Marketplace Technical Foundation

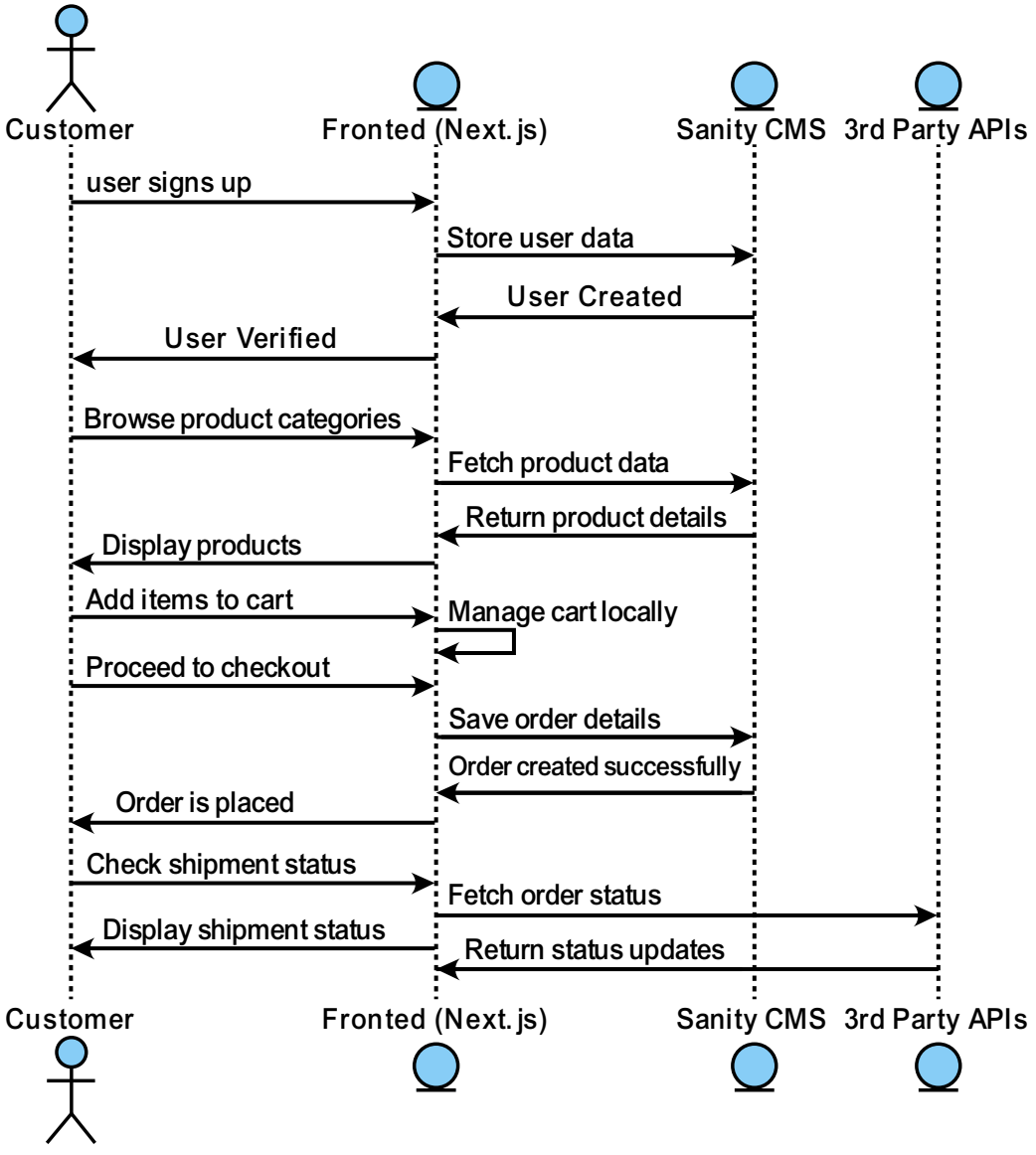
Styleizy

1. System Architecture

The system consists of the following components:

- **Frontend**: Built with Next.js, responsible for rendering pages, managing user interactions, and communicating with the backend and APIs.
- **Backend**: Managed through Sanity CMS to store and handle products, categories, orders, wishlist, and customer data.
- **Third-Party APIs**:
  - **ShipEngine**: For calculating shipping costs and providing real-time shipment tracking.
  - **Stripe**: For secure payment processing.

