпечить зручний та безпечний спосіб комунікації, зменшуючи ризик втрати контактної інформації.

Для підвищення довіри та безпеки обмінів, система матиме функціонал рейтингів. Користувачі зможуть залишати оцінки про свої обміни після завершення обміну. Це сприятиме створенню надійної спільноти користувачів, де можна уникати недобросовісних учасників.

2.3 Загальні обмеження системи

Розробка клієнтської частини програмної системи для обміну речами повинна враховувати певні обмеження, щоб забезпечити стабільну та безпечну роботу додатку. Нижче наведені основні обмеження, які необхідно врахувати:

- для роботи PWA [3] у користувача має бути будь-який браузер, окрім Safari та Firefox;
- у користувача має бути стабільний інтернет;
- користувач має знати або українську, або англійську мову;
- у користувача буде можливість завантажити обмежену кількість фото для товару.

Дотримання цих обмежень сприяє підтримці продуктивності та надійності програмної системи для обміну речами, забезпечуючи при цьому позитивний досвід для всіх користувачів.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 UML-проєктування програмної системи

Під час детального аналізу програмної системи для обміну речами та її проєктування, було створено Use-case діаграму, або діаграму прецедентів [4], яка описує, які функціональні можливості надає система для користувача та як він взаємодіє з системою (див. рис. 3.1).

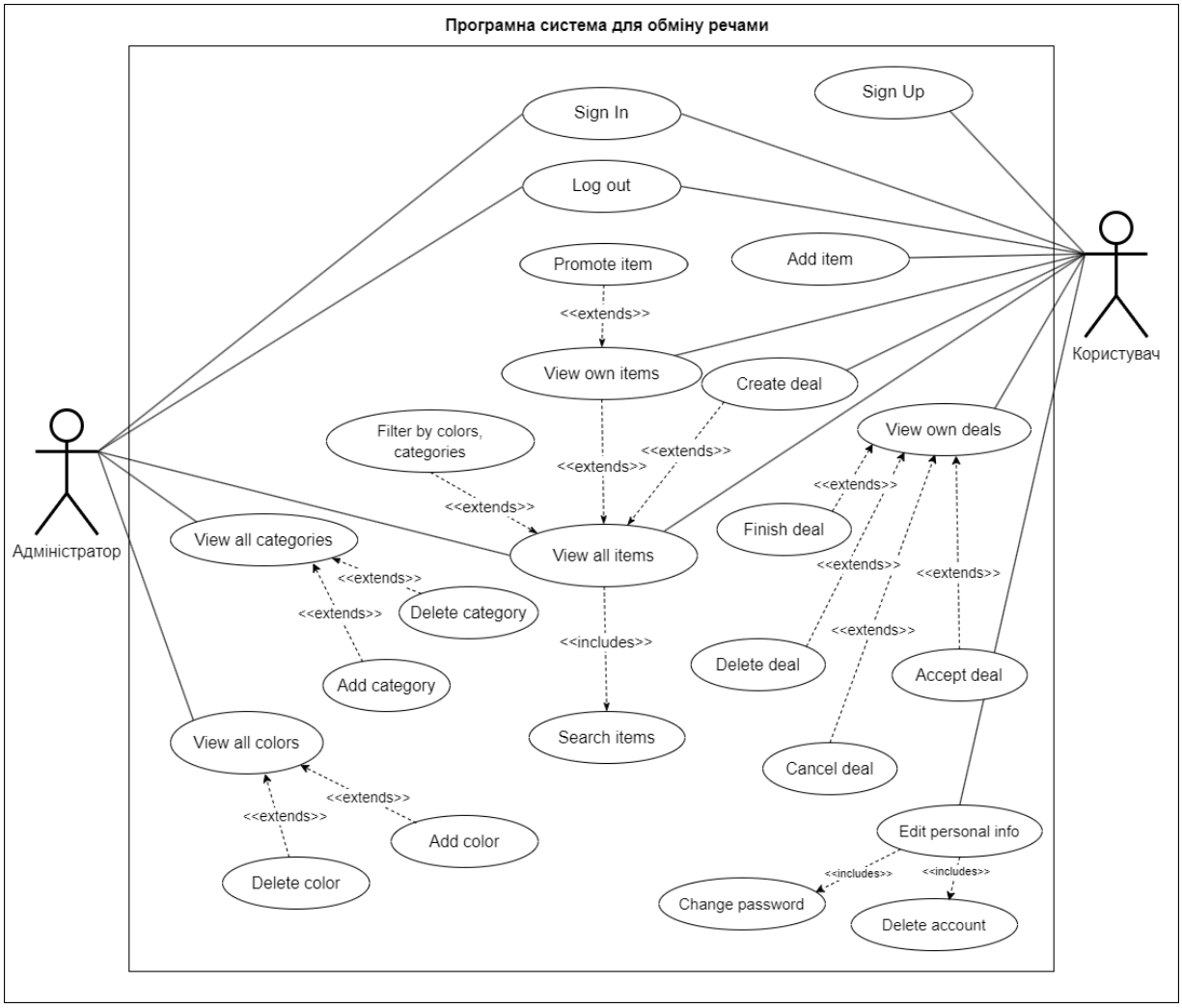

Рисунок 3.1 – Діаграма прецедентів для програмної системи (рисунок виконаний самостійно)

Тільки авторизовані користувачі мають доступ до основних функцій системи, дії, які доступні усім користувачам, навіть не авторизованим, – перегляд усіх доступних товарів, пошук та фільтрація товарів.

Тільки адміністратор має доступ до створення нових та видалення старих кольорів та категорій.

Діаграма активності – це візуальне представлення якогось процесу [5]. На рисунку 3.2 зображено процес реєстрації користувача.

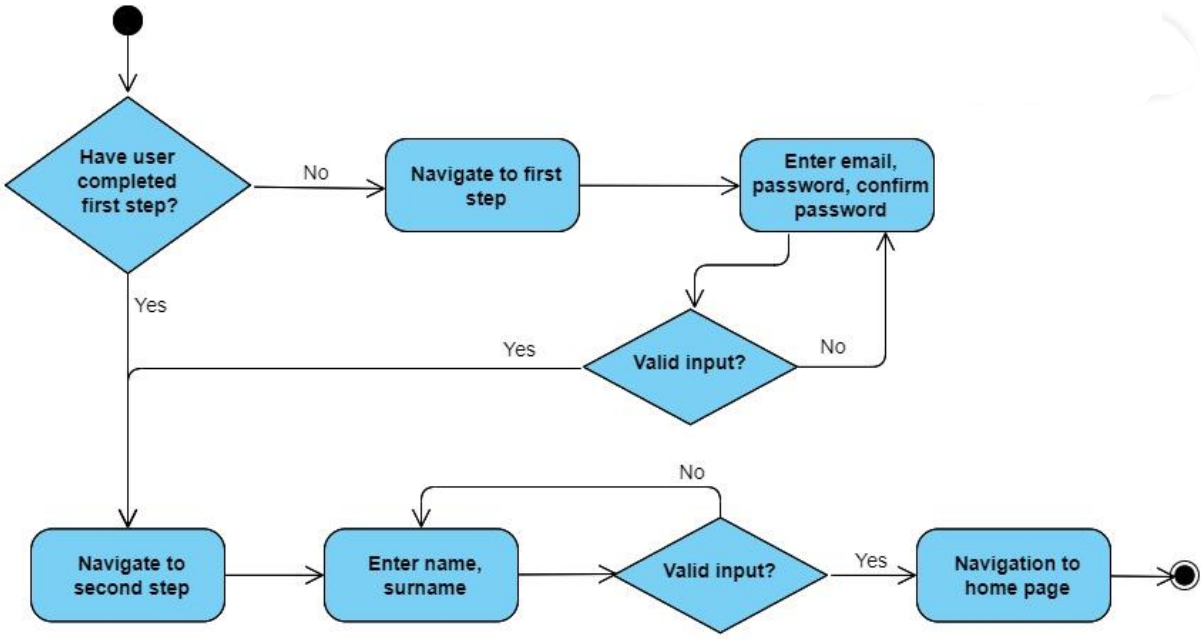

Рисунок 3.2 – Діаграма активності для процесу реєстрації користувача (рисунок виконаний самостійно)

Процес починається з перевірки, чи завершив користувач перший крок. Якщо ні, користувача перенаправляють до першого кроку, де він вводить електронну пошту, пароль та підтверджує пароль. Після цього перевіряється валідність введених даних. Якщо дані не валідні, користувача просять ввести їх знову. Якщо дані валідні, він переходить до другого кроку, де вводить ім'я та прізвище. Потім знову перевіряється валідність введених даних. Якщо дані не валідні, користувача знову просять ввести їх правильно. Якщо дані валідні, користувач переходить на головну сторінку. Завершення процесу реєстрації позначається кінцевою точкою на діаграмі.

3.2 Проєктування архітектури системи

Як було вказано у попередніх пунктах, для веб-частини системи було обрано тришарову архітектуру. Трирівнева архітектура, або багаторівнева – це шаблон

проєктування програмного забезпечення, який поділяє додаток на три окремі взаємопов’язані рівні з метою покращення масштабованості, керованості та гнучкості додатку [6]. На рисунку 3.3 зображена схема архітектури клієнтської частини системи.

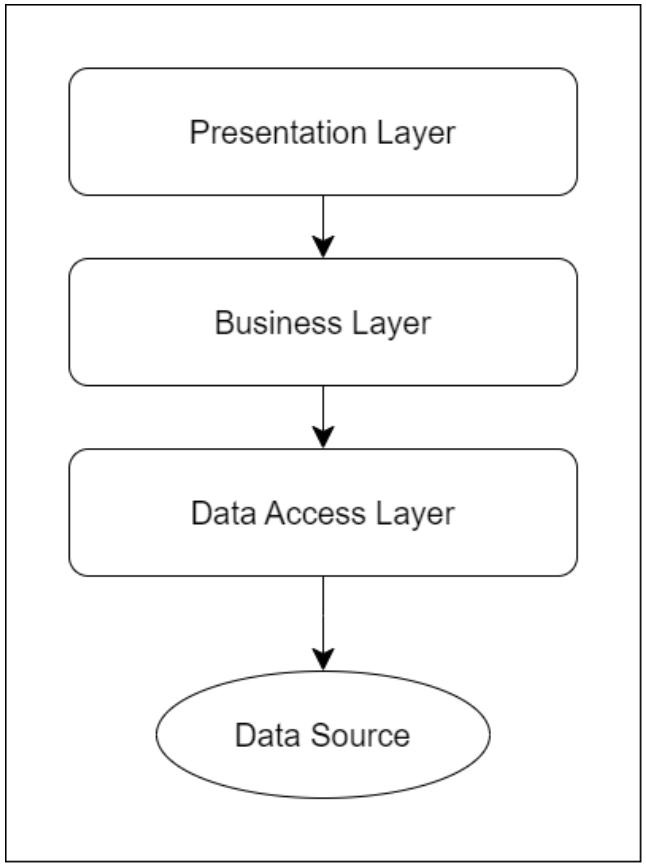

Рисунок 3.3 – Схема архітектури системи (рисунок виконаний самостійно)

Шар представлення (Presentation Layer) – це шар, де знаходяться всі компоненти інтерфейсу користувача; він відповідає за рендеринг даних і взаємодію з користувачами. Шар логіки (Business Layer) зосереджений головній на логіці програми; він містить правила, алгоритми та процеси, які керують функціональністю програми. Шар доступу до даних (Data Access Layer) відповідає за отримання даних з API.

Перевагами тришарової архітектури є:

– масштабованість: кожен шар можливо масштабувати незалежно від потреби;
– повторне використання: бізнес-логіка та функції доступу до даних можуть

використовуватися в різних частинах програми без повторного копіювання одного і того ж коду;

- обслуговування: можна замінити або оновити один шар, не впливаючи на інші шари;
- тестування: легше тестувати кожен рівень окремо;
- безпека: доступ до даних можна суворо контролювати;
- продуктивність: оптимізуючи кожен шар окремо, можна покращити загальну продуктивність застосунку;

У тришаровій архітектурі кожен рівень має чітко визначену роль і відповідальність, що дозволяє зменшити складність коду та покращити його підтримуваність. Presentation Layer забезпечує ефективну взаємодію з користувачем, Business Layer виконує основні бізнес-процеси, а Data Access Layer забезпечує надійний доступ до даних, зберігаючи їх цілісність та безпеку. Завдяки цій структурі стає можливим роздільне розроблення та тестування компонентів системи, що сприяє швидшому виявленню та виправленню помилок, а також впровадженню нових функцій без ризику порушення роботи інших частин системи.

Окрім того, трирівнева архітектура сприяє гнучкості та адаптивності системи до змінних вимог бізнесу, що є важливим фактором для тривалого підтримання та розвитку програмного забезпечення. В умовах динамічного ринку можливість швидко реагувати на зміни стає ключовою перевагою, що підвищує конкурентоспроможність програмної системи.

3.3 Проєктування UI / UX системи або іншого дизайну системи

Після детального аналізу було визначено цільову аудиторію програмної системи для обміну речами, проведено дослідження щодо користувацького досвіду, обрано кольорову палітру і в результаті створено UI/UX дизайн головних екранів платформи [7].

На рисунку 3.4 зображено макет головної сторінки програмної системи.

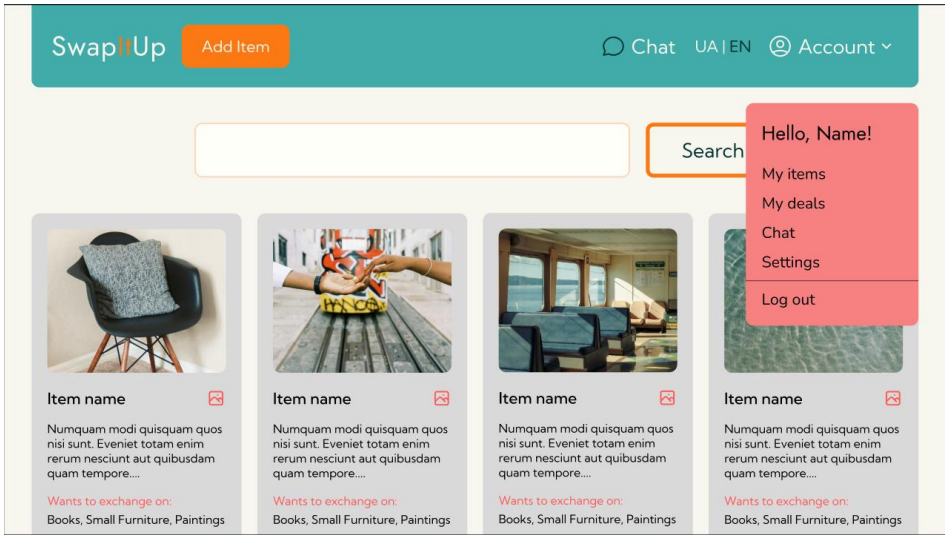

Рисунок 3.4 – Дизайн головної сторінки (рисунок виконано самостійно)

Головна сторінка поділяється на: навігаційну панель (хедер), секцію з пошуком, стрічку з картками товарів та «підвал» сайту. Якщо сайт буде відвіданий з девайсу з маленькою роздільною здатністю екрана, то в хедері замість навігаційної панелі з'явиться бургер меню, в якому вона і буде представлена, також кількість карток в рядку буде плавно зменшуватися разом з роздільною здатність від 4 максимально карток в рядку до 2. Варто зауважити, що хедер та футер мають різний вигляд в залежності від того чи авторизований користувач, на рисунку показаний варіант, коли користувач авторизований.

На рисунку 3.5 зображений вигляд картки товару на головні сторінці.

