**Báo Cáo Assignment: Phát Triển Ứng Dụng Web Bán Bánh Ngọt Trực Tuyến**

**GitHub**

* **Frontend:** [**https://github.com/trung2605/bakery\_assginment\_fe.git**](https://github.com/trung2605/bakery_assginment_fe.git)
* **Backend:** [**https://github.com/trung2605/bakery\_assginment\_be**](https://github.com/trung2605/bakery_assginment_be)

## I. TRANG BÌA

**Đề Tài Assignment:** Phát Triển Ứng Dụng Web Bán Bánh Ngọt Trực Tuyến

**Sinh Viên/Nhóm Sinh Viên:**

* Lê Trí Trung DE190286
* Mai Nguyễn Tiến Đạt DE190698

## II. MỤC LỤC

[**I. TRANG BÌA**](#_qelo2t1nwzr) **1**

[**II. MỤC LỤC**](#_jvxea2wkyokh) **1**

[**III. LỜI MỞ ĐẦU**](#_a4sxvenpcf5d) **3**

[1.1. Giới thiệu chung về đề tài](#_2ht7zaf7cmv0) 3

[1.2. Mục tiêu của dự án](#_720wewwzogus) 3

[1.3. Phạm vi dự án](#_ib0j62djmm09) 3

[1.4. Cấu trúc báo cáo](#_b82vok86gwkt) 4

[**IV. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**](#_yphf0vmpuaff) **4**

[2.1. Phân tích yêu cầu](#_2mpkj2bhja1o) 4

[2.2. Thiết kế Cơ sở dữ liệu (Database Design)](#_4l712vm832ut) 6

[2.3. Thiết kế kiến trúc hệ thống (System Architecture Design)](#_xux3j7c8awvf) 8

[**V. TRIỂN KHAI VÀ PHÁT TRIỂN**](#_qsf7imc9pgn7) **9**

[3.1. Phát triển Backend (Spring Boot)](#_ua76sg26xx7u) 9

[3.1.1. Cấu trúc thư mục](#_z511gpvnhhx) 9

[3.2. Phát triển Frontend (ReactJS)](#_rkypm5lrwutj) 13

[3.2.1. Cấu trúc thư mục](#_e3rgbt5k8do6) 13

[3.2.2. Các trang (Pages)](#_5z7msw7m2qj6) 13

[3.2.3. Các Component (Components)](#_bs0dnja0adaa) 14

[**VI. KẾT QUẢ CHẠY VÀ MÀN HÌNH**](#_2p6q58ixv9tm) **16**

## 

## **III. LỜI MỞ ĐẦU**

### 1.1. Giới thiệu chung về đề tài

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, việc mua sắm trực tuyến đã trở thành một phần không thể thiếu trong cuộc sống hiện đại. Ngành thực phẩm, đặc biệt là bánh ngọt, cũng không nằm ngoài xu hướng này. Dự án "Ứng Dụng Web Bán Bánh Ngọt Trực Tuyến" được phát triển nhằm cung cấp một nền tảng thương mại điện tử đơn giản, thân thiện, giúp các cửa hàng bánh dễ dàng tiếp cận khách hàng tiềm năng và tối ưu hóa quy trình bán hàng. Ứng dụng này tập trung vào việc hiển thị sản phẩm, quản lý giỏ hàng và cung cấp trải nghiệm mua sắm mượt mà cho người dùng.

### 1.2. Mục tiêu của dự án

Mục tiêu chính của dự án bao gồm:

* Xây dựng một giao diện người dùng (Frontend) trực quan, dễ sử dụng, cho phép khách hàng duyệt sản phẩm, xem chi tiết và thêm vào giỏ hàng.
* Phát triển một hệ thống backend mạnh mẽ (Spring Boot) để quản lý dữ liệu sản phẩm, giỏ hàng, và xử lý các yêu cầu từ frontend.
* Triển khai các tính năng tìm kiếm, lọc và phân trang hiệu quả để nâng cao trải nghiệm khám phá sản phẩm của người dùng.
* Đảm bảo tính ổn định và khả năng mở rộng của ứng dụng.
* Áp dụng các kỹ thuật phát triển phần mềm hiện đại như kiến trúc RESTful API, quản lý trạng thái với React Context, và kiểm tra kiểu dữ liệu với PropTypes.

### 1.3. Phạm vi dự án

Dự án này tập trung vào các chức năng cốt lõi sau:

* **Quản lý sản phẩm:** Hiển thị danh sách sản phẩm, xem chi tiết sản phẩm.
* **Tìm kiếm và lọc:** Cho phép người dùng tìm kiếm sản phẩm theo tên, lọc theo danh mục và khoảng giá, cùng với tính năng phân trang.
* **Giỏ hàng:** Khách hàng có thể thêm sản phẩm vào giỏ hàng, cập nhật số lượng và xóa sản phẩm khỏi giỏ.
* **Đăng ký/Đăng nhập:** Hỗ trợ người dùng đăng ký tài khoản mới và đăng nhập để quản lý giỏ hàng cá nhân.
* **Trang chủ:** Hiển thị các sản phẩm nổi bật, sản phẩm mới nhất và video giới thiệu cửa hàng.

Các tính năng nâng cao như quản lý đơn hàng, thanh toán trực tuyến, đánh giá sản phẩm, hoặc trang quản trị chi tiết cho người bán nằm ngoài phạm vi của assignment này nhưng có thể là hướng phát triển trong tương lai.

### 1.4. Cấu trúc báo cáo

Báo cáo này được tổ chức thành các phần chính sau:

* **Phân Tích và Thiết Kế Hệ Thống:** Mô tả các yêu cầu, thiết kế cơ sở dữ liệu và kiến trúc hệ thống.
* **Triển Khai và Phát Triển:** Đi sâu vào chi tiết kỹ thuật của việc phát triển backend và frontend.
* **Kết Quả Chạy và Màn Hình:** Trình bày hình ảnh và hướng dẫn chạy ứng dụng.
* **Đánh giá và Hướng Phát Triển:** Đánh giá những gì đã đạt được và các kế hoạch phát triển tiếp theo.
* **Kết Luận và Tài Liệu Tham Khảo.**

## **IV. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**

### 2.1. Phân tích yêu cầu

**2.1.1. Yêu cầu chức năng (Functional Requirements)**

* **Quản lý Sản phẩm:**
  + Hiển thị danh sách tất cả sản phẩm.
  + Xem chi tiết thông tin của từng sản phẩm (tên, mô tả, giá, số lượng tồn kho, hình ảnh, hạn sử dụng).
  + Lọc sản phẩm theo danh mục.
  + Lọc sản phẩm theo khoảng giá.
  + Tìm kiếm sản phẩm theo tên (hỗ trợ phân trang và sắp xếp).
  + Phân trang cho danh sách sản phẩm.
  + Sắp xếp sản phẩm theo tên, giá (tăng/giảm dần).
* **Quản lý Giỏ hàng:**
  + Thêm sản phẩm vào giỏ hàng.
  + Cập nhật số lượng sản phẩm trong giỏ hàng.
  + Xóa sản phẩm khỏi giỏ hàng.
  + Xem tổng số tiền trong giỏ hàng.
  + Xóa toàn bộ giỏ hàng.
* **Quản lý Tài khoản người dùng:**
  + Đăng ký tài khoản mới (kiểm tra trùng lặp email, số điện thoại, độ dài mật khẩu).
  + Đăng nhập tài khoản (sử dụng email hoặc số điện thoại).
  + Liên kết giỏ hàng với tài khoản người dùng sau khi đăng nhập.
* **Trang chủ:**
  + Hiển thị các sản phẩm nổi bật.
  + Hiển thị các sản phẩm mới nhất.
  + Hiển thị video giới thiệu cửa hàng.

