Hackathon 3 Document Day 2

General E-commerce Marketplace Technical Foundation

Prepared Date: January 16, 2025 by: Azmat Ali

1. Technical Plan Aligned with Business Goals

Business Objectives

- Create a scalable e-commerce platform
- Provide seamless shopping experience
- Enable efficient product management
- Ensure secure payment processing
- Implement order tracking and management

Technical Requirements

Frontend Requirements

- Next.js-based responsive web application
- Server-side rendering for improved SEO
- Client-side state management using React Context/Redux
- Progressive Web App (PWA) capabilities
- Responsive design breakpoints: Mobile (< 768px), Tablet (768px - 1024px), Desktop (> 1024px)

Backend Requirements (Sanity CMS)

- Content management for products, categories, and orders
- Real-time inventory tracking
- Order management system
- Customer data management
- Role-based access control

Third-party Integrations

- Payment Gateway (Stripe)
- Email Service (SendGrid)
- Image CDN (Cloudinary)
- Analytics (Google Analytics)
- Search Implementation (Algolia)

E-Commerce Work Process By Image

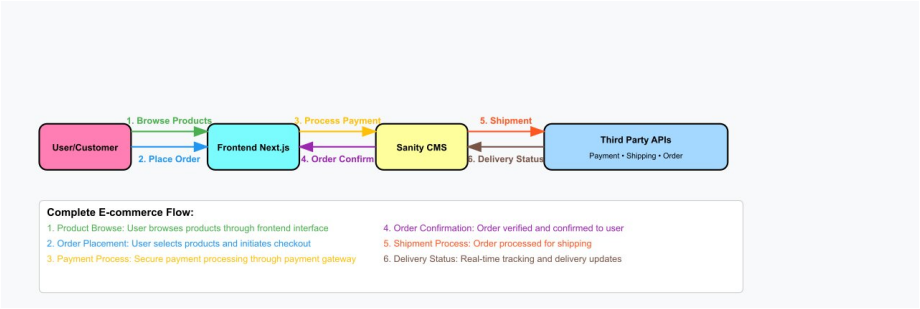

Figure 1: Image

2. System Architecture

High-Level Architecture Diagram

```
graph TD
    A[Frontend - Next.js] --> B[API Layer]
    B --> C[Sanity CMS]
    B --> D[Third-party Services]
    D --> E[Payment Gateway]
    D --> F[Email Service]
    D --> G[Image CDN]
    D --> H[Analytics]
    C --> I[Content Management]
    I --> J[Products]
    I --> K[Orders]
    I --> L[Customers]
```

Core Workflows

Product Browse & Purchase Flow

```
sequenceDiagram
    participant U as User
    participant F as Frontend
    participant S as Sanity CMS
    participant P as Payment Gateway

    U->>F: Browse Products
    F->>S: Fetch Products
    S-->>F: Product Data
    U->>F: Add to Cart
    U->>F: Checkout
    F->>P: Process Payment
    P-->>F: Payment Confirmation
```

```
F->>S: Create Order
S-->>F: Order Confirmation
F->>U: Success Message
```

3. API Requirements

Product Management APIs

Endpoint	Method	Description	Request/Response
`/api/products`	GET	Fetch all products	Response: `{ products: [{id, name, price, stock}] }`
`/api/products/:id`	GET	Get single product	Response: `{ id, name, price, description, stock }`
`/api/categories`	GET	Fetch categories	Response: `{ categories: [{id, name, products}] }`
`/api/orders`	POST	Create order	Request: `{ items, customer, payment }`
`/api/orders/:id`	GET	Get order status	Response: `{ id, status, items, tracking }`

