1. Technology Stack

- Frontend: TypeScript, React, Next.js
- Backend: (Assuming a separate backend team or a third-party service)
- Database: (Assuming a separate database team or a third-party service)

Essential Pages

1. Home
2. Product List
3. Product Details
4. Cart
5. Checkout
6. Order Confirmation

Flowchart
Here is a simplified flowchart:

```
+------------------+
|  User Visits   |
|  Website (Home) |
+------------------+
       |
       |
       v
+------------------+
|  Browse Products  |
|  (Product List)    |
+------------------+
       |
       |
       v
+------------------+
|  View Product    |
|  Details (Product  |
|  Details)         |
+------------------+
       |
       |
       v
+------------------+
|  Add to Cart     |
|  (Cart)          |
+------------------+
       |
       |
       v
+------------------+
|  Checkout        |
|  (Checkout)      |
+------------------+
       |
       |
       v
```

```
+-----------------+
|  Order Confirmation|
|  (Order Confirmation)|
+-----------------+
```

ER Diagram

Here is a simplified ER diagram:

```
+----------------+
|  User          |
+----------------+
|  - User ID (PK)
|  - Name
|  - Email
|  - Password

+----------------+
|  Product       |
+----------------+
|  - Product ID (PK)
|  - Name
|  - Description
|  - Price
|  - Image

+----------------+
|  Product Color|
+----------------+
|  - Product Color ID (PK)
|  - Product ID (FK)
|  - Color

+----------------+
|  Cart          |
+----------------+
|  - Cart ID (PK)
|  - User ID (FK)
|  - Product ID (FK)
|  - Quantity

+----------------+
|  Order         |
+----------------+
|  - Order ID (PK)
|  - User ID (FK)
|  - Order Date
|  - Total Cost

+----------------+
|  Order Item    |
+----------------+
|  - Order Item ID (PK)
|  - Order ID (FK)
|  - Product ID (FK)
|  - Quantity
```

Sanity as a Backend to manage my all content of apparel product:

Product Schema

```
{
  "name": "product",
  "type": "document",
  "fields": [
    {
      "name": "title",
      "type": "string"
    },
    {
      "name": "description",
      "type": "text"
    },
    {
      "name": "price",
      "type": "number"
    },
    {
      "name": "images",
      "type": "array",
      "of": {
        "type": "image"
      }
    },
    {
      "name": "categories",
      "type": "array",
      "of": {
        "type": "reference",
        "to": {
          "type": "category"
        }
      }
    },
    {
      "name": "sizes",
      "type": "array",
      "of": {
        "type": "string"
      }
    },
    {
      "name": "colors",
      "type": "array",
      "of": {
        "type": "string"
      }
    },
    {
      "name": "stock",
      "type": "number"
    },
    {
      "name": "slug",
      "type": "slug",
      "options": {
        "source": "title"
      }
    }
  ]
}
```

Category Schema

```
{
  "name": "category",
  "type": "document",
  "fields": [
    {
      "name": "title",
      "type": "string"
    },
    {
      "name": "description",
      "type": "text"
    },
    {
      "name": "slug",
      "type": "slug",
      "options": {
        "source": "title"
      }
    }
  ]
}
```

Customer Schema

```
{
  "name": "customer",
  "type": "document",
  "fields": [
    {
      "name": "name",
      "type": "string"
    },
    {
      "name": "email",
      "type": "string"
    },
    {
      "name": "phone",
      "type": "string"
    },
    {
      "name": "address",
      "type": "text"
    }
  ]
}
```

Order Schema

```
{
  "name": "order",
  "type": "document",
  "fields": [
    {
      "name": "customer",
      "type": "reference",
      "to": {
        "type": "customer"
      }
    },
    {
      "name": "products",
      "type": "array",
      "of": {
        "type": "reference",
        "to": {
          "type": "product"
        }
      }
    },
    {
      "name": "total",
      "type": "number"
    },
    {
      "name": "status",
      "type": "string"
    },
    {
      "name": "date",
      "type": "datetime"
    }
  ]
}
```

Third Party API's:

Shipment Tracking APIs

1. ShipStation API: Provides access to shipping rates, tracking information, and shipping labels.
2. ShippingEasy API: Offers shipping rates, tracking information, and shipping labels, as well as inventory management and order management.
3. Tracktry API: Provides real-time tracking information for shipments from various carriers.
4. AfterShip API: Offers tracking information, delivery updates, and shipping analytics.

