# ZOLD Clothing Brand Website - Design Document

Prepared for: Rahul  
Prepared by: Coolie (ChatGPT)  
Date: September 8, 2025

## 1. Analysis Report

Project Overview:  
ZOLD is a clothing brand developing an e-commerce website to sell its products online. This system provides customers a responsive storefront to browse, search, and order clothing items, and allows admins to manage products and orders.

Objectives:

- Provide a digital storefront for ZOLD.

- Improve sales channels and customer reach.

- Provide secure and reliable order processing.

Scope:

- In Scope: User Registration/Login, Product Browsing, Shopping Cart, Order Placement, Payment Integration, Order Tracking, Admin Panel.

- Out of Scope: Mobile App, AI Recommendations, Multi-language Support (Phase 2).

Feasibility Summary:

- Technical: Feasible using modern web stack (React/Django or React/Node).

- Operational: Aligns with business goals for online expansion.

- Economic: Reasonable initial cost, expected revenue uplift.

## 2. Software Requirement Specification (SRS)

2.1 Introduction:

- System Name: ZOLD Clothing Brand Website

- Purpose: E-commerce platform for ZOLD customers to purchase products online.

- Users: Customers, Admins

2.2 Functional Requirements:

- User Management: Register/Login, Edit profile, View order history.

- Product Management: Browse by category, View product details, Search & filter.

- Order Management: Add to cart, Place order, Payment, Track order, Cancel/Return.

- Admin Features: CRUD products, Manage orders, View sales reports.

2.3 Non-Functional Requirements:

- Performance: Page load under 3 seconds for typical connections.

- Security: Password hashing, HTTPS, secure payment gateway, input validation.

- Usability: Responsive UI, accessible navigation.

- Scalability: Support for 1000+ concurrent users (initial target).

2.4 System Constraints:

- Browser compatibility for modern browsers (Chrome, Edge, Firefox, Safari).

- Payment integration via third-party gateway (e.g., Stripe/PayPal).

## 3. Design Document

3.1 System Architecture:

- Frontend: React (recommended) or plain HTML/CSS/JS for prototype.

- Backend: Node.js (Express) or Python (Django/Flask).

- Database: PostgreSQL or MySQL.

- Hosting: Cloud (AWS / Heroku / DigitalOcean).

- Payment Gateway: Stripe or PayPal.

### 3.2 Module Design

#### A. User Module

Inputs: Username, Email, Password, Personal Details

Processes: Authentication, Profile Management, Order History Fetching

Outputs: User Dashboard, Profile Details, Past Orders

Entity: User(UserID, Name, Email, Password, Address, Phone)

#### B. Product Module

Inputs: Product details by admin (Name, Price, Size, Category, Stock, Image)

Processes: Display catalog, filtering, search, stock validation

Outputs: Product list page, Product details page

Entity: Product(ProductID, Name, Description, Price, Category, Stock, ImageURL)

