

FINAL PROJECT REPORT

SEMESTER 1, ACADEMIC YEAR: 2024-2025

CT313H: WEB TECHNOLOGIES AND SERVICES

Project/Application name: Pizza Website

GitHub links (for both frontend and backend):

<https://github.com/24-25Sem1-Courses/ct313h03-project-LamSut>

Youtube link:

📺 Pizza Gout - Truong Dang Truc Lam B2111933 (ft. Le Xuan Thanh B21119...

Student ID 1: B2111933

Student Name 1: Truong Dang Truc Lam

Student ID 2: B2111952

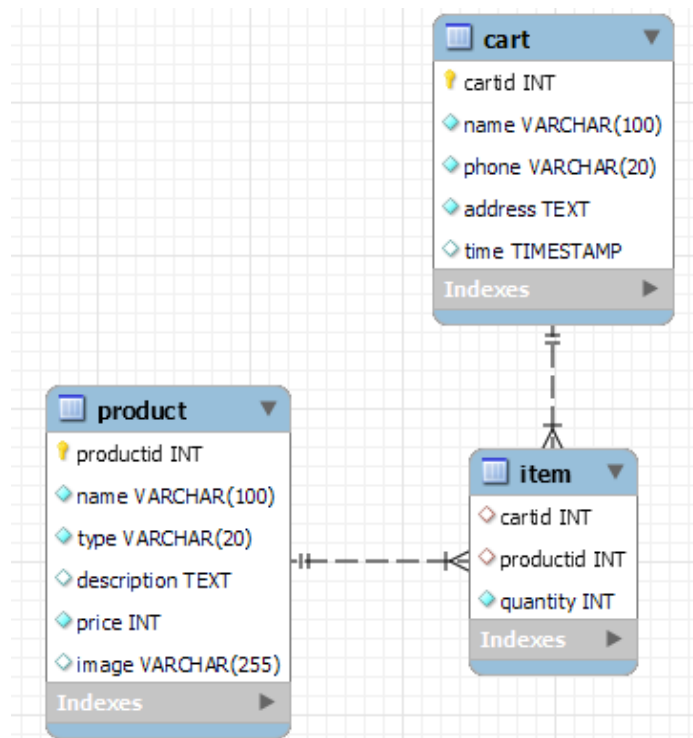
Student Name 2: Le Xuan Thanh

Class/Group Number (e.g, CT313HM01): CT313HM03

I. Introduction

Project/application description: A single page application (SPA) for ordering pizzas, drinks and appetizers. Users can enter their personal information, select their desired products from a predefined menu and finalize their order.

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



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

No.	Task Description	Member
1	Organize structure, design database, initialize the API server and SPA for the project	Truong Dang Truc Lam
2	Implement REST API for CRUD operations and document them with OpenAPI (Swagger)	Truong Dang Truc Lam Le Xuan Thanh
3	Create functions to get data from the server & manage server state with Tanstack Vue Query	Truong Dang Truc Lam
4	Manage sessions for the CartID created by each user, using Express Session on the server-side & Pinia (Persistedstate Plugin) for client-side state	Truong Dang Truc Lam
5	Implement client-side state management for Items of current Cart with Pinia (Persistedstate Plugin)	Le Xuan Thanh
6	Client-side routing with Vue Router	Le Xuan Thanh
7	Design responsive UI for Vue components with Bootstrap : Product List, Cart Items, Pagination, Search Bar, Information Form,...	Truong Dang Truc Lam Le Xuan Thanh
8	Implement Vue pages: MainPage, InformationPage, MenuPage, CheckOrderPage, NotFound	Truong Dang Truc Lam Le Xuan Thanh
9	Build and deploy the SPA with Caddy , deploy the backend API with PM2	Truong Dang Truc Lam

II. Details of implemented features

1. View the menu:

Description: Users can find their desired products from this predefined menu. This menu features a responsive Bootstrap UI with pagination and search, users can also select a product for viewing its details.

