Technical Foundation Plan for Shoe and Fitness E-Commerce Platform

OVERVIEW:

This document outlines the technical requirements, system architecture, API plans, and deployment strategies for developing a dynamic e-commerce platform dedicated to shoes and fitness gear.

---

1. Technology Stack and Tools Overview

Frontend

- **Framework:** Next.js, optimized for server-side rendering and high-performance user interactions.
- **Features:** Intuitive pages, reusable components, and dynamic routing for a seamless shopping experience.

Backend

- **Core:** RESTful APIs to handle business logic and data operations.
- **Database:** MongoDB for managing users, orders, and product inventory.

Content Management

- **CMS:** Sanity to manage dynamic content, such as product details and promotions.

Third-Party Integrations

- **Authentication:** Clerk for secure login, registration, and session handling.
- **Payments:** Stripe to process secure financial transactions for fitness products and shoes.
- **Shipment:** ShipEngine for delivery tracking and real-time shipping cost calculations.

Hosting and Deployment

- **Platform:** Vercel, providing reliable hosting and CI/CD pipelines for continuous updates.

---

2. Core Functionalities and Components

Frontend Functionalities

1. **Homepage:**
    - Showcases featured fitness gear and popular shoe collections.
    - Promotes ongoing sales or new product launches.
    - Includes a navbar for quick access to product categories, offers, and user account settings.
2. **Product Listing Page:**
    - Displays a grid of fitness products (e.g., shoes, workout accessories).
    - Includes sorting and filtering options by price, brand, or category.
3. **Product Details Page:**
    - Provides detailed descriptions of each product, including high-quality images, material specifications, and fitness benefits.
4. **Checkout Page:**
    - Collects user details like shipping address and payment method to finalize purchases.

Backend Functionalities

1. **Dynamic Content Management:**
    - **Products:** Sanity CMS will manage product schemas with fields like name, price, brand, size, and stock levels.
    - **Promotions:** Dynamic schema for discounts or seasonal offers, including fields like title, description, discount percentage, and validity.
2. **APIs:**
   RESTful endpoints will support authentication, product management, cart operations, and order handling (detailed in Section 4).
3. **Real-Time Updates:**

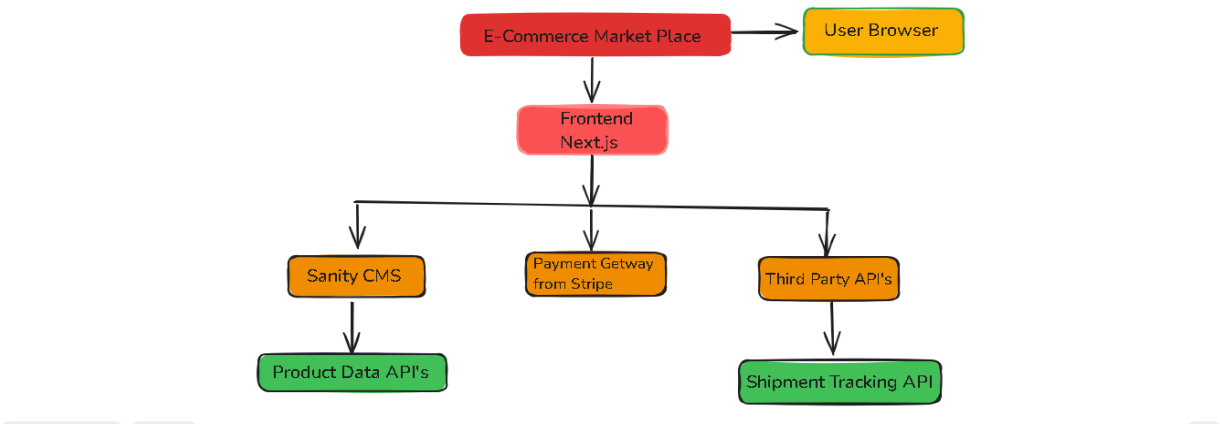
- Content changes, such as adding new shoe collections or updating fitness gear details, will reflect instantly on the frontend via Sanity.

---

2. System Architecture Design

The platform is designed with a modular and scalable service-oriented architecture:

System Diagram:



System Flow Diagram

- User requests from the frontend (e.g., view products, place an order) are routed to the backend via APIs.
- Backend fetches product data from MongoDB or dynamic content from Sanity.
- Backend processes payments via Stripe and shipments via ShipEngine.
- Responses are sent back to the frontend for rendering**.**

---

3. Detailed API Design

. Authentication

- **Register User**: /api/auth/register (POST)

    - **Request Body**:

```
{
  "email": "user@example.com",
  "password": "password123",
  "fullName": "John Doe",
  "address": "123 Main St, City, Country"
}
```

    - **Response**:

```
{
  "message": "User registered successfully",
  "userId": "unique_user_id"
}
```

- **Login User**: /api/auth/login (POST)
    - **Request Body**:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

    - **Response**:

```
{
  "token": "jwt_token",
  "userId": "unique_user_id"
}
```

- **Logout User**: /api/auth/logout (POST)

```
{
  "token": "jwt_token"
```

}

- **Response**:

{

"message": "User logged out successfully"

}

- **Forgot Password**: /api/auth/forgot-password (POST)
  - **Request Body**:

{

"email": "user@example.com"

}

- **Response**:

{

"message": "Password reset email sent"

