FINAL PROJECT REPORT

SEMESTER 1, ACADEMIC YEAR: 2024-2025

CT313H: WEB TECHNOLOGIES AND SERVICES

- **Project/Application name:** Restaurant Order Website
- **GitHub links (for both frontend and backend):**

  https://github.com/24-25Sem1-Courses/ct313h01-project-congdanhctu
- **Link Youtube:** https://www.youtube.com/watch?v=wKcUYGIjN4A
- **Student ID 1:** B2111917
- **Student Name 1:** Pham Cong Danh
- **Student ID 2:** B2111934
- **Student Name 2:** Nguyen Gia Linh
- **Class/Group Number (e.g, CT313HM01):** CT313HM01

Contents

I. Introduction

Project/application description:

This application is a store management system, specifically a website for restaurants, Website will be divided into two functional groups for admin and customers.

+ **Admin**: Administrators can manage information about products, track orders, and manage users (employees, customers), manage promotions, manage desks,...
+ **Customers**: Customers can view the menu, order online, view promotional information, apply promotion code ,....

A list of tables and their structures in the database (could show a CDM/PDM diagram here).

Food Table

	#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định	Ghi chú	Thêm	Hành động
☐	1	**food_id**	int(10)		UNSIGNED	Không	*Không*		AUTO_INCREMENT	Thay đổi Xóa Thêm
☐	2	**food_name**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
☐	3	**food_price**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
☐	4	**food_discount**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
☐	5	**food_desc**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
☐	6	**food_status**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
☐	7	**food_type**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
☐	8	**food_src**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm

Orders Table

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định	Ghi chú	Thêm	Hành động
1	**order_id**	int(10)		UNSIGNED	Không	*Không*		AUTO_INCREMENT	Thay đổi Xóa Thêm
2	**user_id**	int(10)		UNSIGNED	Không	*Không*			Thay đổi Xóa Thêm
3	**table_number**	varchar(255)	utf8mb4_general_ci		Không	*Không*			Thay đổi Xóa Thêm
4	**order_total**	decimal(10,2)			Không	*Không*			Thay đổi Xóa Thêm
5	**order_status**	enum('pending', 'completed', 'cancelled')	utf8mb4_general_ci		Có	pending			Thay đổi Xóa Thêm
6	**created_at**	timestamp			Không	current_timestamp()			Thay đổi Xóa Thêm
7	**updated_at**	timestamp			Không	current_timestamp()			Thay đổi Xóa Thêm
8	**food_id**	int(10)		UNSIGNED	Không	*Không*			Thay đổi Xóa Thêm
9	**food_name**	varchar(255)	utf8mb4_general_ci		Không	*Không*			Thay đổi Xóa Thêm
10	**order_quantity**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm

User Table

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định	Ghi chú	Thêm	Hành động
1	**user_id**	int(10)		UNSIGNED	Không	*Không*		AUTO_INCREMENT	Thay đổi Xóa Thêm
2	**user_name**	varchar(255)	utf8mb4_general_ci		Không	*Không*			Thay đổi Xóa Thêm
3	**user_email**	varchar(255)	utf8mb4_general_ci		Không	*Không*			Thay đổi Xóa Thêm
4	**user_phone**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
5	**user_password**	varchar(255)	utf8mb4_general_ci		Không	*Không*			Thay đổi Xóa Thêm
6	**user_birth**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
7	**user_gender**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
8	**user_type**	int(11)			Không	1			Thay đổi Xóa Thêm

