MY MARKETPLACE BUSINESS PLAN DAY 2.

Marketplace Technical Foundation

[SHOP.CO]

On the first day of my last hackathon, I wrote about my business goals. First, I defined my E-Commerce business, explained why I chose E-Commerce, and outlined the goals for my marketplace. After planning and brainstorming, I figured out how to start my marketplace business and identified the challenges I would face in establishing my future marketplace. I also developed a strategy for my E-Commerce marketplace and considered why the industry would accept my marketplace. Additionally, I highlighted the unique aspects of my marketplace. Now, I have a clear understanding of how a beginner can start their own E-Commerce marketplace business. Moving forward, I will focus on brainstorming the next level, which involves the technical foundation.

My next steps will include setting goals and creating a technical plan for my marketplace E-Commerce business, including the design of the system architecture, APIs, and more.

Transitioning to Technical Planning

According to my thinking which technical requirement we need when we do technical planning I'm going to breakdown my planning:

1. **Frontend Requirements:**

    - Firstly, I've to think about my E-commerce interface, I've to create responsive and user-friendly interface for browsing data, Now I can include compulsory pages in my hackathon like;
        - **Home Page:** I've to display featured products and categories like that I want to sell.
        - **Product Listing Page:** Now I've to showcase my products with detailed information about each product.
        - **Product Details Page:** I've to provide detailed information about each product.

- **Cart Page:** I've to create add to cart, allow users to view and modify selected items.
- **Checkout Page:** I've to users to complete their purchase.
- **Order Conformation Page:** I've to define also user summary like completed order.

2. **Backend Requirements with Sanity CMS:**
    - Secondly, I've to jump on my backend requirements for my hackathon the most frequently backend system is sanity CMS because I've used it many times in my project. Here is my backend requirements:
        - **Managing Product Data:** I've to use sanity CMS for managing product data like (product name, product price, product description, product images etc ...)
        - **Storing Customer Details:** Now I've to create a backend data system for storing my customer details like (customer name, customer email, customer address etc ...)
        - **Recording Order Information:** I've to create a schema for my order information like (order items, order total price, order status etc ...)

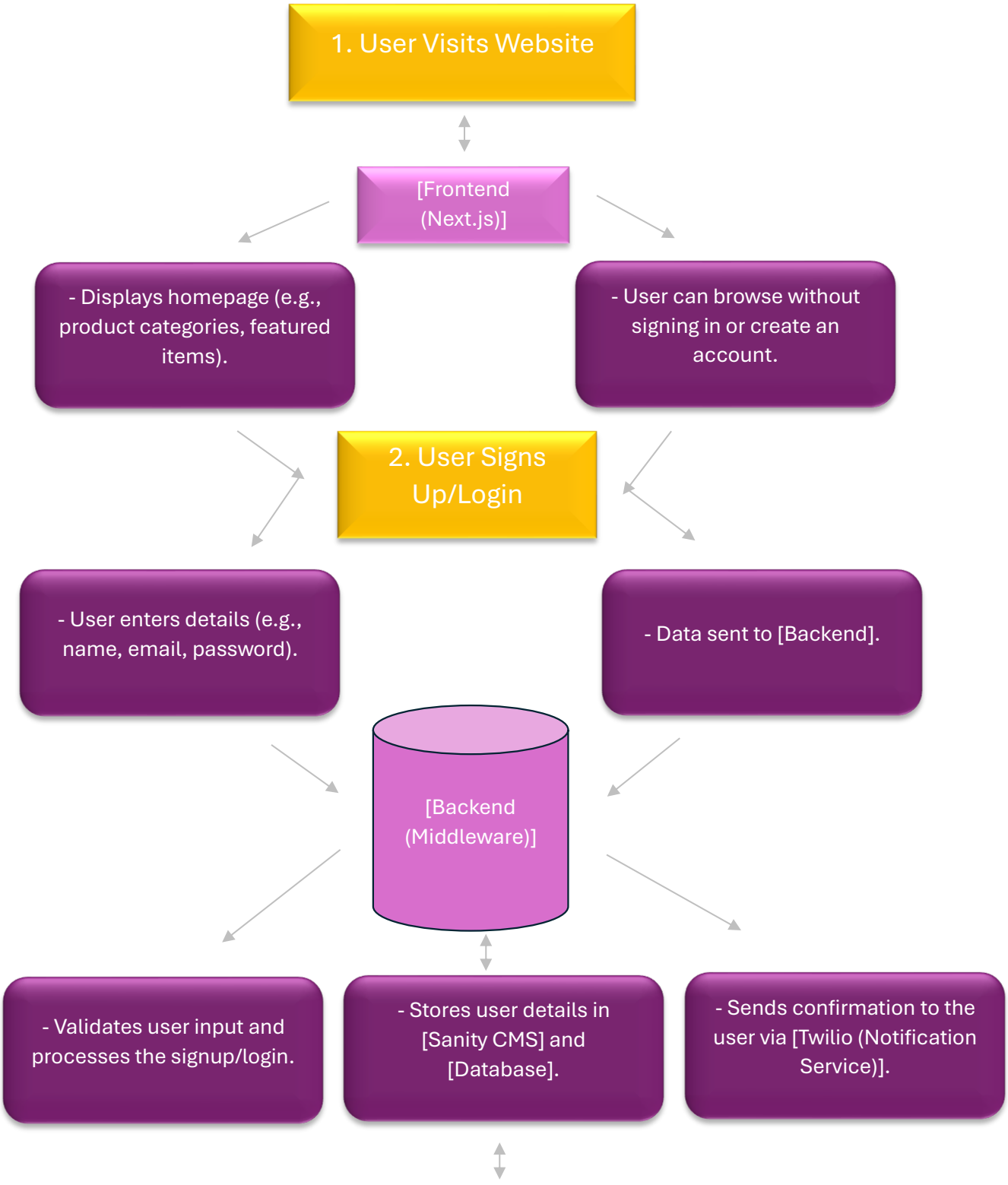
3. **Third-Party API's:**
    - Now I've to use third-party API's for my hackathon by using this we have to use some essential API's like:
        - **Shipment Tracking API:** we have to use third-party API's for tracking and shipment orders and fetch real-time tracking updates for order for this we can use shippo, shipengine and any other
        - **Payment Gateway API's:** for process security payment and store transaction details like we can use paypal, stripe, and any other.

