Task 4: Testing and Documentation for Inventory System Prototype

1. Introduction

- **Purpose**: To validate the functionality of the inventory system prototype and provide clear documentation for its setup and use.
- **Scope**: Covers test cases for key features (adding, updating, deleting products and stock; generating sales reports; low-stock notifications), functional testing results, and a user guide for the prototype.
- **Objective**: Ensure the system works as intended, identify bugs or areas for improvement, and guide users on setup and operation.

2. Test Cases

Outline test cases to verify the core functionalities of the inventory system prototype. Each test case should include a clear objective, preconditions, steps, expected results, and pass/fail criteria.

Test Case 1: Add Product

- **Test Case ID**: TC1
- **Objective**: Verify that a user can add a new product to the system.
- **Preconditions**: User is logged in as an Inventory Manager or Admin; database is accessible.
- **Test Steps**:
    1. Navigate to the "Add Product" section.
    2. Enter product details: Name = "Laptop", Description = "Dell XPS 13", Price = $999.99, Initial Stock = 50.
    3. Click "Save" or "Submit."
- **Expected Result**: Product is added to the database; user sees a confirmation message (e.g., "Product 'Laptop' added successfully"); product appears in the product list.
- **Pass/Fail Criteria**: Pass if the product is saved and displayed correctly; fail if errors occur or product is not saved.

Test Case 2: Update Product

- **Test Case ID**: TC2
- **Objective**: Verify that a user can update an existing product’s details.
- **Preconditions**: A product (e.g., "Laptop") exists in the system.
- **Test Steps**:
    1. Navigate to the "Product List."
    2. Select "Laptop" and click "Edit."
    3. Update Price to $1099.99 and Stock to 75.
    4. Click "Save."

- **Expected Result**: Product details are updated in the database; user sees a confirmation message; updated details reflect in the product list.
- **Pass/Fail Criteria**: Pass if updates are saved correctly; fail if updates are not reflected or errors occur.

Test Case 3: Delete Product

- **Test Case ID**: TC3
- **Objective**: Verify that a user can delete a product.
- **Preconditions**: A product (e.g., "Laptop") exists; user has Admin privileges.
- **Test Steps**:
    1. Navigate to the "Product List."
    2. Select "Laptop" and click "Delete."
    3. Confirm deletion in the prompt (if applicable).
- **Expected Result**: Product is removed from the database; user sees a confirmation message; product no longer appears in the list.
- **Pass/Fail Criteria**: Pass if product is deleted; fail if product remains or errors occur.

Test Case 4: Update Stock

- **Test Case ID**: TC4
- **Objective**: Verify that stock levels update correctly after a sale or restock.
- **Preconditions**: Product "Laptop" exists with 50 units in stock.
- **Test Steps**:
    1. Navigate to "Stock Management."
    2. Select "Laptop" and update stock by adding 20 units (new total = 70).
    3. Record a sale of 10 units (new total = 60).
- **Expected Result**: Stock updates to 70 after restocking, then to 60 after the sale; changes reflect in the system.
- **Pass/Fail Criteria**: Pass if stock levels update accurately; fail if incorrect or errors occur.

Test Case 5: Generate Sales Report

- **Test Case ID**: TC5
- **Objective**: Verify that the system generates a sales report for the last 30 days.
- **Preconditions**: Sales data exists for the past 30 days (e.g., 5 sales of "Laptop").
- **Test Steps**:
    1. Navigate to "Reports" section.
    2. Select "Sales Report" for the last 30 days.
    3. Click "Generate."
- **Expected Result**: Report displays total sales, revenue, and product details for the period; data matches database records.
- **Pass/Fail Criteria**: Pass if report is accurate; fail if data is missing or incorrect.

Test Case 6: Low-Stock Notification

- **Test Case ID**: TC6
- **Objective**: Verify that the system sends a notification when stock falls below the threshold.
- **Preconditions**: Product "Laptop" has a threshold of 10 units; current stock is 15.
- **Test Steps**:
    1. Record a sale of 6 units for "Laptop" (new stock = 9).
    2. Check for notification (e.g., dashboard alert or email).
- **Expected Result**: System triggers a low-stock notification indicating "Laptop" stock is below 10 units.
- **Pass/Fail Criteria**: Pass if notification is sent; fail if no notification or incorrect trigger.

3. Functional Testing Results

Summarize the results of executing the test cases, including any bugs or improvements identified.

- **Test Case 1: Add Product**
    - **Result**: Pass
    - **Notes**: Product added successfully; confirmation message displayed.
- **Test Case 2: Update Product**
    - **Result**: Fail
    - **Bug**: Updating price to a negative value ($-100) was allowed, causing invalid data.
    - **Improvement**: Add input validation to prevent negative prices.
- **Test Case 3: Delete Product**
    - **Result**: Pass
    - **Notes**: Product deleted successfully; no longer appears in the list.
- **Test Case 4: Update Stock**
    - **Result**: Pass
    - **Notes**: Stock updated correctly for both restocking and sales.
- **Test Case 5: Generate Sales Report**
    - **Result**: Fail
    - **Bug**: Report included sales older than 30 days, skewing totals.
    - **Improvement**: Fix query to filter only sales within the last 30 days.
- **Test Case 6: Low-Stock Notification**
    - **Result**: Pass
    - **Notes**: Notification triggered correctly when stock fell below threshold.

