Author name: Iraj Fatima

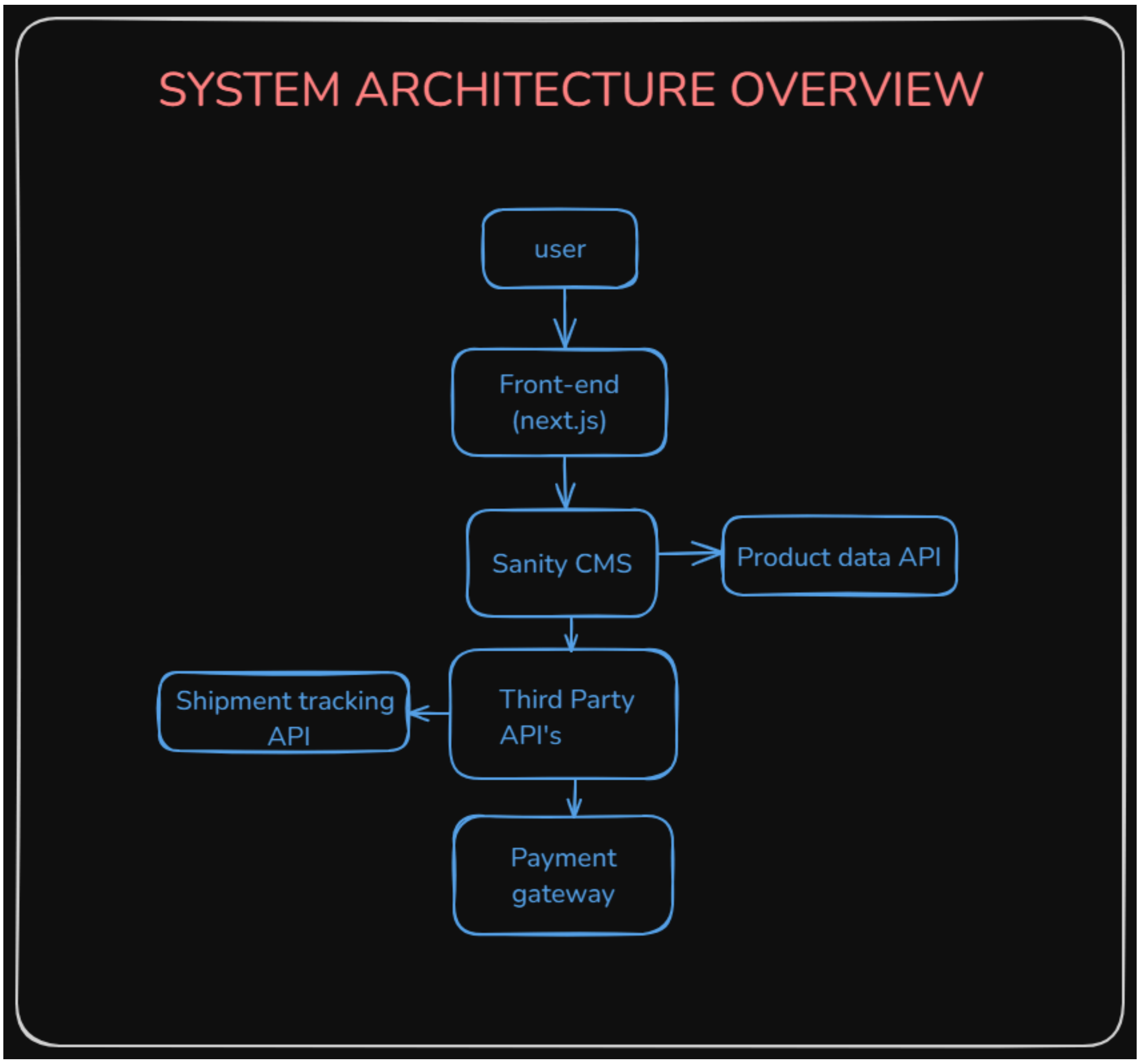
Roll no: 00225562
Slot: Friday (9am-12pm)

HACKATHON 3
DAY-2

Marketplace Technical Foundation - Avion

1- SYSTEM ARCHITECTURE OVERVIEW:

- **Diagram Overview:**



● Component Roles:

Customer Interaction:

Customers interact with the platform through their devices (smartphones, laptops, or tablets) to browse products, add items to the cart, and place orders. The user-friendly and responsive interface ensures a seamless shopping experience.