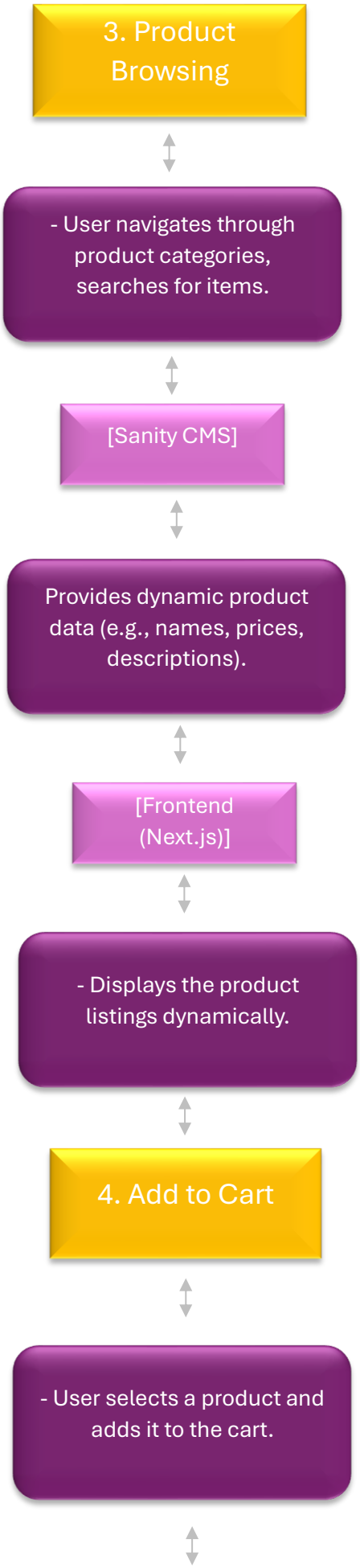
at the end we have to ensure like that's all process API's provide all required data for seamless frontend functionality.