**2.1.2. Yêu cầu phi chức năng (Non-Functional Requirements)**

* **Hiệu suất:**
  + Thời gian tải trang nhanh, đặc biệt là trang danh sách sản phẩm và trang chủ.
  + Phản hồi API trong thời gian chấp nhận được (dưới 500ms cho các tác vụ thông thường).
* **Bảo mật:**
  + Sử dụng HTTPS (khuyến nghị khi triển khai).
  + Mật khẩu người dùng được mã hóa (hashing) trước khi lưu vào cơ sở dữ liệu.
  + Bảo vệ các endpoint API nhạy cảm (ví dụ: thêm/sửa/xóa giỏ hàng) bằng xác thực (authentication) và ủy quyền (authorization) thông qua Spring Security.
  + Xử lý lỗi CSRF (nếu bật).
* **Khả năng sử dụng (Usability):**
  + Giao diện người dùng trực quan, dễ điều hướng.
  + Thiết kế đáp ứng (Responsive Design) để tương thích trên nhiều thiết bị (desktop, tablet, mobile).
  + Thông báo lỗi rõ ràng, thân thiện với người dùng trên frontend.
* **Khả năng bảo trì:**
  + Mã nguồn được tổ chức rõ ràng, dễ đọc và dễ hiểu.
  + Sử dụng các design pattern phù hợp (ví dụ: MVC, Service Layer).
  + Tuân thủ các quy ước đặt tên và coding convention.
* **Khả năng mở rộng (Scalability):**
  + Kiến trúc module hóa, cho phép dễ dàng thêm các tính năng mới.
  + Sử dụng phân trang phía server để xử lý lượng dữ liệu lớn.
* **Độ tin cậy:**
  + Xử lý lỗi một cách duyên dáng (graceful error handling) để ứng dụng không bị crash khi có sự cố.

### 2.2. Thiết kế Cơ sở dữ liệu (Database Design)

Ứng dụng sử dụng cơ sở dữ liệu quan hệ với các bảng chính sau:

* **Bảng users:** Lưu trữ thông tin người dùng.
  + user\_id (VARCHAR(255), PK)
  + first\_name (VARCHAR(255))
  + last\_name (VARCHAR(255))
  + email (VARCHAR(255), UNIQUE, NOT NULL)
  + phone (VARCHAR(20), UNIQUE, NOT NULL)
  + password\_hash (VARCHAR(255), NOT NULL)
  + role (VARCHAR(50), DEFAULT 'USER')
  + created\_at (DATETIME, NOT NULL)
  + last\_modified\_at (DATETIME)
* **Bảng products:** Lưu trữ thông tin sản phẩm.
  + product\_id (VARCHAR(255), PK)
  + name (VARCHAR(255), NOT NULL)
  + description (TEXT)
  + price (DOUBLE, NOT NULL)
  + stock\_quantity (INT, NOT NULL)
  + category (VARCHAR(100))
  + image\_url (VARCHAR(255))
  + expiration\_date (VARCHAR(50))
  + created\_at (DATETIME, NOT NULL)
  + last\_modified\_at (DATETIME)
* **Bảng carts:** Đại diện cho giỏ hàng của người dùng.
  + cart\_id (VARCHAR(255), PK)
  + user\_id (VARCHAR(255), FK to users.user\_id, UNIQUE, có thể NULL nếu là giỏ hàng khách)
  + total\_amount (DOUBLE, DEFAULT 0.0)
  + created\_at (DATETIME, NOT NULL)
  + last\_modified\_at (DATETIME)
* **Bảng cart\_items:** Chi tiết các sản phẩm trong mỗi giỏ hàng.
  + cart\_item\_id (VARCHAR(255), PK)
  + cart\_id (VARCHAR(255), FK to carts.cart\_id, NOT NULL)
  + product\_id (VARCHAR(255), FK to products.product\_id, NOT NULL)
  + quantity (INT, NOT NULL)
  + price\_at\_addition (DOUBLE, NOT NULL) (Giá sản phẩm tại thời điểm thêm vào giỏ)
  + created\_at (DATETIME, NOT NULL)
  + last\_modified\_at (DATETIME)

**Mô hình Thực thể Kết hợp (ERD - Entity-Relationship Diagram):**

* **User** (1) -- (1) **Cart**: Mỗi người dùng có một giỏ hàng duy nhất.
* **Cart** (1) -- (N) **CartItem**: Mỗi giỏ hàng có thể chứa nhiều mặt hàng.
* **Product** (1) -- (N) **CartItem**: Mỗi mặt hàng trong giỏ hàng tham chiếu đến một sản phẩm cụ thể.

### 2.3. Thiết kế kiến trúc hệ thống (System Architecture Design)

Ứng dụng được thiết kế theo kiến trúc Client-Server (hoặc 3-tier architecture) bao gồm:

* **Presentation Layer (Frontend):**
  + Được xây dựng bằng **ReactJS**.
  + Chịu trách nhiệm về giao diện người dùng, tương tác với người dùng và hiển thị dữ liệu.
  + Gửi các yêu cầu HTTP (API calls) đến Backend và xử lý phản hồi.
* **Business Logic Layer (Backend):**
  + Được xây dựng bằng **Spring Boot (Java)**.
  + Chứa các lớp Controller, Service và Repository.
  + **Controller:** Xử lý các yêu cầu HTTP từ Frontend, ánh xạ chúng tới các phương thức nghiệp vụ.
  + **Service:** Chứa logic nghiệp vụ chính của ứng dụng (ví dụ: logic thêm sản phẩm vào giỏ, lọc sản phẩm, mã hóa mật khẩu).
  + **Repository:** Tương tác với cơ sở dữ liệu thông qua Spring Data JPA.
* **Data Access Layer (Database):**
  + Sử dụng hệ quản trị cơ sở dữ liệu **MySQL** (hoặc PostgreSQL/H2 Database cho phát triển).
  + Lưu trữ tất cả dữ liệu của ứng dụng (sản phẩm, người dùng, giỏ hàng, v.v.).

**Sơ đồ Kiến trúc Hệ thống:**

**Công nghệ sử dụng:**

* **Frontend:**
  + **ReactJS:** Thư viện JavaScript để xây dựng giao diện người dùng.
  + **JavaScript (ES6+):** Ngôn ngữ lập trình chính.
  + **HTML5 & CSS3:** Cấu trúc và tạo kiểu giao diện.
  + **Axios:** Thư viện HTTP client để gửi yêu cầu API.
  + **React Router DOM:** Quản lý định tuyến trong ứng dụng Single Page Application (SPA).
  + **Context API:** Quản lý trạng thái toàn cục (ví dụ: thông tin người dùng, trạng thái giỏ hàng).
  + **PropTypes:** Kiểm tra kiểu dữ liệu của props để tăng cường tính vững chắc của component.
  + **Debounce:** Kỹ thuật tối ưu hóa hiệu suất cho các input tìm kiếm.
* **Backend:**
  + **Spring Boot:** Framework Java để xây dựng ứng dụng web RESTful API nhanh chóng.
  + **Java:** Ngôn ngữ lập trình chính.
  + **Spring Data JPA:** Giúp tương tác với cơ sở dữ liệu dễ dàng hơn thông qua các Repository.
  + **Hibernate:** ORM (Object-Relational Mapping) được sử dụng bởi Spring Data JPA.
  + **Lombok:** Thư viện giúp giảm boilerplate code (getter, setter, constructor).
  + **Spring Security:** Cung cấp các tính năng xác thực và ủy quyền.
  + **Jackson:** Thư viện xử lý JSON (serialization/deserialization).
  + **MySQL Connector/J:** Driver để kết nối Java với MySQL.
* **Công cụ và Môi trường:**
  + **Maven/Gradle:** Công cụ quản lý dự án và build cho Java.
  + **Git/GitHub:** Hệ thống kiểm soát phiên bản.
  + **IDE:** IntelliJ IDEA / VS Code.

