MARKETPLACE TECH FOUNDATION DOCS

MARKETPLACE TECHNICAL FOUNDATION DOCUMENTATION

Title: Marketplace Technical Foundation Documentation
Author: Anum Naz
Roll Number: #00222580
Date: 18-1-2025

SYSTEM ARCHITECTURE OVERVIEW

The system architecture of the Marketplace platform is designed to provide a seamless user experience while efficiently managing data and integrating third-party services. The architecture consists of three main components: the Frontend (built using Next.js), the Sanity CMS, and various third-party APIs.

DIAGRAM

```
[Frontend (Next.js)]
       |
       |---> [Sanity CMS]
       |
       |---> [3rd-Party APIs]
```

COMPONENTS AND THEIR ROLES

Frontend (Next.js):

- The Frontend serves as the user interface of the marketplace, providing users with a responsive and interactive experience.
- It handles API communication to fetch and display data, ensuring that users can seamlessly navigate through products, manage their accounts, and place orders.

- Key functionalities include user registration, product browsing, cart management, and order confirmation.

Sanity CMS:

- Sanity CMS acts as the central content management system for the marketplace.
- It stores and manages product data, user details, and order history, allowing for easy updates and scalability.
- The CMS facilitates content creation and editing, ensuring that the marketplace has up-to-date information available for users.

3rd-Party APIs:

- These APIs provide additional functionalities necessary for enhancing the marketplace experience.
- They include services for shipment tracking, payment processing, and other integrations that contribute to a comprehensive user experience.
- The APIs communicate with the Frontend to deliver real-time updates on order status and facilitate secure transactions.

KEY WORKFLOWS

In the Marketplace Technical Foundation, several key workflows facilitate user interaction and streamline operations. Below is a detailed step-by-step guide for each of these workflows: user registration, product browsing, order placement, and shipment tracking.

1. USER REGISTRATION

Step 1: The user navigates to the registration page and inputs their personal details, including name, email, and password.

Step 2: Upon clicking the 'Register' button, the frontend sends a request to the Sanity CMS to store the user data.

Step 3: Sanity CMS processes the request and saves the user information securely in the database.

Step 4: A confirmation email is generated and sent to the user’s email address to verify their account.

Step 5: The user clicks the confirmation link in the email, activating their account and allowing them to log in.

2. PRODUCT BROWSING

Step 1: The user accesses the homepage or a specific category page of the marketplace.

Step 2: The frontend sends a request to the Sanity CMS to fetch all relevant product data based on the selected category or search query.

Step 3: Sanity CMS responds with a list of products that match the criteria, including details such as product name, price, and stock level.

Step 4: The frontend displays the fetched products in a user-friendly format, allowing users to browse through the options.

Step 5: Users can click on individual products to view more detailed information, including images and product specifications.

3. ORDER PLACEMENT

Step 1: Users add desired products to their shopping cart by selecting quantities and clicking the 'Add to Cart' button.

Step 2: Once ready to checkout, users navigate to the cart page and click the 'Checkout' button.

Step 3: The frontend collects order details, including selected products and user information, and sends a request to the Sanity CMS to create a new order.

Step 4: Sanity CMS records the order and responds with an order confirmation, including a unique order ID and status.

Step 5: The user receives an email confirmation of the order, summarizing the purchased items and estimated delivery.

4. SHIPMENT TRACKING

Step 1: To track a shipment, the user enters their unique order ID in the designated tracking section of the marketplace.

Step 2: The frontend sends a request to the relevant third-party API responsible for shipment tracking.

Step 3: The third-party API processes the request and retrieves the current shipment status for the order ID provided.

Step 4: The API sends a response back to the frontend with the latest shipment details, including the current location and estimated delivery date.

Step 5: The frontend displays the shipment status to the user, allowing them to monitor their order's progress in real-time.

API DOCUMENTATION

The following table outlines the key API endpoints for the Marketplace Technical Foundation project, including their methods, purposes, and response examples.

Endpoint	Method	Purpose	Response Example
`/products`	GET	Fetch all products available in the marketplace	`{"id": 1, "name": "Product A", "price": 100}`
`/orders`	POST	Create a new order based on user selections	`{"orderid": 123, "status": "Success"}`
`/shipment`	GET	Fetch the shipment status for a specific order	`{"orderid": 123, "status": "In Transit"}`
`/user/signup`	POST	Register a new user with personal details	`{"message": "User registered successfully."}`
`/user/login`	POST	Authenticate a user using email and password	`{"token": "abc123xyz", "message": "Login successful."}`
`/user/logout`	POST	Log out the authenticated user	`{"message": "User logged out successfully."}`

ENDPOINT DETAILS

`/products`

- **Method:** GET

- **Purpose:** This endpoint retrieves a list of all products available in the marketplace. It allows users to browse through various items.
- **Response Example:**

`/orders`

- **Method:** POST
- **Purpose:** This endpoint is used to create a new order based on items selected by the user. It collects necessary order details and processes them.
- **Response Example:**

`/shipment`

- **Method:** GET
- **Purpose:** This endpoint fetches the current shipment status for a specific order, allowing users to track their purchased items.
- **Response Example:**

These endpoints are integral to the functioning of the Marketplace Technical Foundation, facilitating smooth interactions between users and the underlying system components.

