FurniSphere: Technical Analysis and Documentation

1: Recap of Day 1: Business Focus Vision:

*Establishing a strong business foundation for **FurniSphere** by emphasizing unique selling points like personalized shopping, community-driven features, and sustainable products.*

Key Features:

AI-driven interior design assistance

AR visualization for real-time furniture fitting

Eco-friendly furniture options

Subscription model for exclusive access

Gamified shopping experience

Target Audience: *Focused on young professionals, families, and small businesses, addressing the pain point of visualizing and purchasing furniture online.*

2: Day 2 Transitioning to Technical Planning

Frontend Development Requirements

Technology Stack:

***Next.js** for fast, SEO-optimized pages that load quickly.*

React for reusable components and UI rendering.

***TailwindCSS** for responsive and visually appealing design.*

***AR** Integration with **React-Three/Fiber** for **3D** interactive models in AR.*

Backend Development Requirements

Sanity *CMS for content management and dynamic updates.*

Schema Development:

Custom schemas for products, customers, orders, and shipments.

Database Design: Scalable structure to store product and customer data.

Third-Party Integrations:

Payment APIs *(Stripe, PayPal)*

Shipment tracking *(ShipEngine, AfterShip)*

AI recommendations *(TensorFlow, Algolia)*

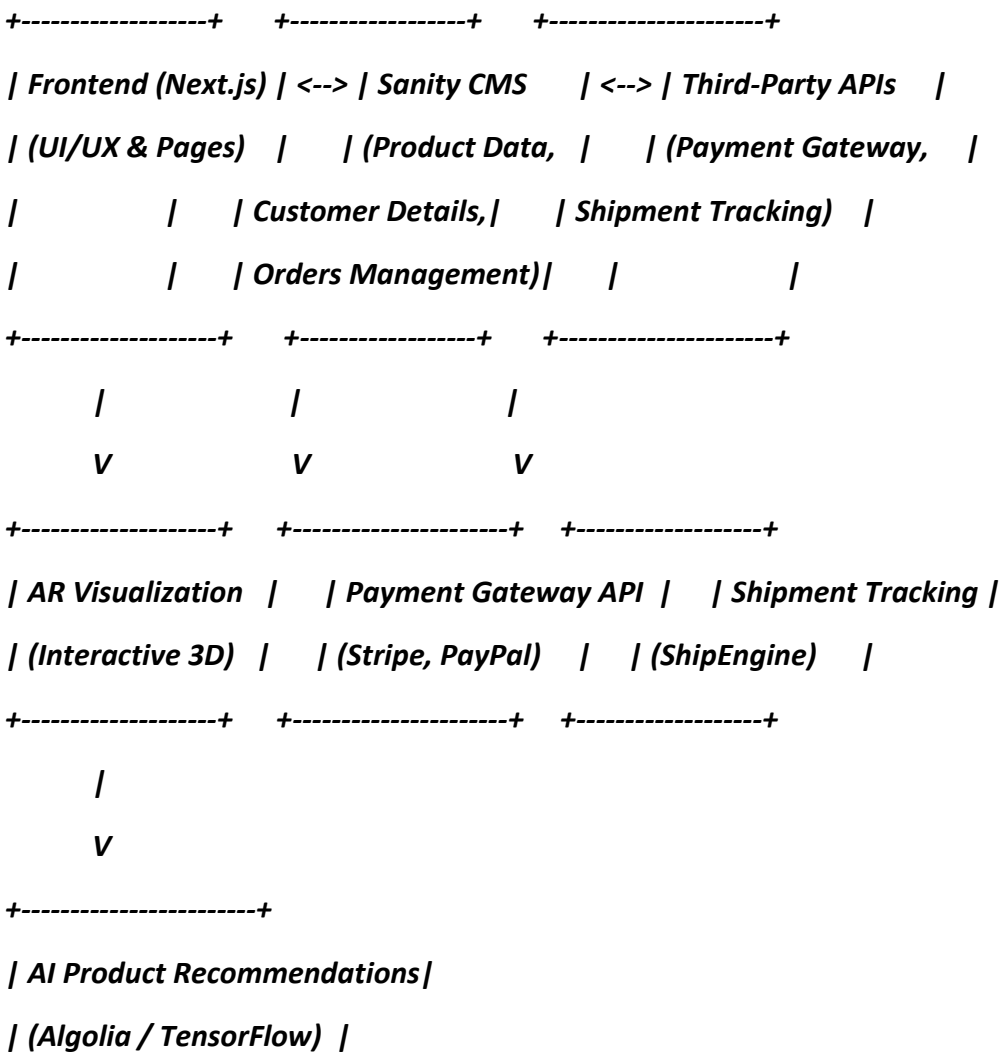
3. System Architecture Overview:

Frontend ***(Next.js****) interacts with* ***Sanity CMS*** *to fetch and display product data.*

Sanity CMS handles dynamic content, managing product data, customer orders, and real-time updates.

