

NUTAN MAHARASHTRA VIDYA PRASARAK MANDAL'S

**Under administrative support of Pimpri Chinchwad
Education Trust**

**Nutan Maharashtra Institute of Engineering and Technology
Talegaon Dabhade, Pune**



PROJECT REPORT ON

“Inventory Management System”

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

(2024-25)

NUTAN MAHARASHTRA VIDYA PRASARAK MANDAL'S

Under administrative support of Pimpri Chinchwad
Education Trust

Nutan Maharashtra Institute of Engineering and Technology

Talegaon Dabhade, Pune



CERTIFICATE

*This is to certify that, this mini project
report entitled: **Inventory Management System**
in this college.*

BY Project Group Members

| NAME: | ROLL NO. |
|--------------------|----------|
| Soham Rajaram Mane | 36 |
| Samruddhi Khot | 29 |
| Aditya Date | 16 |
| Shrut Kamble | 27 |
| Sanskar Gidwani | 20 |

Subject In charge

(Prof. Sarika B. Patil)

Head of the Department

Dr. Ashwini Shinde

Principal

(Dr. S. N. Sapali)

ACKNOWLEDGEMENT

With all respect and gratitude, we would like to thank all people who have helped us directly or indirectly for the completion of this dissertation work.

We thank our project guide **Prof. Sarika B. Patil** for helping us to understand the project topic conceptually in every phase of work. He offered us so much advice, patiently supervising and always guiding in right direction. We have learnt a lot from him and he is truly a dedicated mentor. His encouragement and help made us confident to fulfill my desire and overcome every difficulty we encountered.

We express my heartily gratitude towards **Dr. Ashwini Shinde**, Head of department of Electronics and Telecommunication Engineering for guiding us to understand the work conceptually and also for providing necessary information and required resources with his constant encouragement to complete this dissertation work.

With deep sense of gratitude, we thank our Principal **Dr. S. N. Sapali** and Management of the NMIET for providing all necessary facilities and their constant encouragement and support.

Last but not the least, we thank to all the Teaching & Non-teaching staff members of Electronics and Telecommunication Engineering Department for providing necessary information and required resources. We are ending this acknowledgement with deep indebtedness to my friends who have helped us.

Abstract:-

The Inventory Management System is a comprehensive software solution designed to automate the tracking and management of diverse product stocks within an organization. The system's primary objective is to provide users with a streamlined platform to efficiently manage their inventory levels. Through a user-friendly interface, users can perform various operations such as adding, deleting, modifying, searching, and viewing product information related to their stocks. This enables users to maintain accurate and up-to-date records of their inventory, ensuring optimal stock positions are maintained according to their specific needs.

The system leverages the power of Python programming language and MySQL database management system, connected through an ODBC (Open Database Connectivity) connector. This robust architecture ensures seamless and efficient data handling, enabling swift retrieval and manipulation of inventory data. The MySQL database provides a secure and scalable storage solution, while Python's scripting capabilities facilitate complex logic and business rule implementation.

The system's features include product management, stock management, and reporting capabilities. Users can add new products, update existing product information, and delete obsolete products. The stock management module allows users to track inventory levels, update stock quantities, and set alerts for low stock levels. The system also provides robust search functionality, enabling users to quickly locate specific products or product categories.

To ensure data accuracy and integrity, the system incorporates validation checks and error handling mechanisms. The user-friendly interface is designed to minimize errors and simplify inventory management operations. Additionally, the system's modular architecture enables easy maintenance, updates, and scalability.

Key benefits of the Inventory Management System include enhanced inventory accuracy, improved stock visibility, reduced errors, and increased operational efficiency. By automating inventory management tasks, organizations can optimize their supply chain operations, reduce costs, and improve customer satisfaction. Overall, the Inventory Management System is a powerful tool for organizations seeking to streamline their inventory management processes and improve overall business performance.

Introduction:-

Comprehensive Inventory Management System: Streamlining Inventory Tracking and Management:

The Inventory Management System is a robust, scalable, and user-friendly software solution designed to automate and optimize inventory tracking and management processes. Leveraging the power of Python programming language and MYSQL database management system, this system provides a streamlined platform for organizations to efficiently manage diverse product stocks, ensuring accuracy, visibility, and operational efficiency.

