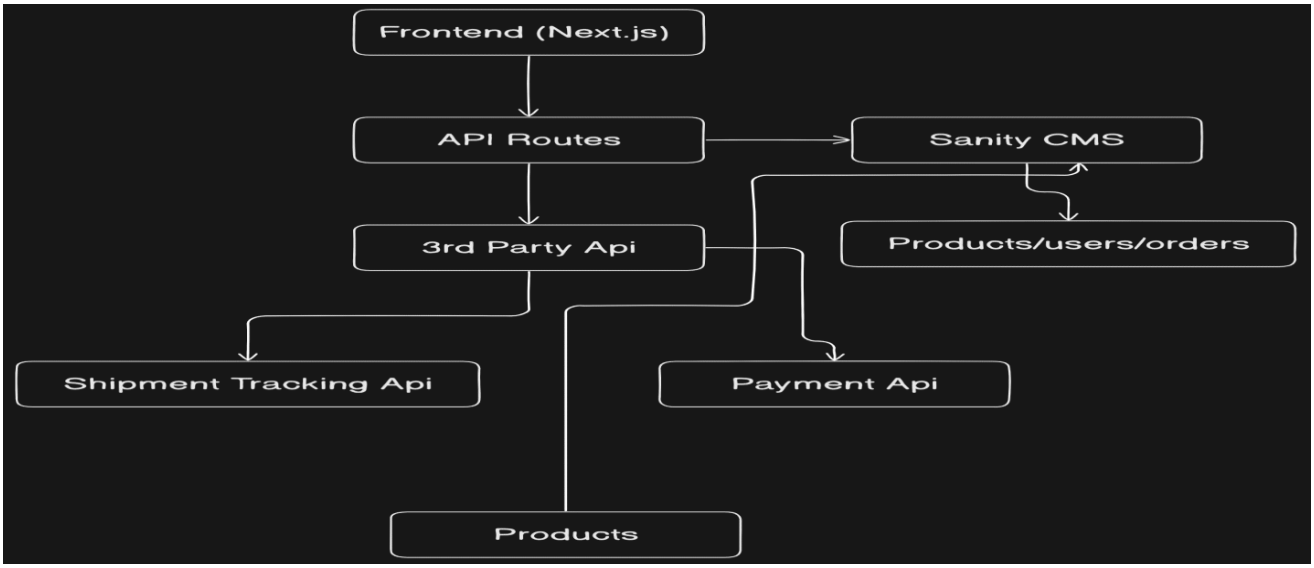
Marketplace Technical Foundation - Comforty

1. System Architecture Overview

System Architecture Diagram



How It Works:

1. **Frontend (Next.js):**
   - Handles pages for login, product browsing, cart management, checkout, and order tracking.
2. **Authentication System (NextAuth.js or Firebase):**
   - Provides user login, logout, and registration functionality.
   - Ensures secure access to user-specific features like order history and saved carts.
3. **Sanity CMS (Backend):**
   - Manages product details, categories, stock, orders, and user profiles.
4. **API Routes in Next.js:**
   - Connects the frontend with Sanity CMS and third-party APIs for cart operations, payments, and delivery tracking.
5. **Third-Party APIs:**
   - **Payment API:** Processes online payments securely.
   - **Delivery Tracking API:** Tracks real-time delivery status for orders.

