DAY 2 PLANNING THE TECHNICAL FOUNDATION

Frontend Requirements:

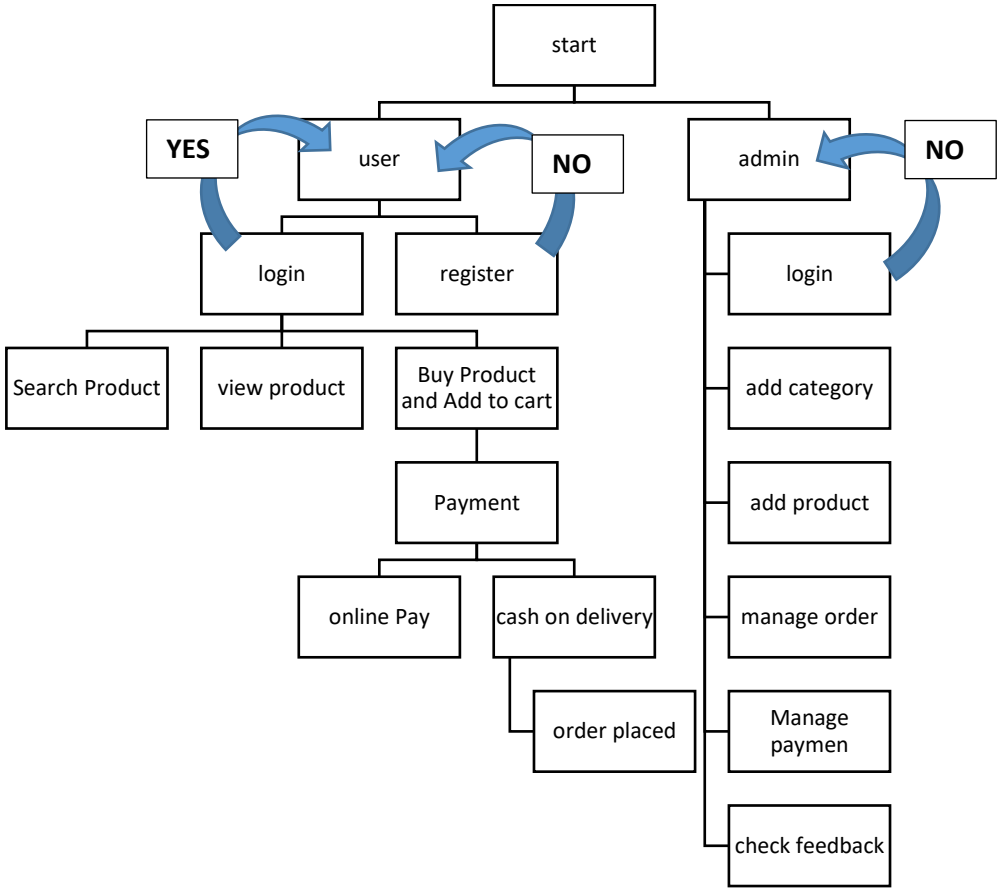
- **User-friendly interface** for browsing and shopping.
- **Responsive design** for seamless experience on both mobile and desktop.
- **Key Pages**: Home, Product Listing, Product Details, Cart, Checkout, Order Confirmation.

Sanity CMS as Backend:

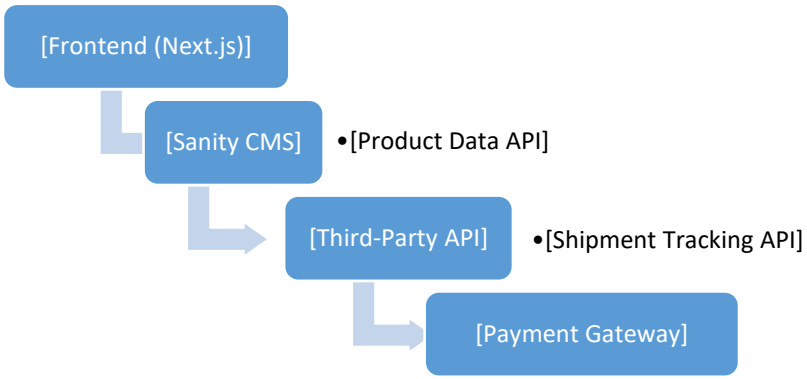
- Use **Sanity CMS** to manage **product data**, **customer details**, and **order records**.
- **Schema design** in Sanity should align with business goals from the start.

Third-Party APIs:

- **API Integrations**: Include shipment tracking, payment gateways, and other backend services.
- Ensure APIs provide required data for frontend functionality (e.g., order updates, payment status).



