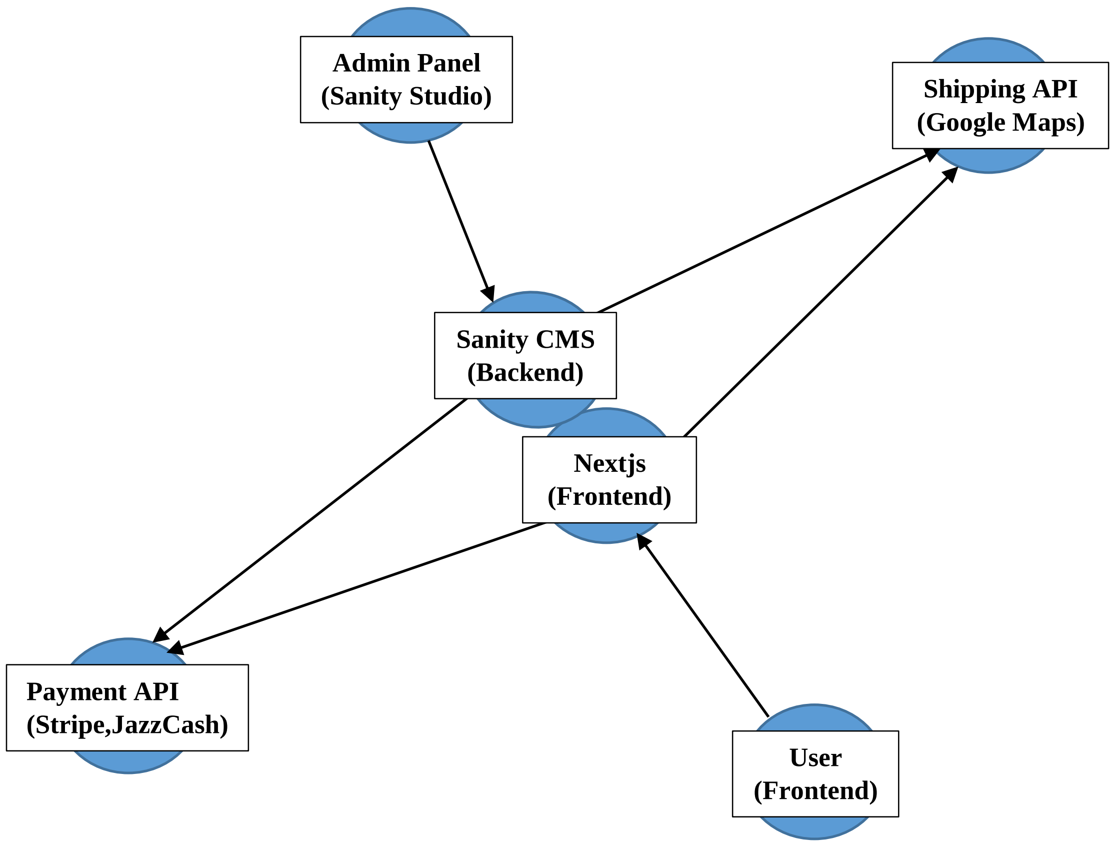
Marketplace Plan & Technical Foundation

Table of Contents:

1. Technical Plan

System Architecture Overview


Admin Panel (Sanity Studio)
Shipping API (Google Maps)
Sanity CMS (Backend)
Nextjs (Frontend)
Payment API (Stripe,JazzCash)
User (Frontend)

Components & Interactions

- **Frontend (Next.js & TailwindCSS):**
    - UI rendering and user interactions
    - Fetching data from Sanity CMS
    - Communicating with Sanity APIs
- **Backend (Sanity CMS as Headless CMS):**
    - Storing product listings, orders, and user data
    - Managing structured content with schemas
    - Handling authentication and authorization via NextAuth.js
- **Third-Party APIs:**
    - Payment Gateway (e.g., Stripe, Easy Paisa, Jazz Cash)
    - Shipping API for order tracking

2. Workflows

Seller Management Workflow

1. New seller registers via **NextAuth.js**.
2. Seller submits **Profile & Business Details**.
3. Admin verifies and **Approves / Rejects** seller.
4. Approved sellers can:
   - List new products.
   - Manage custom orders.
   - Update stock & pricing

User Registration Workflow

1. User visits the website and navigates to the registration page.
2. Enters details (Name, Email, Password, etc.).
3. System validates input and encrypts password.
4. Data is sent to Sanity CMS, which stores it as a document.
5. Email verification is sent.
6. User verifies the email and logs in.

Product Browsing Workflow

1. User visits the homepage.
2. Fetch request is sent to Sanity CMS.
3. Products are displayed with filtering and sorting options.
4. Clicking on a product shows detailed view fetched from Sanity.

