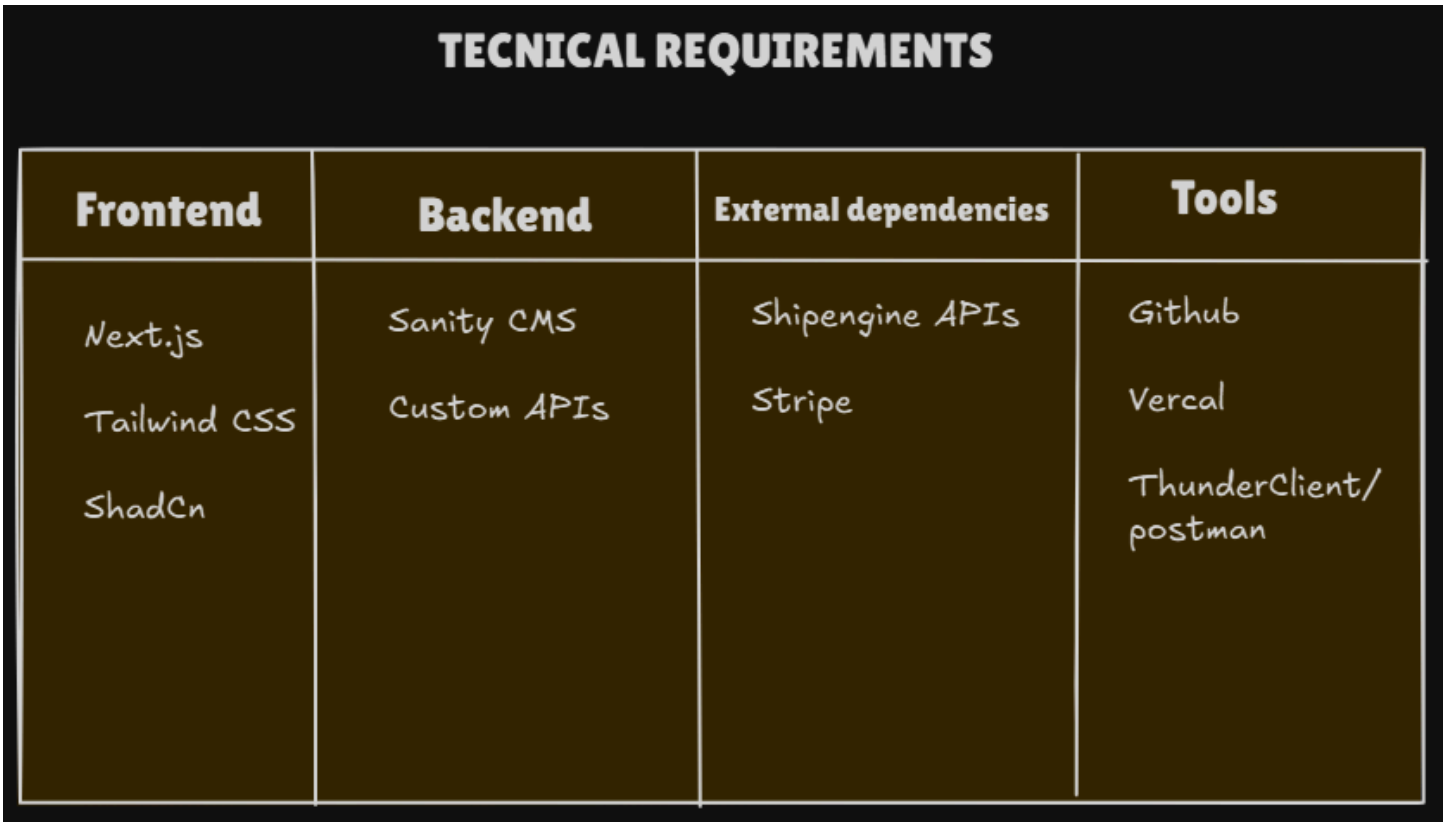
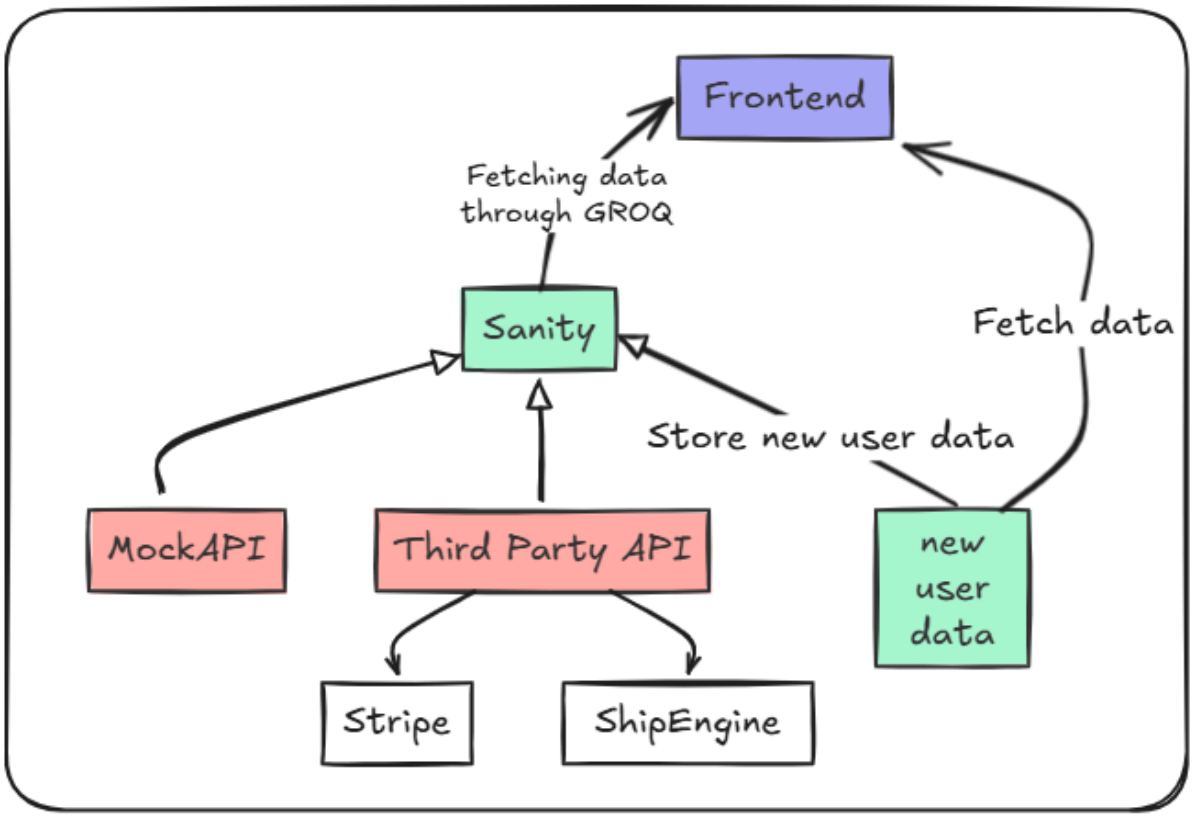
“Marketplace Technical Foundation-[Furniture niche-e-commerce store]”