4. Sanity Schemas

Product Schema

```javascript
export default {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'name',
      title: 'Name',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'slug',
      title: 'Slug',
      type: 'slug',
      options: {
        source: 'name'
      }
    },
    {
      name: 'price',
```

```
      title: 'Price',
      type: 'number',
      validation: Rule => Rule.required().positive()
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text'
    },
    {
      name: 'images',
      title: 'Images',
      type: 'array',
      of: [{ type: 'image' }]
    },
    {
      name: 'category',
      title: 'Category',
      type: 'reference',
      to: [{ type: 'category' }]
    },
    {
      name: 'stock',
      title: 'Stock',
      type: 'number',
      validation: Rule => Rule.required().min(0)
    }
  ]
}
```

Order Schema

```
export default {
  name: 'order',
  title: 'Order',
  type: 'document',
  fields: [
    {
      name: 'orderNumber',
      title: 'Order Number',
      type: 'string',
      validation: Rule => Rule.required()
    },
    {
      name: 'customer',
      title: 'Customer',
```

```
    type: 'reference',
    to: [{ type: 'customer' }]
  },
  {
    name: 'items',
    title: 'Items',
    type: 'array',
    of: [{
      type: 'object',
      fields: [
        {
          name: 'product',
          type: 'reference',
          to: [{ type: 'product' }]
        },
        {
          name: 'quantity',
          type: 'number'
        }
      ]
    }]
  },
  {
    name: 'status',
    title: 'Status',
    type: 'string',
    options: {
      list: [
        'pending',
        'processing',
        'shipped',
        'delivered',
        'cancelled'
      ]
    }
  },
  {
    name: 'totalAmount',
    title: 'Total Amount',
    type: 'number'
  },
  {
    name: 'paymentStatus',
    title: 'Payment Status',
    type: 'string',
    options: {
```

```
        list: [
          'pending',
          'completed',
          'failed',
          'refunded'
        ]
      }
    }
  ]
}
```

5. Implementation Roadmap

Phase 1: Setup & Basic Structure

- Initialize Next.js project
- Set up Sanity CMS
- Implement basic routing
- Create core components

Phase 2: Core Features

- Product listing and details
- Shopping cart functionality
- User authentication
- Basic checkout process

Phase 3: Integration & Testing

- Payment gateway integration
- Order management
- Email notifications
- Testing and bug fixes

6. Best Practices Implementation

Performance Optimization

- Image optimization using next/image
- Lazy loading for product listings
- API response caching
- Code splitting for route-based chunking

Security Measures

- Input validation
- XSS protection
- CSRF tokens

- Secure payment handling

SEO Considerations

- Meta tags management
- Structured data implementation
- Sitemap generation
- robots.txt configuration

7. Quality Assurance Checklist

☐ Mobile responsiveness
☐ Cross-browser compatibility
☐ Performance metrics
☐ Security compliance
☐ API error handling
☐ Loading states
☐ Form validation
☐ Payment flow testing

8. Monitoring & Analytics

- Implement error tracking (Sentry)
- Set up performance monitoring
- Track user behavior
- Monitor API performance
- Track conversion rates

This documentation serves as a comprehensive guide for implementing the e-commerce marketplace, focusing on maintainability, scalability, and user experience.

E-commerce Data Schema Relationships

```
erDiagram
    PRODUCT ||--o{ ORDER_ITEM : contains
    ORDER ||--|{ ORDER_ITEM : includes
    CUSTOMER ||--o{ ORDER : places
    PRODUCT ||--o{ PRODUCT_CATEGORY : belongs_to
    CATEGORY ||--|{ PRODUCT_CATEGORY : has
    ADDRESS ||--o{ CUSTOMER : has
    CUSTOMER ||--o{ REVIEW : writes
    PRODUCT ||--o{ REVIEW : receives
    ORDER ||--o{ ORDER_STATUS : tracks

    PRODUCT {
```

```
    string id PK
    string name
    float price
    int stock
    string description
    string[] images
    datetime created_at
    datetime updated_at
    boolean is_active
}

CATEGORY {
    string id PK
    string name
    string description
    string slug
    string parent_id FK
}

PRODUCT_CATEGORY {
    string product_id FK
    string category_id FK
}

CUSTOMER {
    string id PK
    string first_name
    string last_name
    string email
    string phone
    datetime created_at
    boolean is_active
}

ADDRESS {
    string id PK
    string customer_id FK
    string street
    string city
    string state
    string postal_code
    string country
    boolean is_default
    string type
}
```

```
ORDER {
    string id PK
    string customer_id FK
    float total_amount
    string payment_status
    datetime order_date
    string shipping_address_id FK
    string billing_address_id FK
}

ORDER_ITEM {
    string id PK
    string order_id FK
    string product_id FK
    int quantity
    float unit_price
    float subtotal
}

ORDER_STATUS {
    string id PK
    string order_id FK
    string status
    string description
    datetime created_at
}

REVIEW {
    string id PK
    string product_id FK
    string customer_id FK
    int rating
    string comment
    datetime created_at
}
```

