**Summary Retail\_2024**

**1. Огляд проекту**

1.1 Мета проекту: Розробка системи управління товарообігом на складах.

1.2.Опис функціональності:

У проекті реалізовано функціонал системи управління товарообігом на складах. Зокрема, в системі враховані такі ключові аспекти:

* Облік товарів: Система забезпечить точний та актуальний облік кількості товарів на кожному з 6 складів.
* Ведення замовлень: Можливість створення та обробки замовлень для оптимального управління запасами.
* Видача товарів зі складів: Процес контрольованої видачі товарів, що дозволяє ефективно управляти їх розподілом.
* Звітність: Генерація детальних звітів щодо товарів, їх розташування та обсягів, що сприяє зручному аналізу та прийняттю управлінських рішень.
* Інтерактивний інтерфейс: Створення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, що полегшить взаємодію з системою.

1.3 Результати відповідають передбаченому раніше функціоналу.

Реалізація системи, яка забезпечує точний облік товарів на всіх 6 складах, покращує управління замовленнями та оптимізує процеси видачі товарів.

**2. Виконані задачі**

2.1 Кисельова Марія (PM+Developer):

- Управління проектом та командою, введення репозиторію GitHub, заведення\контроль виконання задач в Jira, проведення Zoom мітингів;

- Розробка Project Plan;

- Розробка шаблонів сторінок за допомогою html&css, редагування сss login/register форм;

- Summary report.

2.2 Мартинюк Андрій (Business Analyst):

- Створення початкової документації проекту (SRS).

2.3 Гудемчук Дарʼя (QA):

- Проведення тестувань для перевірки функціональності та відповідності вимогам;

- Розробка Test summary report.doc, Testing strategy.doc, Test results.pdf, Test Plan.doc.

2.4 Постульга Євген, Крив Микита (Developers):

- Реалізація функціоналу використовуючи Node Js, Javascript, REST API, HTML, CSS та sqlite3;

- Розміщення проекту в загальному доступі – Vercel;

- Документування та підготовка до випуску.

**3. Терміни виконання**

3.1 Заплановані раніше етапи проекту:

1. Аналіз вимог і планування (2 тижні).

2. Розробка інтерфейсу та базового функціоналу (6 тижнів).

3. Тестування та виправлення помилок (2 тижні).

4. Документування та підготовка до випуску (1 тиждень).

Дотримані з незначною похибкою. З моменту створення github репозиторію до здачі проекту пройшло близько 12 тижнів.

**4. Менеджмент проекту**

4.1 Jira:

- Всі задачі та етапи проекту відстежувались в Jira (<https://mariakyselova.atlassian.net/jira/software/projects/RET/boards/3>).

4.2 GitHub:

- Код проекту, зберігається на GitHub (https://github.com/KyselovaMaria/Retail\_2024).

**5. Project Stage**

Проект знаходиться в загальному доступі за посиланням:

<https://retail1-1-evgens-projects-580c01c7.vercel.app/> .

**6. Документація REST API серверного додатку**

**Серверний додаток API**

Цей додаток надає REST API для управління різними аспектами системи.

**Роути**

/order

GET /

Отримати список всіх замовлень.

GET /:id

Отримати конкретне замовлення за його ідентифікатором.

POST /

Створити нове замовлення з наданими даними в тілі запиту.

Приклад JSON-об'єкту для створення нового замовлення:

{

warehouseId: string;

shippingAddress: string;

}

POST /confirm/:id

Підтвердження замовлення, необхідно мати роль менеджер чи адмін.

/warehouse

GET /

Отримати список всіх складів.

GET /:id

Отримати конкретний склад за його ідентифікатором.

GET /withProduct/:id

Отримати склади з певним товаром в них.

POST /

Створити новий склад з наданими даними в тілі запиту.

Приклад JSON-об'єкту для створення нового складу:

{

name: string;

address: string;

}

Приклад HTTP-header:

{

Authorization: Bearer {token}

}

/product

GET /

Отримати список всіх продуктів.

GET /:id

Отримати конкретний продукт за його ідентифікатором.

POST /

Створити новий продукт з наданими даними в тілі запиту.

Приклад JSON-об'єкту для створення нового продукта:

{

name: string;

price: number;

minimumStock: number;

}

Приклад HTTP-header:

{

Authorization: Bearer {token}

}

/stock

POST /addToStock

Додати кількість товару до стоку.

Приклад JSON-об'єкту для створення нового продукта:

{

id: string;

amount: number;

}

Приклад HTTP-header:

{

Authorization: Bearer {token}

}

POST /transferStock

Перенести товар з одного стоку на інший.

Приклад JSON-об'єкту для створення нового продукта:

{

stockToId: string;

stockFromId: string;

amount: number;

}

Приклад HTTP-header:

{

Authorization: Bearer {token}

}

POST /

Створити новий продукт з наданими даними в тілі запиту.

Приклад JSON-об'єкту для створення нового продукта:

{

productId: string;

warehouseId: string;

amount: number;

}

Приклад HTTP-header:

{

Authorization: Bearer {token}

}

/orderListing

POST /

Додати кількість товару до стоку.

Приклад JSON-об'єкту для створення нового продукта:

{

orderId: string;

productId: string;

amount: number;

}

/auth

POST /login

Отримати новий токен через логін.

Приклад JSON-об'єкту:

{

username: string

password: string

}

POST /signup

Створити нового юзера з роллю. roleId: 1 - admin, 2 - manager, 3 customer

Приклад JSON-об'єкту для створення нового юзера:

{

username: string,

email: string,

password: string,

roleId: number,

}

/user

GET /

Отримати дані про користувача з токена, для перевірки авторизації в основному.

/roles

GET /

Отримати існуючі ролі.

**Admin**

Може все, тільки адмін може створювати нові продукти, склади, редагувати та видаляти їх

**Manager**

Може те, що може кастомер. Також може редагувати, додавати і т.п. stocks

**Customer**

Може все окрім того, що унікально для менеджерів і адмінів

**Розгортання серверу:**

**Клонування з репозиторію**

Спочатку треба клонувати віддалений репозиторій за допомогою команди git clone <посилання на репозиторій>.

**Конфігурація проекту**

Треба налаштувати файл змінних середовища .env таким чином, щоб там були поля RETAIL\_DB\_CONN\_URI та SECRET\_AUTH\_KEY для з'єднання з базою даних та створення токенів відповідно.

**Встановлення Node.js та бібліотек**

Спочатку треба встановити Node.js v18.16.1 та встановити pnpm через команду npm install -g pnpm. Після цього встановлюємо необхідні бібліотеки за допомогою команди pnpm install.

**Запуск серверу**

Запускаємо сервер командою pnpm run start, яка білдить і запускає застосунок на порті 3000. Таким чином ми маємо працюючий сервер за адресою localhost:3000.

Для локального запуску проекту вистачить завантажити **VSCode .**