perplexity

Commerce Platform PoC - 3-Day Implementation Plan

CRITICAL TIMELINE: 3 DAYS MAXIMUM

```
Day 1: Setup + Core Models + Authentication
Day 2: Seller Features + Admin KYC Approval
Day 3: Marketplace Features + Integration Testing
```

TECHNOLOGY STACK (AS PER SRS)

Component	Version	Purpose
Go	1.21+	Backend language
Gin	Latest	Web framework
GORM	Latest	ORM
PostgreSQL	14+	Primary database
Redis	7.0+	Cache/sessions
MinIO	Latest	Media storage
Casbin	v2+	RBAC
JWT	HS256	Authentication
Zap	Latest	Logging
Prometheus	Latest	Metrics
Grafana	Latest	Monitoring

External Services (Sandbox Only):

- Payment: Razorpay/PayU/Cashfree sandbox
- Logistics: Shiprocket sandbox
- Search: PostgreSQL full-text

PROJECT STRUCTURE (SINGLE REPO, MODULAR MONOLITH)

```
commerce-platform-poc/
├── cmd/
│   ├── seller-api/main.go
│   ├── marketplace-api/main.go
│   └── admin-api/main.go
├── internal/
│   ├── models/           # ALL database models
│   │   ├── user.go
│   │   ├── seller.go
│   │   ├── kyc.go
│   │   ├── product.go
│   │   ├── sku.go
│   │   ├── inventory.go
│   │   ├── category.go
│   │   ├── cart.go
│   │   ├── order.go
│   │   ├── payment.go
│   │   ├── shipment.go
│   │   ├── return.go
│   │   ├── coupon.go
│   │   ├── review.go
│   │   ├── address.go
│   │   └── audit.go
│   ├── common/
│   │   ├── auth/jwt.go
│   │   ├── config/config.go
│   │   ├── db/postgres.go
│   │   └── errors/errors.go
│   ├── seller/
│   │   ├── handlers/
│   │   ├── services/
│   │   └── routes.go
│   ├── marketplace/
│   │   ├── handlers/
│   │   ├── services/
│   │   └── routes.go
│   ├── admin/
│   │   ├── handlers/
│   │   ├── services/
│   │   └── routes.go
│   └── integrations/
│       ├── payment/razorpay.go
│       ├── logistics/shiprocket.go
│       └── storage/minio.go
├── migrations/
│   ├── 001_users.sql
│   ├── 002_sellers.sql
│   ├── 003_products.sql
│   └── ...
└── docker-compose.yml
```

COMPLETE DATA MODELS (AS PER SRS)