#### C. Order Module

Inputs: Cart items, UserID, Payment details

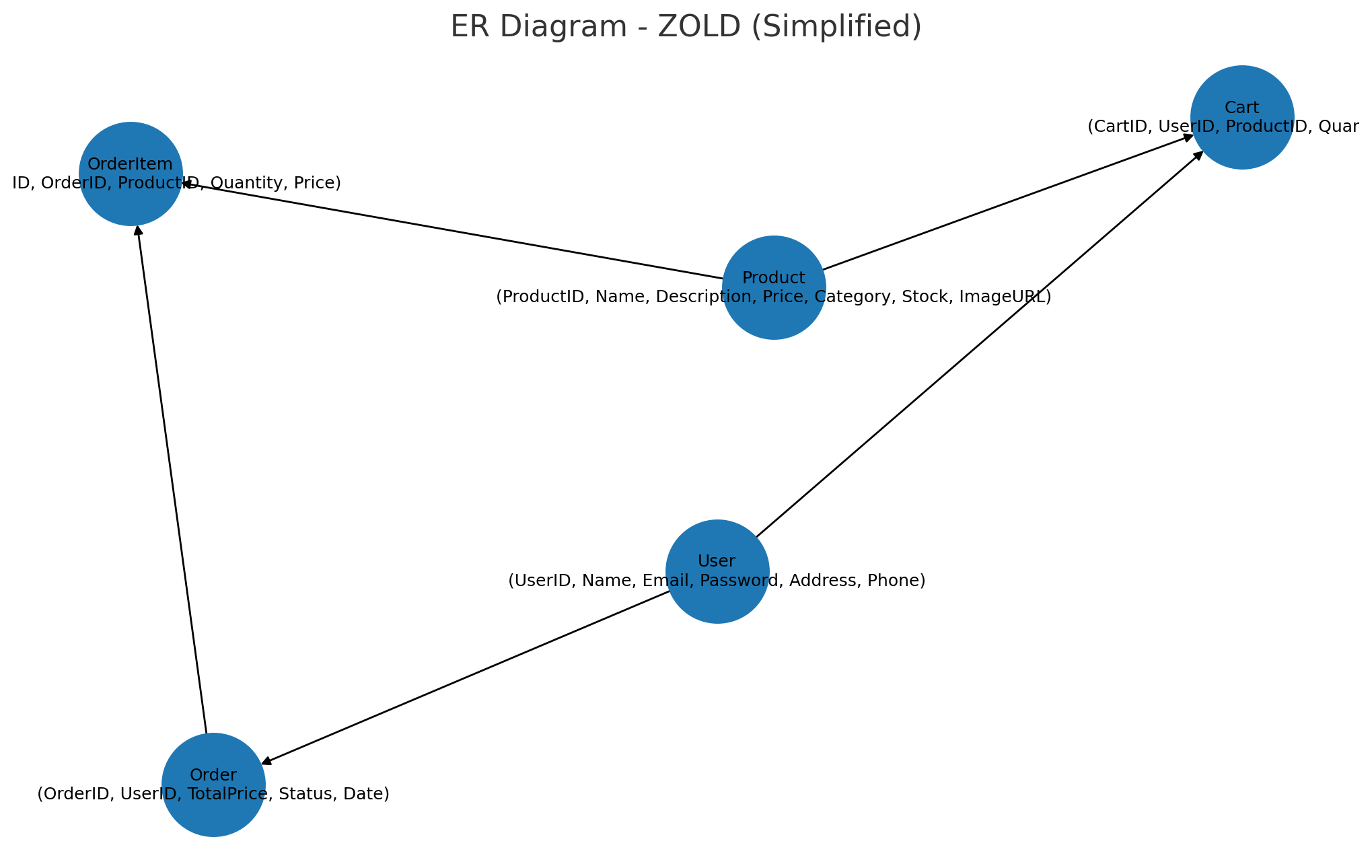
Processes: Order validation, Payment processing, Invoice generation, Order status updates