Payment Gateway APIs

1. Stripe API: Provides a robust payment gateway with features like payment processing, invoicing, and subscription management.
2. PayPal API: Offers payment processing, invoicing, and subscription management, as well as features like PayPal Express Checkout.
3. Braintree API: Offers payment processing, invoicing, and subscription management, as well as features like recurring payments and PayPal integration.

Other Backend Services APIs

1. Mailchimp API: Provides email marketing automation, customer segmentation, and email campaign management.
2. Klaviyo API: Offers email marketing automation, customer segmentation, and email campaign management, as well as features like SMS marketing.
3. Zendesk API: Provides customer support ticketing, chat, and phone support, as well as features like customer self-service and knowledge base management.
4. Google Analytics API: Offers website analytics, conversion tracking, and audience segmentation.

Benefits of Using Third-Party APIs

1. Improved Customer Experience: Provide real-time tracking information, automated email updates, and seamless payment processing.
2. Increased Efficiency: Automate tasks like shipping label generation, payment processing, and customer support ticketing.
3. Enhanced Security: Leverage the security expertise of third-party API providers to protect sensitive customer data.
4. Scalability: Easily integrate with third-party APIs to scale your business and meet growing customer demands.

2.System Design Architecture

Here is a system design flowchart for the apparel e-commerce platform:

```
+-------------------+
|  User Interaction  |
+-------------------+
       |
       |
       v
+-------------------+
|  Client-Side App  |
|  (React, Next.js)  |
+-------------------+
       |
       |
       v
+-------------------+
|  API Request      |
|  (Server-Side App) |
+-------------------+
       |
       |
       v
+-------------------+
|  Server-Side App  |
|  (Node.js, Express) |
+-------------------+
       |
       |
       v
+-------------------+
|  Database Query    |
|  (Sanity CMS)      |
+-------------------+
       |
       |
       v
+-------------------+
|  Data Retrieval    |
|  (Sanity CMS)      |
+-------------------+
       |
       |
       v
+-------------------+
|  API Response      |
|  (Server-Side App) |
+-------------------+
       |
       |
       v
+-------------------+
|  Client-Side Rendering|
|  (React, Next.js)    |
+--------------------+
       |
       |
       v
+--------------------+
|  User Interface     |
|  (UI Components)    |
+--------------------+


+--------------------+
|  Payment Processing  |
|  (Third-Party API)  |
+--------------------+
       |
       |
       v
+--------------------+
|  Shipping Integration |
|  (Third-Party API)  |
+--------------------+
       |
       |
       v
+--------------------+
|  Order Management    |
|  (Server-Side App)  |
+--------------------+
       |
       |
       v
+--------------------+
|  Inventory Management |
|  (Server-Side App)  |
+--------------------+


+--------------------+
|  User Authentication  |
|  (Server-Side App)  |
+--------------------+
       |
       |
       v
+--------------------+
|  Authorization      |
|  (Server-Side App)  |
+--------------------+
       |
       |
       v
```

```
                                      +--------------------+
                                      |  Data Encryption    |
                                      |  (Server-Side App)  |
                                      +--------------------+
```

This flowchart illustrates the system design for the apparel e-commerce platform, including:

1. User interaction and client-side rendering
2. Server-side application and database query
3. Payment processing and shipping integration
4. Order management and inventory management
5. User authentication and authorization
6. Data encryption and security measures

3. The Plan API Requirements are:
Product Endpoints

1. Get All Products

- Endpoint Name: /products
- Method: GET
- Description: Retrieves a list of all products.
- Response Example:

```
[
  {
    "id": 1,
    "name": "T-Shirt",
    "description": "A high-quality t-shirt",
    "price": 19.99,
    "image": "https://example.com/t-shirt.jpg"
  },
  {
    "id": 2,
    "name": "Jeans",
    "description": "A pair of stylish jeans",
    "price": 49.99,
    "image": "https://example.com/jeans.jpg"
  }
]
```