2. Design System Architecture:



3. Plan API Requirements:

Endpoint Name	Method	Description	Payload/Response Example
/express-delivery-status	GET	Fetch real-time delivery updates for perishable items.	**Response:** `{ "orderId": 123, "status": "In Transit", "ETA": "15 mins" }`
/rental-duration	POST	Add rental details for a specific product.	**Payload:** `{ "productId": 456, "duration": "7 days", "deposit": 500 }` **Response:** `{ "confirmationId": 789, "status": "Success" }`
/products	GET	Fetch all product details.	**Response:** `{ "id": 1, "name": "Product A", "price": 100 }`
/products	GET	Fetch all available products from Sanity.	**Response:** `{ "id": 101, "name": "T-shirt", "price": 500, "stock": 10, "image": "image_url" }`
/orders	POST	Create a new order in Sanity.	**Payload:** `{ "customerName": "John Doe", "productId": 101, "paymentStatus": "Paid" }` **Response:** `{ "orderId": 12345, "status": "Success" }`
/shipment	GET	Track order status via third-party API.	**Response:** `{ "shipmentId": 6789, "orderId": 12345, "status": "Shipped", "expectedDelivery": "2025-01-20" }`

4·Technical Documentation:

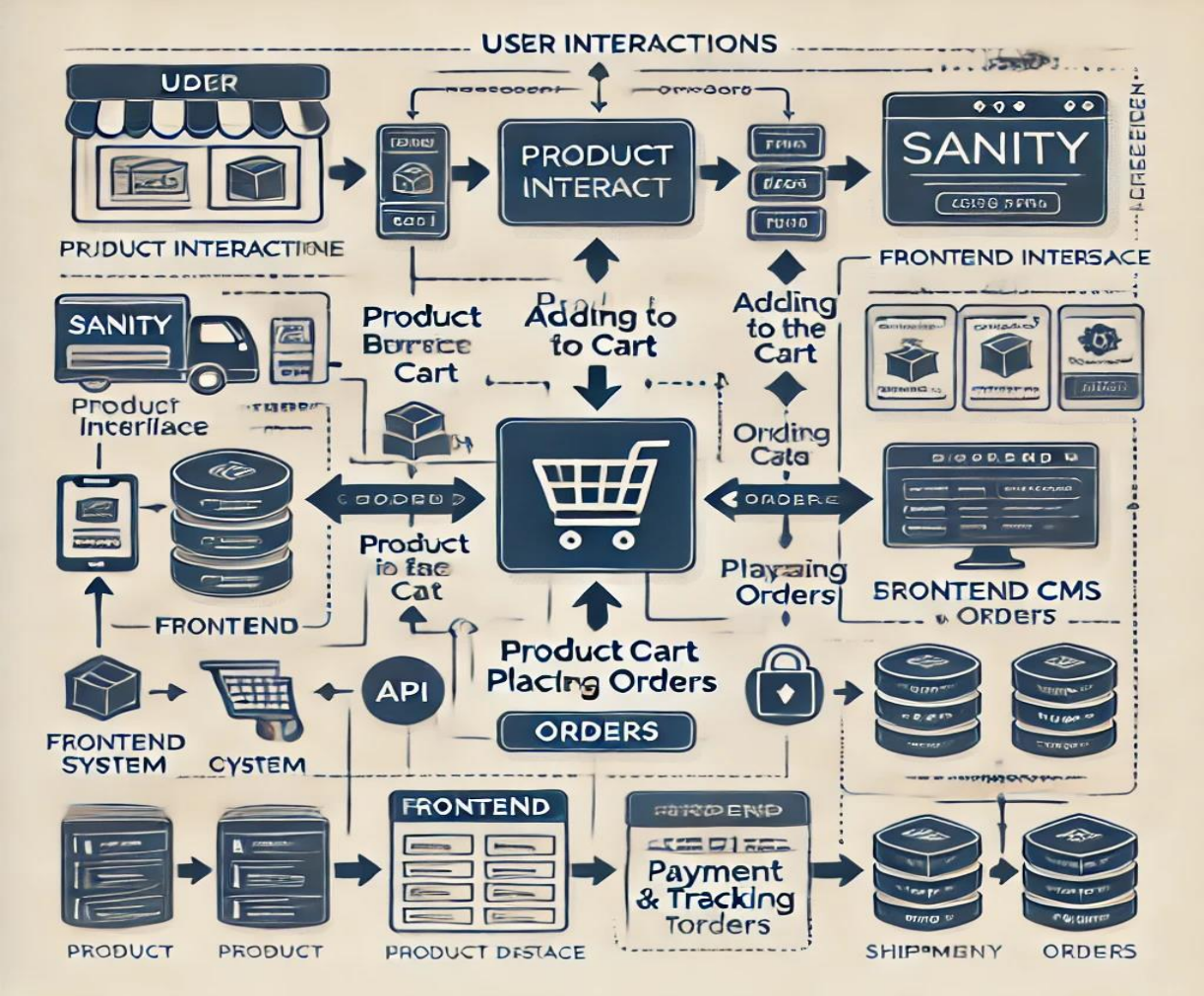
A professional flowchart showing the technical architecture and workflows for an e-commerce marketplace. The flowchart includes these components:

1. User interactions (product browsing, adding to cart, placing orders),

2. Frontend interface (website or app) that connects to the backend via API calls,

3. Backend system interacting with Sanity CMS for product data and a database for orders,

4. Third-party APIs for payment processing and shipment tracking. Arrows indicate the flow of data between components. The design is clean and clear, with labeled boxes for each step and arrows for direction, using a simple and modern style.



