MARKET PLACE TECHNICAL FOUNDATION

(Q-Commerce Food and Resturant web site)

Technical Requirements for Food and Restaurant Website

Project Overview

PREPARED BY
ANILA AHMED
00062949

- Frontend: Next.js (React Framework)
- Backend: Sanity CMS (Headless CMS)
- Styling: Tailwind CSS
- Database: Managed through Sanity schemas
- Payment Gateway: Integration with Stripe
- Live Tracking: Google Maps API
- Dynamic Routing: Next.js dynamic routes for product details, category pages, and user orders
- Hosting: Vercel for frontend and Sanity Studio .

Project Flow Chart

1. Planning Phase

- Define objectives and target audience.
- Outline major features:
- Food product listing (menu)
- Dynamic categories (e.g., Appetizers, Main Course, Desserts)
- Product search and filtering
- Cart and Checkout with payment integration
- Order history and live order tracking

2. Design Phase

Use Figma to create a pixel-perfect design for all screens:

- Home Page
- Product Listing
- Product Detail Page
- Cart
- Checkout
- Order Tracking

3. Development Flow

Frontend Development (Next.js)

1. **Set up Next.js project structure:**

- pages/ for routing
- components/ for reusable UI elements
- styles/ for Tailwind CSS customization
- Dynamic Routing:

2. **Product Categories: /categories/[category]**

- Product Details: /product/[slug]
- Order Tracking: /order/[orderId]
- UI Implementation:

3. **Tailwind CSS for responsive design**

- Component-driven approach for scalability

Backend Setup (Sanity)

1. **Define schemas for data models:**

product schema:

- Name
- Category
- Description
- Price
- Images
- Availability

category schema:

- *Name*
- *Slug*
- *order schema:*
- *Order ID*
- *Customer Details*
- *Products Ordered*
- *Status (e.g., Preparing, Out for Delivery, Delivered)*

2. Use GROQ for querying data.

3. Sanity Studio customization:

- Add rich text editors for product descriptions.
- Preview feature for live updates.

API Integration

- **Sanity API:** Fetch data for products and categories.

Example: https://<projectId>.api.sanity.io/v1/data/query/production

Stripe API: Payment processing.

Google Maps API: Real-time tracking.

Live Features

- Implement server-side rendering (SSR) for SEO optimization on product and category pages.
- Add client-side interactivity using React hooks and Context API for managing cart and checkout state.

Deliverables

- *Responsive web app with pixel-perfect design.*
- *Functional backend with:*
- *Dynamic product management.*
- *Order tracking capabilities.*
- *Payment integration with real transactions.*
- *Documentation for API endpoints and schema.*