2. Get Product by ID

- Endpoint Name: /products/{id}
- Method: GET
- Description: Retrieves a product by its ID.
- Response Example:

```
{
  "id": 1,
  "name": "T-Shirt",
  "description": "A high-quality t-shirt",
  "price": 19.99,
  "image": "https://example.com/t-shirt.jpg"
}
```

3. Create Product

- Endpoint Name: /products
- Method: POST
- Description: Creates a new product.
- Request Body Example:

```
{
  "name": "New T-Shirt",
  "description": "A new high-quality t-shirt",
  "price": 29.99,
  "image": "https://example.com/new-t-shirt.jpg"
}
```

- Response Example:

```
{
  "id": 3,
  "name": "New T-Shirt",
  "description": "A new high-quality t-shirt",
  "price": 29.99,
  "image": "https://example.com/new-t-shirt.jpg"
}
```

4. Update Product

- Endpoint Name: /products/{id}
- Method: PUT
- Description: Updates a product by its ID.
- Request Body Example:

```
{
  "name": "Updated T-Shirt",
  "description": "An updated high-quality t-shirt",
  "price": 39.99,
  "image": "https://example.com/updated-t-shirt.jpg"
}
```

- Response Example:

```
{
  "id": 1,
  "name": "Updated T-Shirt",
  "description": "An updated high-quality t-shirt",
  "price": 39.99,
  "image": "https://example.com/updated-t-shirt.jpg"
}
```

5. Delete Product

- Endpoint Name: /products/{id}
- Method: DELETE
- Description: Deletes a product by its ID.
- Response Example:

```
{
  "message": "Product deleted successfully"
}
```

Order Endpoints

1. Create Order

- Endpoint Name: /orders
- Method: POST
- Description: Creates a new order.
- Request Body Example:

```
{
  "products": [
    {
      "id": 1,
      "quantity": 2
    },
    {
      "id": 2,
      "quantity": 1
    }
  ],
  "customer": {
    "name": "John Doe",
    "email": "john.doe@example.com"
  }
}
```

- Response Example:

```
{
  "id": 1,
  "products": [
    {
      "id": 1,
      "quantity": 2
    },
    {
      "id": 2,
      "quantity": 1
    }
  ],
  "customer": {
    "name": "John Doe",
    "email": "john.doe@example.com"
  }
}
```

2. Get Order by ID

- Endpoint Name: /orders/{id}
- Method: GET
- Description: Retrieves an order by its ID.
- Response Example:

```
{
  "id": 1,
  "products": [
    {
      "id": 1,
      "quantity": 2
    },
    {
      "id": 2,
```

4. The Technical Documentation includes:
System Design Architecture:
Frontend
Backend
DB
TPS (Third Party Service)

Workflow Diagram:
User Interaction user interacts with the frontend, to interact more and more people.
Backend Processing backend receives the request, processes it, and interacts with the database and third-party services as needed.
DB Query the database to retrieve or update data.
TPI (Third Party Integration):third-party services to handle tasks such as payment processing and shipping.

DB Schema Design:
 Product:
    - id (primary key)
    - name
    - description
    - price
    - image
 Order:
    - id (primary key)
    - customer_id (foreign key referencing Customer)
    - order_date
    - total_cost
    - status

```
 Customer:
    - id (primary key)
    - name
    - email
    - password
    - address
 Payment:
    - id (primary key)
    - order_id (foreign key referencing Order)
    - payment_method
    - payment_date
    - payment_amount
```

Technical Roadmap:

1. Week 1-2: Set up development environment, install dependencies, and configure Sanity CMS.
2. Week 3-4: Design and implement frontend components, including product listing, product details, and shopping cart.
3. Week 5-6: Implement backend API endpoints for product management, order management, and payment processing.
4. Week 7-8: Integrate third-party services for payment processing and shipping.
5. Week 9-10: Conduct testing and debugging, ensure compatibility with different browsers and devices.
6. Week 11: Deploy application to production environment, configure monitoring and logging.
7. Week 12: Review and refine application, gather feedback from users, and plan for future development.

5. Collaboration and Refine:

Attend Industry Events
Reach Out and Connect
Building Relationships:
Build a Strong Online Presence
Engage in Meaningful Conversation
Showcase Your Expertise
Collaboration Opportunities:
Joint Ventures
Investment