PRODUCT SCHEMA CODE

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Product Name',
      description: 'The name of the product as it will
appear in the marketplace.',
      validation: Rule => Rule.required().min(1).max(100)
    },
    {
      name: 'price',
      type: 'number',
      title: 'Price',
      description: 'The price of the product in USD.',
```

```
      validation: Rule => Rule.required().min(0)
    },
    {
      name: 'stock',
      type: 'number',
      title: 'Stock Level',
      description: 'The current inventory level of the
product.',
      validation: Rule => Rule.required().min(0)
    },
    {
      name: 'image',
      type: 'image',
      title: 'Product Image',
      description: 'An image of the product to display in
the marketplace.',
      options: {
        hotspot: true
      }
    },
    {
      name: 'description',
      type: 'text',
      title: 'Product Description',
      description: 'A detailed description of the product
features and specifications.',
      validation: Rule => Rule.max(500)
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      description: 'The category to which the product
belongs (e.g., Electronics, Clothing).',
      validation: Rule => Rule.required()
    }
  ]
}
```

EXPLANATION OF FIELDS

Name:

- **Type**: String
- **Title**: Product Name
- **Description**: This field captures the name of the product as it will be displayed to users. It is mandatory and must be between 1 and 100 characters.

Price:

- **Type**: Number
- **Title**: Price
- **Description**: This field records the selling price of the product in USD. It is a required field and must be a non-negative number.

Stock:

- **Type**: Number
- **Title**: Stock Level
- **Description**: This field indicates the current amount of inventory available for the product. It is mandatory and should not be negative.

Image:

- **Type**: Image
- **Title**: Product Image
- **Description**: This field allows the upload of an image representing the product. The option to enable a "hotspot" feature helps in optimizing the image display.

Description:

- **Type**: Text
- **Title**: Product Description
- **Description**: This field is for a detailed description of the product, allowing up to 500 characters. It provides users with essential information about the product's features.

Category:

- **Type**: String
- **Title**: Category

- **Description**: This field classifies the product into a specific category, helping users to browse items based on their interests. It is a required field.

TECHNICAL ROADMAP

MILESTONES AND DETAILS

Setup Frontend

- **Details:** Initialize the Next.js project, install required libraries, and set up the project structure.
- **Deadline:** [Date]

Configure Sanity CMS

- **Details:** Set up the Sanity CMS environment, create necessary schemas for products, users, and orders, and ensure proper data validation rules are in place.
- **Deadline:** [Date]

API Integration

- **Details:** Develop and test the API endpoints for product retrieval, user authentication, order management, and shipment tracking. This includes connecting the frontend to the Sanity CMS and third-party APIs.
- **Deadline:** [Date]

Frontend Development

- **Details:** Build the user interface for all key workflows, including user registration, product browsing, order placement, and shipment tracking. Ensure responsive design and user experience are prioritized.
- **Deadline:** [Date]

Testing & Quality Assurance

- **Details:** Conduct thorough testing of the entire application, including unit tests, integration tests, and user acceptance testing. Address any bugs or issues discovered during testing.
- **Deadline:** [Date]

Deployment

- **Details:** Deploy the application to a production environment, ensuring all components are functioning correctly and securely. Monitor the deployment for any immediate issues.
- **Deadline:** [Date]

Post-Deployment Support

- **Details:** Provide ongoing support and maintenance for the application, including regular updates, performance monitoring, and user feedback collection.
- **Deadline:** Ongoing

TIMELINE OVERVIEW

Milestone	Deadline
Setup Frontend	[Date]
Configure Sanity CMS	[Date]
API Integration	[Date]
Frontend Development	[Date]
Testing & Quality Assurance	[Date]
Deployment	[Date]
Post-Deployment Support	Ongoing

1.USER DATA IN DATABASE

2.SANITY CMS INTEGRATION

3.USER INTERACTION FLOW

4.PRODUCT DATA FETCHING BROWSING CHECKOUT

5.PRODUCT DATA FETCHING BROWSING CHECKOUT

6.PRODUCT UPDATE SANITY (ADMIN PANEL)