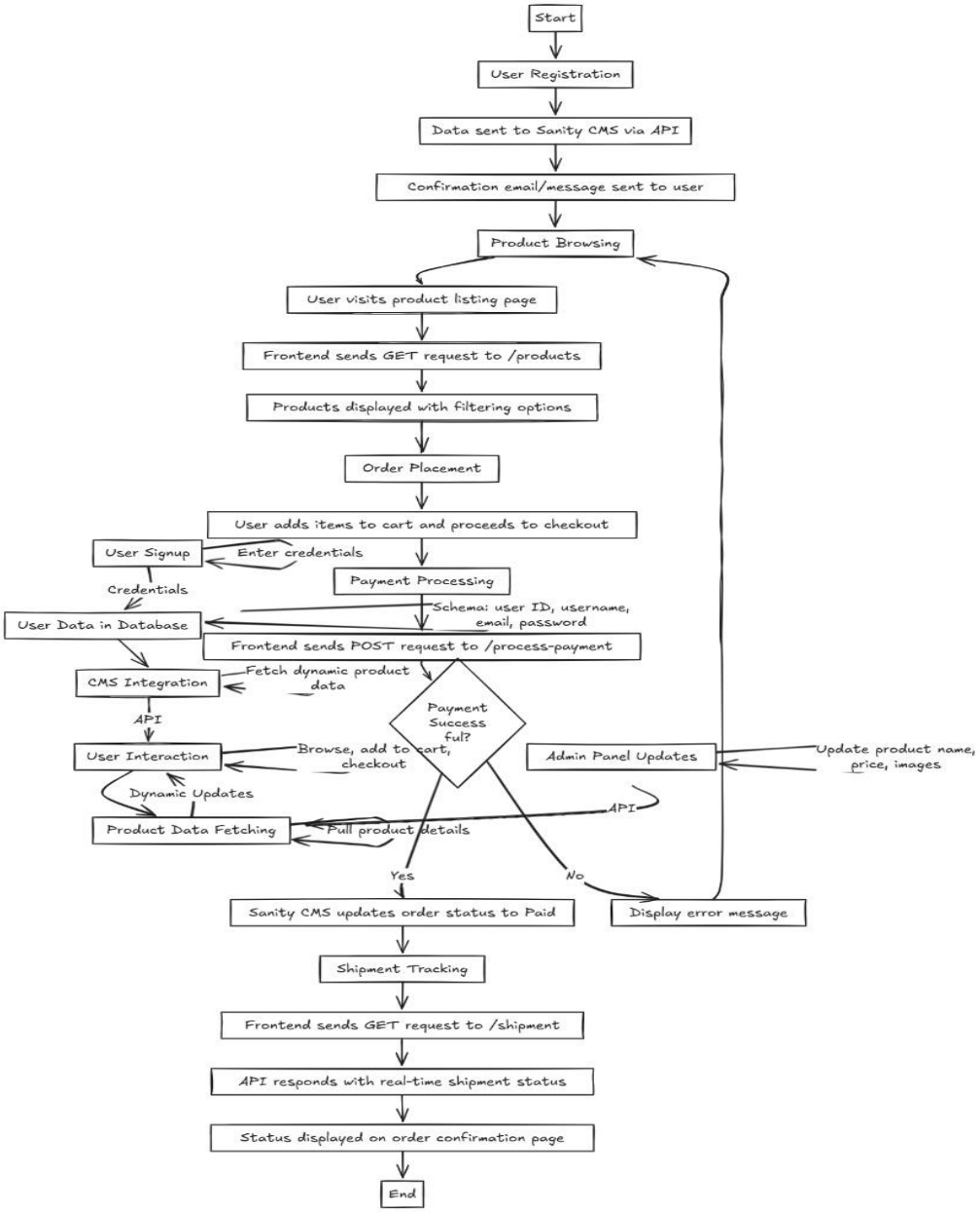
Marketplace Technical Foundation – HS General E-Commerce

1. System Architecture Overflow:

Architecture Diagram:



2. Key Workflows:

- User Registration

- User registers via the frontend.
- Data is sent to Sanity CMS via API and stored.
- A confirmation email or message is sent to the user.

3 - Product Browsing

- User visits the product listing page.
- Frontend sends a GET request to the/products endpoint of the Product Data
- API (Sanity CMS).
- Products are dynamically displayed with filtering options (e.g., "Buy" or "Rent").

4 - Order Placement

- User adds items to the cart and proceeds to checkout
- Frontend sends a POST request to the orders endpoint in Sanity CMS.
- Sanity CMS records the order with status Pending.

5 - Shipment Tracking

- Frontend sends a GET request to the/shipment endpoint of the Shipment Tracking API with the order ID.
- API responds with real-time shipment status (e.g., "In Transit").
- Status is displayed on the order confirmation page.

