FINAL PROJECT REPORT

SEMESTER 1, ACADEMIC YEAR: 2024-2025

CT313H: WEB TECHNOLOGIES AND SERVICES

- **Project/Application name: Fast Food Website**
- **GitHub links (for both frontend and backend): https://github.com/24-25Sem1-Courses/ct313h01-project-lhtoan**
- **Student ID 1: B2111960**
- **Student Name 1: Lê Huy Toàn**
- **Student ID 2:B2112016**
- **Student Name 2: Võ Duy Toàn**
- **Class/Group Number: CT313HM01**
- **Link video: https://youtu.be/HfM6shiwxLE**

I. Introduction

The fast food web application allows users to manage products and customers, offering functionalities to add, edit, and delete items. Customers can also browse products and add them to their shopping cart, making the ordering process seamless and efficient.

We have 4 tables including customers, products, orders and detailorders:

- The user table stores the information about account to login into the web:

Name	Data Type	Description
id	INT (AUTO_INCREMENT)	Unique identifier for each user(Primary Key, auto-incremented).
username	VARCHAR(255)	Username of the customer, stored as a string (maximum 255 characters).
password	VARCHAR(255)	Password of the customer, stored as a string (maximum 255 characters).
user_role	INT	Role of the user (default 1: user, 0: admin)

- The customers table stores the information about the customers who place orders on the system. It includes the following fields:

Name	Data Type	Description

id_customer	INT (AUTO_INCREMENT)	Unique identifier for each customer (Primary Key, auto-incremented).
name	VARCHAR(255)	Name of the customer, stored as a string (maximum 255 characters).
email	VARCHAR(255)	Email address of the customer, stored as a string (maximum 255 characters). This field is unique to prevent duplicate emails.
address	VARCHAR(255)	Address of the customer, stored as a string (maximum 255 characters).
phone	VARCHAR(20)	Phone number of the customer, stored as a string (maximum 20 characters). This field is required.
gender	TINYINT(1)	Gender of the customer, stored as a tiny integer (1 indicates male, 0 indicates female).
avatar	VARCHAR(255)	URL of the customer's avatar image, stored as a string (maximum 255 characters).

- The products table contains details about the products available for customers to order. It has the following fields:

Name	Data Type	Description
product_Id	INT (AUTO_INCREMENT)	Unique identifier for each product (Primary Key, auto-incremented).
product_Name	VARCHAR(255)	Name of the product, stored as a string (maximum 255 characters).
product_Price	INT	Price of the product, stored as an integer.
product_Img	VARCHAR(255)	URL of the product's image, stored as a string (maximum 255 characters).

- The orders table records each order placed by customers. It includes

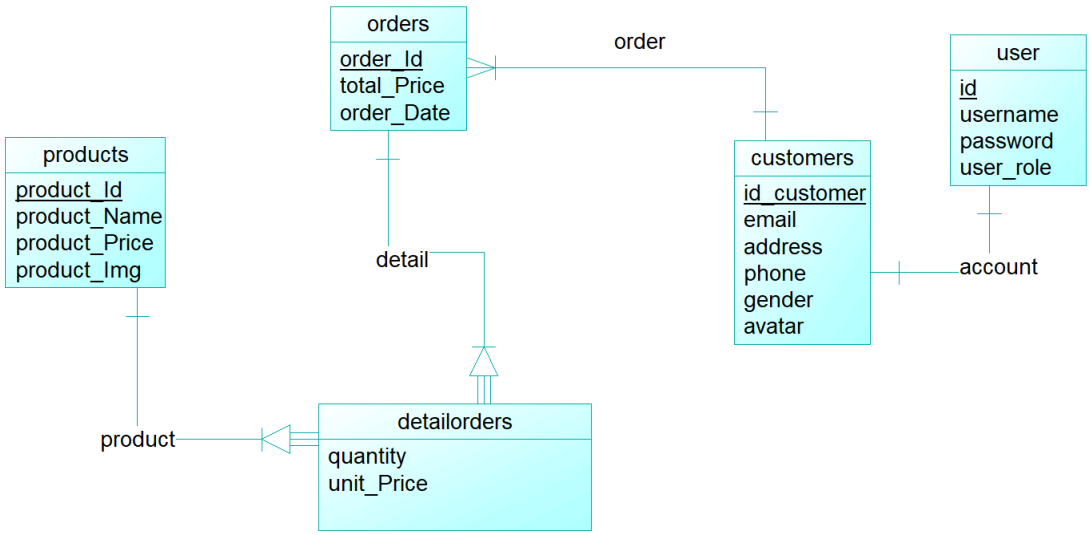
Name	Data Type	Description
order_Id	INT (AUTO_INCREMENT)	Unique identifier for each order (Primary Key, auto-incremented).

customer_Id	INT	Foreign key referencing the id field in the customers table.
order_Date	DATETIME	Date and time when the order was placed.
total_Price	INT	Total price of the order, stored as an integer.

