Day 6 - Final Report: Comprehensive Summary of Days 1–6 [Foodtuck]

REPORT :

Day 4 - Final Report: Comprehensive Summary of Days 1–6

Prepared by: Maria Khan

Date: [22-January-2025]

Day 1: Laying the Business Foundation

On the first day, the focus was on understanding the business requirements and defining a solid foundation for the project. The key achievements were:

1. **Business Goals Defined**:
    - Identified the problem the marketplace aims to solve.
    - Defined the target audience and the unique value proposition (UVP) of the platform.
2. **Preliminary Data Schema Drafted**:
    - Outlined entities such as products, orders, and customers and their relationships using paper and pencil.
3. **Single Focus on Business Needs**:
    - Ensured alignment with real-world business needs before moving to technical planning.

Day 2: Transition to Technical Planning

On Day 2, the business goals were translated into actionable technical requirements, and a high-level system architecture was designed.

1. **Technical Requirements Defined**:

- Frontend: User-friendly interface, responsive design, and key pages (Home, Product Listing, Cart, Checkout).
- Backend: Sanity CMS for managing product, order, and customer data.
- APIs: Integration for shipment tracking, payment gateways, and backend services.

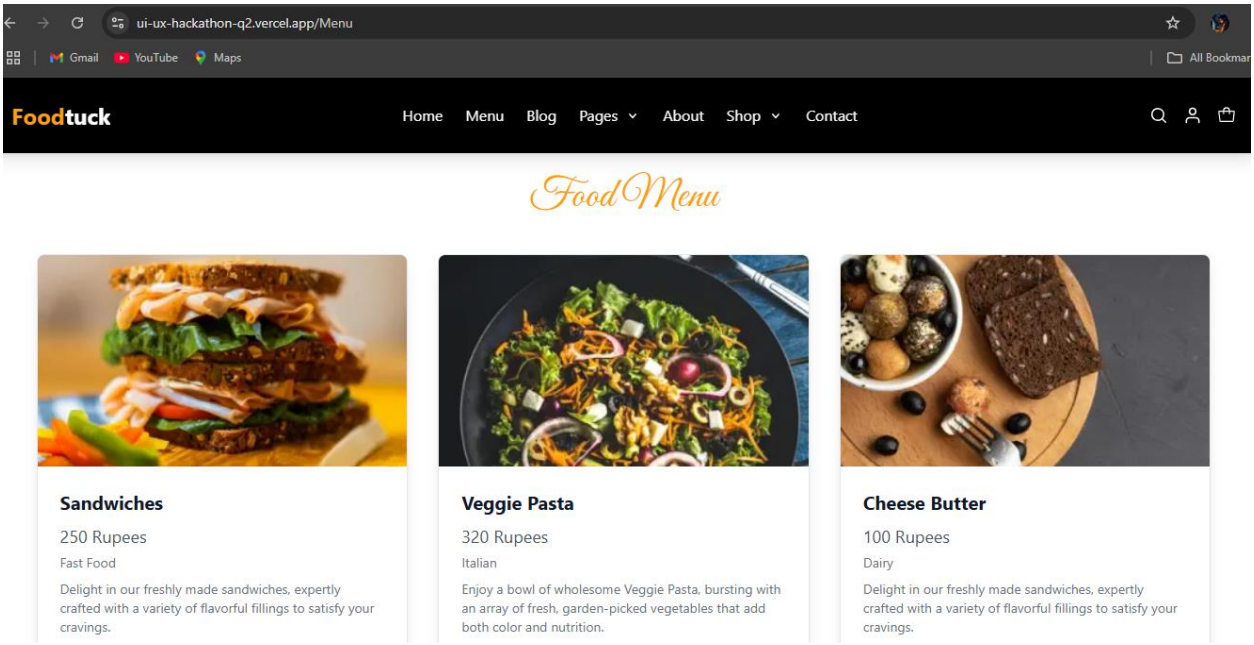
2. **System Architecture Designed**:
   - Created a detailed diagram of component interactions (Frontend ↔ Sanity CMS ↔ APIs).
3. **API Requirements Documented**:
   - Defined endpoints, methods, payloads, and responses, ensuring alignment with workflows.
4. **Sanity Schemas Drafted**:
   - Created schemas for products, orders, and customers with fields tailored to the marketplace type.

Day 3: API Integration, Implementing Core Features

The focus on Day 3 was building the foundation of the platform's functionality.