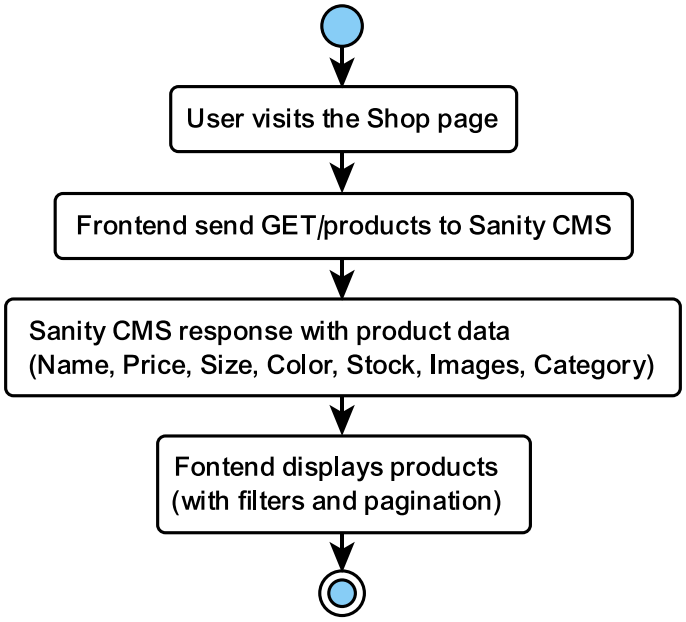
High-Level System Architecture



2. Key Workflows

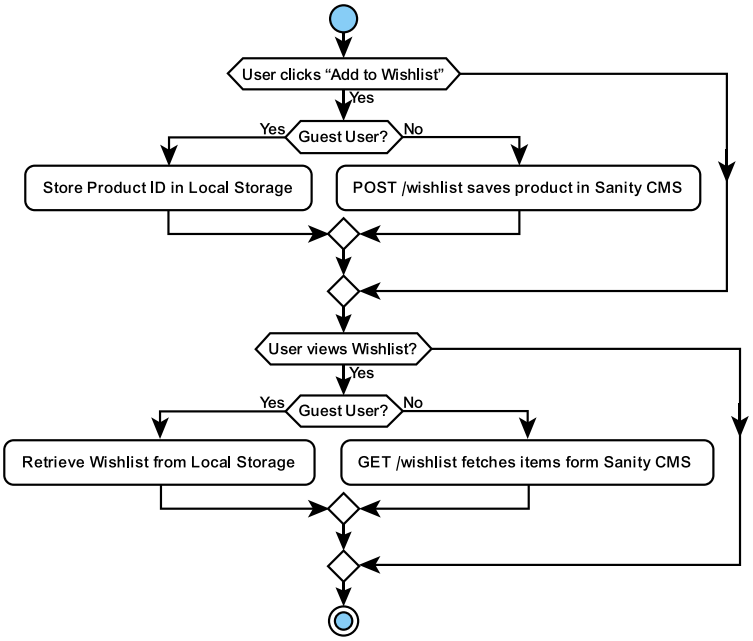
2.1 Product Browsing

1. User visits the **Shop Page**.
2. Frontend sends a GET /products request to Sanity CMS.
3. Sanity CMS responds with product data, including:
    - Name, price, size, color, stock, images, and categories.
4. Frontend displays products with filters and pagination.



