Name: Khurram Shahzad

Roll number: 00116699

Class Slot: Friday (9-12)

Teacher: Sir Hamzah Syed

Technical Foundation Plan for Q-Commerce Website (Restaurant-Based) (Tempelate-9)

High-Level Architecture Diagram Description

1. Frontend (Next.js)

- Positioned at the top layer of the diagram.
- Represents the user interface (UI) and user experience (UX) of your Q-commerce website.

Pages to Include:

1. **Home Page**
   - Overview of the restaurant.
   - Highlight popular dishes and categories.
   - Include search functionality.
2. **Product Listing Page**
   - Display a list of menu items or products.
   - Filter and sort functionality (e.g., by cuisine, price, ratings).
3. **Product Detail Page**
   - Dynamic routing for each product.
   - Details of the menu item, including images, description, price, and availability.
   - Option to add items to the cart.
4. **Cart Page**
   - List of selected items with quantities.
   - Option to update or remove items.
   - Display subtotal, taxes, and total price.
5. **Checkout Page**
   - User details (address, contact information).
   - Payment options.
   - Order summary.

6. **Order Confirmation Page**
    - Display order details, payment status, and shipment details.
    - Include a button to track shipment or print receipt.

High-Level Architecture Diagram Description

2. Content Management and Product Data (Sanity CMS)

- Positioned centrally in the diagram, connected directly to the **Frontend**.
- Manages:
    - Product data (menu items, descriptions, prices, images).
    - Content (blogs, promotions, restaurant information).
- Data flows from Sanity CMS to the Frontend for rendering dynamic content.

3. Authentication (Clerk)

- Integrated with the **Frontend**.
- Handles user authentication and authorization.
- Enables features like:
    - User login/signup.
    - Protected routes (e.g., cart and checkout pages).

4. Payment Gateway (Stripe)

- Connected to the **Frontend**.
- Handles payment processing during checkout.
- Data flow:
    - Frontend sends payment details to Stripe.
    - Stripe processes the payment and returns a success/failure response.

5. Third-Party APIs (ShipEngine)

- Positioned below the **Frontend** layer.
- Used for shipment tracking post-payment.
- Data flow:
    - Frontend sends shipment details to ShipEngine.
    - ShipEngine returns tracking information to the Frontend.