Frontend (Next.js):

The Next.js frontend handles all user interactions and fetches dynamic content like products, categories, and order details via APIs. Its responsive design ensures a consistent shopping experience across devices.

Key Features:

- **User-Friendly Interface**:
  - Easy navigation for browsing products, filtering options, and a clear layout for product information.
  - Minimalistic, clean design to ensure focus on products.
- **Responsive Design**:
  - Use responsive CSS frameworks or libraries (like Tailwind CSS) to adapt seamlessly for mobile, tablet, and desktop views.

Essential Pages:

1. **Home**:
   - Displays hero banners, featured products, and promotional content.
   - Components: Header, Hero, Popular, Footer.
2. **Product Listing**:
   - Shows all products with filters for categories, price, etc.
   - Components: ProductList, Filters.
3. **Product Details**:
   - Displays detailed product information, stock levels, and an "Add to Cart" button.
   - Components: ProductDetail, ImageGallery, Reviews.
4. **Cart**:
   - Lists all selected items, subtotal, and an option to proceed to checkout.
   - Components: CartItems, Summary.

5. **Checkout**:
    - Allows users to enter delivery details, review the order, and make payments.
    - Components: CheckoutForm, PaymentGateway.
6. **Order Confirmation**:
    - Displays order summary and confirmation details post-purchase.
    - Components: OrderSummary.

Sanity CMS:

Sanity CMS serves as the central content management system, managing:

- **Product Information**: Names, prices, descriptions, and images.
- **Inventory Levels**: Real-time stock updates to prevent over-ordering.
- **Promotional Content**: Banners and offers displayed on the frontend. Sanity allows instant updates, ensuring customers always see accurate and up-to-date information.

API Components

Product Data API

- **Purpose**: Supplies product information like pricing, availability, descriptions, and images.
- **Backend Role**: Communicates with Sanity CMS to fetch or update product details.
- **Frontend Role**: Displays fetched data to users in real-time.

Third-Party APIs

1.Shipment Tracking API (ShipEngine Integration):

- **Purpose**: Provides live tracking updates for customer orders.
- **Key Features**:
    - ❖ Real-time order status updates (e.g., in transit, delivered).
    - ❖ Integration with major shipping carriers for accurate delivery timelines.

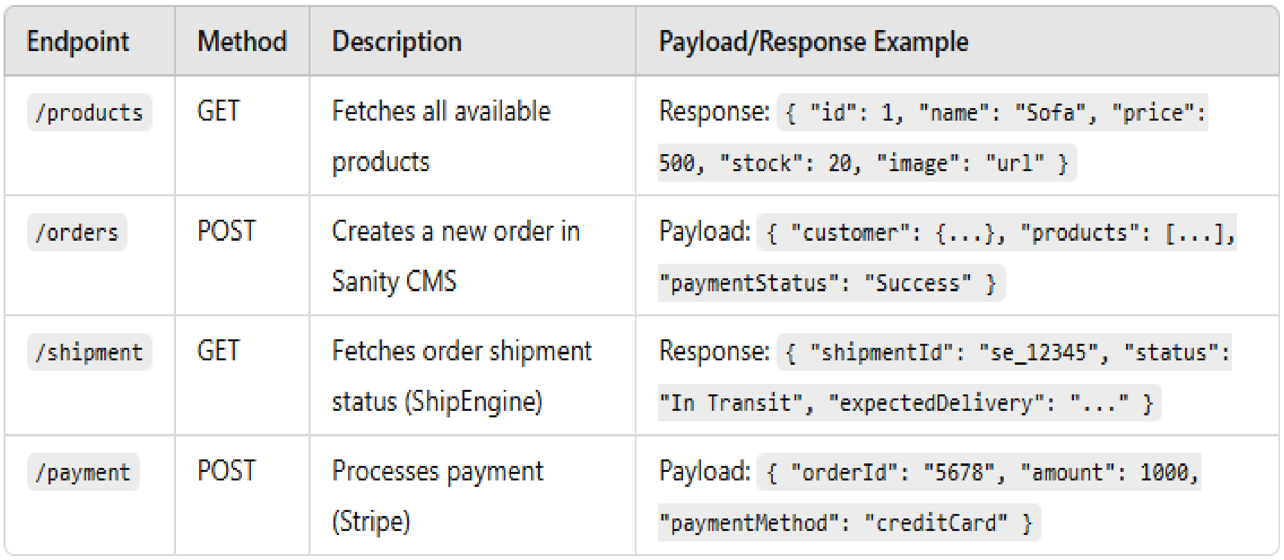
2.Payment Gateway API (Stripe Integration):

- **Purpose**: Facilitates secure payments for transactions, supporting multiple payment methods such as credit cards, wallets, and bank transfers.
- **Key Features**:
  - ❖ Tokenization for secure storage of sensitive customer payment data.
  - ❖ Real-time payment confirmation and receipt generation.

