Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

«СИСТЕМА УПРАВЛІННЯ КОНТЕНТОМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ (за прикладом сервісу Comfy) (Frontend)»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту:

№	П.І.Б	Група
1	Ісаєнко Д.О	КН-П-191
2	Бондарчук М.Є	КН-П-191
3	Єлізар`єв В.А	КН-П-191
4	Лавров А.В	КН-А-191
5	Сідху Т.	КН-Д-192
6	Тітова О.Є	КН-Д-192

Автор звіту:
Бондарчук Михайло Євгенович

Одеса – 2023

АНОТАЦІЯ

Дипломний проєкт присвячений розробці Front-end частини онлайн-гіпермаркету на прикладі Comfy.ua. Проєкт актуальний, оскільки онлайн-торгівля стає все більш популярною серед користувачів, особливо в контексті зростання електронної комерції.

Робота включає дослідження, розробку та тестування інтерфейсу користувача для веб-додатка, який дозволяє зручно взаємодіяти з онлайн-гіпермаркетом електронної торгівлі з великим асортиментом товарів.

Проєкт передбачає різноманітні функції, включаючи авторизацію, реєстрацію, аутентифікацію, пошук, фільтрацію та сортування товарів, систему відгуків та питань до товарів та інше.

Об'єктом дослідження у рамках даної роботи виступає розробка з використанням сучасних технологій, таких як HTML, CSS, JavaScript та одного з популярних фреймворків ReactJS, який використовується для покращення ефективності коду.

ЗМІСТ

ВСТУП

Інтернет-технології надають безліч можливостей для спрощення і поліпшення різних сфер життя, включаючи торгівлю. Зростання популярності онлайн-шопінгу та зростання попиту на зручні та ефективні способи покупок стимулюють розвиток глобального електронного ринку. У цьому контексті виникає потреба в рішеннях, що дозволяють споживачам здійснювати покупки онлайн зі зручністю та ефективністю, а продавцям — розширювати свій бізнес та досягати нових ринків та аудиторій.

Онлайн гіпермаркети виступають як відповідь на ці вимоги, поєднуючи широкий асортимент товарів, зручний інтерфейс та безліч послуг, що полегшують процес покупок.

У даному документі розглядається концепція і розробка онлайн гіпермаркету, який відповідає потребам споживачів і сприяє ефективному функціонуванню бізнесу. Основною метою дослідження є створення інноваційної цифрової платформи, яка забезпечує безперервний доступ до широкого спектра товарів, зручну навігацію та високу якість обслуговування.

Автореферат розпочинається аналізом технічних вимог до проєкту. Далі проводиться проєктування архітектури онлайн гіпермаркету, враховуючи вимоги зручності, безпеки та ефективності. Потім розглядаються етапи розробки, включаючи вибір технологій, розробку серверної частини.
Висновки дослідження демонструють результати, отримані під час розробки, тестування та використання онлайн гіпермаркету. В цілому, автореферат дозволяє зрозуміти сутність та переваги цього рішення.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

Загальний опис

Програмне забезпечення для створення інформаційної системи інтернет-крамниці для організації роздрібної інтернет торгівлі.

Призначення

Надання в Інтернет просторі інформації про доступні товари (оферти продавця) та організації оформлення замовлень (акцепту покупця) через мережу Інтернет. Оперативне управління веб-контентом Інтернет-крамниці. Збереження даних та регулювання доступу до них. Організація взаємодії з іншими інформаційними системами.

Вимоги до функціонала:

- Інтернет-крамниця
    - Навігація по веб вітрині крамниці.
    - Пошук товарів за критеріями.
    - Реєстрація покупців.
    - Формування кошика покупця.
    - Оформлення замовлень.
    - Зворотний зв'язок та відгуки.
- Управління контентом інтернет-крамниці
    - Навігація, перегляд, створення, редагування структури та контенту.
    - Керування доступом та розподіл дозволів користувачів.