SYSTEM ARCHITECTURE



System Architecture Overview

1. Frontend (Next.js)

- **Role**: User interface for browsing products, placing orders, and managing accounts.
- **Operations**: Fetches product, user, and order data using **GROQ queries** from Sanity to display on the UI.

2. Sanity (Headless CMS)

- **Role**: Centralized database for product, user, order, and third-party data.
- **Operations**: Stores product details, user profiles, orders, and integrates third-party data from **Stripe** (payments) and **ShipEngine** (shipping).

3. Backend (APIs)

- **Role**: Middleware connecting frontend, third-party services, and Sanity.
- **Operations**:
  - **Stripe**: Handles payment processing and transaction tracking.
  - **ShipEngine**: Manages shipping rates, labels, and tracking.

4. MockAPI

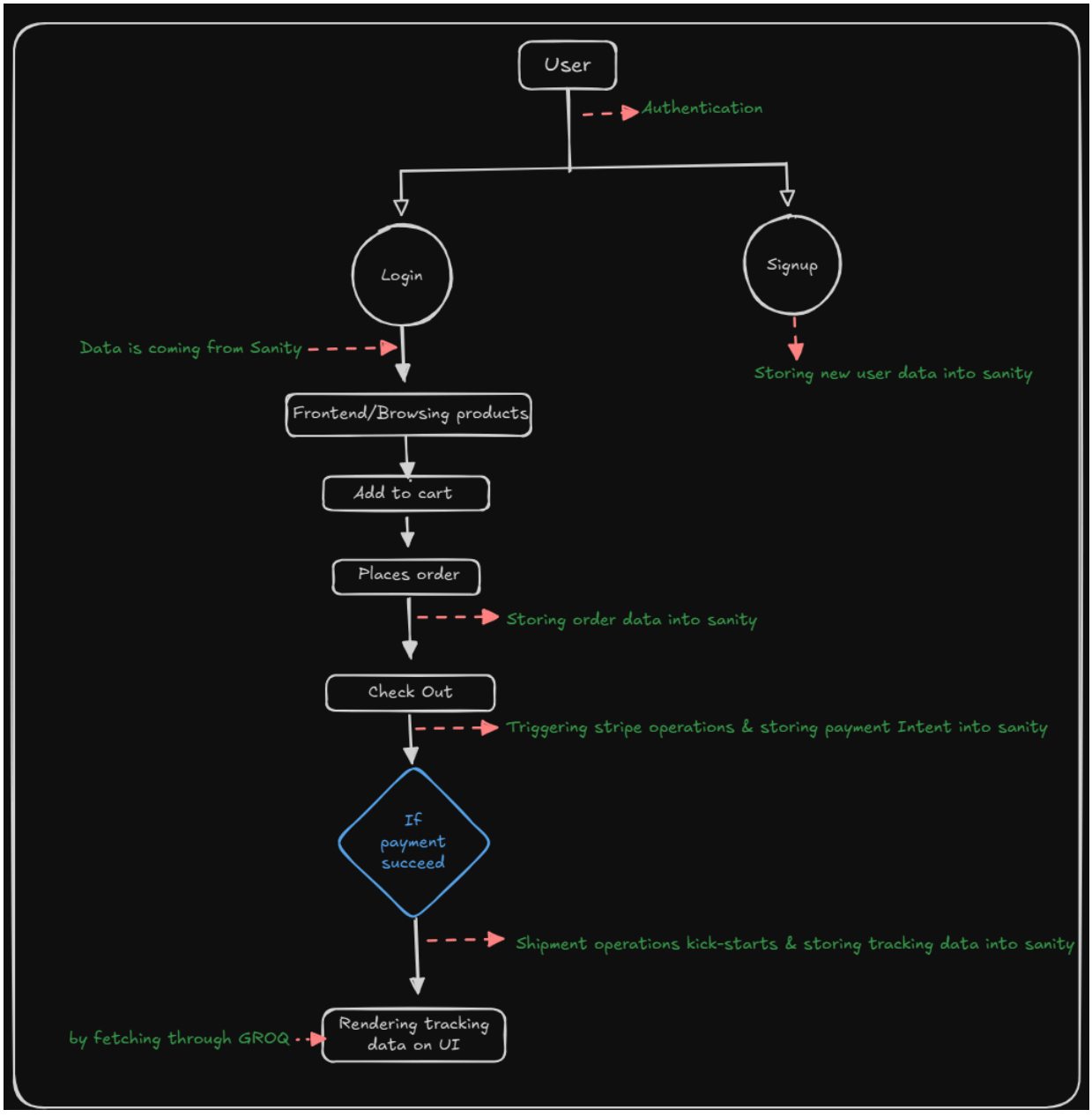
- **Role**: External data source for initial product data.
- **Operations**: Fetches product details (name, description, price) to sync with Sanity.