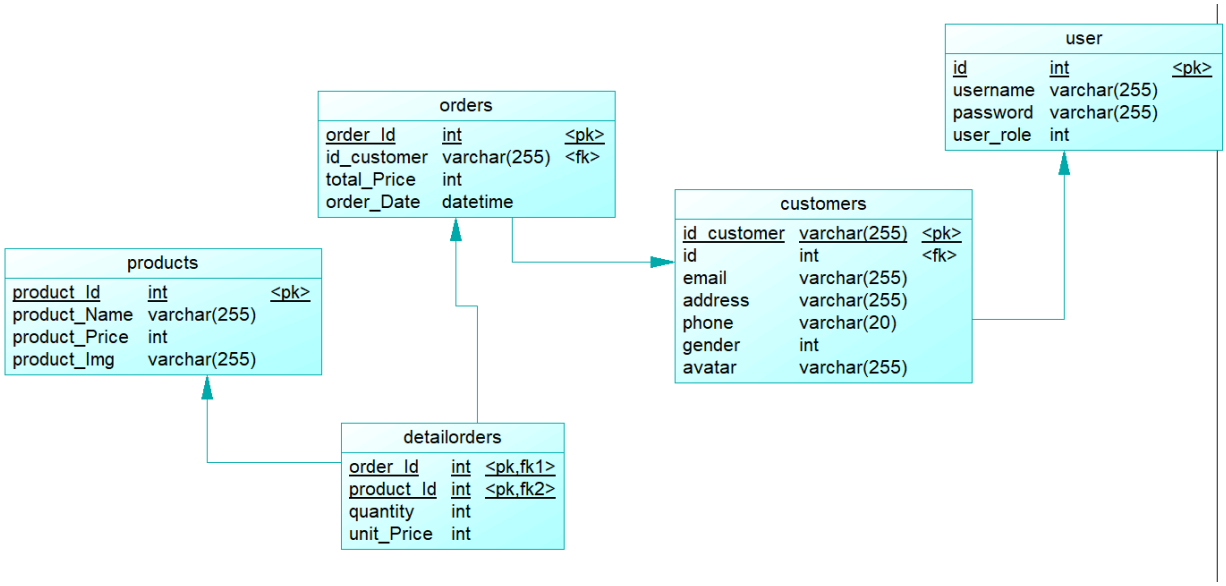
- The detailorders table records the details of each product included in an order, functioning as a bridge between the orders and products tables. It contains the following fields:

Name	Data Type	Description
order_Id	INT	Foreign key referencing the order_Id field in the orders table (part of Primary Key).
product_Id	INT	Foreign key referencing the product_Id field in the products table (part of Primary Key).
quantity	INT	Number of units of the product ordered, stored as an integer.
unit_Price	INT	Unit price of the product at the time of the order, stored as an integer.

- Conceptual Data Model (CDM):



- Physical Data Model (PDM):



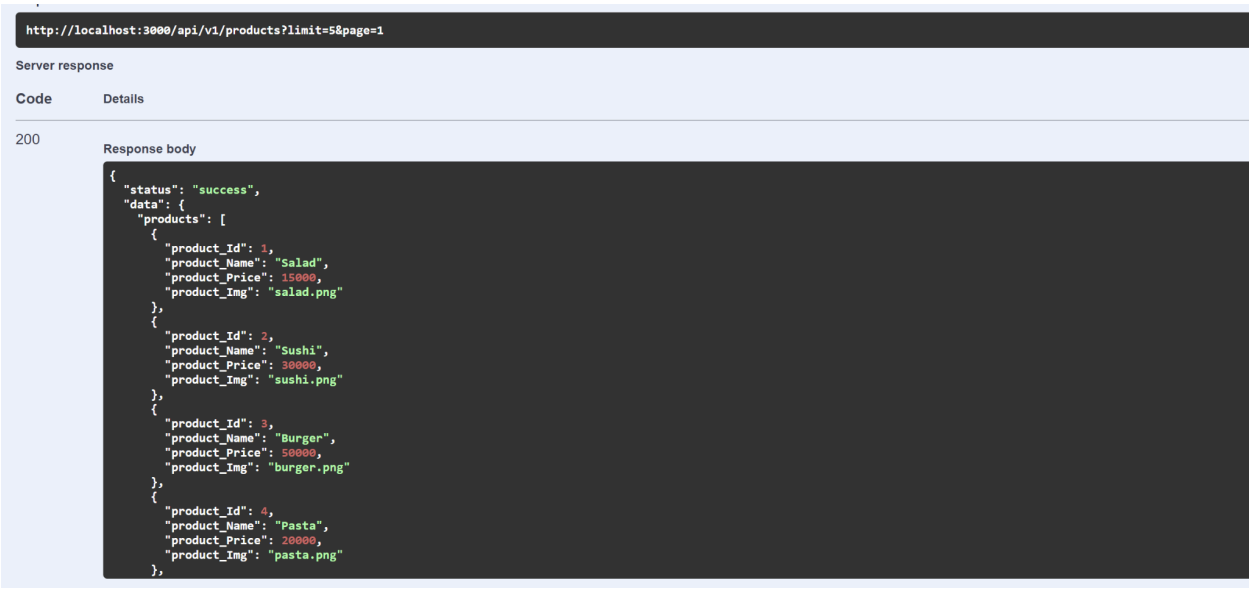
- A task assignment sheet for each member if working in groups.