Promotion Table

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định	Ghi chú	Thêm	Hành động
1	**promo_id**	int(10)		UNSIGNED	Không	*Không*		AUTO_INCREMENT	Thay đổi Xóa Thêm
2	**promo_code**	varchar(255)	utf8mb4_general_ci		Không	*Không*			Thay đổi Xóa Thêm
3	**discount_percentage**	decimal(5,2)			Không	*Không*			Thay đổi Xóa Thêm
4	**start_date**	datetime			Không	*Không*			Thay đổi Xóa Thêm
5	**end_date**	datetime			Không	*Không*			Thay đổi Xóa Thêm
6	**status**	enum('active', 'inactive')	utf8mb4_general_ci		Có	active			Thay đổi Xóa Thêm
7	**description**	varchar(255)	utf8mb4_general_ci		Có	*NULL*			Thay đổi Xóa Thêm
8	**created_at**	timestamp			Không	current_timestamp()			Thay đổi Xóa Thêm
9	**updated_at**	timestamp			Không	current_timestamp()			Thay đổi Xóa Thêm

Table_info table

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định	Ghi chú	Thêm	Hành động
1	**table_id**	int(10)		UNSIGNED	Không	*Không*		AUTO_INCREMENT	Thay đổi Xóa Thêm
2	**table_number**	varchar(255)	utf8mb4_general_ci		Không	*Không*			Thay đổi Xóa Thêm
3	**table_status**	enum('available', 'occupied')	utf8mb4_general_ci		Có	available			Thay đổi Xóa Thêm
4	**table_capacity**	int(11)			Không	*Không*			Thay đổi Xóa Thêm
5	**created_at**	timestamp			Không	current_timestamp()			Thay đổi Xóa Thêm
6	**updated_at**	timestamp			Không	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Thay đổi Xóa Thêm

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

+ Danh is responsible for developing **authController, foodController, and orderController**. working with user authentication, food management, and order processing.
+ Linh is in charge of **tableController, promotionController, and UserController.** Linh will manage desk information, promotions, and user profiles.

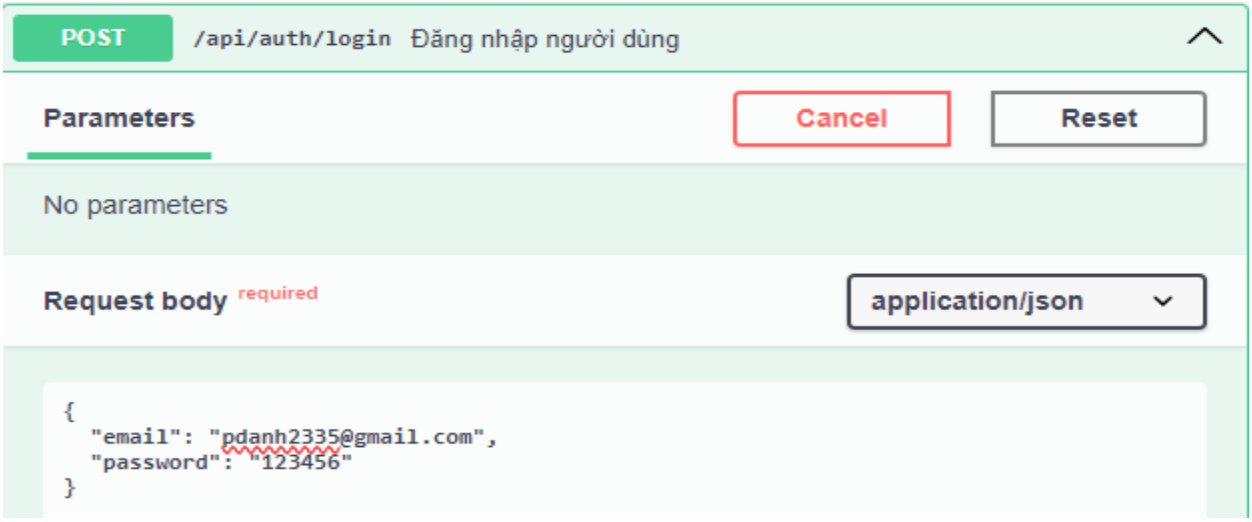
II. Details of implemented features

1. Feature / Application page 1: Log in and Register a User

- **Description:**
  + **Log in:** This feature allows registered users to enter account information (email and password) to access the system. If the information is valid, the user will be saved to the session and can access other system functions.
  + **Registration:** New users can create an account by filling in personal information (name, email, phone, password, password confirmation, date of birth, gender). After successful registration, their account will be saved in the database.