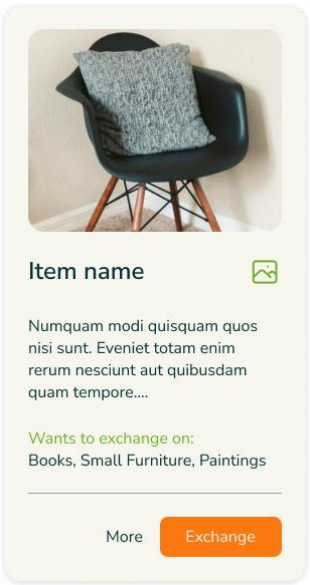

Рисунок 3.5 – Дизайн картки товару (рисунок виконано самостійно)

На картці відображається мінімальна потрібна інформація для того щоб юзер розумів що це за товар, така як: головне зображення товару, його назва, категорія, в яку входить цей товар, неповний опис та категорії товарів, на які власник хоче обміняти даний товар.

На рисунку 3.6 зображений дизайн сторінки з повною інформацією про певний товар.

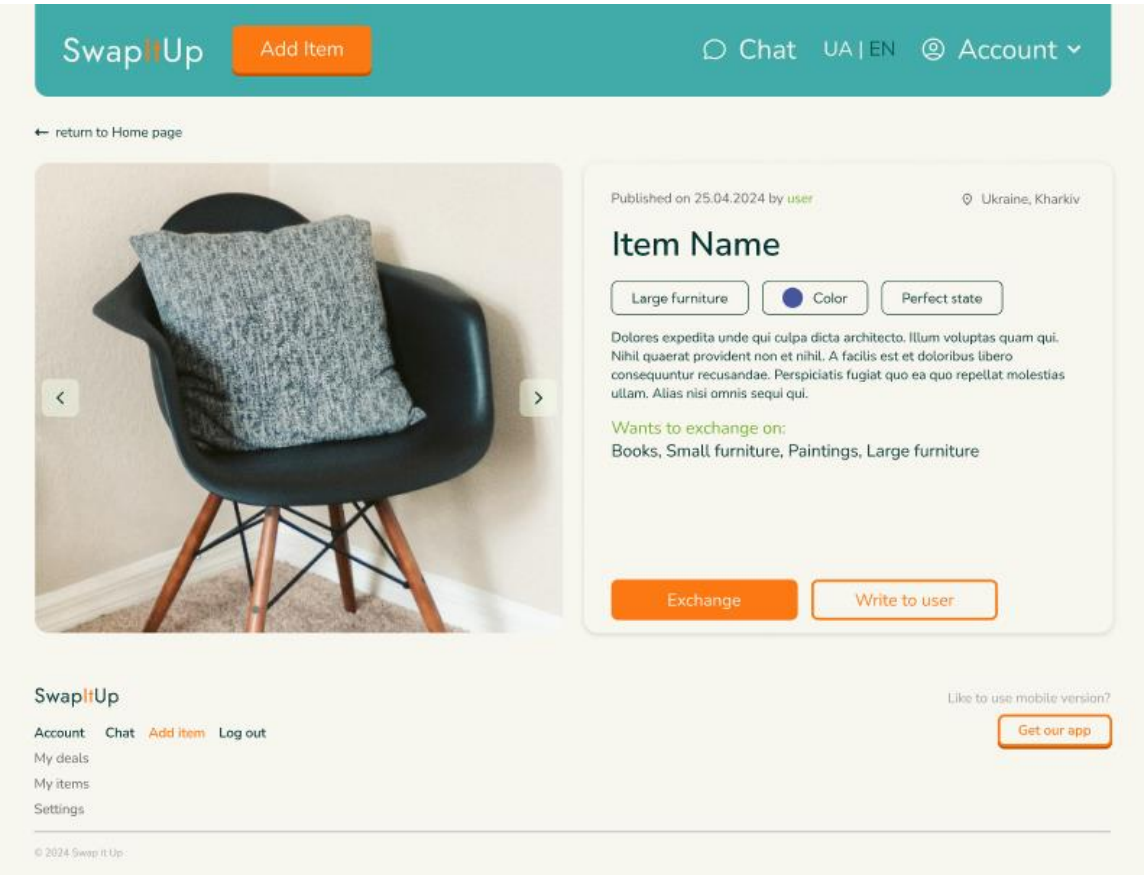

Рисунок 3.6 – Дизайн сторінки з повною інформацією про товар (рисунок виконано самостійно)

На цій сторінці вже можна дізнатися усю наявну інформація про товар, який зацікавив: можна переглянути фотографії у слайдері, назву товару, ким опубліковане оголошення та звідки автор. Також тут відображається інформація щодо категорії, кольору та стану товару, його опис та категорії, з яких автор оголошення хоче отримати запит на обмін цього товару. Саме з цієї сторінки можна розпочати обмін товарами, для цього призначена кнопка «Exchange».

На рисунку 3.7 зображена сторінка додавання нового товару

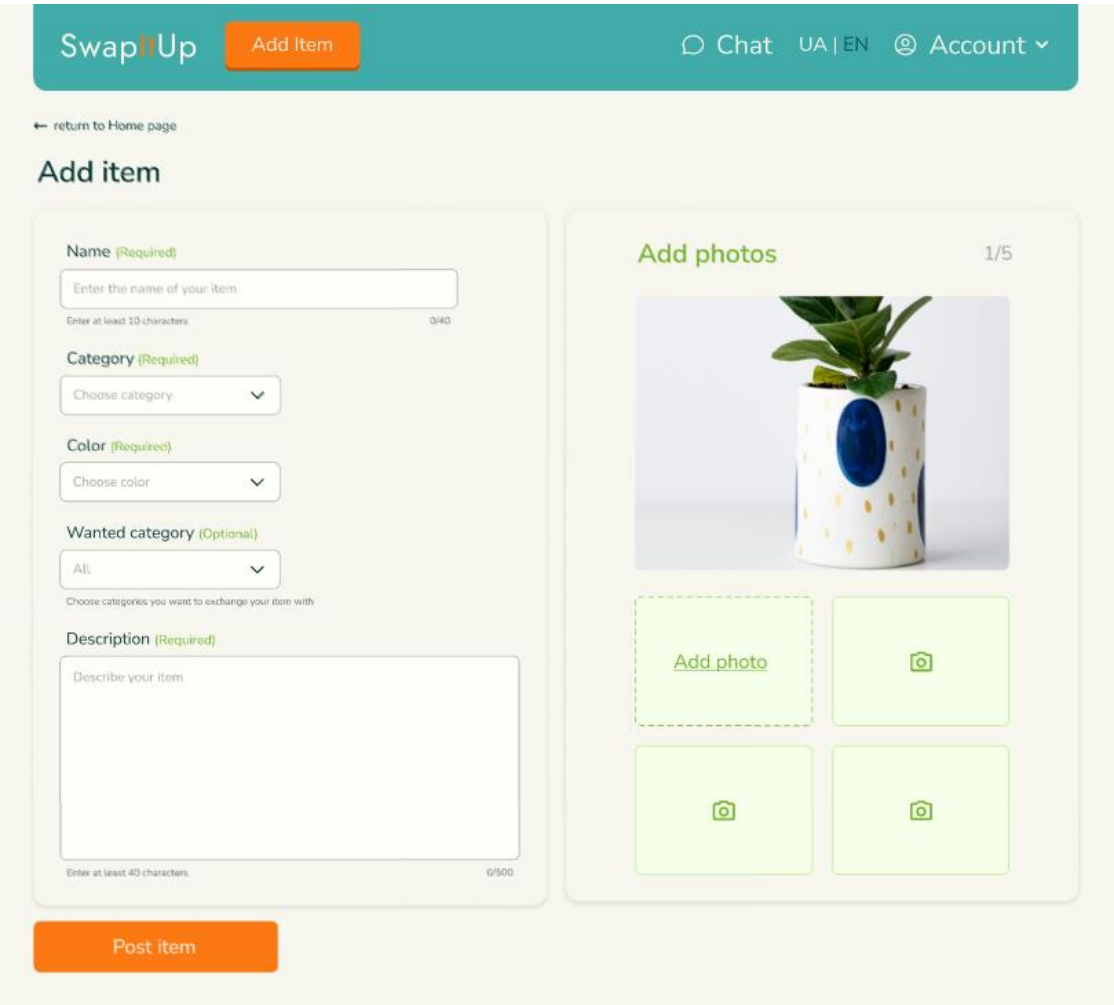

Рисунок 3.7 – Дизайн сторінки додавання нового товару (рисунок виконано самостійно)

На цій сторінці можна побачити форму, яку необхідно заповнити для того, щоб додати товар, в неї входять: поле назви (необхідно)б поле категорії (необхідно), поли кольору (необхідно), поле бажаної категорії, опис (необхідно), а також у користувача є можливість прикріпити до 5 фотографій товару. Варто зазначити, що даний дизайн форми використовується і на сторінці з редагуванням товару, різниця полягає в тому, що на сторінці редагування поля форми вже заповненні відповідною інформацією та немає поля для вибору категорії даного товару.

На рисунку 3.8 зображений процес початку обміну.

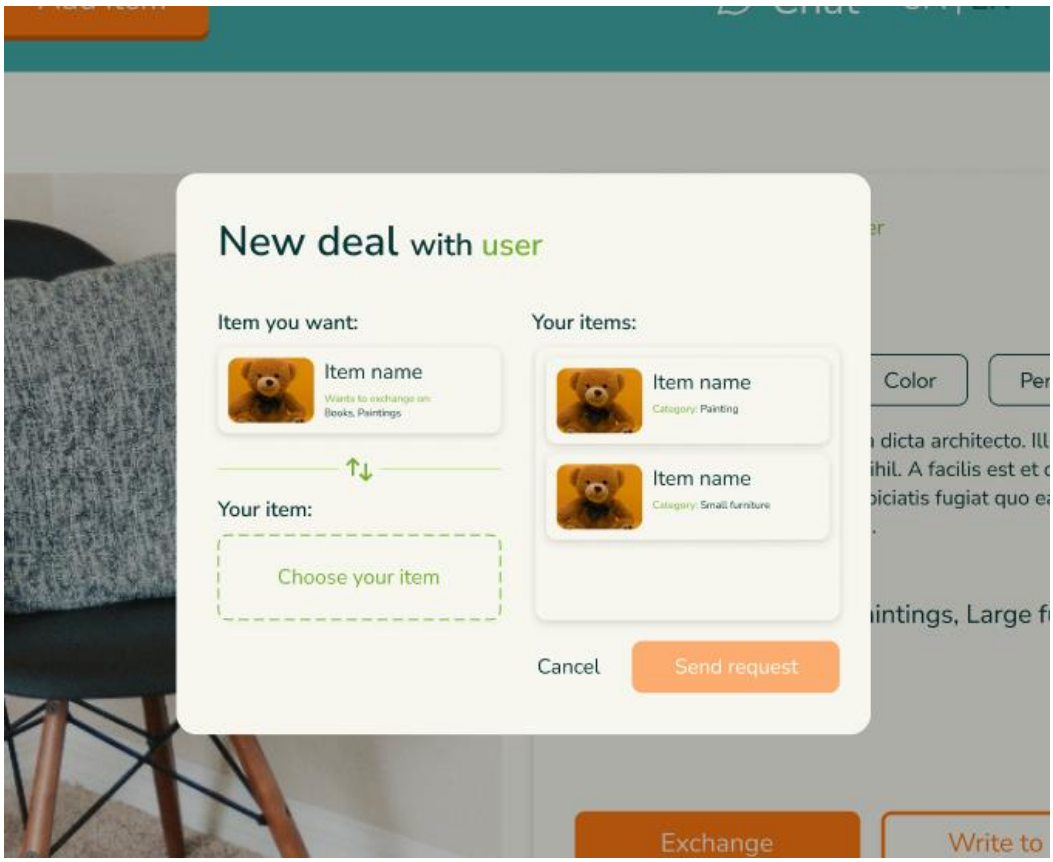

Рисунок 3.8 – Дизайн вікна початку обміну (рисунок виконано самостійно)

Для того, щоб юзер розпочав обмін бажаного товару, йому потрібно натиснути кнопку «Exchange» на сторінці з деталями відповідного товару, після чого з’явиться модальне вікно, яке запропонує користувачу обрати один із своїх товарів та підтвердити початок обміну натиснувши кнопку «Send request». Також у цьому вікні можна побачити коротку інформацію про товар, який користувач хоче отримати, така ж інформація надається і про товари юзера.

На рисунку 3.9 зображена сторінка з налаштування особистих даних акаунту користувача.

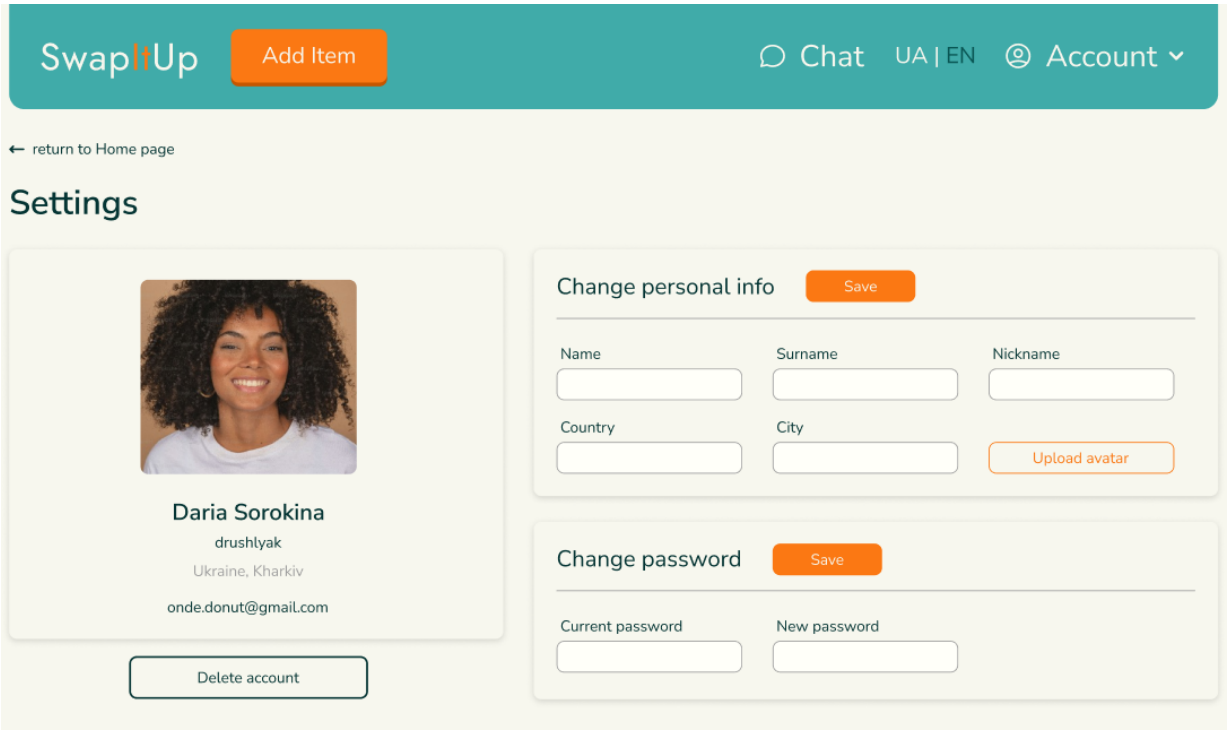

Рисунок 3.9 – Дизайн вікна налаштування особистих даних акаунту користувача (рисунок виконано самостійно)

На даній сторінці користувач може додати та/або відредагувати особисту інформації в своєму акаунті за допомогою відповідної форми, яка включає такі поля: ім’я користувача, його прізвище, нікнейм, країна та місто, також натиснувши кнопку «Upload avatar» користувач має змогу завантажити чи оновити свій аватар. Під цією формою є інша, яка відповідає за зміну паролю.

На рисунку 3.10 зображений дизайн сторінки з автентифікацією користувача.

