Project Report

<u>Inventory Management System</u>

Table of Contents:

Introduction:

1.1 Background:

Inventory management is a critical function for businesses of all sizes. Effective inventory management ensures that businesses have the right products in the right quantities at the right time. Traditional inventory management systems often rely on manual record-keeping, which can be time-consuming, error-prone, and inefficient.

The digital transformation of business processes has led to the development of computerized inventory management systems that automate and streamline inventory-related tasks. These systems provide real-time visibility into inventory levels, facilitate accurate record-keeping, and support data-driven decision-making.

1.2 Problem Statement:

Small and medium-sized businesses face several challenges in managing their inventory:

1. Inefficient manual processes: Manual inventory tracking is labor-intensive and prone to human error.
2. Lack of real-time visibility: Without a digital system, businesses struggle to maintain accurate, up-to-date information about their inventory.
3. Difficulty in tracking transactions: Recording sales, restocks, and other inventory movements manually can lead to discrepancies.
4. **Limited reporting capabilities:** Manual systems make it difficult to generate comprehensive reports for decision-making.
5. Scalability issues: As businesses grow, manual inventory management becomes increasingly complex and unmanageable.

1.3 Objectives:

The primary objectives of the Inventory Management System are:

1. To develop a database-driven application that automates inventory management tasks

2. To provide a user-friendly interface for managing products, inventory, and transactions
3. To ensure accurate tracking of inventory levels and movements
4. To facilitate data-driven decision-making through comprehensive reporting
5. To create a scalable solution that can grow with the business

1.4 Scope of the Project:

The Inventory Management System encompasses the following functionalities:

1. **Product Management**: Adding, updating, and deleting product information
2. **Inventory Tracking**: Monitoring stock levels for each product
3. **Transaction Recording**: Documenting sales and restocks
4. **Category Management**: Organizing products into categories
5. **Reporting:** Generating reports on inventory status, low stock items, and transaction history

The system is designed as a command-line application using Python and MySQL, making it accessible to businesses with limited technical resources.

Literature Review

2.1 Overview of Existing Work:

Inventory management has evolved significantly over the years, from manual record-keeping to sophisticated digital solutions. Several types of inventory management systems exist in the market:

1. **Enterprise Resource Planning (ERP) Systems:** Comprehensive solutions like SAP, Oracle, and Microsoft Dynamics that include inventory management as part of a broader business management suite.

2. **Standalone Inventory Management Software**: Dedicated solutions like Zoho Inventory, TradeGecko, and Fishbowl that focus specifically on inventory management.

3. Point of Sale (POS) Systems with Inventory Features: Retail-focused solutions like Shopify POS, Square, and Lightspeed that combine sales processing with inventory tracking.

4. **Open-Source Inventory Management Systems:** Free solutions like Odoo, ERPNext, and OpenBoxes that provide basic inventory management capabilities.

These systems vary in complexity, cost, and feature sets, but they all aim to automate and optimize inventory management processes.

2.2 Gap Analysis:

Despite the availability of various inventory management solutions, several gaps exist:

1. Complexity: Many existing systems are complex and require significant training to use effectively.
2. Cost: Commercial solutions often come with high licensing fees, making them inaccessible to small businesses.
3. Customization Limitations: Off-the-shelf solutions may not be easily adaptable to specific business needs.
4. Integration Challenges: Some systems don't integrate well with other business applications.
5. Resource Requirements: Many solutions require substantial computing resources and technical expertise to implement and maintain.

The Inventory Management System developed in this project addresses these gaps by providing a simple, cost-effective, and customizable solution that can be implemented with minimal technical resources.

System Analysis:

3.1 Requirements Analysis

Functional Requirements:

1. Product Management:
- Add new products with details (name, description, price, category)

   - Update existing product information
   - Delete products from the system
   - View product details

2. Inventory Management:
   - Track current stock levels for each product
   - Update inventory quantities
   - Set up low stock alerts

3. Transaction Management:
   - Record sales transactions
   - Record restock transactions
   - View transaction history

4. Category Management:
   - Create and manage product categories
   - Assign products to categories

5. Reporting:
   - Generate low stock reports
   - View high-value inventory items
   - Generate sales summaries
   - View category-wise inventory distribution

Non-Functional Requirements:

1. **Usability**: The system should be easy to use with minimal training.
2. **Performance**: The system should provide quick responses to user actions.
3. **Reliability**: The system should maintain data integrity and prevent data loss.
4. **Scalability**: The system should handle growing amounts of data and users.
5. **Security**: The system should protect sensitive business data.