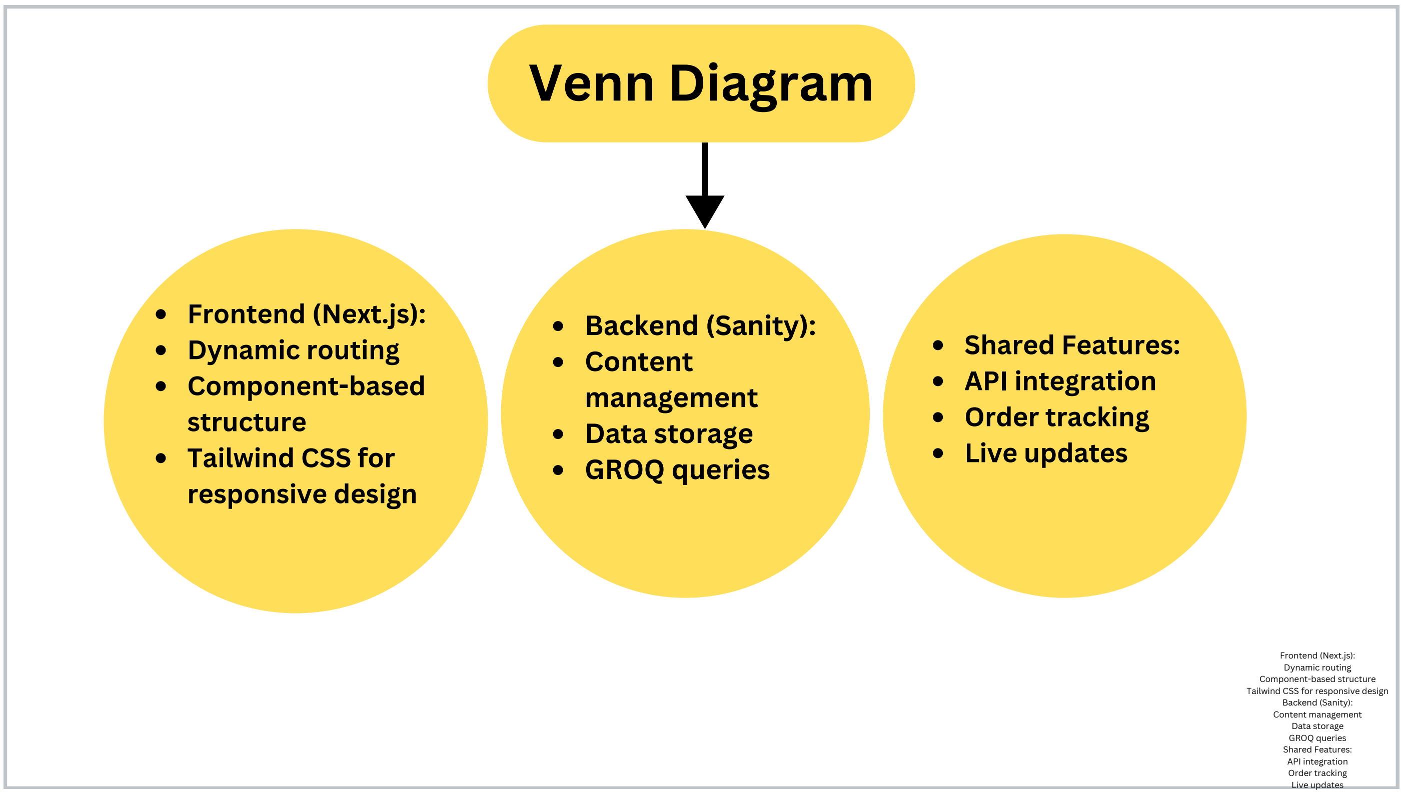
Flow Chart Description

Start -> Requirements Gathering -> Figma Designs -> Set Up Next.js & Tailwind CSS -> Develop Sanity Backend

Develop Frontend Pages:
-> Home
-> Categories
-> Product Detail
-> Cart & Checkout
-> Order Tracking

Integrate APIs:
-> Sanity for Data Fetching
-> Stripe for Payments
-> Google Maps for Tracking

Testing & Debugging -> Deployment to Vercel -> Final Review -> Launch



Frontend (Next.js):
Dynamic routing
Component-based structure
Tailwind CSS for responsive design
Backend (Sanity):
Content management
Data storage
GROQ queries
Shared Features:
API integration
Order tracking
Live updates

Sanity Schemas for Food Items

1. Product Schema

```
export default {
name: 'product',
type: 'document',
title: 'Product',
fields: [
{ name: 'name', type: 'string', title: 'Product Name' },
{ name: 'category', type: 'reference', to: [{ type: 'category' }], title: 'Category' },
{ name: 'description', type: 'text', title: 'Description' },
{ name: 'price', type: 'number', title: 'Price' },
{ name: 'images', type: 'array', of: [{ type: 'image' }], title: 'Images' },
{ name: 'availability', type: 'boolean', title: 'Available?' },
],
};
```

2. Category Schema

```
export default {
name: 'product',
type: 'document',
title: 'Product',
fields: [
{ name: 'name', type: 'string', title: 'Product Name' },
{ name: 'category', type: 'reference', to: [{ type: 'category'
}], title: 'Category' },
{ name: 'description', type: 'text', title: 'Description' },
{ name: 'price', type: 'number', title: 'Price' },
{ name: 'images', type: 'array', of: [{ type: 'image' }], title:
'Images' },
{ name: 'availability', type: 'boolean', title: 'Available?' },
```

3. Order Schema

```
export default {
name: 'order',
type: 'document',
title: 'Order',
fields: [
{ name: 'orderId', type: 'string', title: 'Order ID' },
{ name: 'customerDetails', type: 'object', title: 'Customer Details', fields: [
{ name: 'name', type: 'string', title: 'Customer Name' },
{ name: 'email', type: 'string', title: 'Email' },
{ name: 'phone', type: 'string', title: 'Phone Number' },
] },
{ name: 'products', type: 'array', of: [{ type: 'reference', to: [{ type: 'product' }] }], title: 'Products
Ordered' },
{ name: 'status', type: 'string', title: 'Order Status', options: { list: ['Preparing', 'Out for Delivery',
'Delivered'] } },
],
};
```