Documentation - E-shop Project

1. Requirements

Functional Requirements

- **Database:** Manage Categories, Products, Customers, Orders, and OrderItems.
- **Transactions:** Order creation involves updating multiple tables transactionally.
- **Import:** Import Categories from CSV and Products from JSON.
- **Report:** View aggregated statistics of customer activity.
- **UI:** Web interface to browse products, view orders, and import data.

Non-functional Requirements

- **Language:** Python 3.x with Flask.
- **Database:** MySQL.
- **Pattern:** Repository Pattern (D1).
- **Environment:** Runs on school PCs (configurable via simple file).

---

2. Architecture

High-Level Overview

The application follows a Layered Architecture:

1. **Presentation Layer:** Flask Web App (`app.py` , `templates/`). Handles HTTP requests and renders HTML.
2. **Service Layer (Implicit/Explicit):** `ImportService` handles business logic for file processing.
3. **Data Access Layer:** Repository Pattern (`repositories/`). Abstracts database interactions.
4. **Database Layer:** MySQL Database.

Database Design (ER Model)

- **Categories:** `id` , `name`
- **Products:** `id` , `category_id` (FK), `name` , `price` , `description` , `is_active`
- **Customers:** `id` , `first_name` , `last_name` , `email`
- **Orders:** `id` , `customer_id` (FK), `date` , `status` , `total_price`
- **OrderItems:** `id` , `order_id` (FK), `product_id` (FK), `quantity` , `unit_price`

Relationships:

- Category 1:N Product
- Customer 1:N Order
- Order M:N Product (via OrderItems)

---

3. Libraries & Dependencies

- **Python Standard Library:** `json` , `csv` , `os`
- **External Libraries:**
  - `flask` : Web framework.
  - `mysql-connector-python` : Database driver.

---

4. Configuration & Installation

Configuration

Configuration is managed via `config.json` in the root directory.

```
{
    "database": {
        "host": "localhost",
        "user": "root",
        "password": "student",
        "database": "eshop_db"
    }
}
```

Installation & Run

1. Install dependencies: `pip install -r requirements.txt`
2. Import database: Run SQL commands from `src/database/schema.sql`.
3. Run application: `python -m src.main`
4. Access at: `http://127.0.0.1:5000`

---

5. Error Handling & Testing

- **Database Errors:** Handled in Repositories (try-except blocks). Logs error and returns safe default (e.g., None or empty list).
- **Import Errors:** Validates file existence and format. Returns success/failure status.
- **Transactions:** rollback performed on failure to ensure data consistency.

Verification

- **Manual Testing:** See `test/test_scenarios.md`.
- **Validation:** Check `view_order_summary` in database to verify order correctness.