Le Huy Toan	Vo Duy Toan
Functions: show Customers, show Products, search Product, search Customer, add product to order	Functions: Login, Register, Add a new Product, Update a Product, Update a Customer, Session,

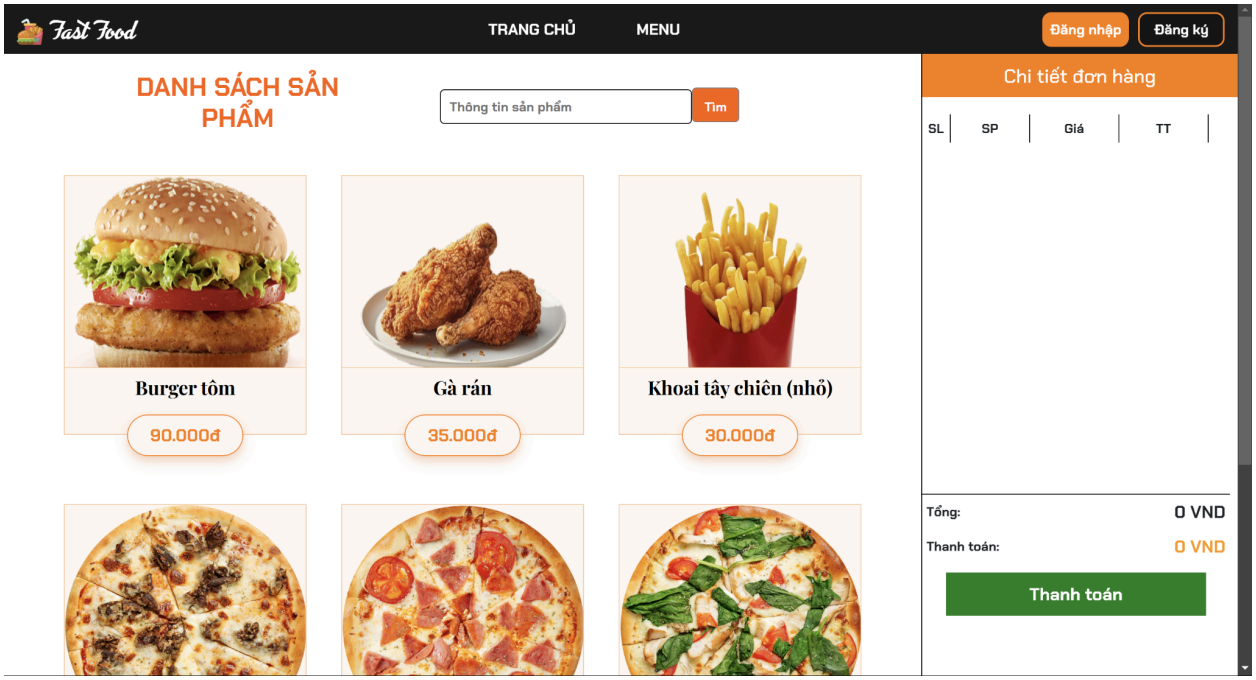
II. Details of implemented features

1. Feature / Application page 1: Get All Products

- **Description:** This API allows retrieving a list of all products from the database, which is used for displaying the list of available products. Users can view existing products along with relevant information and pagination.
- **Screenshots:**



- **Implementation details:**
  - Libraries Used:
    - express: To build the API.
  - Endpoint: GET /api/v1/products
  - Request:
    - limit (number of products per page).
    - page (page number).
  - Response: list of products and pagination information.
  - This feature retrieves the data from the products table.
  - Client-side states are needed to implement this feature: The product list to be displayed on the user interface and pagination information for navigating between pages.