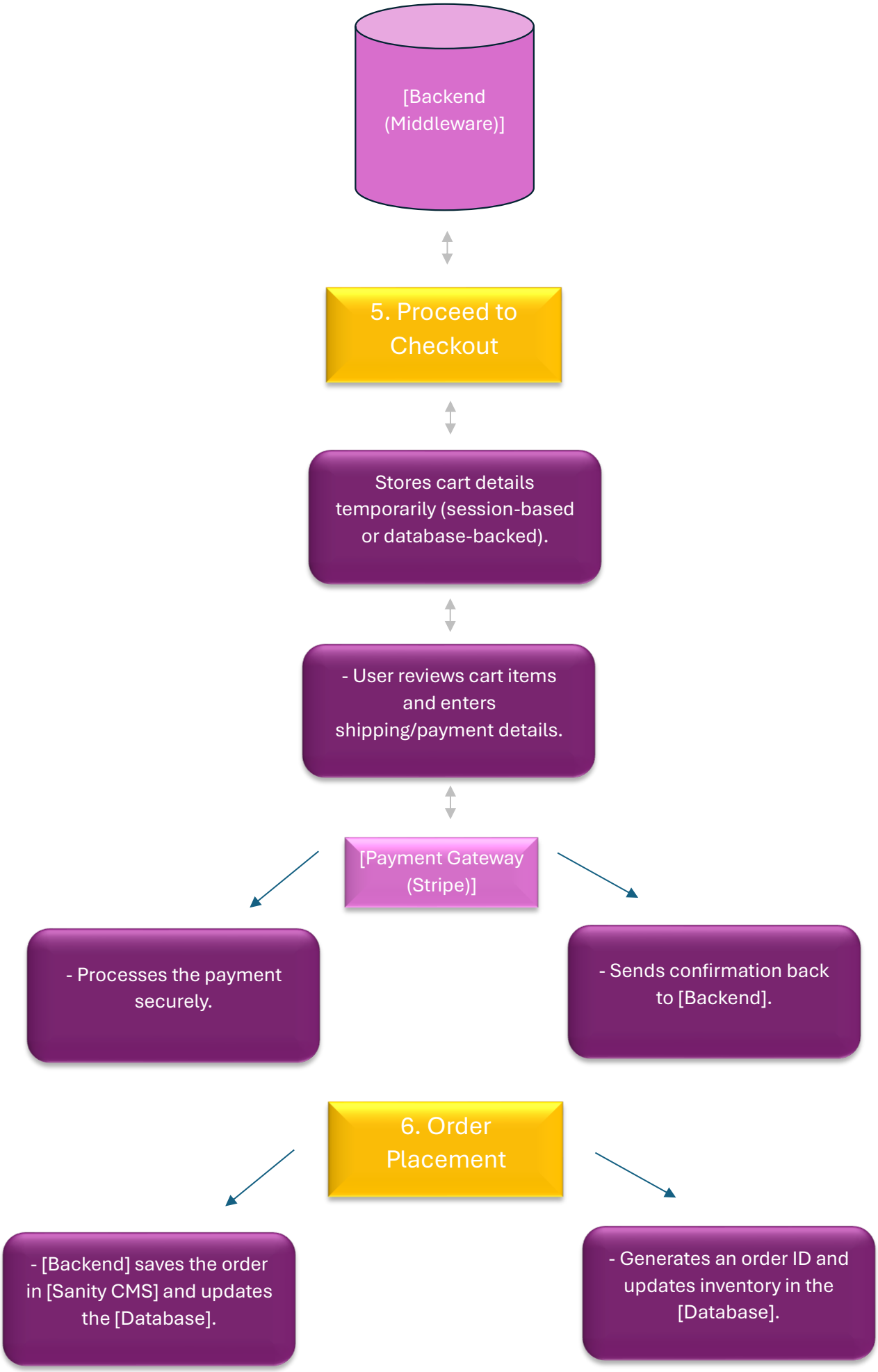
Design System Architecture

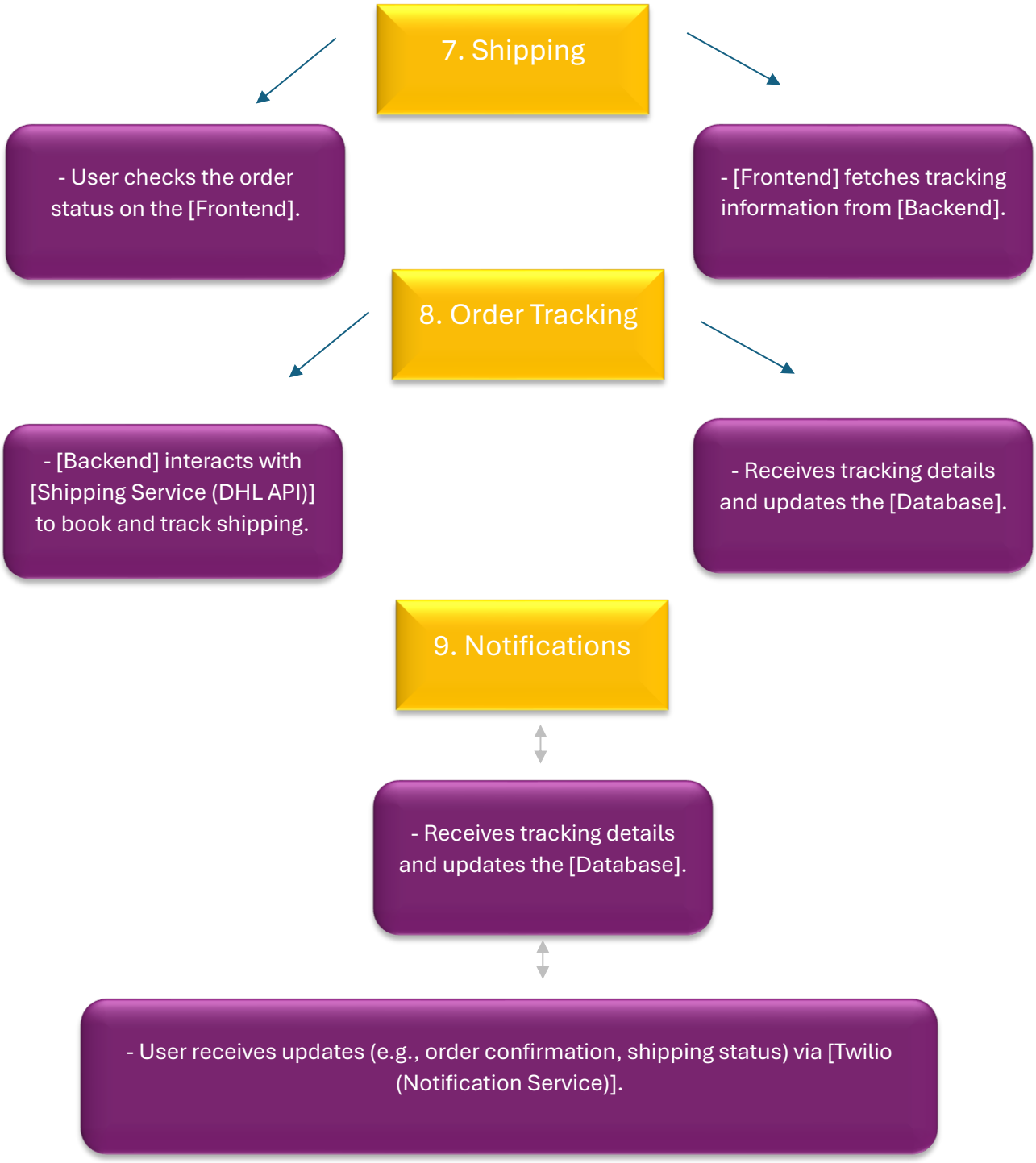
The system starts when a user visits the website, where the frontend provides a dynamic interface. Users can sign up or log in, with their details securely stored in the backend and

CMS. They browse products fetched from Sanity CMS and add items to their cart. At checkout, the backend processes their order, updates the database and inventory, and securely handles payment via a gateway like Stripe. Shipping details are managed through third-party APIs (e.g. shippo, shipengine), with tracking provided to the user. Notifications, like order confirmation and updates, are sent via services like Twilio, ensuring a seamless user experience from browsing to delivery.




3. Product Browsing
- User navigates through product categories, searches for items.
[Sanity CMS]
Provides dynamic product data (e.g., names, prices, descriptions).
[Frontend (Next.js)]
- Displays the product listings dynamically.
4. Add to Cart
- User selects a product and adds it to the cart.


[Backend (Middleware)]
5. Proceed to Checkout
Stores cart details temporarily (session-based or database-backed).
- User reviews cart items and enters shipping/payment details.
[Payment Gateway (Stripe)]
- Processes the payment securely.
- Sends confirmation back to [Backend].
6. Order Placement
- [Backend] saves the order in [Sanity CMS] and updates the [Database].
- Generates an order ID and updates inventory in the [Database].



Here is the short summary od my design system architecture:

1. User Visits Website:

- The journey starts when the user lands on your website. The frontend, powered by Next.js, provides a responsive and dynamic interface.