- **Screenshots:**

  User login



- **Library-use:**
  - **bcrypt:** This library is used to encrypt user passwords when signing up and to compare passwords when signing in. Bcrypt enhances security by ensuring that passwords are not saved as plain text in the database.

- **express-session**: The session management library lets users log in, saving user status on the server.
- **localStorage:** Used on the client side to save user information after login, which helps restore login status when users reload pages or switch between pages.

- **server-side APIs:**
  - **POST /login:**

    Endpoints allow users to send login requests. The server receives email and passwords, checks for valid information, and returns the user type (admin or client).

    - **Submitted data**: { email, password }
    - **Received data**: {user _ type: 1} (Client) or {user _ type: 2} (Admin)

- **Does this feature read/store data? In which table?**
  - **User Table**
    - The user _ name, user _ email, user _ password, user _ phone, user _ birth, user _ gender fields store personal information.
    - The user _ type field defines the type of user (1: customer, 2: admin).

- **Which client-side states are needed to implement this feature?**

  Vuex is a powerful solution for state management in large Vue applications.

  - **The state includes:**

- **user:** Save user login information.
- **admin:** Determines if the user is an admin.
- **localStorage:** Save user information that helps maintain login status even when the user reloads the page.

2. Feature / Application page 2: Food management application

- **Description:**

This is a food management page in the application, allowing users to view the list of dishes, add new dishes, get food details by ID and delete dishes. This feature helps manage the menu of the restaurant effectively.