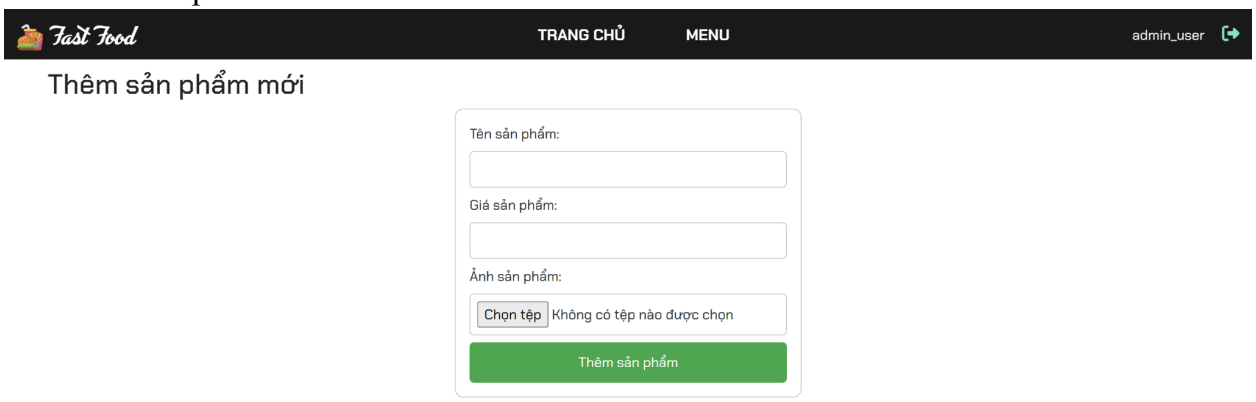


2. Feature / Application page 2: Create a New Product

- **Description:** This feature allows creating a new product and inserting into table products in the database. It is an important part of product management, enabling users to add products with detailed information (product name, price and product image).
- **Screenshots:**

- **Implementation details:**
  - + Libraries Used:
    - + express: To build the API.
    - + multer: To handle file uploads.
  - + Endpoint: POST /api/v1/products
  - + Request:
    - + Body: multipart/form-data with product information.

+ Response: new product information was added to the database.
+ This feature stores the data into the products table.
+ Client-side states are needed to implement this feature: The form enters product details.



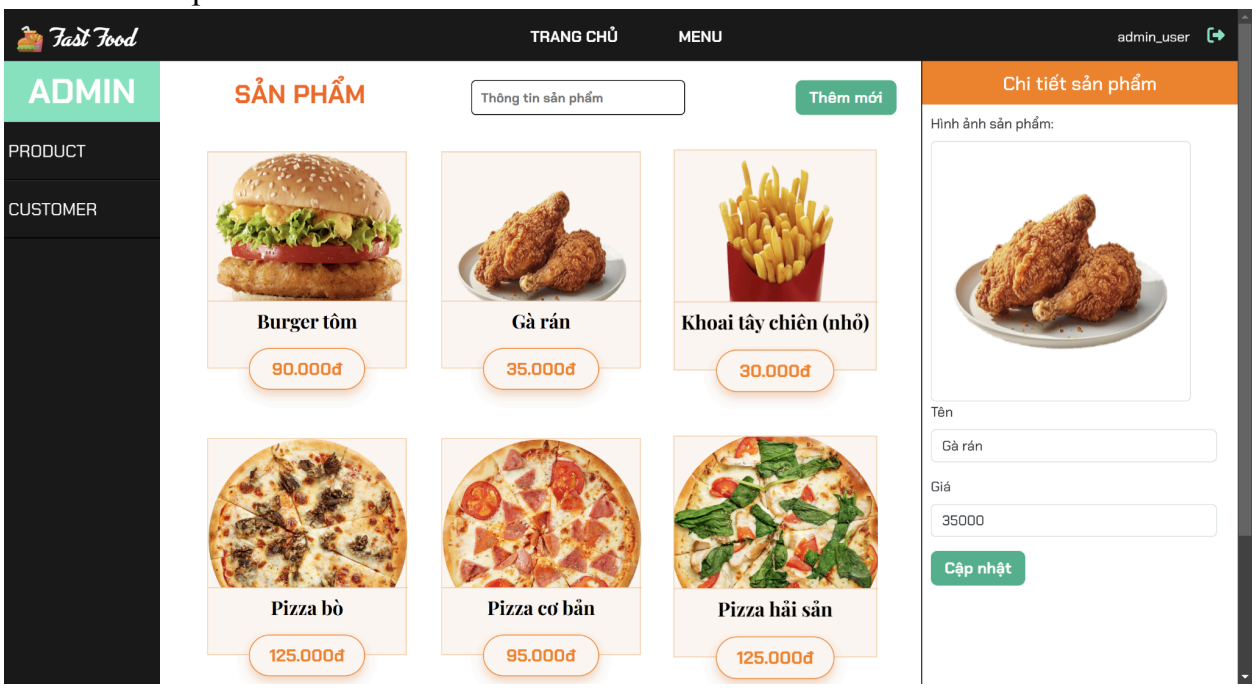
---

3. Feature / Application page 4: Update Product by ID

- **Description:** This feature allows updating the information of a specific product using its ID. This enables users to edit the details of an existing product.
- **Screenshots:**