1. **Frontend Development**:
   - Created essential pages like Home, Product Listing, and Cart using Next.js.
   - Ensured responsiveness across devices with Tailwind CSS.
2. **Backend Configuration**:
   - Configured Sanity CMS for product and customer data.
   - Established a connection between the frontend and Sanity CMS.
3. **API Integration**:
   - Integrated basic APIs for food menu.

```tsx
"use client";

import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import React, { useEffect, useState } from "react";
import { Great_Vibes } from '@next/font/google';

const greatVibes = Great_Vibes({
  weight: ['400'],
  subsets: ['latin'],
  display: 'swap',
});

const Foods = () => {
  const [foods, setFoods] = useState<any[]>([]);
  const [error, setError] = useState<string>("");

  useEffect(() => {
    async function fetchFoods() {
      try {
        const data = await client.fetch(
          `*[_type == "food"]{
            name,
            category,
            available,
            image,
            description,
            price,
            _id
          }`
        );
        setFoods(data);
      } catch (err) {
        setError("Failed to load food data. Please try again later.");
        console.error("Error fetching food data:", err);
      }
    }
    fetchFoods();
  }, []);

  return (
    <div className="container mx-auto px-4 py-8">
      <h1 className={`${greatVibes.className}
 font-normal text-[44px] leading-10 text-[#FF9F0D] text-center`}>
        Food Menu
      </h1>

      {error && (
        <div className="text-red-500 text-center mt-4">
          <p>{error}</p>
        </div>
      )}

      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-8 mt-10">
        {foods.map((food) => (
          <div
            key={food._id}
            className=
"border border-gray-200 rounded-lg overflow-hidden shadow-lg hover:shadow-2xl trans
ition-shadow bg-white"
          >
            <Image
              src={urlFor(food.image).url()}
              alt={food.name}
              width={400}
              height={300}
              className="w-full h-56 object-cover"
            />
            <div className="p-6">
              <h2 className="text-xl font-bold text-gray-900">{food.name}</h2>
              <p className="text-lg text-gray-600 mt-2">
                {food.price} Rupees
              </p>
              <p className="text-sm text-gray-500 mt-1">{food.category}</p>
              <p className="text-sm text-gray-500 mt-2 line-clamp-3">
                {food.description}
              </p>
              <p
                className={`text-sm font-medium mt-4 ${
                  food.available ? "text-green-600" : "text-red-600"
                }`}
              >
                {food.available ? "Available" : "Not Available"}
              </p>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};

export default Foods;
```



Day 4: Dynamic Frontend Components for [Foodtuck], Advanced Features and Workflows

Day 4 involved adding advanced features and refining workflows to enhance the user experience.

1. **Dynamic Features**:
    - Implemented dynamic product listing and filtering based on user preferences.
    - Integrated a cart system with add-to-cart and remove-from-cart functionality.
2. **Collaborative Refinements**:
    - Peer-reviewed the architecture and workflows for scalability and efficiency.

Day 5: Testing ,Error Handling and Backend Integration Refinement

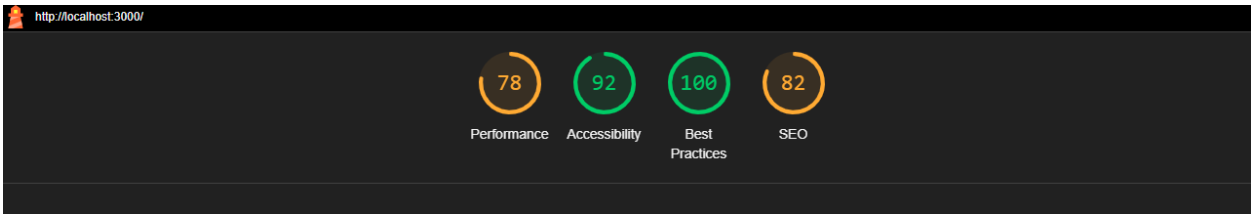
The fifth day emphasized optimizing the platform for scalability and a polished user experience.

1. **Performance Optimization**:

- Improved API response times with optimized queries to Sanity CMS.
- Added lazy loading for images and products to enhance frontend performance.

2. **User Interface Enhancements**:
    - Applied modern, visually appealing styling using Tailwind CSS.
    - Improved the accessibility and responsiveness of key components.
3. **Testing and Debugging**:
    - Conducted extensive testing for workflow accuracy and bug fixing in the frontend and backend.
    - Analyze Performance :
    - Used the browser's Developer Tools (Network and Performance tabs) to identify unused CSS and JavaScript files.
    - • Used Chrome DevTools Lighthouse to analyze website performance, accessibility, best practices, and SEO

Performance testing results:



Test case report :