Screenshots:



PIZZAGOUT

Search



Bacon Pizza

\$8

+ Add



Chicken Pizza

\$7

+ Add



Seafood Pizza

\$8

+ Add



Veggie Pizza

\$6.5

+ Add



Coca-Cola

\$0.5

+ Add



Pepsi

\$0.5

+ Add



Fries

\$1.5

+ Add



Prawn Rice

\$2

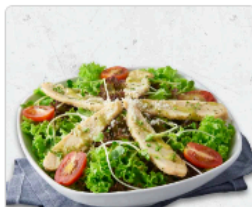
+ Add

← Previous 1 2 3 Next →



PIZZAGOUT

A



Chicken Salad

\$2.5

+ Add



Shrimp Salad

\$2.5

+ Add

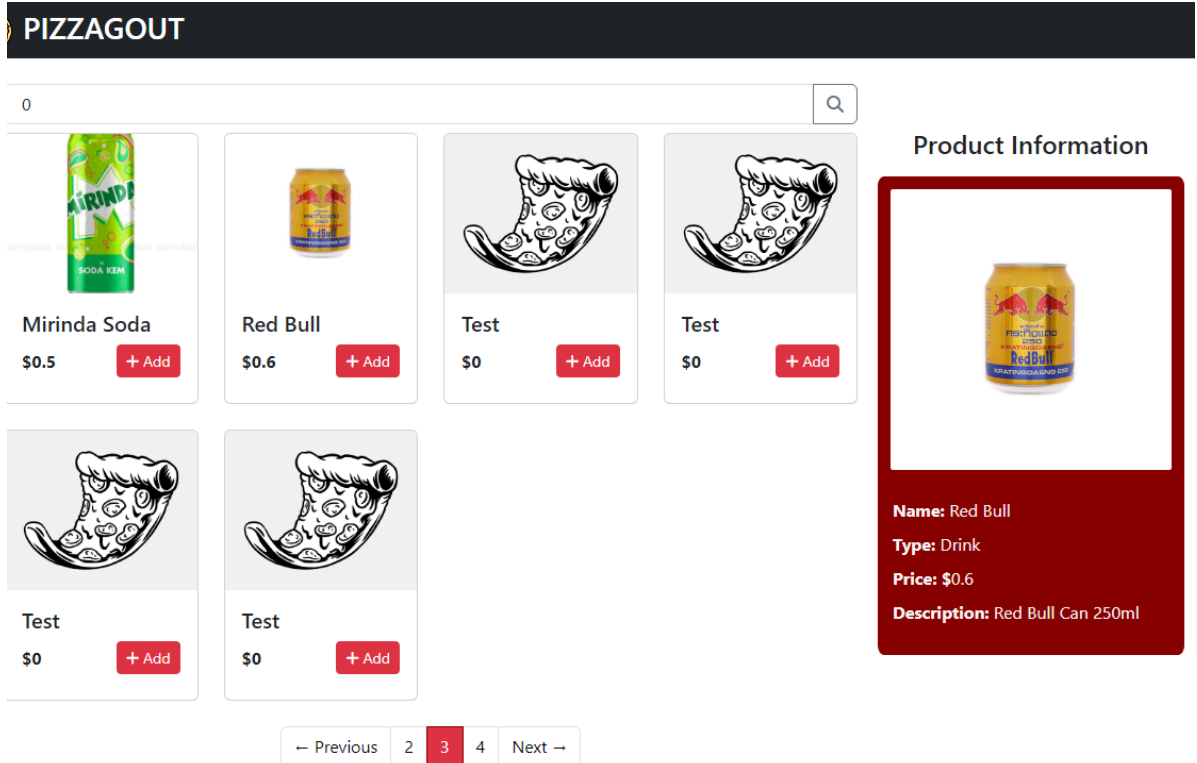


Aquafina

\$0.3

+ Add

← Previous 1 2 3 Next →



Implementation details:

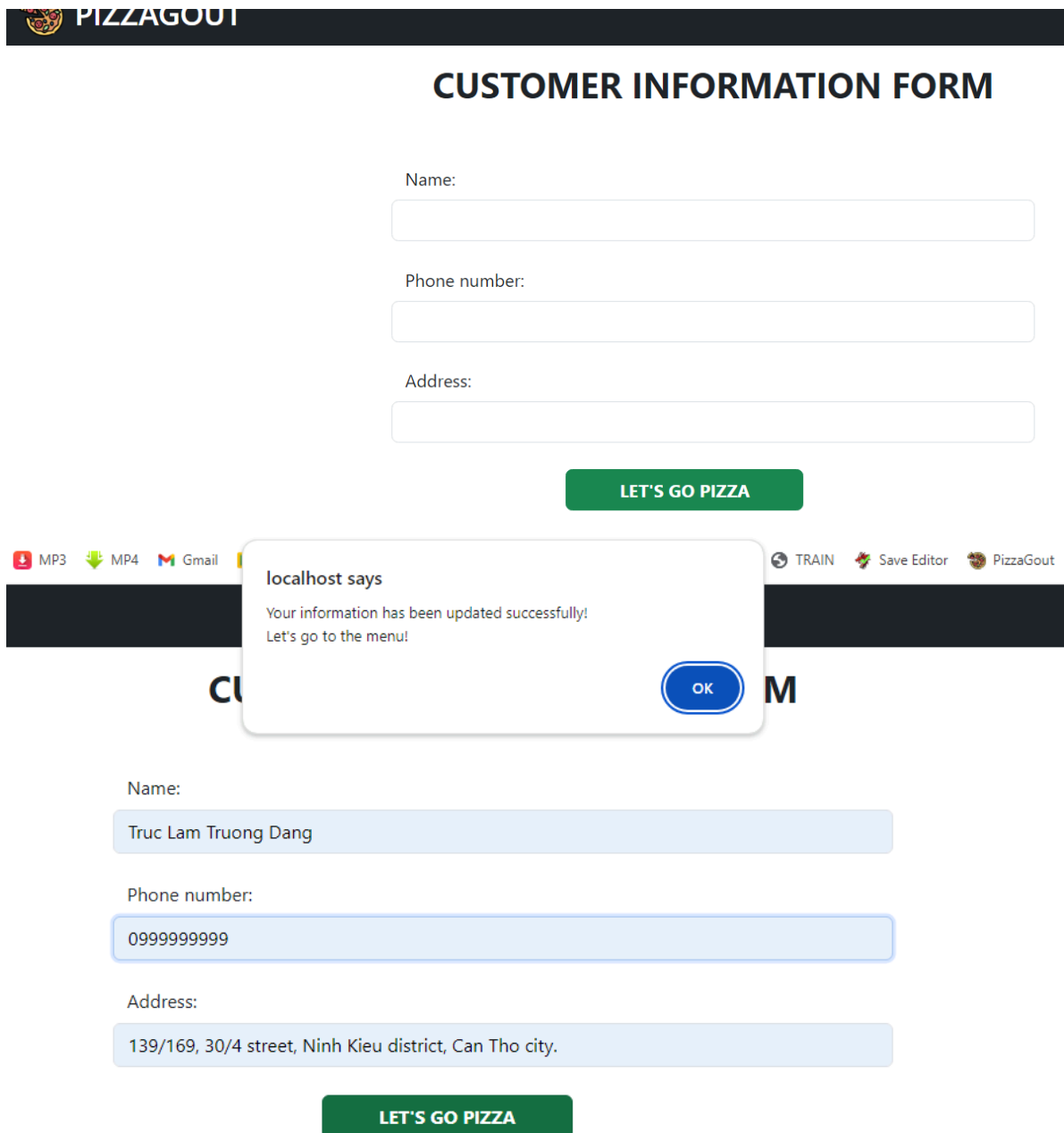
- + The “Add” button will use Pinia with Persistedstate Plugin for client-state, which references the feature “Order products from the menu”.
- + Describe the API:
 - List products with filters (limit, page):
 - Endpoint: `http://localhost:3000/api/v1/product`.
 - GET request with these query parameters:
 - o name: The name of the product.
 - o type: The type of the product.
 - o limit: The number of products to return per page.
 - o page: The current page number.
 - The response is a JSON object with the following structure:
 - o The products array contains the products that match the search criteria.
 - o The metadata object contains information about the pagination, such as the total number of records, the current page number, the number of records per page.
- + This feature reads information of each product from the list that gets from the Product table based on specified filters.