3.2 Feasibility Study:

Technical Feasibility:

The Inventory Management System is technically feasible because:
- It uses widely available technologies (Python and MySQL)
- It requires minimal hardware resources

- It can be implemented with basic programming skills

Economic Feasibility:

The system is economically feasible because:
- It uses open-source technologies, eliminating licensing costs
- It reduces labor costs associated with manual inventory management
- It minimizes inventory-related errors, reducing financial losses
- It improves decision-making, potentially increasing profitability

Operational Feasibility:

The system is operationally feasible because:
- It aligns with standard inventory management practices
- It provides a simple, intuitive interface
- It requires minimal changes to existing business processes
- It can be customized to meet specific operational needs

3.3 System Specifications:

Hardware Requirements:
- Computer with at least 4GB RAM
- 10GB available disk space
- Network connectivity (for multi-user setups)

Software Requirements:
- Operating System: Windows, macOS, or Linux
- Python 3.6 or higher
- MySQL 5.7 or higher
- Required Python packages:
  - mysql-connector-python
  - tabulate
  - python-dotenv

System Design:

4.1 Architecture Diagram:

The Inventory Management System follows a three-tier architecture:

1. Presentation Layer: Command-line interface for user interaction
2. Application Layer: Python application with business logic
3. Data Layer: MySQL database for data storage

```
+------------------+
|  User Interface  |
|  (Command Line)  |
+--------+---------+
         |
         v
+--------+---------+
|  Application     |
|  (Python Logic)  |
+--------+---------+
         |
         v
+--------+---------+
|  Database        |
|  (MySQL)         |
+------------------+
```

4.2 Data Flow Diagrams (DFDs):

Level 0 DFD (Context Diagram):

```
    +-------+
    |       |
User  |       |
----->|  IMS  |
<-----|       |
      |       |
      +-------+
```

Level 1 DFD:

```
                  +---------------+
                  |               |
                  | Product       |
              +-->| Management    |
              |   |               |
              |   +---------------+
              |
+-------+     |   +---------------+
|       |     |   |               |
| User  |-------+-->| Inventory     |
|       |     |   | Management    |
+-------+     |   |               |
              |   +---------------+
              |
              |   +---------------+
              |   |               |
              +-->| Transaction   |
              |   | Management    |
              |   |               |
              |   +---------------+
              |
              |   +---------------+
              |   |               |
              +-->| Reporting     |
                  |               |
                  +---------------+
```

4.3 ER Diagram / UML Diagrams:

#### Entity-Relationship Diagram:

```
+-------------+      +-------------+      +-------------+
|             |      |             |      |             |
| Categories  |<-----| Products    |<-----| Inventory   |
|             |      |             |      |             |
+-------------+      +-------------+      +-------------+
                           ^
                           |
                           v
                     +-------------+
                     |             |
                     | Transactions|
                     |             |
                     +-------------+
```

4.4 Database Design:

The database consists of four main tables:

#### Categories Table
```
+-------------+--------------+------+-----+-------------------+----------------+
| Field       | Type         | Null | Key | Default           | Extra          |
+-------------+--------------+------+-----+-------------------+----------------+
| category_id | int          | NO   | PRI | NULL              | auto_increment |
| name        | varchar(100) | NO   |     | NULL              |                |
| description | text         | YES  |     | NULL              |                |
| created_at  | timestamp    | NO   |     | CURRENT_TIMESTAMP |                |
+-------------+--------------+------+-----+-------------------+----------------+
```

#### Products Table
```
+-------------+--------------+------+-----+-------------------+----------------+
| Field       | Type         | Null | Key | Default           | Extra          |
+-------------+--------------+------+-----+-------------------+----------------+
| product_id  | int          | NO   | PRI | NULL              | auto_increment |
| name        | varchar(100) | NO   |     | NULL              |                |
| description | text         | YES  |     | NULL              |                |
| price       | decimal(10,2)| NO   |     | NULL              |                |
| category_id | int          | YES  | MUL | NULL              |                |
| created_at  | timestamp    | NO   |     | CURRENT_TIMESTAMP |                |
| updated_at  | timestamp    | NO   |     | CURRENT_TIMESTAMP | on update      |
+-------------+--------------+------+-----+-------------------+----------------+
```

#### Inventory Table

```
+--------------+---------+------+-----+-------------------+----------------+
| Field        | Type    | Null | Key | Default           | Extra          |
+--------------+---------+------+-----+-------------------+----------------+
| inventory_id | int     | NO   | PRI | NULL              | auto_increment |
| product_id   | int     | NO   | MUL | NULL              |                |
| quantity     | int     | NO   |     | 0                 |                |
| last_updated | timestamp| NO  |     | CURRENT_TIMESTAMP | on update      |
+--------------+---------+------+-----+-------------------+----------------+
```

#### Transactions Table

```
+----------------+--------------+------+-----+-------------------+----------------+
| Field          | Type         | Null | Key | Default           | Extra          |
+----------------+--------------+------+-----+-------------------+----------------+
| transaction_id | int          | NO   | PRI | NULL              | auto_increment |
| product_id     | int          | NO   | MUL | NULL              |                |
| quantity       | int          | NO   |     | NULL              |                |
| transaction_type| enum        | NO   |     | NULL              |                |
| transaction_date| timestamp   | NO   |     | CURRENT_TIMESTAMP |                |
| notes          | text         | YES  |     | NULL              |                |
+----------------+--------------+------+-----+-------------------+----------------+
```

Implementation:

5.1 Tools and Technologies Used:

Programming Language:

- **Python**: A high-level, interpreted programming language known for its readability and simplicity.