5. GROQ Queries

- **Role**: Query language for fetching and displaying data.
- **Operations**: Retrieves product, order, and user data from Sanity to dynamically update the frontend.

This architecture ensures scalability and flexibility by leveraging **Sanity** for content management, **Stripe** for payments, and **ShipEngine** for shipping, with the **Next.js** frontend providing a smooth user experience.

WORK FLOW



User Workflows

1. Add Product to Cart

- User browses and selects a product on the **Next.js frontend**.
- Frontend fetches product details via **GROQ queries** from **Sanity**.
- User adds the product to the cart, which updates in local storage.
- Cart is displayed with product details and total price.

2. Proceed to Checkout

- User clicks "Checkout" and enters **shipping information**.
- **ShipEngine** calculates shipping rates based on the location.
- **Stripe** processes payment and verifies success/failure.

3. Order Confirmation

- Successful payment triggers order data to be saved in **Sanity**.
- **ShipEngine** generates shipping labels and tracking information.
- Frontend displays order confirmation with tracking details.

4. View Order History

- User navigates to "Order History."
- Frontend fetches user orders from **Sanity**.
- **Sanity** returns order details, and the frontend displays the status of each order.

5. Cancel Order

- User selects an order to cancel.
- Frontend updates the order status in **Sanity** and requests a **Stripe** refund.
- User is notified of cancellation and refund status.

API REQUIREMENTS

Resource	Method/Endpoint	What It Does	Response
Products	GET /api/products	Fetch all products from the database.	200 OK with a list of products: [{id, title, price, description, category, images, stock}]
Products	GET /api/products/:id	Fetch a single product by its ID.	200 OK with the product details: {id, title, price, description, category, images, stock}
Products	POST /api/products	Create a new product in the database.	201 Created with product ID and success message.
Products	PUT /api/products/:id	Update an existing product by its ID.	200 OK with updated product details: {id, title, price, description, category, images, stock}
Products	DELETE /api/products/:id	Delete a product by its ID.	200 OK with success message: {message: 'Product deleted successfully'}
Orders	GET /api/orders	Fetch all orders from the database.	200 OK with a list of orders: [{id, orderNumber, userId, status, totalPrice, createdAt}]
Orders	GET /api/orders/:id	Fetch an order by its ID.	200 OK with order details: {id, orderNumber, userId, status, totalPrice, createdAt}
Orders	POST /api/orders	Create a new order in the database.	201 Created with order ID and success message.
Orders	PUT /api/orders/:id	Update an existing order by its ID.	200 OK with updated order details: {id, orderNumber, userId, status, totalPrice, createdAt}
Orders	DELETE /api/orders/:id	Delete an order by its ID.	200 OK with success message: {message: 'Order deleted successfully'}
Users	GET /api/users	Fetch all users from the database.	200 OK with a list of users: [{id, firstName, lastName, email, address, phone, isAdmin}]
Users	GET /api/users/:id	Fetch a user by their ID.	200 OK with user details: {id, firstName, lastName, email, address, phone, isAdmin}
Users	POST /api/users	Create a new user in the database.	201 Created with user ID and success message.
Users	PUT /api/users/:id	Update an existing user by their ID.	200 OK with updated user details: {id, firstName, lastName, email, address, phone, isAdmin}
Users	DELETE /api/users/:id	Delete a user by their ID.	200 OK with success message: {message: 'User deleted successfully'}
Payments	GET /api/payments	Fetch all payments from the database.	200 OK with a list of payments: [{id, orderId, paymentMethod, paymentStatus, transactionId}]
Payments	GET /api/payments/:id	Fetch a payment by its ID.	200 OK with payment details: {id, orderId, paymentMethod, paymentStatus, transactionId}
Payments	POST /api/payments	Create a new payment record in the database.	201 Created with payment ID and success message.
Payments	PUT /api/payments/:id	Update an existing payment by its ID.	200 OK with updated payment details: {id, orderId, paymentMethod, paymentStatus, transactionId}
Payments	DELETE /api/payments/:id	Delete a payment record by its ID.	200 OK with success message: {message: 'Payment deleted successfully'}
Shipments	GET /api/shipments	Fetch all shipments from the database.	200 OK with a list of shipments: [{id, orderId, shipmentMethod, shipmentStatus, trackingNumber}]
Shipments	GET /api/shipments/:id	Fetch a shipment by its ID.	200 OK with shipment details: {id, orderId, shipmentMethod, shipmentStatus, trackingNumber}
Shipments	POST /api/shipments	Create a new shipment record in the database.	201 Created with shipment ID and success message.
Shipments	PUT /api/shipments/:id	Update an existing shipment by its ID.	200 OK with updated shipment details: {id, orderId, shipmentMethod, shipmentStatus, trackingNumber}
Shipments	DELETE /api/shipments/:id	Delete a shipment record by its ID.	200 OK with success message: {message: 'Shipment deleted successfully'}