    - Створення вітрини товарів.
    - Управління даними користувачів.
    - Оформлення замовлень.
    - Модерація коментарів.
- Реєстрація та авторизація користувачів.
    - Користувач ПЗ має змогу зареєструватись чи увійти у систему як за допомогою авторизаційних даних (email та пароль), так і за допомогою облікових записів популярних соціальних мереж (Google, тощо).
    - Користувач ПЗ має змогу увімкнути та вимкнути функцію двофакторної автентифікації.
- Особистий кабінет покупця
    - Перегляд історії замовлень.
    - Кошик покупця.
    - Реквізити для платежів та доставки.
- Взаємодія з інтернет додатками
    - Web-API моделі та запити для управління аккаунтом користувача.
    - Web-API моделі та запити для одержання, створення та редагування інформації, що формує профіль покупця.
    - Web-API моделі та запити для вибору товару в корзину та оформлення замовлення користувачем.
    - Web-API моделі та запити для створення питань та відгуків, а також для їх оцінок (лайків/дізлайків).

- Web-API моделі та запити для пошуку товарів з необхідними фільтрацією та сортуванням.
- Web-API моделі та запити для отримання категорій товарів, банерів, тощо.

Архітектура:

- Сервер бази даних — серверне програмне забезпечення, що забезпечує збереження та обробку поточних даних.
- Сервіс Web-API – серверне програмне забезпечення, що надає програмний інтерфейс для взаємодії (REST-API) з іншим програмним забезпеченням.
- Веб додаток керування контентом — доступний з веб-простору сайт, що забезпечує візуальний інтерфейс для менеджерів системи.
- Вебкрамниця — доступний з веб-простору сайт, що дозволяє взаємодіяти з інтернет-магазином покупцям.

Технології:

- Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології та методології створення:
  - Бази даних – PostgreSQL, Redis.
  - Сервіс Web-API – REST-API, ASP.NET Core, C#.
  - Веб додаток керування контентом – MVC, ASP.NET Core, Razor Views, C#.
  - Веб додаток для покупця – Node JS, React, Redux.

- При розгортанні програмного забезпечення рекомендується задіяти хмарні технології

- При розробці програмного забезпечення необхідно задіяти зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контролю версій (вибір конкретної системи узгоджується групою проєкт).

- Вихідний код повинен оформлятися відповідно до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.

РОЗДІЛ 1

ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Відповідно до технічного завдання (ТЗ) система була спроєктована за клієнт-серверною розподіленою архітектурою. У складі системи було умовно відокремлено наступні архітектурні частини:

- «Design»,
- «Backend»,
- «DevOps»,
- «Frontend».

Design - частина яка відповідає за створення та розробки вигляду, інтерфейсу та користувацького досвіду (UI/UX). “Дизайн” включає розробку та організацію вигляду сторінок та компонентів, що складають frontend , вибір кольорової схеми, типографіки та графічних елементів, створення інтуїтивно зрозумілої навігації, анімації та взаємодії з користувачем.

Частина дизайну розміщена за посиланням:
https://www.figma.com/file/PETLgTkhYxAYbqXE2jHX30/Untitled-(Copy)?type=design&node-id=0-1&mode=design&t=hPSPT9oSrbfJzsJy-0

Backend - відповідає за обробку запитів користувачів, роботу з базою даних, реалізацію функцій покупки товарів, обробку платежів та інші бізнес-логіки, що стосуються операцій електронної комерції. Вона розроблена мовою програмування C#, за допомогою фреймворку ASP.NET Core 6. Для надійного зберігання персональних було використано СУБД PostgreSQL.

Backend частина розміщена за посиланням:
https://github.com/storedev2023

Frontend - містити розробку та реалізацію веб-сторінок, взаємодію з користувачем, навігацію, відображення товарів, розташування категорій, пошук та фільтрацію товарів, оформлення замовлень та багато іншого. Вона забезпечує зручний та привабливий інтерфейс, де користувачі можуть шукати товари, додавати їх до кошика, здійснювати покупки, переглядати інформацію про товари, редагувати персональні дані тощо.