- **ScreenShot:**

List Dishes

GET /api/foods Lấy danh sách món ăn

Parameters

Try it out

No parameters

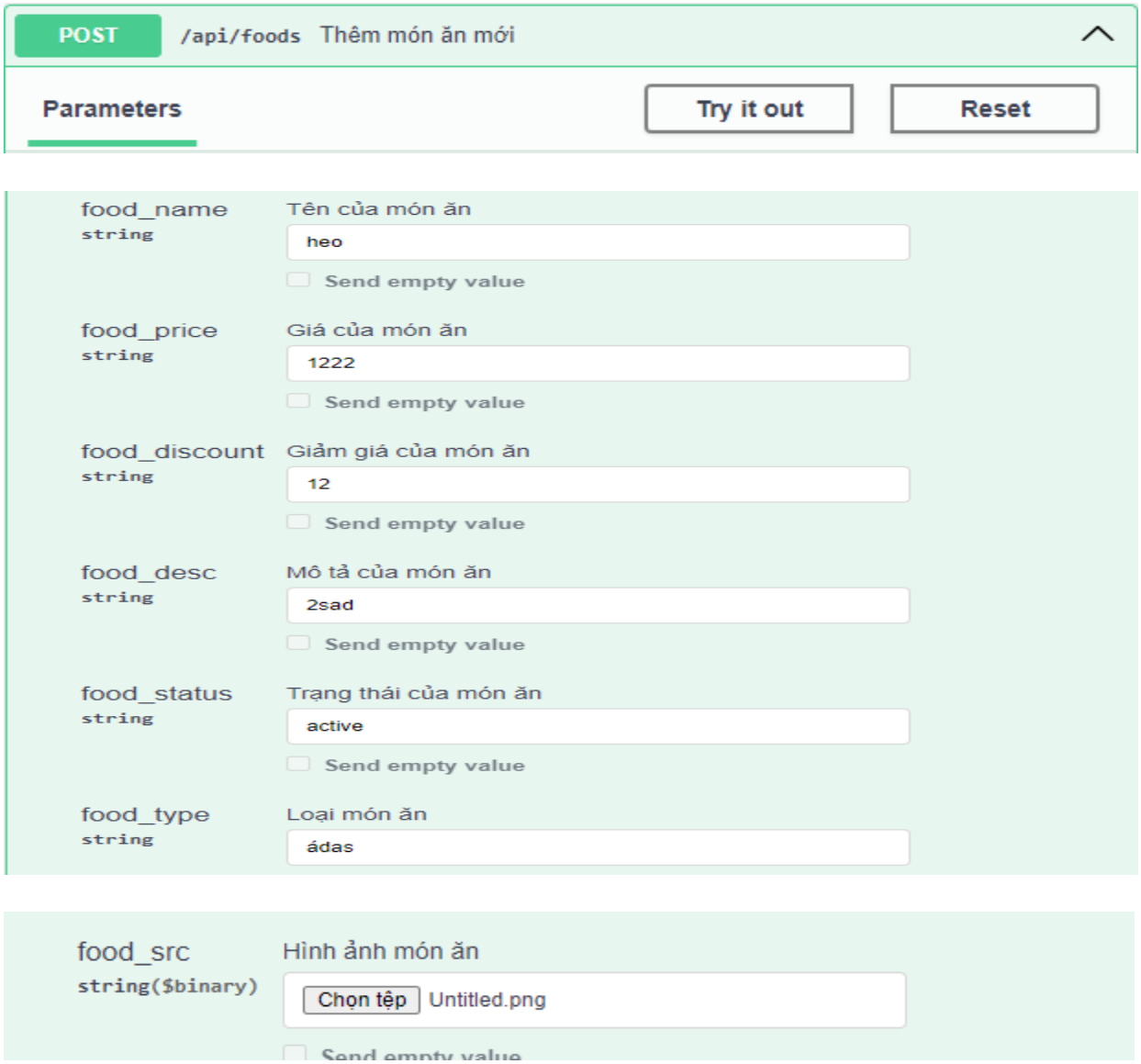
200

Response body


```
[
  {
    "food_id": 93,
    "food_name": "Banh Tieu",
    "food_price": "15000",
    "food_discount": "300",
    "food_desc": "Crispy Vietnamese donut with a sweet and fluffy text
ure.",
    "food_status": "available",
    "food_type": "snack",
    "food_src": "uploads/banhtieu.jpg"
  },
  {
    "food_id": 94,
    "food_name": "Pho Bo",
    "food_price": "40000",
    "food_discount": "5000",
    "food_desc": "Traditional beef pho with savory broth and tender no
odles.",
    "food_status": "available",
    "food_type": "main",
    "food_src": "uploads/phobo.jpg"
  },
  {
    "food_id": 95,
    "food_name": "Bun Cha",
    "food_price": "35000",
    "food_discount": "2000",
```


Download

Add Dish


POST
/api/foods
Thêm món ăn mới
Parameters
Try it out
Reset
food_name
string
Tên của món ăn
heo
Send empty value
food_price
string
Giá của món ăn
1222
Send empty value
food_discount
string
Giảm giá của món ăn
12
Send empty value
food_desc
string
Mô tả của món ăn
2sad
Send empty value
food_status
string
Trạng thái của món ăn
active
Send empty value
food_type
string
Loại món ăn
ádas
food_src
string($binary)
Hình ảnh món ăn
Chọn tệp
Untitled.png

201

Response body

```
{
  "message": "Food created successfully.",
  "foodId": 116
}
```

Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://localhost:8080
connection: keep-alive
content-length: 53
content-type: application/json; charset=utf-8
date: Wed,09 Oct 2024 03:41:52 GMT
etag: W/"35-jydTi6xsroirwIP7Te30CadePT4"
keep-alive: timeout=5
vary: Origin
x-powered-by: Express
```

- **Library use:**
  + **Multer:** This library is used to handle uploading food image files.

- **Server-side APIs:**
  + **GET /api/foods**
    - **Description:** Get a list of all the dishes.
    - **Return data:** Array of food objects, each containing information such as ***food _ id, food _ name, food _ price, food _ discount, food _ desc, food _ status, food _ type, and food _ src.***
  + **POST /api/foods**
    - **Description:** Add new dish.
    - **Sending Data:** The information about the dish includes food _ name, food _ price, food _ discount, food _ desc, food _ status, food _ type, and food _ src image.
    - **Data returned:** ID of newly created dish and successful notification

- **Data Handling:**
  + **Data Storage:** The food data is stored in the food table in the database.

- **Client-side States Needed:**
  + **foodObj**:
    - **Description:** Data entered from the search box to filter dishes by name.
    - **Role:** Filter the list of dishes by keyword.

  + **selectedType**:

    - **Description:** Select the type of dish to filter. Default value: 'All'.
    - **Role**: Filter the list of dishes by type (snack, main, dessert...).
  + **foods:**
    - **Description**: The list of dishes loaded from the API.
    - **Role**: Provide source data for display.
  + **filteredFoods (computed):**
    - **Description**: The list of dishes has been filtered based on selectedType and foodObj.name.
    - **Role**: Combining filters and searches to show only the right dishes.
  + **paginatedFoods (computed):**
    - **Description**: The list of dishes to display on the current page.
    - **Role:** Support pagination.
  + **itemsPerPage:**
    - **Description**: The number of dishes displayed per page. Default value: 6.
    - **Role**: Configure a limit to the number of dishes on a page.

+ **totalPages (computed):**
    - **Description**: The total number of pages is calculated based on the number of filtered dishes and itemsPerPage.
    - **Role**: Provides page numbers to control pagination.

3. Feature / Application page 3: Order Management Feature

- **Description:**

Feature 3 focuses on managing customer orders in a restaurant setting, including food items, tables, and users. It provides API endpoints for CRUD operations on orders, such as listing all orders, creating new orders, fetching a specific order by ID, and deleting orders. The system also handles table statuses dynamically, updating them based on order creation.

- **ScreenShot:**

  - **Create new order**

POST /api/order Tạo đơn hàng mới

Parameters

Cancel Reset

No parameters

Request body required

application/json

```
{
  "food_id": 20,
  "food_name": "Pho Bo",
  "order_quantity": 2,
  "table_number": "2",
  "order_total": 2,
  "user_id": 75,
  "order_status": "pending"
}
```

Curl

```
curl -X 'POST' \
  'http://localhost:5000/api/order' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "food_id": 20,
  "food_name": "Pho Bo",
  "order_quantity": 2,
  "table_number": "2",
  "order_total": 2,
  "user_id": 75,
  "order_status": "pending"
}'
```

Request URL

