Market Place Hackathon 2025

Day 2 : Technical Foundation

Overview:

In this documentation, Technical features have been outlined and plans for this E-commerce platform “Hekto” by giving a clear picture of every basic technical step taken in the production of this Marketplace Such as Technical Requirements, System Architecture, Key work flowchart, API requirements, Data Schema Design and Technical Roadmap.

Technical Requirements:

Front-End Requirements:

- **Framework** : Next.Js
- **Styling** : Tailwind css and Custom css

For Front-end we'll be using Next.js as our choice of framework which can handle both front-end and back-end tasks, for styling we'll be using Tailwind css and custom css to make our Marketplace site's front-end visually pleasing and user-friendly.

Back-end Requirements:

- CMS: Sanity.Io

For Back-end, Sanity.io will be integrated and used as Headless CMS for storing Products data , customer details, order details. Also it'll use it to Fetch the product data and showcase the products on the MarketPlace. Sanity.Io Provide Headless CMS to Manage our Data and update it realtime.

Third Party API's:

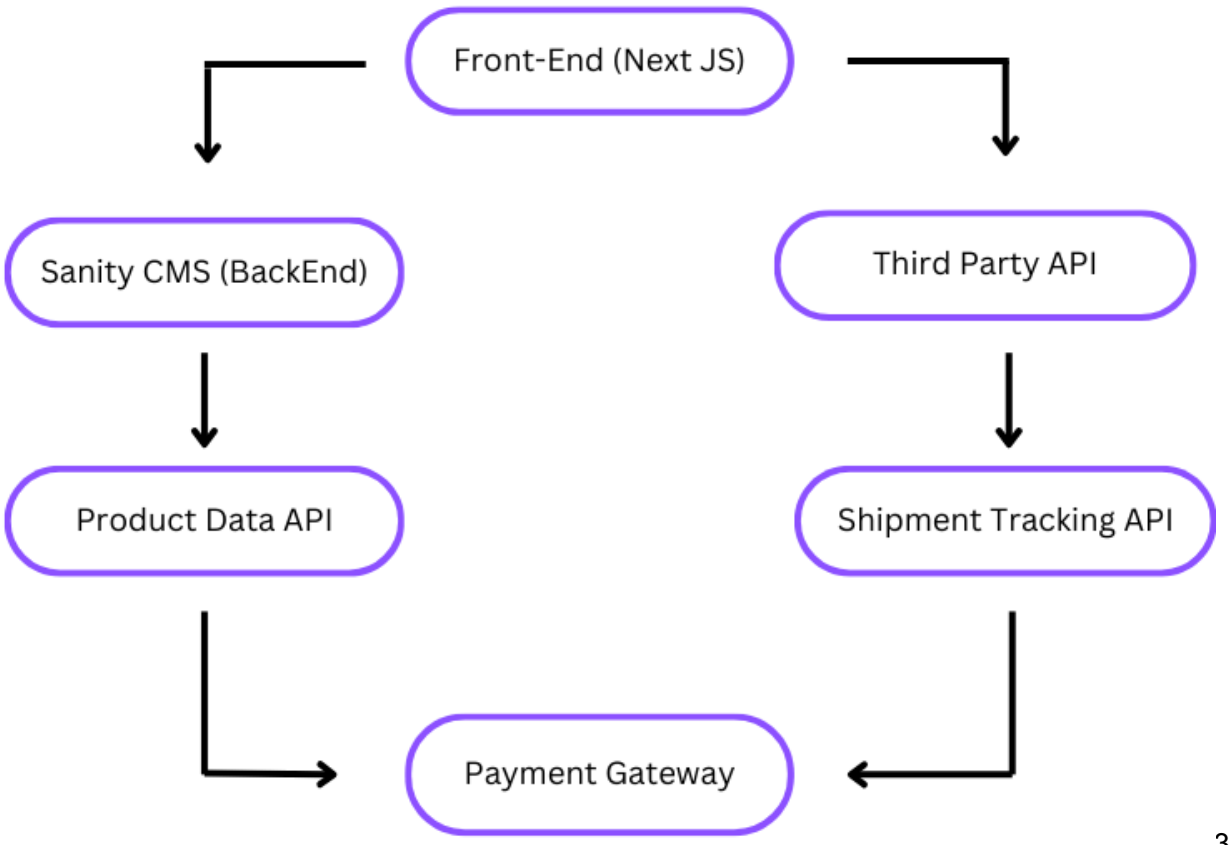
- Products API : Provided API
- Shipment Tracking: Third Party API
- Payment API : Stripe

