Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота № 2

Київ-2025

Лабораторна робота № 2 Розробка структури застосунку

22.10.2025

- Розділити додаток на основні компоненти/модулі, описати їх взаємодію відомими вам методами (наприклад діаграма компонентів ПЗ)
- Описати дані та їх зв’язки (наприклад ER діаграма)
- Описати як дані оновлюються\змінюються\агрегуються на основі ключових сценаріїв, які буде виконувати додаток

Основні рівні (Layers):

Presentation Layer (Frontend):

Технології: React, Vite, TypeScript, React Router.

Роль: Відповідає за інтерфейс користувача, відображення товарів, форм реєстрації та кошика. Він не зберігає дані постійно, а отримує їх через HTTP-запити.

Application Layer (Backend):

Технології: NestJS, Node.js, TypeScript.

Роль: Обробляє бізнес-логіку, валідує дані, керує авторизацією та комунікує з базою даних.

Data Layer (Database):

Технології: MongoDB Atlas (Cloud).

Роль: Фізичне зберігання даних (документів) у форматі BSON.

Розподіл на Модулі (NestJS Modules):

Система розділена на 4 основні модулі, які взаємодіють між собою:

Модуль	Складові (Controller / Service / Schema)	Призначення
AppModule	root	Головний модуль, який збирає всі інші модулі та налаштовує з’єднання з базою даних (MongooseModule.forRoot).
AuthModule	AuthController, AuthService, UserSchema	Відповідає за реєстрацію та логін. Працює безпосередньо з колекцією **users**.
ProductsModule	ProductsController, ProductsService, ProductSchema	Відповідає за каталог книг. Має механізм **seeding** (автозаповнення), якщо база порожня. Експортує ProductsService, щоб **CartModule** міг отримувати назви та ціни книг.
CartModule	CartController, CartService, CartSchema	Керує кошиком. **Залежить від ProductsModule**. При запиті вмісту кошика бере ID з CartSchema, а деталі товарів (картинка, назва, ціна) отримує через ProductsService.

Структура Даних (ER Diagram / Schemas)

Оскільки ми використовуємо MongoDB (NoSQL), ми оперуємо Колекціями та Документами, а не таблицями. Проте зв'язки існують на рівні логіки додатка.

Ось опис сутностей:

А. Колекція users (Користувачі)

Зберігає дані для авторизації.

_id: ObjectId (Auto)

name: String

email: String (Unique)

password: String

B. Колекція products (Товари)

Зберігає каталог книг.

_id: ObjectId (Auto)

productID: Number (наш власний ID для зручності)

productName: String

productPrice: Number

produtDescription: String

frontImg: String (URL)

productReviews: String

C. Колекція cartitems (Кошик)

Зберігає стан кошика. Це проміжна сутність.

_id: ObjectId (Auto)

productID: Number (Foreign Key - посилання на products.productID)