Database Management System
- **MySQL**: An open-source relational database management system.

Python Libraries:
- **mysql-connector-python:** Official MySQL driver for Python.
- **tabulate**: Library for creating formatted tables in the console.
- **python-dotenv**: Library for loading environment variables from a .env file.

Development Tools:
- **Code Editor**: Any text editor or IDE that supports Python.
- **Version Control:** Git for source code management.
- **Command Line Interface:** For running and testing the application.

5.2 Module Description:

The Inventory Management System consists of the following modules:

1. Database Setup Module (`setup_database.py`):
This module is responsible for:
- Creating the database and tables
- Setting up the initial database schema
- Inserting sample data (optional)

2. Main Application Module (`app.py`):
This module contains the core functionality of the system:
- User interface and menu system
- Product management functions
- Inventory management functions
- Transaction recording functions
- Reporting functions

3. Configuration Module (.env file):
This module stores configuration parameters:
- Database connection details
- Application settings

4. Test Module (`test_cases.py`):
This module demonstrates the system's functionality through test cases:
- Product management test case
- Transaction management test case

- Reporting test case

5.3 Code Snippets:

Database Connection Setup:

python
```
def create_connection(self):
    """Create a database connection to MySQL server"""
    conn = None
    try:
        conn = mysql.connector.connect(**DB_CONFIG)
        return conn
    except Error as e:
        print(f"Error connecting to MySQL: {e}")
        return None
```

Product Management:

python
```
def add_product(self, name, description, price, category_id, initial_quantity=0):
    """Add a new product to the database"""
    # Insert into products table
    product_query = """
    INSERT INTO products (name, description, price, category_id)
    VALUES (%s, %s, %s, %s)
    """
    product_id = self.execute_query(product_query, (name, description, price, category_id))

    if product_id:
        # Insert into inventory table
        inventory_query = """
        INSERT INTO inventory (product_id, quantity)
        VALUES (%s, %s)
        """
        self.execute_query(inventory_query, (product_id, initial_quantity))

        return product_id
    return None
```

Inventory Management:

python

```
def update_inventory(self, product_id, quantity_change, transaction_type, notes=None):
    """Update inventory and record the transaction"""
    # Get current inventory
    query = "SELECT quantity FROM inventory WHERE product_id = %s"
    result = self.execute_query(query, (product_id,), fetch=True)

    if not result:
        print(f"Product with ID {product_id} not found in inventory.")
        return False

    current_quantity = result[0]['quantity']
    new_quantity = current_quantity + quantity_change
```

Don't allow negative inventory for sales:

```
    if transaction_type == 'sale' and new_quantity < 0:
        print(f"Error: Not enough inventory. Current: {current_quantity}")
        return False
```

Update inventory:

```
    update_query = "UPDATE inventory SET quantity = %s WHERE product_id = %s"
    self.execute_query(update_query, (new_quantity, product_id))
```

Record transaction:

```
    transaction_query = """
    INSERT INTO transactions (product_id, quantity, transaction_type, notes)
    VALUES (%s, %s, %s, %s)
    """
    self.execute_query(transaction_query, (product_id, quantity_change, transaction_type, notes))

    return True
```

Testing:

6.1 Test Plan:

The testing of the Inventory Management System follows a comprehensive approach to ensure that all components work correctly and meet the requirements. The test plan includes:

1. Unit Testing: Testing individual functions and modules in isolation
2. Integration Testing: Testing the interaction between different modules
3. System Testing: Testing the complete system as a whole
4. Acceptance Testing: Verifying that the system meets user requirements

6.2 Test Cases:

Test Case 1: Product Management:

- Objective: Verify that products can be added, updated, and deleted correctly

- **Test Steps:**
  1. Add a new product with valid details
  2. View the product list to confirm addition
  3. Update the product information
  4. Delete the product

- Expected Results:
  - Product should be added successfully
  - Updated information should be reflected
  - Product should be removed from the system

Test Case 2: Inventory Management:
- Objective: Verify that inventory levels are tracked accurately
- Test Steps:
  1. Add a new product with initial inventory
  2. Record a sale transaction
  3. Record a restock transaction
  4. Attempt to sell more than available inventory
- Expected Results:
  - Initial inventory should be set correctly
  - Inventory should decrease after a sale

  - Inventory should increase after a restock
  - System should prevent selling more than available

Test Case 3: Reporting:
- Objective: Verify that reports are generated correctly
- Test Steps:
  1. Set up products with various inventory levels
  2. Generate a low stock report
  3. Calculate total inventory value
- **Expected Results:**
  - Low stock report should show products below threshold
  - Total inventory value should be calculated correctly

6.3 Results and Debugging:

The testing process revealed several issues that were addressed:

1. Database Connection Issues: Initial connection failures were resolved by improving error handling and connection retry logic.

