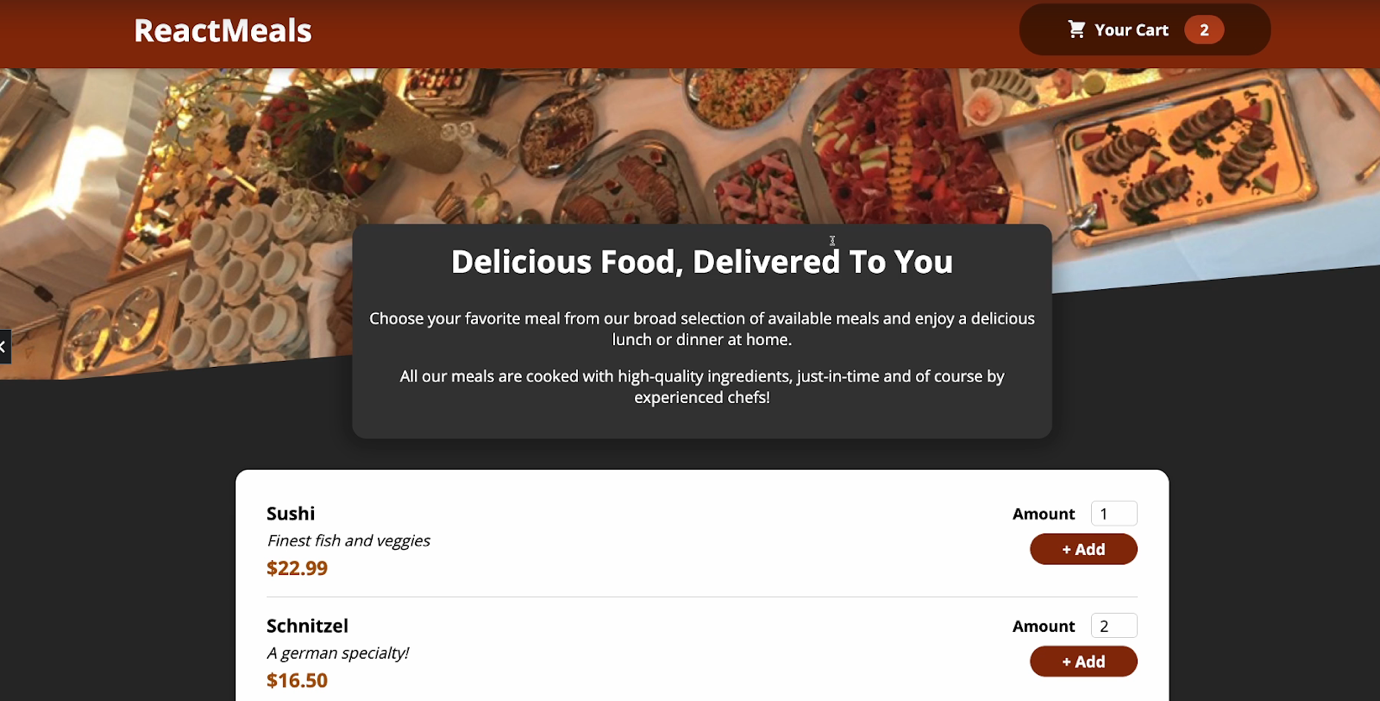
**Lab 9.1. Tạo ứng dụng đặt đồ ăn**

Ở bài Lab này, bạn sẽ được áp dụng các kiến thức đã được học để xây dựng một dự án thực tế, đó là tạo giao diện cho ứng dụng đặt đồ ăn như sau:



Để hoàn thiện được bài Lab này, bạn sẽ cần tải các Resource ở [**link sau**](https://drive.google.com/file/d/14uwvtpJyq1-dr51kH_8t2jit3tWVUTLd/view?usp=sharing). Các Resource này sẽ bao gồm 2 Component là Button và Card để bạn sử dụng cho việc tạo giao diện. Đồng thời Resource cũng cung cấp các file .css để tạo kiểu cho Website.

**1. Tạo Component Header**

Đầu tiên, bạn sẽ tạo Component Header.js cho dự án, Phần Header này sẽ có giao diện như sau:



Ngoài ra, Component này sẽ có thêm một Component con nữa là **HeaderCartButton.js** để hiển thị danh sách số lượng sản phẩm trong giỏ hàng, đồng thời khi click vào Component này cũng sẽ hiển thị ra giỏ hàng (bạn sẽ xử lý phần này ở các bước sau).