SANITY SCHEMAS

Schema for product

```
// product
export const productSchema = {
    name: 'product', title: 'Product', type: 'document', fields: [
      { name: 'title', title: 'Title', type: 'string', validation: Rule => Rule.required().min(5).max(100) },
      { name: 'slug', title: 'Slug', type: 'slug', description: 'Unique URL-friendly identifier for the product', options: { source:
      'title', maxLength: 200 }, validation: Rule => Rule.required() },
      { name: 'description', title: 'Description', type: 'text', validation: Rule => Rule.required().min(20).max(1000) },
      { name: 'price', title: 'Price', type: 'number', validation: Rule => Rule.required().min(0) },
      { name: 'category', title: 'Category', type: 'string', options: { list: [{ title: 'Sofa', value: 'sofa' }, { title: 'Chair', value:
      'chair' }, { title: 'Table', value: 'table' }, { title: 'Bed', value: 'bed' }, { title: 'Storage', value: 'storage' }] }, validation:
      Rule => Rule.required() },
      { name: 'images', title: 'Images', type: 'array', of: [{ type: 'image', options: { hotspot: true } }], validation: Rule => Rule.
      required().min(1) },
      { name: 'material', title: 'Material', type: 'string', options: { list: [{ title: 'Wood', value: 'wood' }, { title: 'Metal', value:
      'metal' }, { title: 'Fabric', value: 'fabric' }, { title: 'Leather', value: 'leather' }] }, validation: Rule => Rule.required() },
      { name: 'dimensions', title: 'Dimensions', type: 'object', fields: [
        { name: 'length', title: 'Length (cm)', type: 'number', validation: Rule => Rule.required().min(1) },
        { name: 'width', title: 'Width (cm)', type: 'number', validation: Rule => Rule.required().min(1) },
        { name: 'height', title: 'Height (cm)', type: 'number', validation: Rule => Rule.required().min(1) }
      ]},
      { name: 'weight', title: 'Weight (kg)', type: 'number', validation: Rule => Rule.required().min(0) },
      { name: 'available', title: 'Available', type: 'boolean', description: 'Indicates whether the product is in stock or not', validation:
      Rule => Rule.required() },
      { name: 'featured', title: 'Featured', type: 'boolean', description: 'Indicates whether the product is featured on the homepage or
      not', validation: Rule => Rule.optional() },
      { name: 'externalId', title: 'External ID', type: 'string', description: 'ID used in third-party systems (e.g., ShipEngine or other
      integrations)', validation: Rule => Rule.optional() },
      { name: 'sizes', title: 'Sizes', type: 'array', of: [{ type: 'string' }], description: 'Available sizes for the product (e.g., Small,
      Medium, Large)', validation: Rule => Rule.optional() },
      { name: 'colors', title: 'Colors', type: 'array', of: [{ type: 'string', options: { list: [{ title: 'Red', value: 'red' }, { title:
      'Blue', value: 'blue' }, { title: 'Green', value: 'green' }, { title: 'Black', value: 'black' }, { title: 'White', value: 'white' }, {
      title: 'Gray', value: 'gray' }, { title: 'Beige', value: 'beige' }, { title: 'Brown', value: 'brown' }] } }], description: 'Available
      colors for the product (select from palette)', validation: Rule => Rule.optional() }
    ],
    preview: {
      select: { title: 'title', media: 'images.0' },
      prepare(selection) {
        const { title, media } = selection
        return { title, media: media || 'https://via.placeholder.com/150' }
```