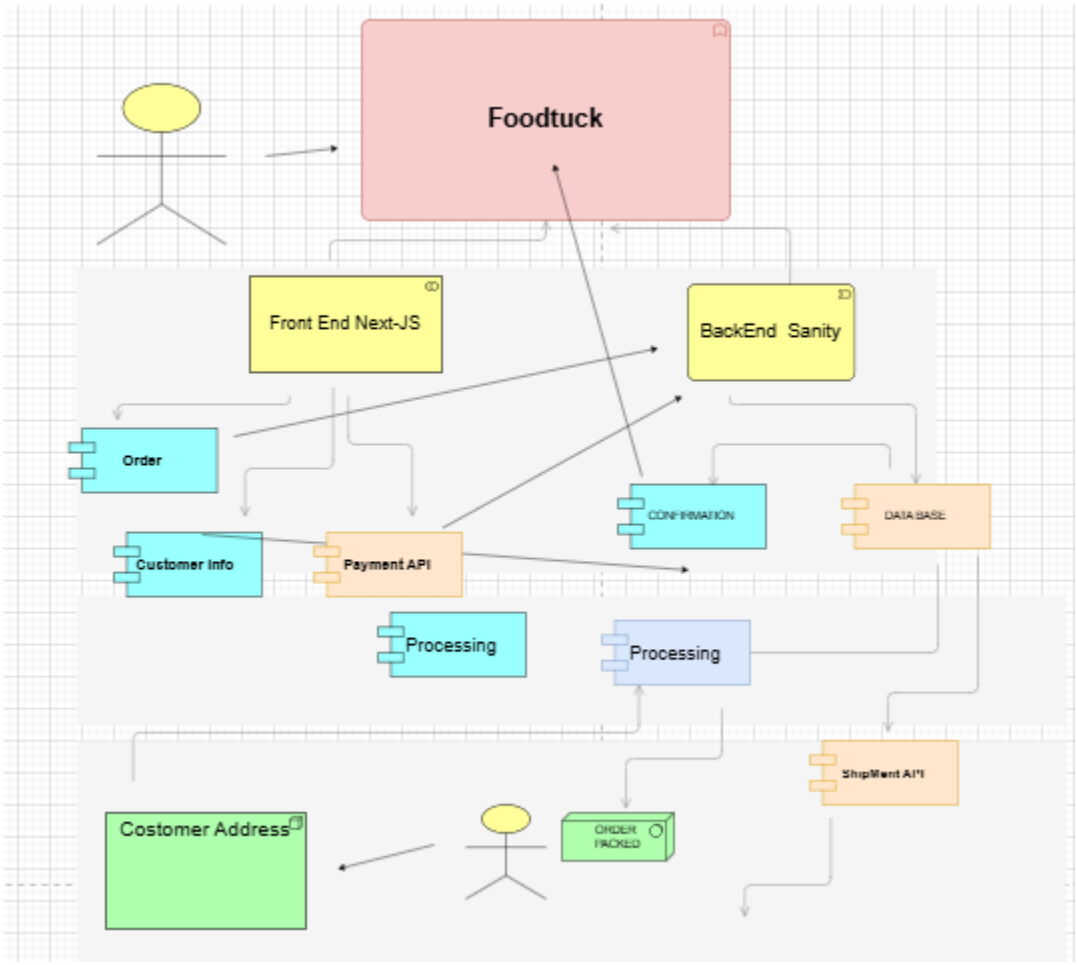
Data Flow

1. **Frontend ↔ Sanity CMS**:
    - Frontend fetches product data and content from Sanity CMS to render pages dynamically.
2. **Frontend ↔ Clerk**:
    - Frontend interacts with Clerk for user authentication and session management.
3. **Frontend ↔ Stripe**:
    - After authentication, the Frontend sends payment details to Stripe for processing.
4. **Frontend ↔ ShipEngine**:
    - Post-payment, the Frontend sends shipment details to ShipEngine and receives tracking information.

Visual Representation (Text-Based Diagram)

```
Copy
+-------------------+
|     Frontend      |
|   (Next.js)       |
|                   |
| - Home Page       |
| - Product Listing |
| - Product Detail  |
| - Cart            |
| - Checkout        |
| - Order Confirmation
+-------------------+
       |  ▲
       |  | (Fetches content/product data)
       ▼  |
+-------------------+
|  Sanity CMS       |
|  (Content &       |
|   Product Data)   |
+-------------------+
       |  ▲
       |  | (Authentication)
       ▼  |
+-------------------+
|     Clerk         |
|  (Authentication) |
+-------------------+
       |  ▲
       |  | (Payment Processing)
       ▼  |
+-------------------+
|     Stripe        |
|  (Payment Gateway)|
+-------------------+
       |  ▲
       |  | (Shipment Tracking)
```

```
      ▼ |
+------------------+
|  ShipEngine      |
| (Shipment API)   |
+------------------+
```



Key Features of the Architecture

1. **Modular Design**: Each component (Frontend, Sanity CMS, Clerk, Stripe, ShipEngine) is independent and scalable.
2. **Dynamic Routing**: Next.js enables dynamic routing for product detail pages (e.g., `/products/[id]`).
3. **Real-Time Updates**: Sanity CMS allows real-time content updates without redeploying the Frontend.
4. **Secure Payments**: Stripe ensures secure and reliable payment processing.
5. **User Management**: Clerk simplifies user authentication and session management.

6. **Shipment Tracking**: ShipEngine provides real-time shipment tracking for a seamless post-purchase experience.

Next Steps

1. **Implement Frontend Pages**:
    - Use Next.js to create the Home, Product Listing, Product Detail, Cart, Checkout, and Order Confirmation pages.
2. **Integrate Sanity CMS**:
    - Set up Sanity CMS for managing product data and content.
3. **Add Authentication**:
    - Integrate Clerk for user authentication.
4. **Set Up Payment Gateway**:
    - Integrate Stripe for payment processing.
5. **Enable Shipment Tracking**:
    - Use ShipEngine API for shipment tracking.

This architecture ensures a scalable, maintainable, and user-friendly Q-commerce website for your restaurant business. Let me know if you need further details or assistance!