2.Category-Specific Instructions

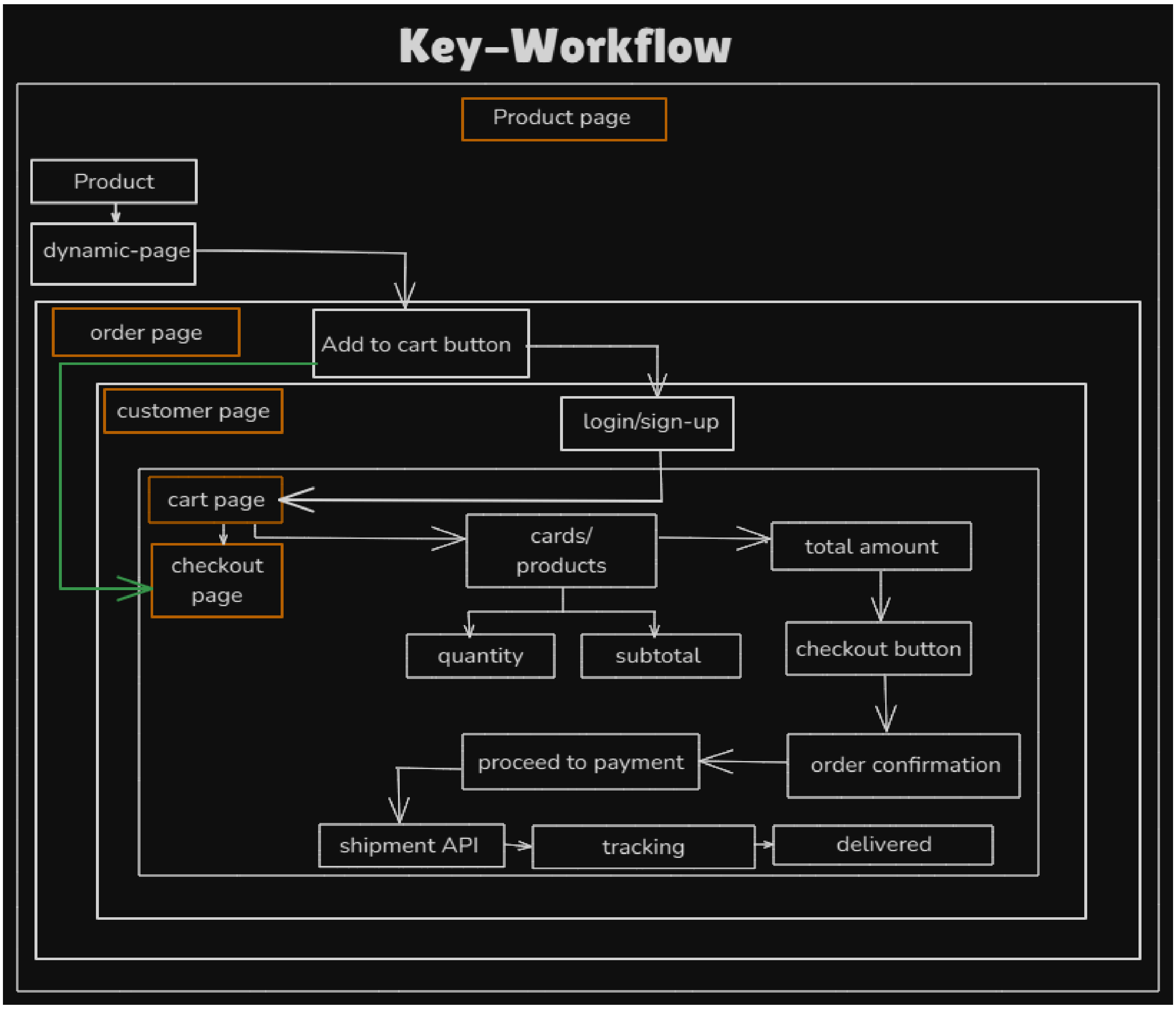
- **General E-Commerce**:
  - Focus: Product browsing, cart management, and order placement.
  - Example Workflows:
    - **Product Browsing**: Customer queries `/products` to fetch listings.
    - **Cart Management**: Update cart via `/cart` (POST/DELETE).
    - **Order Placement**: Submit orders via `/orders` (POST).

