Hackathon Day 2: Technical Planning Documentation

Overview

This document outlines the technical strategy for an E-Commerce Marketplace to empower small businesses by providing a robust platform for online sales. It integrates ideas from Day 1 and adheres to Day 2 guidelines.

Key Technologies

· Frontend: Next.js

· CMS: Sanity

· Order Tracking: ShipEngine

· Database: MongoDB (authentication)

· Hosting: Vercel (frontend), AWS Lambda (backend)

· Payment Gateway: Stripe

Technical Architecture

1. Frontend (Next.js):

- Client-side rendering for speed.
- Server-side rendering for SEO.
- Integrated with Sanity CMS for dynamic content.

2. Backend:

- REST APIs for users, products, orders, and delivery zones.
- Integration with external services like ShipEngine and Stripe.

3. Database (MongoDB):

- NoSQL database with flexible schemas for scalability.

4. CMS (Sanity):

- Handles banners, featured products, and blogs.

5. Shipment Tracking (ShipEngine):

- Real-time tracking and shipment updates.

6. Authentication:

- Secure user data storage with bcrypt encryption.

7. Deployment:

- Frontend: Vercel with CI/CD.
- Backend: AWS Lambda for serverless architecture.

Core Components & Workflow

1. User Authentication:
    - MongoDB stores hashed passwords.
    - JWT issued for session management.
2. Content Management (Sanity):
    - Admins manage listings and dynamic content.
3. Product Browsing:
    - Next.js renders product pages dynamically.
    - APIs for CRUD operations on products.
4. Orders:
    - MongoDB stores order data.
    - ShipEngine tracks shipments.
5. Payments:
    - Stripe handles payments securely.
    - Cash-on-delivery option available.

API Endpoints

Technical Architecture

1. Frontend (Next.js):
    - Client-side rendering for speed.

- Server-side rendering for SEO.
- Integrated with Sanity CMS for dynamic content.

2. Backend:

   - REST APIs for users, products, orders, and delivery zones.
   - Integration with external services like ShipEngine and Stripe.

3. Database (MongoDB):

   - NoSQL database with flexible schemas for scalability.

4. CMS (Sanity):

   - Handles banners, featured products, and blogs.

5. Shipment Tracking (ShipEngine):

   - Real-time tracking and shipment updates.

6. Authentication:

   - Secure user data storage with bcrypt encryption.

7. Deployment:

   - Frontend: Vercel with CI/CD.
   - Backend: AWS Lambda for serverless architecture.

Core Components & Workflow

1. User Authentication:

   - MongoDB stores hashed passwords.
   - JWT issued for session management.

2. Content Management (Sanity):

   - Admins manage listings and dynamic content.

3. Product Browsing:

   - Next.js renders product pages dynamically.
   - APIs for CRUD operations on products.

4. Orders:

- MongoDB stores order data.
- ShipEngine tracks shipments.

5. Payments:

    - Stripe handles payments securely.
    - Cash-on-delivery option available.

API Endpoints

User Management:

- /api/auth/register – User registration
- /api/auth/login – User login

Product Management:

- /api/products – CRUD operations on products

Order Management:

- /api/orders – Manage orders

Payment Management:

- /api/payments – Handle payments

Deployment Plan

- Frontend: Vercel (auto-deploy from GitHub).
- Backend: AWS Lambda (serverless and scalable).
- Database: MongoDB Atlas (daily backups and horizontal scaling).

Security Measures

- HTTPS for secure communication.
- Bcrypt for password encryption.
- Role-based access control.
- PCI-compliant payment integration.

Timeline

- Day 3: Setup Next.js and Sanity, implement authentication.
- Day 4-5: Build product pages and order tracking integration.
- Day 6: Finalize payments and delivery zone features.
- Day 7: End-to-end testing and deployment.

Conclusion

This plan ensures a scalable and secure platform leveraging modern technologies to deliver a seamless user Experience.