Name: Areeba Saleem

Class: Friday 9:00 – 12:00

Roll No: 189589

HACKATHON 03

(DAY:02)

"PLANNING THE TECHNICAL FOUNDATION"

STEP:01 CHOOSE THE TECH STACK

FRONTEND REQUIREMENTS:

A part where user interacts:

- **User Friendly**: Design should be visually appealing must be clean, modern, dynamic to enhance user experience.
- **Responsiveness:** Tailwind CSS, Shaden UI etc. are used to make a website responsive.
- **Essential Pages:**

  Home: Food Tuck main page represent feature products including search bar for navigation.

  Product Listing Pages: This page displays a list of food product with categories and price to make it easy.

  Product Detail Page: A detailed page of single food product includes image price and everything about product.

  Shopping Cart: A page with a summary of selected items, subtotal, shipping options before proceeding to checkout.

  Checkout Page: Final step of process where customer surely enter the required fields to complete their purchase.

  Order Completion: A summary of complete order including details of order payment and expected delivery time.

MORE PAGES:

- About Page
- Contact Page
- Menu Page

- Shop list Page
- Shop Details Page
- Our Chefs Page
- Sign In Page
- Sign Out Page
- Blog List Page
- Blog Details Page
- FAQ Page
- Error Page

SANITY CMS AS BACKEND:

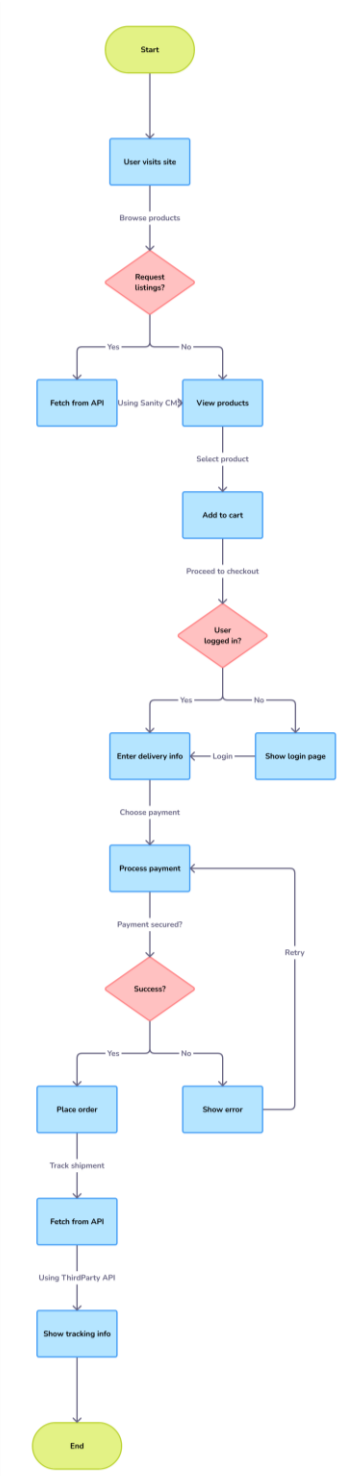
Sanity is a headless CMS that pairs well for building server- rendered, statically generated and optimized application performance. Sanity act as a database for our marketplace.

- **Products:** We will store all information related to products here.
- **Customers:** For tracking of customers profile's, contact info, and their order history.
- **Orders:** Sanity will handle details like order status, timestamps, and ordered products.

THIRD PARTY API'S:

- **Ship Engine:** Used for Order Tracking and Manage Shipment.
- **Stripe:** Used for Paying Online.
- **Razorpay:** Use for local wallets like Easy Paisa and Jazz Cash.

STEP: 02 SYSTEM ARCHITECTURE OVERVIEW:



COMPONENTS ROLES:

- **Next.js (Frontend):** Handles user interactions product browsing, cart management, and order placement.
- **Sanity CMS:** Stores product details, customer info, and order data.
- **Payment Gateway:** Process secure payments through Stripe etc.
- **Ship Engine:** Tracks and updates shipping and delivery information.

KEY WORKFLOW:

General User Workflow:

It shows how user will interact with the step by components.

Browsing Products: User visit the site ---- Frontend calls / products API ---- Products Listings fetched from Sanity CMS ---- Displayed to the user.

Adding Products To Cart: User clicks "Add to Cart" ---- Frontend stores selected product in a session or database.

Order Placement: User proceed to checkout ---- /orders API processes cart data ---- Payment is initiated ---- Confirmation sent to the user.

Shipment Tracking: U*ser views order details ---- /shipment API* retrieves real-time tracking updates from Shipengine ---- Displays delivery ETA.

