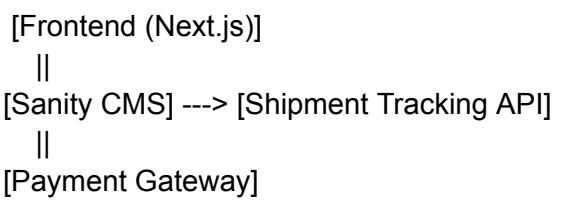
Day 2: Marketplace Technical Foundation - Swiftly

1. System Architecture Overview:

1. **System Components**
    - **Frontend (e.g., Next.js):** Displays the user interface for browsing groceries, placing orders, and tracking shipments.
    - **Backend (Sanity CMS):** Manages product data, customer details, and order records.
    - **Third-Party APIs:** Integrates shipment tracking and payment gateways for real-time updates and secure transactions.

2. **System Architecture Diagram:**



2. Key Workflows:

1. **User Registration Workflow:**
    - User Signs Up on the platform.
    - User data is stored in Sanity CMS.
    - Confirmation email is sent to the user.
2. **User Browsing Products**
    - User visits the homepage.
    - Frontend fetches product data from Sanity CMS via API and displays it.
3. **Order Placement**
    - User adds items to the cart.
    - Proceeds to checkout.
    - Order details are sent to Sanity CMS via API for storage.
4. **Real-Time Shipment Tracking**

By Saira Asif
- Swiftly

- After placing an order, the shipment status is fetched via the Third-Party Shipment Tracking API.
- Updates are displayed on the user's order page.

5. **Secure Payment**
    - Payment details are processed through the Payment Gateway.
    - Upon confirmation, details are stored in Sanity CMS.

3. API Requirements:

Here are some of the API endpoints I will use:

- **Galleries:**
    - GET /galleries: Retrieve all galleries.
    - GET /galleries/{GalleryID}: Retrieve a specific gallery by its ID.
    - POST /galleries: Add a new gallery.
    - PUT /galleries/{GalleryID}: Update an existing gallery.
    - DELETE /galleries/{GalleryID}: Delete a gallery.
- **Products:**
    - GET /products: Retrieve all products.
    - GET /products/{ProductID}: Retrieve a specific product by its ID.
    - POST /products: Add a new product.
    - PUT /products/{ProductID}: Update an existing product.
    - DELETE /products/{ProductID}: Delete a product.
- **Customers:**
    - GET /customers: Retrieve all customers.
    - GET /customers/{CustomerID}: Retrieve a specific customer.
    - POST /customers: Add a new customer.
    - PUT /customers/{CustomerID}: Update customer information.
    - DELETE /customers/{CustomerID}: Delete a customer.
- **Orders:**
    - GET /orders: Retrieve all orders.
    - GET /orders/{OrderID}: Retrieve a specific order by ID.
    - POST /orders: Place a new order.
    - PUT /orders/{OrderID}: Update an order (e.g., change status).
    - DELETE /orders/{OrderID}: Delete an order.
- **Riders:**
    - GET /riders: Retrieve all riders.
    - GET /riders/{RiderID}: Retrieve a specific rider by ID.
    - POST /riders: Add a new rider.
    - PUT /riders/{RiderID}: Update rider information.
    - DELETE /riders/{RiderID}: Delete a rider.
- **Delivery Zones:**
    - GET /zones: Retrieve all delivery zones.
    - GET /zones/{ZoneID}: Retrieve a specific delivery zone.
    - POST /zones: Add a new delivery zone.

By Saira Asif

- Swiftly

- PUT /zones/{ZoneID}: Update a delivery zone.
- DELETE /zones/{ZoneID}: Delete a delivery zone.

4. Technical Requirements

- **Frontend Requirements:**
  - Responsive design for mobile and desktop.
  - Pages: Home, Product Listing, Product Details, Cart, Checkout, Order Confirmation.
- **Backend Requirements (Sanity CMS):**
  - Manage product data, customer details, and order records.
  - Design schemas for products, orders, and customers.
- **Third-Party APIs:**
  - Integrate APIs for shipment tracking and payment gateways.
  - Ensure they provide real-time and secure data.