Frontend частина розміщена за посиланням:
https://github.com/storedev2023/Comfy.Frontend

DevOps - відіграє важливу роль, забезпечуючи постійну автоматизацію процесу розгортання веб-сайту на хмарному сервері AWS EC2. Ми використовуємо Docker для упаковки та розгортання додатка, а для автоматизації використовується Jenkins. У межах методології «DevOps» також враховується проксування, що забезпечує доступ клієнтів до сервера. Для цього використовуємо nginx, який забезпечує безпеку та ефективність зв'язку з сервером. Для моніторингу системи використовується сервіс Prometheus. Він дозволяє відстежувати споживання ресурсів, аналізувати дані та вчасно виявляти будь-які проблеми або незвичайні відхилення у роботі системи. Завдяки цьому ми можемо ефективно керувати та підтримувати нашу інфраструктуру.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ FRONTEND

Frontend частина написана з використанням наступних технологій:

- Середовище розробки Visual Studio Code.
- Створення структури веб-сторінки за допомогою HTML.
- Задання вигляду та стилізації елементів веб-сторінок за допомогою CSS та препроцесором Sass.
- Мова програмування JavaScript.
- Фреймворк ReactJS.
- Додаткові бібліотеки (компоненти) до фреймворку:
    - Redux (Redux Toolkit)
    - Redux-persist
    - React-bootstrap
    - React-router-dom
    - React-slick
    - React-transition-group
    - Pure-react-carousel
    - Axios

У складі клієнтської частини (Frontend) передбачено та реалізовано наступні функції:

- Навігація по сторінках веб-додатку
- Пошук. До нього входить: пошукова система, історія пошукових запитів, короткий перегляд товарів які збігаються із запитами користувача.
- Кошик з переліком усіх доданих до нього товарів та загальною сумою усіх товарів.
- Список переглянутих товарів.
- Авторизація користувача, реєстрація нових користувачів.
- Система коментарів та відгуків.
- Фільтрація товарів.

Початок роботи з інтернет-крамницею починається з головної сторінки де користувач може використати для зручного пересування по веб-додатку меню навігації, або пошук, якщо він шукає конкретні для нього товари, або одразу відвідати сторінку товару який знаходиться у вітрині на головній сторінці.

Нижче наведена навігація по сторінках веб-додатку:

```
<BrowserRouter>
    <Header />
    <Routes>
        <Route path="/" element={<Home />} />
        <Route path='/product/:id/' element={<Product />}  >
            <Route index element={<Info />} />
            <Route path="characteristics" element={<Characteristics />} />
            <Route path="reviews" element={<Reviews />} />
            <Route path="questions" element={<Questions />} />
        </Route>
        <Route path='/profile/:id/' element={<UserPage />} >
            <Route index element={<UserData />} />
            <Route path="orders" element={<UserOrders />} />
            <Route path="wishlist" element={<UserWishlist />} />
            <Route path="review" element={<UserReviews />} />
        </Route>
        <Route path="/search/:value/" element={<SearchPage />} />
        <Route path="/categories/:name/" element={<SubcategoriesPage />} />
        <Route path="/categories/:name/:subName/" element={<ProductsCategoriesPage />} />
        <Route path="/compare/:id" element={<Compare />} />
        <Route path="/order" element={<Order />} />
        <Route path='/404' element={<EmptyPage />} />
        <Route path='*' element={<EmptyPage />} />
    </Routes>
    <Footer />
</BrowserRouter>
```

Основною частиною любої інтернет-крамниці є зручний пошук, який дозволяє знайти саме ті товари, які необхідні користувачу. Для найзручнішого використання пошуку було реалізовано:

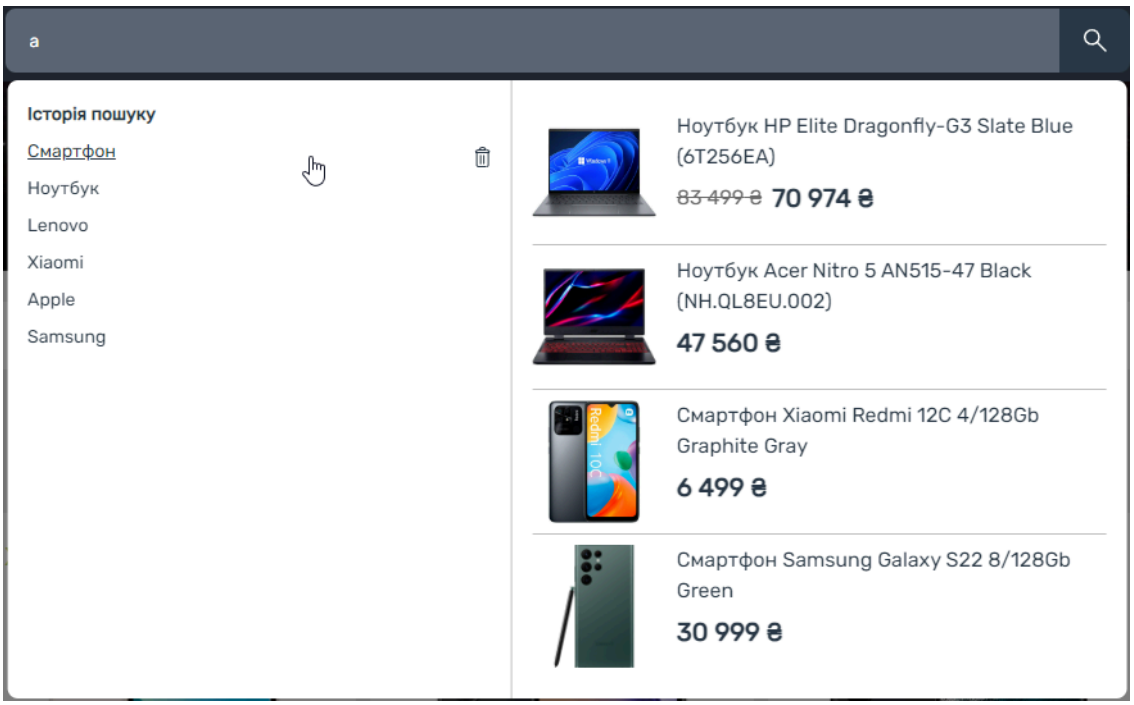
- Історію пошуку, яка має змогу зберігатися певний час та ручне видалення певної історії зі списку.

```
setHistoryToSearchList: (state,action) => {
    const index = state.searchHistory.findIndex(product => product == action.payload)
    if(index === -1){

        if (state.searchHistory.length >= 10) {
            state.searchHistory.pop()
        }
        state.searchHistory.unshift(action.payload)
    }
    else{

        state.searchHistory.splice(index,1)
        state.searchHistory.unshift(action.payload)
    }
},
deleteHistoryInSearchList: (state,action) => {
    console.log(action.payload)
    state.searchHistory = state.searchHistory.filter(history => history !== action.payload)
}
```

- Перегляд короткої інформації товарів, які збігаються із запитами користувача.



