E-Commerce Website Technical Documentation

This document outlines the technical foundation of the e-commerce marketplace, detailing system architecture, key workflows, API endpoints, and the tools used in both frontend and backend development.

The purpose of this document is to provide a clear, structured view of how the system is designed to function, ensuring that all components—frontend, backend, APIs, and external integrations—work seamlessly together to deliver a user-friendly and scalable marketplace experience.

Key sections include:

- **System Architecture**: A high-level overview illustrating how different components interact.
- **Key Workflows**: Step-by-step processes guiding user interactions within the platform.
- **API Endpoints**: Detailed specifications for handling data exchanges between the frontend, backend, and external services.
- **Tools and Technologies**: Descriptions of the tools used for frontend development, backend management, and API integration.

This document serves as a foundation for development and can be utilized for future enhancements, collaboration, and sharing with stakeholders.

1. **Technical Requirements**

The table below outlines the various tools utilized in the development of your e-commerce marketplace. Each tool serves a specific purpose in enhancing the functionality of the platform, from frontend development to backend processing and API integration.

- **Frontend Tools** are responsible for creating the user interface and ensuring a seamless user experience.

- **Backend Tools** handle server-side operations such as data storage, processing, and management.
- **APIs** facilitate communication between different parts of the system, enabling data exchange and interactions with external services.

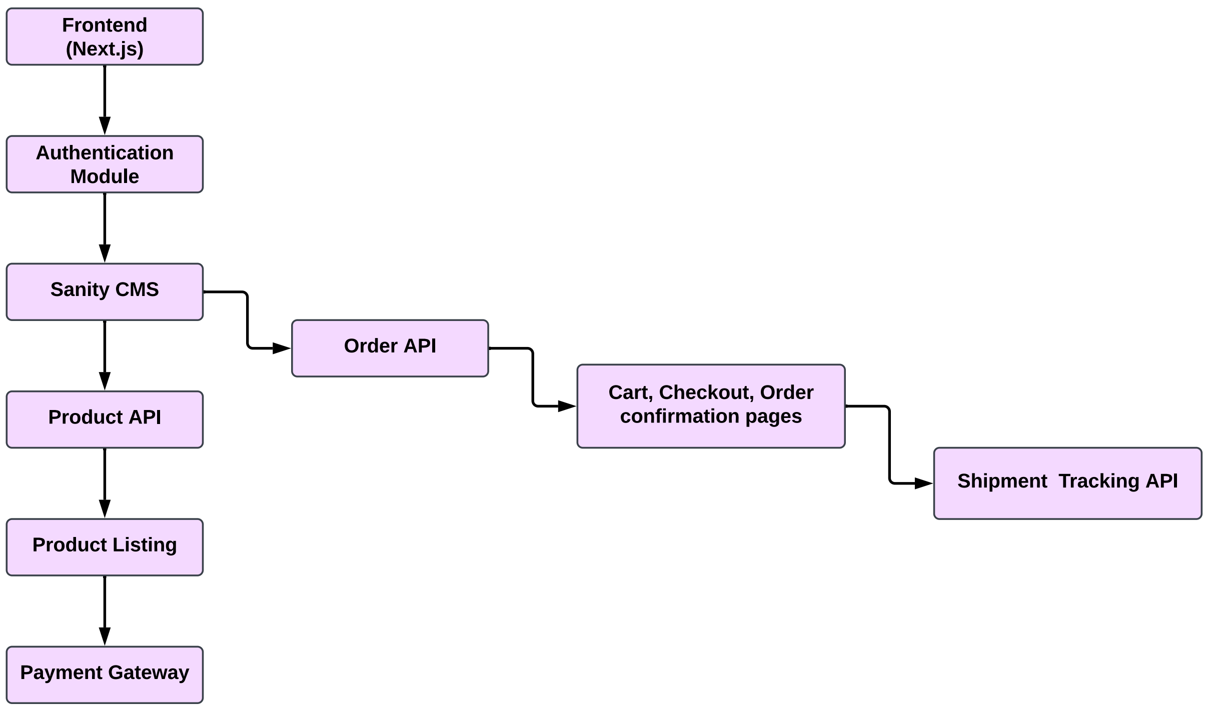
By combining these tools, the system achieves a well-integrated and scalable e-commerce solution.

