Product Management System Documentation

*A Python-based Application with SQL Server Integration*

Submitted by: Santosh Kumar

Roll Number:   
Department :  
Bachelor Of Computer And Application

Submitted to:

Date: July 17, 2025

**Contents**

1. [Abstract 3](#_TOC_250025)
2. [Introduction 3](#_TOC_250024)
   1. [Project Objectives 3](#_TOC_250023)
   2. [Scope 3](#_TOC_250022)
   3. [Motivation 3](#_TOC_250021)
3. [System Requirements 3](#_TOC_250020)
   1. [Hardware Requirements 4](#_TOC_250019)
   2. [Software Requirements 4](#_TOC_250018)
   3. [Installation Steps 4](#_TOC_250017)
4. [Database Schema 4](#_TOC_250016)
   1. [Field Descriptions 4](#_TOC_250015)
   2. [ER Diagram 5](#_TOC_250014)
5. [System Design 5](#_TOC_250013)
   1. [System Architecture 5](#_TOC_250012)
   2. [Flowchart 5](#_TOC_250011)
   3. [Key Functions 6](#_TOC_250010)
6. [Code Implementation 6](#_TOC_250009)
   1. [Code Components 9](#_TOC_250008)
7. [Example Usage 9](#_TOC_250007)
   1. Scenario 1: Adding a Product 9
   2. Scenario 2: Viewing All Products 10
   3. Scenario 3: Searching for a Product 10
   4. Scenario 4: Updating a Product 10
   5. Scenario 5: Deleting a Product 10
8. [Detailed Example Workﬂow with Sample Product 11](#_TOC_250006)
   1. Step 1: Adding a Smartphone 11
   2. Step 2: Viewing All Products 12
   3. Step 3: Searching for the Smartphone 12
   4. Step 4: Updating the Smartphone 13
   5. Step 5: Deleting the Smartphone 13
   6. Step 6: Verifying Deletion 14
9. [Testing and Validation 15](#_TOC_250005)
   1. [Error Handling 15](#_TOC_250004)
10. [Future Enhancements 15](#_TOC_250003)
11. [Conclusion 15](#_TOC_250002)
12. [Glossary 16](#_TOC_250001)

# Abstract

The Product Management System is a console-based Python application designed to manage product information stored in a Microsoft SQL Server database. Utilizing the pyodbc library, the system supports CRUD (Create, Read, Update, Delete) operations through a user-friendly menu-driven interface. This documentation provides a comprehensive overview of the system, including its design, implementation, database schema, testing, and potential enhancements. The project demonstrates robust database connectivity, modular programming, and error han- dling, making it suitable for small to medium-scale inventory management. Diagrams, code explanations, and sample outputs are included to provide a comprehensive understanding of the system.

# Introduction

The Product Management System is developed to facilitate efficient management of product data, including product names, categories, prices, and quantities. The system integrates Python with SQL Server, leveraging the pyodbc library for database connectivity. This project is designed for educational purposes, showcasing fundamental concepts of database management, modular programming, and user interaction in a console environment.

## Project Objectives

* + - To develop a robust system for managing product data with CRUD operations.
    - To demonstrate seamless integration of Python with SQL Server.
    - To implement a user-friendly interface for inventory management.
    - To ensure error handling and data validation for reliable operation.

## Scope

The system focuses on core inventory management functionalities, including adding, viewing, searching, updating, and deleting products. It operates as a console-based application but can be extended to include a graphical user interface (GUI), user authentication, or advanced reporting features.

## Motivation

Inventory management is a critical aspect of many businesses, requiring efficient and reliable systems to track product data. This project serves as a practical implementation of database- driven applications, providing hands-on experience with Python and SQL Server integration.

# System Requirements

To run the Product Management System, the following hardware and software requirements must be met:

## Hardware Requirements

* + - Processor: 1 GHz or faster
    - RAM: 4 GB or higher
    - Storage: 500 MB free disk space

## Software Requirements

* + - Operating System: Windows 10 or later
    - Python: Version 3.8 or higher
    - SQL Server: Microsoft SQL Server 2017 or later
    - Python Libraries: pyodbc
    - ODBC Driver: ODBC Driver 17 for SQL Server

## Installation Steps

1. Install Python from <https://www.python.org.>
2. Install Microsoft SQL Server and create a database named AI\_DataBase.
3. Install the ODBC Driver 17 for SQL Server.
4. Install the pyodbc library using: pip install pyodbc.
5. Execute the provided SQL schema to create the Products table.