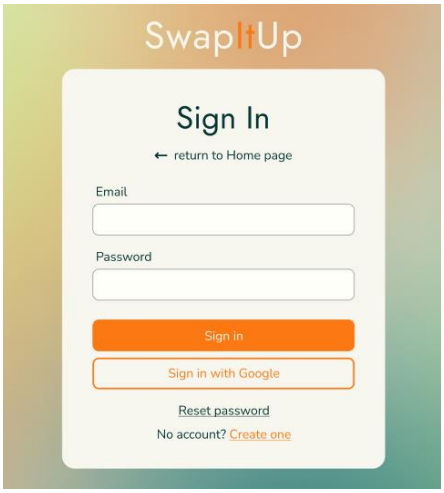

Рисунок 3.10 – Дизайн вікна автентифікації користувача (рисунок виконано самостійно)

На цій сторінці користувач проходить автентифікацію шляхом заповнення полів поштової адреси та паролю від акаунту. Також юзер може повернутися на головну сторінку, або створити новий акаунт.

На рисунку 3.11 зображений дизайн сторінки з реєстрацією користувача, перший крок.

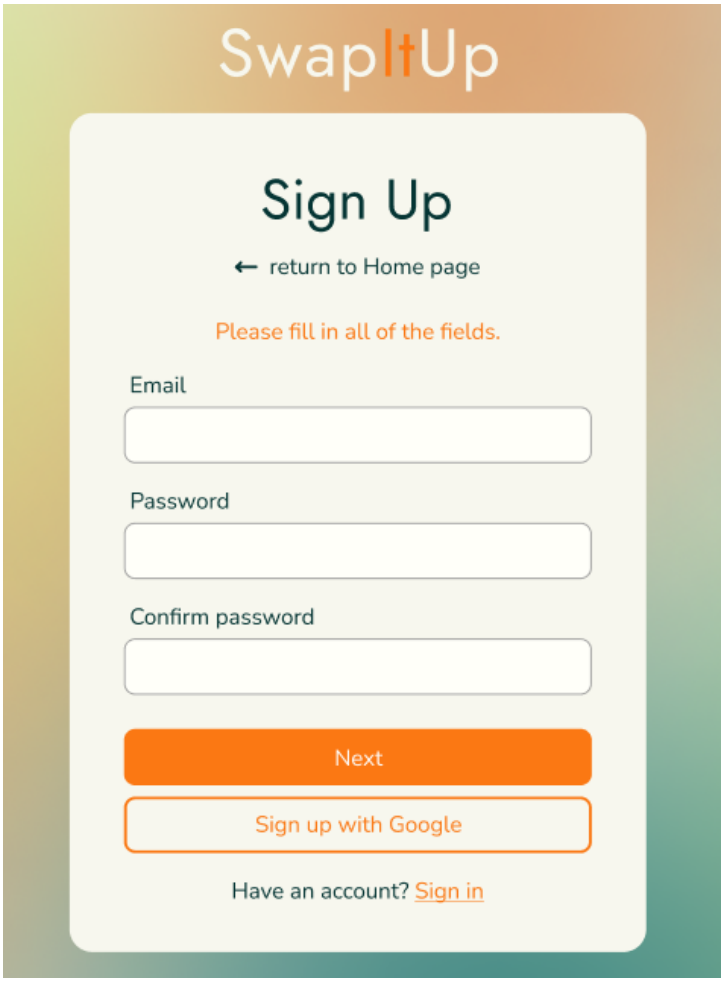

Рисунок 3.11 – Дизайн вікна реєстрації користувача, перший крок (рисунок виконано самостійно)

Сторінка реєстрації складається з двох кроків. На першому кроці користувач має заповнити поля електронної пошти, паролю та підтвердження паролю, після чого, якщо вся інформація введена правильно, користувачу треба заповнити поле з ім’ям та прізвищем, на цьому реєстрація завершується. Другий крок оформлений в такому ж стилі та такій же палітрі кольорів.

На представлених рисунках були зображені одні з основних сторінок та

елементів платформи для обміну речами, інші складові дизайну веб-сайту виконані в подібному стилі, з використанням такої палітри кольорів та схожих елементів. Варто зазначити, що всі вищезгадані сторінки мають адаптивні версії під пристрої з меншою роздільною здатністю екрану, такі як нетбуки, планшети, смартфони тощо.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Опис архітектурного рішення

Для розробки клієнтської частини програмної системи для обміну речами був використаний архітектурний підхід на основі тришарової архітектури. Така архітектура поділяє додаток на три окремі шари: Presentation Layer (шар представлення), Business Layer (шар логіки) та Data Access Layer (шар доступу до даних). Цей підхід забезпечує чітке розділення обов'язків, що сприяє кращій керованості, масштабованості та підтримуваності системи.

Переваги трирівневої архітектури:

- Масштабованість: Кожен шар можна масштабувати незалежно відповідно до потреб, що дозволяє ефективно використовувати ресурси.
- Повторне використання: Логіка бізнесу та функції доступу до даних можуть використовуватися в різних частинах програми без необхідності дублювання коду.
- Тестування: Легше тестувати кожен рівень окремо, що спрощує процес виявлення та виправлення помилок.
- Безпека: Доступ до даних можна суворо контролювати, що підвищує загальний рівень безпеки системи.
- Продуктивність: Оптимізація кожного шару окремо дозволяє підвищити загальну продуктивність додатка.

Presentation Layer (Шар представлення) відповідає за взаємодію з користувачем. Це шар, де знаходяться всі компоненти інтерфейсу користувача, такі як веб-сторінки, форми, кнопки та інші елементи UI. Цей шар забезпечує відображення даних, отриманих з бізнес-логіки, та обробку введених користувачем даних. Використовуючи бібліотеки для створення динамічних і інтерактивних інтерфейсів, таких як React, ми можемо забезпечити швидку і плавну взаємодію з додатком. Tailwind CSS допомагає швидко та ефективно стилізувати компоненти, що сприяє покращенню користувацького досвіду та підвищенню продуктивності системи. Приклад коду компонента картки товару:

```
     <Card {...rest} className={cn("flex flex-col gap-4", className, {
"group cursor-pointer": !myItem })} onClick={handleClickItem}>
           <div className='relative aspect-square max-h-[200px] below-
768:max-h-[150px]'>
             {!imageLoaded && <div className='skeleton-loader !rounded-2xl'
/>}
             <img
               src={img || placeholder}
               alt='item photo'
               className={cn("rounded-2xl object-cover aspect-square max-h-
[200px] w-full below-768:max-h-[150px]", { hidden: !imageLoaded })}
               onLoad={() => setImageLoaded(true)}
             />
           </div>
           <div className='flex justify-between items-start gap-4'>
             <Link
               to={RoutesEnum.ITEM.replace(":id", id!)}
               className='group-hover:text-orange400 hover:text-orange400
hover:no-underline leading-tight break-all font-semibold text-[1.5rem]
mobile:text-[1.25rem]'>
               {title}
             </Link>{" "}
             <div      className='shrink-0      mobile:[&>*]:size-[1.5rem]
[&>*]:size-[2rem]'>{t(category)}</div>
           </div>
           <p className='line-clamp-3'>{description}</p>
           <div className='flex-1 flex flex-col justify-end gap-4'>
             {!myItem && <hr className='h-px bg-white200 w-full' />}
             <p>
               <span   className='text-green600'>{t("Wants   to   exchange
on")}:</span>
               <br />
               <span className='line-clamp-1'>
                 {wanted && wanted.length > 0 ? wanted.map((category, index)
=> (index < wanted.length - 1 ? `${t(category)}, ` : t(category))) : t("All")}
               </span>
             </p>
             {myItem && (
               <>
                 <hr className='h-px bg-white200 w-full' />
                 <div className='w-full flex gap-4 justify-between items-
center mobile:flex-col-reverse'>
                   <div className='flex gap-4 mobile:w-full'>
                     <DeleteIcon  className='size-8  hover:stroke-soft-red
cursor-pointer' onClick={handleDelete} />
                     <EditIcon   className='size-8   hover:stroke-orange400
cursor-pointer' onClick={handleEdit} />
                   </div>
                   <Button size='sm' className='mobile:w-full'>
                     {t("Promote")}
                   </Button>
                 </div>
               </>
             )}
           </div>
         </Card>
```

Business Layer (Шар логіки) є центральним компонентом трирівневої архітектури, що зосереджений на логіці програми. Він містить бізнес-правила, алгоритми та процеси, які керують функціональністю додатка. У цьому шарі відбувається обробка даних, отриманих від Presentation Layer, та взаємодія з Data Access Layer для отримання або збереження даних. Використання TypeScript у поєднанні з React забезпечує високу продуктивність і можливість легкої інтеграції з іншими сервісами, такими як API або зовнішні бібліотеки. Приклад коду зі сторінки реєстрації:

```
    const nextStep = () => {
        setStep((prev) => prev + 1);
      };

      const handleFirstStep = async () => {
        if (errors.email || errors.password || errors.confirmPassword) {
          validateField("email");
          validateField("password");
          validateField("confirmPassword");
          return;
        }
        const response = await register({ email, password });
        if (response.error) {
          if ("data" in response.error) {
            const payload = response.error.data as ErrorResponse;
            let message = "";
            for (const key in payload.errors) {
              if (key !== "DuplicateUserName") message += payload.errors[key].map((value) => value).join("\n");
            }
            toast.error(message || t("Register failed"), { className: "!bg-error" });
            return;
          }
          toast.error(t("Register failed"), { className: "!bg-error" });
          return;
        }
        await login({ email, password });
        nextStep();
      };

      const handleSecondStep = async () => {
        if (errors.name || errors.surname) {
          validateField("name");
          validateField("surname");
          return;
        }
        try {
          const response = await addDetails({ name, surname });
          if (response.error) {
            toast.error(t("Adding details failed"), { className: "!bg-
```

```
error" });
              return;
            }
            navigate(RoutesEnum.HOME);
          } catch (error) {
            toast.error(JSON.stringify(error), { className: "!bg-error" });
          }
        };
```

Data Access Layer (Шар доступу до даних) відповідає за з'єднання з джерелами даних, такими як веб-сервіси або API. Він містить методи для отримання, збереження та модифікації даних. Цей шар забезпечує абстракцію доступу до даних, що дозволяє змінювати джерела даних або методи доступу до них без впливу на інші шари системи. Використання бібліотеки RTK Query спрощує роботу з API та забезпечує ефективне управління станом даних і кешуванням відповідей. Код із файлу apiSlice.ts, який повністю відповідає за цей шар:

```
     reducerPath: "api",
       baseQuery:                  fetchBaseQuery({                  baseUrl:
`${import.meta.env.VITE_SERVER_URL}`, credentials: "include" }),
       endpoints: (builder) => ({
         //#region user
         login: builder.mutation({
           query: (credentials) => ({
             url: "/user/login",
             method: "POST",
             body: credentials,
           }),
         }),
         logout: builder.mutation<any, void>({
           query: () => ({
             url: "/user/logout",
             method: "DELETE",
           }),
         }),
         register: builder.mutation({
           query: (credentials) => ({
             url: "/user/register",
             method: "POST",
             body: credentials,
           }),
         }),
         changePassword: builder.mutation({
           query: (passwords) => ({
             url: "/user/ChangePassword",
             method: "POST",
             body: passwords,
           }),
```

```
}),
addDetails: builder.mutation({
  query: (details) => ({
    url: "/user/details",
    method: "POST",
    body: details,
  }),
}),
```

Загалом, трирівнева архітектура забезпечує високу гнучкість, покращену продуктивність та спрощене обслуговування системи, що робить її оптимальним вибором для розробки програмної системи для обміну речами.

4.2 Обґрунтування вибору програмних рішень

Для того, щоб реалізувати клієнтську частину програмної системи для обміну речами, яка включає в себе веб-застосунок та його мобільну версію, було обрано такі технологічні рішення: Visual Studio Code – середовище розробки, Vite – білдер проекту, React разом з мовою TypeScript, Tailwind – бібліотека класів для легкої стилізації, RTK Query – бібліотека, що дозволяє легко робити звернення до серверу, працювати з відповіддю та її кешем, i18n – бібліотека для локалізації, react-router-dom – для маршрутизації на сайті та Formik – для керування станом великих форм та валідації полів, разом з Formik було прийнято рішення використовувати yup – це бібліотека, за допомогою якої можна легко робити валідаційні схеми для полів у формі. Для мобільного застосунку була обрана технологія PWA.

Visual Studio Code (VS Code) було обрано як середовище розробки завдяки його легкості, багатофункціональності та широкій підтримці плагінів. VS Code підтримує TypeScript та інші мови програмування, що дозволяє легко налаштувати робоче середовище для роботи з React. Крім того, його інтеграція з системами контролю версій, такими як Git, значно спрощує процес управління кодом та співпраці в команді. Середовище також пропонує широкий спектр інструментів для налагодження та тестування коду, що робить його ідеальним вибором для розробки сучасних веб-застосунків.

Vite було обрано як білдер проекту завдяки його швидкості та ефективності. На відміну від традиційних білдерів, таких як Webpack, Vite забезпечує миттєве

завантаження та швидкий перегляд змін у коді завдяки використанню сучасних браузерних можливостей. Це значно підвищує продуктивність розробників та знижує час на компіляцію проекту. Vite також добре інтегрується з React та підтримує TypeScript, що робить його зручним інструментом для побудови високопродуктивних додатків.

React було обрано для створення клієнтської частини завдяки його компонентній архітектурі та великій спільноті розробників. React забезпечує високу гнучкість та можливість повторного використання компонентів, що спрощує розробку та підтримку коду.

Варто зазначити, що React реалізує підхід SPA. Single Page Application (SPA) – це підхід до створення веб-застосунків [8], де весь контент завантажується на одну сторінку, а подальші дії користувача, такі як навігація або взаємодія, не вимагають повного перезавантаження сторінки. Це забезпечує більш швидку та плавну взаємодію з додатком, покращуючи користувацький досвід. SPA дозволяє додатку бути більш реактивним та інтерактивним, знижуючи затримки при переходах між сторінками та взаємодією з сервером.

React ідеально підходить для розробки SPA завдяки своїй можливості створювати динамічні та інтерактивні інтерфейси. Використовуючи віртуальний DOM, React мінімізує кількість маніпуляцій з реальним DOM, що покращує продуктивність додатку. Крім того, React підтримує компонентний підхід, що дозволяє розбити інтерфейс на невеликі, незалежні частини, які можна повторно використовувати та легко тестувати.

TypeScript забезпечує додатковий рівень надійності для React-застосунків, завдяки своїй статичній типізації. Це означає, що багато помилок можуть бути виявлені на етапі компіляції, а не під час виконання, що знижує ризик виникнення помилок у продакшн-версії додатку. TypeScript також покращує автозаповнення та навігацію по коду, що підвищує продуктивність розробників та полегшує підтримку великого кодового бази.

Переваги використання SPA з React та TypeScript включають:

– Покращена продуктивність: SPA знижують затримки при навігації,

оскільки лише необхідні дані завантажуються з сервера, а не вся сторінка.

- Кращий користувацький досвід: SPA надають більш плавну та інтерактивну взаємодію без перезавантаження сторінок.
- Простота підтримки: Компонентна архітектура React та типізація TypeScript спрощують підтримку та розширення коду.
- Масштабованість: Можливість розбивати додаток на окремі компоненти дозволяє легше масштабувати та розвивати проект.

Загалом, поєднання React та TypeScript для розробки SPA забезпечує потужну платформу для створення сучасних, продуктивних та легких у підтримці веб-застосунків, що повністю відповідає вимогам програмної системи для обміну речами.