Core Models

```
// internal/models/user.go
type User struct {
    ID           uint      `gorm:"primaryKey"`
    Name         string    `gorm:"not null"`
    Email        string    `gorm:"unique"`
    Phone        string    `gorm:"unique"`
    PasswordHash string
    Status       int       `gorm:"default:1"` // 1=active, 0=inactive
    TwoFAEnabled bool      `gorm:"default:false"`
    CreatedAt    time.Time
    UpdatedAt    time.Time
}

// internal/models/seller.go
type Seller struct {
    ID          uint   `gorm:"primaryKey"`
    LegalName   string `gorm:"not null"`
    DisplayName string
    GSTIN       string
    PAN         string
    BankRef     string
    Status      int `gorm:"default:0"` // 0=pending, 1=approved, 2=rejected
    RiskScore   int `gorm:"default:0"`
    CreatedAt   time.Time
    UpdatedAt   time.Time
}

type SellerUser struct {
    ID       uint `gorm:"primaryKey"`
    SellerID uint `gorm:"not null"`
    UserID   uint `gorm:"not null"`
    Role     string
    Status   int `gorm:"default:1"`
}

// internal/models/kyc.go
type KYC struct {
    ID          uint   `gorm:"primaryKey"`
    SellerID    uint   `gorm:"not null"`
    Type        string // PAN, GSTIN, etc.
    DocumentURL string
    Status      int    `gorm:"default:0"` // 0=pending, 1=approved, 2=rejected
    Remarks     string
    CreatedAt   time.Time
}
```

```go
// internal/models/category.go
type Category struct {
    ID               uint            `gorm:"primaryKey"`
    ParentID         *uint
    Name             string          `gorm:"not null"`
    AttributesSchema json.RawMessage `gorm:"type:jsonb"`
    SEOSlug          string
    CreatedAt        time.Time
}

// internal/models/product.go
type Product struct {
    ID          uint   `gorm:"primaryKey"`
    SellerID    uint   `gorm:"not null"`
    CategoryID  uint   `gorm:"not null"`
    Title       string `gorm:"not null"`
    Description text
    Brand       string
    Status      int `gorm:"default:0"` // 0=draft, 1=published
    Score       int `gorm:"default:0"`
    CreatedAt   time.Time
    UpdatedAt   time.Time
}

// internal/models/sku.go
type SKU struct {
    ID         uint            `gorm:"primaryKey"`
    ProductID  uint            `gorm:"not null"`
    SKUCode    string          `gorm:"unique;not null"`
    Attributes json.RawMessage `gorm:"type:jsonb"`
    PriceMRP   decimal.Decimal `gorm:"type:decimal(10,2)"`
    PriceSell  decimal.Decimal `gorm:"type:decimal(10,2)"`
    TaxPct     decimal.Decimal `gorm:"type:decimal(5,2)"`
    Barcode    string
    CreatedAt  time.Time
}

// internal/models/inventory.go
type Inventory struct {
    ID         uint `gorm:"primaryKey"`
    SKUID      uint `gorm:"not null"`
    LocationID uint `gorm:"not null"`
    OnHand     int  `gorm:"default:0"`
    Reserved   int  `gorm:"default:0"`
    Threshold  int  `gorm:"default:0"`
    UpdatedAt  time.Time
}

// internal/models/cart.go
type Cart struct {
    ID        uint   `gorm:"primaryKey"`
    UserID    uint   `gorm:"not null"`
    Currency  string `gorm:"default:INR"`
    CreatedAt time.Time
}
```

```
type CartItem struct {
    ID        uint            `gorm:"primaryKey"`
    CartID    uint            `gorm:"not null"`
    SKUID     uint            `gorm:"not null"`
    Qty       int             `gorm:"not null"`
    Price     decimal.Decimal `gorm:"type:decimal(10,2)"`
    CreatedAt time.Time
}

// internal/models/order.go
type Order struct {
    ID            uint            `gorm:"primaryKey"`
    UserID        uint            `gorm:"not null"`
    Total         decimal.Decimal `gorm:"type:decimal(10,2)"`
    Tax           decimal.Decimal `gorm:"type:decimal(10,2)"`
    Shipping      decimal.Decimal `gorm:"type:decimal(10,2)"`
    Status        int             `gorm:"default:0"` // 0=new, 1=confirmed, 2=shipped, et
    PaymentStatus int             `gorm:"default:0"` // 0=pending, 1=captured, 2=failed
    AddressID     uint            `gorm:"not null"`
    CreatedAt     time.Time
    UpdatedAt     time.Time
}

type OrderItem struct {
    ID         uint            `gorm:"primaryKey"`
    OrderID    uint            `gorm:"not null"`
    SKUID      uint            `gorm:"not null"`
    Qty        int             `gorm:"not null"`
    Price      decimal.Decimal `gorm:"type:decimal(10,2)"`
    Tax        decimal.Decimal `gorm:"type:decimal(10,2)"`
    SellerID   uint            `gorm:"not null"`
    ShipmentID *uint
}

// internal/models/payment.go
type Payment struct {
    ID        uint            `gorm:"primaryKey"`
    OrderID   uint            `gorm:"not null"`
    IntentID  string
    Provider  string          // razorpay, payu, etc.
    Amount    decimal.Decimal `gorm:"type:decimal(10,2)"`
    Status    int             `gorm:"default:0"` // 0=pending, 1=captured, 2=failed
    TxnRef    string
    CreatedAt time.Time
}

// internal/models/shipment.go
type Shipment struct {
    ID        uint   `gorm:"primaryKey"`
    OrderID   uint   `gorm:"not null"`
    Provider  string // shiprocket, etc.
    AWB       string
    Status    int    `gorm:"default:0"` // 0=created, 1=shipped, 2=delivered
    ETA       *time.Time
    CreatedAt time.Time
}
```

```go
// internal/models/return.go
type Return struct {
    ID          uint   `gorm:"primaryKey"`
    OrderID     uint   `gorm:"not null"`
    OrderItemID uint   `gorm:"not null"`
    Reason      string
    Status      int `gorm:"default:0"` // 0=requested, 1=approved, 2=rejected
    RefundID    *uint
    CreatedAt   time.Time
}

type Refund struct {
    ID          uint            `gorm:"primaryKey"`
    PaymentID   uint            `gorm:"not null"`
    Amount      decimal.Decimal `gorm:"type:decimal(10,2)"`
    Status      int             `gorm:"default:0"` // 0=pending, 1=processed
    ProcessedAt *time.Time
    CreatedAt   time.Time
}

// internal/models/coupon.go
type Coupon struct {
    ID         uint            `gorm:"primaryKey"`
    Code       string          `gorm:"unique;not null"`
    Type       int             // 1=percentage, 2=fixed
    Value      decimal.Decimal `gorm:"type:decimal(10,2)"`
    Conditions json.RawMessage `gorm:"type:jsonb"`
    StartAt    time.Time
    EndAt      time.Time
    UsageLimit int
    CreatedAt  time.Time
}

// internal/models/review.go
type Review struct {
    ID        uint            `gorm:"primaryKey"`
    UserID    uint            `gorm:"not null"`
    ProductID uint            `gorm:"not null"`
    Rating    int             `gorm:"not null"`
    Text      string
    Media     json.RawMessage `gorm:"type:jsonb"`
    Status    int             `gorm:"default:1"` // 1=approved, 0=pending
    CreatedAt time.Time
}

// internal/models/address.go
type Address struct {
    ID       uint   `gorm:"primaryKey"`
    UserID   *uint  // for marketplace users
    SellerID *uint  // for sellers
    Line1    string `gorm:"not null"`
    Line2    string
    City     string `gorm:"not null"`
    State    string `gorm:"not null"`
    Country  string `gorm:"not null"`
    Pin      string `gorm:"not null"`
    CreatedAt time.Time
}

// internal/models/audit.go
type AuditLog struct {
    ID       uint            `gorm:"primaryKey"`
    Actor    string          `gorm:"not null"`
    Action   string          `gorm:"not null"`
    Entity   string          `gorm:"not null"`
    EntityID uint            `gorm:"not null"`
    Meta     json.RawMessage `gorm:"type:jsonb"`
    CreatedAt time.Time
}

// internal/models/media.go
type Media struct {
    ID         uint   `gorm:"primaryKey"`
    EntityType string `gorm:"not null"` // product, review, etc.
    EntityID   uint   `gorm:"not null"`
    URL        string `gorm:"not null"`
    Type       string // image, video
    AltText    string
    Sort       int `gorm:"default:0"`
    CreatedAt  time.Time
}
```