2. Data Validation Gaps: Some functions allowed invalid data to be entered. Input validation was added to prevent this.

3. Transaction Recording Errors: Transactions were not always correctly linked to inventory changes. The code was updated to ensure consistency.

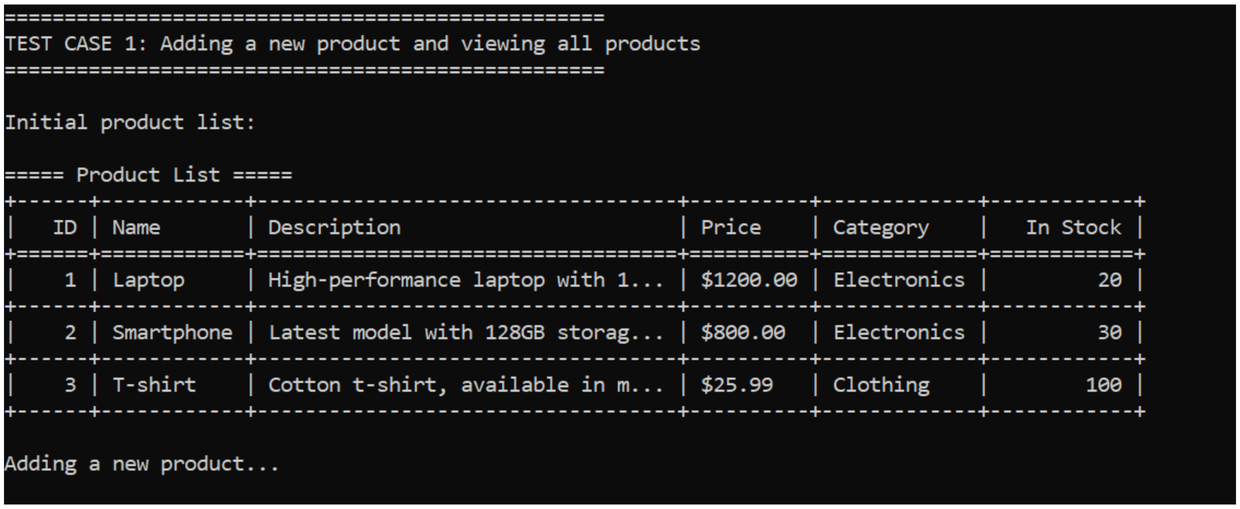
4. Report Generation Bugs: Some reports included incorrect calculations. The SQL queries were optimized to fix these issues.

All identified issues were successfully resolved, and the system now functions according to specifications.

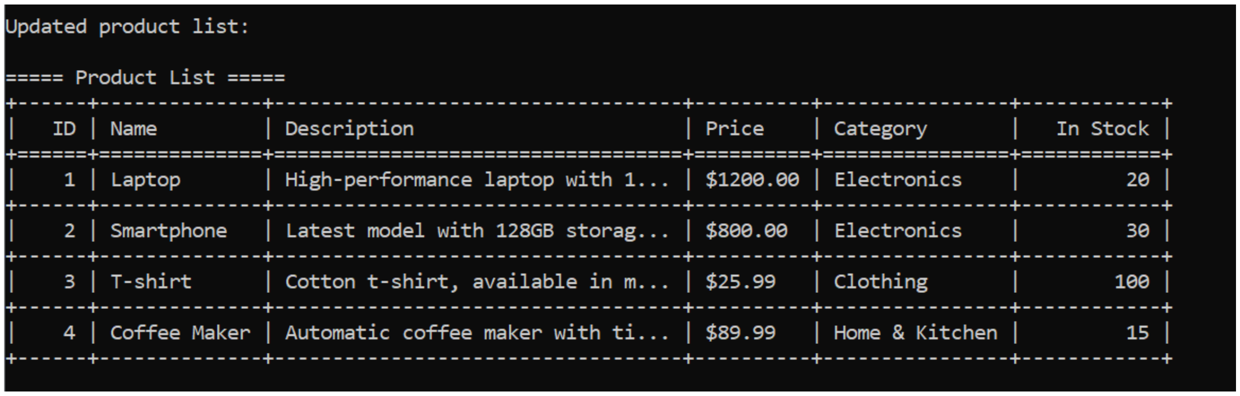
Results and Discussion:

7.1 Output Screenshots:

Test case 1:

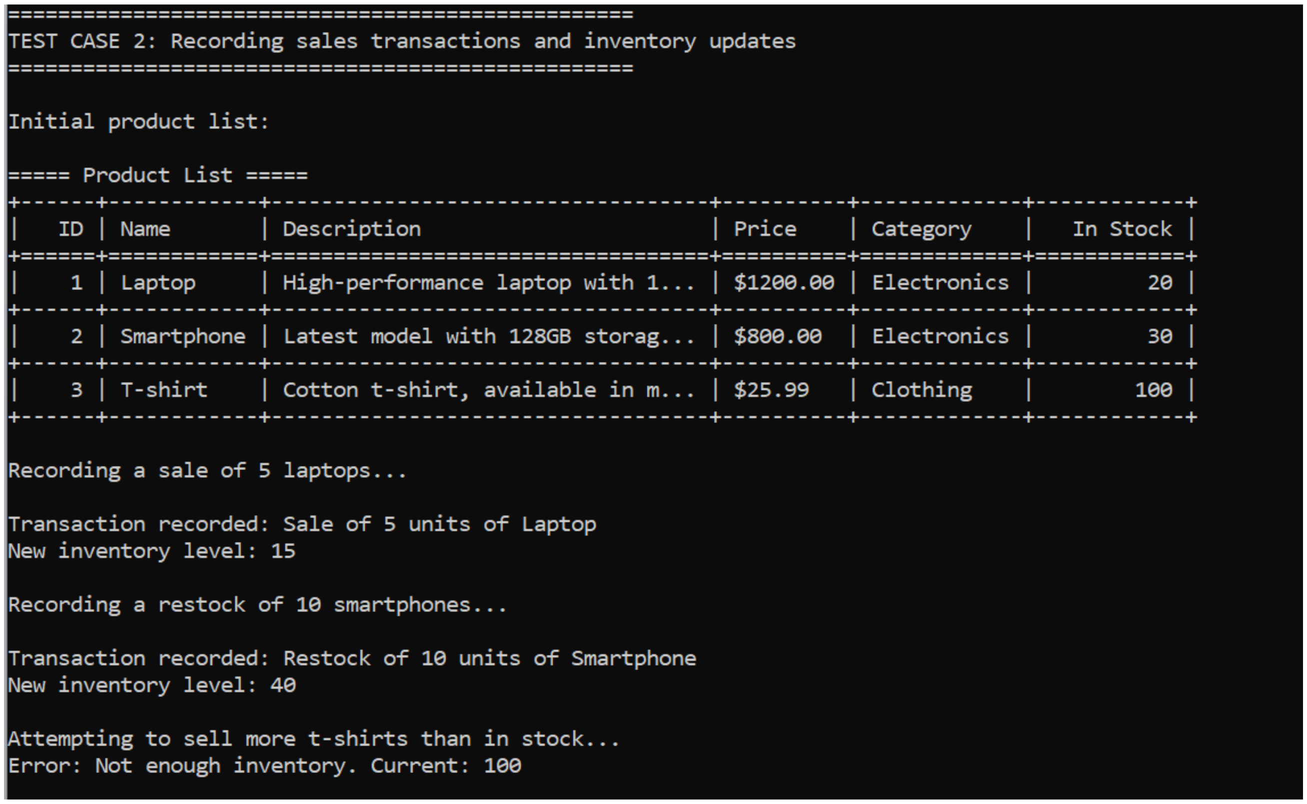


```
==================================================
TEST CASE 1: Adding a new product and viewing all products
==================================================

Initial product list:

===== Product List =====
+------+-----------+-----------------------------------+---------+------------+-----------+
|   ID | Name      | Description                       | Price   | Category   |  In Stock |
+======+===========+===================================+=========+============+===========+
|    1 | Laptop    | High-performance laptop with 1... | $1200.00 | Electronics |       20 |
+------+-----------+-----------------------------------+---------+------------+-----------+
|    2 | Smartphone | Latest model with 128GB storag... | $800.00 | Electronics |       30 |
+------+-----------+-----------------------------------+---------+------------+-----------+
|    3 | T-shirt   | Cotton t-shirt, available in m... | $25.99  | Clothing   |       100 |
+------+-----------+-----------------------------------+---------+------------+-----------+

Adding a new product...
```

