Ayaan Zeeshan

Marketplace Builder Hackathon 2025

Hackathon Day 2: Planning The Technical Foundation

Welcome to Day 2 of my exciting journey in the Marketplace Builder Hackathon 2025! Today, I focused on designing a robust system architecture and planning the integration between the frontend, backend, and third-party services for a seamless user experience.

System Architecture Highlights

The system is designed with scalability and user-friendliness in mind, ensuring smooth communication between different components. Here's a quick overview:

Frontend & Backend Workflow

Frontend Framework:

Built using Next.js with a modern UI powered by Tailwind CSS and ShadCN UI components. State management is handled by Redux for predictable state updates.

Backend:

Leveraging Sanity.io as the CMS, which also manages API routes to serve data dynamically via RESTful APIs.

Communication:

Frontend communicates with the backend through standard HTTP methods like GET, POST, PUT, and DELETE.

Third-Party API Integration

To extend the platform's functionality, I’ve outlined the use of third-party APIs for essential features:

Purpose:

The API will manage tasks like payment processing and real-time product availability.

Integration Workflow:
The frontend sends API requests to the backend, which processes the data and communicates with the third-party service to fetch or update information.
Security Measures:
Data is exchanged securely using HTTPS and API key-based authentication for reliable operations.

Defined API Routes

Below are some core API routes that define the system's functionality:

GET /api/products – Fetch a list of available products.
POST /api/order – Create a new order and initiate payment processing.
GET /api/order/:id – Retrieve the details of a specific order.
PUT /api/order/:id – Update the order status (e.g., shipped, delivered).
GET /api/payment/verify – Verify payment status through the third-party API.
Additional routes for payment gateway interactions and other features are also planned.

Tech Stack Overview

Frontend: Next.js, Tailwind CSS
Backend: Sanity.io
API Integration: Third-party services for payments and product availability
Authentication: Secure user login and transactions via Clerk Authentication

Data Flow Documentation

The data flow between components is outlined as follows:

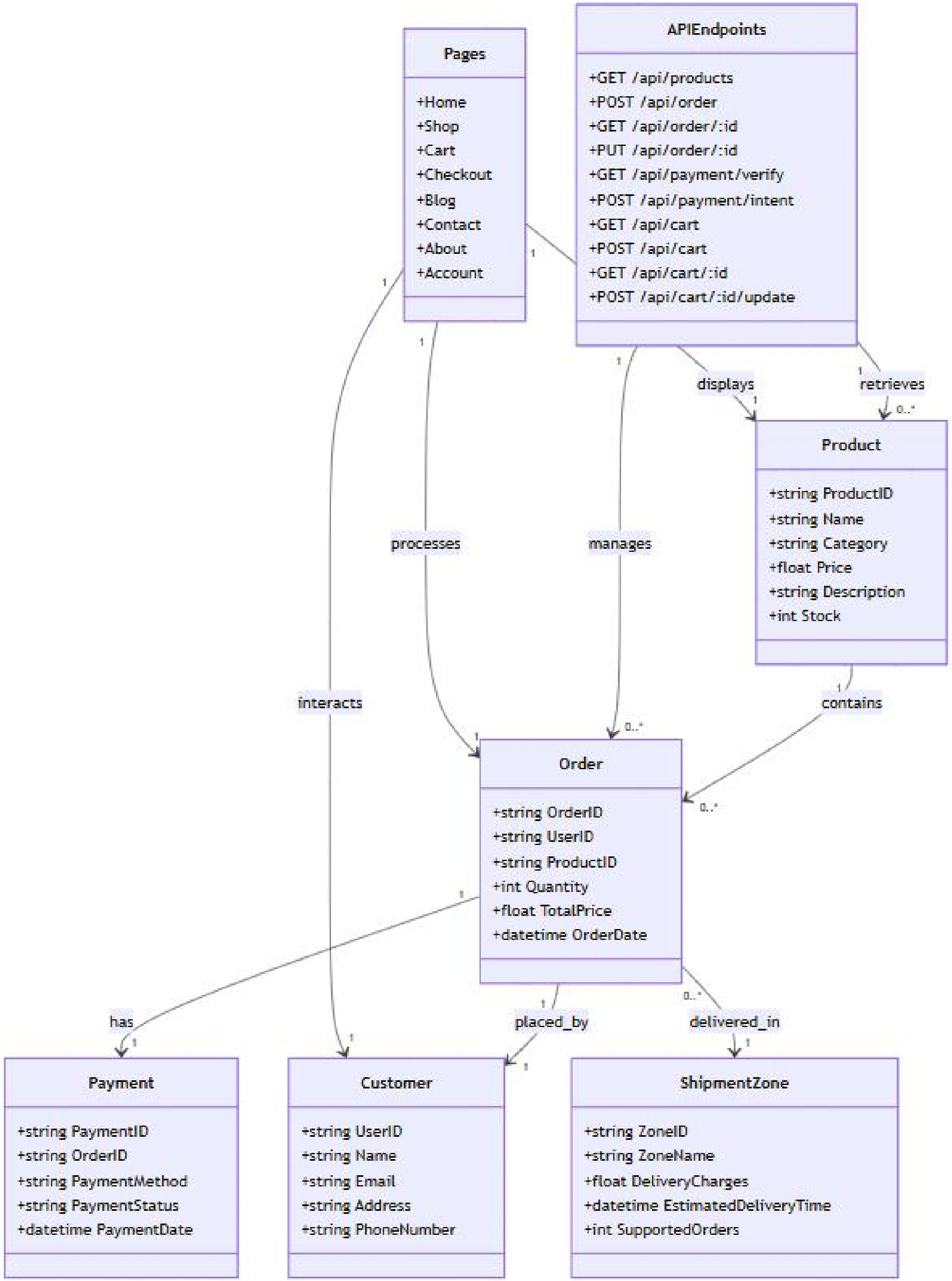
Frontend:
Sends requests to the backend through API endpoints.

Backend:

Processes requests, interacts with the third-party API, and returns responses.