For Products we'll be using the provided API / Products Dataset to upload on Sanity and showcase it on our marketplace. For Shipment tracking API its To be Decided. For Payment API we'll be using swift for fast, secure and reliable payment procedures.

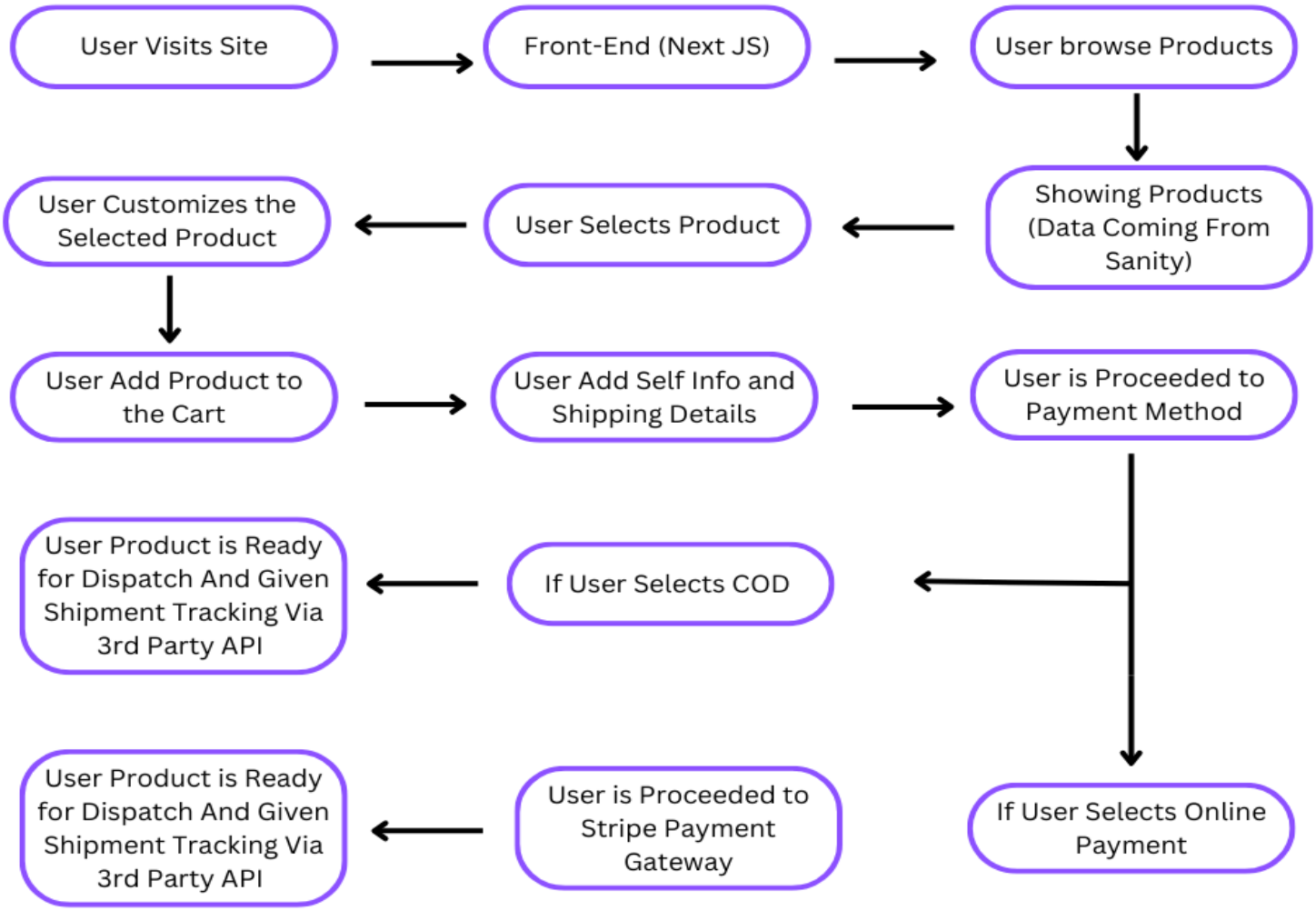
Design System Architecture:

The fundamental Chart Below Shows the Technical architecture and Concept of User flow of Our MarketPlace.

