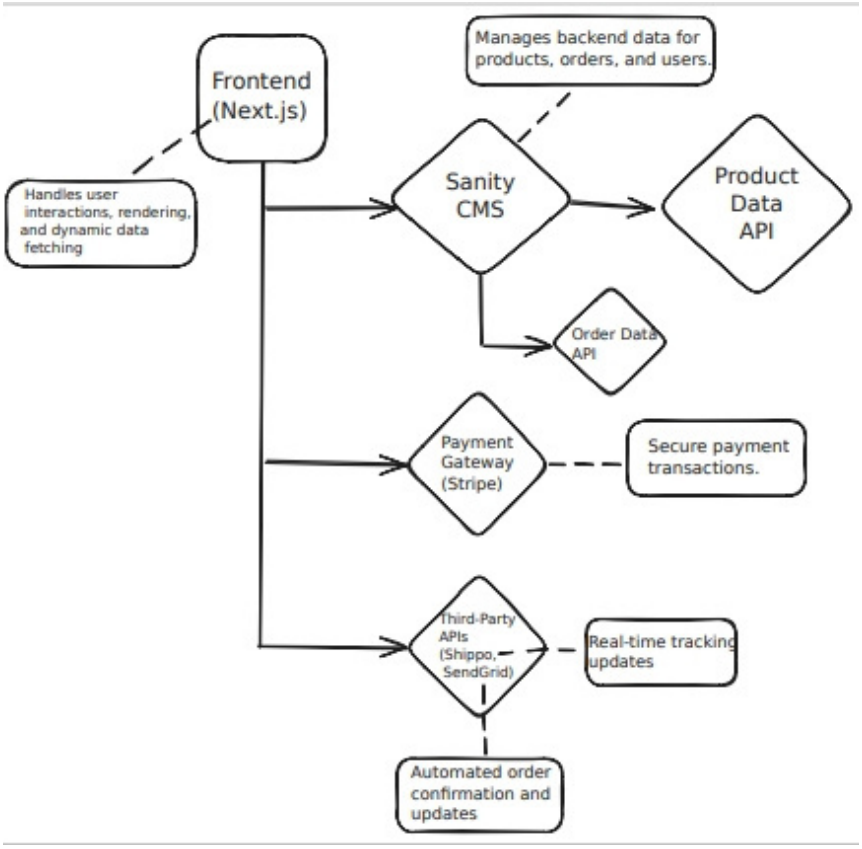
Marketplace Technical Foundation - (E-Commerce)

1. System Architecture Overview

Diagram



Component Descriptions

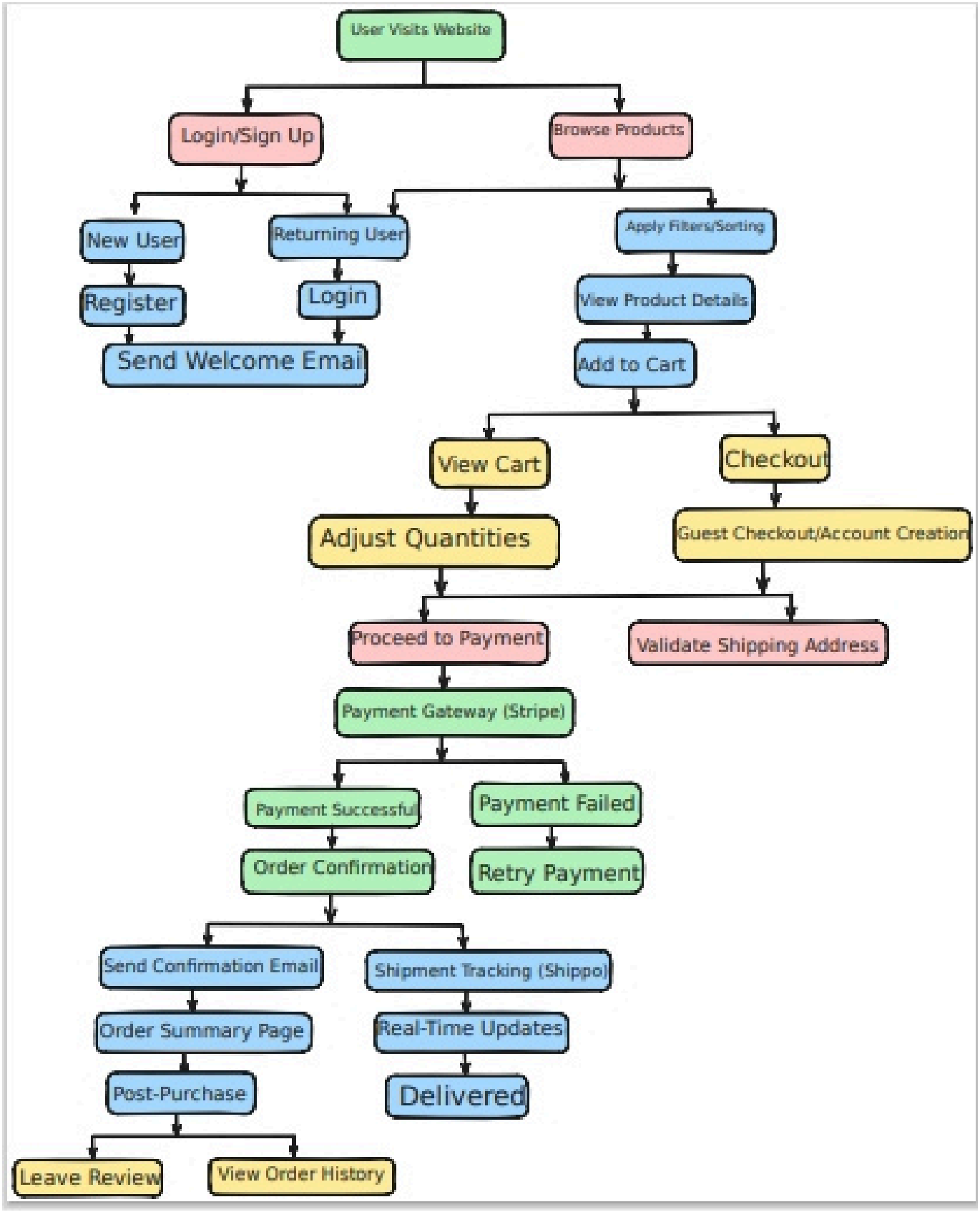
1. Frontend (Next.js): Provides a user-friendly interface for browsing products, managing carts, and completing orders.
2. Sanity CMS: Manages product data, customer details, and order records. Acts as the central database for the marketplace.
3. Third-Party APIs: Integrates services for shipment tracking, payment processing, and other backend functionalities.
4. Payment Gateway: Handles secure payment processing and confirmation.