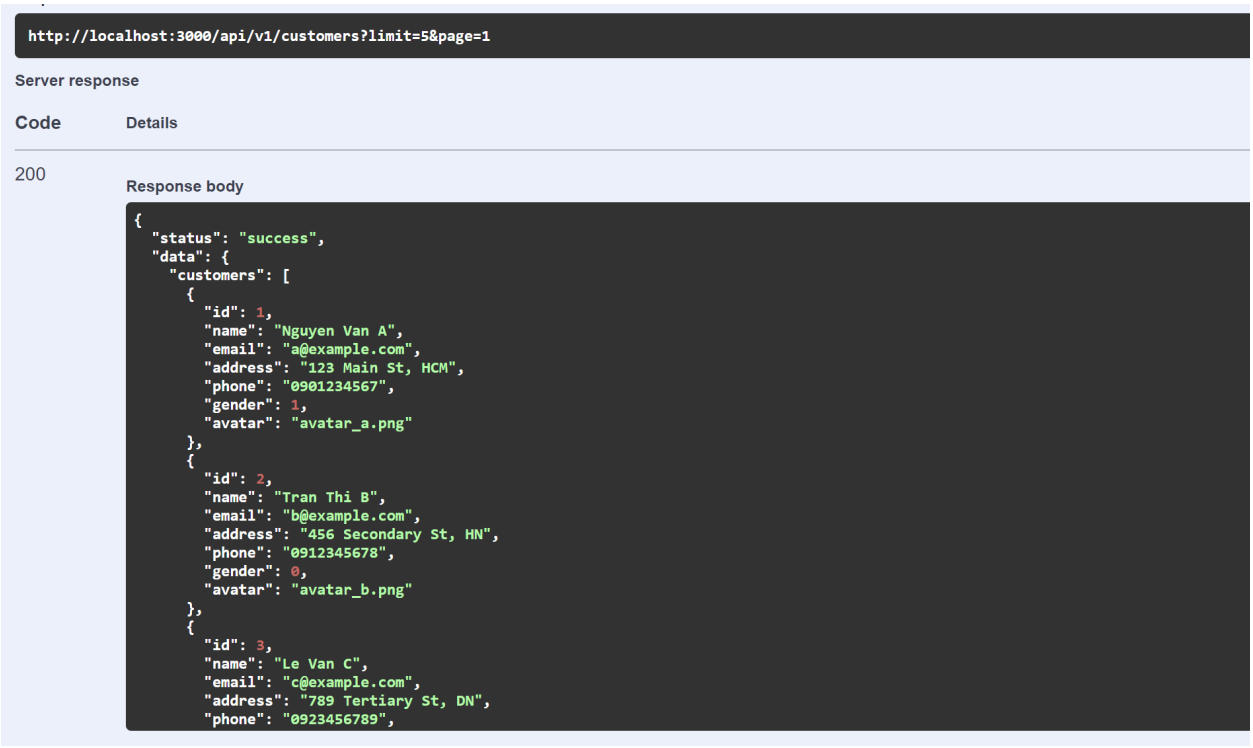
- **Implementation details:**
  + Libraries Used:
    + express: To build the API.
    + multer: To handle file uploads.
  + Endpoint: PUT /api/v1/products/{product_Id}
  + Request:
    + Body: multipart/form-data with product information.

+ Response: updated product information.
+ This feature updates the product information in the products table.
+ Client-side states are needed to implement this feature: The form enters product details.