- **System Overview**

The Inventory Management System is a cutting-edge solution that enables organizations to:

1. Automate inventory tracking and management
2. Maintain accurate and up-to-date records of inventory levels
3. Optimize stock positions according to specific needs
4. Enhance inventory accuracy and visibility

- **Key Features**

1. Secure Access: Access is password protected so only authorized individuals can access the Inventory Management System

1. Product Management: Add, delete, modify, and view product information

2. Stock Management: Track inventory levels, update stock quantities, and set alerts for low stock levels

3. Reporting Capabilities: Generate customizable reports on inventory levels, stock movements, and product sales

4. Search Functionality: Quickly locate specific products or product categories

5. Validation Checks and Error Handling: Ensure data accuracy and integrity

6. User-Friendly Interface: Minimize errors and simplify inventory management operations

7. Modular Architecture: Enable easy maintenance, updates, and scalability

8. Secure Delete: Delete operation is password protected so only authorized individuals can perform the task

- **Benefits**

1. Enhanced Inventory Accuracy: Real-time tracking and updates minimize errors and discrepancies

2. Improved Stock Visibility: Instant access to inventory levels and stock movements

3. Reduced Errors: Automated processes minimize manual errors and inconsistencies

4. Increased Operational Efficiency: Streamlined processes and automated reporting save time and resources
5. Optimized Supply Chain Operations: Automating inventory management tasks reduces costs and improves customer satisfaction
6. Safety: Inventory Management System is password protected so only authorized individuals can access it and Delete operation is again separately password protected so among those authorized persons not all can delete the data.

Scope:-

- **Functional Scope:**

1. Product Management: Add, delete, modify, search, and view product information.
2. Stock Management: Track inventory levels, update stock quantities, and set alerts for low stock.
3. User Management: Define roles and permissions for users.
4. Reporting: Generate reports on inventory levels, product movements, and stock valuation.
5. Data Security: Ensure data integrity and confidentiality. And system access and delete operation is password protected.

- **Technical Scope:**

1. Programming Languages: Python.
2. Database Management System: MYSQL.
3. Connectivity: ODBC connector.
4. User Interface: Graphical User Interface (GUI) using tkinter.
5. Operating System: Cross-platform compatibility.

- **Business Scope:**

1. Small to Medium-Sized Enterprises (SMEs).
2. Retail and Wholesale Businesses.
3. Manufacturing Industries.
4. Warehouse Management.
5. Supply Chain Management.

Requirements:-

- **Software:** MySQL, Python, pyodbc, ODBC Driver, tkinter.
- **Hardware:** Personal computer or server for MySQL.
- **Technologies:** MySQL, Python, pyodbc, tkinter.

Data Modelling Features:-

The system uses the following data models:

1. **Products Table:** Stores product details like product id, name, description, category, and price.

| | product_id | name | description | category | price |
|---|------------|------|-------------|----------|-------|
| * | NULL | NULL | NULL | NULL | NULL |

2. **Inventory Table:** Tracks stock levels for each product, linked via a foreign key.

| | inventory_id | product_id | quantity |
|---|--------------|------------|----------|
| * | NULL | NULL | NULL |

3. **TransactionLog Table:** Showcase the transaction type, quantity changed with the corresponding timestamp.

| | transaction_id | product_id | transaction_type | quantity_changed | timestamp |
|---|----------------|------------|------------------|------------------|-----------|
| * | NULL | NULL | NULL | NULL | NULL |

Data Dictionary:-

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 | |
| category | varchar(50) | YES | | NULL | |
| price | decimal(10,2) | YES | | NULL | |

Inventory Table

| | Field | Type | Null | Key | Default | Extra |
|---|--------------|------|------|-----|---------|----------------|
| ► | inventory_id | int | NO | PRI | NULL | auto_increment |
| | product_id | int | NO | MUL | NULL | |
| | quantity | int | YES | | 0 | |

Transaction Table

| | Field | Type | Null | Key | Default | Extra |
|---|------------------|-------------|------|-----|-------------------|-------------------|
| ► | transaction_id | int | NO | PRI | NULL | auto_increment |
| | product_id | int | NO | MUL | NULL | |
| | transaction_type | varchar(10) | YES | | NULL | |
| | quantity_changed | int | YES | | NULL | |
| | timestamp | timestamp | YES | | CURRENT_TIMESTAMP | DEFAULT_GENERATED |

Relational Database Design:-



Data Normalization:-

Data normalization in the Inventory Management System ensures that the database is organized efficiently, reducing redundancy and dependencies while maintaining data integrity. The database design follows the principles of normalization, typically up to the **Third Normal Form (3NF)**, which is suitable for most inventory management applications. Below is how each table and its structure align with normalization standards:

1. First Normal Form (1NF):

- All tables in the system have atomic values, meaning each column contains only indivisible values.
- Each table has a primary key (product_id for Products, inventory_id for Inventory, and transaction_id for TransactionLog), ensuring each record is uniquely identifiable.
- The tables do not contain repeating groups or arrays, making each field contain only a single piece of information.

2. Second Normal Form (2NF):

- The database is organized to ensure that every non-primary key attribute is fully dependent on the primary key.
- For example, in the **Inventory** table, quantity is dependent only on inventory_id and indirectly on product_id.
- Similarly, in the **TransactionLog** table, the transaction_type, quantity_changed, and timestamp are all fully dependent on the transaction_id.