Schema for Order

```
// order
export const orderSchema = {
  name: 'order', title: 'Order', type: 'document', fields: [
    { name: 'orderNumber', title: 'Order Number', type: 'string', description: 'Unique order number for the transaction', validation: Rule
    => Rule.required() },
    { name: 'user', title: 'User', type: 'reference', to: [{ type: 'user' }], description: 'Reference to the user who placed the order',
    validation: Rule => Rule.required() },
    { name: 'status', title: 'Order Status', type: 'string', options: { list: [{ title: 'Pending', value: 'pending' }, { title:
    'Processing', value: 'processing' }, { title: 'Shipped', value: 'shipped' }, { title: 'Delivered', value: 'delivered' }, { title:
    'Cancelled', value: 'cancelled' }] }, description: 'Current status of the order', validation: Rule => Rule.required() },
    { name: 'totalAmount', title: 'Total Amount', type: 'number', description: 'Total amount of the order including shipping and taxes',
    validation: Rule => Rule.required().min(0) },
    { name: 'shippingAddress', title: 'Shipping Address', type: 'object', fields: [
      { name: 'street', title: 'Street Address', type: 'string', validation: Rule => Rule.required() },
      { name: 'city', title: 'City', type: 'string', validation: Rule => Rule.required() },
      { name: 'state', title: 'State/Province', type: 'string', validation: Rule => Rule.required() },
      { name: 'zipCode', title: 'Zip Code', type: 'string', validation: Rule => Rule.required() },
      { name: 'country', title: 'Country', type: 'string', validation: Rule => Rule.required() }
    ], description: 'Shipping address for the order', validation: Rule => Rule.required() },
    { name: 'items', title: 'Items', type: 'array', of: [{ type: 'object', fields: [
      { name: 'product', title: 'Product', type: 'reference', to: [{ type: 'product' }], validation: Rule => Rule.required() },
      { name: 'quantity', title: 'Quantity', type: 'number', validation: Rule => Rule.required().min(1) },
      { name: 'price', title: 'Price', type: 'number', validation: Rule => Rule.required().min(0) }
    ] }], description: 'List of products in the order with quantity and price', validation: Rule => Rule.required().min(1) },
    { name: 'paymentStatus', title: 'Payment Status', type: 'string', options: { list: [{ title: 'Pending', value: 'pending' }, { title:
    'Paid', value: 'paid' }, { title: 'Failed', value: 'failed' }] }, description: 'Payment status of the order', validation: Rule => Rule.
    required() },
    { name: 'paymentMethod', title: 'Payment Method', type: 'string', options: { list: [{ title: 'Credit Card', value: 'credit_card' }, {
    title: 'PayPal', value: 'paypal' }, { title: 'Stripe', value: 'stripe' }, { title: 'Cash on Delivery', value: 'cod' }] }, description:
    'Method of payment used for the order', validation: Rule => Rule.required() },
    { name: 'trackingData', title: 'Tracking Data', type: 'reference', to: [{ type: 'trackingdata' }], description: 'Reference to the
    tracking data of the order', validation: Rule => Rule.optional() },
    { name: 'createdAt', title: 'Created At', type: 'datetime', description: 'Timestamp when the order was placed', validation: Rule =>
    Rule.required() },
    { name: 'updatedAt', title: 'Updated At', type: 'datetime', description: 'Timestamp when the order was last updated', validation: Rule
    => Rule.required() }
  ],
  preview: {
    select: { title: 'orderNumber', subtitle: 'status', media: 'user.image' }
```