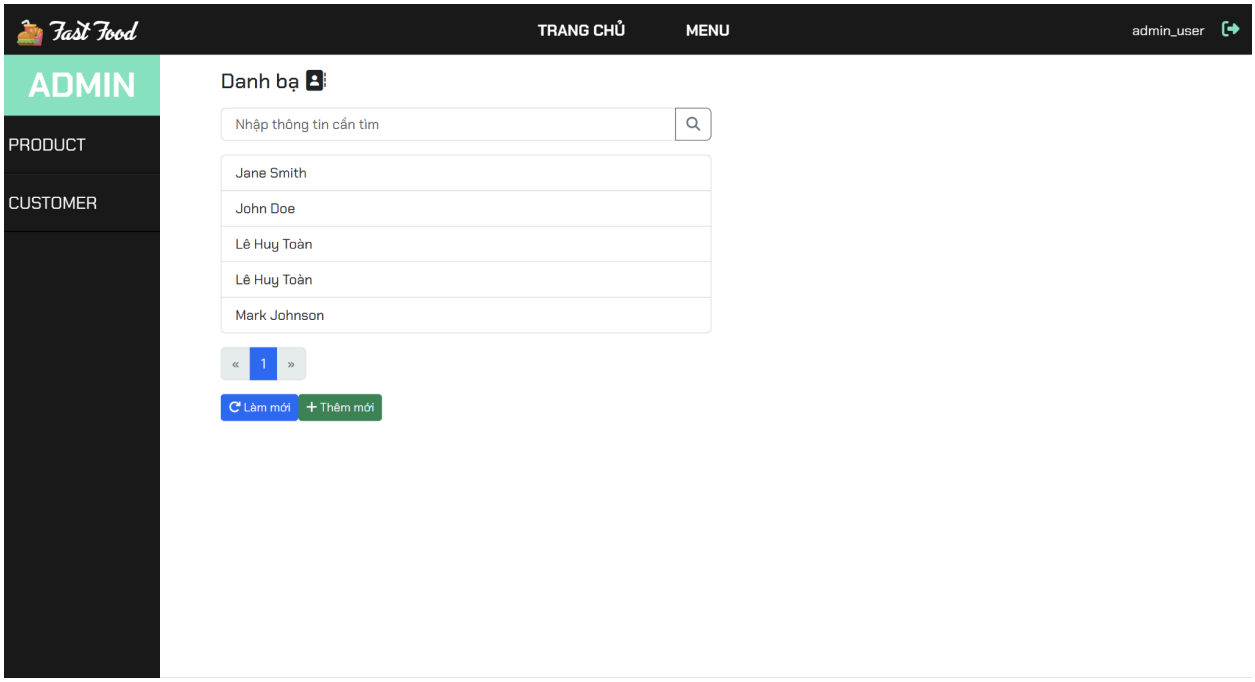


4. Feature / Application page 6: Get All Customers

- **Description:** This API allows retrieving a list of all customers from the table customers in the database. It is used to display the list of available customers, enabling users to view existing customers along with relevant information and pagination.
- **Screenshots:**



- **Implementation details:**
  - Libraries Used:
    - express: To build the API.
  - Endpoint: GET /api/v1/customers
  - Request:
    - limit (number of products per page).
    - page (page number).
  - Response: A list of customers and pagination information.
  - This feature retrieves data from the customers table.
  - Client-side states are needed to implement this feature: The customer list to be displayed on the user interface and pagination information for navigating between pages.