3. Third Normal Form (3NF):

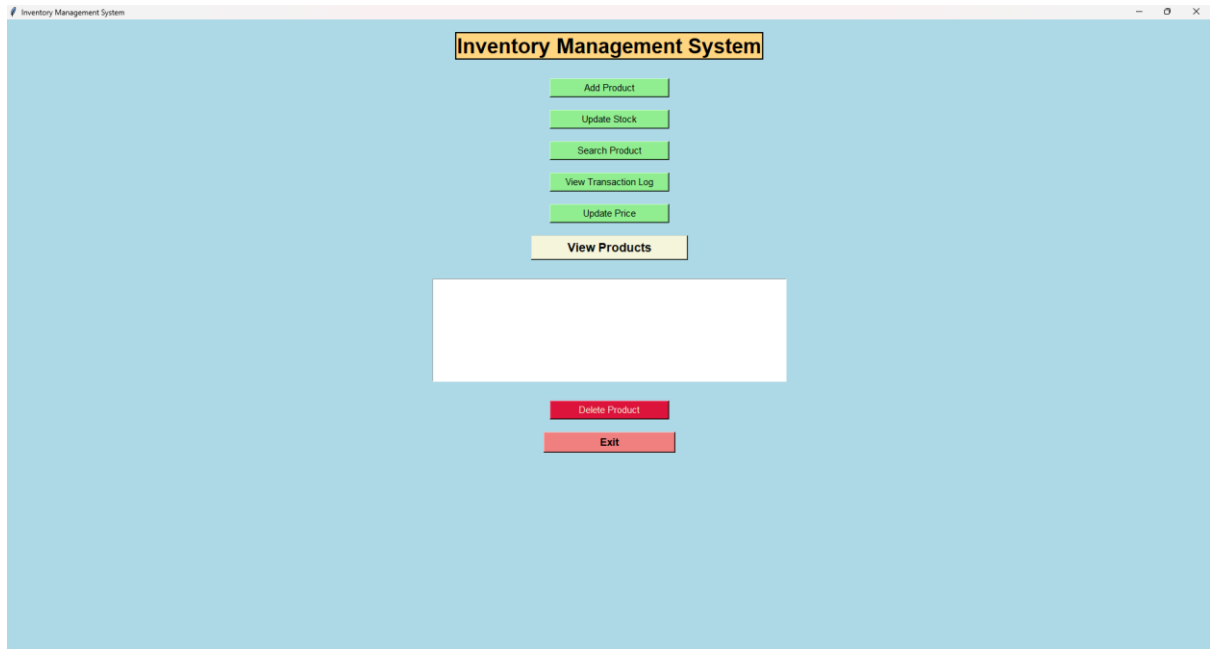
- The design ensures that all non-primary key attributes are only dependent on the primary key and not on other non-key attributes, removing transitive dependencies.
- For instance, in the **Products** table, fields like name, description, category, and price are directly related to the primary key (product_id) and do not depend on each other.

By following these normalization steps, the database achieves an efficient structure where:

- Data redundancy is minimized. For example, product information is stored once in the Products table, and the inventory and transaction logs reference this data via foreign keys (product_id).
- Updates, deletions, and insertions can be performed without causing data anomalies (e.g., ensuring consistent stock levels and product details across different tables).

This normalized structure helps maintain data integrity and ensures the system operates smoothly and efficiently, which is crucial for managing inventory and tracking product transactions accurately.

Graphical User Interface:-



Source code:-

SQL QUERY:

```
CREATE DATABASE InventoryManagement;  
USE InventoryManagement;
```

```
SHOW TABLES;
```

```
-- Products Table
```

```
CREATE TABLE Products (  
    product_id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    description TEXT,  
    category VARCHAR(50),  
    price DECIMAL(10, 2)  
);
```

```
-- Inventory Table
```

```
CREATE TABLE Inventory (  
    inventory_id INT AUTO_INCREMENT PRIMARY KEY,  
    product_id INT NOT NULL,  
    quantity INT DEFAULT 0,  
    FOREIGN KEY (product_id) REFERENCES Products(product_id) ON DELETE  
    CASCADE  
);
```

```
-- Transaction Log Table
```

```
CREATE TABLE TransactionLog (  
    transaction_id INT AUTO_INCREMENT PRIMARY KEY,  
    product_id INT NOT NULL,  
    transaction_type VARCHAR(10), -- 'ADD' or 'REMOVE'  
    quantity_changed INT,  
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (product_id) REFERENCES Products(product_id) ON DELETE  
    CASCADE  
);
```

```
-- Adding some sample products into the Products table
```

```
INSERT INTO Products (name, description, category, price)  
VALUES  
( 'Laptop', '15 inch laptop with 8GB RAM', 'Electronics', 50000.00),  
( 'Smartphone', 'Latest Android smartphone', 'Electronics', 25000.00),  
( 'Desk Chair', 'Ergonomic desk chair', 'Furniture', 7000.00);
```

```
-- Add corresponding stock levels into the Inventory table
```

```
INSERT INTO Inventory (product_id, quantity)  
VALUES  
(1, 10), -- 10 Laptops in stock
```