Schema for user

```
// user
export const userSchema = {
  name: 'user', title: 'User', type: 'document', fields: [
    { name: 'name', title: 'Name', type: 'string', description: 'Full name of the user', validation: Rule => Rule.required().min(3).max
    (100) },
    { name: 'email', title: 'Email', type: 'string', description: 'User email address', validation: Rule => Rule.required().email() },
    { name: 'phone', title: 'Phone', type: 'string', description: 'User phone number', validation: Rule => Rule.optional().min(10).max
    (15) },
    { name: 'address', title: 'Address', type: 'object', fields: [
      { name: 'street', title: 'Street Address', type: 'string', validation: Rule => Rule.optional() },
      { name: 'city', title: 'City', type: 'string', validation: Rule => Rule.optional() },
      { name: 'state', title: 'State/Province', type: 'string', validation: Rule => Rule.optional() },
      { name: 'zipCode', title: 'Zip Code', type: 'string', validation: Rule => Rule.optional() },
      { name: 'country', title: 'Country', type: 'string', validation: Rule => Rule.optional() }
    ]},
    { name: 'createdAt', title: 'Created At', type: 'datetime', description: 'Timestamp when the user was created', validation: Rule =>
    Rule.required() },
    { name: 'updatedAt', title: 'Updated At', type: 'datetime', description: 'Timestamp when the user details were last updated',
    validation: Rule => Rule.required() },
    { name: 'orders', title: 'Orders', type: 'array', of: [{ type: 'reference', to: [{ type: 'order' }] }], description: 'List of orders
    made by the user', validation: Rule => Rule.optional() },
    { name: 'profilePicture', title: 'Profile Picture', type: 'image', description: 'User profile image', options: { hotspot: true },
    validation: Rule => Rule.optional() },
    { name: 'role', title: 'Role', type: 'string', options: { list: [{ title: 'Admin', value: 'admin' }, { title: 'Customer', value:
    'customer' }, { title: 'Guest', value: 'guest' }] }, description: 'User role within the platform', validation: Rule => Rule.required
    () }
  ],
  preview: {
    select: { title: 'name', media: 'profilePicture' },
    prepare(selection) {
      const { title, media } = selection
      return { title, media: media || 'https://via.placeholder.com/150' }
    }
  }
}
```

Schema for payment

```
  //payment
export const paymentSchema = {
    name: 'payment', title: 'Payment', type: 'document', fields: [
      { name: 'order', title: 'Order', type: 'reference', to: [{ type: 'order' }], description: 'Reference to the related order',
      validation: Rule => Rule.required() },
      { name: 'user', title: 'User', type: 'reference', to: [{ type: 'user' }], description: 'Reference to the user who made the payment',
      validation: Rule => Rule.required() },
      { name: 'paymentIntentId', title: 'Payment Intent ID', type: 'string', description: 'ID generated by Stripe for the payment process',
      validation: Rule => Rule.required() },
      { name: 'status', title: 'Payment Status', type: 'string', options: { list: [{ title: 'Pending', value: 'pending' }, { title:
      'Succeeded', value: 'succeeded' }, { title: 'Failed', value: 'failed' }, { title: 'Canceled', value: 'canceled' }] }, description:
      'Current status of the payment', validation: Rule => Rule.required() },
      { name: 'amount', title: 'Amount', type: 'number', description: 'The total amount paid by the user', validation: Rule => Rule.required
      ().min(1) },
      { name: 'currency', title: 'Currency', type: 'string', description: 'Currency in which the payment was made', validation: Rule => Rule.
      required().min(3).max(3) },
      { name: 'paymentMethod', title: 'Payment Method', type: 'string', options: { list: [{ title: 'Credit Card', value: 'credit_card' }, {
      title: 'PayPal', value: 'paypal' }, { title: 'Stripe', value: 'stripe' }] }, description: 'Method used for the payment', validation:
      Rule => Rule.required() },
      { name: 'paymentDate', title: 'Payment Date', type: 'datetime', description: 'Timestamp when the payment was made', validation: Rule
      => Rule.required() },
      { name: 'transactionId', title: 'Transaction ID', type: 'string', description: 'The transaction ID returned from the payment gateway',
      validation: Rule => Rule.optional() },
      { name: 'paymentDetails', title: 'Payment Details', type: 'text', description: 'Additional details or notes about the payment
      (optional)', validation: Rule => Rule.optional() }
    ],
    preview: {
      select: { title: 'paymentIntentId', subtitle: 'status', media: 'user.profilePicture' },
      prepare(selection) {
        const { title, subtitle, media } = selection
        return { title, subtitle, media: media || 'https://via.placeholder.com/150' }
      }
    }
  }
```