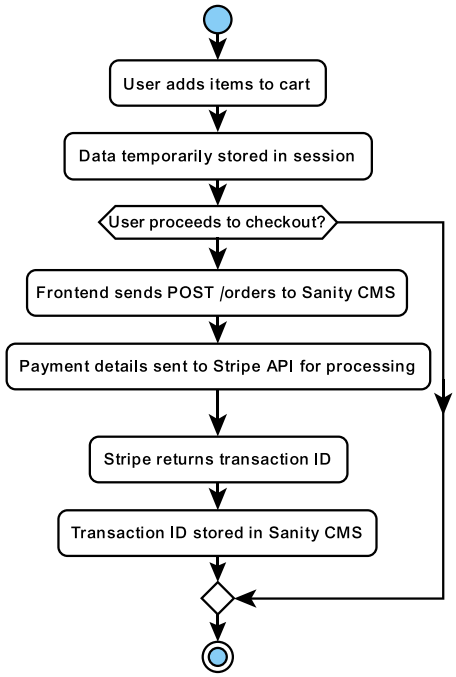
2.2 Wishlist Management

- **Add to Wishlist**:
    - User clicks the **Add to Wishlist** button.
    - For guests: Product ID is stored in local storage.
    - For registered users (future): POST /wishlist request saves the product in Sanity CMS.
- **View Wishlist**:
    - Guest users: Retrieve data from local storage.
    - Registered users: GET /wishlist fetches items from Sanity CMS.



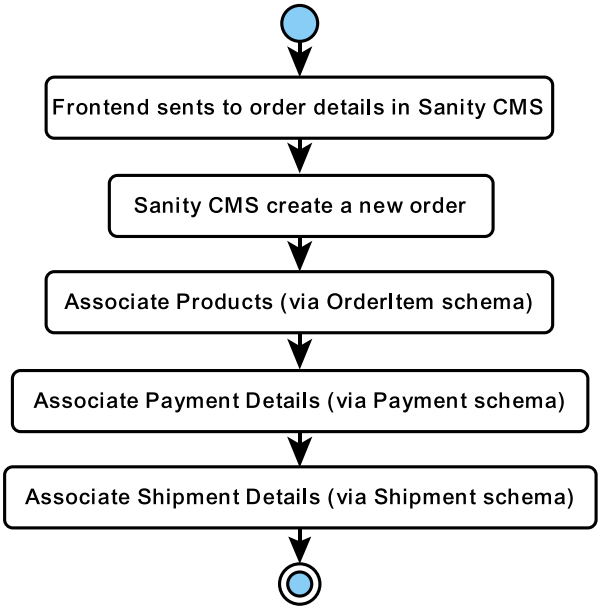
2.3 Cart and Checkout

1. User adds items to the cart:
   - Data temporarily stored in the session.
2. At checkout:
   - Frontend sends POST /orders to Sanity CMS.
   - Payment details are sent to Stripe API for processing.
3. Stripe confirms payment and returns a transaction ID, which is stored in Sanity CMS.