(2, 20), -- 20 Smartphones in stock
(3, 15); -- 15 Chairs in stock

-- cross check

```
SELECT *  
FROM Products;
```

```
SELECT *  
FROM Inventory;
```

```
SELECT *  
FROM TransactionLog;
```

```
SET FOREIGN_KEY_CHECKS = 0;  
TRUNCATE TABLE Products;  
SET FOREIGN_KEY_CHECKS = 1;
```

```
TRUNCATE TABLE Inventory;
```

```
TRUNCATE TABLE TransactionLog;
```

```
DESC Products;  
DESC Inventory;  
DESC TransactionLog;
```

PYTHON PROGRAM (Using pyodbc and tkinter):

```
import pyodbc  
import tkinter as tk  
from tkinter import messagebox, simpledialog  
from tkinter import ttk  
  
# Connection string for ODBC DSN  
conn_str = 'DSN=InventoryDB;UID=root;PWD=Soham@1206'  
  
# Function to display an error message in a messagebox  
def show_error(message):  
    messagebox.showerror("Error", message)  
  
# Function to display an information message in a messagebox  
def show_info(message):  
    messagebox.showinfo("Info", message)  
  
# Function to authenticate the user  
def authenticate_user(root):  
    password = simpledialog.askstring("Login", "Enter your password:", show='*',  
parent=root)  
    return password == "SOHAM"
```

```

# Function to add a new product
def add_product_gui(name, description, category, price, quantity):
    try:
        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()

        product_query = """
            INSERT INTO Products (name, description, category, price)
            VALUES (?, ?, ?, ?)
        """

        cursor.execute(product_query, (name, description, category, price))
        conn.commit()

        product_id = cursor.execute("SELECT LAST_INSERT_ID()").fetchone()[0]

        inventory_query = """
            INSERT INTO Inventory (product_id, quantity)
            VALUES (?, ?)
        """

        cursor.execute(inventory_query, (product_id, quantity))
        conn.commit()

        show_info(f"Product '{name}' added successfully with Product ID: {product_id}")

    except pyodbc.Error as err:
        show_error(f"Error: {err}")

    finally:
        if conn:
            cursor.close()
            conn.close()

# Function to update stock
def update_stock_gui(product_id, quantity_change):
    try:
        quantity_change = int(quantity_change)

        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()

        transaction_type = 'ADD' if quantity_change > 0 else 'REMOVE'

        update_query = """
            UPDATE Inventory
            SET quantity = quantity + ?
            WHERE product_id = ?
        """

        cursor.execute(update_query, (quantity_change, product_id))
        conn.commit()

```



```

log_query = """
    INSERT INTO TransactionLog (product_id, transaction_type, quantity_changed)
    VALUES (?, ?, ?)
    """

cursor.execute(log_query, (product_id, transaction_type, abs(quantity_change)))
conn.commit()

show_info(f"Stock for Product ID {product_id} updated successfully.")

except ValueError:
    show_error(f"Error: quantity_change must be a valid integer.")

except pyodbc.Error as err:
    show_error(f"Database Error: {err}")

finally:
    try:
        if cursor:
            cursor.close()
    except pyodbc.Error:
        pass

    try:
        if conn:
            conn.close()
    except pyodbc.Error:
        pass

# Function to view all products
def view_products_gui():
    try:
        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()

        query = """
            SELECT p.product_id, p.name, p.description, p.category, p.price, i.quantity
            FROM Products p
            JOIN Inventory i ON p.product_id = i.product_id
            """

        cursor.execute(query)

        for widget in product_list.winfo_children():
            widget.destroy()

        columns = ("product_id", "name", "description", "category", "price", "quantity")
        tree = ttk.Treeview(product_list, columns=columns, show="headings")

        tree.heading("product_id", text="Product ID")

```

```

tree.heading("name", text="Name")
tree.heading("description", text="Description")
tree.heading("category", text="Category")
tree.heading("price", text="Price")
tree.heading("quantity", text="Quantity")

tree.column("product_id", width=80)
tree.column("name", width=150)
tree.column("description", width=200)
tree.column("category", width=100)
tree.column("price", width=80)
tree.column("quantity", width=80)

rows = cursor.fetchall()
for row in rows:
    tree.insert("", "end", values=(row.product_id, row.name, row.description,
row.category, row.price, row.quantity))

tree.pack(fill="both", expand=True)

except pyodbc.Error as err:
    show_error(f"Error: {err}")

finally:
    if conn:
        cursor.close()
        conn.close()

# Function to search products
def search_product_gui(name):
    try:
        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()

        query = """
        SELECT p.product_id, p.name, p.description, p.category, p.price, i.quantity
        FROM Products p
        JOIN Inventory i ON p.product_id = i.product_id
        WHERE p.name LIKE ?
        """

        cursor.execute(query, ('%' + name + '%',))

        rows = cursor.fetchall()

        for widget in product_list.winfo_children():
            widget.destroy()

        if rows:
            columns = ("product_id", "name", "description", "category", "price", "quantity")
            tree = ttk.Treeview(product_list, columns=columns, show="headings")