Schema for shipment

```
  // ShipmenttrackingData
export const trackingDataSchema = {
    name: 'shipmentData', title: 'Shipment Data', type: 'document', fields: [
      { name: 'order', title: 'Order', type: 'reference', to: [{ type: 'order' }], description: 'Reference to the order associated with this
      tracking data', validation: Rule => Rule.required() },
      { name: 'trackingNumber', title: 'Tracking Number', type: 'string', description: 'The tracking number provided by the carrier',
      validation: Rule => Rule.required() },
      { name: 'carrier', title: 'Carrier', type: 'string', options: { list: [{ title: 'UPS', value: 'ups' }, { title: 'FedEx', value:
      'fedex' }, { title: 'DHL', value: 'dhl' }, { title: 'USPS', value: 'usps' }] }, description: 'The shipping carrier', validation: Rule
      => Rule.required() },
      { name: 'status', title: 'Status', type: 'string', options: { list: [{ title: 'In Transit', value: 'in_transit' }, { title:
      'Delivered', value: 'delivered' }, { title: 'Out for Delivery', value: 'out_for_delivery' }, { title: 'Pending', value:
      'pending' }] }, description: 'The current status of the shipment', validation: Rule => Rule.required() },
      { name: 'estimatedDeliveryDate', title: 'Estimated Delivery Date', type: 'datetime', description: 'The estimated date the order will
      be delivered', validation: Rule => Rule.optional() },
      { name: 'currentLocation', title: 'Current Location', type: 'string', description: 'Current location of the package during transit (e.
      g., City, State)', validation: Rule => Rule.optional() },
      { name: 'updates', title: 'Tracking Updates', type: 'array', of: [{ type: 'object', fields: [
        { name: 'timestamp', title: 'Timestamp', type: 'datetime', description: 'The time of the update' },
        { name: 'status', title: 'Status', type: 'string', description: 'Current status at the time of the update' },
        { name: 'location', title: 'Location', type: 'string', description: 'Location at the time of the update' },
        { name: 'description', title: 'Description', type: 'text', description: 'A detailed description of the tracking update' }
      ]}],
      description: 'Array of updates for the shipment throughout its journey', validation: Rule => Rule.optional() },
      { name: 'createdAt', title: 'Created At', type: 'datetime', description: 'Timestamp when the tracking data was created', validation:
      Rule => Rule.required() },
      { name: 'updatedAt', title: 'Updated At', type: 'datetime', description: 'Timestamp when the tracking data was last updated',
      validation: Rule => Rule.required() }
    ],
    preview: {
      select: { title: 'trackingNumber', subtitle: 'status', media: 'order.images.0' },
      prepare(selection) {
        const { title, subtitle, media } = selection
        return { title, subtitle, media: media || 'https://via.placeholder.com/150' }
      }
    }
  }
```