2. Key Workflows

1. **User Registration:**
   - Step 1: User signs up through the frontend.
   - Step 2: User data is stored in Sanity CMS.
   - Step 3: Confirmation email is sent to the user.
2. **Product Browsing:**
   - Step 1: User selects a product category.
   - Step 2: Sanity CMS API fetches product details.
   - Step 3: Products are dynamically displayed on the frontend.

3. **Order Placement:**
    - Step 1: User adds items to the cart and proceeds to checkout.
    - Step 2: Order details are sent to Sanity CMS.
    - Step 3: Payment is processed via the Payment Gateway.
    - Step 4: Confirmation is sent to the user.
4. **Shipment Tracking:**
    - Step 1: Order status updates are fetched via a third-party API.
    - Step 2: Real-time status is displayed to the user.

Diagram



3. API Requirements

Endpoint	Method	Purpose	Payload/Response
/products	GET	Fetch all product details	{ id, name, price, stock, image }
/orders	POST	Create a new order in Sanity	{ customerInfo, productDetails, paymentStatus }
/shipment	GET	Track order status via third-party API	{ shipmentId, orderId, status, expectedDeliveryDate }
/express-delivery-status	GET	Fetch real-time delivery updates for perishable items	{ orderId, status, ETA }
/rental-duration	POST	Add rental details for a specific product	{ productId, duration, deposit }

4. Sanity Schema Examples

Product Schema

```
export default {
        name: 'product',
        type: 'document',
        fields: [
   { name: 'name', type: 'string', title: 'Product Name' },
   { name: 'price', type: 'number', title: 'Price' },
   { name: 'stock', type: 'number', title: 'Stock Level' },
   { name: 'image', type: 'image', title: 'Product Image' }
 ]
};
```

Order Schema

```
export default {
 name: 'order',
 type: 'document',
 fields: [
  { name: 'customerInfo', type: 'object', fields: [
```

```
    { name: 'name', type: 'string', title: 'Customer Name' },
    { name: 'email', type: 'string', title: 'Customer Email' }
  ]},
  { name: 'productDetails', type: 'array', of: [{ type: 'reference', to: { type: 'product' } }], title: 'Products' },
  { name: 'paymentStatus', type: 'string', title: 'Payment Status' }
 ]
};
```

5. Collaborative Feedback and Refinement

1. Group Discussions:
   - Share ideas for system architecture and API designs.
2. Peer Reviews:
   - Exchange feedback with teammates to refine workflows and documentation.
3. Version Control:
   - Use GitHub to track changes and maintain a transparent version history.

Key Outcomes

1. Technical Plan Aligned with Business Goals:
   - A clear roadmap reflecting marketplace-specific requirements.
2. System Architecture Visualized:
   - Detailed diagrams showing component interactions.
3. API Requirements Documented:
   - Well-defined endpoints with example payloads and responses.
4. Sanity Schemas Drafted:
   - Schema designs for handling products, orders, and customers.
5. Portfolio-Ready Submission:
   - A polished technical document suitable for professional presentations.