** Day 1 **

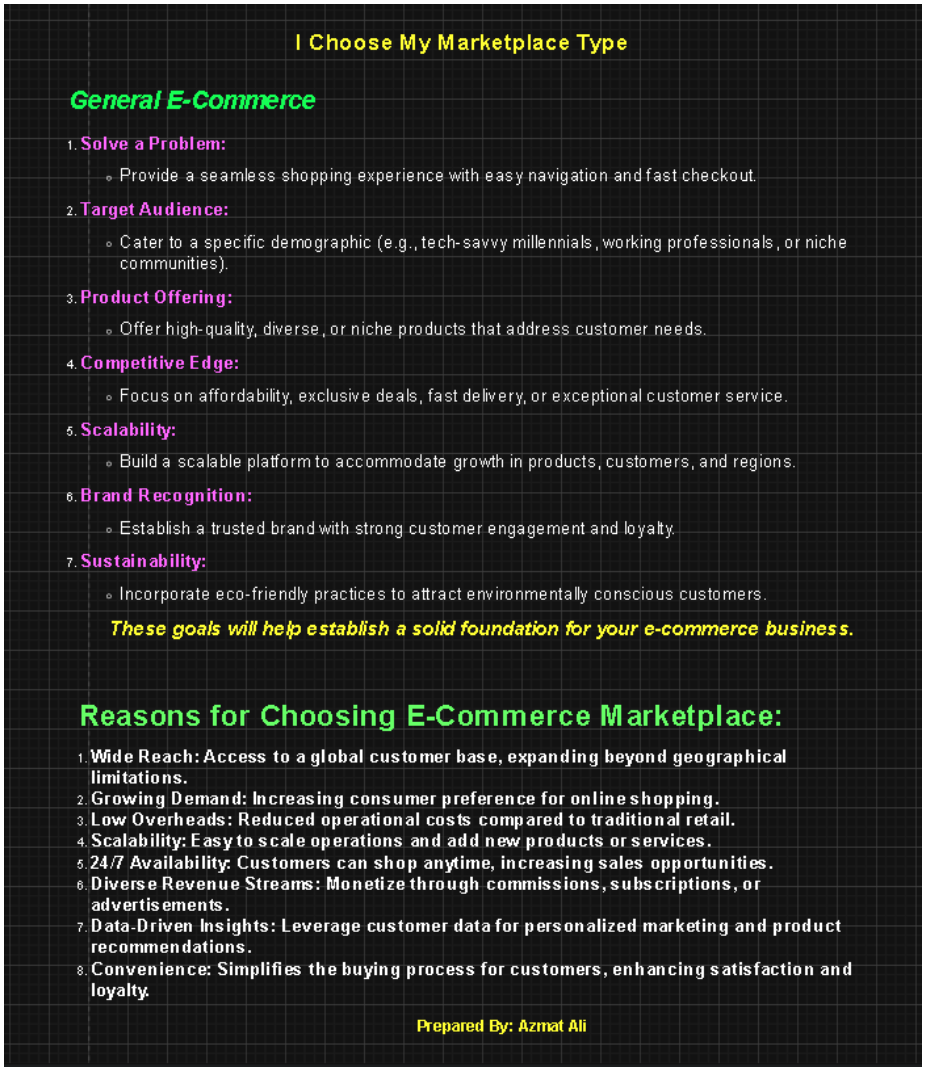

Figure 2: General