Outputs: Order Confirmation, Tracking status, Order history

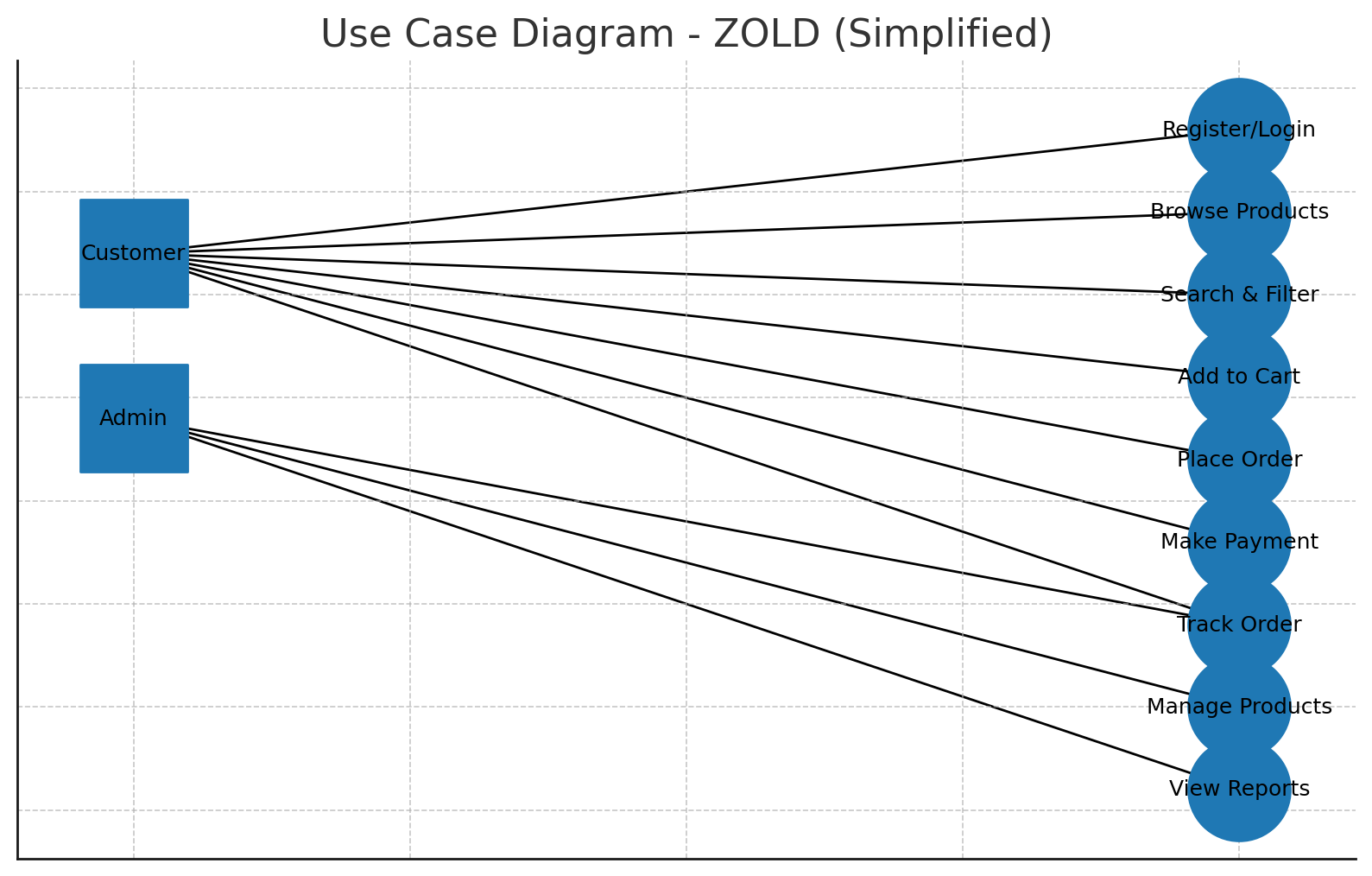
Entities: Order(OrderID, UserID, TotalPrice, Status, Date), OrderItem(OrderItemID, OrderID, ProductID, Quantity, Price)