Tailwind CSS було обрано як бібліотеку для стилізації завдяки її утилітарному підходу, який дозволяє швидко та ефективно створювати адаптивні та красиві інтерфейси. На відміну від традиційних CSS-фреймворків, Tailwind надає можливість використовувати класи безпосередньо в HTML-розмітці, що значно прискорює процес розробки. Крім того, Tailwind легко інтегрується з React та забезпечує високу продуктивність за рахунок мінімізації використання зайвих стилів, також він легко налаштовується за допомогою відповідного конфігураційного файлу (рисунок 4.1).

Рисунок 4.1 – Конфігураційний файл Tailwind CSS (рисунок виконаний самостійно)

У цьому файлі для проекту було задано: кольорова палітра (стовпчик зліва), шрифти, додані до існуючих свої тіні, задні фони та точки роздільної здатності екранів (breakpoint).

RTK Query було обрано для роботи з API завдяки його простоті та ефективності. Ця бібліотека дозволяє легко здійснювати запити до серверу, обробляти відповіді та кешувати дані. RTK Query значно знижує кількість коду, необхідного для роботи з даними, та забезпечує простий і зрозумілий API для інтеграції з React-компонентами. Це робить його відмінним вибором для побудови масштабованих додатків з великою кількістю запитів до серверу. На рисунку 4.2 зображено використання цієї технології на прикладі авторизації.

```
export const apiSlice = createApi({
  reducerPath: "api",
  baseQuery: fetchBaseQuery({ baseUrl: `${import.meta.env.VITE_SERVER_URL}`, credentials: "include" }),
  endpoints: (builder) => ({
    //#region user
    login: builder.mutation({
      query: (credentials) => ({
        url: "/user/login",
        method: "POST",
        body: credentials,
      }),
    }),
export const {
  useLoginMutation,
  // #region ···
} = apiSlice;
const [login, { isLoading }] = useLoginMutation();
const { errors, touched, handleBlur, values, handleChange, handleSubmit } = useFormik({
  initialValues: { email: "", password: "" },
  onSubmit: async () => {
    try {
      const loginResp = await login({ email, password });
      if (loginResp.error) {
        toast.error(t("Wrong credentials"), { className: "!bg-error" });
        return;
      }
      const detailsResponse = await getDetails();
      if (detailsResponse.isError) {
        navigate(RoutesEnum.SIGN_UP, { state: { step: 1 } });
        return;
      }
      navigate(RoutesEnum.HOME);
    } catch (error) {
      toast.error(JSON.stringify(error), { className: "!bg-error" });
    }
  },
  validationSchema: loginSchema,
});
```

Рисунок 4.2 – Використання RTK Query на прикладі авторизації (рисунок виконаний самостійно)

Щоб авторизувати користувача потрібно зробити 3 простих дії. Перше – створити apiSlice в якому треба написати звернення до сервера та експортувати хук з однойменною назвою новоствореного методу, що надає цей інструмент. Друге – використати хук у потрібному компоненті та деструктуризувати метод який треба використати для авторизації. Третє – використати цей метод та передати в нього тіло з відповідними даними, що потребує сервер. Все дуже просто та прозоро, також RTK Query надає купу інструментів для роботи з запитам, починаючи від змінної, яка відповідає за стан завантаження інформації, закінчуючи роботою з кешем запитів.

Технологія i18n була обрана для локалізації додатку завдяки її потужності та гнучкості. Ця бібліотека підтримує динамічне завантаження мовних ресурсів та забезпечує простий API для перекладу інтерфейсу. Використання i18n дозволяє легко додавати нові мови та адаптувати додаток для різних ринків. Крім того, бібліотека добре інтегрується з React та забезпечує високу продуктивність.

React-router-dom було обрано для маршрутизації завдяки його простоті та широким можливостям. Ця бібліотека дозволяє легко визначати маршрути та керувати переходами між сторінками в додатку. Використання react-router-dom забезпечує гнучку та зручну навігацію, підтримку динамічних маршрутів та можливість створення захищених сторінок. Це робить його ідеальним вибором для побудови сучасних односторінкових додатків.

Formik було обрано для керування станом великих форм та валідації полів завдяки його простоті та ефективності. Formik забезпечує простий API для обробки форм, управління їх станом та валідації даних. Використання yup для валідації полів дозволяє легко створювати валідаційні схеми та забезпечує гнучкість у визначенні правил валідації. Ця комбінація дозволяє значно спростити роботу з формами та забезпечує надійну валідацію введених даних. На рисунку 4.3 зображений приклад використання цієї технології.

```
const { errors, touched, handleBlur, values, handleChange, handleSubmit, setFieldValue } = useFormik({
  initialValues: initialValues,
  onSubmit: async () => {
    try {
      const response = await addItem({
        name,
        description,
        color: color!.value,
        category: category!.value,
        state: parseInt(state!.value),
        wantedCategory: wantedCategory ? wantedCategory.map(({ value }) => value) : [],
        pictureIds,
      });
      console.log(response);
      if (response.error) toast.error(t("Something went wrong, try again later."), { className: "!bg-error" });
      else {
        toast.success(t("Item added successfully."), { className: "!bg-green100" });
        navigate(RoutesEnum.MY_ITEMS);
      }
    } catch (error) {
      toast.error(JSON.stringify(error), { className: "!bg-error" });
    }
  },
  validationSchema: addItemSchema,
});
```

Рисунок 4.3 – Приклад використання Formik (рисунок виконаний самостійно)

Проект був написаний з використанням саме хуку useFornik, він надає багато інструментів для роботи зі складними формами.

Всі ці технології взаємодіють між собою, утворюючи єдину, добре структуровану та ефективну архітектуру. VS Code забезпечує зручне середовище розробки, Vite прискорює процес побудови, React та TypeScript забезпечують надійний фундамент для розробки компонентів, Tailwind спрощує процес стилізації, RTK Query полегшує роботу з даними, i18n забезпечує багатомовність, react-router-dom управляє навігацією, а Formik та yup спрощують роботу з формами. Усі ці рішення разом утворюють потужну та гнучку платформу для створення сучасного веб-застосунку з мобільною версією.

Для мобільної версії програмної системи для обміну речами було обрано підхід прогресивних веб-застосунків (Progressive Web Apps, PWA). PWA поєднують найкращі риси веб- і мобільних застосунків, забезпечуючи користувачам швидкий доступ до функціональності через браузер, а також можливість встановлення на мобільний пристрій як рідного застосунку.

Використання PWA для мобільної версії програмної системи для обміну речами дозволяє забезпечити користувачам швидкий і зручний доступ до функціональності без необхідності завантажувати великий обсяг даних чи встановлювати додаткові програми. Це робить PWA ідеальним вибором для мобільних додатків, які потребують швидкого часу відгуку і високої доступності. Завдяки своїм перевагам, PWA можуть значно покращити користувацький досвід та забезпечити високу продуктивність мобільного застосунку.

4.3 Опис розроблених інтерфейсів

Під час розробки фронт-енд частини програмної системи для обміну речами було дотримано визначеного дизайну, про який йшла мова раніше. Більшість сторінок та елементів залишилися незмінними, але деякі з них зазнали значних чи не дуже змін, тому вважаю доцільним продемонструвати лише ті сторінки чи елементи які відрізняються від початкового дизайну та/або мають такі елементи, які не були продемонстровані на макеті.

Варто розпочати з головного – домашня сторінка, адже саме вона зазнала найбільших змін (рисунок 4.4).

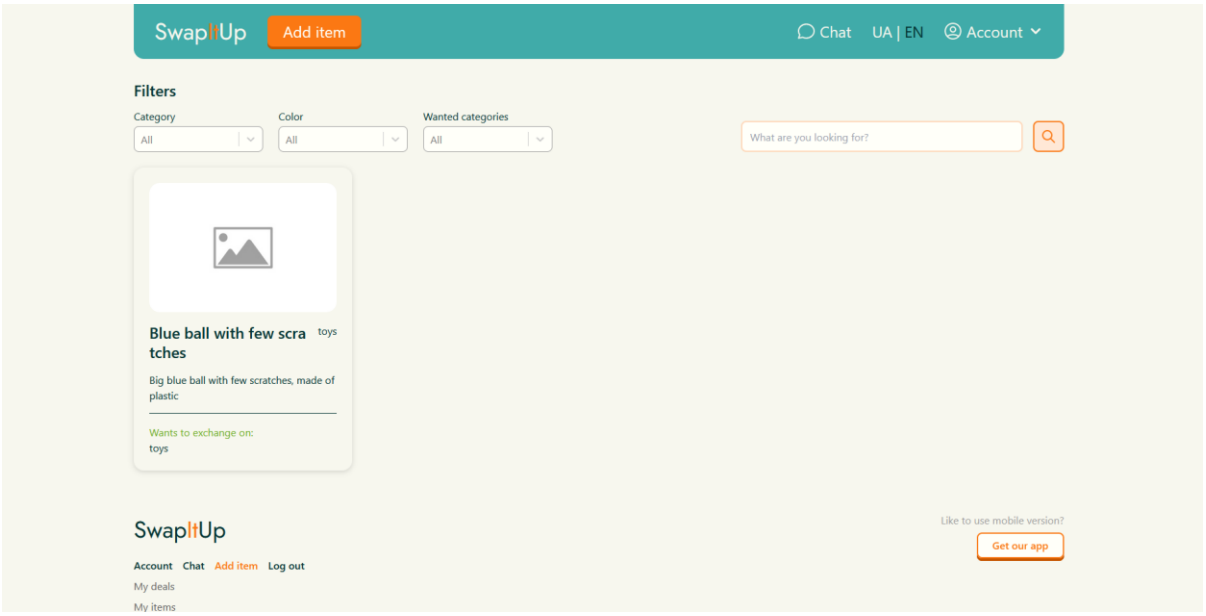

Рисунок 4.4 – Домашня сторінка розробленої програми (рисунок виконаний самостійно)

Невеликі зміни відбулися у хедері, додався «Підвал» сайту, але з того, що одразу стає помітним – це панель з пошуком та фільтрацією, раніше там був лише пошук, але довелося розширити дану панель шляхом додавання трьох фільтрів: за категорією до якої належить товар, за кольором товару та за бажаною категорією на яку хочуть обміняти товар.

Також деяких змін зазнала картка товару, тепер існує два її види: своя картка, та картка іншого користувача, приклад на рисунку 4.5.

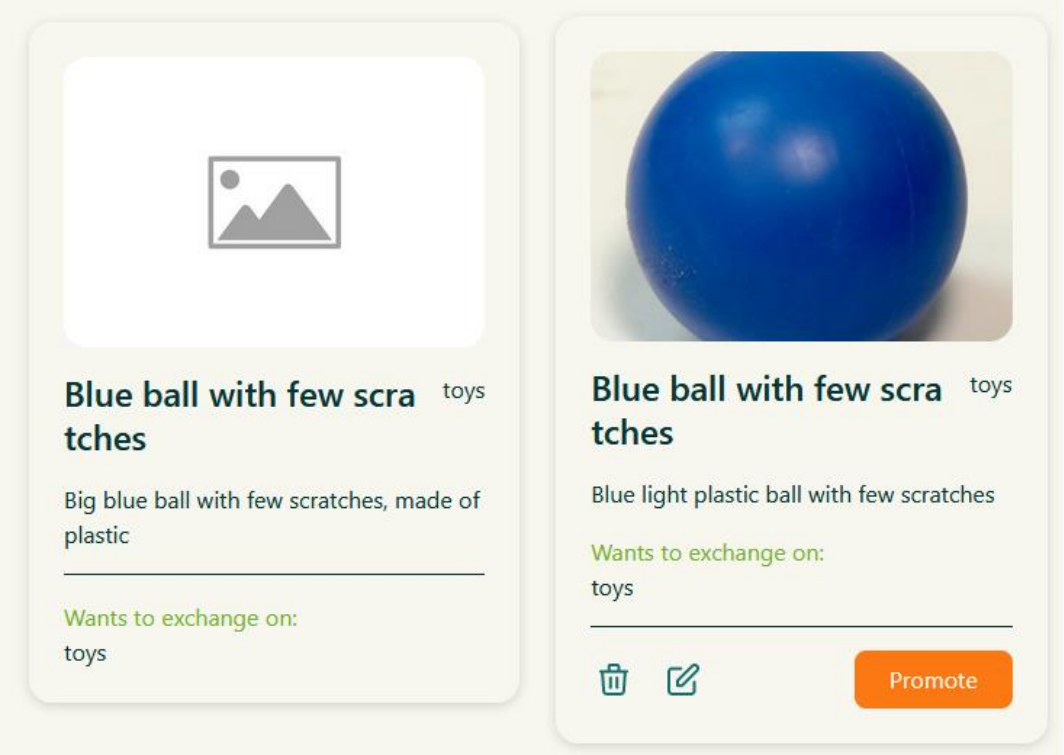

Рисунок 4.5 – Варіанти картки товару (рисунок виконаний самостійно)

На рисунку вище зображено два варіанти для картки товару, зліва – картка товару іншого користувача, а справа – картка свого товару. Тепер на чужій картці немає ніяких кнопок, а навігація на сторінку з подробицями про товар відбувається шляхом натискання на саму картку. А вже на варіанті картки власного товару є кнопки для просування та видалення з редагуванням інформації.

Варто сказати про валідаційні повідомлення та сповіщення користувача щодо певної дії, а точніше її статусу, адже цю. інформацію не можна знайти на макеті чи дизайні клієнтської частини. На рисунку 4.6 зображені декілька полів з повідомленням про валідаційну помилку.

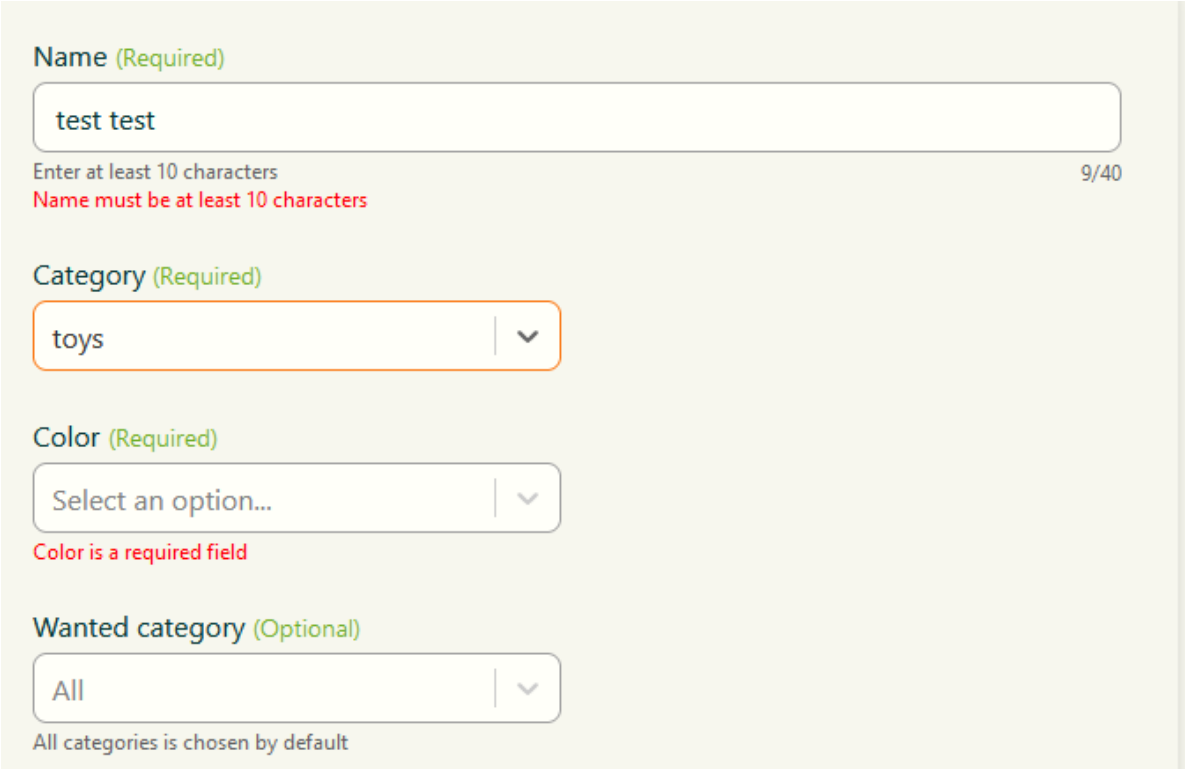

Рисунок 4.6 – Декілька полів з повідомленням про валідаційну помилку (рисунок виконаний самостійно)

З рисунка видно, що поля, які потребують заповнення сповіщають юзера про це, а також значення поля «Name» щонайменше повинно складатися з 10 символів, про що користувача сповіщає відповідне повідомлення. В свою чергу заповненні поля, які відповідають всім валідаційним правилам, не показують ніякої помилки.