2. Enter user information for the cart

Description: Users can fill this form and submit their information to their cart, each cart contains a CartID.

- If there is no stored CartID (no cart exists), this feature will display a blank form, then users can create a new cart and generate a CartID for their cart.
- If their CartID has been stored, this feature will display a form with filled information (the old information that user has submitted), then users can modify and submit to update their information on the existing cart.

Screenshots:



The screenshot displays the 'CUSTOMER INFORMATION FORM' for 'PIZZA GOUT'. The form includes input fields for 'Name:', 'Phone number:', and 'Address:'. A green 'LET'S GO PIZZA' button is positioned below the form. A modal dialog box is overlaid on the form, titled 'localhost says', with the message 'Your information has been updated successfully! Let's go to the menu!' and an 'OK' button. The background of the form is dark, and the text is white. The modal dialog is white with a blue border and a blue 'OK' button. The form fields are light blue with white text. The 'Phone number:' field contains the value '0999999999'. The 'Address:' field contains the value '139/169, 30/4 street, Ninh Kieu district, Can Tho city.'.

PIZZA GOUT

CUSTOMER INFORMATION FORM

Name:

Phone number:

Address:

LET'S GO PIZZA

localhost says

Your information has been updated successfully!
Let's go to the menu!

OK

Name:

Truc Lam Truong Dang

Phone number:

0999999999

Address:

139/169, 30/4 street, Ninh Kieu district, Can Tho city.

LET'S GO PIZZA

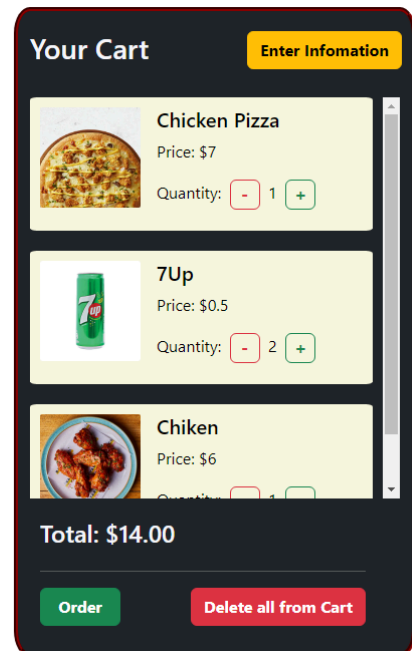
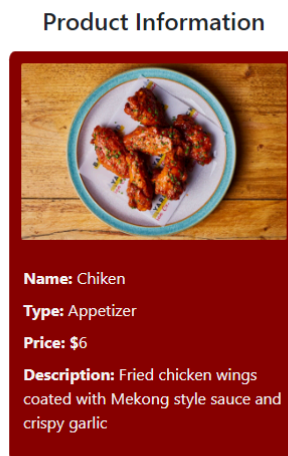
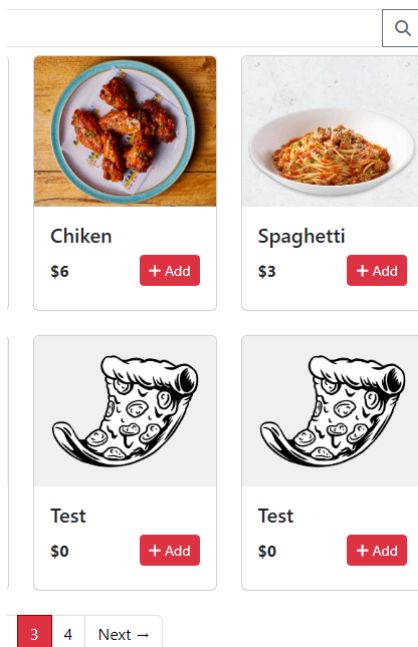
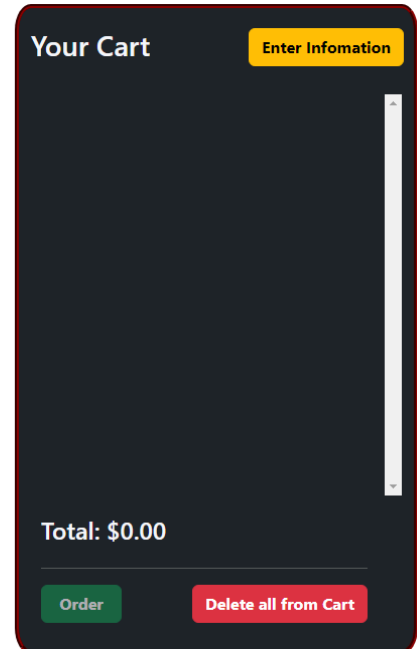
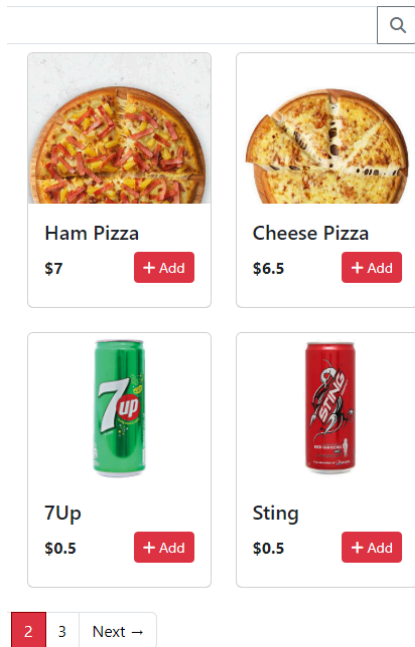
Implementation details:

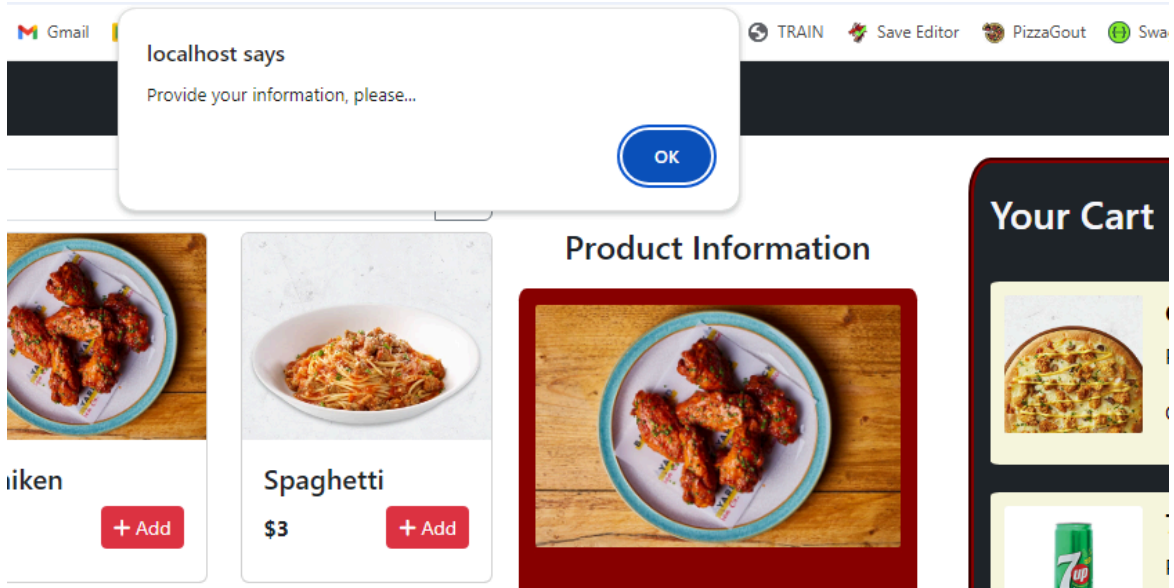
- + This feature manages sessions for the CartID created by each user, using Express Session on the server-side & Pinia (Persistedstate Plugin) for client-side state. Users can reuse their old cart with their provided information, they don't need to fill their information from scratch again (create a new cart) each time they get access to this website.
- + Describe the APIs:
 - If no CartID has been stored:
 - Enter information for a new cart:
 - o Endpoint: `http://localhost:3000/api/v1/cart`.
 - o POST request that requires the information of the user
The data is sent in a multipart/form-data format.
 - o The response is a JSON object with the data that contains detailed information of the newly created cart.
 - o This API also setup session for CartID
(`req.session.cartId` = CartID of the created cart)
 - If there is a stored CartID:
 - Get cart information:
 - o Endpoint: `http://localhost:3000/api/v1/cart/:cartId`.
 - o GET request that requires only ID of cart (`cartId`).
 - o The response is a JSON object with the data that contains information about the cart, which has the same `cartId` with input.
 - Update cart information:
 - o Endpoint: `http://localhost:3000/api/v1/cart/:cartId`.
 - o PUT request that requires ID of cart (`cartId`), updated information of user (name, phone, address).
 - o The data is sent in a multipart/form-data.
 - o The response is a JSON object with the updated data that contains information about the cart, which has the same `cartId` with input.
- + If no CartID has been stored, this feature will write (create) new cart data to the Cart table based on the provided information.
- + If there is a stored CartID, this feature will read information of users from the Cart table based on CartID, then it will autofill the form for them. If users have any modifications on their information, they will write (update) their new cart information to the Cart table.
- + Pinia with Persistedstate Plugin will store the CartID of the created cart in localStorage, users can reuse their old cart even if they have accidentally closed the browser.