Technical Design Architecture:



User Flow Chart:



Plan API Requirements:

SCHEMA:

Product Schema:

```
export const product = {
  name: "product",
  type: "document",
  title: "Product",
  fields: [
    { name: "name", type: "string", title:
"Name" },
    { name: "image", type: "image", title:
"Image" },
    { name: "price", type: "number", title:
"Price" },
    { name: "discount", type: "number", title:
"Discount" },
    { name: "isFeatured", type: "boolean",
title: "Is Featured" },
    { name: "stock", type: "number", title:
"Stock" },
    { name: "category", type: "string", title:
"Category" }
  ]
};
```

Order Schema

```
export const order = {
  name: "order",
  type: "document",
  title: "Order",
  fields: [
    { name: "orderId", type: "string", title:
"Order ID" },
    { name: "productId", type: "reference", to:
[{ type: "product" }], title: "Product ID" },
    { name: "quantity", type: "number", title:
"Quantity" }
  ]
};
```

Customer Schema:

```
export const customer = {
  name: "customer",
  type: "document",
  title: "Customer",
  fields: [
    { name: "customerId", type: "string", title:
"Customer ID" },
    { name: "customerName", type: "string",
title: "Customer Name" },
```

```
    { name: "contactInfo", type: "string",
title: "Contact Info" }
  ]
};
```

Shipment Schema:

```
export const shipment = {
  name: "shipment",
  type: "document",
  title: "Shipment",
  fields: [
    { name: "shipmentId", type: "string", title:
"Shipment ID" },
    { name: "orderId", type: "reference", to: [{
type: "order" }], title: "Order ID" },
    { name: "status", type: "string", title:
"Status" }
  ]
};
```

Delivery Schema:

```
export const delivery = {
  name: "delivery",
  type: "document",
  title: "Delivery",
  fields: [
```

```
    { name: "address", type: "string", title:
"Address" },
    { name: "coverageArea", type: "string",
title: "Coverage Area" },
    { name: "assignedDriver", type: "string",
title: "Assigned Driver" }
  ]
};
```

Payment Schema:

```
export const payment = {
  name: "payment",
  type: "document",
  title: "Payment",
  fields: [
    { name: "price", type: "number", title:
"Price" },
    { name: "discount", type: "number", title:
"Discount" },
    { name: "paymentMethod", type: "string",
title: "Payment Method" }
  ]
};
```

ENDPOINTS:

1. Product
   - Endpoint: GET /products
   - Endpoint: GET /products/{id}
   - Endpoint: POST /products
   - Endpoint: DELETE /products/{id}

2. Order
   - Endpoint: GET /orders
   - Endpoint: GET /orders/{id}
   - Endpoint: POST /orders
   - Endpoint: DELETE /orders/{id}

3. Customer
   - Endpoint: GET /customers
   - Endpoint: GET /customers/{id}
   - Endpoint: POST /customers
   - Endpoint: DELETE /customers/{id}

4. Shipment
   - Endpoint: GET /shipments
   - Endpoint: GET /shipments/{id}
   - Endpoint: POST /shipments
   - Endpoint: DELETE /shipments/{id}

5. Delivery
   - Endpoint: GET /deliveries
   - Endpoint: GET /deliveries/{id}
   - Endpoint: POST /deliveries
   - Endpoint: DELETE /deliveries/{id}

6. Payment
   - Endpoint: GET /payments
   - Endpoint: GET /payments/{id}
   - Endpoint: POST /payments
   - Endpoint: DELETE /payments/{id}