Також програмна система для обміну речей сповіщає юзера про статус дії, яка щойно відбулася, приклад на рисунку 4.7.

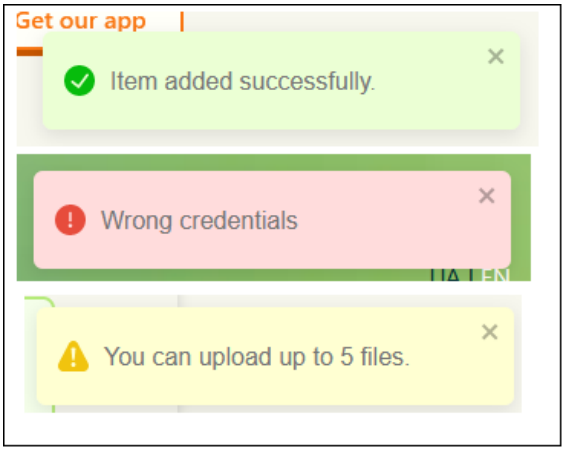

Рисунок 4.7 – Різні види сповіщення користувача (рисунок виконаний самостійно)

На рисунку вище зображено три види сповіщення: успіх – сповіщає користувача про дію, що була успішно виконана, наприклад товар успішно додався, помилка – сповіщає про якусь помилку під час дії, наприклад неправильні дані при авторизації, попередження – сповіщає про якусь дію, що не відповідає певним правилам програми, наприклад якщо під час додавання товару спробувати додати більше ніж 5 фотографій. Такі сповіщення важливо робити, аби забезпечити кращий досвід користування програмою користувачу [9].

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування є важливою частиною розробки програмного забезпечення, оскільки воно забезпечує перевірку коректності функціонування всіх компонентів системи. У процесі розробки програмної системи для обміну речами було застосовано ручне тестування для перевірки працездатності, стабільності та безпеки системи [10].

Під час ручного тестування було виконано низку перевірок, що включали заповнення різних форм з правильними та неправильними даними для оцінки коректності обробки введеної інформації. Це дозволило виявити можливі помилки валідації та забезпечити правильне функціонування системи при різних сценаріях введення. Також було перевірено роботу системи при використанні різних облікових записів, що дозволило оцінити коректність розмежування доступу та дотримання політики безпеки.

Основні кроки, які виконувались під час тестування, включали:

- Перевірка заповнення форм правильними та неправильними даними для перевірки валідації.
- Тестування створення, редагування та видалення об'єктів обміну для перевірки коректності CRUD операцій.
- Виконання дій під різними обліковими записами для перевірки правильності розмежування доступу.
- Перевірка автентифікації та авторизації для забезпечення безпеки даних користувачів.
- Тестування інтерфейсу користувача для забезпечення зручності та інтуїтивності користування системою.

Ці дії дозволили забезпечити високу якість програмного забезпечення та впевненість у його стабільному функціонуванні. Ручне тестування, хоча і є трудомістким процесом, виявилось ефективним засобом для перевірки різних аспектів системи, оскільки дозволило виявити та виправити помилки на ранніх

етапах розробки. Це забезпечило створення надійної та зручної у використанні системи для обміну речами, що відповідає вимогам користувачів та бізнесу.

Для прикладу нижче буде продемонстровано тестування форми реєстрації.

Крок 1: Введення невалідних даних.

На першому кроці тестування форми реєстрації було введено невалідні дані, що спричинило виникнення трьох різних помилок (рисунок 5.1):

– Вказано неіснуючий або неправильно відформатований електронний лист.
– Пароль не відповідає встановленим вимогам (наприклад, занадто короткий або не містить необхідних символів).
– Повторний пароль не співпадає з першим введеним паролем.

Після спроби надсилання форми, система коректно відобразила відповідні повідомлення про помилки для кожного з невірних полів, що підтверджує правильність реалізації валідації введених даних

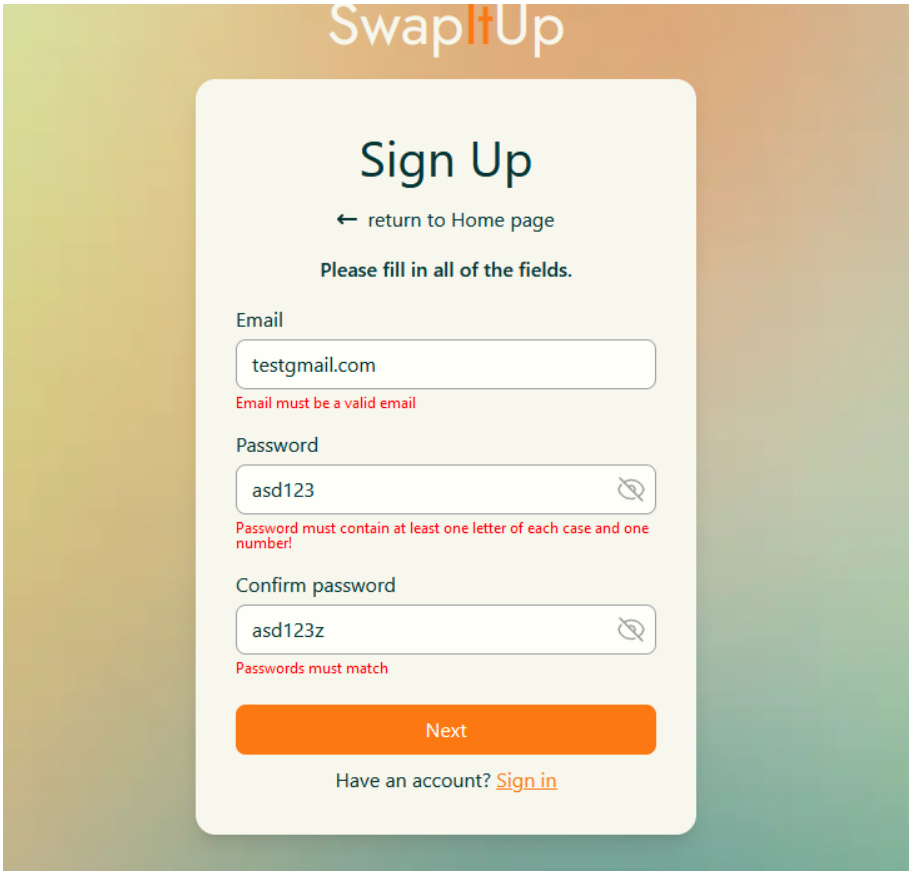

Рисунок 5.1 – Відображення трьох різних валідаційних помилок (рисунок виконаний самостійно)

Крок 2: Введення валідних даних.

На другому кроці було введено валідні дані у всі поля форми реєстрації (рисунок 5.2):

– Коректний електронний лист.

– Пароль, який відповідає усім вимогам безпеки.

– Повторний пароль, що співпадає з першим введеним паролем.

Після надсилання форми система успішно перейшла до другого кроку реєстрації, підтверджуючи правильність та повноту введених даних.

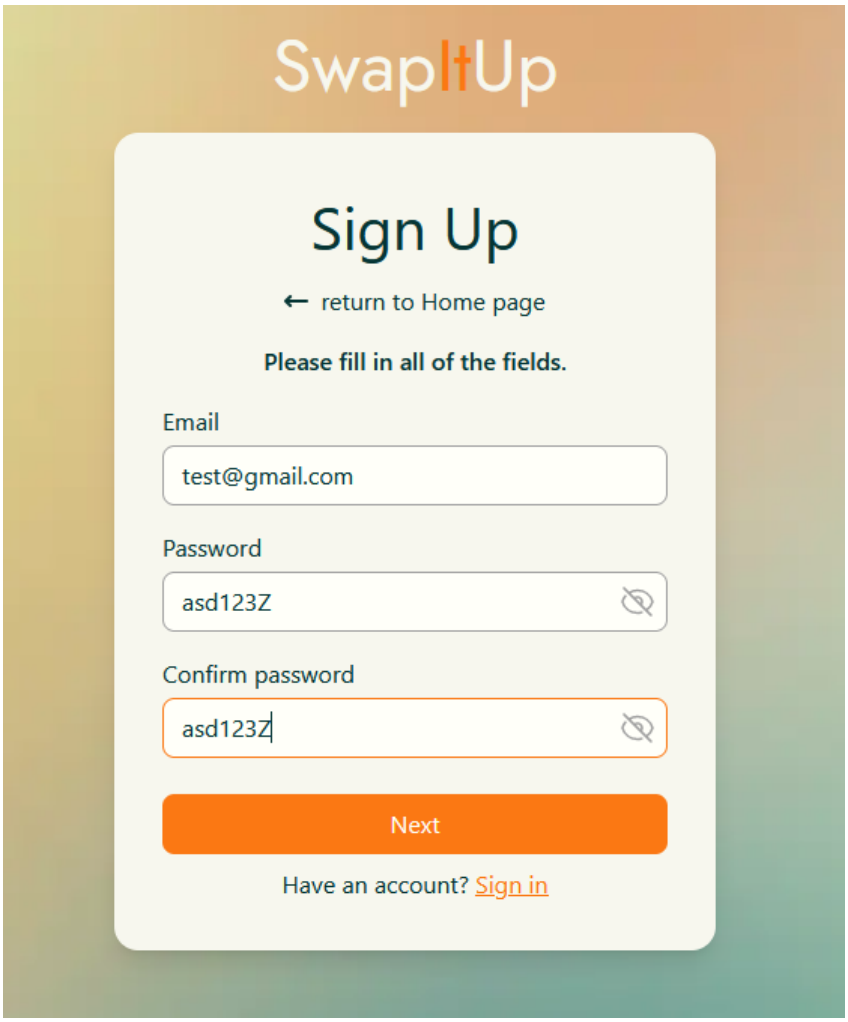

Рисунок 5.2 – Форма з коректними даними (рисунок виконаний самостійно)

Крок 3: Завершення реєстрації.

На третьому кроці було введено правильні дані у всі поля форми другого кроку реєстрації (рисунок 5.3). Після цього система успішно перенесла користувача на головну сторінку, вже авторизованим, що підтвердило успіх процесу реєстрації.

Цей процес підтвердив, що форма реєстрації працює коректно, забезпечуючи необхідний рівень валідації даних та успішне створення нового користувача.

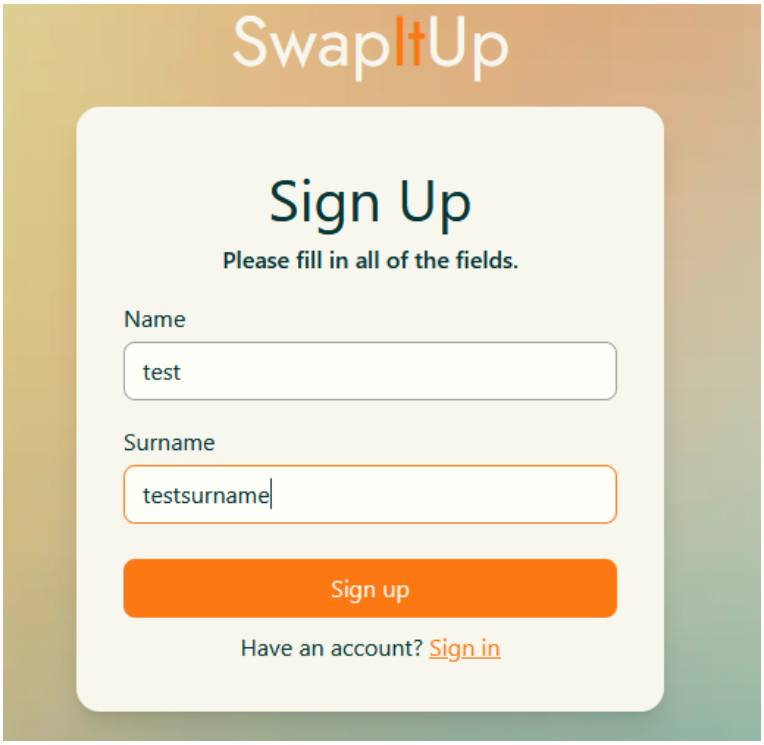

Рисунок 5.3 – Другий крок з коректними даними (рисунок виконаний самостійно)

Таким чином, проведене ручне тестування підтвердило, що розроблене програмне забезпечення є функціональним, безпечним та готовим до використання у реальних умовах.

ВИСНОВКИ

У результаті виконання даної кваліфікаційної роботи було сформовано ряд ключових висновків щодо програмної системи для обміну речами, які відображають основні аспекти проєкту та його потенційний вплив.

Аналіз предметної галузі допоміг визначити актуальність та значущість розробки системи для обміну речами. Існуючі аналоги, такі як Freecycle або OLIO, підкреслили важливість простоти та зручності інтерфейсу, а також безпечності обмінів. Сформульовані цілі проєкту відповідають виявленим проблемам та мають на меті створити інтуїтивно зрозумілу, зручну та безпечну платформу для обміну речами. Були описані головна функціональність та технології, за допомогою яких планується створити фронтенд даної системи. Були спроєктовані діаграма прецедентів, на якій описано як користувачі можуть взаємодіяти з системою та які можливості надає система, діаграма активностей, на якій описано процес реєстрації користувача. Була детально описана обрана архітектура клієнтської частини та надана її схема. Був представлений створений UI/UX системи, на прикладі деяких основних екранів платформи. А також продемонстровані зміни певних сторінок чи елементів під час розробки, повідомлення користувача про валідаційні помилки та сповіщення юзера про статус щойно виконаної дії.

Ручне тестування, проведене в процесі розробки програмної системи для обміну речами, дозволило забезпечити високу якість і стабільність функціонування системи. Завдяки ретельному тестуванню було підтверджено, що програмне забезпечення відповідає вимогам користувачів та бізнесу, є безпечним, зручним у використанні і готовим до впровадження в реальні умови.

Усі вищезазначені аспекти підкреслюють важливість та потенційну користь програмної системи для обміну речами. Правильно спроектована та реалізована система може стати не лише зручним інструментом для обміну речами, але й важливим кроком у розвитку свідомого споживання.

Результати даної роботи були представлені у вигляді тез на XXVIII Міжнародному молодіжному форумі «Радіоелектроніка та молодь у XXI столітті» (див. додаток А).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Graeber D. Debt: the first 5,000 years. Brooklyn, New York, USA : Melville House, 2011. 534 с.

2. Tidwell J., Brewer C., Valencia A. Designing interfaces: patterns for effective interaction design. O'Reilly Media, Incorporated, 2020. 500 с.

3. Progressive web apps | web.dev. *web.dev*. URL: https://web.dev/explore/progressive-web-apps (дата звернення: 09.05.2024).

4. Rational software architect 9.6.1. *IBM - United States*. URL: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case (дата звернення: 21.05.2024).

5. What is activity diagram?. *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/ (дата звернення: 21.05.2024).

6. What Is Three-Tier Architecture? | IBM. *IBM - United States*. URL: https://www.ibm.com/topics/three-tier-architecture (дата звернення: 02.05.2024).