Here is the flowchart illustrating the technical architecture and workflows for the e-commerce marketplace. It visualizes the interactions and data flow between the user, frontend, backend, and third-party APIs for clarity.

5. Collaborating and Refining our Technical Plan:

Here are the important points:

1. **Group Discussions**:
   - Brainstorm with your peers to solve challenges and improve ideas.
   - Use tools like Slack, Google Meet, or Discord for communication.
   - Focus on innovating system architecture and API design.
2. **Peer Review**:
   - Share your plans with others for constructive feedback.
   - Review data schemas, workflows, and documentation for improvements.
3. **Version Control**:
   - Use GitHub or similar tools to track changes.
   - Commit updates regularly and write clear commit messages.
4. **Divide Tasks**:
   - Work as a team for brainstorming, but each member should prepare their own document for submission.
   - Maintain individuality within a shared framework.
5. **Individual Submission**:
   - Ensure your final submission reflects your personal understanding and approach.

Sanity CMS Schema for Rental E-commerce (Men's Clothing):

1. Product Schema

This schema defines the clothing items available for rent.

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' }, // e.g., "Formal Suit"
    { name: 'category', type: 'string', title: 'Category' }, // e.g., "Suits", "Casual Wear"
    { name: 'size', type: 'array', title: 'Available Sizes', of: [{ type: 'string' }] }, // e.g., ["S", "M", "L", "XL"]
    { name: 'price', type: 'number', title: 'Rental Price per Day' },
    { name: 'deposit', type: 'number', title: 'Security Deposit' },
    { name: 'stock', type: 'number', title: 'Stock Quantity' },
    { name: 'images', type: 'array', title: 'Product Images', of: [{ type: 'image' }] },
    { name: 'description', type: 'text', title: 'Product Description' },
    { name: 'condition', type: 'string', title: 'Condition' }, // e.g., "New", "Like New", "Good"
  ],
};
```

2. Order Schema:

Tracks user orders and rental details.

```
export default {
  name: 'order',
  type: 'document',
  title: 'Order',
  fields: [
    { name: 'orderId', type: 'string', title: 'Order ID' },
    { name: 'customerName', type: 'string', title: 'Customer Name' },
    { name: 'customerContact', type: 'string', title: 'Customer Contact' },
    { name: 'product', type: 'reference', title: 'Product', to: [{ type: 'product' }] },
    { name: 'rentalDuration', type: 'string', title: 'Rental Duration' }, // e.g., "7 days"
    { name: 'startDate', type: 'datetime', title: 'Rental Start Date' },
    { name: 'endDate', type: 'datetime', title: 'Rental End Date' },
    { name: 'totalPrice', type: 'number', title: 'Total Price' },
    { name: 'depositPaid', type: 'boolean', title: 'Deposit Paid' },
    { name: 'status', type: 'string', title: 'Order Status' }, // e.g., "Pending", "Approved", "Completed"
  ],
};
```

3. Customer Schema:

Stores customer details for rental history and communication.

```
export default {
  name: 'customer',
  type: 'document',
  title: 'Customer',
  fields: [
    { name: 'name', type: 'string', title: 'Full Name' },
    { name: 'email', type: 'string', title: 'Email Address' },
    { name: 'phone', type: 'string', title: 'Phone Number' },
    { name: 'address', type: 'text', title: 'Address' },
    { name: 'orderHistory', type: 'array', title: 'Order History', of: [{ type: 'reference', to: [{ type: 'order' }] }] },
  ],
};
```

4. Review Schema:

Allows customers to leave reviews for rented products.

```
export default {
  name: 'review',
  type: 'document',
  title: 'Review',
  fields: [
    { name: 'product', type: 'reference', title: 'Product', to: [{ type: 'product' }] },
    { name: 'customer', type: 'reference', title: 'Customer', to: [{ type: 'customer' }] },
    { name: 'rating', type: 'number', title: 'Rating', validation: Rule => Rule.min(1).max(5) },
    { name: 'comment', type: 'text', title: 'Comment' },
    { name: 'date', type: 'datetime', title: 'Review Date' },
  ],
};
```

SCHEMA WORKS:

- **Product Schema** manages the clothing inventory.
- **Order Schema** handles rental transactions and tracking.
- **Customer Schema** saves user information for personalized service.
- **Review Schema** enables user feedback for products.