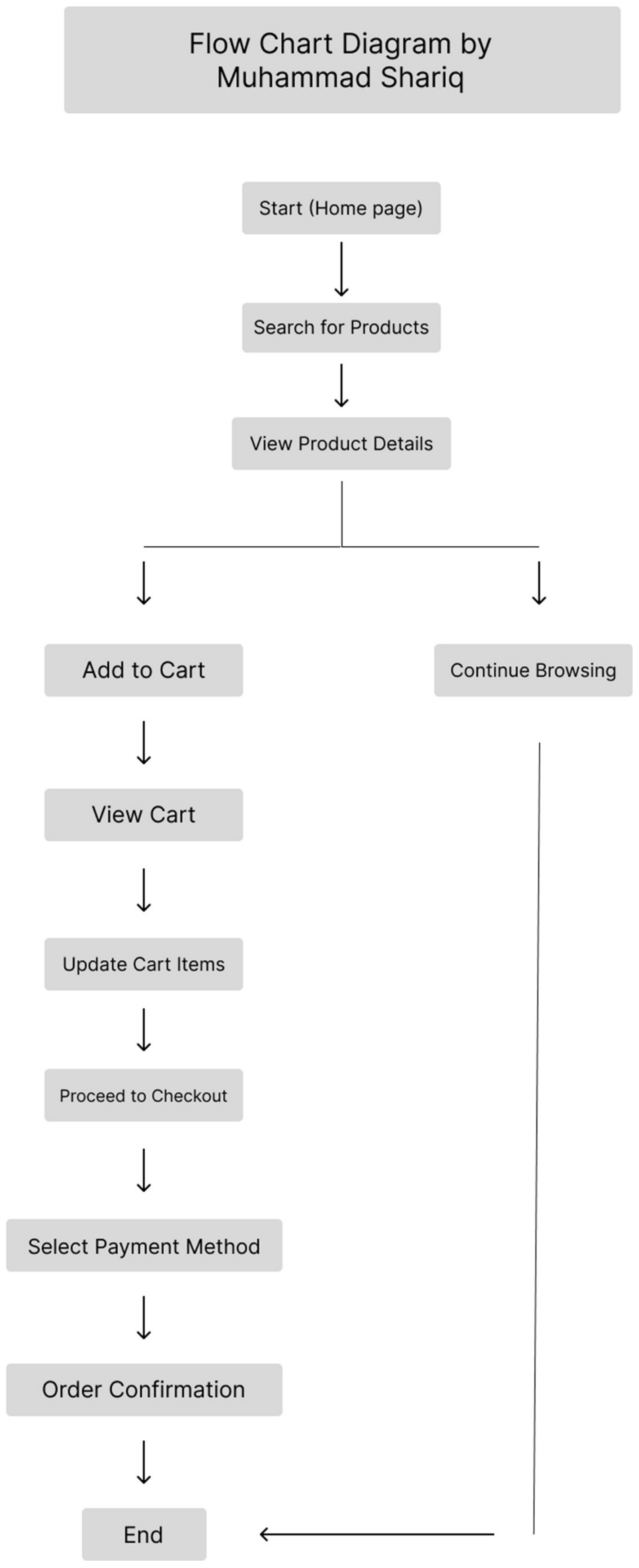
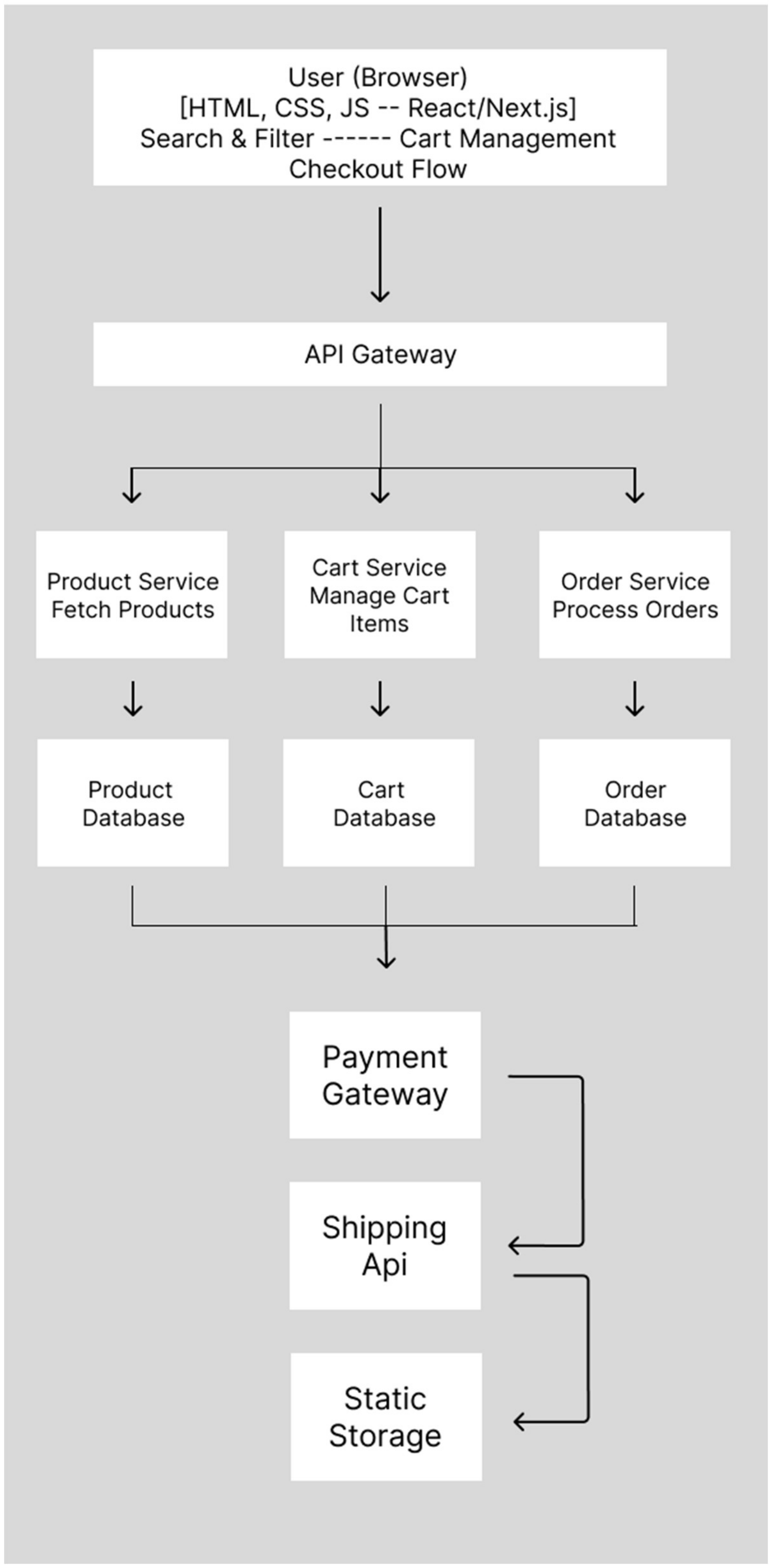
Marketplace Technical Foundation – Muhammad Shariq’s Marketplace