```

```

tree.heading("product_id", text="Product ID")
tree.heading("name", text="Name")
tree.heading("description", text="Description")
tree.heading("category", text="Category")
tree.heading("price", text="Price")
tree.heading("quantity", text="Quantity")

tree.column("product_id", width=80)
tree.column("name", width=150)
tree.column("description", width=200)
tree.column("category", width=100)
tree.column("price", width=80)
tree.column("quantity", width=80)

for row in rows:
    tree.insert("", "end", values=(row.product_id, row.name, row.description,
row.category, row.price, row.quantity))

tree.pack(fill="both", expand=True)

else:
    show_info("No products found matching the search criteria.")

except pyodbc.Error as err:
    show_error(f"Error: {err}")

finally:
    if conn:
        cursor.close()
        conn.close()

# Function to delete products
def delete_product_gui(product_id):
    try:
        conn = None
        password = simpledialog.askstring("Password Confirmation", "Enter your password to
confirm:", show='*')

        if password != "BATMAN":
            show_error("Incorrect password. Deletion canceled.")
            return

        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()

        cursor.execute("DELETE FROM Inventory WHERE product_id = ?", (product_id,))
        conn.commit()

        cursor.execute("DELETE FROM Products WHERE product_id = ?", (product_id,))

```

```

conn.commit()

show_info(f"Product ID {product_id} deleted successfully.")

except pyodbc.Error as err:
    show_error(f"Error: {err}")

finally:
    if conn:
        cursor.close()
        conn.close()

# Function to view transaction logs
def view_transaction_log_gui(product_id):
    try:
        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()

        query = """
        SELECT transaction_id, transaction_type, quantity_changed, timestamp
        FROM TransactionLog
        WHERE product_id = ?
        ORDER BY timestamp DESC
        """
        cursor.execute(query, (product_id,))

        rows = cursor.fetchall()

        for widget in product_list.winfo_children():
            widget.destroy()

        if rows:
            columns = ("transaction_id", "transaction_type", "quantity_changed", "timestamp")
            tree = ttk.Treeview(product_list, columns=columns, show="headings")

            tree.heading("transaction_id", text="Transaction ID")
            tree.heading("transaction_type", text="Type")
            tree.heading("quantity_changed", text="Quantity Changed")
            tree.heading("timestamp", text="Timestamp")

            tree.column("transaction_id", width=120)
            tree.column("transaction_type", width=100)
            tree.column("quantity_changed", width=120)
            tree.column("timestamp", width=150)

            for row in rows:
                tree.insert("", "end", values=(row.transaction_id, row.transaction_type,
                row.quantity_changed, row.timestamp))

            tree.pack(fill="both", expand=True)

```

```

        else:
            show_info(f"No transaction history found for Product ID {product_id}.")

except pyodbc.Error as err:
    show_error(f"Error: {err}")

finally:
    if conn:
        cursor.close()
        conn.close()

# Function to update price of a product
def update_price_gui(product_id, new_price):
    try:
        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()

        update_query = """
            UPDATE Products
            SET price = ?
            WHERE product_id = ?
        """

        cursor.execute(update_query, (new_price, product_id))
        conn.commit()

        show_info(f"Price for Product ID {product_id} updated successfully.")

    except pyodbc.Error as err:
        show_error(f"Error: {err}")

    finally:
        if conn:
            cursor.close()
            conn.close()

# Function to create the main GUI
def create_gui(root):
    global product_list

    for widget in root.winfo_children():
        widget.destroy()

    root = tk.Tk()
    root.title("Inventory Management System")
    root.geometry("600x500")
    root.configure(bg="#ADD8E6")

    tk.Label(root, text="Inventory Management System", bg="#FFD580", font=("Arial", 25,
"bold"), bd=2, relief="solid").pack(pady=20)