```
http://localhost:5000/api/order
```

Code	Details
201	Response body `{ "message": "Order created successfully" }` Download

- **Server-side api:**
  - **POST /api/order**
    - **Description:** Create a new order.
    - **Request Data:**
      - **food_id (integer):** ID of the food item.
      - **food_name (string):** Name of the food item.
      - **order_quantity (integer):** Number of food items ordered.
      - **table_number (string):** Table number.
      - **order_total (float):** Total cost of the order.
      - **user_id (integer**): ID of the customer.
      - **order_status (string):** Default value is "pending".
    - **Response:**
      - **201:** Order created successfully.
      - **400:** Invalid data provided.
      - **500:** Error creating order.

- **Does this feature read/store data? In which table?**

  - **Read data in "Table" table:**
    - **Purpose:** Checks the current status of the selected table to confirm if it's available for assignment.

- **Action:** Likely involves reading the table_status column of the tables table to determine if the table is "available".

- **Store Data In Order table:**
  - **Purpose:** Saves the details of the new order into the database.
  - **Action:** Inserts a new record into the orders table.

- **Which client-side states are needed to implement this feature?**

❖ **Food Details**

- **State: foodDetails**
- **Type: ref({})**
- **Purpose:** Stores information about the selected food item (e.g., name, description, price, discount, and image source).

❖ **Order Quantity**

- **State**: order_quantity
- **Type**: ref(1)
- **Purpose**: Tracks the number of units the user wants to order.

❖ **Available Tables**

- **State**: availableTables
- **Type**: ref([])
- **Purpose**: Stores the list of available tables that the user can select from.

❖ **Selected Table**

- **State**: selectedTable
- **Type**: ref(null)
- **Purpose**: Stores the table selected by the user for the order.

- **Promotion Details**
  - **State:** **promotion**
  - **Type:** **ref({ valid: false, discount_percentage: 0 })**
  - **Purpose:** Stores the status of the promo code and the discount percentage.
- **Promo Message**
  - **State:** **promoMessage**
  - **Type:** **ref("")**
  - **Purpose: Displays messages about the promo code's validity (e.g., success or error).**
- **Total Price**
  - **State:** **totalPrice**
  - **Type:** **computed**
  - **Purpose:** Calculates the total cost of the order before applying the promo discount.
- **Final Total Price**
  - **State:** **finalTotalPrice**
  - **Type:** **computed**
  - **Purpose:** Calculates the total cost of the order after applying the promo discount.

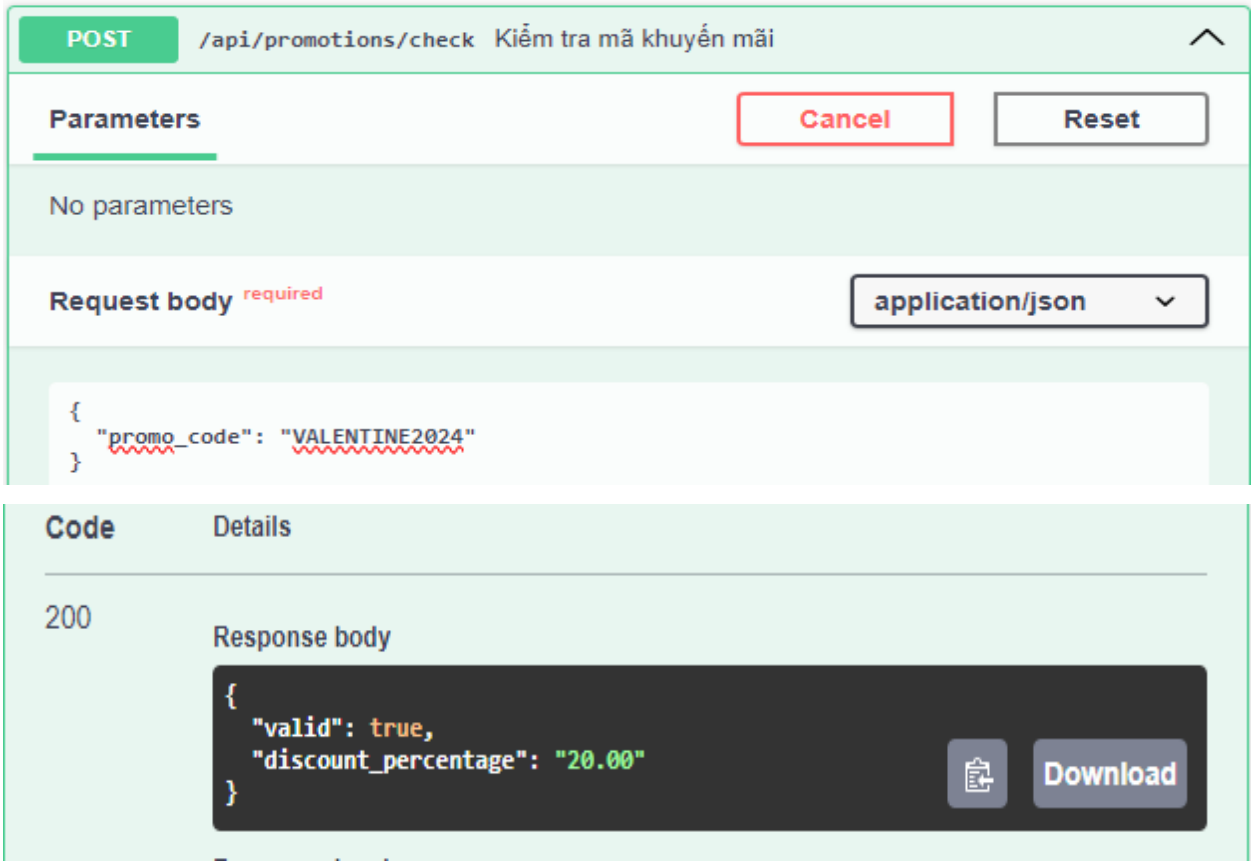
4. Feature / Application page 4: Promotions Management Page

- **Description:**

The Promotions Management Page is used for managing promotional campaigns for a business. It provides functionalities for creating, retrieving, updating, and deleting promotion campaigns, as well as checking the validity of promotional codes. The promotions contain details such as the promo code, discount percentage, start and end dates, and status (active/inactive).

- **ScreenShot:**

Check a promotion



Example Value | Schema