# Database Schema

The system uses a single table, Products, to store product information. The schema is de- fined as follows:

**CREATE TABLE** Products (

productId **INT PRIMARY KEY IDENTITY**(1,1),

productName **VARCHAR**(100), category **VARCHAR**(50), price **DECIMAL**(10,2),

quantity **INT**

);

1

2

3

4

5

6

7

## Field Descriptions

* + - productId: Unique identifier for each product, auto-incremented by SQL Server.
    - productName: Name of the product, up to 100 characters.
    - category: Product category, up to 50 characters.
    - price: Product price, stored as a decimal with two places.
    - quantity: Available stock quantity, stored as an integer.

## ER Diagram

The Entity-Relationship (ER) diagram for the Products table is shown below, created using TikZ.

productId

productName

category

price

quantity

PK

Products

# System Design

The Product Management System follows a modular design with separate functions for each operation. The system architecture includes a Python-based frontend (console interface) and a SQL Server backend, connected via the pyodbc library.

## System Architecture

The system consists of:

* + - **Frontend**: A console-based menu-driven interface for user interaction.
    - **Backend**: SQL Server database (AI\_DataBase) for data storage.
    - **Connectivity**: pyodbc library for database operations.

## Flowchart

The flowchart below illustrates the menu-driven workflow of the system, created using TikZ.



Start

Display Menu

User Choice

1

2

3

4

5

Add ProducVtiew ProduSctesarch ProdUucptdate ProdDucetlete Product 6

Exit

## Key Functions

* + - get\_connection(): Establishes a connection to the SQL Server database.
    - add\_product(): Inserts a new product into the Products table.
    - view\_all\_products(): Retrieves and displays all products.
    - search\_product\_by\_id(): Searches for a product by its ID.
    - update\_product(): Updates details of an existing product.
    - delete\_product(): Deletes a product by its ID.
    - main(): Implements the menu-driven interface.

# Code Implementation

The complete Python code for the Product Management System is provided below, with de- tailed explanations of each component.

**import** pyodbc

# Establishes connection to SQL Server database

**def** get\_connection():

**try**:

conn = pyodbc.connect(’DRIVER={ODBC Driver 17 for SQL Server

};SERVER=localhost;DATABASE=AI\_DataBase; Trusted\_Connection=yes’)

**return** conn

**except** Exception as e:

**print**(" Connection Error:", e)

**return** None

1

2

3

4

5

6

7

8

9

10

11

12 # Adds a new product to the database

13 **def** add\_product():

14 name = **input**(" Product Name: ")

15 category = **input**(" Category: ")

16 price = **float**(**input**(" Price: "))

17 quantity = **int**(**input**(" Quantity: "))

18

19 conn = get\_connection()

20 **if** conn:

21 cursor = conn.cursor()

22 cursor.execute(

23 "INSERT INTO Products (productName, category, price, quantity) VALUES (?, ?, ?, ?)",

24 (name, category, price, quantity)

25 )

26 conn.commit()

27 **print**(" Product added successfully.")

28 conn.close()

29

30 # Displays all products in the database

31 **def** view\_all\_products():

32 conn = get\_connection()

33 **if** conn:

34 cursor = conn.cursor()

35 cursor.execute("SELECT \* FROM Products")

36 rows = cursor.fetchall()

37 **print**("\n All Products:")

38 **for** row **in** rows:

39 **print**(f"ID: {row.productId} | Name: {row.productName} | Category: {row.category} | Price: {row.price} | Qty:

{row.quantity}")

40 conn.close()

41

42 # Searches for a product by ID

43 **def** search\_product\_by\_id():

44 pid = **int**(**input**(" Enter Product ID to Search: "))

45 conn = get\_connection()

46 **if** conn:

47 cursor = conn.cursor()

48 cursor.execute("SELECT \* FROM Products WHERE productId = ?", (pid,))

49 row = cursor.fetchone()

50 **if** row:

51 **print**(f" Found: {row.productName} | Category: {row. category} | Price: {row.price} | Quantity: {row. quantity}")

52 **else**:

53 **print**(" Product not found.")

54 conn.close()

55

56 # Updates an existing product

57 **def** update\_product():

58 pid = **int**(**input**(" Enter Product ID to Update: "))

59 name = **input**(" New Product Name: ")

60 category = **input**(" New Category: ")

61 price = **float**(**input**(" New Price: "))

62 quantity = **int**(**input**(" New Quantity: "))

63

64 conn = get\_connection()

65 **if** conn:

66 cursor = conn.cursor()

67 cursor.execute("""

68 UPDATE Products

69 SET productName = ?, category = ?, price = ?, quantity =

?

70 WHERE productId = ?

71 """, (name, category, price, quantity, pid))

72

73 **if** cursor.rowcount:

74 **print**(" Product updated.")

75 **else**:

76 **print**(" Product ID not found.")