7. Norman D. A. Design of everyday things. MIT Press, 1998. 270 с.

8. React best practices for single-page applications - graph. *Digital Consultancy & Transformation Agency - London*. URL: https://graph.digital/blog/react-best-practices (дата звернення: 07.05.2024).

9. Improving user experience: the art of notifications in UX design. *BTNG Unlimited - Design Subscription*. URL: https://www.btng.studio/insights/improving-user-experience-the-art-of-notifications-in-ux-design (дата звернення: 23.05.2024).

10. A balanced look: manual testing advantages and disadvantages. *KiwiQA*. URL: https://www.kiwiqa.com/manual-testing-advantages-and-disadvantages/ (дата звернення: 02.06.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

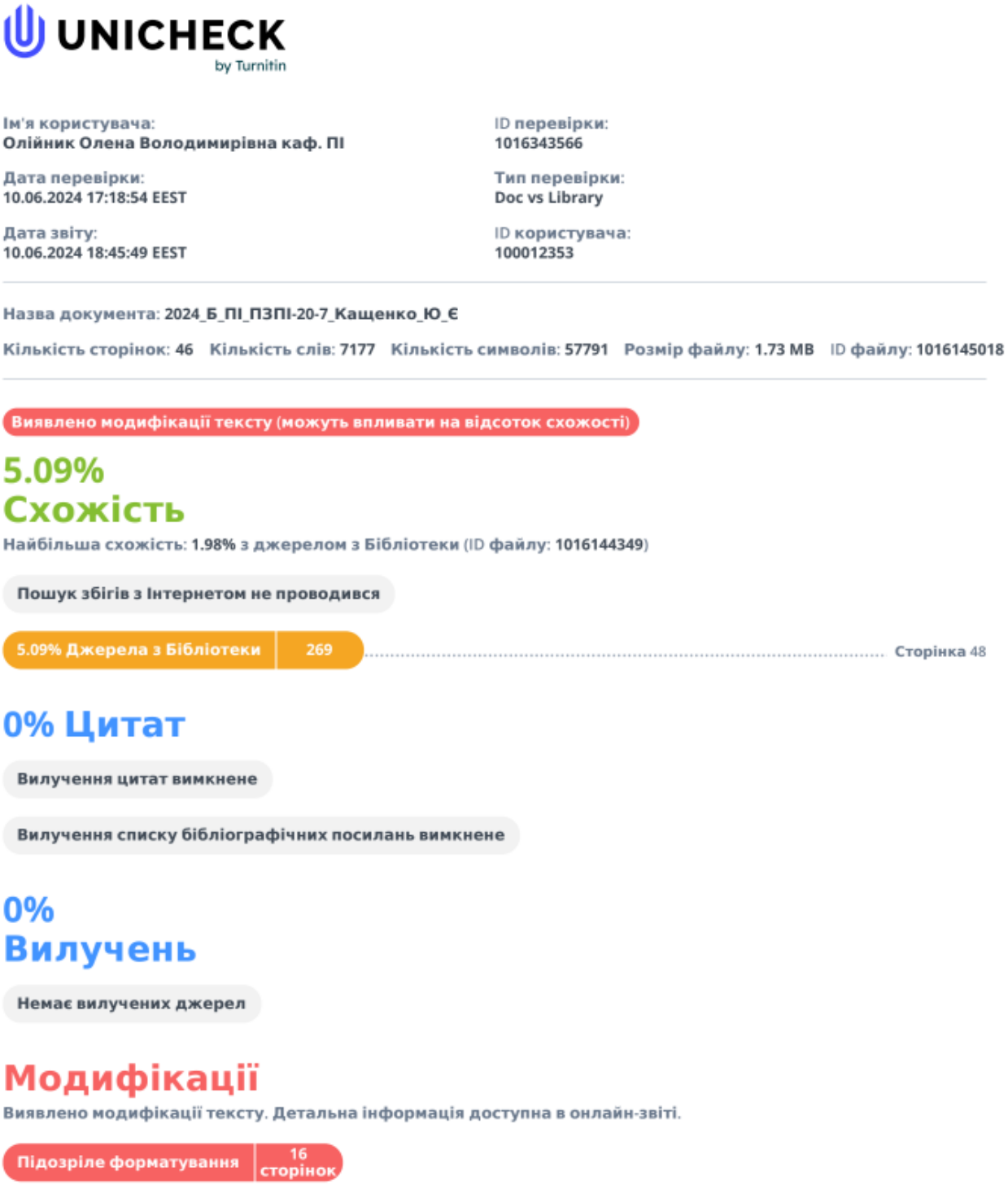

Рисунок А.1 – Звіт Unicheck

ДОДАТОК Б

Слайди презентації

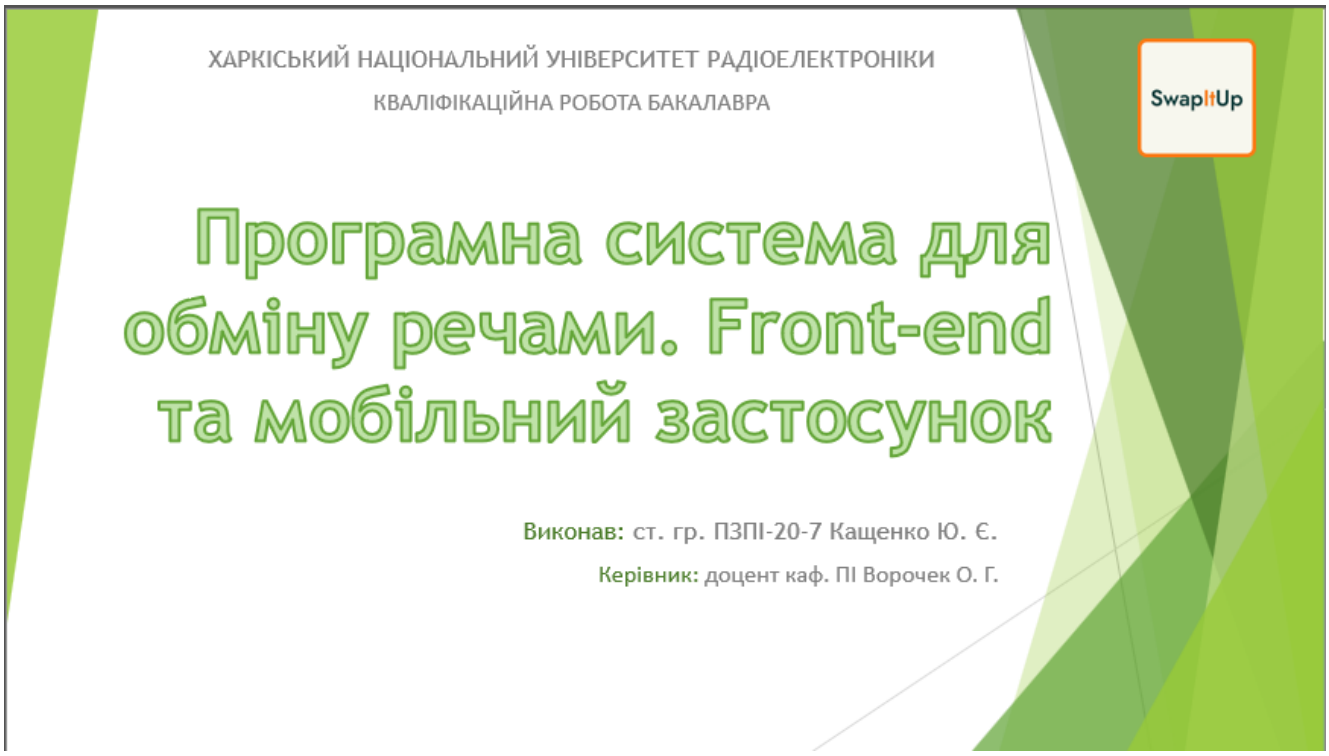

Рисунок Б.1 – Перший слайд

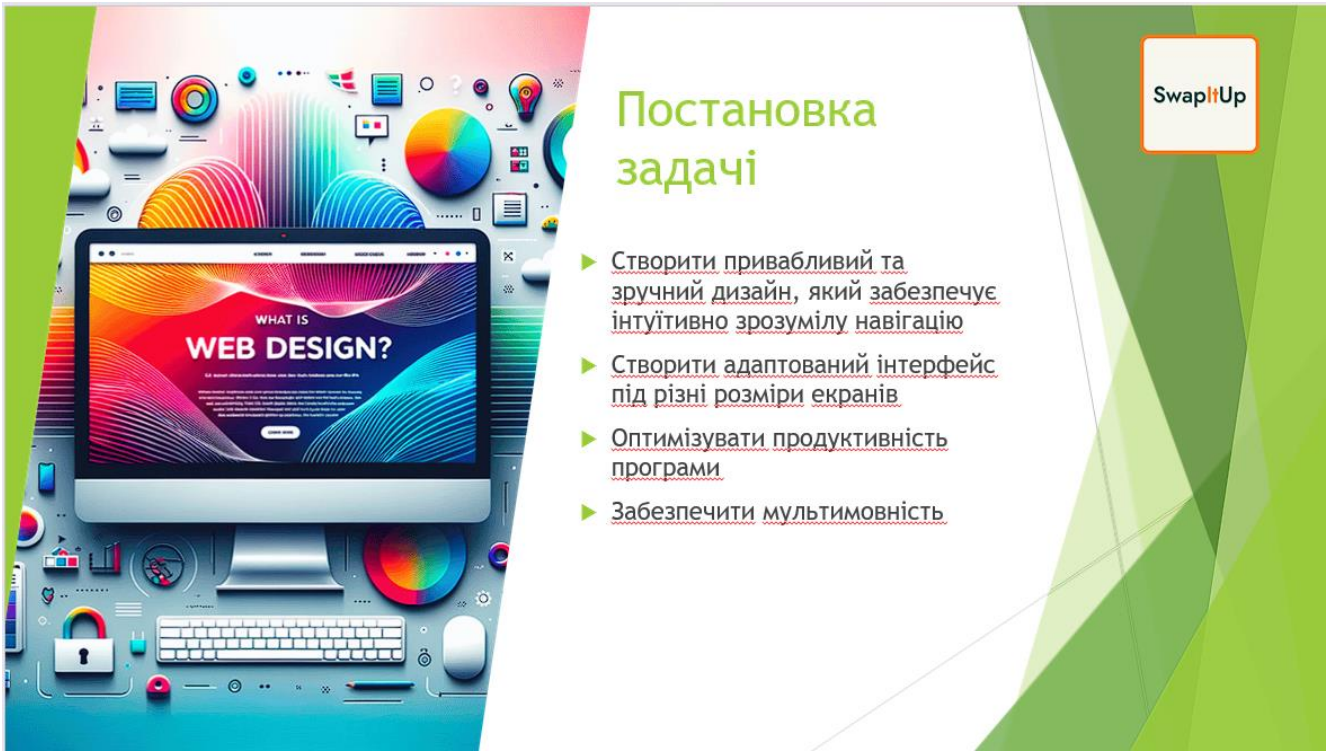

Рисунок Б.2 – Другий слайд

Рисунок Б.3 – Третій слайд

Рисунок Б.4 – Четвертий слайд

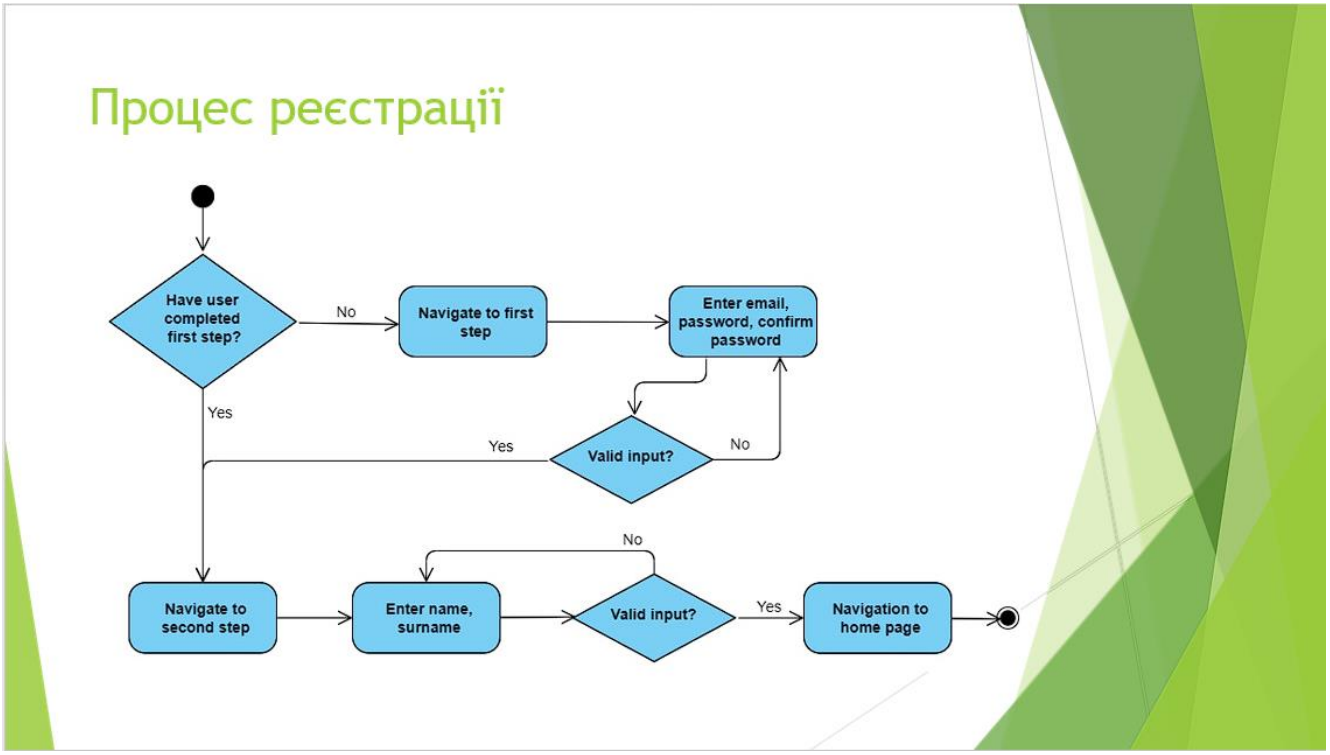

Рисунок Б.5 – П’ятий слайд

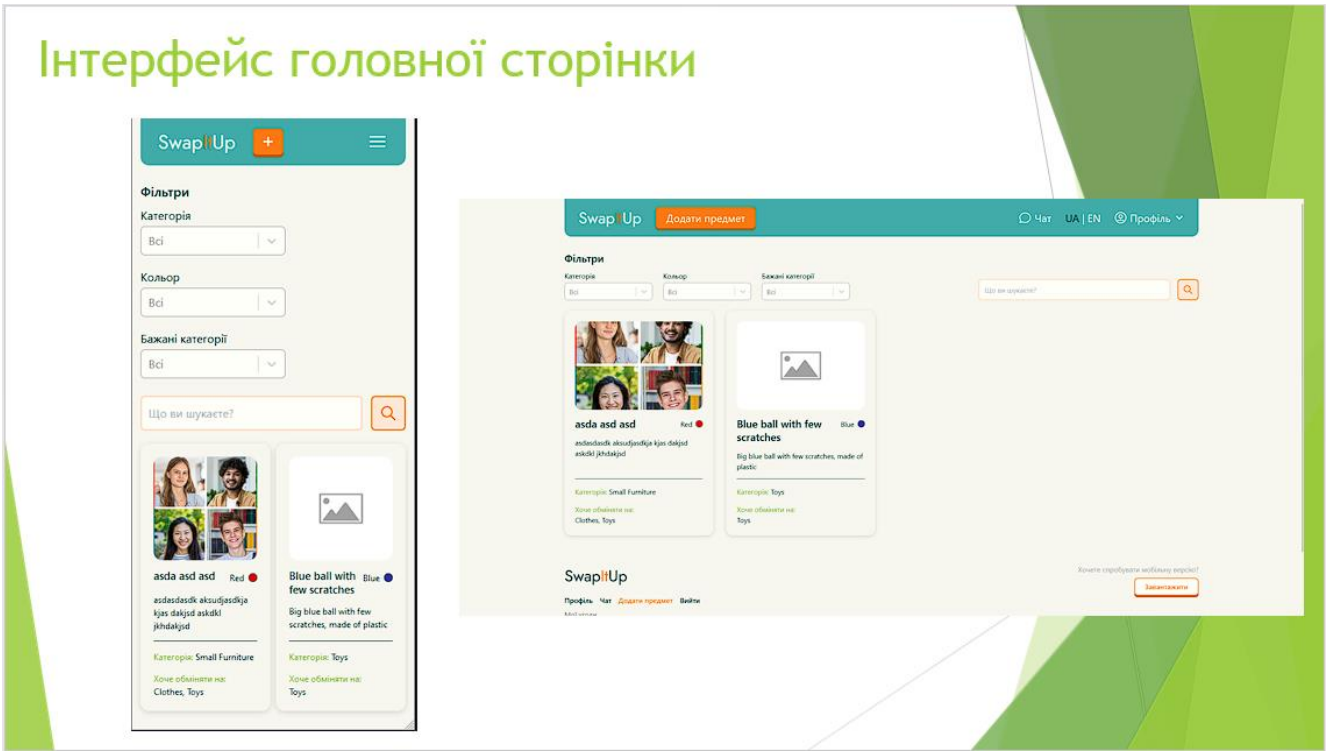