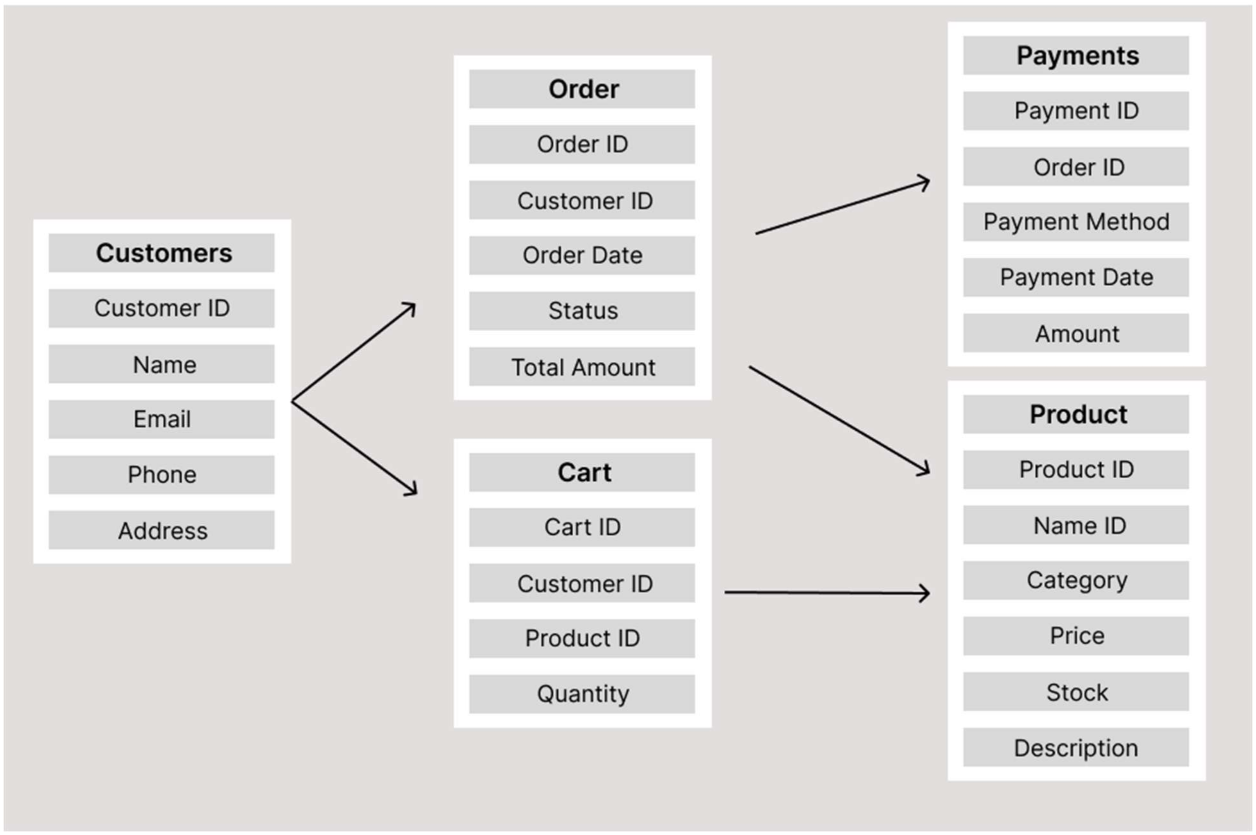
Table of Content:


Flow Chart Diagram by Muhammad Shariq
Start (Home page)
Search for Products
View Product Details
Add to Cart
Continue Browsing
View Cart
Update Cart Items
Proceed to Checkout
Select Payment Method
Order Confirmation
End

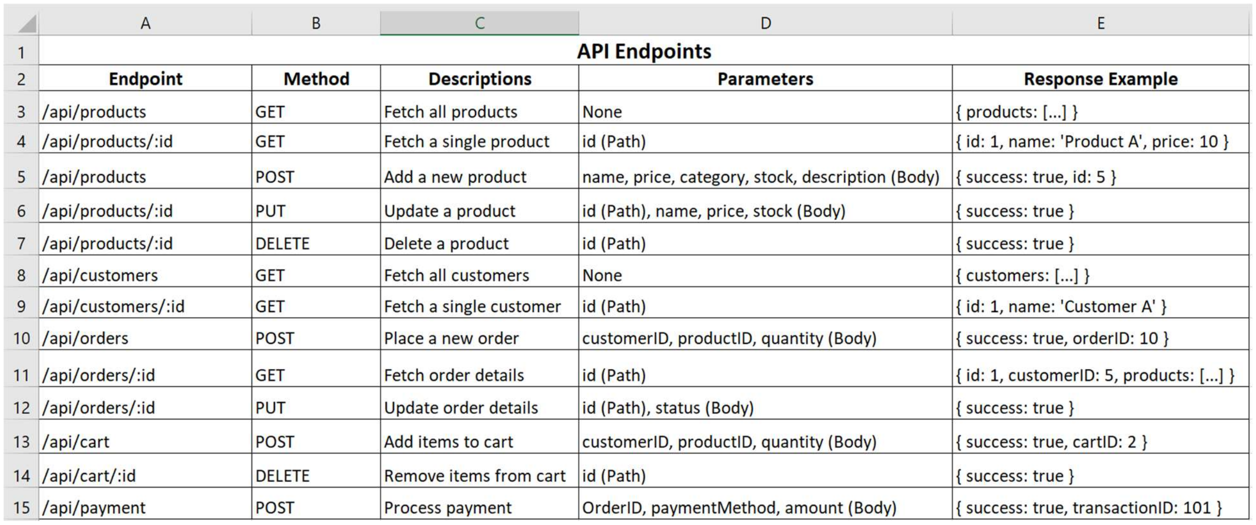
System Architecture by Muhammad Shariq



ER Diagram Relationship by Muhammad Shariq



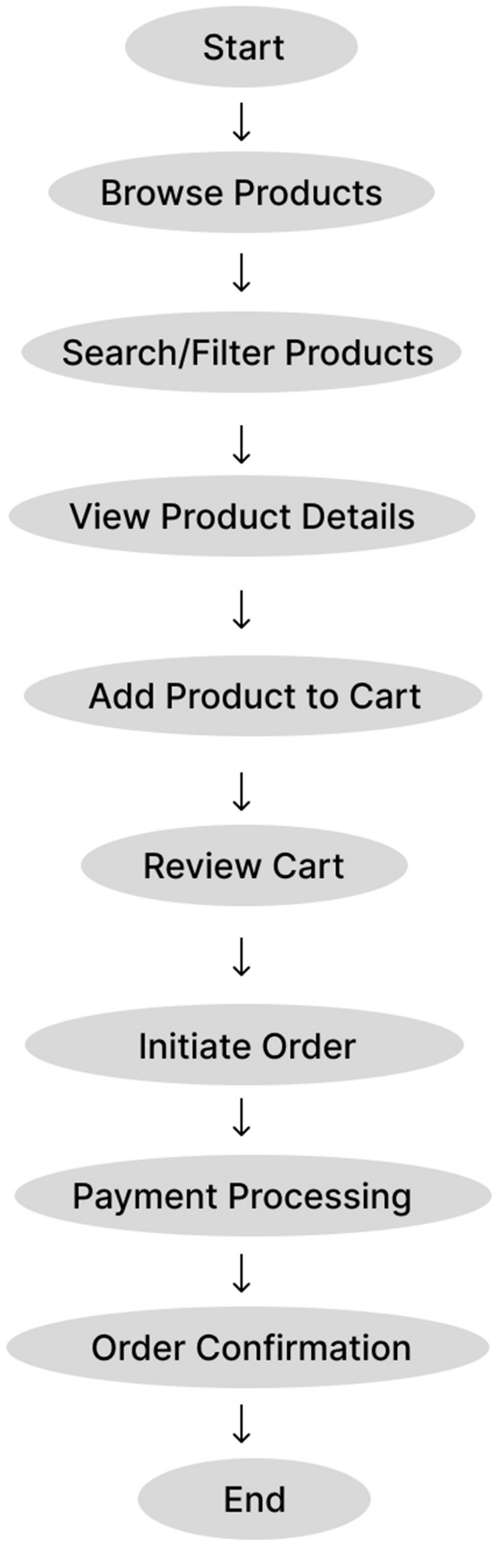
API Endpoints



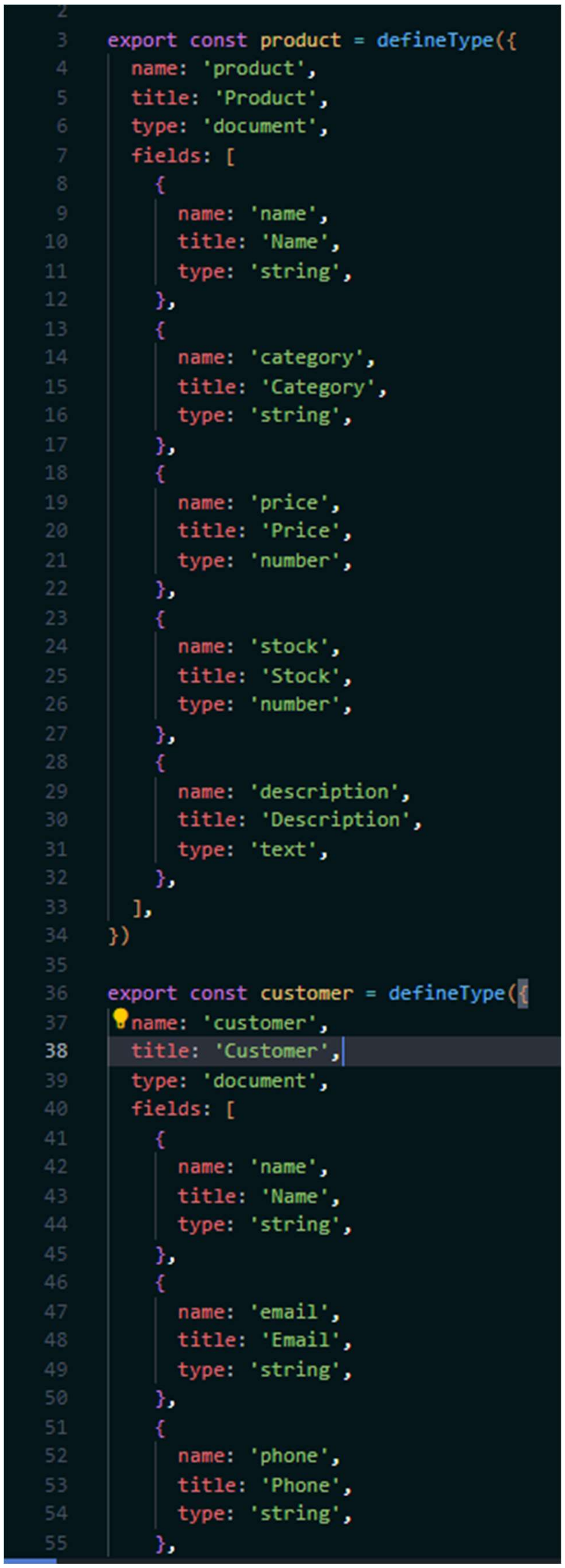
API Endpoints

Endpoint	Method	Descriptions	Parameters	Response Example
/api/products	GET	Fetch all products	None	{ products: [...] }
/api/products/:id	GET	Fetch a single product	id (Path)	{ id: 1, name: 'Product A', price: 10 }
/api/products	POST	Add a new product	name, price, category, stock, description (Body)	{ success: true, id: 5 }
/api/products/:id	PUT	Update a product	id (Path), name, price, stock (Body)	{ success: true }
/api/products/:id	DELETE	Delete a product	id (Path)	{ success: true }
/api/customers	GET	Fetch all customers	None	{ customers: [...] }
/api/customers/:id	GET	Fetch a single customer	id (Path)	{ id: 1, name: 'Customer A' }
/api/orders	POST	Place a new order	customerID, productID, quantity (Body)	{ success: true, orderID: 10 }
/api/orders/:id	GET	Fetch order details	id (Path)	{ id: 1, customerID: 5, products: [...] }
/api/orders/:id	PUT	Update order details	id (Path), status (Body)	{ success: true }
/api/cart	POST	Add items to cart	customerID, productID, quantity (Body)	{ success: true, cartID: 2 }
/api/cart/:id	DELETE	Remove items from cart	id (Path)	{ success: true }
/api/payment	POST	Process payment	OrderID, paymentMethod, amount (Body)	{ success: true, transactionID: 101 }

