Hackathon 3 Day 2

Planning the Technical Foundation

Marketplace Technical Foundation - Woodcraft Furniture E-commerce

Technical Requirements:

Frontend:

- Build a modern frontend using HTML, Next.JS
- Tailwind CSS for styling

Backend:

- Managed by Sanity CMS to store and update product, customer, and order data

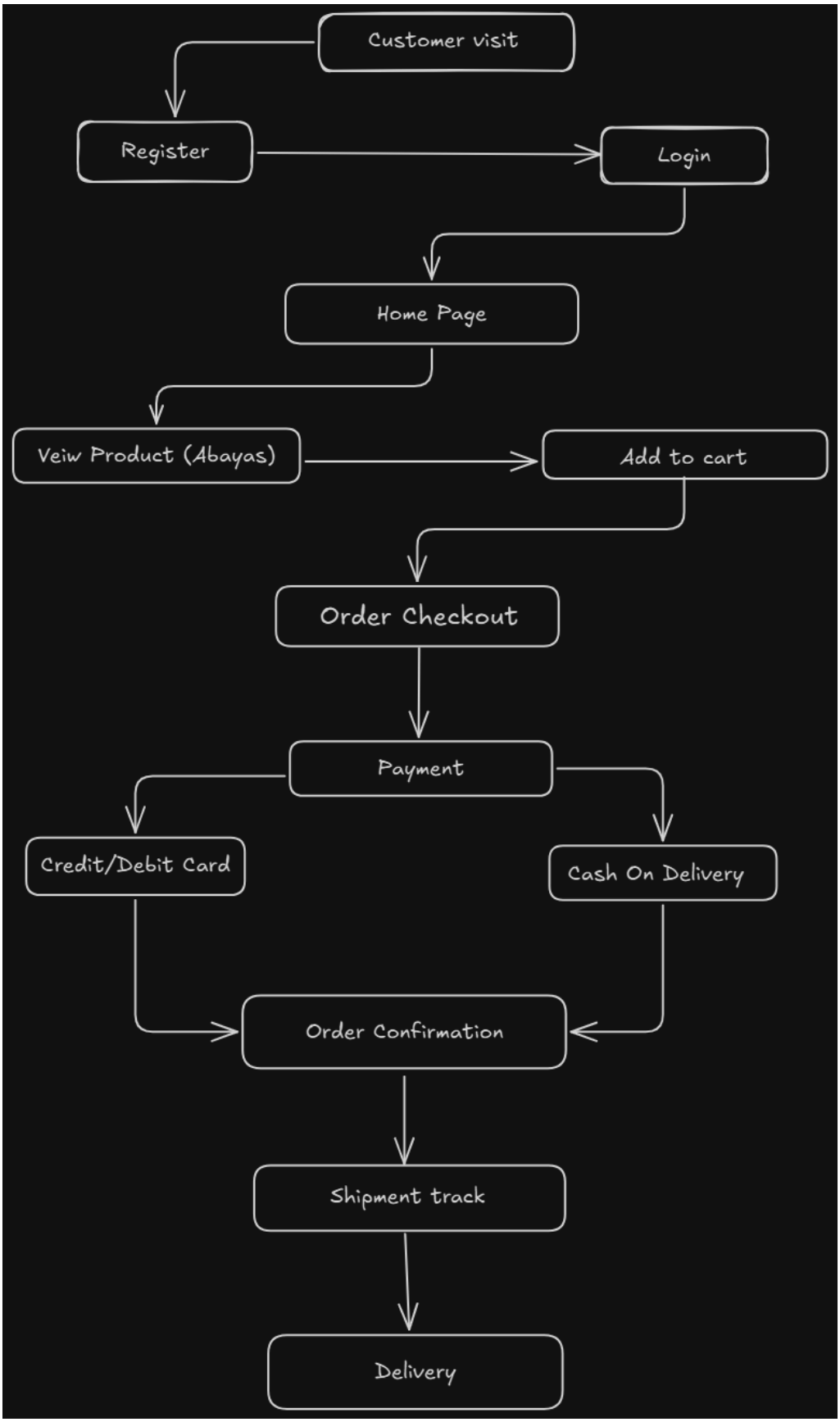
APIs:

- Integration with Third-party APIs for:
  - Shipping APIs
  - Payment APIs
  - Order tracking APIs

Key Features:

- Product listing, filtering, and search for furniture categories
- Secure checkout with multiple payment options
- Real-time shipment tracking
- Hassle-free return and exchange process

Frontend Requirement Flow Chart:

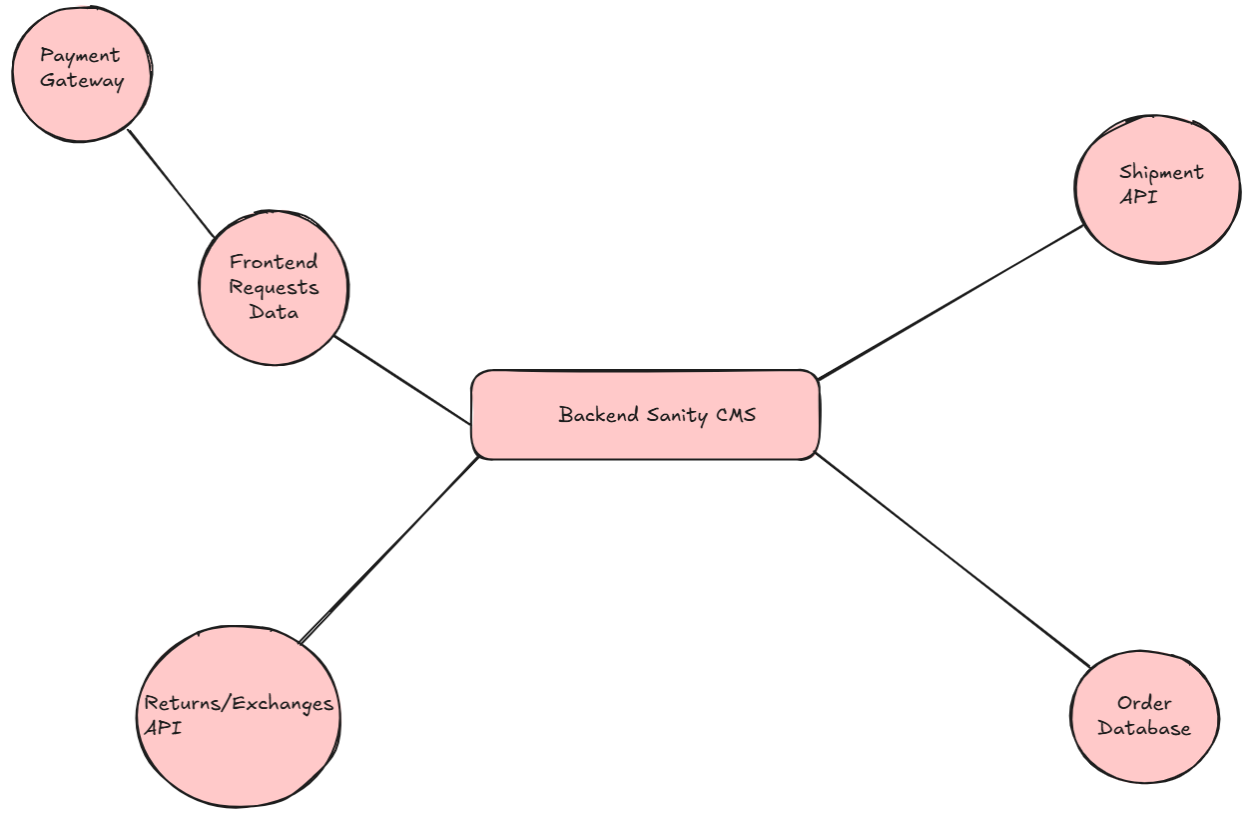

Customer visit
Register
Login
Home Page
Veiw Product (Abayas)
Add to cart
Order Checkout
Payment
Credit/Debit Card
Cash On Delivery
Order Confirmation
Shipment track
Delivery

Frontend Requirements:

- Users receive notifications and can leave product reviews
- User-friendly interface tailored for browsing furniture collections
- Responsive design for mobile and desktop users
- Essential pages: Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation

---

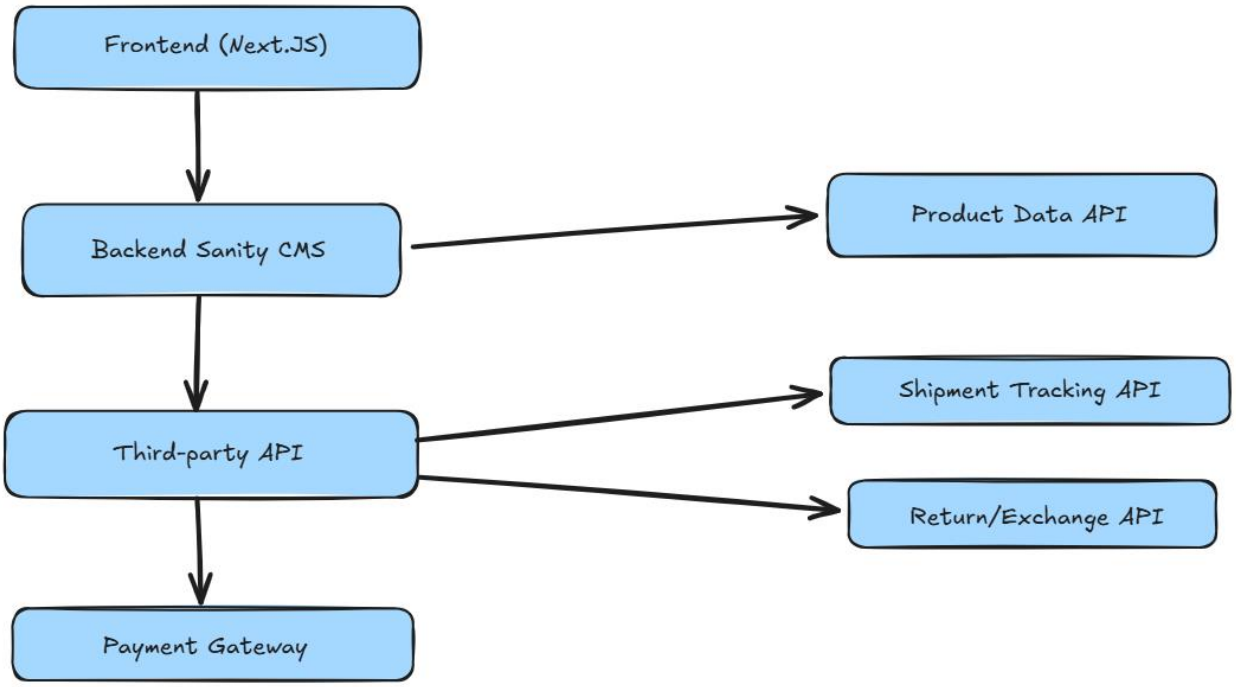
Backend Requirement Flow Chart:



Backend Requirements:

- Frontend sends product data requests to the CMS
- Sanity CMS handles product, order, and customer data
- API fetches shipment tracking information
- Processes payments securely via a third-party payment gateway
- Records order details and updates statuses in the database
- Handles returns and exchanges via API and CMS integration

Design System Architecture



Components:

Frontend:

- Framework: Next.JS
    - Purpose: User interaction and display

Backend:

- Tool: Sanity CMS
    - Purpose: Stores product, customer, and order data

Third-party APIs:

- **Payment Gateway:** Processes secure payments
- **Shipment Tracking API:** Updates and displays delivery status
- **Returns and Exchange API:** Handles user return requests

EndpointsAPI’s

API Endpoints for Woodcraft Furniture

Endpoint	Method	Description
/api/products	GET	Fetch a list of all products (query params: category, priceRange)
/api/products/:id	GET	Fetch details of a specific product by ID
/api/products	POST	Add a new product (Admin only)
/api/products/:id	PUT	Update an existing product by ID (Admin only)
/api/products/:id	DELETE	Delete a product by ID (Admin only)
/api/orders	POST	Create a new order
/api/orders/:id	GET	Fetch details of a specific order by ID
/api/orders/:id/status	PATCH	Update the status of an order by ID (Admin only)
/api/customers/:id	GET	Fetch details of a specific customer by ID
/api/customers	POST	Register a new customer
/api/returns	POST	Create a new return or exchange request
/api/returns/:id	GET	Fetch details of a return or exchange request by ID

SchemaSanity.Js

```
export const schemaTypes = [
  productSchema,
  orderSchema,
  customerSchema,
  returnExchangeSchema,
];


// Product Schema
export const productSchema = {
  name: "product",
  type: "document",
  title: "Product",
  fields: [
    { name: "name", type: "string", title: "Product Name" },
    { name: "price", type: "number", title: "Price" },
    { name: "description", type: "text", title: "Description" },
    { name: "stock", type: "number", title: "Stock Level" },
    { name: "category", type: "string", title: "Category" },
    { name: "dimensions", type: "string", title: "Dimensions" },
    { name: "material", type: "string", title: "Material" },
    { name: "images", type: "array", of: [{ type: "image" }], title: "Product Images" },
  ],
```

```
};


// Order Schema
export const orderSchema = {
  name: "order",
  type: "document",
  title: "Order",
  fields: [
    { name: "customer", type: "reference", to: [{ type: "customer" }], title: "Customer" },
    { name: "orderDate", type: "datetime", title: "Order Date" },
    { name: "items", type: "array", of: [{ type: "reference", to: [{ type: "product" }] }], title: "Ordered Items" },
    { name: "totalAmount", type: "number", title: "Total Amount" },
    { name: "paymentStatus", type: "string", title: "Payment Status", options: { list: ["Paid", "Pending", "Failed"] } },
    { name: "shipmentStatus", type: "string", title: "Shipment Status", options: { list: ["Pending", "Shipped", "Delivered", "Returned"] } },
  ],
};


// Customer Schema
export const customerSchema = {
  name: "customer",
  type: "document",
  title: "Customer",
```

```
  fields: [
    { name: "name", type: "string", title: "Customer Name" },
    { name: "email", type: "string", title: "Email Address" },
    { name: "phone", type: "string", title: "Phone Number" },
    { name: "address", type: "text", title: "Shipping Address" },
  ],
};

// Return/Exchange Schema
export const returnExchangeSchema = {
  name: "returnExchange",
  type: "document",
  title: "Return or Exchange",
  fields: [
    { name: "order", type: "reference", to: [{ type: "order" }], title: "Order" },
    { name: "reason", type: "text", title: "Reason for Return/Exchange" },
    { name: "status", type: "string", title: "Status", options: { list: ["Requested", "In Progress", "Completed"] } },
    { name: "dateRequested", type: "datetime", title: "Date Requested" },
  ],
};
```