```

```

tk.Button(root, text="Add Product", width=20, bg="#90EE90", font=("Arial", 11),
command=add_product_form).pack(pady=10)

tk.Button(root, text="Update Stock", width=20, bg="#90EE90", font=("Arial", 11),
command=update_stock_form).pack(pady=10)

tk.Button(root, text="Search Product", width=20, bg="#90EE90", font=("Arial", 11),
command=search_product_form).pack(pady=10)

tk.Button(root, text="View Transaction Log", width=20, bg="#90EE90", font=("Arial",
11), command=view_transaction_log_form).pack(pady=10)

tk.Button(root, text="Update Price", width=20, bg="#90EE90", font=("Arial", 11),
command=update_price_form).pack(pady=10)

tk.Button(root, text="View Products", width=20, bg="#F5F5DC", font=("Arial", 14,
"bold"), command=view_products_gui).pack(pady=10)

product_list = tk.Text(root, height=10, width=70)
product_list.pack(pady=20)

tk.Button(root, text="Delete Product", width=20, bg="#DC143C", fg="#F5F5DC",
font=("Arial", 11), command=delete_product_form).pack(pady=10)

tk.Button(root, text="Exit", width=20, bg="#F08080", font=("Arial", 13, "bold"),
command=root.quit).pack(pady=10)

root.mainloop()

def start_app():
    global root
    root = tk.Tk()
    root.withdraw()

    if authenticate_user(root):
        root.deiconify()
        create_gui(root)
    else:
        show_error("Incorrect password. Access denied.")
        root.destroy()

# Forms for each operation
def add_product_form():
    create_form("Add Product", ["Name", "Description", "Category", "Price", "Quantity"],
add_product_gui)

def update_stock_form():
    create_form("Update Stock", ["Product ID", "Quantity Change (+/-)"], update_stock_gui)

```

```
def search_product_form():
    create_form("Search Product", ["Product Name"], search_product_gui)

def delete_product_form():
    result = messagebox.askyesno("Confirmation", "Are you sure you want to delete a
product?")
    if result:
        create_form("Delete Product", ["Product ID"], delete_product_gui)

def view_transaction_log_form():
    create_form("View Transaction Log", ["Product ID"], view_transaction_log_gui)

def update_price_form():
    create_form("Update Price", ["Product ID", "New Price"], update_price_gui)

def create_form(title, fields, submit_action):
    form_win = tk.Toplevel()
    form_win.title(title)

    entries = []
    for idx, field in enumerate(fields):
        tk.Label(form_win, text=field).grid(row=idx, column=0, padx=10, pady=5)
        entry = tk.Entry(form_win)
        entry.grid(row=idx, column=1)
        entries.append(entry)

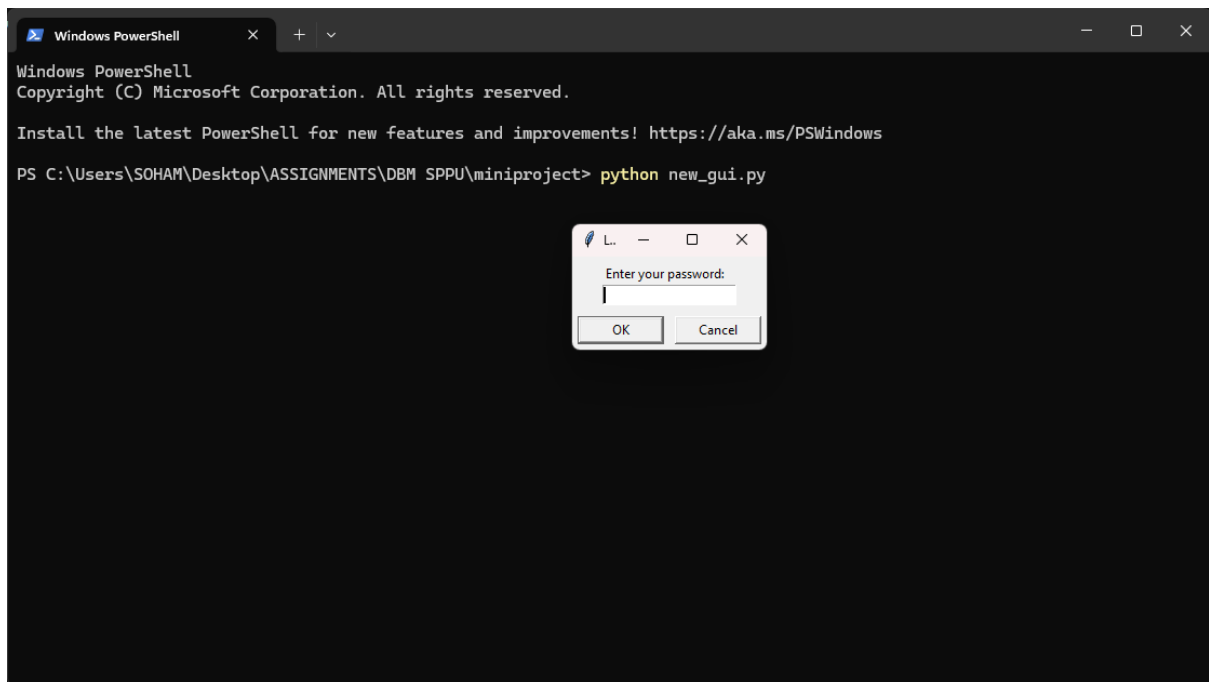
    def submit():
        values = [entry.get() for entry in entries]
        submit_action(*values)
        form_win.destroy()

    tk.Button(form_win, text="Submit", command=submit).grid(row=len(fields), column=1,
pady=10)

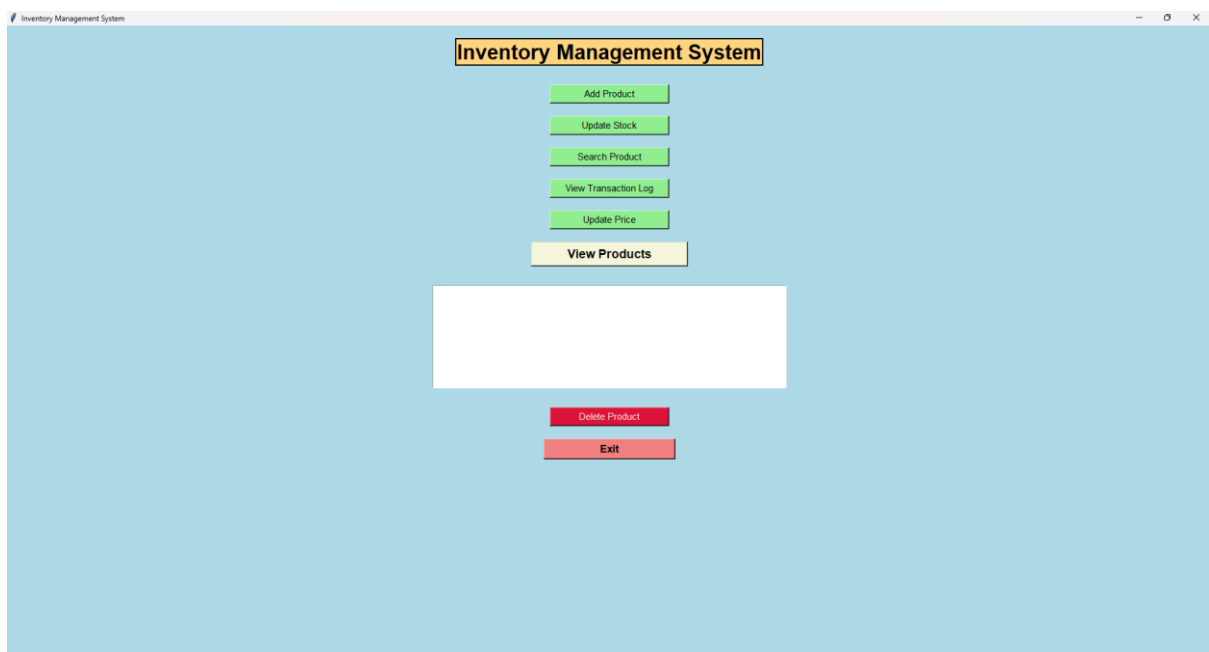
start_app()
```