## 4. Diagrams

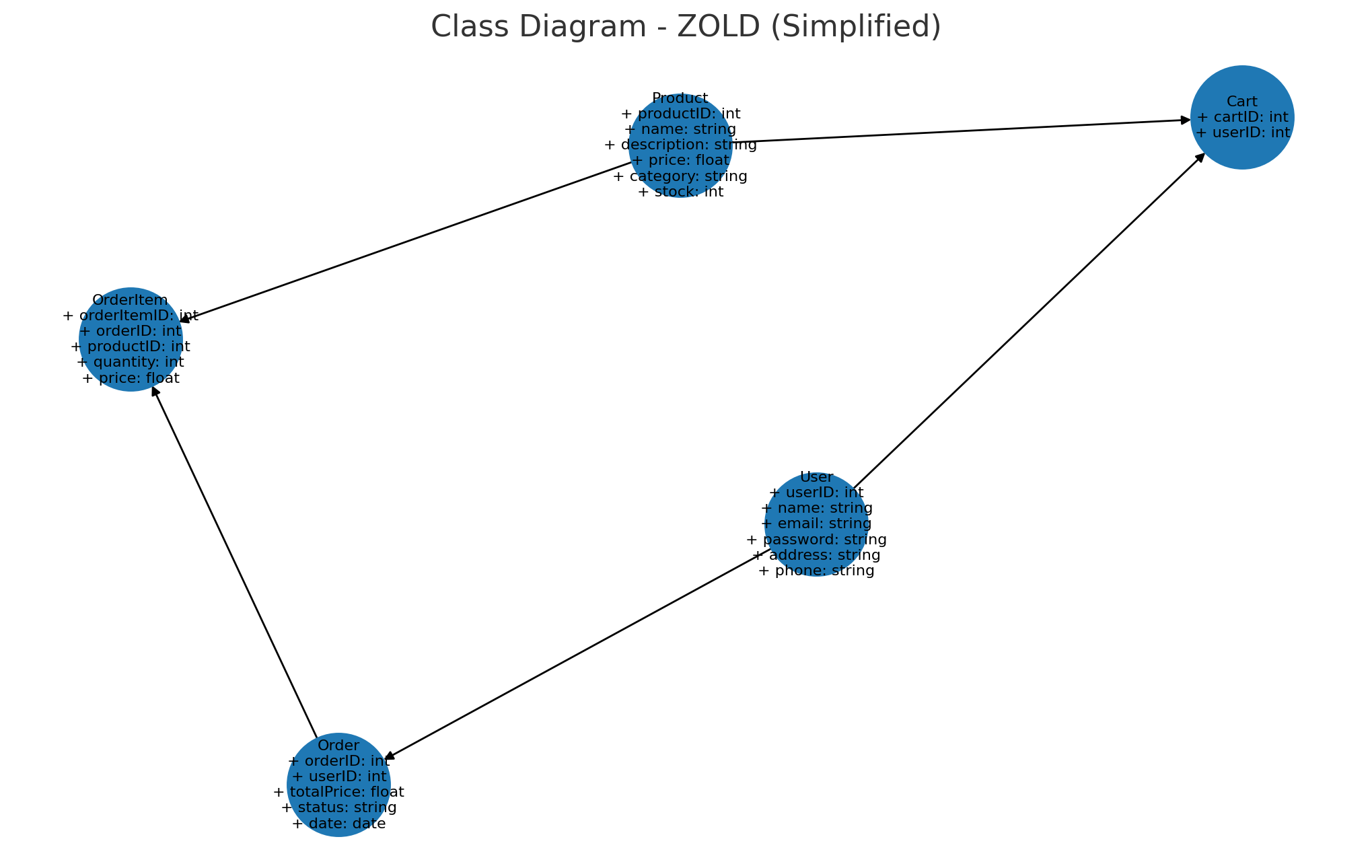
ER Diagram (Simplified):



Use Case Diagram (Simplified):



Class Diagram (Simplified):



## 5. Additional Details

5.1 Sample REST API Endpoints:

- POST /api/register - Register user

- POST /api/login - Login and receive token

- GET /api/products - List products (supports query params for filtering)

- GET /api/products/{id} - Product details

- POST /api/cart - Add item to cart

- GET /api/cart - View cart

- POST /api/orders - Create order (payment step)

- GET /api/orders/{id} - Order details / tracking

- PUT /api/admin/products - Admin Create/Update/Delete products (protected)

5.2 Database Schema (Summary):

- users (id, name, email, password\_hash, address, phone, created\_at)

- products (id, name, description, price, category, stock, image\_url, created\_at)

- cart\_items (id, user\_id, product\_id, quantity, added\_at)

- orders (id, user\_id, total\_price, status, created\_at)

- order\_items (id, order\_id, product\_id, quantity, price)