In this document, we will use the **Systems Development Life Cycle** to guide the development of the Supply Chain Management prototype.

**Step 1: Planning**

**High level objectives**

The system should serve as a supply chain management system for a wholesale seller. Customers should be able to place and manage orders through the system. Owners should be able to manage inventory through the system.

**Scope**

Database that services both users placing orders and owners using it for warehouse inventory.

**Timeline**

5-day build before interview.

**Step 2: Analysis**

**Requirements**

* Track orders and inventory using a SQL database containing an orders table and inventory table.
* Provide an interface for the user to place, update, and cancel orders.
* Ability to create, retrieve, update, and delete (CRUD) orders and inventory.
* Link orders table to inventory table through product’s SKU unit.
* Enable both manual test cases (Excel) and automated test cases (Pytest/GitHub Actions).

**User Stories**

* As an owner, I want to make sure that orders above warehouse quantity cannot be placed.
* As a user, I want to be able to place an order to the LDB system and receive status updates.

**Step 3: Design**

* Create a sample database in Excel to provide a starting point for SQL schema.
* Create a Python sqlite3 database schema with sample data from Excel to initialize the supply chain database.
  + Two tables. One for orders, the other for warehouse inventory.
  + Seek table data from Excel sample database.
* Write CRUD operations to modify orders table and inventory table in database.
* Write a Python console program for customers to place, check, update, or cancel an order.
  + Placing an order should only work if there is enough inventory to meet it.
  + Placing an order should only work if the product exists.
  + Placing a valid order should result in the inventory being updated to reflect the order.
* Write a Python console program for owners to update inventory.
  + Owners should be able to modify (add, subtract, or remove) items of inventory.

**Step 4: Development**

**Completed:**

* Wrote schema.sql to and db-init.py to create and seed orders and warehouse tables.
* Developed Python CRUD functions for orders table: create\_order, retrieve\_order, update\_order, and delete\_order.
  + Refactored duplicate code to Connect, Execute, and Close Cursor object interacting with database using CEC\_cursor function.
* Wrote Python console program to input new orders from customer side, which are then reflected in the SQL database.

**Step 5: Testing**

The below are examples of some test cases.

* **Unit Testing**: validate individual CRUD functions for valid inputs and outputs.
* **Integration Testing:** validate Python CRUD logic with database backend
  + **Ex:** Creating order results in order being reflected in database.
* **E2E (System Testing):** User inputting order should result in order being reflected in database.

**6. Implementation**

* Deployment
* Prototype packaged in GitHub repo with clear README.
* schema.sql for DB initialization.
* Console for user interaction
* End Users: Interview demo environment.

**7. Maintenance**

* **Planned Enhancements:**
  + Add performance/load testing (simulate multiple orders).
  + Expand schema to cover more supply chain entities (e.g., shipments, suppliers).
  + Integrate with AWS (S3 for receipts).
* **Feedback:** Adapt improvements based on user/interviewer feedback.