6 - Payment Processing

- Frontend sends a POST request to the /process-payment endpoint of the Payment Gateway API with payment details.
- Payment is processed securely, and the API return a success or failure response.
- Sanity CMS updates the order status to Paid upon success.

3 – Data Schema:

- **Entities**:
- **Products**: Name, description, price, category, image URLs, stock.
- **Customers**: Name, email, shipping addresses, wishelist, reviews.
- **Orders**: Order ID, product IDs, customer info, payment status, shipping details.
- **Payments**: Transaction ID, amount, payment method, date.

- **Sellers**: Store name, product list, ratings.
- **Reviews**: Ratings, feedback, customer name.
- **Subscriptions**: Customer ID, products, delivery frequency.

Schema Diagram:

```
Customers ---< Orders >--- Products >--- Categories
       |                  |           |
  Reviews & Ratings      Payments   Sellers
       |
   Subscriptions
```

4 - API Endpoints:

Authentication

1. **User Registration**
2. **Endpoint**: `/api/auth/register`
3. **Method**: POST
4. **Description**: Register a new user.
5. **Request Example**:

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

Response Example:

```
{
    "message": "User registered successfully",
```

```
    "userId": 123
}
```

6. **User Login**

Endpoint: `/api/auth/login`

Method: POST

Description: Log in an existing user.

Request Example:

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

Response Example:

```
{
    "message": "Login successful",
    "token": "your-jet-token"
}
```

Products

1. **Get All Products**

Endpoint: `/api/products`

Method: GET

Description: Fetch all available products.

Response Example:

```
[
    {
        "id": 1,
        "name": "Natural Face Wash",
        "price": 25.99,
        "category": "Skincare",
        "image": "url-to-image"
    },
    {
        "id": 2,
        "name": "Herbal Shampoo",
        "price": 15.99,
        "category": "Haircare",
        "image": "url-to-image"
    }
]
```


2. **Get Product Details**

Endpoint: `/api/products/:id`

Method: GET

Description: Fetch details of a single product by ID.

Response Example:


```
{
    "id": 1,
    "name": "Natural Face Wash",
    "price": 25.99,
    "category": "Skincare",
    "description": "Gentle face wash with organic ingredients.",
    "image": "url-to-image"
}
```


3. **Search Products**

Endpoint: `/api/products/search`

Method: GET

Query Parameters: `?q=face wash`

Description: Search products by name or category.

Response Example:

```
[
    {
        "id": 1,
        "name": "Natural Face Wash",
        "price": 25.99
    }
]
```

Categories

1. **Get Categories**

Endpoint: `/api/categories`
Method: GET
Description: Fetch all product categories.
Response Example:

```
[
    "Skincare",
    "Haircare",
    "Bath & Body",
    "Wellness"
]
```

Cart

1. **Add Item to Cart**

Endpoint: `/api/cart`

Method: POST

Description: Add an item to the user's cart.

Request Example:

```
{
    "productId": 1,
    "quantity": 2
}
```

Response Example:

```
{
    "message": "Item added to cart successfully"
}
```

2. **View Cart**

Endpoint: `/api/cart`

Method: GET

Description: Fetch all items in the user's cart.

Response Example:

```
[
    {
        "productId": 1,
        "name": "Natural Face Wash",
        "quantity": 2,
        "price": 25.99
    }
]
```

3. **Remove Item from Cart**

Endpoint: `/api/cart/:productId`

Method: DELETE

Description: Remove a product from the cart by product ID.

Response Example:

```
{
    "message": "Item removed from cart successfully"
}
```

Orders

1. **Place Order**

Endpoint: `/api/orders`

Method: POST

Description: Create a new order.

Request Example:

```
{
    "cart": [
        {
            "productId": 1,
            "quantity": 2
        }
    ],
    "paymentMethod": "Credit Card",
    "shippingAddress": "123 Street, City, Country"
}
```

Response Example:

```
{
    "message": "Order placed successfully",
    "orderId": 456
```

```
}
```

2. **View Orders**

Endpoint: `/api/orders`

Method: GET

Description: Fetch all orders for the logged-in user.

Response Example:

```
[
    {
        "orderId": 456,
        "total": 51.98,
        "status": "Processing"
    }
]
```

Reviews

1. **Add a Review**

Endpoint: `/api/reviews`

Method: POST

Description: Add a review for a product.

Request Example:

```
{
    "productId": 1,
    "rating": 5,
    "comment": "Amazing product!"
}
```

Response Example:

```
{
    "message": "Review submitted successfully"
}
```

2. **Get Reviews for a Product**

Endpoint: `/api/reviews/:productId`

Method: GET

Description: Fetch all reviews for a specific product.

Response Example:

```
[
    {
        "user": "John Doe",
        "rating": 5,
        "comment": "Amazing product!"
    }
]
```