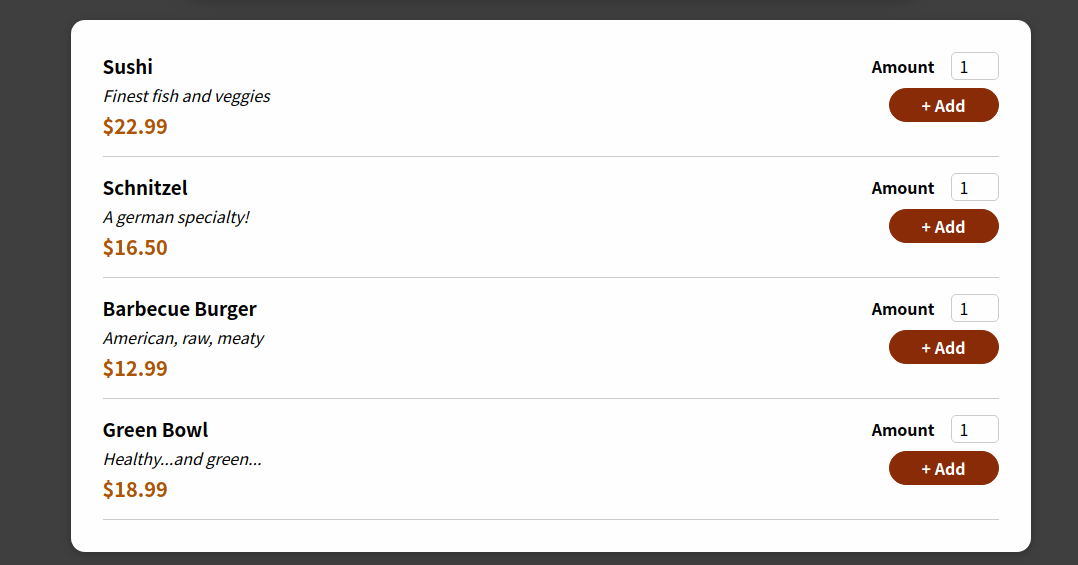
**2. Hiển thị danh sách các Meal**

Tiếp theo, bạn sẽ tạo Component Meals.js  để hiển thị ra danh sách các món ăn có thể Oder, Component này sẽ chứa 2 Component con nữa như sau

Component MealsSummary.js để hiển thị đoạn văn bản như sau:



Component **AvailableMeals.js** để hiển thị ra danh sách các món ăn có thể order với giao diện như sau:



**Lưu ý**: Bạn sẽ sử dụng Dummy Data sau tượng trưng cho danh sách của các món ăn:

|  |
| --- |
| const DUMMY\_MEALS = [  {    id: 'm1',    name: 'Sushi',    **description**: 'Finest fish and veggies',    price: 22.99,  },  {    id: 'm2',    name: 'Schnitzel',    **description**: 'A german specialty!',    price: 16.5,  },  {    id: 'm3',    name: 'Barbecue Burger',    **description**: 'American, raw, meaty',    price: 12.99,  },  {    id: 'm4',    name: 'Green Bowl',    **description**: 'Healthy...and green...',    price: 18.99,  },   ]; |

Bạn sẽ hiển thị danh sách các món ăn, bạn cũng có thể tạo ra 2 Component con nữa như sau:

* **MealItem.js**: Hiển thị thông tin của một món ăn.
* **MealItemForm.js**: Hiển thị Form để có thể thêm món ăn đó vào giỏ hàng. Chúng ta sẽ xử lý sự kiện này ở các bước sau.

**3. Tạo CartContext**

Để quản lý danh sách các món ăn đã được thêm vào giỏ hàng, bạn sẽ tạo một Context mới là CartContext, Context này sẽ gồm các thuộc tính như sau:

|  |
| --- |
| const CartContext = React.createContext({   items: [],   totalAmount: 0,   addItem: (item) => {},   removeItem: (id) => {} }); |

Đồng thời, bạn cũng cần tạo một **CartProvider** để quản lý context này.

**4. Xử lý sự kiện Cart sử dụng useReducer**

Đầu tiên, bạn sẽ cần tạo một Reducer để có thể thay đổi các trạng thái của context (sử dụng useReducer hook), bạn sẽ có 2 Action Type như sau:

1. **ADD**:

Lúc này, bạn sẽ thêm món ăn vào trong State **items** của CartContext, sau đó tính lại **totalAmount** của tất cả đồ ăn và cập nhật các State tương ứng vào CartContext. Thông tin của món ăn sẽ nằm ở trường **action.item** truyền vào.

**Lưu ý**: Bạn cũng sẽ cần kiểm tra xem món ăn ăn đó đã tồn tại trong **items** chưa (có phần tử nào trùng id không), nếu có thì hãy cập nhật số lượng tương ứng chứ không phải add thêm một phần tử mới vào **items**.