## **V. TRIỂN KHAI VÀ PHÁT TRIỂN**

### 3.1. Phát triển Backend (Spring Boot)

### 3.1.1. Cấu trúc thư mục

Dự án Backend tuân thủ cấu trúc thư mục tiêu chuẩn của Spring Boot:

src/main/java/com/example/bakery

├── BakeryApplication.java // Lớp khởi chạy ứng dụng

├── config // Cấu hình Spring Security, CORS

├── controller // Các REST API endpoints

├── dto // Data Transfer Objects (DTOs)

├── entity // Các lớp Entity ánh xạ với bảng DB

├── repository // Các interface tương tác với DB

└── service // Logic nghiệp vụ

**3.1.2. Mô hình (Model)**

Các lớp Entity được định nghĩa trong package com.example.bakery.model (hoặc entity tùy theo cách bạn tổ chức). Chúng ánh xạ các đối tượng Java với các bảng trong cơ sở dữ liệu.

* **Product.java**: Đại diện cho một sản phẩm.
* **User.java**: Đại diện cho một người dùng.
* **Cart.java**: Đại diện cho một giỏ hàng.
* **CartItem.java**: Đại diện cho một mặt hàng trong giỏ hàng.

Ví dụ về Cart.java với Auditing và xử lý toString():

// package com.example.bakery.model;

// import ...

@Entity

@Table(name = "carts")

@Getter @Setter

@ToString(exclude = {"cartItems"}) // Loại trừ cartItems để tránh StackOverflowError

@EntityListeners(AuditingEntityListener.class)

public class Cart {

@Id private String cartId;

@OneToOne(fetch = FetchType.LAZY) @JoinColumn(name = "user\_id") private User user;

@OneToMany(mappedBy = "cart", cascade = CascadeType.ALL, orphanRemoval = true) private Set<CartItem> cartItems = new HashSet<>();

@CreatedDate @Column(name = "created\_at", nullable = false, updatable = false) private LocalDateTime createdAt;

@LastModifiedDate @Column(name = "last\_modified\_at", nullable = false) private LocalDateTime lastModifiedAt;

// ... addCartItem, removeCartItem, equals, hashCode ...

}

Việc sử dụng @ToString(exclude = {"cartItems"}) là rất quan trọng để tránh lỗi StackOverflowError khi in đối tượng Cart do quan hệ vòng lặp với CartItem (và fetch = FetchType.LAZY tránh LazyInitializationException).

**3.1.3. Repository**

Các interface Repository (trong com.example.bakery.repository) mở rộng JpaRepository của Spring Data JPA, cung cấp các phương thức CRUD cơ bản mà không cần viết code. Ngoài ra, các phương thức truy vấn tùy chỉnh được định nghĩa để hỗ trợ lọc, tìm kiếm và phân trang:

* UserRepository: findByEmail, existsByEmail, existsByPhone.
* ProductRepository: findByCategory, findByNameContainingIgnoreCase, findByPriceBetween, và các kết hợp khác cho phân trang và tìm kiếm.

**3.1.4. Service**

Các lớp Service (trong com.example.bakery.service) chứa logic nghiệp vụ chính, tương tác với các Repository và xử lý dữ liệu trước khi gửi đến Controller hoặc lưu vào DB.

* **UserService**: Chứa logic đăng ký người dùng, mã hóa mật khẩu (hashPassword), kiểm tra sự tồn tại của email/số điện thoại.
* **ProductService**: Xử lý việc lấy sản phẩm với các bộ lọc phức tạp (danh mục, khoảng giá, từ khóa tìm kiếm) và phân trang. Logic này được xây dựng để tổng hợp các điều kiện tìm kiếm và gọi các phương thức phù hợp từ ProductRepository.
* **CartService**: Quản lý các thao tác liên quan đến giỏ hàng (thêm, cập nhật, xóa sản phẩm trong giỏ).
* **IdGeneratorService**: Dịch vụ hỗ trợ tạo ID duy nhất cho các Entity.

**3.1.5. Controller**

Các lớp Controller (trong com.example.bakery.controller) là các REST API endpoints, xử lý các yêu cầu HTTP từ frontend.

* **AuthController**: Xử lý các yêu cầu đăng ký (/register) và đăng nhập.
  + Sử dụng @Valid và BindingResult để bắt các lỗi validation từ DTO (@Size cho mật khẩu, @NotBlank, @Email, @Pattern).
  + Kiểm tra trùng lặp email và số điện thoại trước khi lưu vào DB.
* **ProductController**: Cung cấp các API để lấy danh sách sản phẩm (có phân trang, lọc, tìm kiếm), và xem chi tiết sản phẩm.
  + Sử dụng @RequestParam để nhận các tham số lọc, tìm kiếm, phân trang.
* **CartController**: Cung cấp các API để thêm, cập nhật, xóa sản phẩm khỏi giỏ hàng.
  + Sử dụng @RequestBody để nhận DTO AddItemToCartRequest.

**3.1.6. Xử lý lỗi Backend**

Hệ thống backend được thiết kế để trả về các thông báo lỗi rõ ràng cho frontend:

* **Lỗi Validation (400 Bad Request):** Khi dữ liệu đầu vào không hợp lệ (ví dụ: mật khẩu dưới 8 ký tự, email/số điện thoại trống), AuthController sẽ trả về AuthResponse với errorMessage chi tiết được thu thập từ BindingResult.
* **Lỗi Logic Nghiệp vụ (400 Bad Request):** Khi có lỗi trùng lặp dữ liệu (email/số điện thoại đã tồn tại), AuthController sẽ trả về AuthResponse với thông báo cụ thể (ví dụ: "Email đã được sử dụng.").
* **Lỗi Server (500 Internal Server Error):** Các lỗi không lường trước được sẽ được bắt bởi try-catch và trả về thông báo lỗi chung cho người dùng, đồng thời log chi tiết trên server để debug.

**3.1.7. Auditing**

Để tự động quản lý các trường created\_at và last\_modified\_at trong các Entity (như Cart, User, Product), tính năng JPA Auditing của Spring Data đã được sử dụng:

* Annotation @EnableJpaAuditing được thêm vào lớp BakeryApplication.java để kích hoạt Auditing.
* Trong các Entity, các trường thời gian được chú thích bằng @CreatedDate và @LastModifiedDate.
* Annotation @EntityListeners(AuditingEntityListener.class) được thêm vào lớp Entity để lắng nghe các sự kiện lifecycle và tự động điền giá trị.

### 3.2. Phát triển Frontend (ReactJS)

### 3.2.1. Cấu trúc thư mục

Dự án Frontend tuân thủ cấu trúc thư mục chuẩn của React:

src

├── assets // Hình ảnh, video tĩnh

├── components // Các component tái sử dụng (ProductCard, CartItem, ...)

├── context // Context API (AuthContext, CartContext)

├── hooks // Các custom hook (useInView)

├── pages // Các component trang chính (HomePage, ProductsPage, ...)

├── services // Các hàm gọi API (productService, cartService, authService)

├── App.js // Component gốc

├── index.js // Entry point

└── ...