2. User Signup/Login:

- The user signs up or logs in. The backend handles validation and securely stores their data in the CMS and database. A confirmation notification is sent.

3. Product Browsing:

- Users browse products with data fetched dynamically from Sanity CMS, ensuring the latest information is displayed.

4. Add to Cart and Checkout:

- Items are added to the cart, and users proceed to checkout where they enter payment and shipping details.

5. Payment Processing:

- The payment gateway (e.g., Stripe) handles secure transactions and sends confirmation to the backend.

6. Order Placement:

- The backend records the order in Sanity CMS and updates the inventory in the database.

7. Shipping and Tracking:

- The backend records the order in Sanity CMS and updates the inventory in the database.

8. Notifications:

- Users receive notifications (order confirmation, shipment status, etc.) through Twilio or email.

My Plan API Requirements

I'm going to breakdown my API's requirements firstly I've to understand the requirements.

Step 1: Understanding the API's Requirements:

For this I've to create my E-Commerce workflow and marketplace type. My API's would manage the data flow between the frontend, backend (Sanity CMS), and third-party services.

For this My Main Goals would be:

1. Fetching data for the frontend (e.g. products, categories)
2. Record user actions (e.g. placing orders)
3. Interact with third-party services (e.g. shipment tracking, payment gataway).

Step 2: My API's Documentation:

Here is my API's structure documentation for each marketplace:

1. **Product API**
   Fetch details of all available products in my project from the sanity CMS.
   - **Endpoint Name**: /products
   - **Method**: GET
   - I've to fetch all available products including e.g. name, price, stock, description, images etc

   Response Example:

```
[
    {
        "id" : 1,
        "name" : "Product A" ,
        "Price" :  100,
        "stock" : 50,
        "image": "ProductA.jpg"
    },
    {
        "id" : 2,
        "name" : "Product B" ,
        "Price" :  200,
        "stock" : 20,
        "image": "ProductB.jpg"
    }
]
```

2. **Single Product API**
   Now I've to fetch detailed information about specific product.
   - **Endpoint Name**: /products/{id}
   - **Method**: GET

- I've to fetch all available single products detailed using its unique ID.

  Response Example:

  {
  "id" : 1,
  "name" : "Product A" ,
  "description" : "High-quality product A" ,
  "Price" : 100,
  "stock" : 50,
  "image": "ProductA.jpg" ,
  "categories": ["New Arrival", "Top Selling"]
  },

3. **Add to Cart API**

I've to create a product to the user's cart.

- **Endpoint Name**: /cart
- **Method**: POST
- I've to implement functionality add items to cart.

Request Payload:

{
"userId" : 123,
"productId" : 1,
"quantity" : 2
},

Response Example:

{
"cartId" : 456,
"productId" : " Item added to cart successfully" ,
},

4. **View Cart API**

Now I've to retrieve the items in the user's cart.

- **Endpoint Name**: /cart/{userId}
- **Method**: GET
- I've to fetch all items added to the cart by a specific user.

```
{
“cartId” : 456,
“userId” :123,
“items” :  [

      {
            “productId” :1,
             “name” : ”Product A”,
            “price” : 100,
            “quantity” :  2,
            “total” : 200,
      },
      {
            “productId” : 2,
             “name” : ”Product B”,
            “price” : 200,
            “quantity” :  1,
            “total” : 200,
      },
            ],


            “cartTotal” : 400,
},
```


5. Place Order API

I’ve to create a new order and save it in sanity CMS.

- **Endpoint Name**: /orders
- **Method**: POST
- I’ve to submit the user’s order and save details in sanity CMS.

Request Payload:


```
{
“userId” : 123,
“userId” : 456,
“paymentStatus” : ”Paid”,
“deliveryAddress” : ”123 Main Street, City, Country”,

},
```


Response Example:

```
{
“orderId” :789,
“status” : ”Order Placed” ,
“estimatedDelivery” : “2025-01-18” ,
},
```

6. **Order Details API**

Now I’ve to fetch details of a specific order.

- **Endpoint Name**: /orders/{orderId}
- **Method**: GET
- I’ve to retrieve the order’s details, including products, total cost, and delivery status

Response Example:

```
{
“orderId” : 789,
“userId” :123,
“items” :  [
            “productId” :1,
             “name” : ”Product A”,
            “price” : 100,
            “quantity” :  2,
      },
      {
            “productId” : 2,
             “name” : ”Product B”,
            “price” : 200,
            “quantity” :  1,
      },
            ],

           “orderTotal” : 400,
          “paymentStatus” : ”Paid”,
          “deliveryStatus” : ”Shipped”,
         “deliveryDate” : ”2025-01-18”,

      },
```

7. **Shipment Tracking API**

Track the shipping status of an order.