3. Order products from the menu

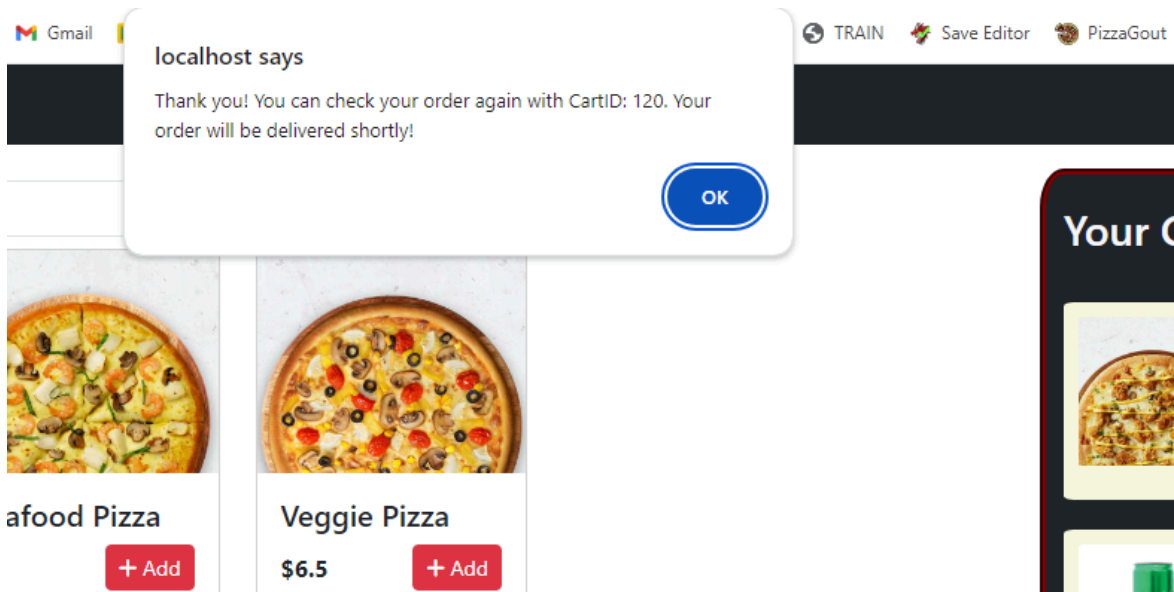
Description: When users click on the “Add” button of a product from the menu, it will be displayed in the cart on the right side. Then users can increase or decrease the quantity of each item (product) in the cart, they can also delete each item or clear all items from the cart. After everything is done, they can submit their order. An alert window will display the CartID, which can be used to check their order.