Рисунок Б.6 – Шостий слайд

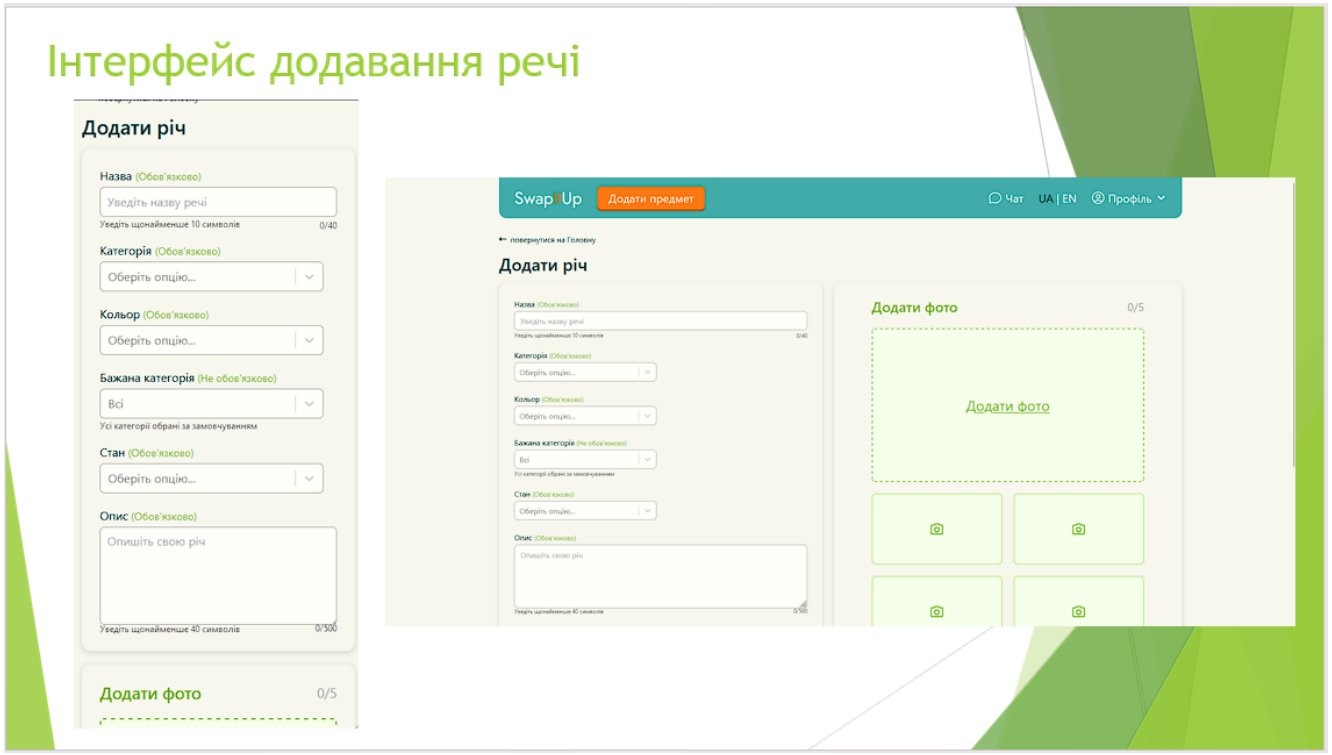

Рисунок Б.7 – Сьомий слайд

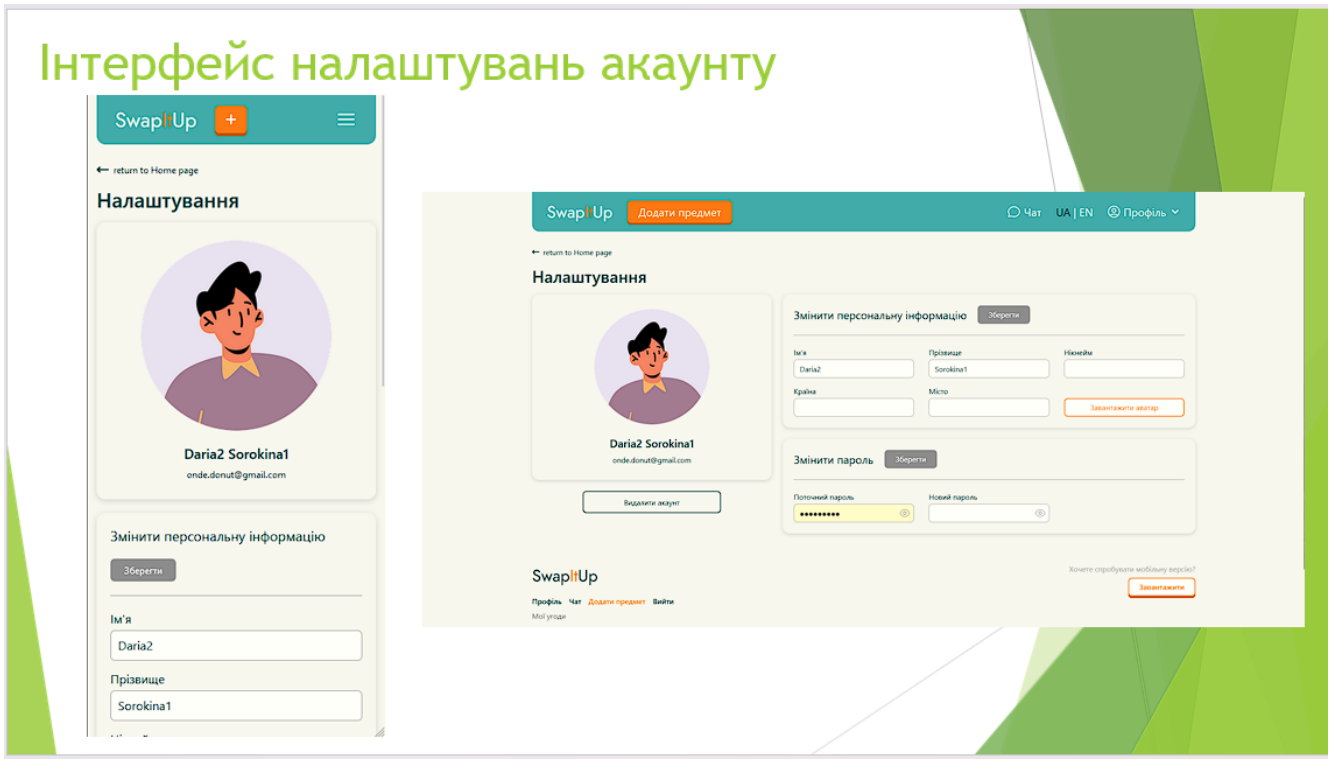

Рисунок Б.8 – Восьмий слайд

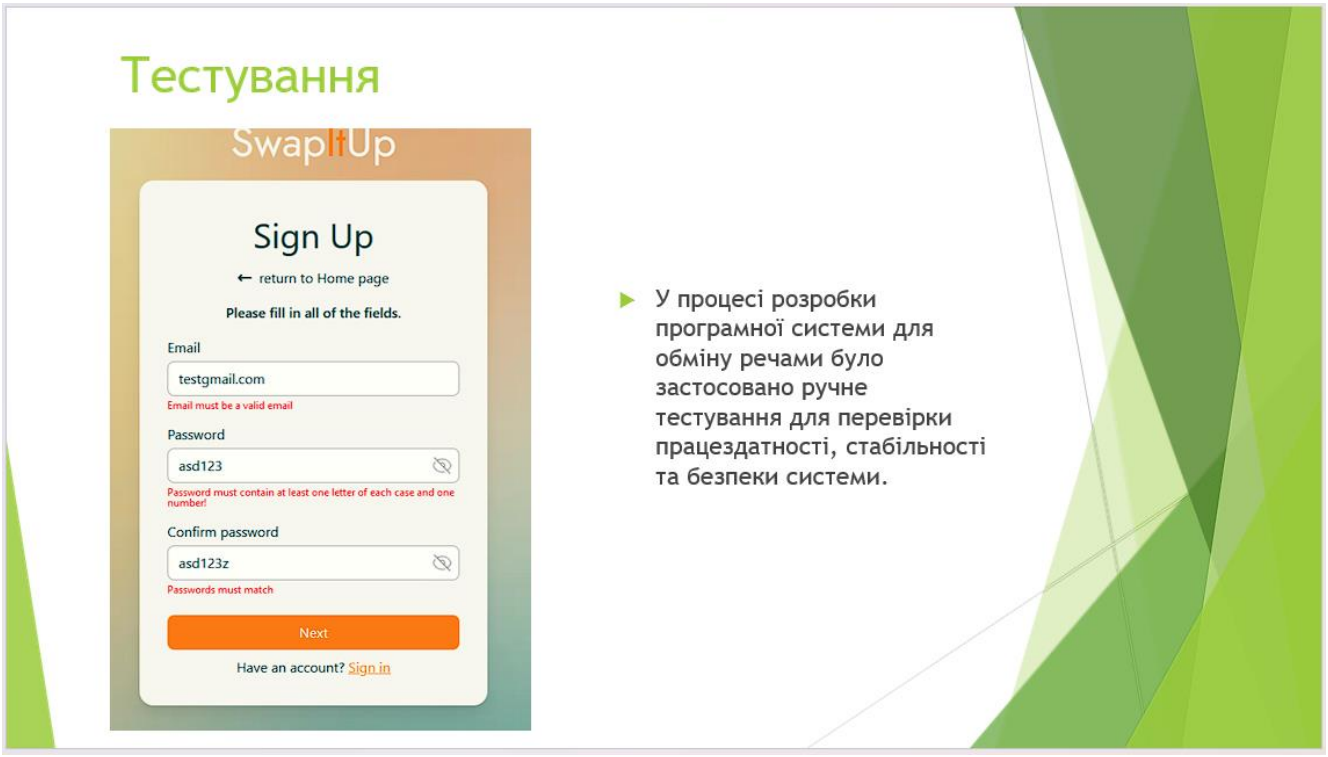

Рисунок Б.9 – Дев’ятий слайд

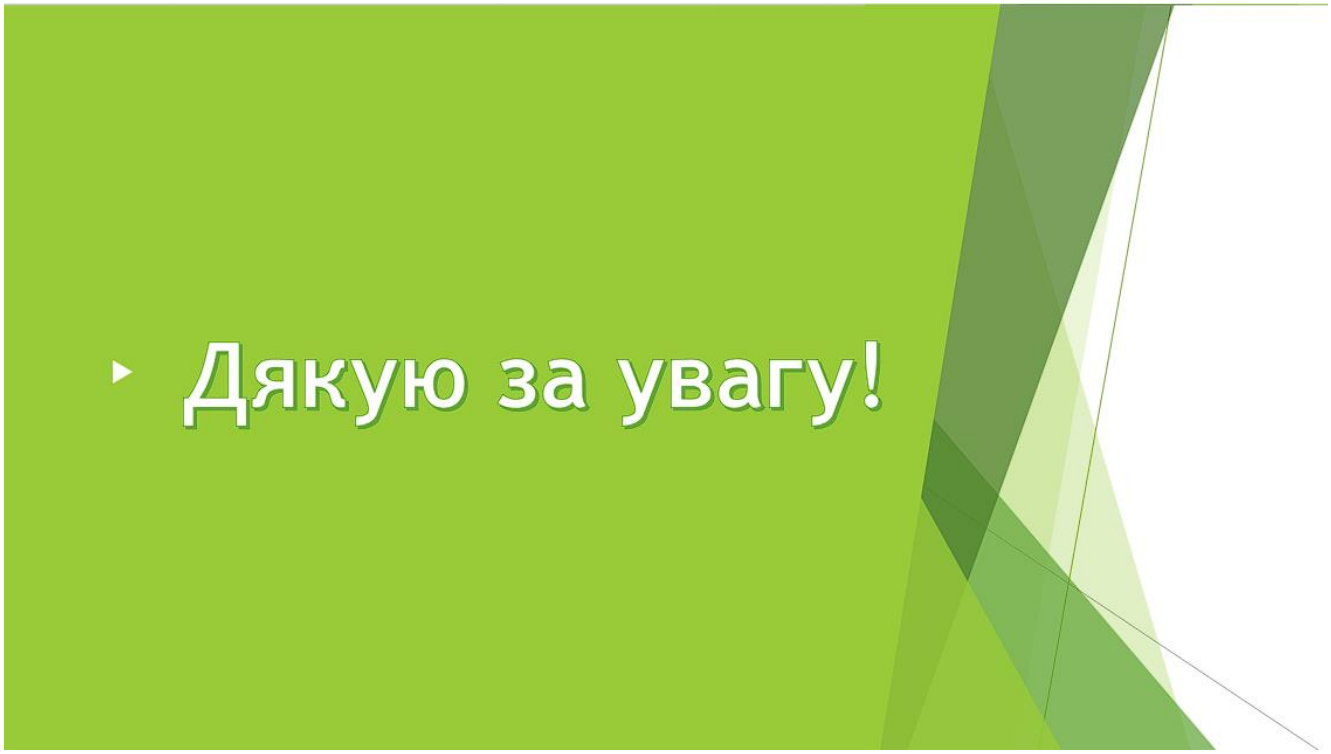

Рисунок Б.10 – Десятий слайд

ДОДАТОК В

Специфікація програмного продукту

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Факультет комп'ютерних наук

Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
для проєкту з темою
«Програмна система для обміну речами»

Виконав ст. гр. ПЗПІ-20-7 Кащенко Ю. Є.
Керівник роботи доцент кафедри ПІ Ворочек О. Г.

2024

Рисунок В.1 – Сторінка 1 специфікації ПЗ

1 ВСТУП

1.1 Огляд продукту

Програмна система для обміну речами – це інтернет-платформа, що дозволяє користувачам обмінюватися предметами, які їм більше не потрібні, на речі, які вони бажають отримати. Ця система забезпечує безпечний, ефективний та зручний процес обміну речами між користувачами.

1.2 Мета

Створення зручної та ефективної платформи, яка спростить процес пошуку речей та допоможе позбутися речей, які не потрібні, шляхом обміну.