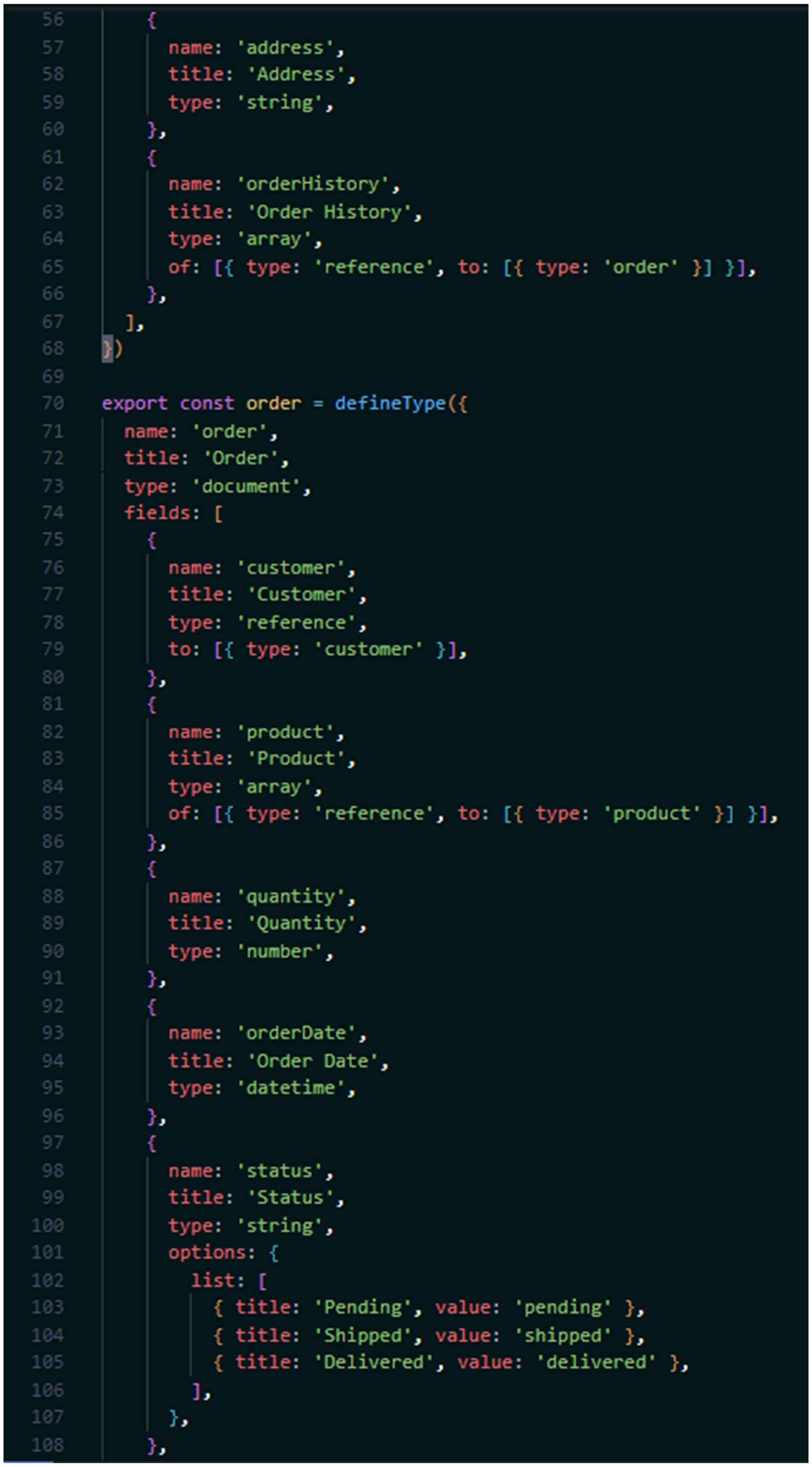
Key Workflow by Muhammad Shariq



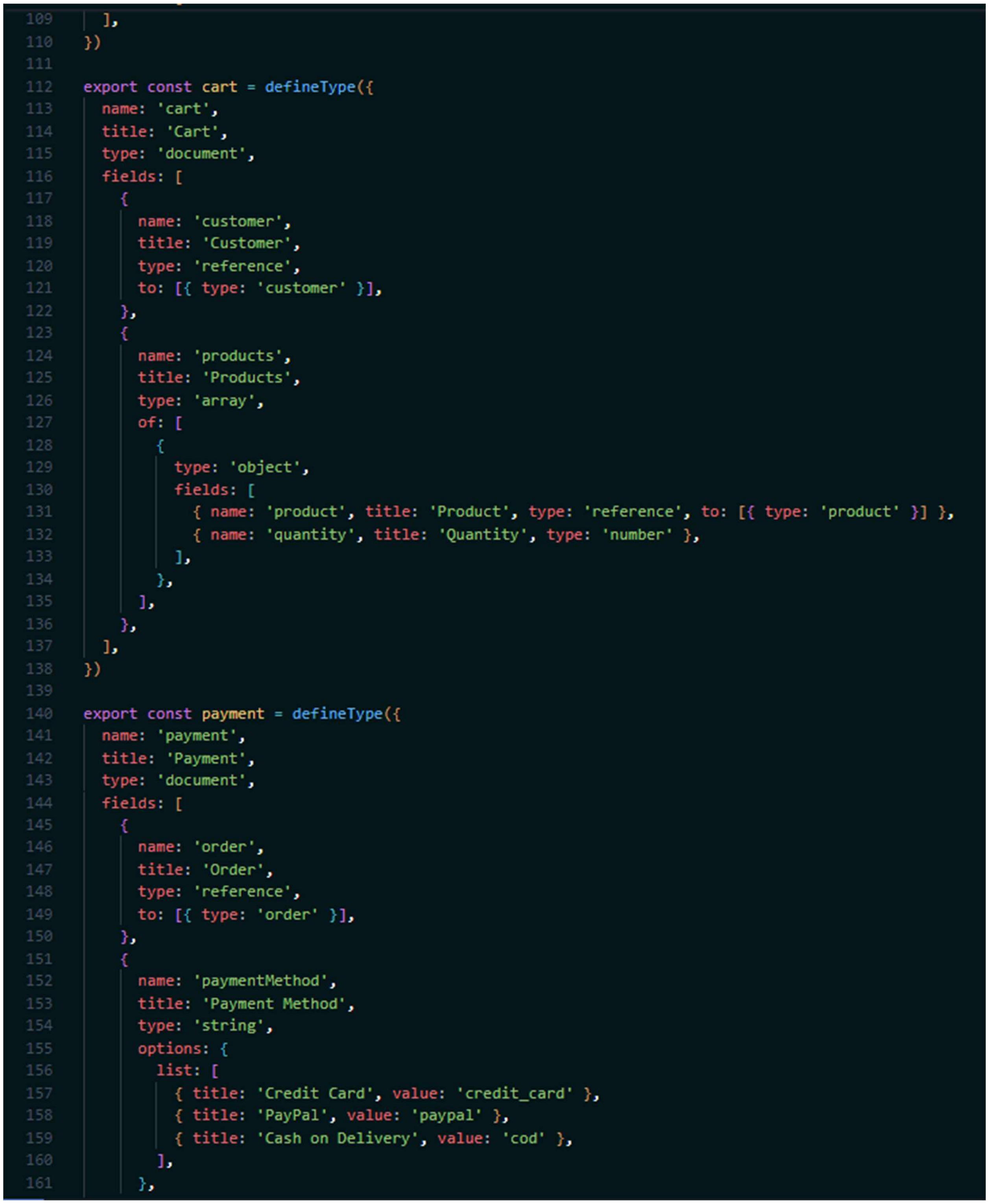
Sanity Schema Design



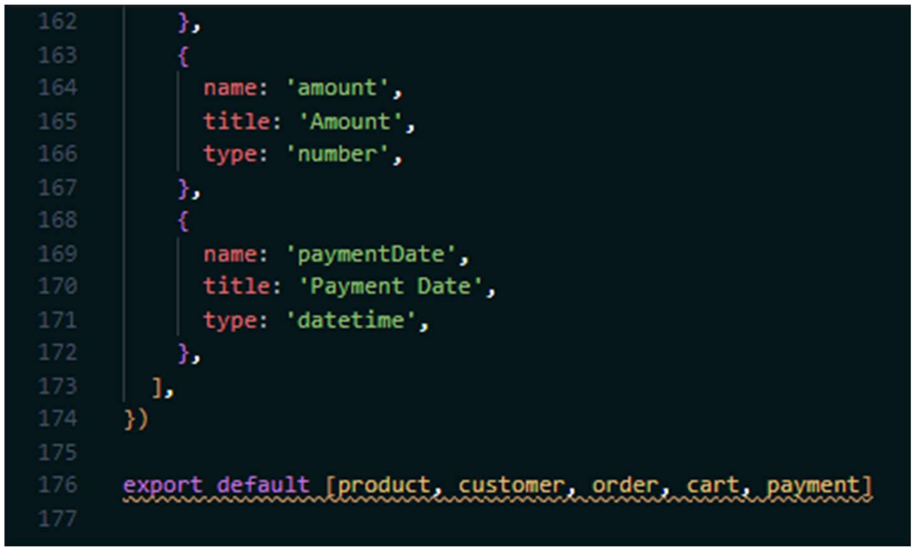
```
export const product = defineType({
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'name',
      title: 'Name',
      type: 'string',
    },
    {
      name: 'category',
      title: 'Category',
      type: 'string',
    },
    {
      name: 'price',
      title: 'Price',
      type: 'number',
    },
    {
      name: 'stock',
      title: 'Stock',
      type: 'number',
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
    },
  ],
})

export const customer = defineType({
  name: 'customer',
  title: 'Customer',
  type: 'document',
  fields: [
    {
      name: 'name',
      title: 'Name',
      type: 'string',
    },
    {
      name: 'email',
      title: 'Email',
      type: 'string',
    },
    {
      name: 'phone',
      title: 'Phone',
      type: 'string',
    },
```



```
    {
      name: 'address',
      title: 'Address',
      type: 'string',
    },
    {
      name: 'orderHistory',
      title: 'Order History',
      type: 'array',