Screenshots:





If user didn't provide your information, redirect to InformationPage



Users can use this CartID to check their order again

Implementation details:

- + This feature is affected by the CartID that uses Express Session on the server-side & Pinia (Persistedstate Plugin) for client-side state. If there is no stored CartID (users didn't provide their information), users cannot submit their order, they will be redirected to InformationPage instead.
- + This feature also stores items in the cart with Pinia (Persistedstate Plugin).

- + After users confirm their order, CartID will be removed from the session. All client-side states will be cleared too, which means the data of CartID and items in the cart will be removed from localStorage.
- + Describe the APIs:
 - Confirm an order (insert rows to Item table):
 - Endpoint:
http://localhost:3000/api/v1/cart/:cartId/product/:productId
 - POST request that requires the following information:
 - o cartId: the ID of the cart which you want to add.
 - o productId: the ID of the product that you want to add.
 - o quantity: quantity of item.
 - The data is sent in a multipart/form-data format.
 - The response is a JSON object with the data that contains detailed information of the newly created item.
 - Clear CartID from session:
 - Endpoint: http://localhost:3000/api/v1/clear
 - GET request with no body is required.
 - The server-side session's CartID property will be set to null (req.session.cartId = null)
 - The response is a JSON object with the data that contains the value of CartID (null)
- + Upon clicking the “Order” button, this feature will write (create) new data of items to the Item table based on the provided information (CartID, ProductID, quantity).
- + Users can add items to cart, delete items from cart, increase or decrease the quantity of each item with Pinia (Persistedstate Plugin).
 - The data of each item in the cart will be stored in localStorage, users can continue their ordering process even if they have accidentally closed the browser.
 - The data that is stored in localStorage will be cleared if users have confirmed their order. After that they have to create a new cart for the next order (provide information & add items to cart).

4. Check order again

Description: With the provided CartID from the alert after every order, users can use that CartID to check their order again.

Screenshots:




YOUR ORDER

No items in the cart.


If the order does not exist




YOUR ORDER



Chicken Pizza
Price: \$7
Quantity: 1
Subtotal: \$7



7Up
Price: \$0.5
Quantity: 2
Subtotal: \$1



Chicken
Price: \$6
Quantity: 1
Subtotal: \$6

Total: \$14

BACK

Implementation details:

- + Describe the API:
 - Get all items in cart
 - Endpoint: `http://localhost:3000/api/v1/cart/:cartId/product`.
 - GET request with only `cartId` as query parameter.
 - The response is a JSON object with: The items array, which exists in the selected cart, contains information of each item (`CartID`, `ProductID`, product name, quantity, price and image).
- + This feature reads information of each item from the list of items which get from the Item table based on specified `CartID`.

Note:

- The app always checks if there is an existing `CartID` (in both server-side session or client-side `localStorage`). If there is no stored `CartID`, it will make an API request for fetching `CartID` in server-side session.
- Describe the API:
 - Endpoint: `http://localhost:3000/api/v1/`
 - GET request with no body is required.
 - The response is a JSON object with the data that contains the value of `CartID` which stored in session (`req.session.cartId`)
 - If a cart has been created by the user, the value is the `CartID` of that created cart (refer to “Enter information for a new cart” API).
 - Otherwise the value is set to null by default.
- If successful, this `CartID` will be stored in `localStorage` with `Pinia Persistedstate Plugin`. With that, this application doesn't have to fetch `CartID` in the server-side session with everytime refresh.