Ras Healthcare Marketplace: Technical Foundation Document

Day 2 Goal

Transitioning from the business-focused planning of Day 1 to establishing the technical groundwork for Ras Healthcare. This stage outlines the system architecture, workflows, and API requirements that align with our business objectives.

Recap of Day 1

1. **Business Goals Defined:**
    - Problem: Ensure reliable access to high-quality healthcare supplements.
    - Target Audience: All age groups in Pakistan.
    - Unique Value Proposition: Trustworthy, affordable, and accessible healthcare products.
2. **Data Schema Drafted:**
    - Entities: Products, Orders, Customers, Shipments, and Delivery Zones.
3. **Solid Foundation:**
    - Focused on business requirements to streamline technical planning.

Technical Requirements

Frontend Framework & Core

- **Next.js 15.0.0:** React framework with App Router.
- **React 19.0.0 (RC version):** UI library.
- **TypeScript:** Programming language for type safety and scalability.

Styling & UI

- **Tailwind CSS 3.4.1:** Utility-first CSS framework.
- **Radix UI Components:** Dialog, Avatar, Dropdown Menu, Label, Popover, Scroll Area, Select, Switch, Tabs, Toggle.
- **GSAP:** Animation and effects platform with React integration.
- **Lenis:** Smooth scrolling.
- **Embla Carousel:** Carousel creation.

Form Management & Validation

- **react-hook-form:** Form management.
- **zod:** Schema validation.

Rich Text Editing

- **TipTap Editor:** Includes extensions for character count, highlights, links, and task lists.

Data Management & API

- **@tanstack/react-query:** Data fetching and caching.
- **Appwrite:** Backend as a Service for managing database and user authentication.
- **Trax:** Future shipment tracking integration.

Utilities & Helpers

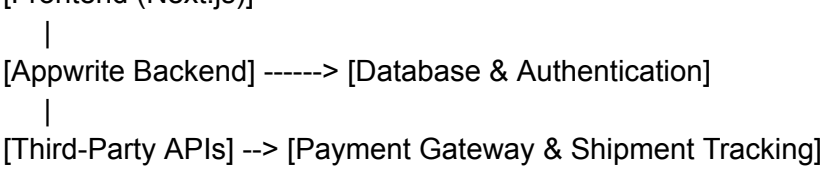
- **date-fns:** Date manipulation.
- **clsx & tailwind-merge:** Class name utilities.
- **lucide-react:** Icon library.

Development Tools

- **ESLint:** Code linting.
- **TypeScript configuration:** Ensures adherence to best practices.

System Architecture

High-Level Overview



Workflow Examples

1. **Product Browsing:**
    - User browses the catalog.
    - Frontend fetches product details from Appwrite.
2. **Order Placement:**
    - User adds items to cart and proceeds to checkout.
    - Order details saved in Appwrite.
3. **Shipment Tracking:**
    - Shipment status retrieved via Trax API.

    - Real-time updates displayed to the user.
4. **Payment Processing:**
    - Payment details processed securely via a gateway.
    - Confirmation recorded in Appwrite.

API Requirements

General Endpoints

1. **Products**
    - **Endpoint:** `/products`
    - **Method:** GET
    - **Description:** Fetch all available products.
    - **Response:** `{ "id": 1, "name": "Multivitamin", "price": 1500, "stock": 50 }`
2. **Orders**
    - **Endpoint:** `/orders`
    - **Method:** POST
    - **Description:** Create a new order.
    - **Payload:** `{ "customerId": 101, "productIds": [1, 2], "total": 3000 }`
3. **Shipment Tracking**
    - **Endpoint:** `/shipment`
    - **Method:** GET
    - **Description:** Fetch shipment status.
    - **Response:** `{ "orderId": 1001, "status": "Delivered", "ETA": "2 days" }`

Specialized Features

- **Real-Time Updates:** Ensure APIs support live shipment tracking.
- **Payment Integration:** Secure transaction handling via third-party gateways.

Sanity Schema Example

```
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: 'description', type: 'text', title: 'Description' },
    { name: 'image', type: 'image', title: 'Product Image' },
  ],
};
```

Collaboration and Refinement

1. **Group Discussions:** Brainstorm solutions for architectural and API challenges.
2. **Peer Review:** Gather feedback on system architecture and workflows.
3. **Version Control:** Utilize GitHub for tracking changes and fostering collaboration.

Key Outcomes

1. **Aligned Technical Plan:** A robust plan linked to Day 1 business goals.
2. **System Architecture:** Clear diagrams showing component interactions.
3. **Detailed API Requirements:** Endpoint details with methods and expected responses.
4. **Appwrite Schemas:** Drafted to manage core data entities.
5. **Portfolio-Ready Document:** Professional documentation for showcasing technical planning.

Next Steps

- Transition to Day 3: Implement the technical foundation by creating schemas and integrating APIs.
- Test workflows to ensure seamless operation across all system components.