```
{
  "valid": true,
  "discount_percentage": 0
}
```

400	Mã khuyến mãi hết hạn hoặc không hợp lệ	*No links*

Media type

application/json

Example Value | Schema

```
{
  "valid": false,
  "message": "string"
}
```

404	Không tìm thấy chương trình khuyến mãi	*No links*

Media type

application/json

Example Value | Schema

```
{
  "valid": false,
  "message": "string"
}
```

500	Lỗi máy chủ	*No links*

- **Server-side APIs:**
  - **POST /api/promotions/check**
    - **Purpose:** Checks if a promo code is valid and within its active date range**.**
    - **Data Format (Request):**

```
{
  "valid": true,
  "discount_percentage": "20.00"
}
```

    - **Data Format (Response):**

```
{
  "valid": true,
  "discount_percentage": "20.00"
}
```

- **Does this feature read/store data? In which table?**
  - **Read data**:
    - Query the promotions table to check if the promo code exists.
    - Get the details of the promo code, including:
      - Start time (start _ date).
      - End time (end _ date).
      - Discount percentage (discount _ percentage).
- **Client-side states are implemented this feature:**
  - **State Variable:** promotion
    - Sub-properties:
      - promotion.value.valid: Tracks whether the promotion code is valid or invalid.
      - promotion.value.discount_percentage: Stores the percentage discount for valid promotion codes
  - **Purpose:**
    - Indicates if the promo code is valid and how much discount it provides.

5. Feature / Application page 5: Table Management Page

- **Description:**

The Table Management Application is used to manage the tables in a restaurant. It allows users to create, retrieve, update, and delete tables in the system. Each table has attributes like table_number, table_status, and

table_capacity. The application is essential for managing seating arrangements and table availability in real-time.

- **ScreenShot:**

  - **Update a table:**

Code	Description	Links

200	Cập nhật bàn thành công
	Media type
	application/json
	Controls Accept header.
	Example Value | Schema
	{
	"message": "Table updated successfully"
	}
		No links

404	Không tìm thấy bàn	No links
500	Lỗi máy chủ	No links

Server-Side APIs:

- **PUT /api/table/{id}:**
  + **Purpose**: Updates a table's details
  + **Request Format**

```
{
  "table_number": "A2",
  "table_status": "available",
  "table_capacity": 6
}
```

  + **Response format:**

```
{
  "message": "Table updated successfully"
```

}

- **Data Storage (Database Table):**

  This function performs both read and write data.

  - **Data Reading Function:**
    - Query to determine if a table exists based on table _ id.
  - **Data Recording Function:**
    - **Write new data to table _ info, including the columns:**
      - **table _ number:** Table number.
      - **table _ status:** Table status (e.g. occupied, available).
      - **table _ capacity:** Table capacity.

- **Client-side states:**
  - **Table Data States:** currentTable
    - **Type**: ref ({})
    - **Related**: Contains the data of the currently edited table, including table _ number, table _ status, and table _ capacity.
    - **Role**: This data is sent to the API when the table is updated.
  - **Editing States:** isEditing
    - **Type:** ref (false)
    - **Related**: Defines the current state of the editing interface.
    - **Role:** Controls the display of the editing modal and allows the user to perform updates.