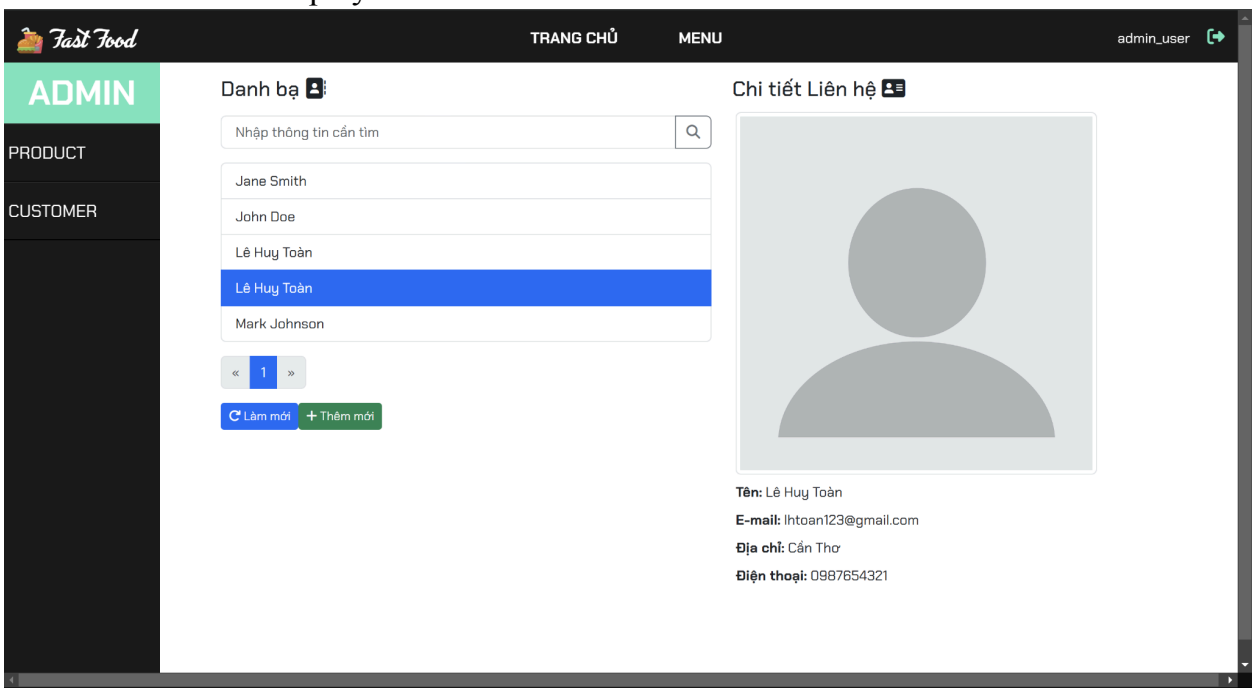


5. Feature / Application page 8: Get Customer by ID

- **Description:** This feature allows retrieving detailed information about a specific customer using their ID. This is useful when users want to view detailed information about a customer.
- **Screenshots:**

- **Implementation details:**
  - Libraries Used:
    - express: To build the API.
  - Endpoint: GET /api/v1/customers/{id}
  - Request:
    - Parameter: id (ID of the customer).
  - Response: Customer information.
  - This feature retrieves data from the customers table.

+ Client-side states needed to implement this feature: Customer information to be displayed in detail on the user interface.



6. Feature / Application page 9: Update Customer by ID

- **Description:** This feature allows updating the information of a specific customer using their ID. It enables users to edit the details of an existing customer.
- **Screenshots:**

```
http://localhost:3000/api/v1/customers/4
```

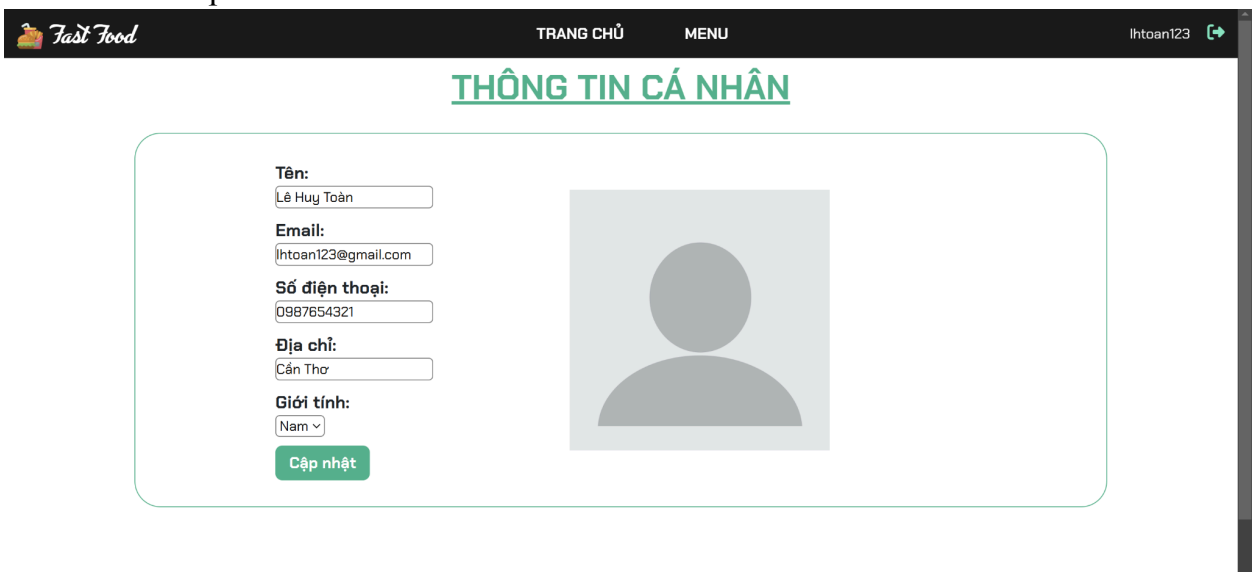
Server response

Code	Details
200	Response body


```
{
  "status": "success",
  "data": {
    "customer": {
      "name": "Lê Huy Toàn",
      "email": "toan123@example.com",
      "address": "ninh kiều cần thơ",
      "phone": "09876543213",
      "gender": "1",
      "avatar": "/public/uploads/1728819070583-674809794.jpg"
    }
  }
}
```


- **Implementation details:**
  + Libraries Used:
    + express: To build the API.
    + multer: To handle file uploads.
  + Endpoint: PUT /api/v1/customers/{id}
  + Request:

+ Body: multipart/form-data with updated customer information.
+ Response: Updated customer information.
+ This feature updates the customer information in the customers table.
+ Client-side states needed to implement this feature: The form for entering updated customer details.