E-Commerce For Me

Figure 3: Image

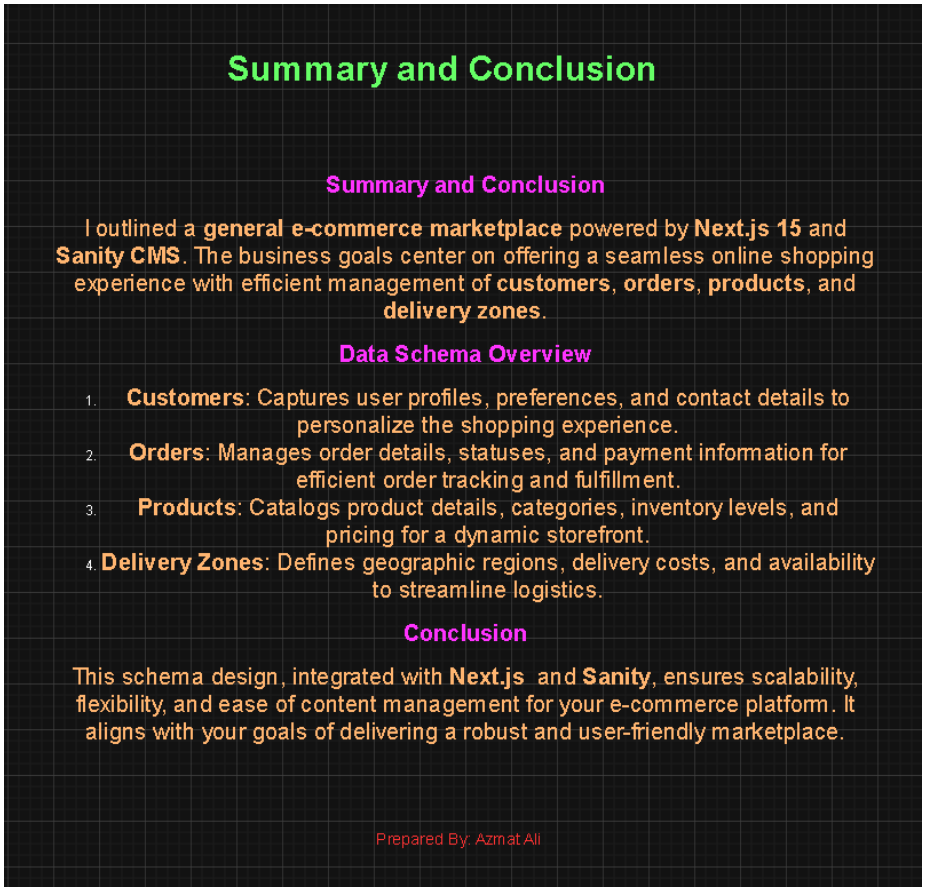