- **Endpoint Name**: /shipment/{orderId}
- **Method**: GET
- I've to fetch the current shipment status of an order using its ID

Response Example:


```
{
"shipmentId" : 12345,
"orderId" : 789,
"status" : "In Transit",
"expectedDeliveryDate" : "2025-01-18",

},
```


8. **Payment Processing API**

Process payments securely using a third-party gateway.

- **Endpoint Name**: /payment
- **Method**: POST
- Now I've to create process payment for an order

Request Payload


```
{
"orderId" :789,
"amount" :400,
"paymentMethod" : "Credit Card",
"cardDetails" : {
"cardNumber" : "1234-5678-9876-5432",
"expiryDate" : "01/26",
"cvv" : "123",
}
```


Response Payload


```
{

"paymentId" : 56789,
"amount" : 789,
"status" : "Payment Successful",
"cardDetails" : {
"cardNumber" : "1234-5678-9876-5432",
"transactionDate" : "2025 – 01 – 16T10:30:00Z",
}
```


9. **User Registration API**

Now I've to create a Register a new user

- **Endpoint Name**: /users/register
- **Method**: POST
- Create a new user account

Request Payload

```
{
"name" : "John Doe",
"email" : "john.doe@example.com",
"password" : "securepassword",
}
```

Response Payload

```
{
"userId" :123,
"status" : "Registration Successful",
}
        {
```

10. **User Login API**

Now authenticate a user and generate a token.

- **Endpoint Name**: /users/login
- **Method**: POST
- Login in a user and return an authentication token.

Request Payload

```
{
"email" : "john.doe@example.com",
"password" : "securepassword",
}
```

Response Payload

```
{
"token" : "abc123xyz456",
"status" : "Login Successful",
}
```

Now these all API's end point is essential for e-commerce backend system. Whenever we create and plan e-commerce website we have to use these all essential concepts

Design System Architecture

Now I've made this diagram by my own in this diagram I've explain the cycle of work flow how we do work in our technical journey . Now I'm going to breakdown my diagram;

1. Frontend (React/Next.js):

- ❖ The user interacts with the interface (e.g., viewing products, adding to cart, placing orders).
- ❖ Sends requests to the backend for data and actions.

2. Backend (Middleware):

- ❖ Acts as a bridge between the frontend and other components.
- ❖ Processes business logic, such as validating orders or calculating totals.
- ❖ Sends and receives data from the database, CMS, and third-party APIs.

3. Database:

- ❖ Stores all important information like:

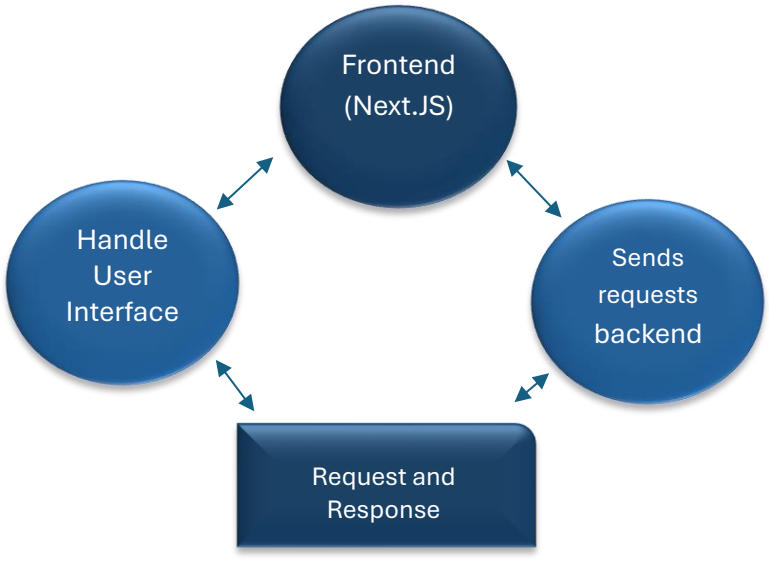
1) User details (e.g., accounts, preferences).
2) Products (e.g., inventory, pricing).
3) Orders (e.g., order history, status).