### 3.2.2. Các trang (Pages)

* **HomePage.jsx**:
  + Hiển thị các sản phẩm nổi bật (featuredProducts) và sản phẩm mới nhất (latestProducts) bằng cách gọi API thông qua productService.
  + Tích hợp video giới thiệu cửa hàng (flour\_-\_49549-1080p.mp4) sử dụng thẻ <video> với các thuộc tính autoplay, loop, muted, playsInline để video tự động phát ngay khi trang tải.
  + Sử dụng useInView custom hook để thêm hiệu ứng animation trượt lên mượt mà khi người dùng cuộn đến phần video.
* **ProductsPage.jsx**:
  + Quản lý trạng thái cho danh sách sản phẩm, phân trang, lọc (theo danh mục, khoảng giá) và tìm kiếm.
  + Sử dụng useEffect để gọi API productService.getPaginatedProducts mỗi khi các tham số lọc, tìm kiếm hoặc phân trang thay đổi.
  + Triển khai tính năng tìm kiếm phía server với **debounce** để tối ưu hóa số lượng yêu cầu API khi người dùng nhập từ khóa.
  + Hiển thị danh sách sản phẩm bằng cách map qua mảng products và render ProductCard cho mỗi sản phẩm.
* **ProductDetailPage.jsx**:
  + Hiển thị thông tin chi tiết của một sản phẩm cụ thể dựa trên productId từ URL.
  + Cho phép người dùng thêm sản phẩm vào giỏ hàng từ trang chi tiết.
* **CartPage.jsx**:
  + Hiển thị danh sách các sản phẩm trong giỏ hàng (cart.cartItems).
  + Cho phép người dùng cập nhật số lượng và xóa từng mặt hàng khỏi giỏ.
  + Hiển thị tổng số tiền của giỏ hàng.

### 3.2.3. Các Component (Components)

* **ProductCard.jsx**: Component tái sử dụng để hiển thị thông tin cơ bản của một sản phẩm (tên, giá, hình ảnh).
  + Nhận một đối tượng product làm prop.
  + Sử dụng PropTypes để định nghĩa rõ ràng cấu trúc và kiểu dữ liệu của prop product, giúp phát hiện lỗi sớm trong quá trình phát triển.
* **CartItem.jsx**: (Nếu được tách ra từ CartPage.jsx) Component hiển thị chi tiết một mặt hàng trong giỏ hàng. Cũng sử dụng PropTypes để kiểm tra prop item.
* **LoginForm.jsx & RegisterForm.jsx**: Các component form cho phép người dùng đăng nhập và đăng ký.

**3.2.4. Service API (Axios)**

Các file service (trong src/services/, ví dụ productService.js, cartService.js, authService.js) chứa các hàm JavaScript để gửi yêu cầu HTTP (GET, POST) đến các REST API của backend bằng thư viện Axios.

* **axiosConfig.js**: Cấu hình một instance Axios chung để tự động thêm JWT token vào header Authorization cho các yêu cầu đã xác thực, và xử lý CORS.
* Các hàm service được thiết kế để nhận các tham số cần thiết và trả về dữ liệu hoặc ném lỗi để frontend xử lý.

**3.2.5. Context API**

Ứng dụng sử dụng React Context API để quản lý trạng thái toàn cục, giúp chia sẻ dữ liệu giữa các component mà không cần truyền props qua nhiều cấp:

* **AuthContext.js**: Quản lý trạng thái đăng nhập của người dùng (logged-in status, user info, JWT token, cartId của người dùng). Cung cấp các hàm login, logout, register.
* **CartContext.js**: Quản lý trạng thái giỏ hàng (danh sách sản phẩm trong giỏ, tổng tiền, trạng thái tải). Cung cấp các hàm addToCart, updateCartItemQuantity, removeCartItem, fetchCart.

**3.2.6. Định tuyến (Routing)**

Sử dụng thư viện react-router-dom để quản lý các tuyến đường trong ứng dụng. Các tuyến đường chính bao gồm:

* /: Trang chủ (HomePage).
* /products: Trang danh sách sản phẩm (ProductsPage).
* /products/:productId: Trang chi tiết sản phẩm (ProductDetailPage).
* /cart: Trang giỏ hàng (CartPage).
* /login: Trang đăng nhập (LoginPage).
* /register: Trang đăng ký (RegisterPage).

**3.2.7. Validation Frontend**

Để cung cấp phản hồi tức thì cho người dùng và giảm tải cho backend, một số validation cơ bản được thực hiện ở frontend:

* Trong LoginForm.jsx và RegisterForm.jsx, logic validation được thêm vào hàm handleSubmit để kiểm tra các trường bắt buộc, định dạng email/số điện thoại, và độ dài mật khẩu trước khi gửi yêu cầu đến backend.

**3.2.8. PropTypes**

PropTypes được sử dụng rộng rãi trong dự án để kiểm tra kiểu dữ liệu của các props được truyền vào các component.

* **ProductCard.jsx**: Định nghĩa PropTypes.shape cho prop product để đảm bảo nó có các thuộc tính như productId (string, required), name (string, required), price (number, required), imageUrl (string, required).
* Việc này giúp phát hiện lỗi sớm trong quá trình phát triển nếu một component nhận được props sai kiểu, cải thiện khả năng đọc và bảo trì mã.

**3.3. Xử lý lỗi và Debugging**

Trong quá trình phát triển, một số lỗi phổ biến đã được gặp phải và khắc phục:

* **com.fasterxml.jackson.databind.exc.MismatchedInputException**: Lỗi này xảy ra khi định dạng JSON gửi từ frontend không khớp với cấu trúc DTO ở backend (cụ thể là trường productId được gửi dưới dạng object thay vì string).
  + **Giải pháp:** Điều chỉnh code frontend (trong ProductCard.jsx và hàm addToCart trong CartContext.js) để đảm bảo chỉ gửi chuỗi productId trong payload JSON.
* **HTTP 403 Forbidden**: Lỗi này thường liên quan đến Spring Security, có thể do thiếu quyền truy cập cho endpoint hoặc thiếu/sai token xác thực.
  + **Giải pháp:** Kiểm tra cấu hình SecurityConfig để đảm bảo endpoint được phép truy cập (.authenticated() hoặc .permitAll()) và đảm bảo frontend gửi JWT token hợp lệ trong header Authorization.
* **Column 'created\_at' cannot be null**: Lỗi này xảy ra khi một trường NOT NULL trong DB không được điền giá trị.
  + **Giải pháp:** Kích hoạt JPA Auditing trong Spring Boot (@EnableJpaAuditing) và chú thích các trường thời gian trong Entity (@CreatedDate, @LastModifiedDate) để chúng được tự động điền giá trị.
* **Quy trình Debugging:**
  + Sử dụng console.log chi tiết ở frontend để kiểm tra giá trị của các biến và payload trước khi gửi yêu cầu.
  + Sử dụng tab "Network" trong Developer Tools của trình duyệt để kiểm tra request/response HTTP (status code, headers, payload).
  + Kiểm tra log của backend (Spring Boot) để xem chi tiết stack trace của lỗi server.
  + Sử dụng breakpoint và debug mode trong IDE để theo dõi luồng thực thi code backend.