Third-party ***APIs*** *integrate* ***Stripe*** *for payment and* ***ShipEngine*** *for delivery tracking.*

System Architecture Diagram


Frontend (Next.js) <--> Sanity CMS <--> Third-Party APIs
(UI/UX & Pages) (Product Data, (Payment Gateway,
Customer Details,| Shipment Tracking)
Orders Management)|
AR Visualization
(Interactive 3D)
Payment Gateway API
(Stripe, PayPal)
Shipment Tracking
(ShipEngine)
AI Product Recommendations|
(Algolia / TensorFlow)

4. API Requirements & Endpoints

Product API

(GET, POST, PUT, DELETE)

GET /products: *Fetch all products available.*

Data: *{ "id": "123", "name": "Sofa", "price": 299, "stock": 100, "image_url": "url" }*

GET /products/{id}: *Fetch product details.*

Data: *{ "id": "123", "name": "Sofa", "description": "Comfortable leather sofa", "price": 299, "image_url": "url" }*

POST /products: *Add a new product.*

Data: *{ "name": "Sofa", "price": 299, "stock": 100, "category": "Living Room", "image_url": "url" }*

PUT /products/{id}: Update product details.

Data: *{ "price": 299, "stock": 95 }*

DELETE /products/{id}: Delete a product.

Data: *{ "id": "123" }*

Customer API

(POST, GET, PATCH)

POST /customers: *Register a new customer.*

Data: *{ "name": "Muqaddas Fatima", "email": "muqaddasfatima576@gmail.com", "password": "password123" }*

GET /customers/{id}: *Get customer details.*

Data: *{ "id": "123", "name": "Muqaddas Fatima", "email": "muqaddasfatima576@gmail.com" }*

PATCH /customers/{id}: *Update customer information (e.g., loyalty points, preferences).*

Data: *{ "loyalty_points": 100 }*

Order API

(POST, GET, PUT, DELETE)

POST /orders: *Place a new order.*

Data: *{ "customer_id": "1", "products": [{"id": "123", "quantity": 2}], "total_price": 598, "shipping_address": "address" }*

GET /orders/{id}: *Retrieve order details.*

Data: *{ "order_id": "1", "status": "Pending", "items": [{"product_id": "123", "quantity": 2}], "total_price": 598 }*

PUT /orders/{id}: *Update order status.*

Data: *{ "status": "Shipped" }*

DELETE /orders/{id}: *Cancel an order.*

Data: *{ "id": "123" }*

Shipment API

(GET, POST)

GET /shipment/{orderId}: *Retrieve shipment tracking information.*

POST /shipment: *Create a new shipment.*

Payment API

(POST)

POST /payment: *Initiate payment through Stripe/PayPal.*

AI/Recommendation API

(GET)

GET /recommendations: *Fetch personalized product recommendations.*

5. Authentication & Authorization

Authentication:

*Clerk handles user authentication via **OAuth 2.0 (Google, Facebook)** or **JWT** for maintaining sessions.*

Authorization:

Role-Based Access Control (RBAC):

***Admin Role:** Full access (can add/update/delete products, manage orders).*

***Customer Role:** Access to browse products, place orders, view order history.*

***Guest Role:** Limited access (can only browse products).*

Validation:

*Use **Joi** or **Yup** for schema validation to ensure correct and secure data handling.*

6. Brief Schema Definitions

Product Schema:

{

name: String,

description: String,

price: Number,

stock: Number,

image_url: URL,

category: String,

isFeatured: Boolean

}

Endpoints: *Used in GET /products and POST /products.*

Order Schema:

{

customer_id: String,

products: Array,

total_price: Number,

shipping_address: String,

order_status: { type: String, enum: ["Pending", "Shipped", "Delivered"] },

created_at: Date

}

Endpoints: *Used in POST /orders, GET /orders/{id}, PUT /orders/{id}.*

Customer Schema:

{

name: String,

email: String,

password: String,

shipping_address: String,

order_history: Array

}

Endpoints: *Used in POST /customers and GET /customers/{id}.*

AI Recommendations Schema:

{

customer_id: String,

recommended_products: Array

}

Endpoints: *Used in GET /recommendations.*

8. Security Best Practices

Rate Limiting & CAPTCHA:

Protect against brute force attacks by implementing rate limiting and ***CAPTCHA*** *for login and checkout processes.*

Data Encryption:

Ensure sensitive customer data is encrypted in transit (using ***HTTPS****) and at rest (using encryption mechanisms in the database).*

Two-Factor Authentication (2FA): *Implement* ***2FA*** *for users and admins for an additional layer of security.*

9. Testing and Documentation

API Testing: *Use* ***Postman*** *to test API endpoints.*

Frontend Testing: *Ensure components are rendered properly, especially interactive features like* ***AR****.*

Integration Testing: *Ensure seamless interaction between* ***frontend*** *and* ***backend****.*

Real-Time Data Testing: *Test Sanity* ***CMS*** *real-time data updates and ensure they are reflected on the frontend.*

By defining clear API requirements, schemas, and system architecture, we're setting a solid foundation for FurniSphere's success. This comprehensive documentation will ensure the platform is scalable, efficient, and delivers a seamless user experience, blending cutting-edge technologies with real-world needs.

As we move forward, it's important to remember that this isn't just about building another tech product—it's about creating something that makes shopping for furniture an engaging, personalized experience for people everywhere. Each feature, from AR fitting to AI recommendations, reflects our commitment to innovation and customer satisfaction.

We're all part of this journey, and our collective efforts will bring FurniSphere to life in a way that impacts both the business world and the daily lives of our users. Let's build something great together!

Thank You✨ (✿◕‿◕)

— Muqaddas Fatima