Database Design Document: DevOps Graduation Project

This document presents the database design for the DevOps Graduation Project — a simple shopping website.
It defines the database structure, relationships, and key entities required to support the system's functionality.
The design focuses on scalability, normalization, and compatibility with MySQL as the database engine.

1. Purpose

The purpose of this document is to outline the logical and physical design of the database system used in the
shopping application. It ensures that the data model aligns with the business requirements such as user
management, product listing, purchases, and transaction tracking.

2. Database Overview

The system uses a **MySQL** relational database to store all persistent data. The backend microservices
(Auth, Products, Orders) interact with the database through RESTful APIs developed using FastAPI.

3. Entities and Relationships

The core entities in the database include:
- **Users**: Stores user information, authentication details, and account balance (coins).
- **Products**: Contains details about available products including name, price, and stock quantity.
- **Orders**: Tracks purchases made by users, including total amount and purchase date.
- **Order_Items**: Maps products to orders to support multiple items per purchase.
- **Transactions**: Records all balance changes for audit and reporting.

4. Entity Relationship Diagram (ERD)

Below is a textual representation of the ERD (can be visualized later using tools like draw.io

or MySQL Workbench):

Users (UserID, Username, Email, PasswordHash, Coins, CreatedAt)
Products (ProductID, Name, Description, Price, Stock, CreatedAt)
Orders (OrderID, UserID, TotalAmount, OrderDate)
Order_Items (OrderItemID, OrderID, ProductID, Quantity, Subtotal)
Transactions (TransactionID, UserID, Amount, Type, TransactionDate, Description)

Relationships:
- One User can have many Orders (1:N)
- One Order can contain many Products through Order_Items (M:N)
- One User can have many Transactions (1:N)

5. Table Definitions

Users Table

- UserID (INT, PK, AUTO_INCREMENT)
- Username (VARCHAR 50)
- Email (VARCHAR 100, UNIQUE)
- PasswordHash (VARCHAR 255)
- Coins (DECIMAL 10,2, DEFAULT 0)
- CreatedAt (DATETIME, DEFAULT CURRENT_TIMESTAMP)

Products Table

- ProductID (INT, PK, AUTO_INCREMENT)
- Name (VARCHAR 100)
- Description (TEXT)
- Price (DECIMAL 10,2)
- Stock (INT)
- CreatedAt (DATETIME, DEFAULT CURRENT_TIMESTAMP)

Orders Table

- OrderID (INT, PK, AUTO_INCREMENT)
- UserID (INT, FK -> Users.UserID)
- TotalAmount (DECIMAL 10,2)
- OrderDate (DATETIME, DEFAULT CURRENT_TIMESTAMP)

Order_Items Table

- OrderItemID (INT, PK, AUTO_INCREMENT)
- OrderID (INT, FK -> Orders.OrderID)
- ProductID (INT, FK -> Products.ProductID)
- Quantity (INT)
- Subtotal (DECIMAL 10,2)

Transactions Table

- TransactionID (INT, PK, AUTO_INCREMENT)
- UserID (INT, FK -> Users.UserID)
- Amount (DECIMAL 10,2)
- Type (ENUM('CREDIT','DEBIT'))
- TransactionDate (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- Description (VARCHAR 255)

6. Database Normalization

The database design adheres to the Third Normal Form (3NF):
- Each table has a primary key.
- No repeating groups or redundant data.
- Non-key attributes depend solely on the primary key.
This ensures minimal redundancy and efficient querying.

7. Data Flow Summary

1. A user registers and logs in via the Auth microservice (stored in Users table).
2. The user views available products (Products table).
3. Upon purchase, an order is created (Orders, Order_Items tables) and coins are deducted (Transactions table).
4. Users can review previous purchases and balances through API endpoints connecting to these tables.

8. Implementation Considerations

- Use MySQL Workbench for initial schema setup.
- Use SQLAlchemy ORM with FastAPI for data access.
- Initialize tables using Alembic for migration management.
- Use connection pooling for scalability.

- Back up database automatically using AWS RDS snapshot or cron-based script.

9. Conclusion

This database design provides a robust and normalized schema for the shopping website project,
ensuring efficient data management, security, and scalability for future enhancements.