## **VI. KẾT QUẢ CHẠY VÀ MÀN HÌNH**

**4.1. Hướng dẫn cài đặt và chạy**

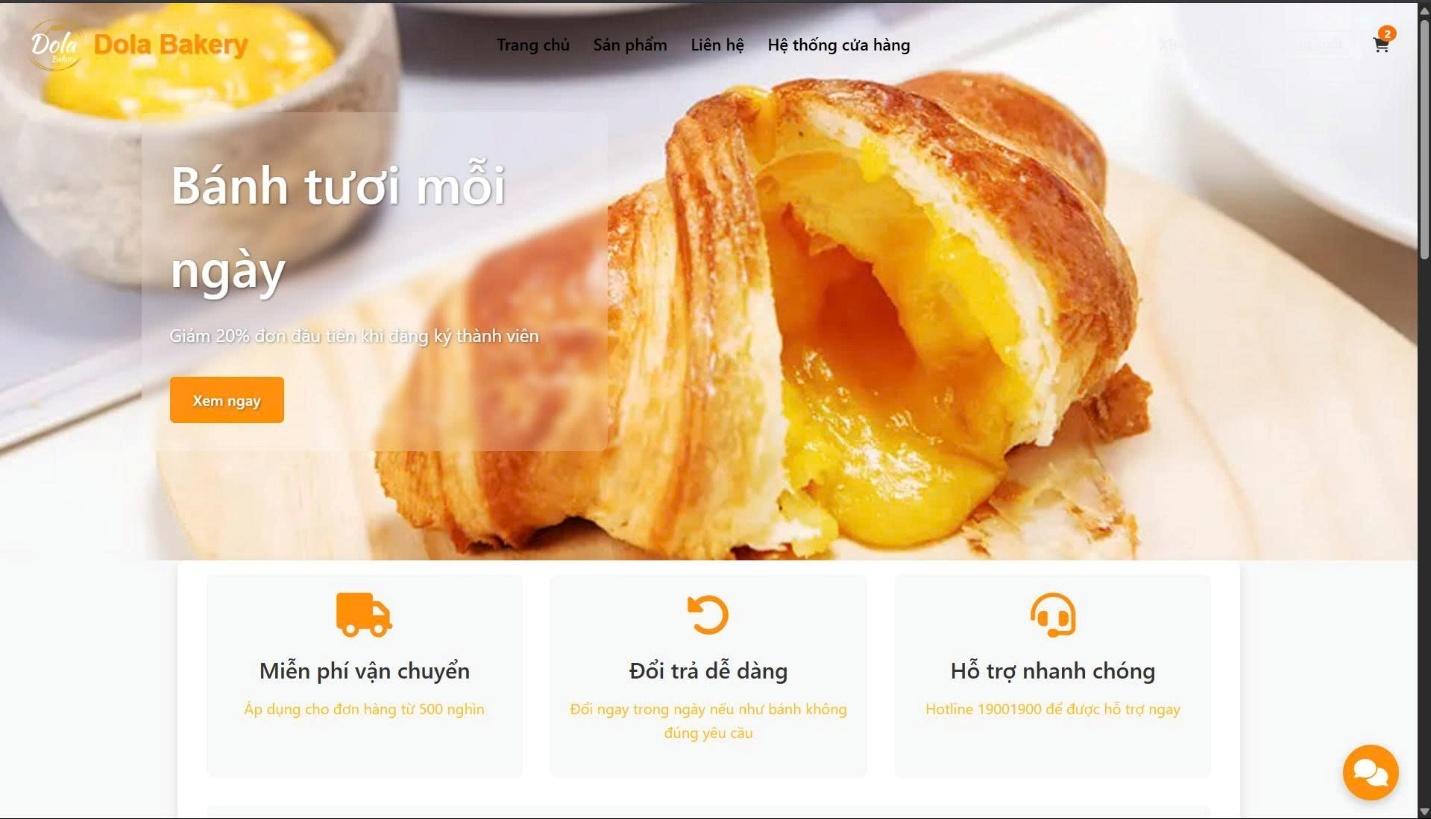
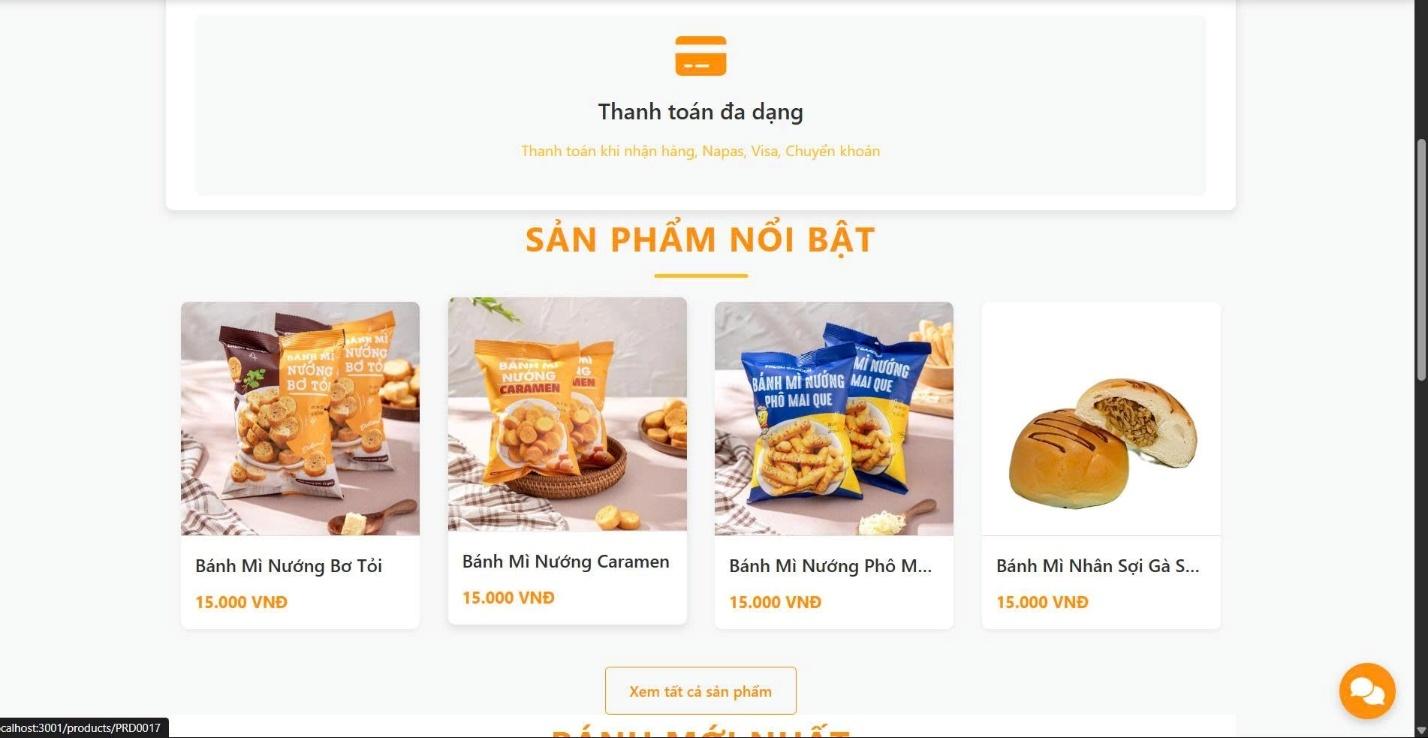
Để chạy dự án này, bạn cần thực hiện các bước sau:

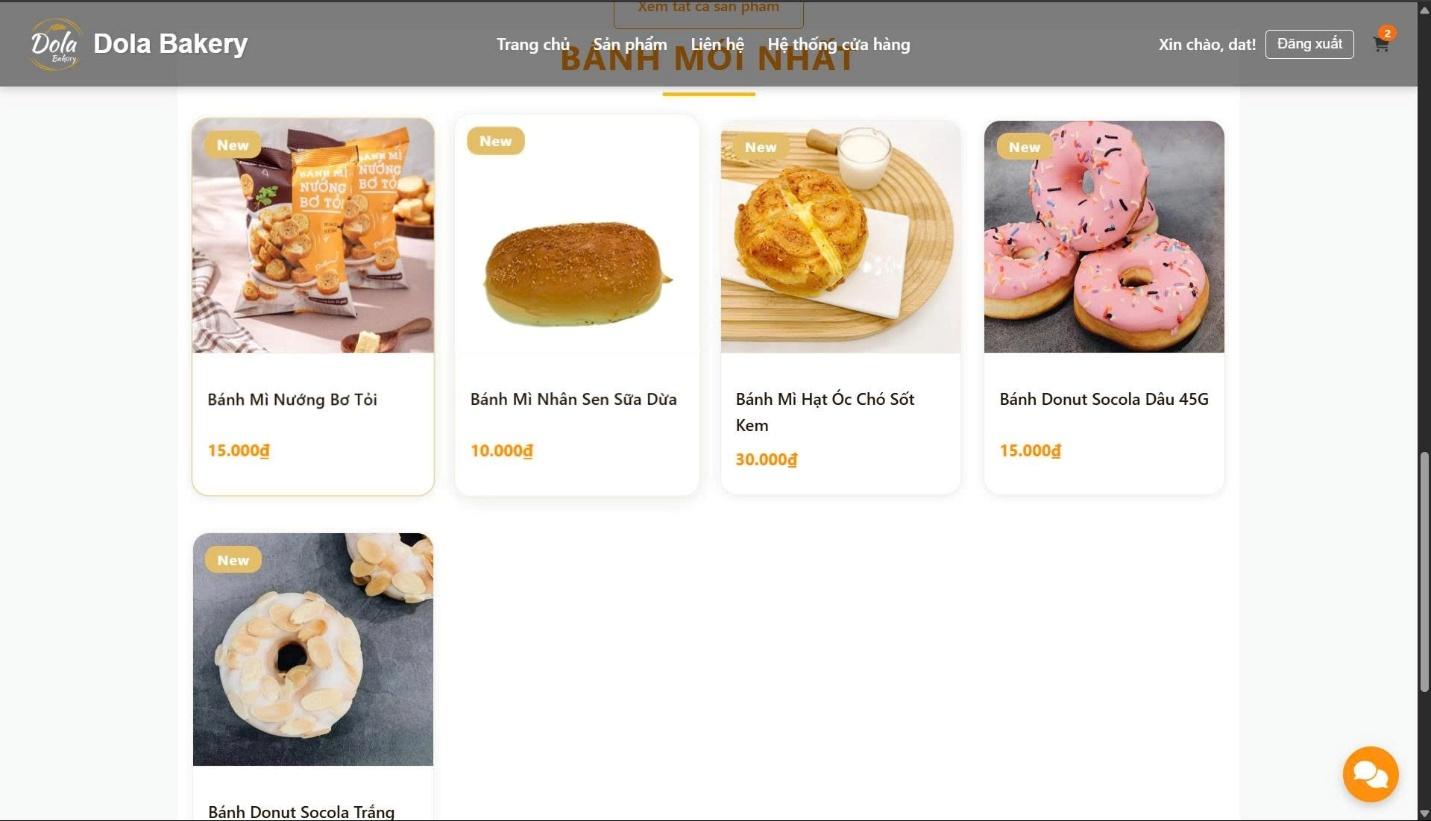
1. **Cài đặt Backend:**
   * **Yêu cầu:** Java 17+, Maven/Gradle, MySQL (hoặc cơ sở dữ liệu tương thích).
   * **Clone Repository:** git clone [URL\_repository\_backend\_của\_bạn]
   * **Cấu hình Database:** Mở file src/main/resources/application.properties (hoặc application.yml) và cấu hình thông tin kết nối MySQL của bạn (URL, username, password).
   * **Build Project:** Mở terminal trong thư mục gốc của backend và chạy mvn clean install (hoặc gradle clean build).
   * **Chạy ứng dụng:** mvn spring-boot:run (hoặc chạy từ IDE). Backend sẽ chạy trên http://localhost:8080.
2. **Cài đặt Frontend:**
   * **Yêu cầu:** Node.js (phiên bản LTS), npm/yarn.
   * **Clone Repository:** git clone [URL\_repository\_frontend\_của\_bạn]
   * **Cài đặt Dependencies:** Mở terminal trong thư mục gốc của frontend và chạy npm install (hoặc yarn install).
   * **Chạy ứng dụng:** npm start (hoặc yarn start). Frontend sẽ chạy trên http://localhost:3000.

**4.2. Các ảnh chụp màn hình**

Dưới đây là một số ảnh chụp màn hình minh họa các chức năng chính của ứng dụng:

* **Trang chủ (HomePage):** Hiển thị banner, các tính năng nổi bật, danh sách sản phẩm nổi bật, sản phẩm mới nhất và phần video giới thiệu. Video sẽ tự động phát khi trang tải.

****



 A screen shot of a computer program

AI-generated content may be incorrect.

* **Trang danh sách sản phẩm (ProductsPage):** Hiển thị lưới sản phẩm với các tùy chọn lọc theo danh mục, khoảng giá, sắp xếp và ô tìm kiếm. Phân trang được hiển thị ở cuối danh sách.

A screenshot of a menu

AI-generated content may be incorrect.A screenshot of a food website

AI-generated content may be incorrect.A screenshot of a food menu

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect. A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect. A screen shot of a computer program

AI-generated content may be incorrect.

* **Trang chi tiết sản phẩm (ProductDetailPage):** Hiển thị thông tin chi tiết của một sản phẩm, bao gồm hình ảnh, tên, mô tả, giá, số lượng tồn kho, và nút "Thêm vào giỏ hàng".