77 conn.commit()

78 conn.close()

79

80 # Deletes a product by ID

81 **def** delete\_product():

82 pid = **int**(**input**(" Enter Product ID to Delete: "))

83 conn = get\_connection()

84 **if** conn:

85 cursor = conn.cursor()

86 cursor.execute("DELETE FROM Products WHERE productId = ?", ( pid,))

87

88 **if** cursor.rowcount:

89 **print**(" Product deleted.")

90 **else**:

91 **print**(" Product ID not found.")

92 conn.commit()

93 conn.close()

94

95 # Menu-driven interface

96 **def** main():

97 **while** True:

98 **print**("\n===============================")

99 **print**(" PRODUCT MANAGEMENT SYSTEM")

100 **print**("===============================")

101 **print**("1. Add Product")

102 **print**("2. View All Products")

103 **print**("3. Search Product by ID")

104 **print**("4. Update Product")

105

**print**("5. Delete Product")

**print**("6. Exit")

choice = **input**(" Enter your choice (1-6): ")

**if** choice == ’1’: add\_product()

**elif** choice == ’2’: view\_all\_products()

**elif** choice == ’3’: search\_product\_by\_id()

**elif** choice == ’4’: update\_product()

**elif** choice == ’5’: delete\_product()

**elif** choice == ’6’:

**print**(" Exiting program. Bye!")

**break else**:

**print**(" Invalid input. Please enter 1-6.")

# Entry point

**if**  name == " main ": main()

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

## Code Components

* + - **Database Connection**: The get\_connection() function establishes a connection to the AI\_DataBase database using pyodbc.
    - **CRUD Operations**: Each operation is implemented as a separate function for modularity and reusability.
    - **Error Handling**: Basic error handling is included for database connections and invalid inputs.
    - **User Interface**: A console-based menu allows users to select operations easily.

# Example Usage

This section provides example scenarios of how the Product Management System can be used, along with sample inputs and outputs.

## Scenario 1: Adding a Product Input:

1

Product Name: Laptop Category: Electronics Price: 999.99

2

3

4

5

Quantity: 10

## Output:

1

Product added successfully.

2

## Scenario 2: Viewing All Products Output:

1

All Products:

ID: 1 | Name: Laptop | Category: Electronics | Price: 999.99 | Qty: 10

ID: 2 | Name: Mouse | Category: Accessories | Price: 19.99 | Qty: 50

2

3

4

## Scenario 3: Searching for a Product Input:

1

Enter Product ID to Search: 1

2

## Output:

1

Found: Laptop | Category: Electronics | Price: 999.99 | Quantity: 10

2

## Scenario 4: Updating a Product Input:

1

Enter Product ID to Update: 1 New Product Name: Gaming Laptop New Category: Electronics

New Price: 1299.99

New Quantity: 8

2

3

4

5

6

## Output:

1

Product updated.

2

## Scenario 5: Deleting a Product Input:

1

Enter Product ID to Delete: 2

2

## Output:

1

Product deleted.

2

# Detailed Example Workﬂow with Sample Product

This section illustrates the complete workflow of the Product Management System using a spe- cific product, a "Smartphone," as an example. Each operation (Add, View, Search, Update, Delete) is demonstrated with step-by-step actions, inputs, and outputs to provide a clear under- standing of the system’s functionality.

## Step 1: Adding a Smartphone Flow:

1. User starts the program and sees the main menu.
2. User selects option "1" to add a product.
3. System prompts for product details: name, category, price, and quantity.
4. User enters details for a "Smartphone."
5. System executes an SQL INSERT query to add the product to the Products table.
6. System confirms successful addition.

## Input:

=============================== PRODUCT MANAGEMENT SYSTEM

===============================

1. Add Product
2. View All Products
3. Search Product by ID
4. Update Product
5. Delete Product
6. Exit

Enter your choice (1-6): 1 Product Name: Smartphone Category: Electronics Price: 799.99

Quantity: 15

1

2

3

4

5

6

7

8

9

10

11

12

13

14

## Output:

1

Product added successfully.

2

## Step 2: Viewing All Products Flow:

1. User selects option "2" from the main menu to view all products.
2. System connects to the database and executes an SQL SELECT query to retrieve all records from the Products table.
3. System displays all products, including the newly added "Smartphone."

## Input:

=============================== PRODUCT MANAGEMENT SYSTEM

===============================

1. Add Product
2. View All Products
3. Search Product by ID
4. Update Product
5. Delete Product
6. Exit

Enter your choice (1-6): 2

1

2

3

4

5

6

7

8

9

10

## Output:

1

All Products:

ID: 1 | Name: Smartphone | Category: Electronics | Price: 799.99 | Qty: 15

2

3

## Step 3: Searching for the Smartphone Flow:

1. User selects option "3" to search for a product by ID.
2. System prompts for the product ID.
3. User enters the ID of the "Smartphone" (e.g., 1).
4. System executes an SQL SELECT query with a WHERE clause to find the product.
5. System displays the product details or an error if the ID is not found.