Frontend	Backend	APIs	Tools
• Next.js	• Sanity CMS	• Stripe	• GitHub
• Tailwind CSS	• Clerk	• Twilio SendGrid	• Postman
• Shadcn/UI			• Vercel

2. **System Architecture**

The system architecture of the E-Commerce website consists of multiple interconnected components to ensure a seamless user experience and efficient backend operations. Below is a high-level diagram of the architecture:

System Architecture



3. **Frontend (Next.js)**

The frontend layer manages the user interface, including responsive design for both desktop and mobile platforms. Key pages include:

- **Home**
- **Product Listing**
- **Product Details**
- **Cart**
- **Checkout**
- **Order Confirmation**

Features:

- Routing with dynamic data fetching.

- User-friendly interface with smooth navigation.
- Responsive design supporting mobile, tablet, and desktop views.

4. **Authentication Module**

The Authentication Module handles user-related operations such as:

- User Registration
- Login/Logout
- Password Recovery
- Session Management

Key Functions:

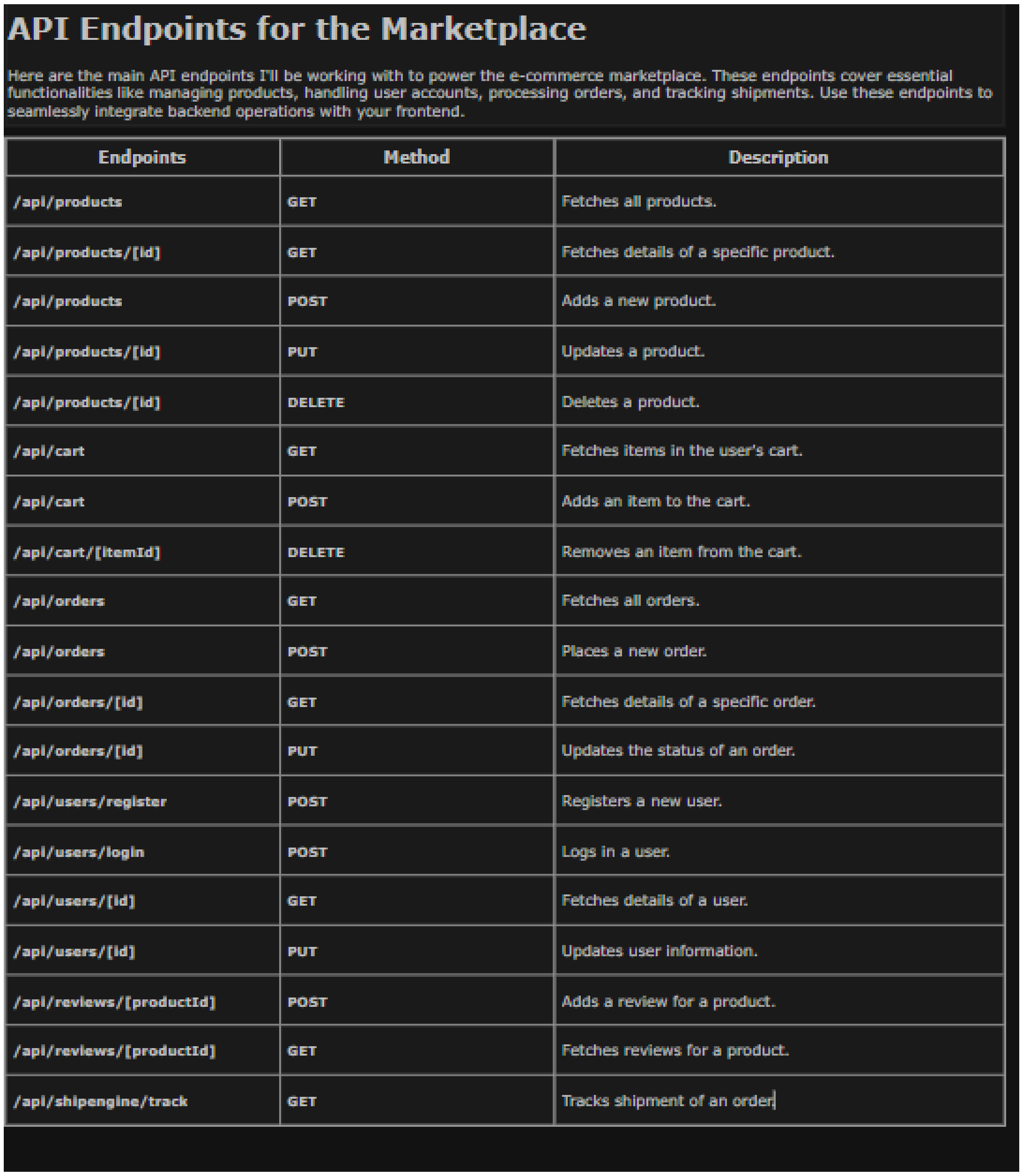
- Securely manages user credentials.
- Verifies user identity for accessing restricted resources.