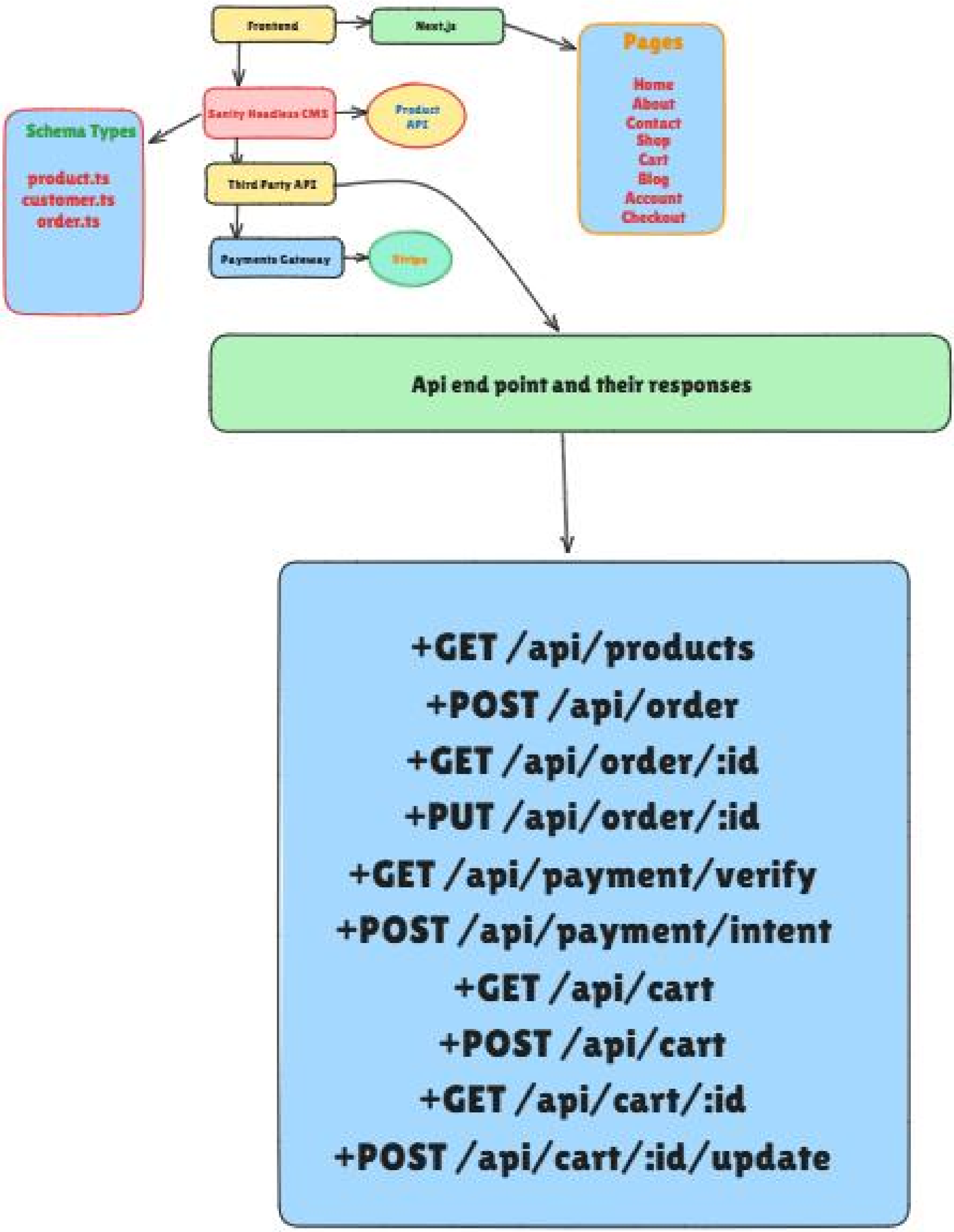
Visual Data Schema

The data structure for entities like products, orders, and payments is carefully designed. Check out the Data Schema Diagram for a visual representation.


Pages
+Home
+Shop
+Cart
+Checkout
+Blog
+Contact
+About
+Account
APIEndpoints
+GET /api/products
+POST /api/order
+GET /api/order/:id
+PUT /api/order/:id
+GET /api/payment/verify
+POST /api/payment/intent
+GET /api/cart
+POST /api/cart
+GET /api/cart/:id
+POST /api/cart/:id/update
1
1
1
1
1
displays
1
retrieves
0..*
Product
+string ProductID
+string Name
+string Category
+float Price
+string Description
+int Stock
processes
manages
interacts
1
contains
1
0..*
0..*
Order
+string OrderID
+string UserID
+string ProductID
+int Quantity
+float TotalPrice
+datetime OrderDate
1
1
0..*
has
1
placed_by
1
1
delivered_in
1
Payment
+string PaymentID
+string OrderID
+string PaymentMethod
+string PaymentStatus
+datetime PaymentDate
Customer
+string UserID
+string Name
+string Email
+string Address
+string PhoneNumber
ShipmentZone
+string ZoneID
+string ZoneName
+float DeliveryCharges
+datetime EstimatedDeliveryTime
+int SupportedOrders

Day 2 Task

Frontend Next.js



Frontend

Next.js

Pages

Home
About
Contact
Shop
Cart
Blog
Account
Checkout

Sanity Headless CMS

Product API

Schema Types

product.ts
customer.ts
order.ts

Third Party API

Payments Gateway

Stripe

Api end point and their responses

+GET /api/products
+POST /api/order
+GET /api/order/:id
+PUT /api/order/:id
+GET /api/payment/verify
+POST /api/payment/intent
+GET /api/cart
+POST /api/cart
+GET /api/cart/:id
+POST /api/cart/:id/update