API ENDPOINTS (AS PER SRS)

Authentication

- `POST /v1/auth/register`
- `POST /v1/auth/login`
- `POST /v1/auth/refresh`
- `POST /v1/auth/otp`

Seller API

- `POST /v1/sellers` (create)
- `GET /v1/sellers/{id}`
- `PATCH /v1/sellers/{id}`
- `POST /v1/sellers/{id}/kyc` (upload)
- `GET /v1/sellers/{id}/kyc` (list)
- `POST /v1/sellers/{id}/products`
- `POST /v1/products/{id}/skus`
- `GET /v1/skus/{id}/inventory`

- `PATCH /v1/skus/{id}/inventory`
- `POST /v1/media/upload`

Marketplace API

- `GET /v1/products?query=&category=&filters=`
- `GET /v1/products/{id}`
- `POST /v1/carts`
- `POST /v1/carts/{id}/items`
- `PATCH /v1/carts/{id}/items/{itemId}`
- `POST /v1/checkout`
- `POST /v1/orders`
- `POST /v1/payments/intents`
- `POST /v1/payments/capture`
- `POST /v1/products/{id}/reviews`
- `GET /v1/products/{id}/reviews`

Admin API

- `GET /v1/admin/kyc/pending`
- `POST /v1/admin/kyc/{id}/approve`
- `POST /v1/admin/kyc/{id}/reject`
- `POST /v1/admin/catalog/categories`
- `PATCH /v1/admin/catalog/attributes`

Logistics & Webhooks

- `POST /v1/shipments/rates`
- `POST /v1/shipments/{id}/label`
- `GET /v1/shipments/{id}/track`
- `POST /v1/hooks/payments`
- `POST /v1/hooks/logistics`

3-DAY IMPLEMENTATION SCHEDULE

DAY 1: FOUNDATION

Morning (4 hours)

1. **Environment Setup**

```
# PostgreSQL, Redis, MinIO setup
docker-compose up -d

# Go module init
go mod init commerce-platform-poc
go get github.com/gin-gonic/gin
go get gorm.io/gorm gorm.io/driver/postgres
go get github.com/go-redis/redis/v8
go get github.com/casbin/casbin/v2
```

2. **Database Schema**
   - Run all migration files (001-020)
   - Seed basic categories and admin user
3. **Common Infrastructure**
   - Database connection
   - JWT middleware
   - Error handling
   - Config management