1.3 Межі

Система обмежена функціоналом обміну речами, управлінням обліковими записами користувачів, створенням оголошень товарів та спілкуванням між користувачами. Вона не включає функціонал продажу речей за гроші.

1.4 Посилання

1. Tidwell J., Brewer C., Valencia A. Designing interfaces: patterns for effective interaction design. O'Reilly Media, Incorporated, 2020. 500 с.

2. Progressive web apps | web.dev. *web.dev*. URL: https://web.dev/explore/progressive-web-apps.

3. Hengkyawan J. Clean Architecture in ASP .NET Core Web API. Medium. URL: https://juldhais.net/clean-architecture-in-asp-net-core-web-api-4e5ef0b96f99.

4. Norman D. A. Design of everyday things. MIT Press, 1998. 270 с.

1.5 Означення та абревіатури

PWA (Progressive Web Application) – прогресивний веб-додаток, що поєднує в собі кращі властивості веб-сайтів та мобільних додатків.

Рисунок В.2 – Сторінка 2 специфікації ПЗ

UI (User Interface) – інтерфейс користувача, система засобів, що забезпечують взаємодію користувача з програмою.

UX (User Experience) – досвід користувача, загальне враження та досвід, який отримує користувач під час взаємодії з продуктом.

API (Application Programming Interface) – інтерфейс програмування додатків, набір правил та механізмів, що дозволяють взаємодіяти різним програмним компонентам.

REST (Representational State Transfer) – стиль архітектури для створення веб-сервісів, що дозволяє взаємодіяти з ресурсами через стандартні HTTP-методи.

CRUD (Create, Read, Update, Delete) – базові операції для управління даними в інформаційних системах.

SPA (Single Page Application) – односторінковий додаток, що завантажує одну HTML-сторінку та динамічно оновлює контент на ній, не перезавантажуючи сторінку повністю.

Рисунок В.3 – Сторінка 3 специфікації ПЗ

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Програмна система для обміну речами має значний потенціал розвитку та впливу на різні сфери життя суспільства. Завдяки актуальності теми обміну речами, кількість користувачів системи може суттєво зрости, адже люди все частіше шукають способи зменшити свій екологічний слід та зекономити гроші. Люди більше спілкуватимуться та взаємодіятимуть між собою, що зміцнить соціальні зв'язки в межах окремих громад та міст. Екологічна стійкість платформи полягає в зменшенні кількості відходів, оскільки речі будуть повторно використовуватися, а не викидатися, що допоможе зменшити навантаження на сміттєзвалища та зберегти природні ресурси.

Використання сучасних технологій забезпечує високий рівень продуктивності та зручності користування, відкриваючи можливості для подальшого розвитку та впровадження нових функцій, таких як інтеграція з соціальними мережами. Програмна система також може залучити до співпраці різні організації, включаючи благодійні фонди, екологічні рухи, місцеві уряди та бізнеси, що підсилить вплив та розширить можливості платформи.

2.2 Функції продукту

Функціонал, доступний звичайному користувачу:

- реєстрація у системі;
- авторизація та автентифікація;
- редагування особистих даних;
- додавання оголошень про товари;
- редагування інформації товарів;
- видалення оголошень;
- пошук товарів;
- фільтрація товарів за певними параметрами;
- чат з іншими користувачами;

Рисунок В.4 – Сторінка 4 специфікації ПЗ

– створення обмінних угод;
– оцінювання обмінних угод.

Функціонал, доступний адміністратору:

– авторизація та автентифікація;
– створення категорій;
– видалення категорій;
– додавання кольорів;
– видалення кольорів.

2.3 Характеристики користувачів

У системі існують рівні доступу, або користувачів, які визначають, у кого є доступ до даних та функцій. Таких рівнів двоє, а саме:

– «user», який надається користувачам, які авторизовані у системі, у них є доступ до усіх основних функцій системи;
– «admin», який надається певному користувачу, який вже створений у системі, йому доступні функції адміністрування у системі;

Це розділення допомагає захистити систему від несанкціонованого доступу, забезпечує належне управління доступом, мінімізує ризики витоку конфіденційної інформації та і, загалом, забезпечує її стабільну та безпечну роботу.

2.4 Загальні обмеження

Для забезпечення стабільної та безпечної роботи програмної системи для обміну речами, користувачі мають дотримуватися певних обмежень та правил. Тож серверна частина даної системи має такі обмеження:

– користувачі можуть завантажити до 5 зображень для свого товару;
– певні поля, наприклад, назва товару має містити мінімум 10 та максимум 70 символів;
– користувач має пройти процес авторизації та аутентифікації, щоб отримати доступ до майже усіх функцій;

Рисунок В.5 – Сторінка 5 специфікації ПЗ

– до функцій, які пов'язані з категоріями та кольорами, має доступ тільки користувач з роллю адміністратора;
– деякі поля є обов'язковими для заповнення.

Клієнтська частина має такі обмеження:

– для роботи PWA у користувача має бути будь-який браузер, окрім Safari та Firefox;
– у користувача має бути стабільний інтернет;
– користувач має знати або українську, або англійську мову;
– у користувача буде можливість завантажити обмежену кількість фото для товару.

2.5 Припущення та залежності

Припущення:

– передбачається, що користувачі будуть зацікавлені в обміні речами як екологічно дружньому та економічно вигідному способі отримання потрібних речей;
– передбачається, що користувачі матимуть стабільний доступ до інтернету, необхідний для роботи веб-додатку та PWA;
– передбачається, що користувачі будуть активні у взаємодії один з одним;
– припускається, що команда розробників матиме достатні технічні ресурси, знання та час для розробки, впровадження та підтримки системи.

Залежності:

– система повинна відповідати законодавчим вимогам щодо зберігання та обробки даних, захисту персональної інформації та електронної комерції;
– розвиток системи залежить від активного зворотного зв'язку користувачів, що дозволяє визначати необхідні покращення та нові функції, які відповідають потребам користувачів.

Рисунок В.6 – Сторінка 6 специфікації ПЗ

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Повинен бути інтуїтивно зрозумілим, зручним та сучасним. Користувачі повинні мати можливість реєструватися та входити у систему, переглядати, шукати та фільтрувати доступні речі для обміну. Інтерфейс повинен дозволяти додавання нових речей для обміну з можливістю додавання фото та опису, створення обмінних угод з можливістю прийняття або відхилення їх.

3.1.2 Апаратний інтерфейс

Має підтримувати роботу на мобільних пристроях (смартфонах і планшетах) та персональних комп'ютерах. Він повинен бути сумісним з різними операційними системами.

3.1.3 Програмний інтерфейс

Повинен бути реалізований RESTful API для доступу до усіх функцій. Серверна частина повинна взаємодіяти з базою даних PostgreSQL.

3.1.4 Комунікаційний інтерфейс

Взаємодія між частинами програмної системи здійснюється завдяки підтримки REST API з використанням HTTP запитів та даних формату JSON. Кожна частина повинна реалізувати можливість обробки HTTP запитів завдяки відповідних бібліотек.

Взаємодія між частинами програмної системи здійснюється за допомогою REST API з використанням HTTP запитів та даних формату JSON. Кожна частина повинна реалізувати можливість обробки HTTP запитів.

Рисунок В.7 – Сторінка 7 специфікації ПЗ

3.1.5 Обмеження пам’яті

Система повинна ефективно використовувати доступну пам'ять, забезпечуючи стабільну роботу навіть під високими навантаженнями. Це включає оптимізацію запитів до бази даних та використання кешування для зменшення навантаження на сервер.

3.1.6 Операції

Основні операції включають реєстрацію та авторизацію користувачів, створення та управління оголошеннями, пошук та фільтрацію товарів, організацію обмінів, обмін повідомленнями між користувачами та підтримку адміністративних функцій системи.

3.1.7 Функції продукту

Функціональність системи включає всі основні операції, необхідні для повноцінного обміну речами між користувачами, з урахуванням безпеки, надійності та продуктивності.

3.2 Властивості програмного продукту

Програмне забезпечення повинно мати такі властивості, як надійність та захищеність даних, мати можливість масштабуватись, доносити необхідну інформацію до майбутніх користувачів, відповідати усім нормам користування ПЗ.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система повинна бути надійною та стійкою до збоїв, забезпечуючи безперервну роботу і доступність функцій для користувачів у будь-який час.

Рисунок В.8 – Сторінка 8 специфікації ПЗ

3.3.2 Доступність

Система повинна бути доступною для користувачів у будь-який час, мінімізуючи час простою та забезпечуючи безперебійну роботу.

3.3.3 Безпека

Система повинна забезпечувати захист даних користувачів та запобігати несанкціонованому доступу, включаючи використання шифрування даних та механізмів автентифікації.

3.3.4 Супроводжуванність

Код системи повинен бути зрозумілим та легким для підтримки, оновлення та розширення функціональності. Це включає дотримання принципів модульності та використання документованих API.

3.3.5 Переносимість

Система повинна підтримувати роботу на різних платформах та пристроях, забезпечуючи однаковий користувацький досвід незалежно від пристрою.

3.3.6 Продуктивність

Система повинна забезпечувати високу продуктивність, здатну обробляти велику кількість одночасних запитів без значних затримок або зниження якості обслуговування.

3.4 Вимоги бази даних

База даних повинна бути масштабованою та бути готовою до великого об’єму даних, який слід обробити у доволі швидкий проміжок часу.

3.5 Інші вимоги

Інші вимоги включають дотримання всіх стандартів та норм, необхідних

Рисунок В.9 – Сторінка 9 специфікації ПЗ

для забезпечення коректної роботи системи, включаючи вимоги до безпеки, конфіденційності даних та відповідність законодавчим нормам.

Рисунок В.10 – Сторінка 10 специфікації ПЗ

ДОДАТОК Г

Тези за темою атестаційної роботи

УДК 004.9:504 DOI: https://doi.org/10.30837/IYF.IIS.2024.656

ПРОГРАМНА СИСТЕМА ДЛЯ ОБМІНУ РЕЧАМИ ЯК ЗАСІБ ПІДТРИМКИ 3R-ПІДХОДУ РОЗУМНОГО СПОЖИВАННЯ

Сорокіна Д. Є., Кащенко Ю. Є.
Науковий керівник – к.т.н., доцент Ворочек О. Г.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: daria.sorokina@nure.ua, yukhym.kashchenko@nure.ua

This work describes a current problem of modern society – the inefficient use of resources due to insufficient exchange of things between people. The focus is on the need to create a software system to facilitate this process. This work justifies the need for this system due to the increase in consumption and waste, which has a negative impact on the environment. The software system offers an effective mechanism for sharing things between users to reduce waste and encourage a culture of resource use. The report also discusses the possible benefits to society, such as cost savings, promotion of resource recovery and support for social mutual assistance. It highlights the key features and benefits of the proposed software system aimed at optimizing the exchange of things and preserving the environment.

У сучасному світі все більше людей цінують екологічний спосіб життя та усвідомлюють необхідність вторинного використання товарів для збереження ресурсів і зменшення кількості відходів. Програмна система для обміну речами призначена для створення зручного та ефективного середовища для обміну різноманітними предметами між користувачами. Вона надає можливість людям обмінюватись з іншими користувачами речами, які їм більше не потрібні, що допомагає зменшити виробництво нових товарів і споживання ресурсів. Дана програма є не просто технічним інструментом, але і стратегічним компонентом, що реагує на зростаючі потреби користувачів [1]. Основна ідея даної системи полягає у створенні онлайн-платформи, на якій користувачі зможуть розміщувати оголошення про речі, якими вони хочуть обмінятися або які шукають. Система допомагає з'єднати людей, які потребують обміну, надаючи їм зручні інструменти для пошуку, перегляду та контакту з іншими користувачами. Домовленості про обмін можуть відбуватися у мобільному додатку або у веб-застосунку.

З огляду на зростаючу увагу до екологічних проблем, потрібність створення системи для обміну речами є стратегічним кроком до шляху сталого споживання. Система сприяє переходу від традиційної моделі споживання, заснованої на безперервному придбанні нових товарів, до більш сталої моделі, яка базується на збереженні та використанні наявних ресурсів. Чим більше товарів обмінюється між користувачами, тим менше товарів потрапляє на сміттєзвалище. Це допомагає зменшити обсяги

Рисунок А.1 – Перша сторінка тез

відходів та викидів, пов'язаних зі звичайним викиданням непотрібних речей.

У поточних реаліях, коли бренди прагнуть якомога швидше продати свій одяг і у якомога більших кількостях, випускаючи нові колекції наче конвеєр, індустрія моди є одним з головних причин забруднення нашого довкілля [2]. На швейну промисловість припадає 10% від загальних викидів вуглекислого газу щороку, вона також є другим найбільшим споживачем прісної води, на її частку припадає 20% усіх промислових забруднень води [3]. Таким чином розробка системи для обміну речами не тільки дозволяє уникнути надмірного споживання, а разом з цим зменшує споживання природних ресурсів та викиди вуглецю, що пов'язані з виробництвом та транспортуванням.

Ще одним популярним явищем на сьогодні є «буккросинг», процес звільнення книг, що рятує книги від безцільного знаходження на полицях [4]. Програмна система для обміну речами допоможе буккросингу розширити географічно область своїх дій, що призведе до збільшення кількості книг, якими можна обмінюватися та розширення меж обміну. Крім того, вона полегшить відстеження руху книг та також прибере необхідність фізичних зустрічей для учасників. Це зробить процес обміну більш ефективним та зручним.

Наша система пропонує можливість використання як мобільного застосунку, так і веб-застосунку для пошуку товарів та проведення обмінів. Для розробки серверної частини програмної системи використовуються мови C# та технології ASP.NET Core 7, у якості бази даних обрано PostgreSQL. Клієнтська частина реалізується за допомогою технології React на основі мови JavaScript. Для мобільної версії системи доцільне використання фреймворку React Native.

Аналогічні програмні рішення пропонують різні підходи та можливості, відзначаються власними перевагами та недоліками. Представником такого типу систем є відома платформа для торгівлі OLX. Її перевагами є: широка популярність даної платформи і доступність її у багатьох країнах, але разом з цим головним недоліком є те, що OLX не спеціалізується на обміні товарами, а, в першу чергу, є торговельним майданчиком. Іншим доступним аналогом є Facebook Marketplace. Його головною перевагою є те, що дана платформа є частиною соціальної мережі Facebook, що надає велике охоплення та зручну комунікацію. Разом з цим це є і недоліком даного аналогу, так як багато людей не мають акаунту у Facebook, а створювати його лише для можливості доступу до Marketplace буде для когось недоречним. Також ще одним недоліком є те, що на даній платформі більш розповсюджена торгівля, а не обмін.

Усе вищеозначене зумовлює доцільність запропонованої програмної системи задля вирішення задач формування більш зручного та

Рисунок А.2 – Друга сторінка тез

сприятливого для людини середовища, в якому панує ефективне використання та споживання ресурсів.

Список використаних джерел:

1. Куценко А., Колесник О. Розробка компоненту системи веб-застосунку «каталог одягу». Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві: Всеукр. наук.-практ. конф. здобувачів вищ. освіти і молодих вчених, 22 листопада 2023 р. Харків. нац. автомоб.-дор. ун-т., Харків, 2023. С. 198–201.
2. Вплив швидкої моди на екологію та цінність сприйняття одягу. Dubhumans. URL: https://dubhumans.com/vplyv-shvydkoi-mody-na-ekolohiiu-ta-tsinnist-sprryiniattia-odiahu/ (дата звернення: 29.02.2024).
3. Industrial water usage to produce these items | the71percent. The 71 Percent. URL: https://www.the71percent.org/industrial-water-usage/ (дата звернення: 29.02.2024).
4. Буккросинг. URL: https://pulyny.school.org.ua/bukkrosing-09-38-29-05-01-2022/ (дата звернення: 29.02.2024).

Рисунок А.3 – Третя сторінка тез