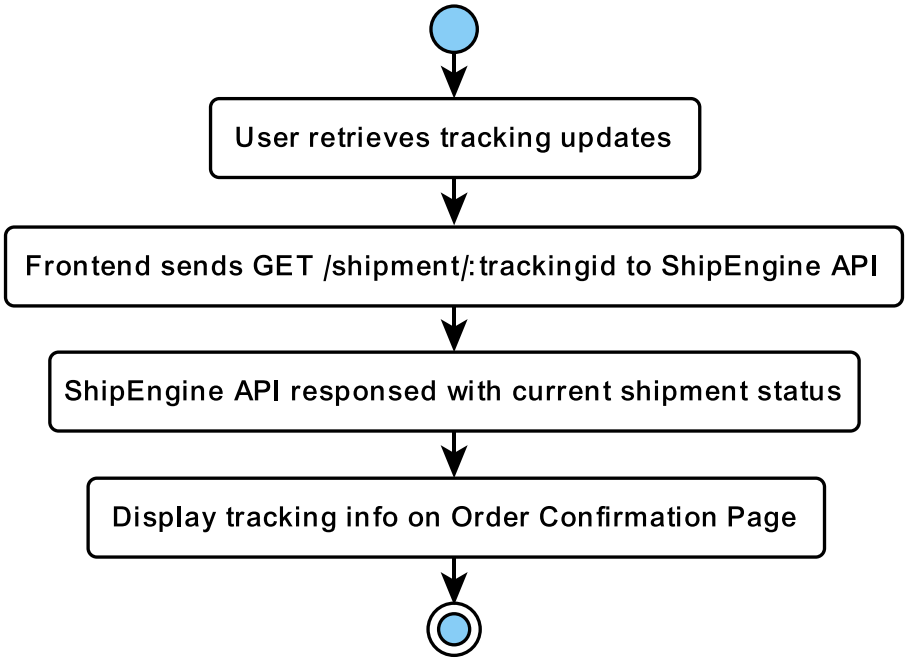
2.4 Order Placement

1. Frontend sends order details to the Orders schema in Sanity CMS.
2. Sanity CMS creates a new order and associates:
   - Products (via OrderItem schema).
   - Payment details (via Payment schema).
   - Shipment details (via Shipment schema).



2.5 Shipment Tracking

1. User retrieves tracking updates:
   - Frontend sends a GET /shipment/:trackingId request.
   - ShipEngine API responds with current shipment status.
2. Tracking information is displayed on the **Order Confirmation Page**.



3. API Endpoints

Endpoint	Method	Purpose	Payload	Response

/products	GET	Fetch all products	None	[
				{
				"id": "1",
				"name": "Classic Sneakers",
				"price": 100,
				"size": ["8", "9", "10"],
				"category": "Casual",
				"stock": 20,
				"images": ["url1", "url2"]
				}
				],

/products/:id	GET	Fetch details for a specific product.	None	{
				"id": "1",
				"name": "Classic Sneakers",
				"price": 100,
				"size": ["8", "9", "10"],
				"category": "Casual",
				"stock": 20,
				"images": ["url1", "url2"]
				}

/orders	POST	Create a new order in Sanity.	{
			"customer": {
			"name": "Jane Doe",
			"email": "jane@example.com",
			"phone": "+123456789",
			"address": "123 Main St, City, Country"
			},
			"orderItems": [
			{
			"productId": "1",
			"quantity": 2
			},
			],
			"paymentId": "pi_abc123",
			"shipment": {
			"carrier": "FedEx",
			"trackingId": "12345"
			}
			}	

/shipments/:id	GET	Track order status via ShipEngine	None	{
				"trackingId": "12345",
				"status": "In Transit",
				"carrier": "FedEx",
				"expectedDeliveryDate": "2025-01-20"
				}

/checkout	POST	Handle payment processing with Stripe and return a success or failure response.	{
			"orderId": "order_123",
			"paymentMethod": "card",
			"cardDetails": {
			"cardNumber": "4242424242424242",
			"expiryDate": "12/25",
			"cvv": "123"
			}
			}
				{
				"status": "success",
				"paymentId": "pi_abc123",
				"message": "Payment processed successfully."
				}

/wishlist	GET	Retrieve saved items for a guest or registered user.	none	[{
				"productId": "1",
				"name": "Classic Sneakers",
				"price": 100,
				"image": "url1"
				}
				]

/wishlist	POST	Add a product to the wishlist.	{
			"productId": "1"
			}
				{
				"status": "success",
				"message": "Product added to wishlist."
				}