A screenshot of a food website

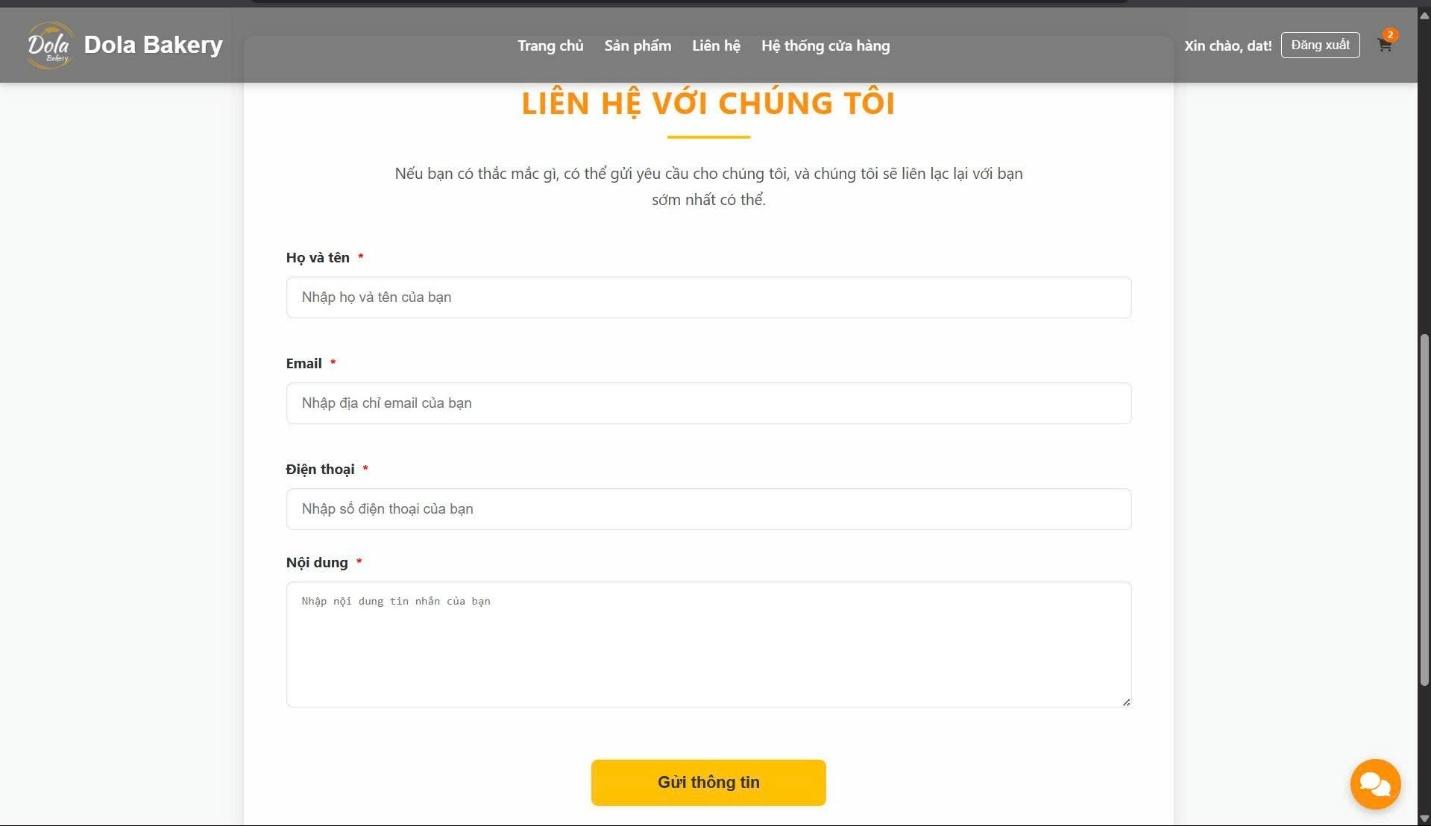
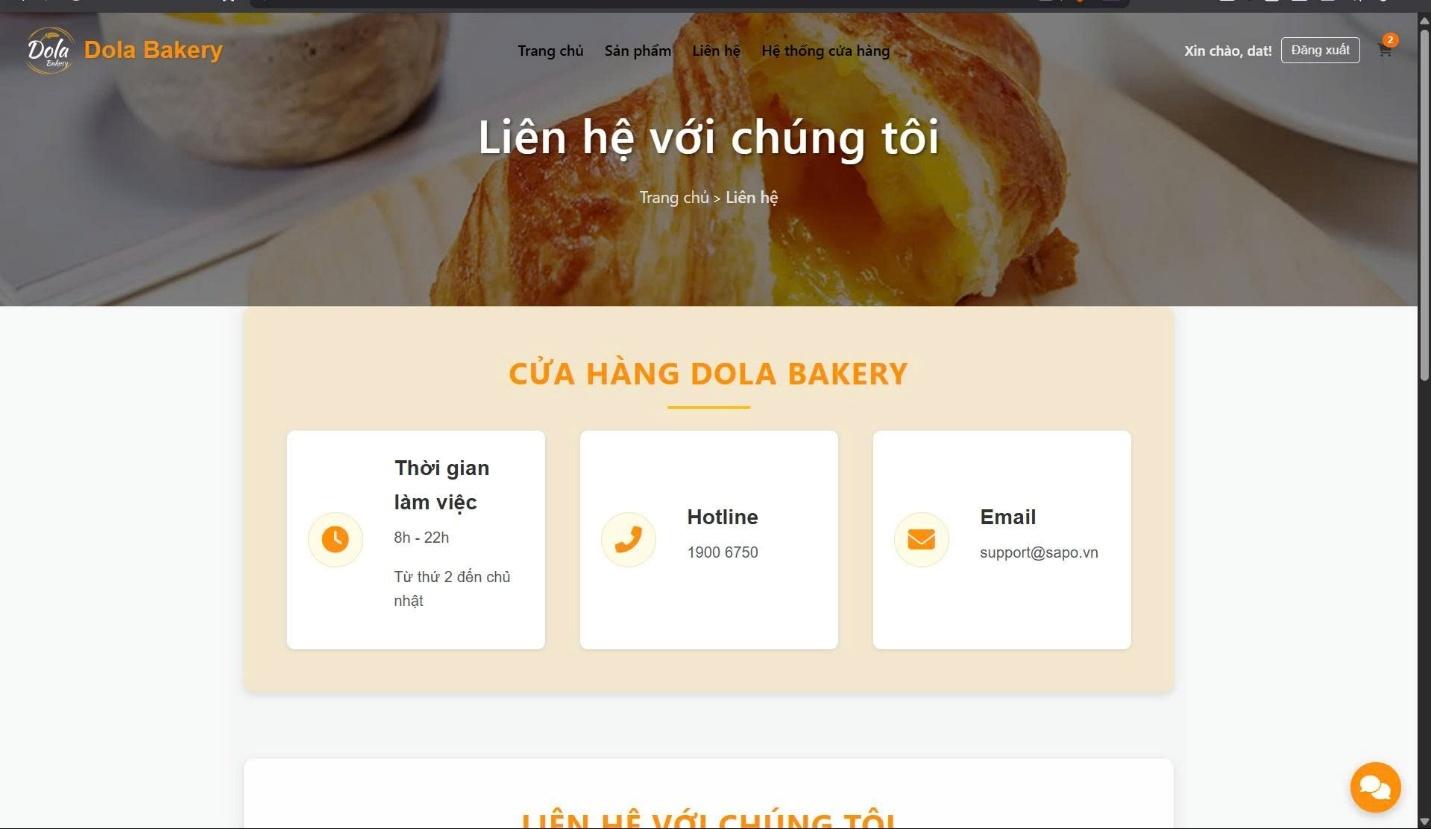
AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

* **Trang giỏ hàng (CartPage):** Hiển thị các sản phẩm đã thêm vào giỏ, số lượng, đơn giá, thành tiền, tổng tiền và các nút thao tác (cập nhật số lượng, xóa, xóa tất cả).



* **Form Đăng ký/Đăng nhập (RegisterForm/LoginForm):** Minh họa giao diện form và các thông báo lỗi validation (ví dụ: mật khẩu dưới 8 ký tự, email đã tồn tại).Để minh họa trực quan hơn các tính năng và luồng hoạt động của ứng dụng, một video demo ngắn

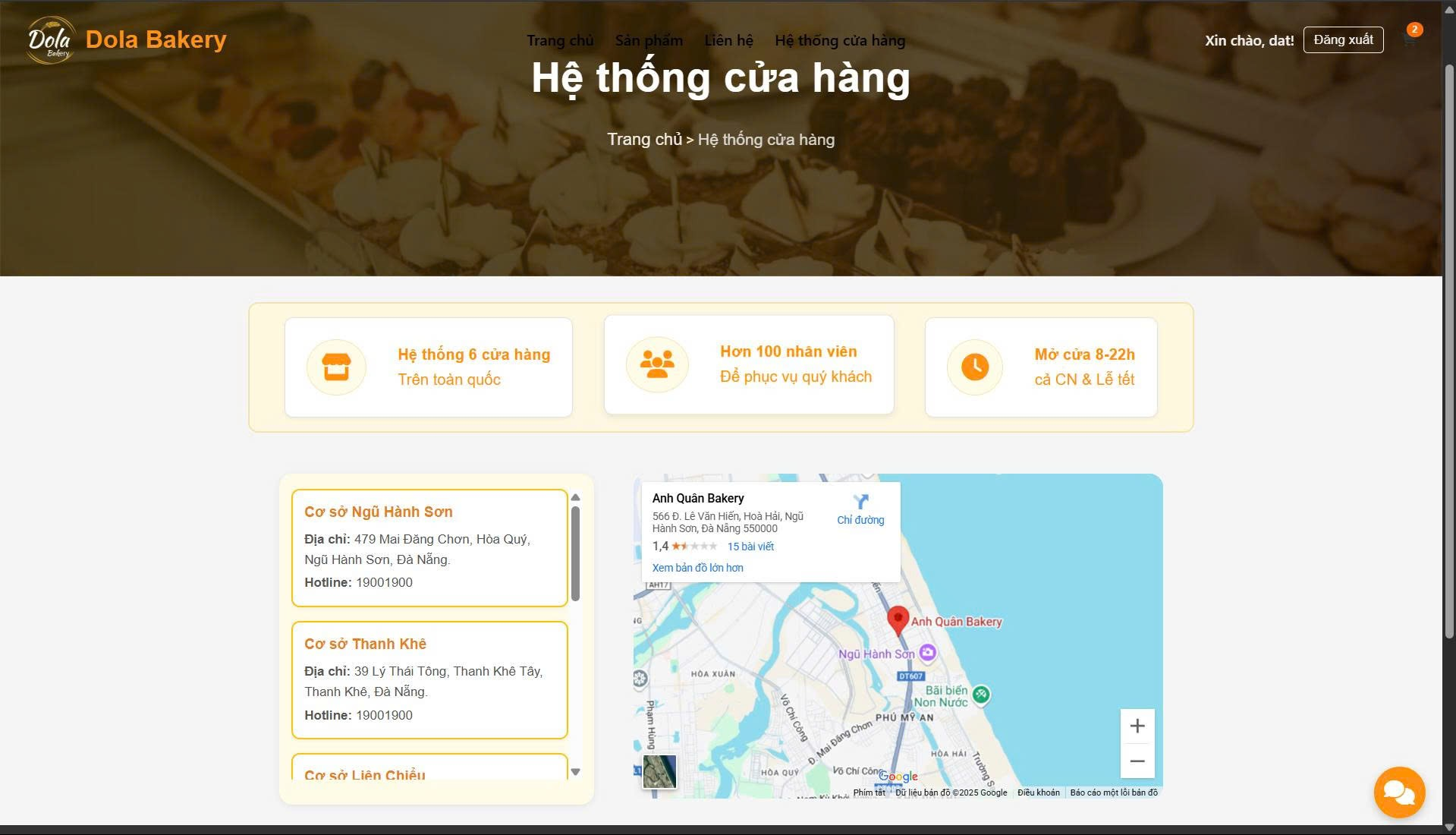
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* **Hệ thống cửa hàng:** Dùng API Google Map để lấy địa chỉ các cửa hàng và hiển thị lên website