Testing Documents:-

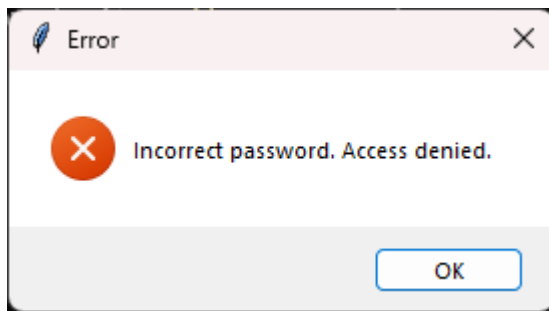
Password to access Inventory Management System:



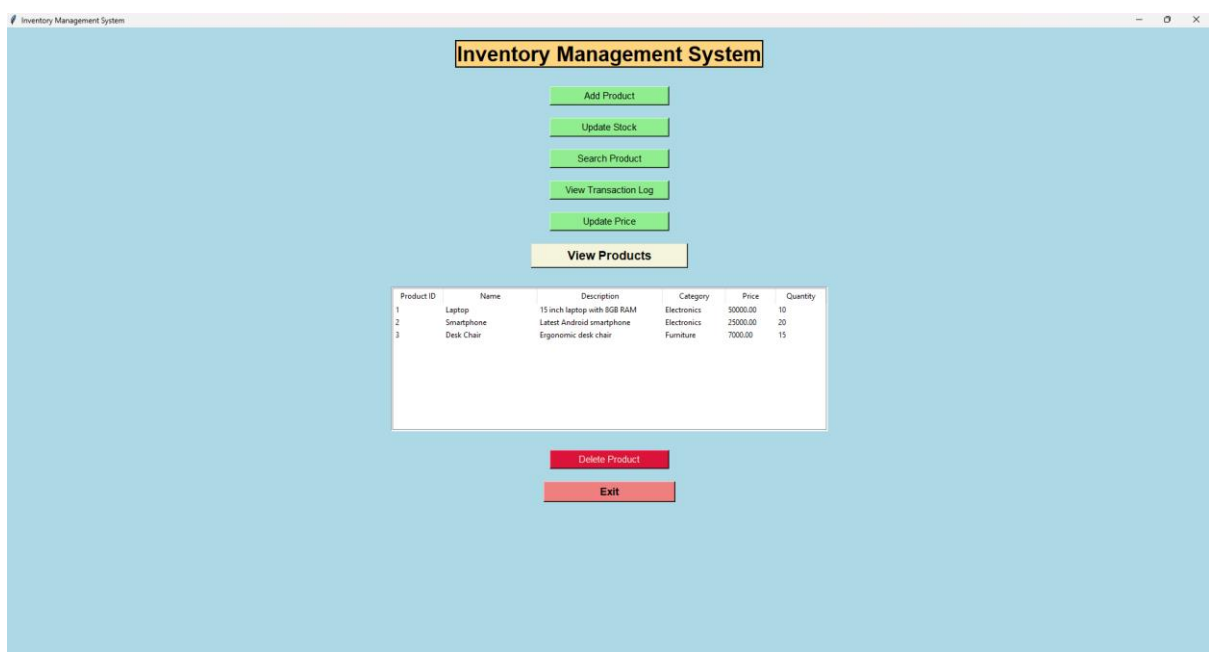
Successful login (Correct Password):