5. **Sanity CMS**

Sanity CMS acts as the backend database for storing and managing:

- **Products**: Including name, price, description, images, and stock details.
- **Orders**: Including customer information, product details, payment status, and shipment tracking.
- **Users**: Storing information like user credentials, addresses, and order history.

6. **API Endpoints**



API Endpoints for the Marketplace

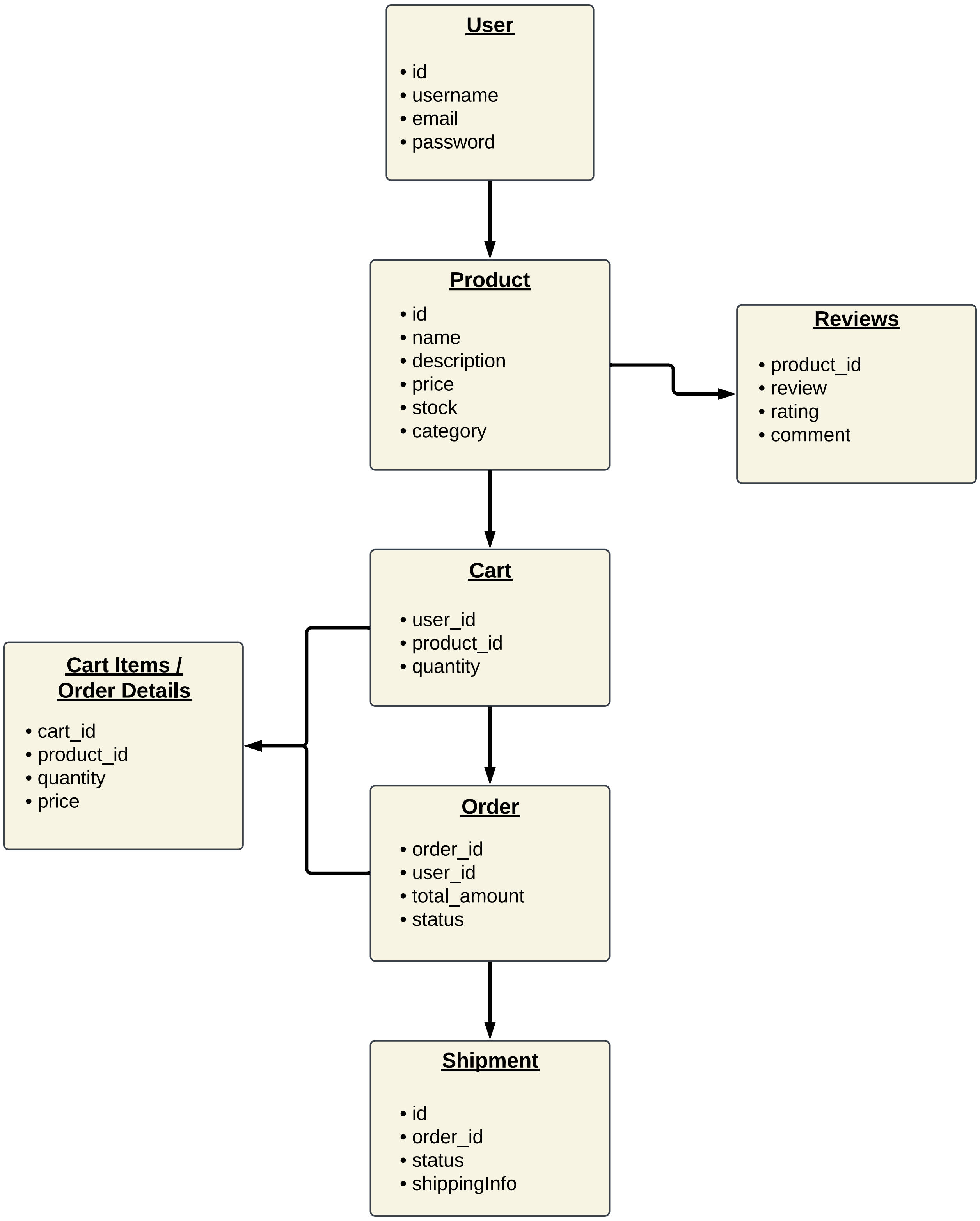
Here are the main API endpoints I'll be working with to power the e-commerce marketplace. These endpoints cover essential functionalities like managing products, handling user accounts, processing orders, and tracking shipments. Use these endpoints to seamlessly integrate backend operations with your frontend.