3.API SPecifications:



Endpoint	Method	Description	Payload/Response Example
`/products`	GET	Fetches all available products	Response: `{ "id": 1, "name": "Sofa", "price": 500, "stock": 20, "image": "url" }`
`/orders`	POST	Creates a new order in Sanity CMS	Payload: `{ "customer": {...}, "products": [...], "paymentStatus": "Success" }`
`/shipment`	GET	Fetches order shipment status (ShipEngine)	Response: `{ "shipmentId": "se_12345", "status": "In Transit", "expectedDelivery": "..." }`
`/payment`	POST	Processes payment (Stripe)	Payload: `{ "orderId": "5678", "amount": 1000, "paymentMethod": "creditCard" }`

4.Workflow:



- ➔ User Browses Products:
  - ◆ Customer opens the website.
  - ◆ Frontend fetches product data from `/products` via a GET request.
  - ◆ Sanity CMS provides product listings with real-time stock updates.
- ➔ User Adds Products to Cart:
  - ◆ User selects a product and clicks "Add to Cart."
  - ◆ Frontend sends the product ID and quantity to the cart service.
  - ◆ The cart service stores the data and calculates the subtotal.
- ➔ User Places Order:
  - ◆ Customer proceeds to checkout and enters delivery and payment details.
  - ◆ Frontend sends the order data to the backend via `/orders` (POST).

- ◆ Payment API processes the payment.
- ◆ Order details are stored, and inventory is updated in Sanity CMS.

5.Data schema design:

- Entities
  - ❖ Product: Represents the items available for purchase.
  - ❖ Order: Tracks customer orders and associated products.
  - ❖ Customer: Stores customer information and their order history.
  - ❖ Shipment: Tracks the delivery details of an order.
  - ❖ Payment: Tracks payment information for orders.
- Relationships
  - ❖ A Product can belong to multiple Orders.
  - ❖ An Order is linked to one Customer, one Shipment, and one Payment.
  - ❖ A Shipment belongs to one Order.
  - ❖ A Payment belongs to one Order.

Schema Design:

- **Product schema:**

```
export const productSchema = {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'id', type: 'string', title: 'Product ID', readOnly: true },
    { name: 'name', type: 'string', title: 'Name' },
     { name: 'description', type: 'text', title: 'Description' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock' },
    { name: 'category', type: 'string', title: 'Category' },
    { name: 'material', type: 'string', title: 'Material' },
    { name: 'color', type: 'array', of: [{ type: 'string' }], title: 'Color Options' },
    { name: 'tags', type: 'array', of: [{ type: 'string' }], title: 'Tags' },
    ],
  };
```

- **Order schema:**

```
export const orderSchema = {
        name: 'order',
       type: 'document',
        fields: [
              { name: 'id', type: 'string', title: 'Order ID', readOnly: true },
              { name: 'customerId', type: 'reference', to: [{ type: 'customer' }], title: 'Customer ID'
              },
              { name: 'products',
              type: 'array', of:
               [{ type: 'object',
               fields: [
              { name: 'productId', type: 'reference', to: [{ type: 'product' }], title: 'Product ID' },
               { name: 'quantity', type: 'number', title: 'Quantity' } ] }], title: 'Products', },
                     { name: 'orderDate', type: 'datetime', title: 'Order Date' },
                     { name: 'status', type: 'string', options: { list: ['Pending', 'Shipped', 'Delivered']
                     }, title: 'Order Status' },
                      { name: 'shippingAddress', type: 'string', title: 'Shipping Address' },
                      { name: 'paymentStatus', type: 'string', options: { list: ['Pending', 'Completed']
                     }, title: 'Payment Status' },
                     { name: 'totalPrice', type: 'number', title: 'Total Price' },
       ],
 };
```