1. **DELETE**

Đây là Action dùng để xóa đồ ăn có id tương ứng với action.id được truyền vào. Mỗi lần Action này được gọi thì bạn sẽ trừ số lượng của món ăn đó đi 1. Sau đó tính lại **totalAmount** và cập nhật các State tương ứng vào CartContext.

**Lưu ý:** Nếu như số lượng của món ăn đó bị giảm về 0. Bạn sẽ cần xóa món ăn đó khỏi **items**.

Đồng thời, bạn cũng cần cập nhật hai hàm addItem và removeItem ở CartContext để dispatch được đúng Action tương ứng.

|  |
| --- |
| **const** addItemToCartHandler = (item) => {  dispatchCartAction({ **type**: 'ADD', item: item }); };  **const** removeItemFromCartHandler = (id) => {  dispatchCartAction({ **type**: 'REMOVE', id: id }); }; |

**5. Thêm sự kiện vào DOM**

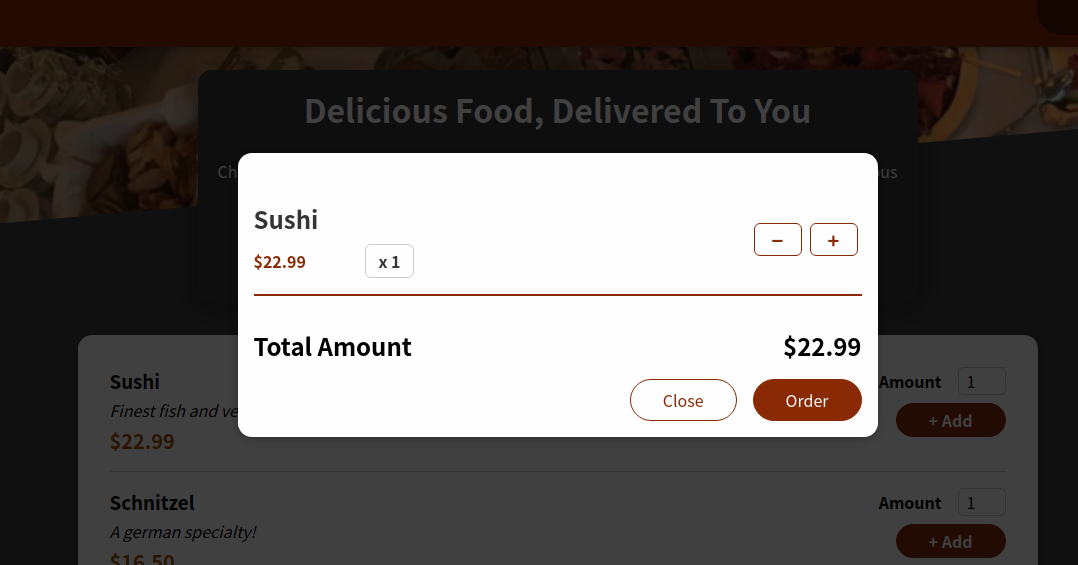
Tiếp theo, bạn sẽ thêm các sự kiện để khi người dùng click vào nút Add thì sẽ thêm các món ăn đã chọn vào giỏ hàng.

**Lưu ý:** Ở MealItemForm.js, bạn có thể sử dụng ref để lấy được các input tương ứng với số lượng item muốn add. Sau đó sẽ gọi hàm addItem đã có ở context để thêm vào giỏ hàng.

Tiếp theo ở **HeaderCartButton**, bạn sẽ cần sử dụng **useEffect** để xử lý mỗi khi state items của CartContext thay đổi (được thêm / xóa món ăn) thì sẽ cập nhật lại số hiển thị tương ứng.

**5. Hiện thị Cart**

Tiếp theo, bạn sẽ tạo chức năng hiển thị giỏ hàng cho người dùng. Giỏ hàng sẽ được hiển thị mỗi khi nhấn vào **HeaderCartButton.** Bạn sẽ cần tạo thêm một Component mới là **Cart.js** tượng trưng cho giỏ hàng. Component này sẽ là một Modal với giao diện như sau:



Các dữ liệu ở Modal này sẽ được lấy từ CartContext. Khi nhấn vào nút “Close" thì sẽ đóng Modal.

**Lưu ý:**  Bạn sẽ cần sử dụng **React Portal** để điều chỉnh vị trí của Modal ra hẳn bên ngoài Container App.

**6. Xử lý sự kiện Remove**

Cuối cùng, bạn sẽ cần xử lý sự kiện khi người dùng xóa một món ăn trong Cart và gọi hàm deleteItem từ CartContext với dữ liệu tương ứng.