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
TC001	Validate product listing page	Open product page > Verify products	Products displayed correctly	Products displayed correctly	Passed	Low	-	No issues found
TC002	Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message	Error message shown	Passed	Medium	-	Handled gracefully
TC003	Check cart functionality	Add product to cart > Verify cart contents	Cart updates with added product	Cart updates as expected	Passed	High	-	Works as expected
TC004	Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly to screen size	Responsive layout working as intended	Passed	Medium	-	Test successful
TC005	Test error message on API failure	Simulate API failure > Check error UI	Error UI displays appropriate message	Error message shown	Passed	High	-	Managed gracefully
TC006	Validate fallback UI for missing data	Simulate empty data > Verify fallback message	"No items found" message displays	Fallback UI working correctly	Passed	Medium	-	Test completed
TC007	Analyze performance via Lighthouse	Run Lighthouse test > Review recommendations	Issues identified and optimization suggestions	Performance optimization started	Passed	Medium	-	Lighthouse verified
TC008	Measure page load time	Open website > Record initial load and	Initial load time under 2 seconds	Performance meets criteria	Passed	High	-	Optimization successful

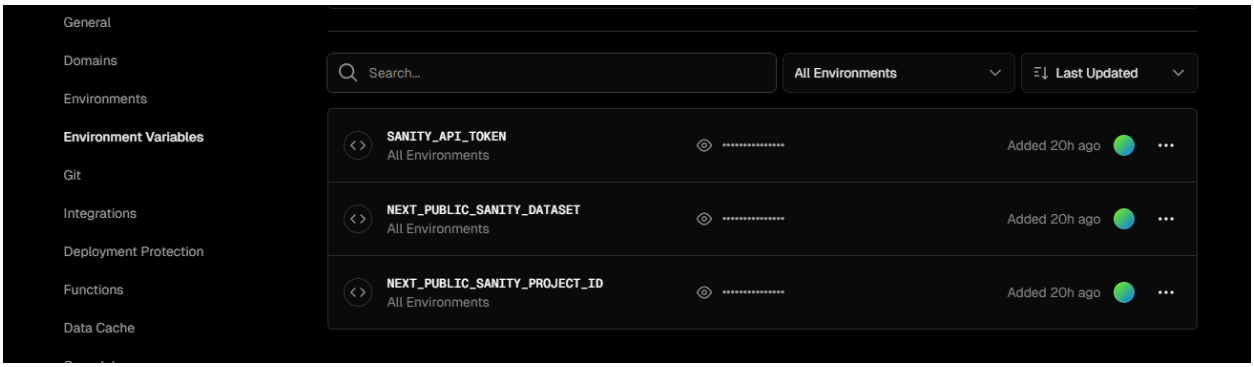
		interaction times						
TC009	Test on multiple browsers	Open site on Chrome, Firefox, Safari, and Edge	Consistent rendering and functionality	Consistent across all tested browsers	Passed	Medium	-	No issues found
TC010	Test responsiveness on physical device	Test on a mobile device > Verify layout	Layout adjusts properly	Responsive layout verified	Passed	Medium	-	Test successful
TC011	Simulate real-world user flow	Perform tasks like browsing, adding to cart, checkout	Tasks complete without errors	User flow works smoothly	Passed	High	-	Usability confirmed

Day 6: Deployment Preparation and Staging Environment Setup

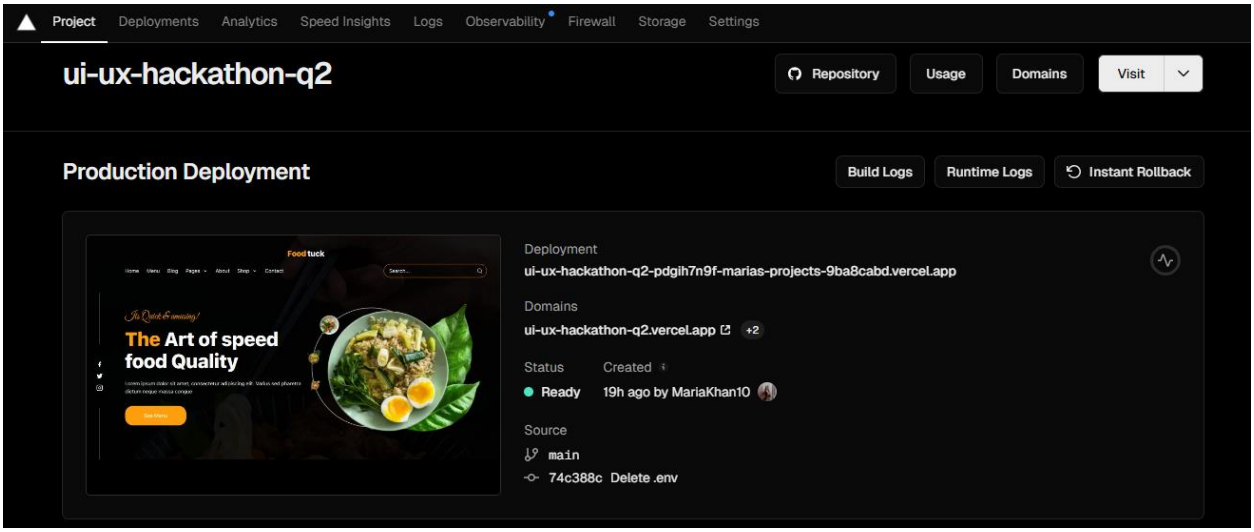
Objective:
Day 6 focused on preparing the marketplace for deployment by configuring a staging environment, ensuring seamless functionality in a production-like setup, and organizing all project files and documents for professional deployment.

