DAY 02 OF HACKATHON

PLANNING THE TECHNICAL FOUNDATION

Introduction:

Today, we are focusing on building the technical foundation for our marketplace, creating a high-level architecture diagram, and outlining how system components interact.

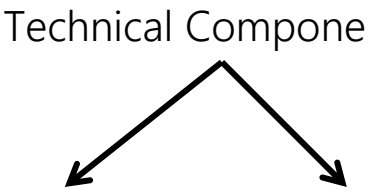
System Overview:

Our marketplace will connect users to a dynamic inventory of products fetched from a backend CMS (Sanity) and external APIs. The frontend will be built using Next.js and styled with Tailwind CSS, while payment gateways like Stripe and PayPal handle transactions.

Prepared By Abdullah Kashif

System Architecture Diagram

Below is a high-level architecture diagram illustrating the interaction between the frontend, backend, and external services.



Frontend

Framework: Next.js

Styling: TailWind Css

Key pages:

- Landing page
- Product Details Page
- Checkout Page
- Login/Register Page
- Orders Page

Backend

CMS: Sanity CMS

Data Models:

- Product
  (Name, Description, Price, Discount etc)
- Users
  (Name, Email, Phone, Address, etc)
- Orders:
  (Shipment ID, Customer Info, Payment Status)

External APIs

Product API: Fetch from sanity.

Payment Gateways:

- Stripe: Card payments.
- PayPal: Account and linked card payments.

User Flow

Landing Page → Product Details Page

- User lands on the homepage.
- Product data is fetched from Sanity CMS.
- User clicks a product to see details.

Next Actions: User click on a specific product to

- Frontend (Next.js) sends a request to the backend (Sanity CMS) to fetch product data.
- Backend retrieves product details (name, price, discount, buying percentage, availability) and sends them to the frontend.
- Frontend displays the list of products.

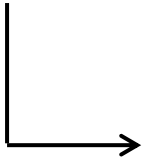
Kash

- Frontend (Next.js) sends a request to the backend (Sanity CMS) to fetch product data.
- Backend retrieves product details (name, price, discount, buying percentage, availability) and sends them to the frontend.
- Frontend displays the list of products.

Product Details Page → Checkout

User selects one of three options:

- Add to Wishlist: Saves the product for later.
- Add to Cart: Adds the product to the shopping cart for future checkout.
- Proceed to Checkout: Moves to payment and shipping details.



If "Add to Wishlist" is selected:

- Frontend sends a request to the backend to update the user's wishlist with the selected product.
- Confirmation is displayed to the user.

If "Add to Cart" is selected:

- Frontend sends a request to update the user's cart in the backend.
- Confirmation is displayed to the user.

Prepared By Abdullah Kashif

Checkout Flow

Logged-In User:

- User is directed to the Checkout Page.
- Frontend fetches:
  - User's saved shipping address from the backend.
  - Product details in the cart from the backend.
- Displays:
  - Shipping address (with an option to add a new address).
  - Shipping fee and estimated delivery time.
  - Total price, including discounts and taxes.
  - Payment options (Stripe or PayPal).
- User selects a payment option and proceeds.

Guest User:

- User is redirected to the Login/Register Page.
- Provides:
  - Name, Email, Password, Phone Number.
  - Optionally, shipping address.
- Backend saves user details in the database.
- User is redirected back to the Checkout Page to proceed.

Payment:

- User selects a payment method:
  - Stripe: Enters card details securely.
  - PayPal: Redirected to PayPal for account-based payment.
- Payment Gateway verifies the payment and sends a success/failure status back to the backend.

Abd

1. Post-Payment Actions:
   - If payment is successful:
     - Order data is saved in the backend:
       - Shipment ID
       - Customer ID
       - Customer contact details (phone, email, address)
       - Payment status
       - Product details
     - Backend triggers email confirmation to the user (optional).
     - User is redirected to the Order Confirmation Page.
   - If payment fails:
     - User is notified and can retry.

Login/Register Flow

Guest User Navigating to Checkout:

Redirected to the Login/Register Page.

User Action:

- For login: Enters email and password.
- For registration: Provides name, email, phone number, and password.

System Response:

- Backend verifies login credentials or creates a new user record.
- On successful login/registration:
    - User session is created.
    - Redirects the user to the previous page (e.g., Checkout).

Technical Roadmap

Phase 1: Project Setup and Planning

Goals and Scope

- Define project objectives, features, and scope.
- Finalize technology stack: Next.js, Tailwind CSS, Sanity CMS, Stripe, PayPal.

Development Environment Setup

- Initialize a GitHub repository.
- Set up Next.js project and integrate Tailwind CSS.
- Configure Sanity CMS for the backend.

API Documentation

- Define endpoints for user, product, order, and payment management.

Phase 2: Frontend Development

Design System

- Create reusable components using Tailwind CSS for consistent UI/UX.

Prepared By Abdullah Kashif

Core Pages

1. Landing Page: Fetch and display products from Sanity CMS.
2. Product Details Page: Show product information and actions (add to cart, wishlist, etc.).
3. Checkout Page: Display cart summary, shipping address, and payment options.
4. Login/Register Page: Authentication for guest users.
5. Orders Page: Show order history and shipment statuses.

Integration

- Connect frontend with backend APIs for dynamic data.

Phase 3: Backend Development

Sanity CMS Setup

- Define schemas for products, users, orders, and payments.
- Enable webhooks for real-time updates.

API Integration

- Implement the Api which given :

Phase 4: Third-Party Integrations

Payment Gateways

- Integrate Stripe and PayPal APIs for secure transactions.

External Product API

- Set up synchronization with external APIs for product updates.

Phase 5: Testing and Deployment

Testing

- Unit tests for components and APIs.
- End-to-end tests for key user flows (checkout, payment, order tracking).

Deployment

- Deploy the frontend on Vercel.
- Deploy Sanity CMS on a managed platform.

FOLLOW ME ON



Prepared By Abdullah Kashif