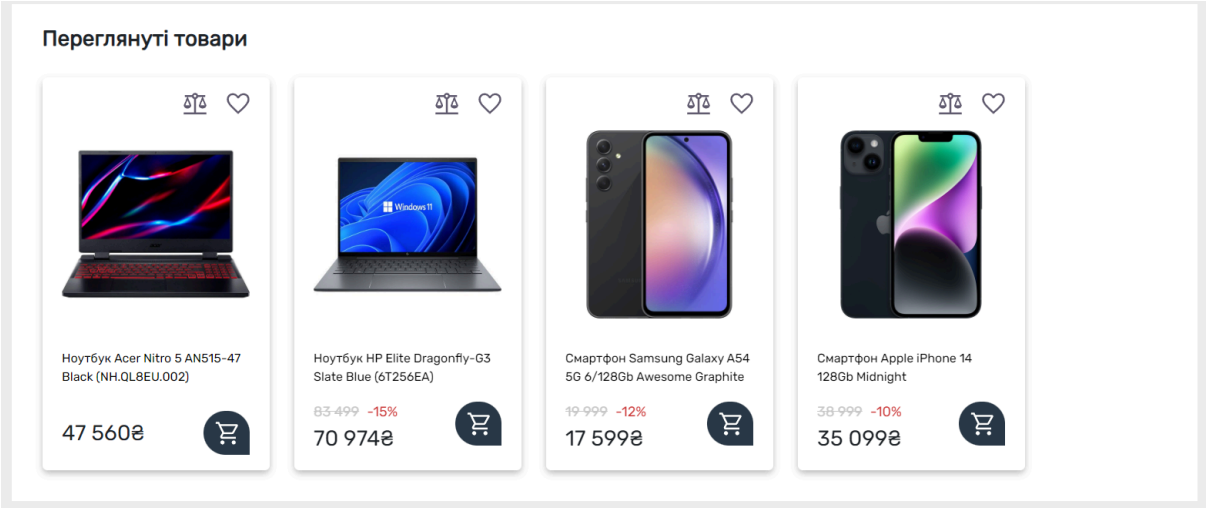
Ще однією важливою частиною є кошик куди користувач має змогу додати та видаляти товари. Важливо, щоб кошик зберігав товари після перезавантаження будь-якої сторінки веб-додатку для цього було вирішено використовувати Redux. Redux - це популярна бібліотека яка дозволяє гнучко взаємодіяти з даними.

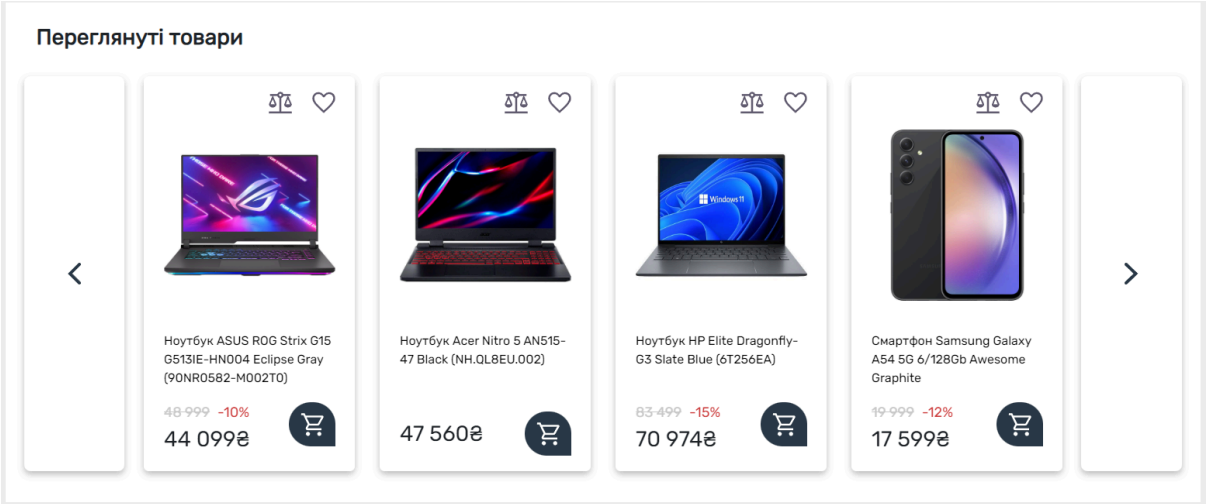
```
import { createSlice } from "@reduxjs/toolkit"

const cartSlice = createSlice({
    name: 'cart',
    initialState: {
        itemsInCart: []
    },
    reducers:{
        setItemInCart: (state,action) => {
            state.itemsInCart.push(action.payload)
        },
        deleteItemFromCart: (state,action) => {
            state.itemsInCart = state.itemsInCart.filter(product => product.id !== action.payload)
        }
    }
});

export const { setItemInCart, deleteItemFromCart } = cartSlice.actions;
export default cartSlice.reducer;
```