Order Placement Workflow

1. User selects a product and clicks "Add to Cart."
2. Proceeds to checkout and fills in delivery details.
3. Payment details are entered and verified via Stripe/Jazz Cash.
4. On successful payment, order is stored in Sanity CMS.
5. Order tracking details are fetched from third-party APIs.

Customization Request Workflow

1. Customer selects a product & clicks "Request Customization".
2. Enters details (Size, Color, Material, Design Preferences).
3. Request is sent to Seller via **Sanity CMS API**.
4. Seller **Reviews** → **Accepts / Modifies / Rejects** request.
5. If accepted, **Sanity CMS updates price & availability**.
6. Customer confirms and proceeds to checkout.

Order Cancellation Workflow

1. Customer clicks "Cancel Order".
2. Sanity CMS checks **Order Status:**
   - If **Pending** → Cancel instantly.
   - If **Shipped** → Request Admin Approval.
3. Refund is processed via Stripe / Jazz Cash API if eligible.
4. Order status is updated in Sanity CMS.

Shipment Tracking Workflow

1. Order stored in Sanity CMS with tracking ID.
2. Third-party shipping API fetches live tracking data.

Failed Delivery Workflow

1. Shipping API detects **Failed Attempt**.
2. System sends reattempt request **OR** starts refund process.
3. If reattempt requested, new estimated delivery date is set.
4. If refund is initiated, customer is notified via email.

Return & Refund Workflow

1. Customer submits **Return Request**.
2. Seller **Reviews Request** → **Accepts / Rejects**.
3. If accepted:
   - Refund is initiated via Stripe API.
   - Order status updated in **Sanity CMS**.
4. If rejected, system provides reason to customer.

3. API Requirements

API Name	Endpoint	Method	Payload (Example)	Response (Example)
Seller Verification	/api/sellers/verify	POST	{ sellerId, documents }	{ approval Status }
User Registration	/api/auth/signup	POST	{name, email}	{ user ID, token }
Login	/api/auth/login	POST	{ email, password }	{ token, user Role }
Get Products	/api/products	GET	{category, price}	[{ id, name, price }]
Create Order	/api/orders	POST	{ user ID, items, payment Method }	{ order ID, status }
Customization Request	/api/custom-orders	POST	{ product ID, user ID, details }	{ request ID, status }
Order Cancellation	/api/orders/cancel	POST	{ orderId, reason }	{ success: true, refundStatus }
Chat Messaging	/api/chat/send	POST	{ sender ID, receiver ID, message }	{ message ID, timestamp }
Payment	/api/payments/webhook	POST	{stripe Payload}	{success: true}
Return Order	/api/orders/return	POST	{ orderId, reason }	{ returnId, approvalStatus }

3. Sanity Schema Design

Seller Schema

```
export default {
  name: 'seller',
  type: 'document',
  title: 'Seller',
  fields: [
    {name: 'name', type: 'string'},
    {name: 'email', type: 'string'},
    {name: 'storeName', type: 'string'},
    {name: 'verificationStatus', type: 'string', options: { list:
['Pending', 'Verified', 'Rejected'] } }
  ]
};
```

User Schema

```
export default {
  name: 'user',
  type: 'document',
  title: 'User',
  fields: [
    {name: 'name', type: 'string'},
    {name: 'email', type: 'string', unique: true},
    {name: 'role', type: 'string', options: {list: ['customer',
'seller', 'admin']}}
  ]
}
```

Product Schema

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    {name: 'name', type: 'string'},
    {name: 'price', type: 'number'},
    {name: 'image', type: 'image'},
    {name: 'category', type: 'reference', to: [{ type: 'category' }]},
    {name: 'stock', type: 'number'}
  ]
};
```

Order Schema

```
export default {
  name: 'order',
  type: 'document',
  title: 'Order',
  fields: [
    {name: 'userId', type: 'reference', to: [{ type: 'user' }] },
    {name: 'items', type: 'array', of: [{ type: 'reference', to: [{
type: 'product' }] }] },
    {name: 'customRequest', type: 'string' },
    {name: 'orderStatus', type: 'string', options: { list: ['Pending',
'Shipped', 'Delivered', 'Cancelled', 'Returned'] } },
    {name: 'paymentStatus', type: 'string', options: { list:
['Pending', 'Paid', 'Refunded'] } }
  ]
};
```

4. Collaboration Notes

Team Contributions

- **Frontend:** Implemented product display, filtering, and checkout flow.
- **CMS Management:** Created schemas for dynamic content management.
- **API Integration:** Implemented Stripe webhook for payments and real-time order updates.

Challenges & Resolutions

- **Sanity CMS syncing issues:** Resolved by using GROQ queries and real-time webhooks.
- **Payment integration complexities:** Used Stripe/Jazz Cash API with webhook validation.

Feedback Incorporation

- Improved UI based on user feedback for better navigation.
- Enhanced product detail page with more descriptive images and user reviews.