CMPUT 291 Mini Project 1

E-Commerce Database System
Design Document

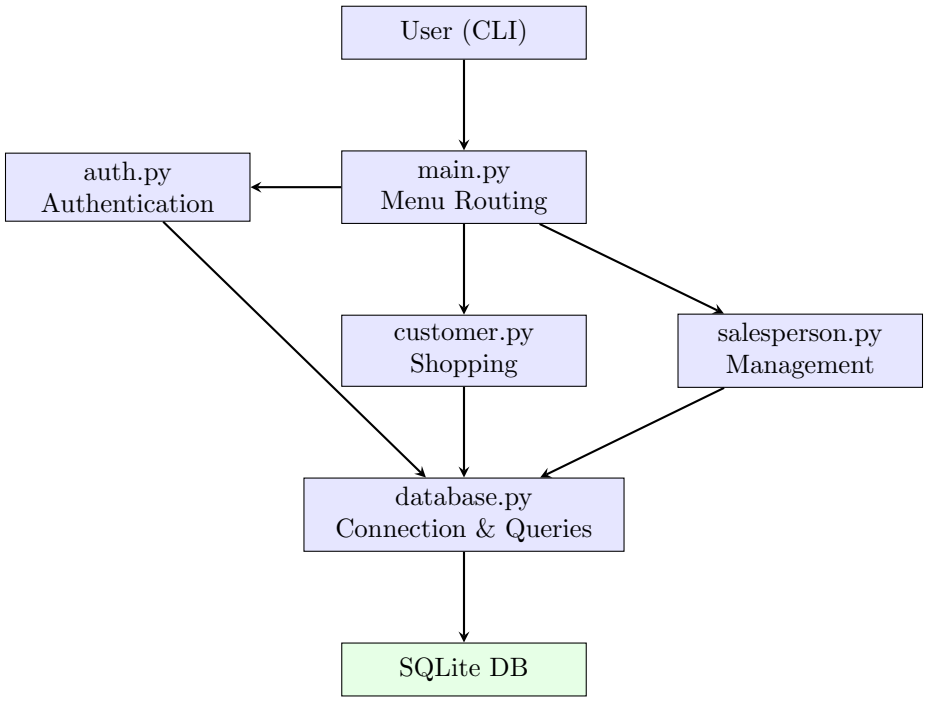
Chidinma Obi-Okoye – obiokoye
Diepreye Charles-Daniel – diepreye
Ugonna Noble Jr Akpulonu – akpulonu

November 3, 2025

1 System Overview and User Guide

Overview: A command-line e-commerce database system built with Python 3 and SQLite. It supports **customers** (shop, checkout, view orders) and **salespersons** (manage inventory, analyze sales).

Architecture:



Usage:

```
python3 main.py prj-test.db
```

Customers: login/signup → search → add to cart → checkout → view history. Salespersons: login → update products → generate reports.

2 Detailed Software Design

database.py: Central DB handler with secure parameterized queries. `execute_query()` (SELECT), `execute_update()` (INSERT/UPDATE/DELETE), `close()` (cleanup). Ensures transactions and prevents SQL injection.

auth.py: Handles login and registration. Validates user credentials, prevents duplicates, masks passwords using `getpass`. Returns user role for routing.

customer.py: Implements search, cart, checkout, and order history. Logs all actions (search/view). Checkout creates orders, updates stock, and clears cart atomically. Supports pagination for large order lists.

salesperson.py: Supports product management and reporting. `manage_product()` updates price/stock. `generate_sales_report()` aggregates weekly metrics. `top_selling/viewed_products()` ranks products with tie handling.

main.py: Entry point managing sessions and user routing. On login, creates session; on exit, updates session end time. Routes to `customer_menu()` or `salesperson_menu()` accordingly.

3 Testing Strategy

Approach: Unit, integration, and manual testing ensure correctness and robustness.

Automated Tests (127 total):

Category	Tests	Coverage
Authentication, Search, Cart, Checkout, Orders, Reports, SQL Security, Sessions	127	100%

Key Scenarios:

- Login/signup, invalid input handling.
- Multi-keyword product search, case-insensitivity.
- Checkout integrity: stock validation, transaction rollback.
- Sales reports, top product ranking, SQL injection attempts.

Bug Summary: 23 total issues found; 5 critical (integrity), 12 major, 6 minor—all resolved. Example: missing cart clearing fixed by adding `DELETE FROM cart WHERE cid=? AND sessionNo=?`.

Manual Tests: Executed on lab servers (ucomm-3140-w01.cs.ualberta.ca) with empty and large datasets. Verified password masking, query safety, and report accuracy.

4 Team Work and Coordination

Division of Work:

Person	Work Item	Hours
Chidinma Obi-Okoye	Database + Auth + Main + SQL Security	12
Diepreye Charles-Daniel	Search + Cart + Checkout + History	15
Ugonna Noble Jr Akpulonu	Reports + Management + Document + LLM Docs	13
All Members	Integration, Bug Fixes, Demo Prep	10 each

Coordination:

- **Communication:** Discord (daily), Zoom (3× weekly).
- **Code Management:** GitHub with branches, pull requests, and reviews.
- **Task Tracking:** GitHub Projects board (To Do/In Progress/Done).

Workflow: Each feature developed on a separate branch → PR review → merge after CI passes → test on shared DB.

Timeline:

- Week 1: Setup, DB, Auth, Core Features.
- Week 2: Integration, Testing, Docs, Demo Prep.
- Nov 4: Final submission.

Design & Security Highlights:

- Modular structure: independent, testable modules.
- Central DB handler for consistent access.
- Parameterized queries, password masking, and strict validation.
- Transaction-safe checkout and pagination for scalability.

Conclusion

This project demonstrates modular database design, secure coding, and collaborative development. The team achieved full functional coverage, reliable data handling, and maintainable structure within the 2-week timeline.