Key Activities and Achievements

1. **Staging Environment Configuration:**
    - **Hosting Platform Selection: Vercel was evaluated for deployment.**
    - **GitHub Integration: Connected the repository to the selected hosting platform.**
    - **Build and Deployment Settings: Configured the settings to ensure successful staging builds.**
    - **Environment Variables: Secured sensitive data by setting up environment variables within the hosting platform.**



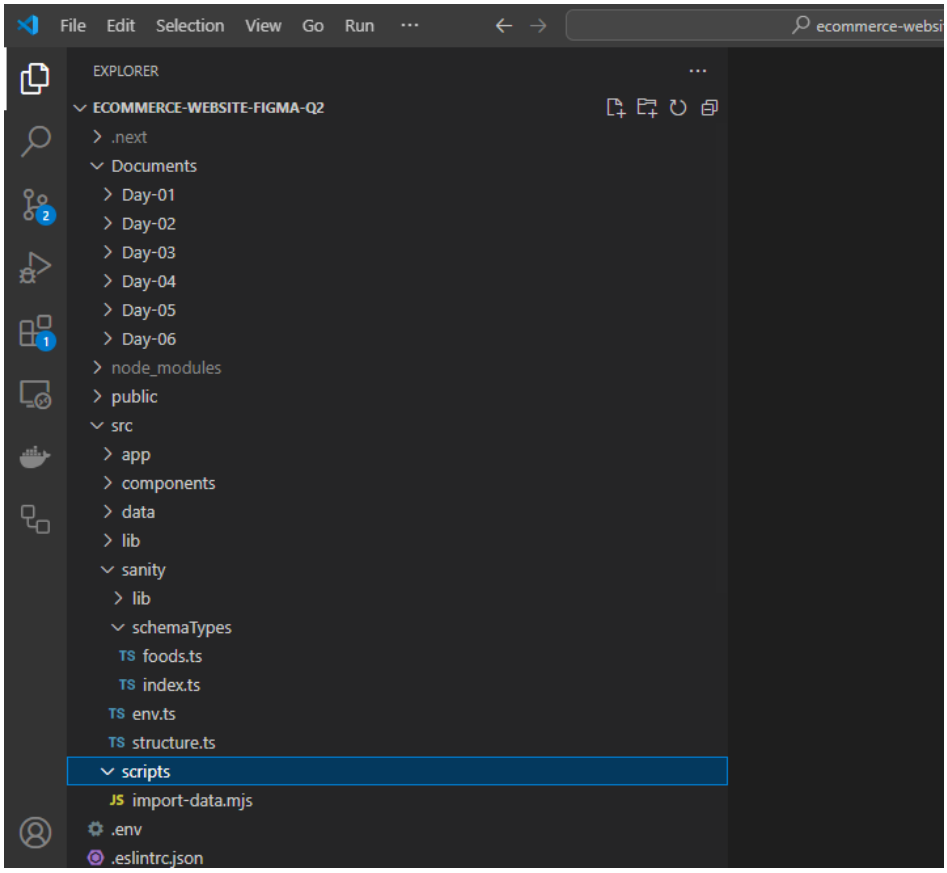
2. **Production-Like Testing:**
    - **Validated the application in a staging environment to simulate real-world production scenarios.**
    - **Identified and resolved any pre-deployment issues to enhance system readiness.**
3. **Professional Environment Management:**
    - **Learned about industry-standard practices for managing non-production and production environments (e.g., UAT, PROD, DR).**



4. **Documentation and Testing:**
    - **Conducted comprehensive staging environment testing and documented the results.**

- **Created deployment documentation, including performance reports and test cases.**

5. **GitHub Repository Organization:**
    - **Structured the repository with a clear folder hierarchy (documents/, src/, public/, etc.).**
    - **Ensured consistent naming conventions for files and folders.**
    - **Added a README.md file summarizing the project structure and contents for clarity.**



Key Learning Outcomes:

- **Mastered the process of configuring and testing a staging environment.**
- **Developed professional documentation and file organization practices, crucial for collaborative and scalable development.**

Final Outcome:

By the end of Day 6:

1.A fully deployed production environment for the marketplace.

2. Environment variables securely configured.

3. Test case and performance reports documenting staging tests.

4. All project files and documentation organized in a GitHub repository.

5. A professional README.md file summarizing project activities and results.