**VII. ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN**

**5.1. Đánh giá dự án**

Dự án đã hoàn thành các mục tiêu chính được đề ra trong phạm vi assignment, bao gồm:

* **Kiến trúc rõ ràng:** Ứng dụng được xây dựng trên kiến trúc Client-Server với Spring Boot backend và ReactJS frontend, đảm bảo tính module hóa và dễ bảo trì.
* **Chức năng cốt lõi:** Các tính năng quản lý sản phẩm (hiển thị, lọc, tìm kiếm, phân trang), quản lý giỏ hàng và xác thực người dùng đã được triển khai thành công.
* **Trải nghiệm người dùng:** Giao diện trực quan, thân thiện, và các tính năng như debounce cho tìm kiếm, animation video trên trang chủ góp phần cải thiện trải nghiệm.
* **Xử lý lỗi:** Các lỗi phổ biến như không khớp JSON payload, lỗi bảo mật 403, và lỗi ràng buộc cơ sở dữ liệu đã được xác định và khắc phục, cho thấy khả năng debug và xử lý vấn đề hiệu quả. Việc trả về thông báo lỗi chi tiết từ backend giúp frontend cung cấp phản hồi tốt hơn cho người dùng.
* **Chất lượng mã:** Việc sử dụng PropTypes ở frontend và các annotation Auditing ở backend đã nâng cao chất lượng mã, giúp kiểm soát kiểu dữ liệu và tự động hóa các tác vụ lặp lại.

**Thách thức và Giải pháp:** Trong quá trình phát triển, một số thách thức đã được gặp phải:

* **Đồng bộ dữ liệu Frontend-Backend:** Lỗi MismatchedInputException là một ví dụ điển hình về sự không khớp giữa cấu trúc JSON gửi đi và DTO nhận ở backend. Giải pháp là kiểm tra kỹ payload và đảm bảo nó khớp với DTO.
* **Cấu hình Spring Security:** Lỗi 403 Forbidden thường gây khó khăn trong việc xác định nguyên nhân. Việc kiểm tra cấu hình quyền truy cập và cơ chế xác thực (JWT token) là rất quan trọng.
* **Quản lý trạng thái phức tạp:** Với nhiều bộ lọc, phân trang và tìm kiếm, việc quản lý trạng thái trong ProductsPage đòi hỏi sự cẩn thận với useEffect và dependency array để tránh vòng lặp vô hạn hoặc lỗi fetch dữ liệu. Kỹ thuật debounce đã được áp dụng để tối ưu hóa.
* **Auditing:** Đảm bảo các trường created\_at, last\_modified\_at được tự động điền giá trị là một bước quan trọng để duy trì tính toàn vẹn dữ liệu.

**Bài học kinh nghiệm:**

* Tầm quan trọng của việc định nghĩa rõ ràng "API Contract" (cấu trúc JSON của request/response) giữa frontend và backend.
* Hiểu sâu về cơ chế bảo mật của Spring Security.
* Kỹ năng debug là tối quan trọng, đặc biệt là việc đọc hiểu stack trace và sử dụng công cụ phát triển của trình duyệt.
* Quản lý trạng thái hiệu quả trong React với useState, useEffect, và Context API.
* Lợi ích của việc sử dụng các thư viện kiểm tra kiểu dữ liệu như PropTypes để tăng cường độ tin cậy của mã.

**5.2. Hướng phát triển tương lai (Future Work)**

Để mở rộng và cải thiện ứng dụng, các tính năng sau có thể được xem xét trong tương lai:

* **Hệ thống Đặt hàng và Thanh toán:** Triển khai quy trình đặt hàng đầy đủ, tích hợp cổng thanh toán trực tuyến (ví dụ: Stripe, PayPal, VNPay).
* **Trang quản trị (Admin Dashboard):** Cho phép người bán quản lý sản phẩm, đơn hàng, người dùng, và danh mục.
* **Quản lý Người dùng nâng cao:** Hồ sơ người dùng, lịch sử mua hàng, thay đổi mật khẩu.
* **Đánh giá và Bình luận sản phẩm:** Cho phép khách hàng để lại đánh giá và bình luận về sản phẩm.
* **Hệ thống khuyến mãi và mã giảm giá:** Tích hợp các chương trình khuyến mãi.
* **Tối ưu hóa hiệu suất:** Tối ưu hóa truy vấn cơ sở dữ liệu, lazy loading hình ảnh, code splitting ở frontend.
* **Triển khai (Deployment):** Đưa ứng dụng lên môi trường production (ví dụ: Heroku, AWS, Google Cloud).
* **Thêm Test:** Viết Unit/Integration/End-to-End tests cho cả frontend và backend.
* **Sử dụng TypeScript:** Chuyển đổi dự án sang TypeScript để có kiểm tra kiểu dữ liệu mạnh mẽ hơn trong quá trình phát triển.

**VIII. KẾT LUẬN**

Dự án "Ứng Dụng Web Bán Bánh Ngọt Trực Tuyến" đã được phát triển thành công, đáp ứng các yêu cầu cơ bản của assignment. Qua quá trình thực hiện, nhóm đã học hỏi và áp dụng nhiều kiến thức quan trọng về phát triển web full-stack, từ thiết kế cơ sở dữ liệu, xây dựng API với Spring Boot, đến phát triển giao diện người dùng tương tác với ReactJS. Việc đối mặt và giải quyết các lỗi thực tế đã củng cố kỹ năng debug và tư duy giải quyết vấn đề. Dự án này là một nền tảng vững chắc để tiếp tục phát triển và mở rộng trong tương lai.

**IX. TÀI LIỆU THAM KHẢO**

* Tài liệu chính thức của ReactJS: https://react.dev/
* Tài liệu chính thức của Spring Boot: https://spring.io/projects/spring-boot
* Tài liệu chính thức của Spring Data JPA: https://docs.spring.io/spring-data/jpa/reference/
* Tài liệu chính thức của Spring Security: https://docs.spring.io/spring-security/reference/
* Tài liệu chính thức của Axios: https://axios-http.com/docs/intro
* Tài liệu chính thức của PropTypes: https://react.dev/learn/typechecking-with-proptypes
* MDN Web Docs (HTML, CSS, JavaScript): https://developer.mozilla.org/
* Stack Overflow và các diễn đàn lập trình khác.