## Input:

=============================== PRODUCT MANAGEMENT SYSTEM

===============================

1. Add Product
2. View All Products
3. Search Product by ID
4. Update Product
5. Delete Product

1

2

3

4

5

6

7

8

9

6. Exit

Enter your choice (1-6): 3 Enter Product ID to Search: 1

10

11

## Output:

1

Found: Smartphone | Category: Electronics | Price: 799.99 | Quantity: 15

2

## Step 4: Updating the Smartphone Flow:

1. User selects option "4" to update a product.
2. System prompts for the product ID and new details.
3. User enters the ID of the "Smartphone" and updated details (e.g., new price and quantity).
4. System executes an SQL UPDATE query to modify the product record.
5. System confirms the update or displays an error if the ID is not found.

## Input:

=============================== PRODUCT MANAGEMENT SYSTEM

===============================

1. Add Product
2. View All Products
3. Search Product by ID
4. Update Product
5. Delete Product
6. Exit

Enter your choice (1-6): 4 Enter Product ID to Update: 1 New Product Name: Smartphone Pro New Category: Electronics

New Price: 899.99

New Quantity: 12

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

## Output:

1

Product updated.

2

## Step 5: Deleting the Smartphone Flow:

1. User selects option "5" to delete a product.
2. System prompts for the product ID.
3. User enters the ID of the "Smartphone" (e.g., 1).
4. System executes an SQL DELETE query to remove the product.
5. System confirms the deletion or displays an error if the ID is not found.

## Input:

=============================== PRODUCT MANAGEMENT SYSTEM

===============================

1. Add Product
2. View All Products
3. Search Product by ID
4. Update Product
5. Delete Product
6. Exit

Enter your choice (1-6): 5 Enter Product ID to Delete: 1

1

2

3

4

5

6

7

8

9

10

11

## Output:

1

Product deleted.

2

## Step 6: Verifying Deletion Flow:

1. User selects option "2" to view all products to confirm the deletion.
2. System retrieves and displays the current list of products, which should not include the deleted "Smartphone."

## Input:

=============================== PRODUCT MANAGEMENT SYSTEM

===============================

1. Add Product
2. View All Products
3. Search Product by ID
4. Update Product
5. Delete Product
6. Exit

Enter your choice (1-6): 2

1

2

3

4

5

6

7

8

9

10

## Output:

1

All Products:

[No products found]

2

3

# Testing and Validation

The system was tested to ensure reliability and correctness. The following test cases were executed:

Table 1: Test Cases for Product Management System

## Test Case Input Expected Output

Add Product Name: Phone, Category: Electronics, Price: 599.99, Quantity: 20 Product added succ View All Products None List of all products

Search Product ID: 1 Product details or

Update Product ID: 1, Updated details Product updated or

Delete Product ID: 1 Product deleted or

Invalid Choice Choice: 7 Invalid input. Pleas

## Error Handling

* + - **Database Connection Errors**: Handled in get\_connection() with appropriate er- ror messages.
    - **Invalid Inputs**: The system checks for valid numeric inputs for price, quantity, and product ID.
    - **Non-existent IDs**: Search, update, and delete operations check for valid IDs and display appropriate messages.

# Future Enhancements

The Product Management System can be extended with the following features:

* **User Authentication**: Implement login functionality to secure the system.
* **Web Interface**: Develop a web-based frontend using Flask or Django.
* **Advanced Search**: Add filters for searching by category or price range.
* **Data Validation**: Implement stricter input validation (e.g., non-negative prices).
* **Reporting**: Generate inventory reports (e.g., low-stock alerts).
* **Multi-user Support**: Allow multiple users to manage products concurrently.

# Conclusion

The Product Management System successfully demonstrates the integration of Python with SQL Server for efficient inventory management. The modular design, robust error handling, and user-friendly interface make it a reliable solution for small-scale inventory needs. The inclusion of diagrams, test cases, and detailed example workflows ensures a comprehensive understanding of the system. Future enhancements can make the system more versatile and suitable for larger applications.

# Glossary

* **CRUD**: Create, Read, Update, Delete operations for managing data.
* **pyodbc**: A Python library for connecting to ODBC-compliant databases.
* **SQL Server**: A relational database management system by Microsoft.
* **ODBC**: Open Database Connectivity, a standard API for database access.

# References

* Python Official Documentation: [https://www.python.org](http://www.python.org/)
* pyodbc Documentation: https://github.com/mkleehammer/pyodbc
* Microsoft SQL Server Documentation: https://docs.microsoft.com/en-us/ sql
* TikZ Documentation: [https://www.ctan.org/pkg/pgf](http://www.ctan.org/pkg/pgf)