Figure 4: Image

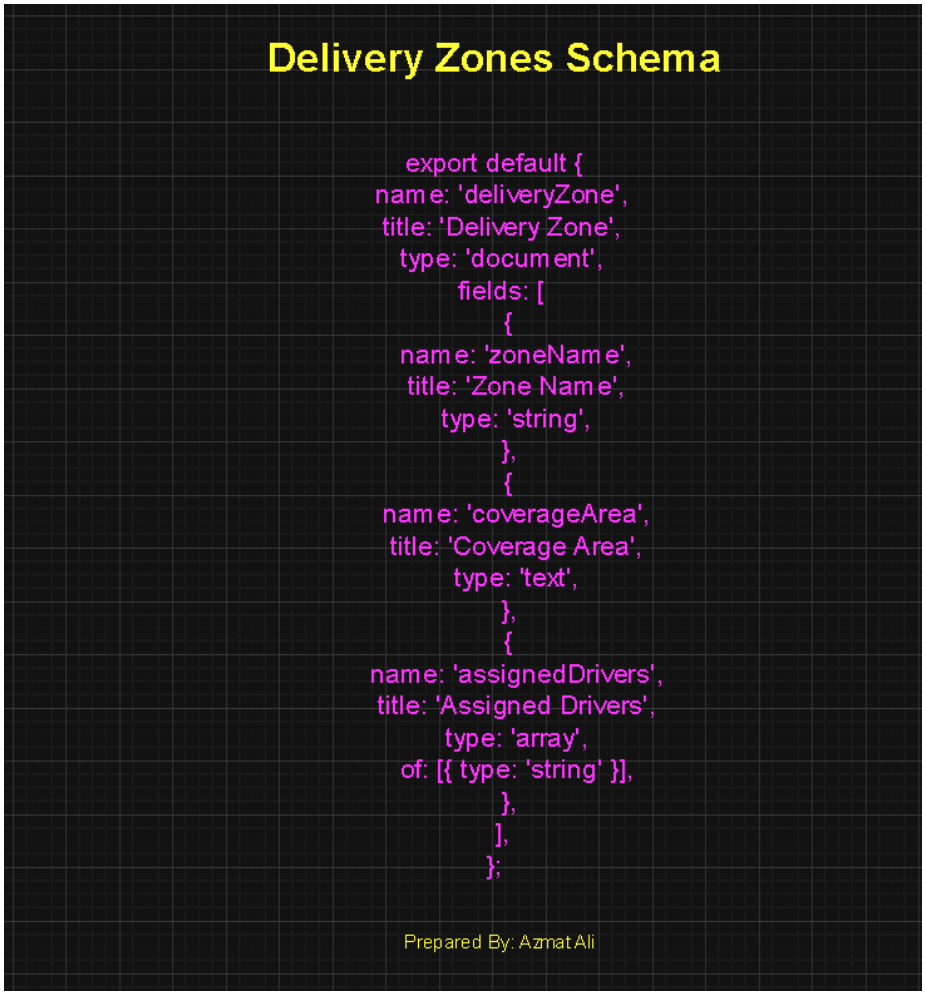

Figure 5: Image