Endpoints	Method	Description
/api/products	GET	Fetches all products.
/api/products/[id]	GET	Fetches details of a specific product.
/api/products	POST	Adds a new product.
/api/products/[id]	PUT	Updates a product.
/api/products/[id]	DELETE	Deletes a product.
/api/cart	GET	Fetches items in the user's cart.
/api/cart	POST	Adds an item to the cart.
/api/cart/[itemId]	DELETE	Removes an item from the cart.
/api/orders	GET	Fetches all orders.
/api/orders	POST	Places a new order.
/api/orders/[id]	GET	Fetches details of a specific order.
/api/orders/[id]	PUT	Updates the status of an order.
/api/users/register	POST	Registers a new user.
/api/users/login	POST	Logs in a user.
/api/users/[id]	GET	Fetches details of a user.
/api/users/[id]	PUT	Updates user information.
/api/reviews/[productId]	POST	Adds a review for a product.
/api/reviews/[productId]	GET	Fetches reviews for a product.
/api/shipengine/track	GET	Tracks shipment of an order.

This image provides a visual representation of the Product API, Order APIs, Cart/Order and Shipment API showcasing key endpoints and their respective functionalities.

7. User Flow

This User Flow diagram illustrates the steps a user takes when interacting with the marketplace:


User
• id
• username
• email
• password
Product
• id
• name
• description
• price
• stock
• category
Reviews
• product_id
• review
• rating
• comment
Cart
• user_id
• product_id
• quantity
Cart Items / Order Details
• cart_id
• product_id
• quantity
• price
Order
• order_id
• user_id
• total_amount
• status
Shipment
• id
• order_id
• status
• shippingInfo

This technical documentation provides a comprehensive overview of the e-commerce website's architecture, components, and functionalities. It serves as a valuable resource for developers, stakeholders, and team members involved in the development, maintenance, and enhancement of the platform. By following the outlined specifications and leveraging the documented tools and technologies, the team can ensure consistent development practices and maintain a robust, scalable e-commerce solution that meets both business requirements and user expectations.