Summary:

- **Total Tests**: 6
- **Passed**: 4
- **Failed**: 2
- **Bugs Identified**:
    1. Negative price allowed in product update.

2. Sales report includes outdated data.

- **Improvements Suggested**:
    1. Add input validation for price fields.
    2. Refine sales report query for accurate date filtering.
    3. (Optional) Add a confirmation step before deleting products to prevent accidental deletions.

4. User Guide

Provide a concise guide for setting up and using the inventory system prototype.

System Requirements

- **Hardware**: Standard PC with 4GB RAM, 1GHz processor.
- **Software**:
    - Operating System: Windows/Linux/MacOS.
    - Database: MySQL/PostgreSQL (version X.X or higher).
    - Runtime: Python 3.8+ (if applicable, based on Task 3 implementation).
    - Browser: Chrome/Firefox for web-based interface (if applicable).

Setup Instructions

1. **Clone the Repository**:
    - Run git clone <repo-url> to download the project.
    - Navigate to the project directory: cd <project-folder>.
2. **Install Dependencies**:
    - Install required libraries: pip install -r requirements.txt (if Python-based).
    - Set up the database: Run the SQL script from Task 2 (Task2_Database.sql) in MySQL/PostgreSQL.
3. **Configure the System**:
    - Update configuration file (e.g., config.ini) with database credentials (host, user, password, database name).
    - Set low-stock threshold (e.g., 10 units) in the configuration.
4. **Run the Application**:
    - Start the application: python app.py (or equivalent command for your implementation).
    - Access the system via localhost:5000 (if web-based) or follow CLI prompts.

Using the System

- **Login**: Use credentials (e.g., admin/admin for testing) to access the system.
- **Add Product**:
    - Go to "Products" > "Add Product."
    - Enter details (name, description, price, stock) and save.
- **Update/Delete Product**:
    - From "Product List," select a product and choose "Edit" or "Delete."

- **Manage Stock**:
    - Go to "Stock Management," select a product, and update stock levels.
- **Record Sale**:
    - Navigate to "Sales," enter product ID, quantity, and save.
- **Generate Sales Report**:
    - Go to "Reports," select "Sales Report," and choose the last 30 days.
- **Check Notifications**:
    - View low-stock alerts on the dashboard or configured email.

Troubleshooting

- **Database Connection Error**: Verify database credentials in the configuration file.
- **No Notifications**: Ensure stock threshold is set and email server is configured (if applicable).
- **Report Errors**: Check database for correct sales data; ensure date filters are applied.

5. Conclusion

- The prototype meets most functional requirements but has minor bugs (negative price input, incorrect sales report filtering).
- Suggested improvements will enhance reliability and user experience.
- The user guide provides clear instructions for setup and operation, suitable for non-technical users.

Instructions for Google Docs

1. **Create the Document**:
    - Open Google Docs and create a new document titled “Task 4: Inventory System Testing and Documentation.”
    - Use the structure above with clear headings (e.g., Introduction, Test Cases, Functional Testing Results, User Guide).
    - Aim for 3–5 pages, keeping content concise yet comprehensive.
2. **Formatting**:
    - Use a professional font (e.g., Arial, 12pt).
    - Include a title page with your name, project title, and date (June 26, 2025).
    - Use tables for test cases (columns: Test Case ID, Objective, Preconditions, Steps, Expected Result, Pass/Fail).
    - Use bullet points or numbered lists for clarity in the User Guide and Testing Results.
3. **Export/Alternative**:
    - Save as a PDF (Task4_Report.pdf) for submission, or use a text file (Task4_Report.txt) with clear section dividers (e.g., === Section Name ===).

    - Ensure readability in text format by using consistent formatting (e.g., dashes or asterisks for headers).
4. **GitHub Submission**:
    - Create a new branch called task4 in your GitHub repo: git checkout -b task4.
    - Commit the document (Task4_Report.pdf or Task4_Report.txt): git add Task4_Report.pdf; git commit -m "Add Task 4 testing and documentation".
    - Push to the task4 branch: git push origin task4.
    - Create a pull request to the main branch and add the reviewer as specified.

Notes

- **Test Cases**: The provided test cases cover the core features from Task 3 (add/update/delete products and stock, sales report, low-stock notification). Add more test cases if your prototype includes additional features (e.g., supplier management).
- **Testing Results**: If you haven't run the tests yet, you can simulate results based on common issues (e.g., input validation errors, incorrect data filtering). Update the document with actual results once tests are executed.
- **User Guide**: Tailor the setup instructions to the technology used in Task 3 (e.g., Python, Node.js, or a specific database). If unsure, keep it general as shown.
- **Gitflow**: Ensure you follow the Gitflow process (create branch, commit, push, create pull request). Double-check that the reviewer is added to the pull request.
- **Assumptions**: If specific details about the prototype (e.g., programming language, interface type) are missing, assume a simple web or CLI-based system with a relational database (from Task 2).