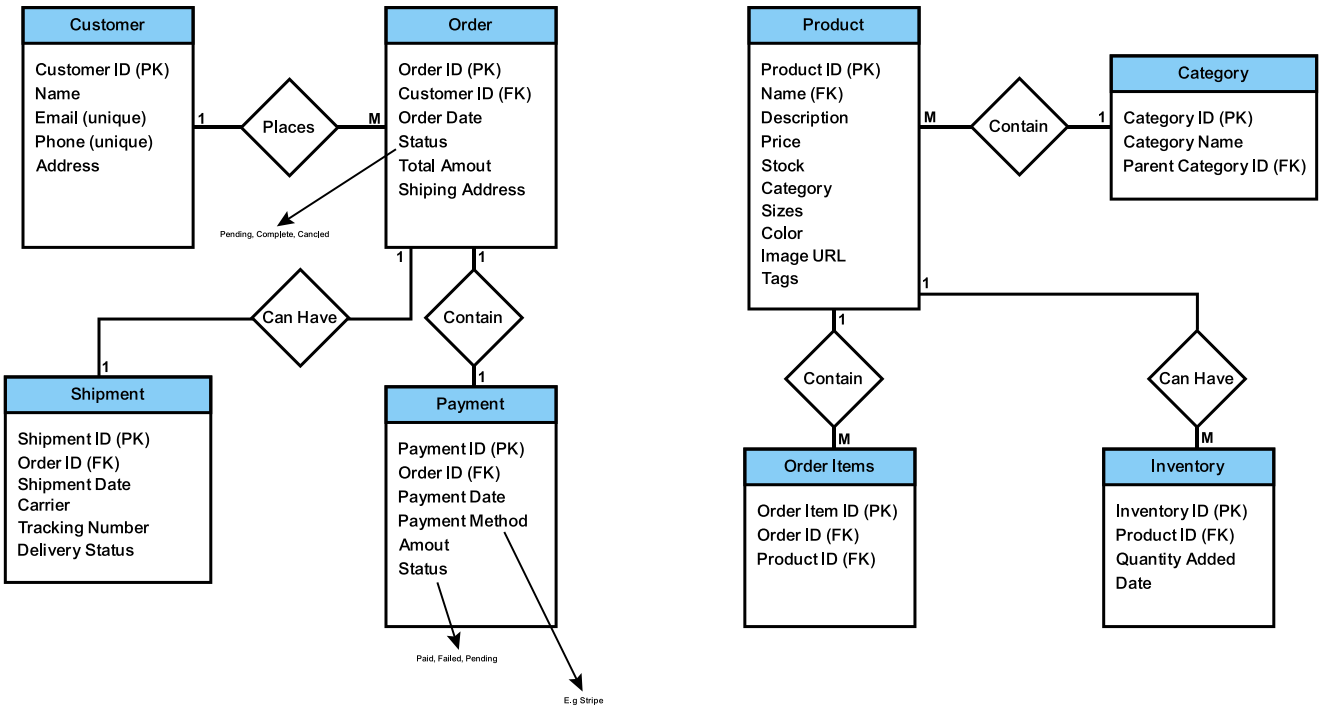
/wishlist/:id	DELETE	Remove a product from the wishlist.	None	{
				"status": "success",
				"message": "Product removed from wishlist."
				}

/categories	GET	Fetch a list of all product categories.	None	[{ "id": "1", "name": "Sneakers", "slug": "sneakers", "description": "Casual and comfortable footwear", "image": "url-to-category-image" },]
/categories/:id	GET	Fetch details of a specific category by its ID.	None	{ "id": "1", "name": "Sneakers", "slug": "sneakers", "description": "Casual and comfortable footwear", "image": "url-to-category-image" },

4. Data Schemas

- ER Diagram to represent Schemas/Entites and their relationships

Detailed ER (Entity-RelationShip) Diagram



Schemas design for entities

Products

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Name' },
    { name: 'description', type: 'text', title: 'Description' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock' },
    { name: 'sizes', type: 'array', of: [{ type: 'string' }], title: 'Sizes' },
    { name: 'categories', type: 'reference', to: [{ type: 'category' }], title: 'Categories' },
    { name: 'images', type: 'array', of: [{ type: 'image' }], title: 'Images' },
  ],
};
```

Category