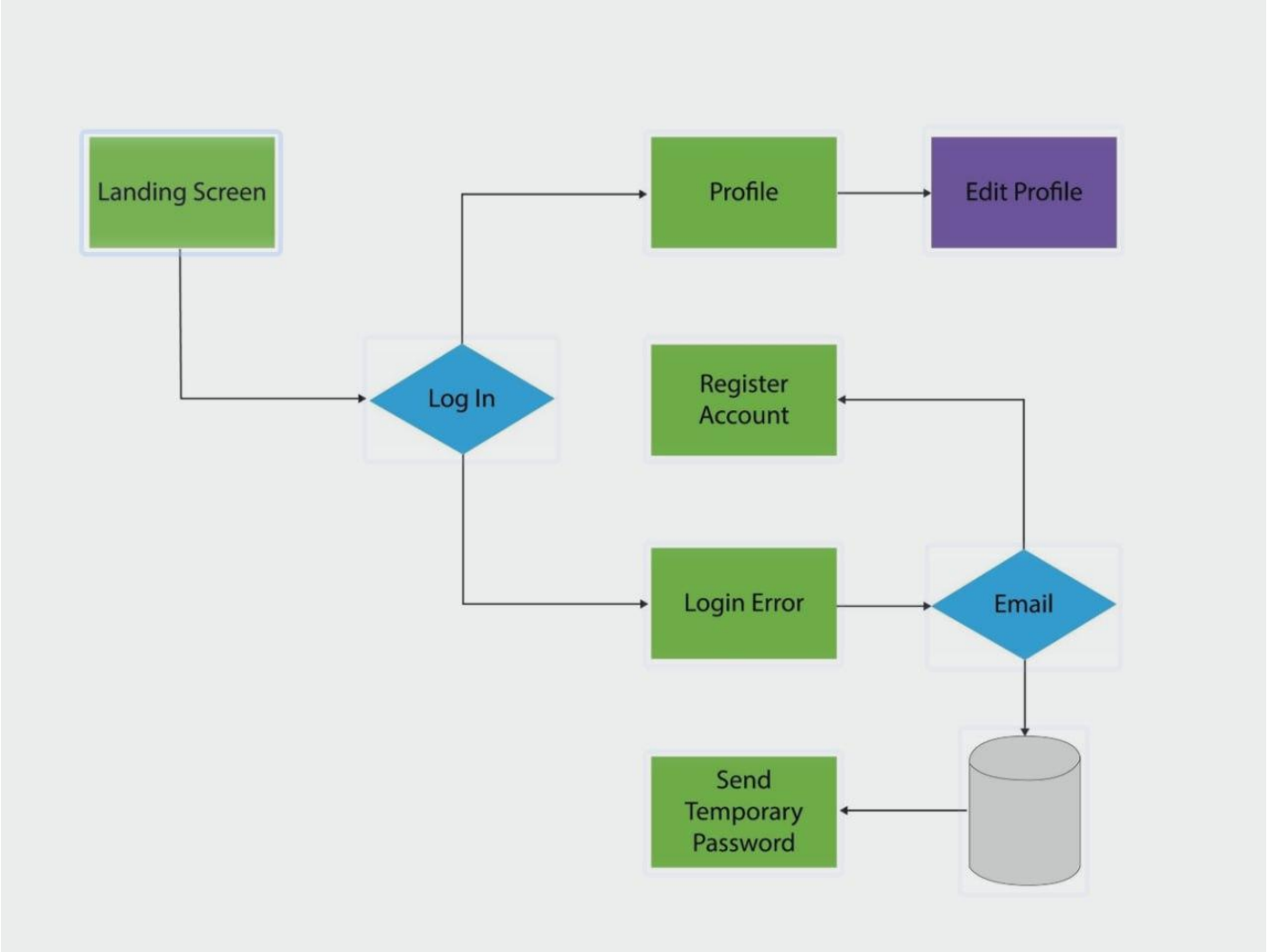
quantity: Number

Зв'язок: CartItem -> Product (Зв'язок "Один до Одного" в контексті запису кошика, але логічно це "Багато товарів у кошику").

3. Сценарії оновлення та агрегації даних (Data Flow)

Опишемо, що відбувається з даними в ключових сценаріях, які ми реалізували.

Сценарій 1: Реєстрація нового користувача



Input: Користувач вводить email, name, password на React-формі.

Process:

Frontend відправляє POST /api/register.

Backend (AuthService) робить запит у БД: findOne({ email }).

Перевірка: Якщо користувач існує -> Помилка 409.

Дія: Якщо ні -> Створює новий об'єкт User і викликає .save().

Data Update: У колекції users з'являється новий документ.

Сценарій 2: Перегляд Кошика (Агрегація даних)

Це найскладніший сценарій, де відбувається об'єднання даних з двох колекцій.

Input: Користувач відкриває сторінку /cart.

Process:

Frontend відправляє GET /cart.

Backend (CartService) робить запит у БД: cartModel.find(). Отримує список ID та кількості: [{ productID: 1, quantity: 2 }, { productID: 2, quantity: 1 }].

Агрегація (Aggregation): Сервіс проходить циклом по цьому списку. Для кожного елемента він викликає productsService.findOne(productID).

Сервіс "склеює" об'єкти: бере quantity з кошика і додає до нього name, price, img з продукта.

Output: Frontend отримує повний JSON з усіма даними для відображення красивої таблиці.

Сценарій 3: Додавання товару в кошик (Upsert - Update or Insert)

Input: Користувач натискає "Add to Cart" (Product ID: 1).

Process:

Frontend відправляє POST /cart з { productID: 1, quantity: 1 }.

Backend перевіряє, чи існує такий продукт узагалі.

Backend перевіряє, чи вже є цей товар у кошику (cartModel.findOne).

Логіка оновлення:

ЯКЩО Є: Бере старе значення quantity і додає до нього нове. Виконує save() (Update).

ЯКЩО НЕМАЄ: Створює новий запис (Insert).

Data Update: Дані в колекції cartitems змінюються або додаються.

Що варто додати в майбутньому (для звіту):

Зв'язок User <-> Cart: Зараз кошик "спільний" для всіх. Варто додати поле userId в CartSchema, щоб у кожного користувача був свій власний кошик.

Зв'язок User <-> Orders: Створити колекцію orders для збереження історії покупок після натискання "Place Order".