      of: [{ type: 'reference', to: [{ type: 'order' }] }],
    },
  ],
})

export const order = defineType({
  name: 'order',
  title: 'Order',
  type: 'document',
  fields: [
    {
      name: 'customer',
      title: 'Customer',
      type: 'reference',
      to: [{ type: 'customer' }],
    },
    {
      name: 'product',
      title: 'Product',
      type: 'array',
      of: [{ type: 'reference', to: [{ type: 'product' }] }],
    },
    {
      name: 'quantity',
      title: 'Quantity',
      type: 'number',
    },
    {
      name: 'orderDate',
      title: 'Order Date',
      type: 'datetime',
    },
    {
      name: 'status',
      title: 'Status',
      type: 'string',
      options: {
        list: [
          { title: 'Pending', value: 'pending' },
          { title: 'Shipped', value: 'shipped' },
          { title: 'Delivered', value: 'delivered' },
        ],
      },
    },
```



```
  ],
})

export const cart = defineType({
  name: 'cart',
  title: 'Cart',
  type: 'document',
  fields: [
    {
      name: 'customer',
      title: 'Customer',
      type: 'reference',
      to: [{ type: 'customer' }],
    },
    {
      name: 'products',
      title: 'Products',
      type: 'array',
      of: [
        {
          type: 'object',
          fields: [
            { name: 'product', title: 'Product', type: 'reference', to: [{ type: 'product' }] },
            { name: 'quantity', title: 'Quantity', type: 'number' },
          ],
        },
      ],
    },
  ],
})

export const payment = defineType({
  name: 'payment',
  title: 'Payment',
  type: 'document',
  fields: [
    {
      name: 'order',
      title: 'Order',
      type: 'reference',
      to: [{ type: 'order' }],
    },
    {
      name: 'paymentMethod',
      title: 'Payment Method',
      type: 'string',
      options: {
        list: [
          { title: 'Credit Card', value: 'credit_card' },
          { title: 'PayPal', value: 'paypal' },
          { title: 'Cash on Delivery', value: 'cod' },
        ],
      },
```



```
    },
    {
      name: 'amount',
      title: 'Amount',
      type: 'number',
    },
    {
      name: 'paymentDate',
      title: 'Payment Date',
      type: 'datetime',
    },
  ],
})

export default [product, customer, order, cart, payment]
```