```
export default {
  name: 'category',
  type: 'document',
  title: 'Category',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Category Name',
      validation: (Rule) => Rule.required(),
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
    },
    {
      name: 'imageUrl',
      type: 'url',
      title: 'Category Image',
    },
  ],
};
```

Customer

```
export default {
  name: 'customer',
  type: 'document',
  title: 'Customer',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Name',
      validation: (Rule) => Rule.required(),
    },
    {
      name: 'email',
      type: 'string',
      title: 'Email',
      validation: (Rule) => Rule.required().email(),
    },
    {
      name: 'phone',
      type: 'string',
      title: 'Phone Number',
    },
```

```
    {
      name: 'address',
      type: 'object',
      title: 'Address',
      fields: [
        { name: 'street', type: 'string', title: 'Street' },
        { name: 'city', type: 'string', title: 'City' },
        { name: 'state', type: 'string', title: 'State' },
        { name: 'zip', type: 'string', title: 'ZIP Code' },
        { name: 'country', type: 'string', title: 'Country' },
      ],
    },
  ],
};
```

Orders

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'customer', type: 'reference', to: [{ type: 'customer' }], title: 'Customer' },
    { name: 'orderItems', type: 'array', of: [{ type: 'reference', to: [{ type: 'orderItem' }] }], title: 'Order
Items' },
    { name: 'payment', type: 'reference', to: [{ type: 'payment' }], title: 'Payment' },
    { name: 'shipment', type: 'reference', to: [{ type: 'shipment' }], title: 'Shipment' },
    { name: 'status', type: 'string', title: 'Status' },
  ],
};
```

Order Items

```
export default {
  name: 'orderItems',
  type: 'document',
  title: 'Order Items',
  fields: [
    {
      name: 'orderId',
      type: 'reference',
      to: [{ type: 'order' }],
      title: 'Order ID',
      validation: (Rule) => Rule.required(),
    },
    {
      name: 'productId',
      type: 'reference',
      to: [{ type: 'product' }],
      title: 'Product ID',
      validation: (Rule) => Rule.required(),
```

```
    },
    {
      name: 'quantity',
      type: 'number',
      title: 'Quantity',
      validation: (Rule) => Rule.min(1).required(),
    },
    {
      name: 'price',
      type: 'number',
      title: 'Price Per Unit',
      validation: (Rule) => Rule.min(0).required(),
    },
  ],
};
```

Inventory

```
export default {
  name: 'inventory',
  type: 'document',
  title: 'Inventory',
  fields: [
    {
      name: 'productId',
      type: 'reference',
      to: [{ type: 'product' }],
      title: 'Product ID',
      validation: (Rule) => Rule.required(),
    },
    {
      name: 'quantity',
      type: 'number',
      title: 'Quantity',
      validation: (Rule) => Rule.min(0).required(),
    },
    {
      name: 'lastUpdated',
      type: 'datetime',
      title: 'Last Updated',
      validation: (Rule) => Rule.required(),
    },
  ],
};
```

Payment

```
export default {
  name: 'payment',
  type: 'document',
  fields: [
```

```
    { name: 'paymentId', type: 'string', title: 'Payment ID' },
    { name: 'amount', type: 'number', title: 'Amount' },
    { name: 'status', type: 'string', title: 'Status' },
    { name: 'method', type: 'string', title: 'Payment Method' },
  ],
};
```

5. Technical Roadmap

Milestone 1: Complete schema definitions in Sanity CMS.

Milestone 2: Build frontend components with Next.js. (Already completed)

Milestone 3: Fetch data in frontend (Next js)

Milestone 4: Integrate ShipEngine and Stripe APIs.

Milestone 5: Test real-time features for cart updates and shipment tracking.