- **Customer schema:**

```
export const customerSchema = {
        name: 'customer',
        type: 'document',
        fields: [
          { name: 'id', type: 'string', title: 'Customer ID', readOnly: true },
          { name: 'name', type: 'string', title: 'Full Name' },
          { name: 'email', type: 'string', title: 'Email Address' },
         { name: 'phone', type: 'string', title: 'Phone Number' },
          { name: 'addresses', type: 'array', of: [{ type: 'string' }], title:
         'Addresses', },
         { name: 'orderHistory', type: 'array', of: [{ type: 'reference', to: [{ type:
         'order' }] }], title: 'Order History', },
      ],
};
```

- **Shipment schema:**

```
export const shipmentSchema = {
    name: 'shipment',
    type: 'document',
    fields: [
      { name: 'id', type: 'string', title: 'Shipment ID', readOnly: true },
      { name: 'orderId', type: 'reference', to: [{ type: 'order' }], title: 'Order
      ID' },
      { name: 'status', type: 'string', options: { list: ['In Transit', 'Delivered'] },
      title: 'Shipment Status' },
      { name: 'shippedDate', type: 'datetime', title: 'Shipped Date' },
      { name: 'expectedDeliveryDate', type: 'datetime', title: 'Expected
      Delivery Date' },
      { name: 'carrier', type: 'string', title: 'Carrier' },
      { name: 'trackingNumber', type: 'string', title: 'Tracking Number' },
   ],
};
```

- **Payment schema:**

```
export const paymentSchema =
      {
      name: 'payment',
      type: 'document',
      fields: [
        { name: 'id', type: 'string', title: 'Payment ID', readOnly: true },
        { name: 'orderId', type: 'reference', to: [{ type: 'order' }], title: 'Order
        ID' },
        { name: 'amount', type: 'number', title: 'Amount' },
        { name: 'paymentMethod', type: 'string', options: { list: ['Credit Card',
        'PayPal', 'COD'] }, title: 'Payment Method' },
        { name: 'paymentStatus', type: 'string', options: { list: ['Pending',
        'Completed', 'Failed'] }, title: 'Payment Status' },
        { name: 'transactionDate', type: 'datetime', title: 'Transaction Date' },
      ],
  };
```

6.Technical Roadmap:

The technical roadmap outlines the step-by-step plan to develop, test, and deploy the marketplace efficiently while ensuring high-quality deliverables.

Phase 1: Planning

- ❖ **Duration**: 1 Week
- ❖ **Activities**:
  - ➢ Define project requirements and features.
  - ➢ Finalize architecture and technology stack (Next.js, Sanity CMS, ShipEngine, Stripe).
- ❖ **Deliverables**: Requirements document and system architecture diagram.

Phase 2: Backend Development

- ❖ **Duration**: 2 Weeks
- ❖ **Activities**:
  - ➢ Set up Sanity CMS schemas for products, orders, and customers.
  - ➢ Develop API endpoints for products (`/products`), orders (`/orders`), and shipment tracking (`/shipment`).
  - ➢ Integrate ShipEngine for shipments and Stripe for payments.
- ❖ **Deliverables**: Functional backend with APIs connected to Sanity CMS and third-party services.

Phase 3: Frontend Development

- **Duration**: 2 Weeks
- **Activities**:
  - Create responsive pages: Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation.
  - Fetch and display dynamic content via APIs.
  - Integrate Stripe for payment flows and ShipEngine for shipment tracking.
- **Deliverables**: User-friendly and responsive interface.

Phase 4: Testing

- **Duration**: 1 Week
- **Activities**:
  - Unit and integration tests for APIs.
  - End-to-end testing of key workflows (browsing, cart, checkout, shipment tracking).
  - Stress testing for scalability.
- **Deliverables**: Test reports and a stable application.

Phase 5: Deployment

- **Duration**: 1 Week
- **Activities**:
  - Deploy frontend (e.g., Vercel) and backend to production environments.
  - Set up CI/CD pipelines for seamless updates.
- **Deliverables**: Fully deployed and accessible live application.

Phase 6: Post-Launch

- **Duration**: Ongoing
- **Activities**:
  - Monitor performance and address issues.
  - Gather user feedback for improvements.

    - Add advanced features like personalized recommendations and loyalty programs.
- **Deliverables**: Continuous updates and feature enhancements.

Conclusion

This roadmap ensures a structured and timely execution of the project. Each phase focuses on critical milestones, leading to the delivery of a robust and scalable marketplace. Post-launch, ongoing support will refine the platform based on user feedback and evolving business needs.