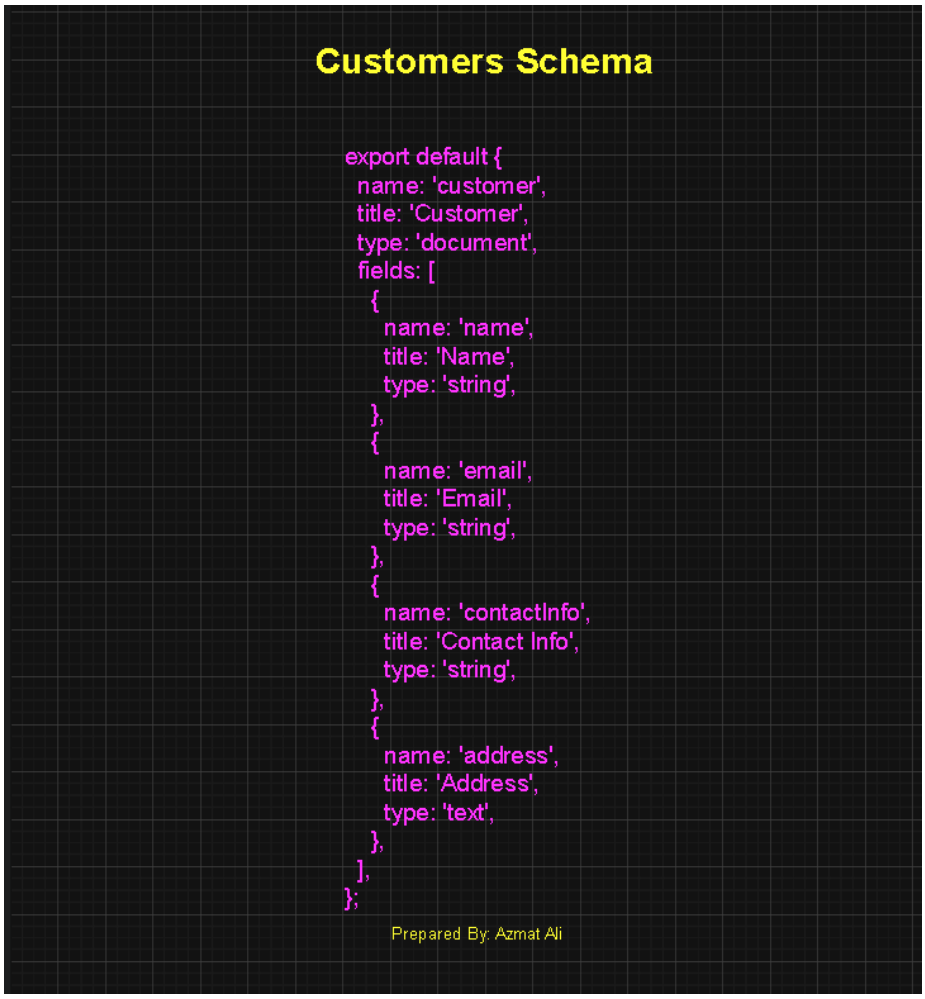

Figure 6: Image

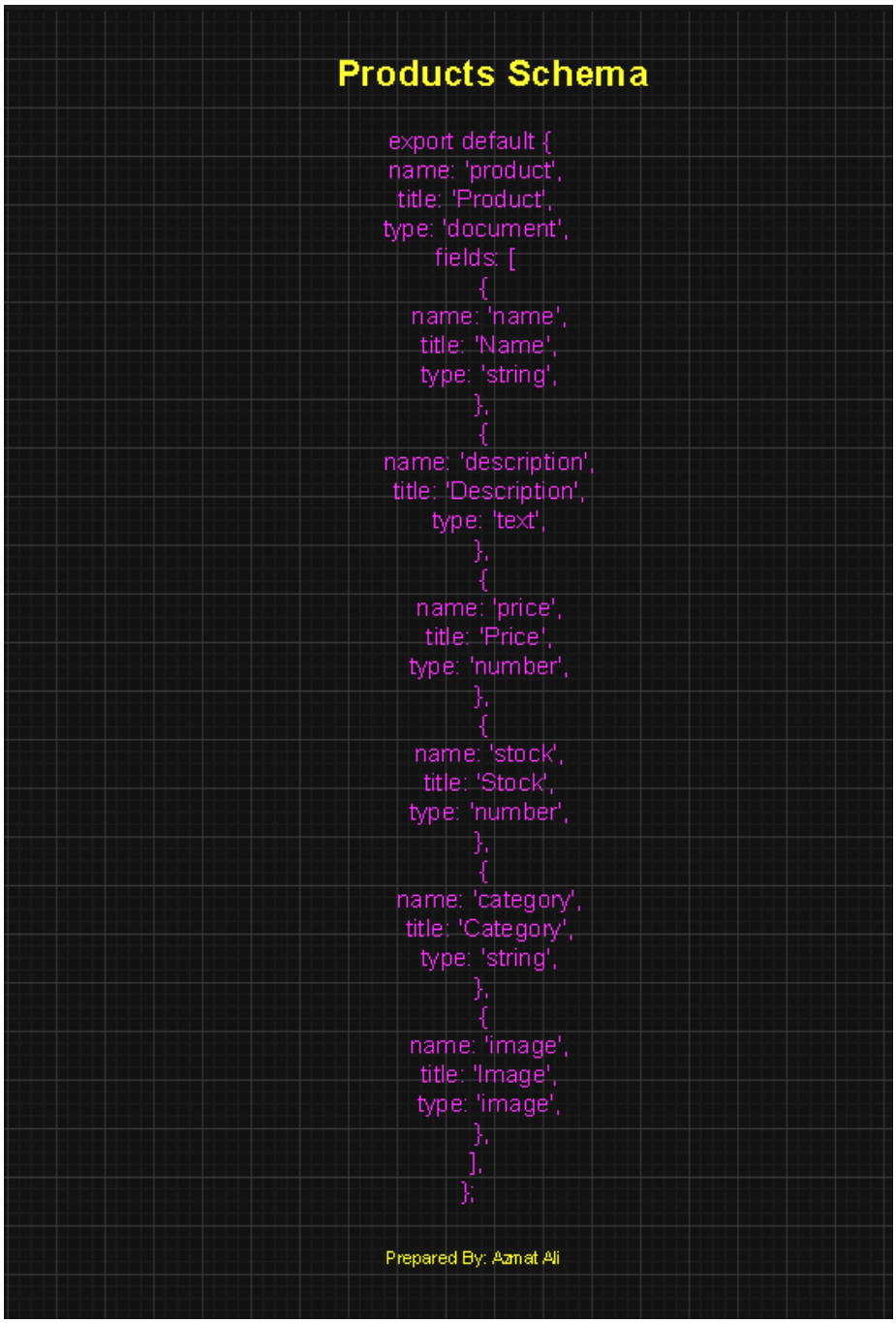