7. Feature / Application page 11: Create a New Order

- **Description:** This feature allows creating a new order for a customer by providing the customer's ID. Users can submit order details (date, total price) to create a new order record in the system.
- **Screenshots:**

Request URL

```
http://localhost:3000/api/v1/cart/customers/1
```

Server response

Code | Details

201

Response body

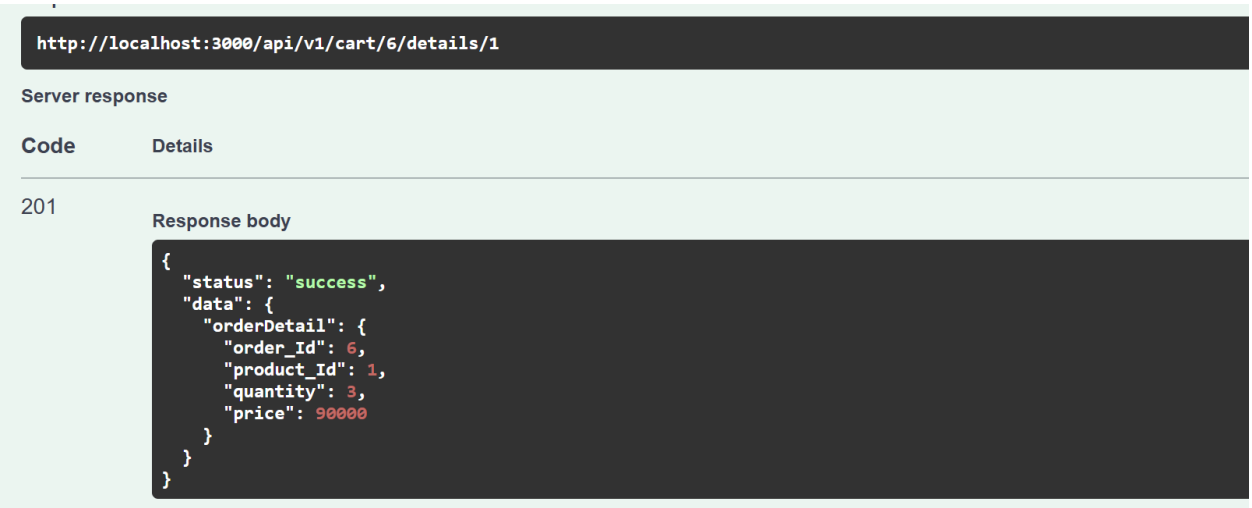

```
{
  "status": "success",
  "data": {
    "order_Id": 6,
    "order": {
      "order_Id": 6,
      "id": "1",
      "order_Date": "2024-11-15T12:52:26.200Z",
      "total_Price": 234000
    }
  }
}
```


- **Implementation details:**
  + Libraries Used:
    + express: To build the API.
  + Endpoint: POST /api/v1/cart/customers/{id}
  + Request:

        + Parameter: id (ID of the customer).
        + Body: application/json containing the order details.
    + Response: Newly created order information
    + Data is stored in the orders table, which is related to the customers table through customer_Id.
    + Client-side states needed to implement this feature: The form state to capture customer and order details for submission.

8. Feature / Application page 15: Create Order Detail

- **Description:** This feature allows users to create a new order detail. It is useful when users want to add information about a product to a specific order.
- **Screenshots:**



- **Implementation details:**
    + Libraries Used:
        + express: To build the API.
    + Endpoint: POST /api/v1/cart/{order_Id}/details/{product_Id}
    + Request:
        + Parameter:
            + order_Id: ID of the order (integer, required).
            + product_Id: ID of the product(integer, required).
        + Body: application/json containing the order detail.
    + Response: the details of the newly created order.
    + Data is stored in the Orderdetail table, which is related to the product table through product_Id and the order table through order_Id
    + Client-side states needed to implement this feature: The form state to capture customer and order details for submission.

Fast Food

TRANG CHỦ MENU

lhtoan123

DANH SÁCH SẢN PHẨM

Thông tin sản phẩm

Tìm

Burger tôm

90.000đ

Gà rán

35.000đ

Khoai tây chiên (nhỏ)

30.000đ

Chi tiết đơn hàng

SL	SP	Giá	TT
3	Khoai tây chiên (nhỏ)	30.000	90.000
1	Gà rán	35.000	35.000
1	Pizza cơ bản	95.000	95.000
1	Pizza hải sản	125.000	125.000

Tổng: 345.000 VND

Thanh toán: 345.000 VND

Thanh toán