ISDN Sales Distribution System - Overall System Design

This document details the software architecture, design patterns, and development environment for the ISDN Sales Distribution System.

1. Use Case Diagram

Describes the functional requirements from the perspective of different actors.

```
usecaseDiagram
    actor "System Administrator" as Admin
    actor "Sales Representative" as SalesRep
    actor "RDC Inventory Clerk" as Inventory
    actor "Logistics Officer" as Logistics
    actor "Delivery Driver" as Driver
    actor "Head Office Manager" as Manager
    actor "Online Customer" as Customer

    package "Management System" {
        usecase "Manage Users & Permissions" as UC1
        usecase "Database Configuration" as UC2
        usecase "Manage Product Catalog" as UC3
        usecase "Order Fulfillment Workflow" as UC4
        usecase "Real-time Inventory Tracking" as UC5
        usecase "Route & Delivery Assignment" as UC6
        usecase "Business Intelligence & Analytics" as UC7
        usecase "Invoice Generation" as UC8
    }

    package "Customer Portal" {
        usecase "Browse/Search Products" as UC9
        usecase "Manage Shopping Cart" as UC10
        usecase "Secure Checkout (Stripe)" as UC11
        usecase "Track Order Status" as UC12
    }

    Admin --> UC1
    Admin --> UC2

    SalesRep --> UC3
    SalesRep --> UC4
    SalesRep --> UC8

    Inventory --> UC5
    Inventory --> UC3

    Logistics --> UC6
    Logistics --> UC7

    Driver --> UC6
    Driver --> UC12

    Manager --> UC7
    Manager --> UC8

    Customer --> UC9
    Customer --> UC10
    Customer --> UC11
    Customer --> UC12
```

2. Class Diagram

Represents the data structures and relationships within the MongoDB database.

```
classDiagram
    class User {
        +ObjectId _id
        +String name
        +String email
        +String role
        +String status
        +String company
        +String phone
        +String address
        +Date lastLogin
        +login()
        +updateProfile()
    }

    class Product {
        +ObjectId _id
        +String name
        +String sku
        +Number price
        +Number stock
        +Number available
        +String rdc
        +String status
        +updateStock()
        +validateAvailability()
    }

    class Order {
        +ObjectId _id
        +String orderId
        +String customer
        +String address
        +Number totalAmount
        +String status
        +String priority
        +Date orderDate
        +String paymentMethod
        +processPayment()
        +updateStatus()
    }

    class OrderItem {
        +ObjectId product
        +Number quantity
        +Number price
    }

    class Role {
        <<enumeration>>
        ADMIN
        SALES
        INVENTORY
        LOGISTICS
        DRIVER
        MANAGER
        CUSTOMER
    }

    User "1" --> "0..*" Order : manages/places
    Order "1" *-- "1..*" OrderItem : contains
    Product "1" -- "0..*" OrderItem : reference
    User .. Role : strictly enforced
```

3. 3-Tier Architecture Diagram

Illustrates the separation of concerns across the technology stack.

```
graph TD
    subgraph "Presentation Tier"
        A["Customer Frontend (React + Vite)"]
        B["Management Dashboard (React + Vite)"]
        C["Mobile View (Responsive CSS)"]
    end

    subgraph "Application Tier (Backend)"
        D["REST API (Node.js/Express)"]
        E["Authentication (JWT/Passport)"]
        F["Business Logic Services"]
        G["Payment Integration (Stripe SDK)"]
    end

    subgraph "Data Tier"
        H["Primary Database (MongoDB)"]
        I["Object Data Modeling (Mongoose)"]
    end

    A & B & C <-->|"HTTPS / JSON"| D
    D <--> E
    D <--> F
    F <--> G
    F <--> I
    I <--> H
```

4. Development Tools & Environment

The following tools and technologies were utilized in the development of this system:

Category	Tools / Technologies
Frontend	React 18, Vite, Tailwind CSS, Lucide React, Framer Motion
Backend	Node.js, Express.js, TypeScript, Nodemon
Database	MongoDB, Mongoose ODM
Payment Gateway	Stripe API, React-Stripe-JS
Diagramming	Mermaid.js (UML Generation)
Version Control	Git
API Testing	Postman / Insomnia
Package Management	NPM (Node Package Manager)
IDE	VS Code

5. Security Measures

- **Password Hashing**: (Planned) Bcrypt storage.
- **Data sanitization**: MongoDB query protection.
- **Authentication**: Role Based Access Control (RBAC).

Generated by ISDN Sales Distribution System (Community) Architecture Team.