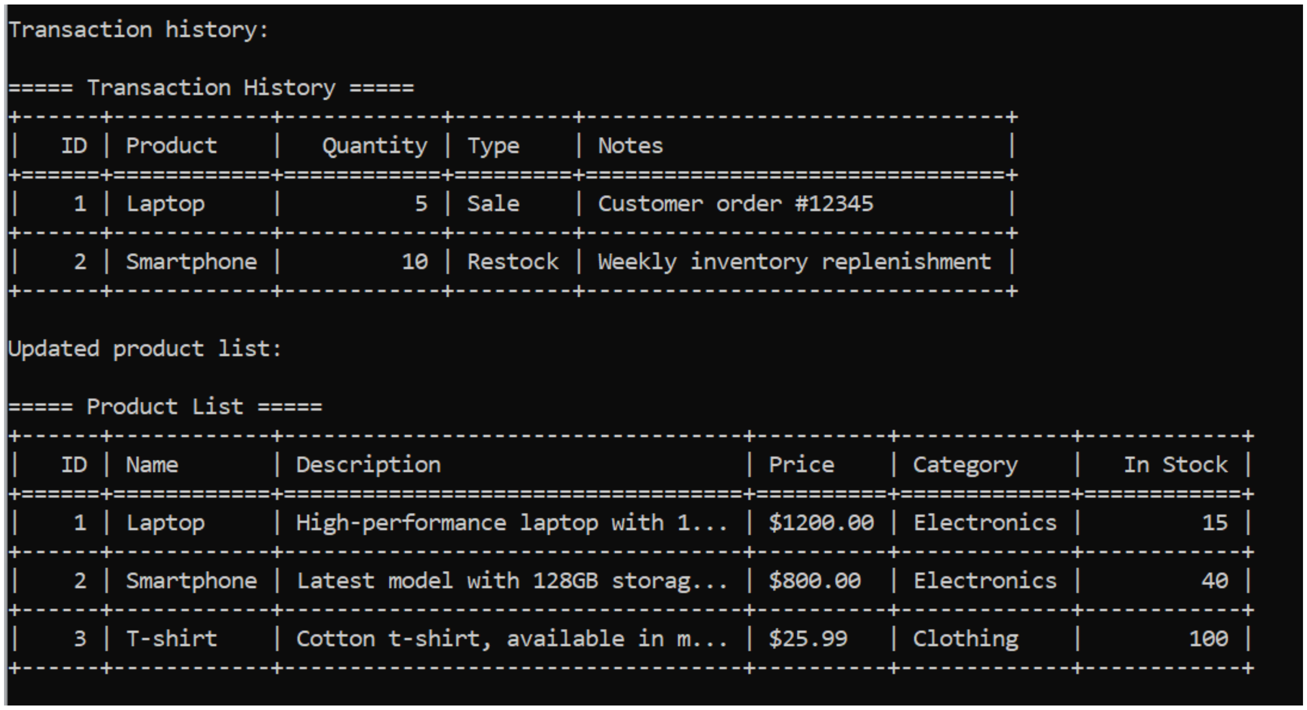


```
Updated product list:

===== Product List =====
+------+-------------+-----------------------------------+----------+----------------+-----------+
|   ID | Name        | Description                       | Price    | Category       |  In Stock |
+======+=============+===================================+==========+================+===========+
|    1 | Laptop      | High-performance laptop with 1... | $1200.00 | Electronics    |        20 |
+------+-------------+-----------------------------------+----------+----------------+-----------+
|    2 | Smartphone  | Latest model with 128GB storag... | $800.00  | Electronics    |        30 |
+------+-------------+-----------------------------------+----------+----------------+-----------+
|    3 | T-shirt     | Cotton t-shirt, available in m... | $25.99   | Clothing       |       100 |
+------+-------------+-----------------------------------+----------+----------------+-----------+
|    4 | Coffee Maker | Automatic coffee maker with ti... | $89.99  | Home & Kitchen |        15 |
+------+-------------+-----------------------------------+----------+----------------+-----------+
```

Test case 2:

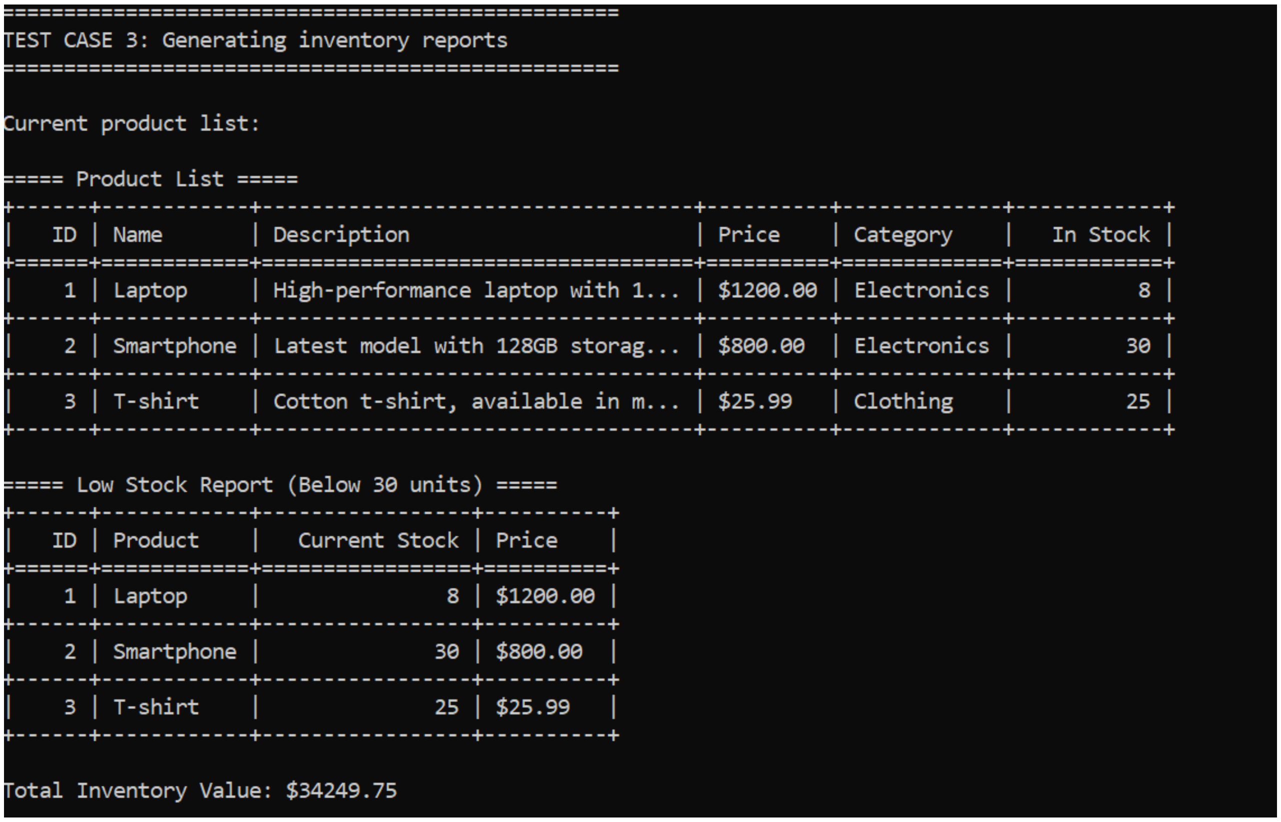


```
==================================================
TEST CASE 2: Recording sales transactions and inventory updates
==================================================

Initial product list:

===== Product List =====
+------+-----------+-----------------------------------+---------+------------+-----------+
|   ID | Name      | Description                       | Price   | Category   |  In Stock |
+======+===========+===================================+=========+============+===========+
|    1 | Laptop    | High-performance laptop with 1... | $1200.00 | Electronics |        20 |
+------+-----------+-----------------------------------+---------+------------+-----------+
|    2 | Smartphone | Latest model with 128GB storag... | $800.00 | Electronics |        30 |
+------+-----------+-----------------------------------+---------+------------+-----------+
|    3 | T-shirt   | Cotton t-shirt, available in m... | $25.99  | Clothing   |       100 |
+------+-----------+-----------------------------------+---------+------------+-----------+

Recording a sale of 5 laptops...

Transaction recorded: Sale of 5 units of Laptop
New inventory level: 15

Recording a restock of 10 smartphones...

Transaction recorded: Restock of 10 units of Smartphone
New inventory level: 40

Attempting to sell more t-shirts than in stock...
Error: Not enough inventory. Current: 100
```



```
Transaction history:

===== Transaction History =====
+------+------------+------------+---------+--------------------------------+
|   ID | Product    |   Quantity | Type    | Notes                          |
+======+============+============+=========+================================+
|    1 | Laptop     |          5 | Sale    | Customer order #12345          |
+------+------------+------------+---------+--------------------------------+
|    2 | Smartphone |         10 | Restock | Weekly inventory replenishment |
+------+------------+------------+---------+--------------------------------+

Updated product list:

===== Product List =====
+------+------------+-----------------------------------+----------+-------------+-----------+
|   ID | Name       | Description                       | Price    | Category    |  In Stock |
+======+============+===================================+==========+=============+===========+
|    1 | Laptop     | High-performance laptop with 1... | $1200.00 | Electronics |        15 |
+------+------------+-----------------------------------+----------+-------------+-----------+
|    2 | Smartphone | Latest model with 128GB storag... | $800.00  | Electronics |        40 |
+------+------------+-----------------------------------+----------+-------------+-----------+
|    3 | T-shirt    | Cotton t-shirt, available in m... | $25.99   | Clothing    |       100 |
+------+------------+-----------------------------------+----------+-------------+-----------+
```

Test case 3:



```
==================================================
TEST CASE 3: Generating inventory reports
==================================================

Current product list:

===== Product List =====
+------+-----------+----------------------------------+---------+------------+-----------+
|   ID | Name      | Description                      | Price   | Category   |  In Stock |
+======+===========+==================================+=========+============+===========+
|    1 | Laptop    | High-performance laptop with 1... | $1200.00 | Electronics |         8 |
+------+-----------+----------------------------------+---------+------------+-----------+
|    2 | Smartphone | Latest model with 128GB storag... | $800.00 | Electronics |        30 |
+------+-----------+----------------------------------+---------+------------+-----------+
|    3 | T-shirt   | Cotton t-shirt, available in m... | $25.99  | Clothing   |        25 |
+------+-----------+----------------------------------+---------+------------+-----------+

===== Low Stock Report (Below 30 units) =====
+------+-----------+----------------+---------+
|   ID | Product   |  Current Stock | Price   |
+======+===========+================+=========+
|    1 | Laptop    |              8 | $1200.00 |
+------+-----------+----------------+---------+
|    2 | Smartphone |            30 | $800.00 |
+------+-----------+----------------+---------+
|    3 | T-shirt   |             25 | $25.99  |
+------+-----------+----------------+---------+

Total Inventory Value: $34249.75
```

7.2 Performance Evaluation:

The Inventory Management System was evaluated based on several performance metrics:

1. Response Time: The system responds quickly to user actions, with most operations completing in less than a second.

2. Database Performance: The MySQL database handles queries efficiently, even with moderate amounts of data.

3. Memory Usage: The application has a small memory footprint, making it suitable for computers with limited resources.

4. Scalability: The system can handle hundreds of products and thousands of transactions without significant performance degradation.