STEP: 03 PLAN API'S REQUIREMENTS:

End Points:	Method:	Purpose:	Response:
/products	GET	Fetch all products list.	Products Details (ID, name, price, stock, image)
/shipment	GET	Track order status (3rd party API).	Shipment ID, Order ID, status, expected delivery date
/orders	POST	Create new order in sanity.	Customer Info, Product Detail,

CONCLUSION:

A well-designed food tuck marketplace API provides essential endpoints for vendors, customers, and administrators. By providing these endpoints, the API empowers a scalable and efficient food tuck marketplace ecosystem.

STEP:04 TECHNICAL DOCUMENTATION:

1: OBJECTIVES:

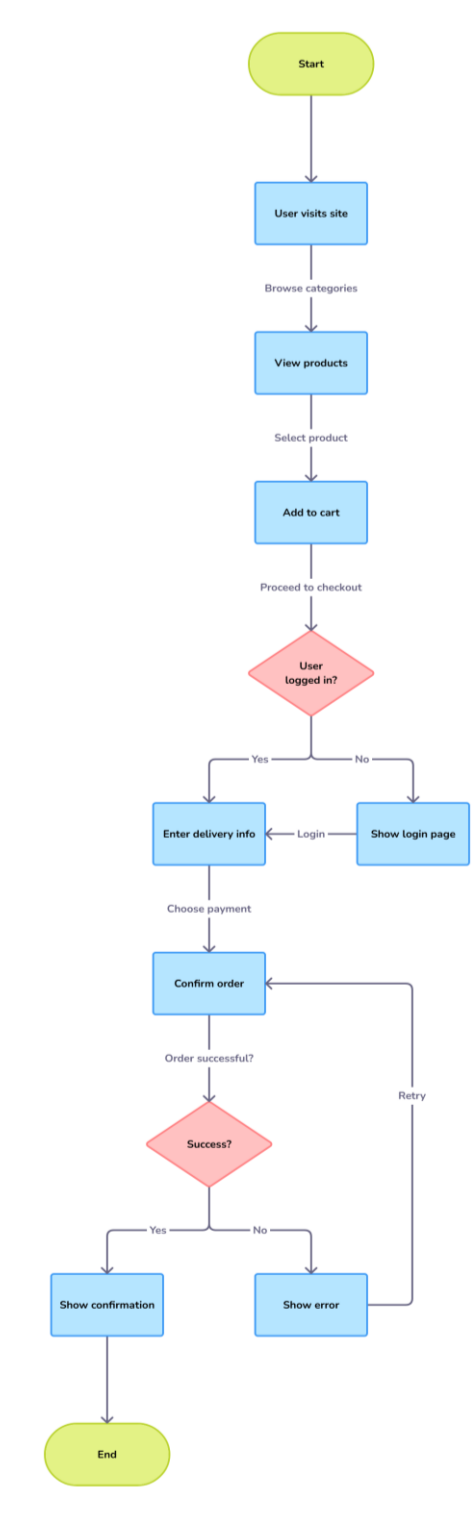
To build a user friendly scalable, user-friendly, q-commerce platform with the following features:

- Product browsing and management via sanity CMS.
- Authentication with Clerk.
- Order tracking with ShipEngine API.
- Secure payment via stripe.
- Modern tools like useContext for cart functionality.

2 : API ENDPOINTS:

Endpoints	Method	Purpose	Response
/products	GET	Fetch products data from sanity CMS.	{“id”:1,”name”:”Product A price”:100}
/shippping-label	POST	Generate a shipping label using ShipEngine label ID	{“labelled”:”LABEL123”, “status”:Generated}
/track-order	GET	Retrieve order status using ShipEngine label ID.	{shipmentID:” SHIP123”, ”status”: In transit}
/check-out	POST	Integrate stripe for payment proceeding.	{“orderid”:123, “status’:”Success”

3: WORKFLOW DIAGRAM:



4: <u>SANITY SCHEMA EXAMPLE:</u>

```
export const product = {
name :”product”,
type :”document”,
title :”Product”,
fields : [
{
name :”image”,
title: ”Product Image”,
type: ”image”,
},
{
name :”name”,
title: ”Product Title”,
type: ”string”,
},
{
name :”price”,
title: ”Product Price”,
type: ”number”,
},
{
name :”price_id”,
title:” Stripe Price ID”,
type: ”string”,
```

```
},
{
name :”description”,
title: ”Product Description”,
type: ”text”,
},
{
name :”slug”,
title: ”Product Slug”,
type: ”slug”,
options: {
source:”name”,
},
},
{
name:” stock”,
title: ”Product Stock”,
type: ”number”,
},
],
};
```

CONCLUSION:

Well-written documentation enables developers, users, and stakeholders to understand complex technical concepts and processes. Clear and concise documentation reduces errors, improves efficiency, and enhances overall user experience. By prioritizing technical documentation, organizations can promote knowledge sharing, collaboration, and innovation.