<u>Collaboration Notes:</u>

1. Initial Planning and Data Structuring

- **Objective:** Build a user-friendly website for browsing and purchasing products.
- **Schema Design:** We structured the core entities:
    - **Products:** With fields like ProductID, Name, Category, Price, Stock, and Description.
    - **Customers:** Including CustomerID, Name, Email, Phone, Address, and OrderHistory.
    - **Orders:** Tracking order details like OrderID, CustomerID, ProductID, Quantity, OrderDate, and Status.
    - **Cart:** Linking customers to selected products with CartID, CustomerID, ProductID, and Quantity.
    - **Payments:** Managing transaction details such as OrderID, PaymentMethod, Amount, and PaymentDate.

2. User Interaction Flow

- Designed a **user interaction flowchart** detailing key workflows:
    - **Product Browsing:** Users can search and filter products.
    - **Adding to Cart:** Products can be added to the cart with specific quantities.
    - **Order Placement:** Finalizing the purchase process.
    - **Checkout:** Managing payments and confirming orders.

3. System Architecture

- Created a **system architecture diagram** highlighting:
    - **Frontend:** Built for user interaction with intuitive UI/UX.
    - **Backend:** Handles business logic, order processing, and cart management.
    - **Database:** Designed for efficient storage and management of products, customers, and orders.

    - **API Layer:** Connects frontend with backend to enable seamless communication.

4. ER Diagram

- Developed an **ER Diagram** to visualize relationships between entities:
    - Products are linked to orders and carts.
    - Customers are connected to orders and carts.
    - Orders and payments are associated for transaction tracking.

5. API Design

- Designed API endpoints for efficient backend communication:
    - CRUD operations for **Products**, **Orders**, and **Cart**.
    - Endpoints to fetch and update data (e.g., /api/products, /api/orders).
    - Defined **methods**, **parameters**, and **response examples**.

6. Sanity Schema

- Created a **Sanity schema** in TypeScript for CMS integration:
    - Defined document schemas for Products, Customers, Orders, Cart, and Payments.
    - Enabled dynamic content management for seamless updates.

Outcome

We now have a well-planned, technically sound foundation for the website, covering:

1. **Data management** with clearly defined schemas and relationships.
2. **APIs** to support seamless interaction between components.
3. A robust **user experience flow** to simplify navigation and purchase processes.
4. Scalability through the integration of tools like **Sanity CMS**.

This approach ensures an organized, functional, and efficient website ready for development and deployment.