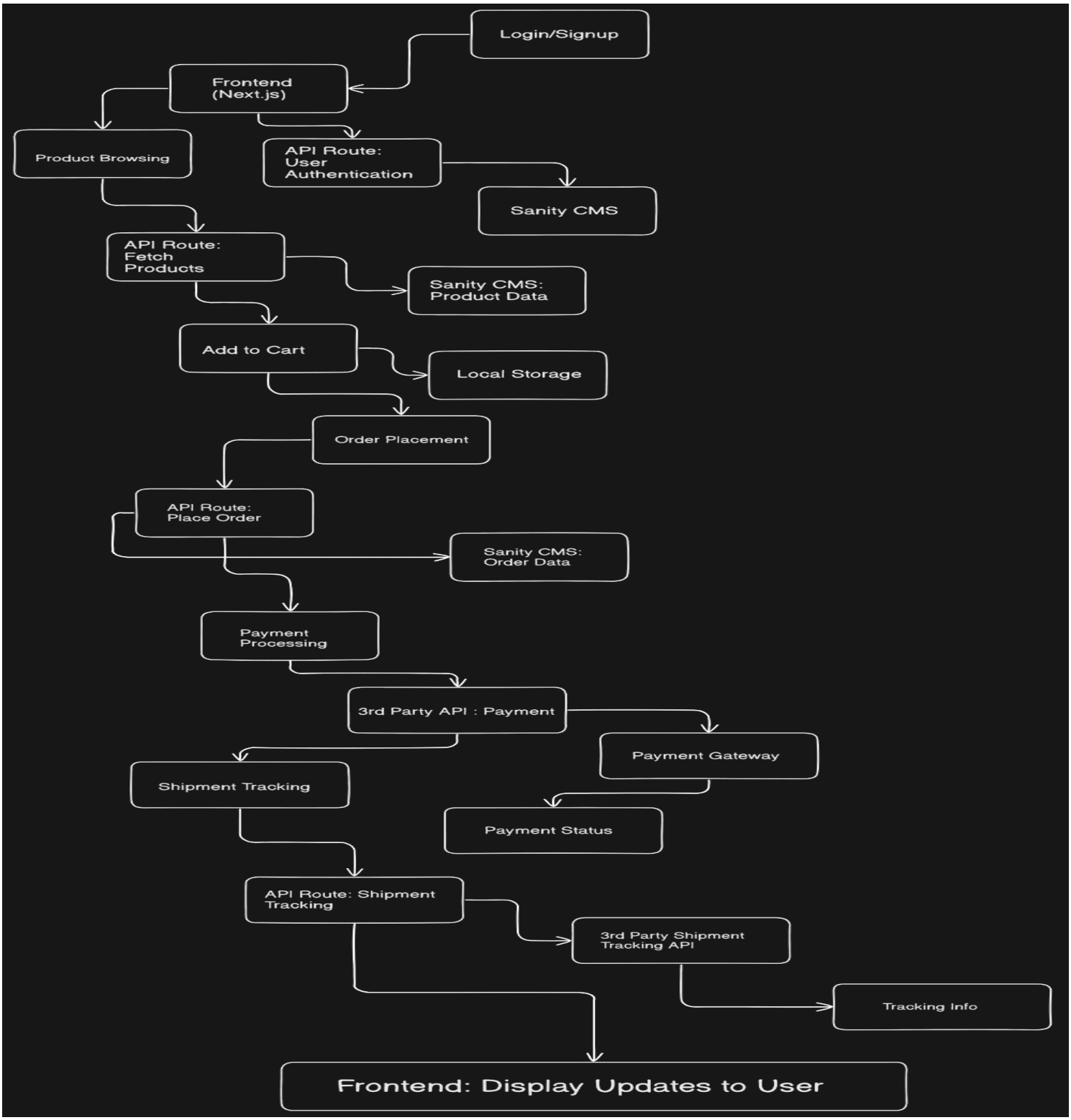
2. Key Workflows

1. **User Registration & Login:**
    - **Registration:**
        - User signs up with email/password or third-party login (e.g., Google).
        - Profile is saved in Sanity CMS with their name, email, and password.
    - **Login:**
        - User logs in to access their dashboard, saved carts, and order history.
2. **Browsing Products:**
    - Products are fetched from Sanity CMS and displayed with category filters (Essential, Deluxe, Premium).
3. **Adding Products to Cart:**
    - Guest Users: Cart items are stored locally (in browser storage).
    - Logged-In Users: Cart items are stored in Sanity CMS and synced across sessions.
4. **Placing an Order:**
    - At checkout, the system validates stock availability and processes payment via the Payment API.
    - Order details are saved in Sanity CMS under the user's profile.
5. **Tracking Orders:**
    - Logged-in users can view all their orders in the dashboard.
    - Guests can track orders using the order ID and email.

3. Api endpoints

Endpoint	Method	Purpose	Response
/api/register	POST	Registers a new user	{ "success": true, "userId": "123" }
/api/login	POST	Logs in a user	{ "success": true, "token": "abcd1234" }
/api/logout	POST	Logs out the user	{ "success": true }
/api/products	GET	Fetch all product details	{ "id": 1, "name": "Essential Chair", "price": 100, "stock": 10 }
/api/cart	POST	Adds items to the cart	{ "success": true, "cart": [{ "productID": 1, "quantity": 2 }] }
/api/checkout	POST	Processes payment and creates order	{ "orderID": "ORD123", "status": "Payment Successful", "total": 200 }
/api/delivery-status	GET	Tracks delivery status	{ "orderID": "ORD123", "status": "Out for Delivery", "estimatedDelivery": "2025-01-18" }
/api/user/orders	GET	Fetches the logged-in user's orders	{ "orders": [{ "orderId": "ORD123", "status": "Delivered" }] }

Workflow Diagram



4. Sanity Schema Example

User Schema

```
import { defineType, defineField } from 'sanity'

export const userSchema = defineType({
  name: 'user',
  title: 'User',
  type: 'document',
  fields: [
    defineField({
      name: 'userId',
      type: 'string',
      title: 'User ID',
    }),
    defineField({
      name: 'email',
      type: 'string',
      title: 'Email Address',
    }),
    defineField({
      name: 'orders',
      type: 'array',
      title: 'Orders',
      of: [{ type: 'reference', to: [{ type: 'order' }] }],
    }),
  ],
})
```

Product Schema

```
export const productSchema = defineType({
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
      defineField({
        name: 'name',
        type: 'string',
        title: 'Product Name',
      }),
      defineField({
        name: 'price',
        type: 'number',
        title: 'Price',
      }),
      defineField({
        name: 'stock',
        type: 'number',
        title: 'Stock Quantity',
      }),
      defineField({
        name: 'description',
        type: 'text',
        title: 'Product Description',
      }),
    ],
  })
```

5. Technical Roadmap

Steps to Implement

1. **Authentication:**
    - Integrate **NextAuth.js** for secure login/logout functionality.
    - Create protected routes for user dashboards (e.g., /dashboard).
2. **Backend Setup (Sanity CMS):**
    - Add schemas for products, orders, and users.
    - Set up GROQ queries to fetch user-specific data.
3. **Frontend Development (Next.js):**
    - Build login and registration forms.
    - Create a **dashboard page** where logged-in users can view their orders and saved carts.
    - Add order tracking functionality.
4. **API Development:**
    - Implement secure API routes for authentication, order management, and delivery tracking.
5. **Testing & Deployment:**
    - Test workflows end-to-end (login, cart, checkout, and tracking).
    - Deploy the app to **Vercel** for scalability.