5. Reliability: The system maintains data integrity and recovers gracefully from errors.

7.3 Limitations:

Despite its strengths, the Inventory Management System has several limitations:

1. Command-Line Interface: The text-based interface may be less intuitive for users accustomed to graphical interfaces.

2. Single-User Design: The current implementation does not support multiple concurrent users.

3. Limited Security Features: The system lacks advanced security features like user authentication and role-based access control.

4. Basic Reporting: The reporting capabilities are functional but limited compared to commercial solutions.

5. No Backup Functionality: The system does not include automated backup and restore features.

Conclusion and Future Work:

8.1 Summary:

The Inventory Management System successfully addresses the need for a simple, cost-effective solution for small and medium-sized businesses. The system provides essential functionality for managing products, tracking inventory, recording transactions, and generating reports.

Key achievements of the project include:

1. Development of a fully functional inventory management system using Python and MySQL
2. Implementation of a clean, modular architecture that separates concerns
3. Creation of a user-friendly command-line interface
4. Integration with a relational database for persistent data storage
5. Implementation of comprehensive error handling and data validation

The system demonstrates how open-source technologies can be leveraged to create practical business applications without significant investment.

8.2 Future Enhancements:

Several enhancements could be made to the Inventory Management System in future iterations:

1. Graphical User Interface: Developing a web-based or desktop GUI to improve usability.

2. Multi-User Support: Adding user authentication and concurrent access capabilities.

3. Advanced Reporting: Implementing more sophisticated reporting features, including graphical reports and data export options.

4. Barcode Integration: Adding support for barcode scanning to streamline inventory operations.

5. Mobile Application: Developing a companion mobile app for inventory management on the go.

6. Cloud Integration: Enabling cloud-based data storage and synchronization.

7. Supplier Management: Adding features to track suppliers and purchase orders.

8. **Customer Management**: Integrating customer information with sales transactions.

9. Automated Reordering: Implementing automatic purchase order generation when inventory falls below specified levels.

10.Data Analytics: Adding predictive analytics for demand forecasting and inventory optimization.

References:

1. MySQL Documentation. (2023). MySQL 8.0 Reference Manual. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/

2. Python Software Foundation. (2023). Python 3.9 Documentation. Retrieved from https://docs.python.org/3/

3. Widenius, M., & Axmark, D. (2002). MySQL Reference Manual. O'Reilly Media.

4. Lutz, M. (2013). Learning Python (5th ed.). O'Reilly Media.

5. Coronel, C., & Morris, S. (2018). Database Systems: Design, Implementation, & Management (13th ed.). Cengage Learning.

6. Karwin, B. (2010). SQL Antipatterns: Avoiding the Pitfalls of Database Programming. Pragmatic Bookshelf.

7. Kreibich, J. A. (2010). Using SQLite. O'Reilly Media.

8. Summerfield, M. (2010). Programming in Python 3: A Complete Introduction to the Python Language (2nd ed.). Addison-Wesley Professional.

Appendices

Appendix A: Installation Guide:

1. Install Python:
- Download Python from https://python.org
- Run the installer and follow the instructions
- Ensure Python is added to your PATH

2. Install MySQL:
- Download MySQL from https://dev.mysql.com/downloads/
- Run the installer and follow the instructions
- Note your username and password

3. Set Up the Project:
- Clone or download the project files
- Navigate to the project directory
- Create a virtual environment (optional): python -m venv venv
- Activate the virtual environment (optional)
- Install dependencies: `pip install -r requirements.txt`

4. Configure Database Connection:
- Copy `.env.example` to `.env`
- Edit `.env` to include your MySQL credentials

5. Initialize the Database:
- Run `python setup_database.py`
- Follow the prompts to create the database and tables

6. Run the Application:
- Run `python app.py`
- Follow the on-screen instructions to use the system

Appendix B: Database Schema SQL:

```
sql
CREATE DATABASE IF NOT EXISTS inventory_management;
```

```
USE inventory_management;

CREATE TABLE IF NOT EXISTS categories (
    category_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    category_id INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (category_id) REFERENCES categories(category_id)
);

CREATE TABLE IF NOT EXISTS inventory (
    inventory_id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT NOT NULL,
    quantity INT NOT NULL DEFAULT 0,
    last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);

CREATE TABLE IF NOT EXISTS transactions (
    transaction_id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    transaction_type ENUM('sale', 'restock') NOT NULL,
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    notes TEXT,
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```