}

---

2. Products

- **List Products**: /api/products (GET)
  - **Query Parameters**:
    - category=<category_name>
    - priceRange=<minPrice>,<maxPrice>
    - sortBy=<price|popularity|rating>
  - **Response**:

[

{

"id": "product_id",

"name": "Product Name",

```
    "price": 29.99,
    "category": "Shoes",
    "rating": 4.5,
    "imageUrl": "image_url"
  }
]
```


- **Product Details**: /api/products/:id (GET)


```
{
  "id": "product_id",
  "name": "Product Name",
  "price": 29.99,
  "category": "Shoes",
  "description": "Detailed description of the product",
  "stock": 50,
  "rating": 4.5,
  "reviews": [
    {
      "userId": "user_id",
      "rating": 5,
      "reviewText": "Great product!",
      "createdAt": "2025-01-20T12:00:00Z"
    }
  ],
  "imageUrls": ["image_url_1", "image_url_2"]
}
```


- **Search Products**: /api/products/search (GET)

- **Query Parameters**:
  - q=<search_term>
- **Response**:

```
[
 {
  "id": "product_id",
  "name": "Product Name",
  "price": 29.99
 }
]
```

---

3. Cart

- **Add to Cart**: /api/cart (POST)
  - **Request Body**:

```
{
 "userId": "unique_user_id",
 "productId": "product_id",
 "quantity": 2
}
```

  - **Response**:

```
{
 "message": "Product added to cart"
}
```

- **Fetch Cart**: /api/cart/:userId (GET)
  - **Response**:

```
{
```

```
 "userId": "unique_user_id",
 "items": [
  {
   "productId": "product_id",
   "quantity": 2,
   "price": 29.99
  }
 ]
}
```


- **Remove from Cart**: /api/cart/remove (POST)
    - **Request Body**:


```
{
 "userId": "unique_user_id",
 "productId": "product_id"
}
```


    - **Response**:


```
{
 "message": "Product removed from cart"
}
```


---

4. Orders

- **Create Order**: /api/orders (POST)
    - **Request Body**:


```
{
 "userId": "unique_user_id",
 "cartItems": [
```

```
  {
    "productId": "product_id",
    "quantity": 2
  }
 ],
 "shippingAddress": "123 Main St, City, Country",
 "paymentMethod": "credit_card",
 "totalPrice": 59.98
}
```


- **Response**:


```
{
 "orderId": "order_id",
 "status": "Pending",
 "estimatedDelivery": "2025-01-25"
}
```


- **Fetch Order**: /api/orders/:orderId (GET)
  - **Response**:


```
{
 "orderId": "order_id",
 "status": "Shipped",
 "totalPrice": 59.98,
 "shippingAddress": "123 Main St, City, Country",
 "items": [
  {
    "productId": "product_id",
    "quantity": 2
```

```
  }
 ],
 "trackingNumber": "tracking_number"
}
```


- **Cancel Order**: /api/orders/cancel/:orderId (POST)
    - **Response**:


```
{
 "message": "Order cancelled successfully"
}
```


---

5. Payment

- **Process Payment**: /api/payment (POST)
    - **Request Body**:


```
{
 "userId": "unique_user_id",
 "orderId": "order_id",
 "paymentDetails": {
  "method": "credit_card",
  "cardNumber": "4111111111111111",
  "expiryDate": "12/25",
  "cvv": "123"
 }
}
```


    - **Response**:


```
{
 "message": "Payment processed successfully",
```

"paymentId": "payment_id"

}

- **Refund Payment**: /api/payment/refund (POST)
    - **Request Body**:

{

"paymentId": "payment_id",

"reason": "Product return"

}

    - **Response**:

{

"message": "Refund processed successfully"

}

---

6. Shipment

- **Create Shipment**: /api/shipment (POST)
    - **Request Body**:

{

"orderId": "order_id",

"address": "123 Main St, City, Country",

"shipping Method": "express"

}

Response:

{

"shipmentId": "shipment_id",

"trackingNumber": "tracking_number",

"estimatedDelivery": "2025-01-25"

}

- **Track Shipment**: /api/shipment/:trackingId (GET)
    - **Response**:

{
"status": "In transit",
"currentLocation": "City, Country",
"estimatedDelivery": "2025-01-25"
}

---

5. Security Considerations

1. **Authentication & Authorization:**
    - Role-Based Access Control (RBAC) restricts access to sensitive operations.
    - JWT tokens are used to securely manage user sessions.
2. **Data Security:**
    - HTTPS ensures encrypted communication between users and the platform.
    - Passwords are hashed with bcrypt before storage.
3. **API Security:**
    - All inputs are validated to prevent SQL injection and XSS attacks.
    - Rate limiting prevents abuse of APIs.

---

6. Deployment Strategy

1. **Continuous Integration & Deployment:**
    - Automated CI/CD pipelines via GitHub Actions ensure efficient builds and deployments.

2. **Environment Configuration:**
    - Sensitive data (e.g., API keys, database URIs) is securely stored in .env files.
3. **Hosting:**
    - Vercel provides efficient deployment with minimal downtime and fast performance.

---

7. Conclusion

This structure sets the foundation for building a user-friendly, scalable, and secure shoe and fitness e-commerce platform. With features like real-time content updates, a seamless shopping experience, and secure payment processing, the platform ensures an engaging and reliable experience for fitness enthusiasts and shoe lovers alike.