Figure 7: Image

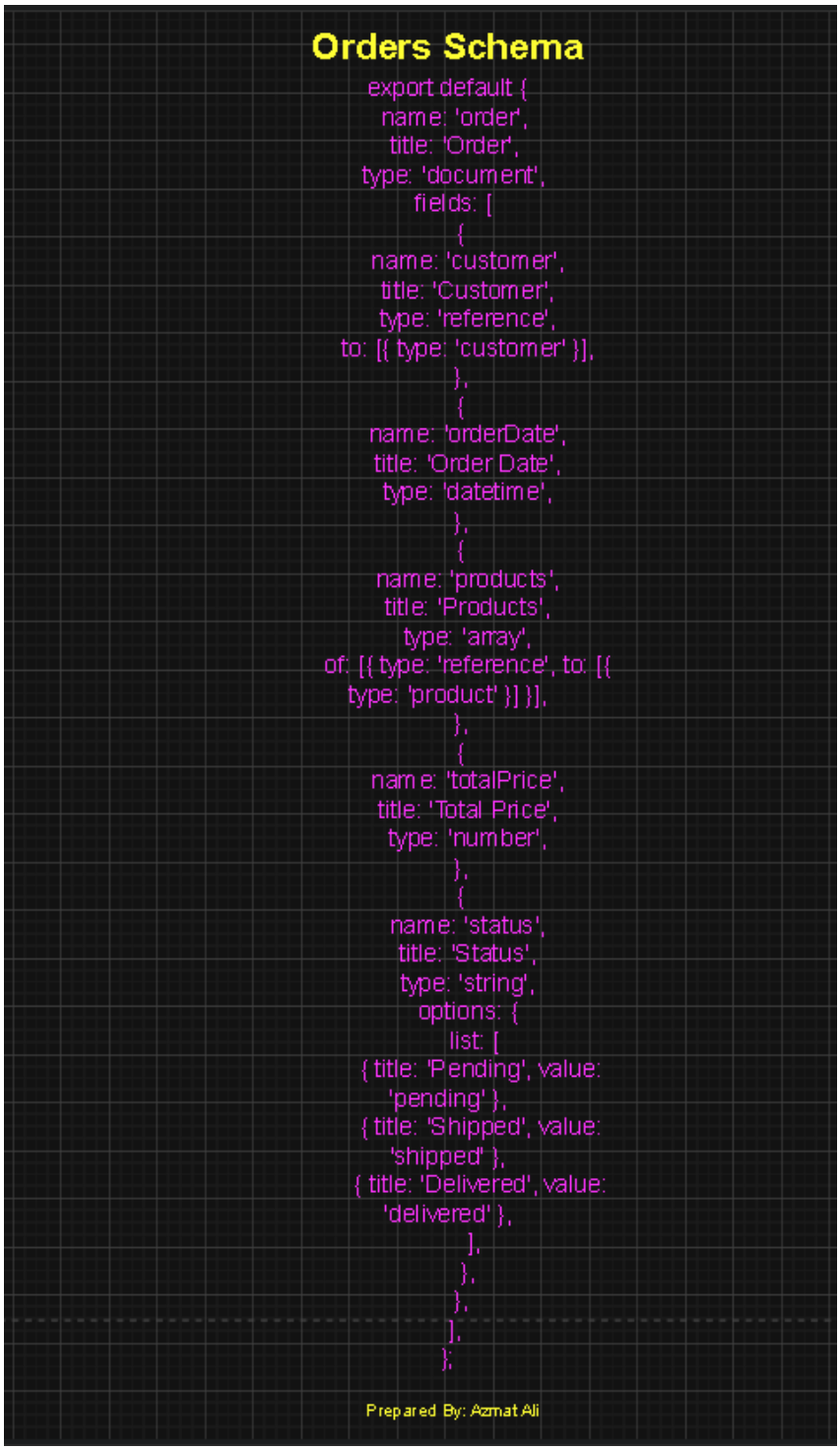