Список переглянутих товарів це одна з корисних рис нашої інтернет-крамниці. Завдяки цьому списку користувач має змогу повернутись на раніше переглянуті сторінки товарів. Як і кошик список повинен зберігати інформацію про переглянуті раніше товари та надавати змогу перейти до них, або додати у кошик. Зберігання функціонує по принципу кошика, навіть якщо сторінку було перезавантажено товари залишаться у списку. Максимум можна зберігатися до 10 товарів. Якщо переглянута мала кількість товарів (від 1 до 4) то список має вигляд звичайного блоку, але якщо товарів вже більше (від 5 до 10) то звичайний блок перетворюється на зручний слайдер. Приклади слайдера:





РОЗДІЛ 3

РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Після компіляції програмного коду об'єднуються серверна та клієнтська частини.

Початок роботи з інтернет-магазином починається з головної сторінки, де користувач може використати навігацію, щоб перейти на сторінку товарів певної категорії, або використати пошук для знаходження необхідних йому товарів, або переглянути вітрину товарів.

Щоб оформити замовлення, користувач повинен додати хоча б один товар до кошика та бути зареєстрованим і авторизованим на сайті.

Після оформлення та підтвердження замовлення дані про нього передаються на серверну частину де додаються у базу даних. Користувач може перевірити своє замовлення на сторінці “Замовлення” у своєму профілі.

ВИСНОВКИ

Веб-додаток, який був розроблений і реалізований на момент написання цього документу, перебуває на ранній стадії розвитку і має потенціал для поліпшення та додаткового розширення. Є багато компонентів, які можна покращити та додати для поліпшення функціональності та користувацького досвіду.

Але вже реалізовані багато основних можливостей: навігація, пошук, фільтрація товарів, перегляд товару, додавання товару до кошика, оформлення заказу, перегляд сторінки користувача, тощо. Завдяки йому користувач має змогу вибрати один або більше товарів та зробити заказ.

Інтернет-крамниця складена за розподіленою клієнт-серверною архітектурою та складається з трьох частин, розміщених за адресами:

- Частина дизайну: https://www.figma.com/file/PETLgTkhYxAYbqXE2jHX30/Untitled-(Copy)?type=design&node-id=0-1&t=myj3NjbIQoztUXro-0
- Backend частина: https://github.com/storedev2023/Comfy.Backend
- Frontend частина: https://github.com/storedev2023/Comfy.Frontend

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

№ п.п.	Ресурс
1.	HTML (https://developer.mozilla.org/en-US/docs/Web/HTML)
2.	CSS (https://www.w3schools.com/css/default.asp)
3.	SASS (https://sass-lang.com/guide/)
4.	JavaScript (https://developer.mozilla.org/en-US/docs/Web/JavaScript)
5.	Visual Studio Code (https://code.visualstudio.com/docs)
6.	GitHub Desktop (https://desktop.github.com/)
7.	React-Redux (https://react-redux.js.org/)
8.	Redux Toolkit (https://redux-toolkit.js.org/)
9.	Redux-persist (https://www.npmjs.com/package/redux-persist)
10.	React-bootstrap (https://react-bootstrap.netlify.app/)
11.	React-router-dom (https://www.npmjs.com/package/react-router-dom)
12.	React-slick (https://react-slick.neostack.com/)
13.	React-transition-group (https://reactcommunity.org/react-transition-group/css-transition)
14.	Pure-react-carousel (https://www.npmjs.com/package/pure-react-carousel)
15.	Axios (https://www.npmjs.com/package/axios)

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

- Stackoverflow
- Hubr
- Medium
- MND Web Docs
- DEV
- Figma
- ChatGPT
- YouTube
- Google authentication documentation
- GitHub
- JWT documentation
- IANA JWT specification
- Metanit