Afternoon (4 hours)

4. **Authentication System**
   - User registration/login
   - JWT token generation
   - RBAC setup with Casbin
   - Password hashing
5. **Basic API Framework**
   - Gin router setup
   - Middleware chain
   - Health check endpoints

DAY 2: SELLER + ADMIN

Morning (4 hours)

1. **Seller Registration & KYC**
    - Seller account creation
    - KYC document upload to MinIO
    - Bank verification stubs
2. **Product Catalog**
    - Product/SKU creation
    - Media upload handling
    - Category assignment

Afternoon (4 hours)

3. **Admin KYC Approval**
    - KYC queue listing
    - Approve/reject workflow
    - Audit logging
4. **Inventory Management**
    - Stock updates
    - Reservation logic
    - Low stock alerts

DAY 3: MARKETPLACE + INTEGRATION

Morning (4 hours)

1. **Product Discovery**
    - Search with PostgreSQL full-text
    - Product listing with filters
    - Product detail pages
2. **Cart & Checkout**
    - Cart operations
    - Checkout validation
    - Address management

Afternoon (4 hours)

3. **Payment Integration**
    - Razorpay sandbox integration
    - Payment capture flow
    - Order creation
4. **Integration Testing**
    - End-to-end happy path
    - Order status updates
    - Return initiation

TEAM ALLOCATION

Team 1: Seller Features

Responsibilities: Seller registration, KYC, catalog management, inventory
Models: User, Seller, SellerUser, KYC, Product, SKU, Inventory, Media, Address
Endpoints: All seller API endpoints

Team 2: Marketplace Features

Responsibilities: User auth, product discovery, cart, checkout, payments, orders
Models: User, Cart, CartItem, Order, OrderItem, Payment, Review, Address
Endpoints: All marketplace API endpoints

Team 3: Admin Features

Responsibilities: KYC approval, catalog governance, basic reports, CMS
Models: AuditLog, Category
Endpoints: All admin API endpoints

SETUP INSTRUCTIONS

Prerequisites

```
# Install Go 1.21+
sudo apt install golang-go

# Install PostgreSQL 14+
sudo apt install postgresql postgresql-contrib

# Install Redis
sudo apt install redis-server
```

```
# Install Docker
sudo apt install docker.io docker-compose
```

Project Setup

```
# Clone and setup
git clone <repo>
cd commerce-platform-poc

# Start services
docker-compose up -d

# Install Go dependencies
go mod tidy

# Run migrations
psql -U commerce_user -d commerce_poc -f migrations/001_users.sql
# ... run all migrations

# Start services
go run cmd/seller-api/main.go &
go run cmd/marketplace-api/main.go &
go run cmd/admin-api/main.go &
```

Environment Variables

```
export DB_HOST=localhost
export DB_PORT=5432
export DB_USER=commerce_user
export DB_PASS=password
export DB_NAME=commerce_poc
export REDIS_URL=redis://localhost:6379
export MINIO_ENDPOINT=localhost:9000
export MINIO_ACCESS_KEY=minioadmin
export MINIO_SECRET_KEY=minioadmin
export JWT_SECRET=your-secret-key
export RAZORPAY_KEY_ID=rzp_test_xxx
export RAZORPAY_KEY_SECRET=xxx
```

CRITICAL SUCCESS CRITERIA

End-to-End Happy Path

1. ✅ Seller registers → uploads KYC → admin approves
2. ✅ Seller creates product → adds inventory
3. ✅ User searches product → adds to cart → checkout
4. ✅ Payment captured → order created → seller can process

Must-Have Features

- [x] Authentication (JWT)
- [x] RBAC (Casbin)
- [x] File upload (MinIO)
- [x] Payment sandbox (Razorpay)
- [x] PostgreSQL full-text search
- [x] Basic audit logging
- [x] Simple returns initiation

Performance Targets

- API response time < 500ms
- Database queries optimized
- File uploads working
- Search functionality operational

TESTING PLAN

Day 3 Final Testing

```
# Test complete flow
curl -X POST localhost:8080/v1/sellers/register
curl -X POST localhost:8081/v1/admin/kyc/approve
curl -X POST localhost:8080/v1/sellers/1/products
curl -X GET localhost:8082/v1/products/search
curl -X POST localhost:8082/v1/carts
curl -X POST localhost:8082/v1/checkout
```

This plan sticks strictly to the SRS scope and is achievable in 3 days with focused execution.

⁂

1. Document-9-1.docx
2. GoSocialPlan.pdf