Figure 8: Image

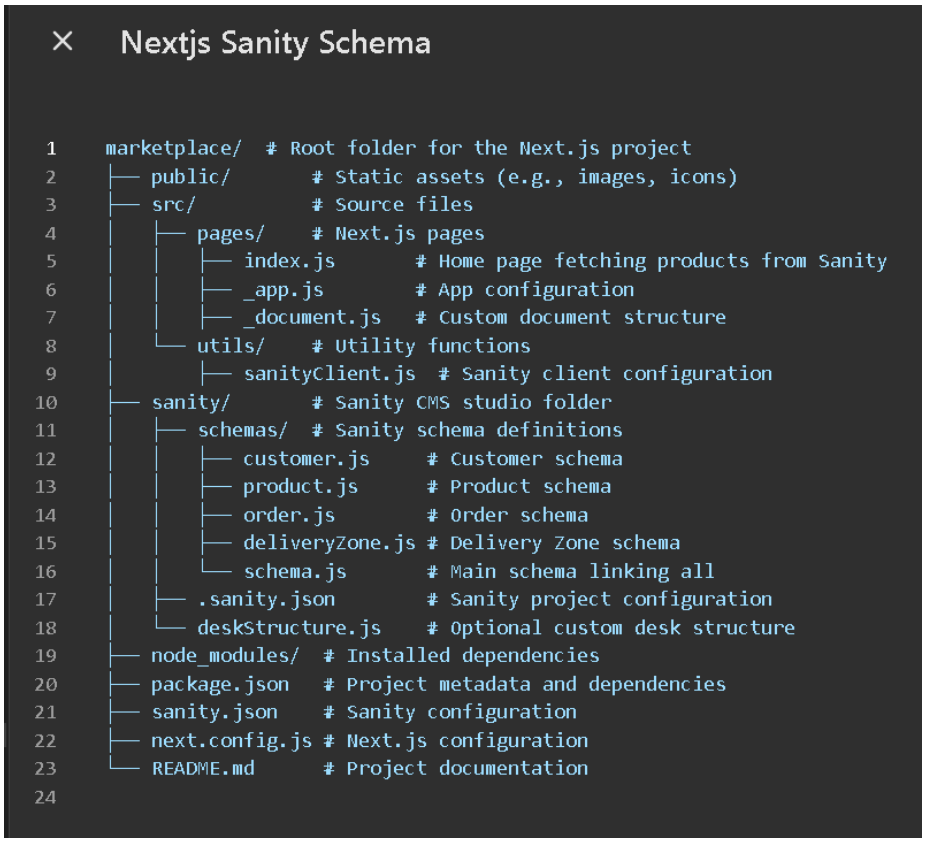

Figure 9: Image