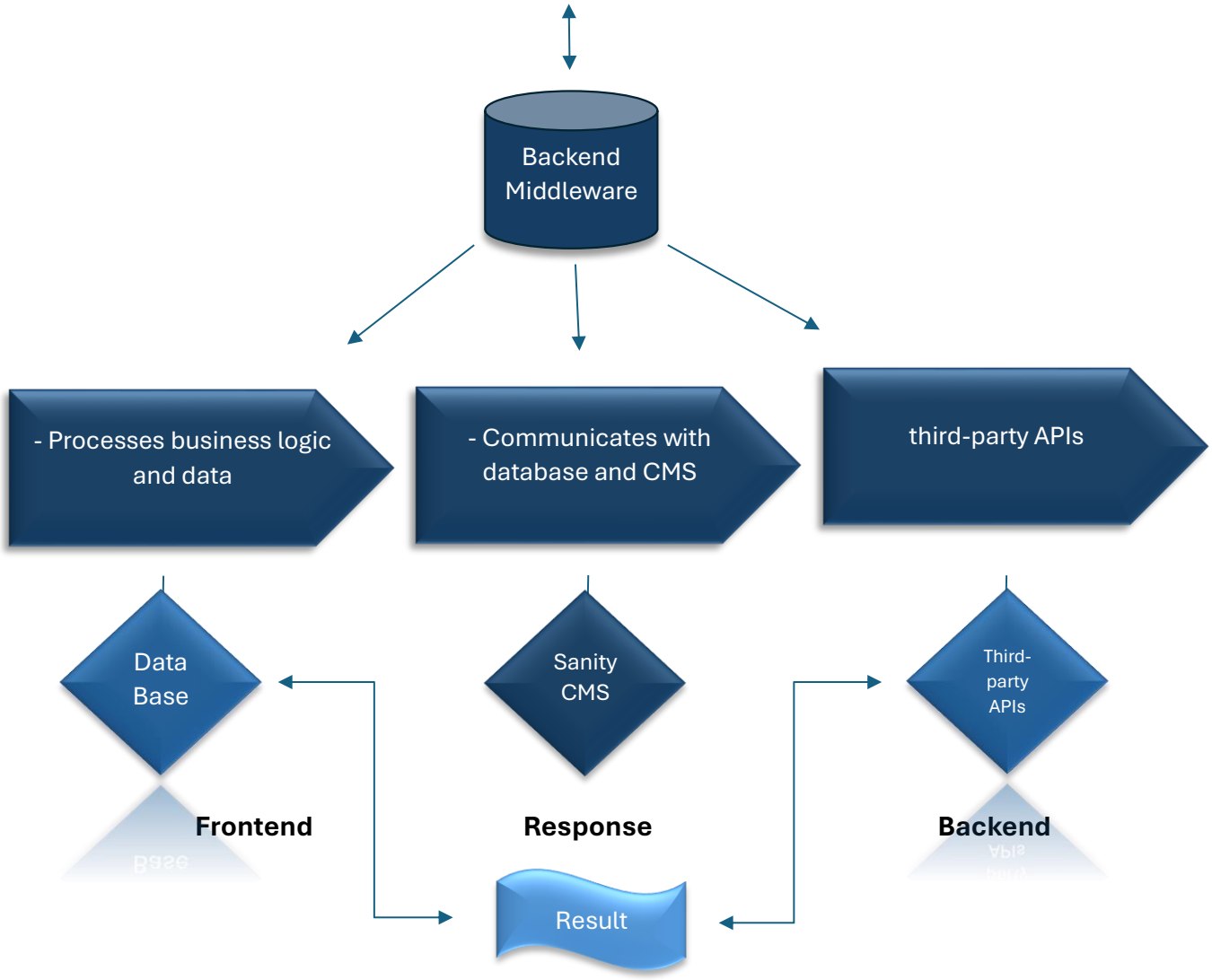
4. Sanity CMS:

- ❖ Manages dynamic content like product information, categories, and promotions.
- ❖ Allows easy updates for non-technical users without touching code.

5. Third-party APIs:

- ❖ Processes payments securely and sends confirmation.
- ❖ Tracks shipments and provides delivery updates




Backend
Middleware
- Processes business logic and data
- Communicates with database and CMS
third-party APIs
Data Base
Sanity CMS
Third-party APIs
Frontend
Response
Backend
Result