Hackathon 3 Day 2

Technical Planning Documentation for Avion Marketplace

---

1. Overview

This document outlines the technical plan for **Avion**, an E-Commerce marketplace that empowers small businesses and artisans by providing a platform to sell **handmade furniture and homeware** products, such as lamps, pots, chairs, vases, and sofas. The marketplace aims to deliver a robust, scalable, and user-friendly platform for customers to browse, purchase, and track orders, while enabling sellers to easily list their products and manage orders.

---

2. Key Technologies

- **Frontend**: Frontend: Next.js with TypeScript and Tailwind CSS
- **CMS**: Sanity (for managing dynamic content like product listings)
- **Order Tracking**: ShipEngine (for real-time shipment updates)
- **Payment Gateway**: Stripe (for secure payment processing)
- **Hosting & Deployment**: Vercel (for frontend), AWS Lambda (for backend), MongoDB Atlas (for database)

---

3. Technical Architecture

System Overview

1. **Frontend (Next.js)**:

- **Client-side rendering** for speed and responsiveness.
- **Server-side rendering** for SEO and product page preloading.
- **Sanity CMS integration** to dynamically fetch content.

2. **Backend**:
    - **REST APIs** for managing users, products, orders, and deliveries.
    - Integration with **ShipEngine** for shipment tracking and **Stripe** for payment processing.
3. **CMS (Sanity)**:
    - Manages dynamic content like product listings.
    - Uses **GROQ queries** to fetch content and display it on the frontend.
4. **Payment Gateway (Stripe)**:
    - Handles secure payment processing.
    - Supports multiple payment options like credit card, Google Pay, Apple Pay, and Cash on Delivery (COD).
5. **Order Tracking (ShipEngine)**:
    - Real-time order tracking using **ShipEngine API**.
    - API endpoint for fetching shipment status.

---

4. System Components and Workflow

1. **User Registration/Login**:
    - **Input**: User credentials (email, password).
    - **Database**: MongoDB for storing user credentials securely.
    - **Outcome**: JWT token issued for session management.
2. **Content Management (Sanity CMS)**:

- **Admin Role**: Manages product listings.
- **Outcome**: Content is dynamically rendered on the frontend via **GROQ queries**.

3. **Product Browsing and Checkout**:
    - **Frontend**: Displays product details, fetched dynamically.
    - **API**:

      GET /products to list products,
      GET /products/:id for product details,
      POST /cart to manage shopping cart.

    - **Outcome**: Users can browse products, add them to cart, and proceed to checkout.
4. **Order Management**:
    - **API**:
      POST /orders to create orders,
      GET /orders/:id to fetch order details.

    - **Outcome**: Orders are processed, stored in MongoDB, and cannot be edited once created.
5. **Shipment Tracking**:
    - **Integration**: ShipEngine for tracking orders in real time.
    - **API Endpoint**: *GET /shipments/:orderId* for tracking shipment status.
    - **Outcome**: Users receive real-time delivery updates.
6. **Payment Processing**:
    - **API Endpoint**: *POST /payments* to initiate payments, *GET /payments/status* for payment status.
    - **Outcome**: Payment successful only when payment gateway confirms (Stripe, Jazz Cash, EasyPaisa, etc.).

---

5. API Endpoints

User Management

- *POST /api/auth/register*: Register a new user.
- *POST /api/auth/login*: Login and generate JWT token.
- *GET /api/users/profile*: Fetch user profile details.
- *PUT /api/users/update*: Update user details (profile, password).

Product Management

- *GET /api/products*: List all products.
- *GET /api/products/:id*: Fetch a specific product's details.
- *POST /api/products*: Add a new product (seller role required).
- *PUT /api/products/:id*: Edit product details (seller role required).
- *DELETE /api/products/:id*: Delete a product (seller role required).

Order Management

- *POST /api/orders*: Create a new order.
- *GET /api/orders*: List all orders for the authenticated user.
- *GET /api/orders/:id*: Fetch details of a specific order.

Payment Management

- *POST /api/payments*: Initiate a payment transaction.
- *GET /api/payments/status*: Check the status of a payment.

Shipment Management

- *POST /api/shipments*: Create a new shipment record.
- *GET /api/shipments/:orderId*: Track shipment status.

---

6. Data Schema

Users

- *user_id*: Unique identifier for the user.
- *username*: User's full name.
- *email*: User's email.
- *role*: User role (admin, seller, customer).
- *order_ids*: List of orders created by the user.

Products

- *product_id*: Unique identifier for the product.
- *name*: Name of the product.
- *price*: Price of the product.
- *description*: Product description.
- *image_url*: URL for the product image.

Orders

- *order_id*: Unique order identifier.
- *customer_id*: Reference to the customer placing the order.
- *products*: List of ordered product IDs.
- *total_price*: Total price of the order.
- *status*: Current status (e.g., Pending, Shipped, Delivered).
- *payment_status*: Payment confirmation (e.g., Paid, Unpaid).

---

7. Deployment Plan

- **Frontend (Next.js)**: Hosted on **Vercel**.
    - **CI/CD**: Automated deployment via GitHub.

- **Backend**: Serverless functions hosted on **AWS Lambda**.
    - **Scaling**: Automatic scaling based on traffic.
- **Database**: **MongoDB Atlas** for cloud database hosting with automated backups.
- **CI/CD**: GitHub Actions for continuous integration and deployment.

Conclusion

This technical plan outlines the necessary steps to develop **Avion**, a modern, secure, and scalable marketplace. The plan leverages state-of-the-art technologies and frameworks to deliver a seamless experience for customers and sellers. By following this roadmap, the project will provide a highly functional and flexible platform for the sale of handmade furniture and homeware products.