Unsuccessful login (Incorrect password):



View Products:




Products Table:

| | product_id | name | description | category | price |
|---|------------|------------|-----------------------------|-------------|----------|
| ▶ | 1 | Laptop | 15 inch laptop with 8GB RAM | Electronics | 50000.00 |
| | 2 | Smartphone | Latest Android smartphone | Electronics | 25000.00 |
| | 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 |
| * | NULL | NULL | NULL | NULL | NULL |

Inventory Table:

| | inventory_id | product_id | quantity |
|---|--------------|------------|----------|
| ▶ | 1 | 1 | 10 |
| | 2 | 2 | 20 |
| | 3 | 3 | 15 |
| * | NULL | NULL | NULL |

Add Product:



Add Product

—

□

×

Name

Watch

Description

iece for wrist elegance

Category

Fashion


Price

4500

Quantity


10

Submit



Info

×



Product 'Watch' added successfully with Product ID: 4

OK

| Product ID | Name | Description | Category | Price | Quantity |
|------------|------------|--------------------------------------|-------------|----------|----------|
| 1 | Laptop | 15 inch laptop with 8GB RAM | Electronics | 50000.00 | 10 |
| 2 | Smartphone | Latest Android smartphone | Electronics | 25000.00 | 20 |
| 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 | 15 |
| 4 | Watch | Stylish timepiece for wrist elegance | Fashion | 4500.00 | 10 |

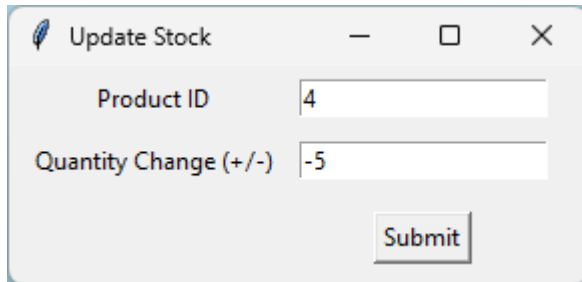
| | product_id | name | description | category | price |
|---|------------|------------|--------------------------------------|-------------|----------|
| ▶ | 1 | Laptop | 15 inch laptop with 8GB RAM | Electronics | 50000.00 |
| | 2 | Smartphone | Latest Android smartphone | Electronics | 25000.00 |
| | 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 |
| | 4 | Watch | Stylish timepiece for wrist elegance | Fashion | 4500.00 |
| * | NULL | NULL | NULL | NULL | NULL |

| | inventory_id | product_id | quantity |
|---|--------------|------------|----------|
| ▶ | 1 | 1 | 10 |
| | 2 | 2 | 20 |
| | 3 | 3 | 15 |
| | 4 | 4 | 10 |
| * | NULL | NULL | NULL |

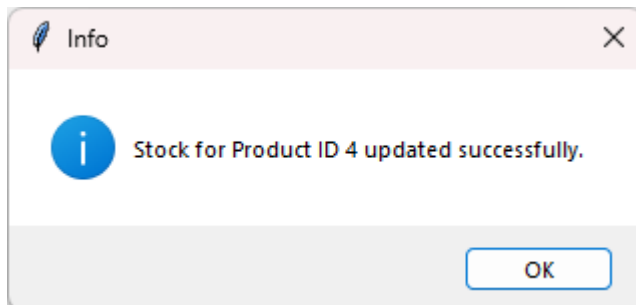
Update Stock:

Quantity Change (+/-): +ve value shows increment

-ve value shows decrement



A dialog box titled "Update Stock" with a feather icon. It contains two input fields: "Product ID" with the value "4" and "Quantity Change (+/-)" with the value "-5". A "Submit" button is located at the bottom right.



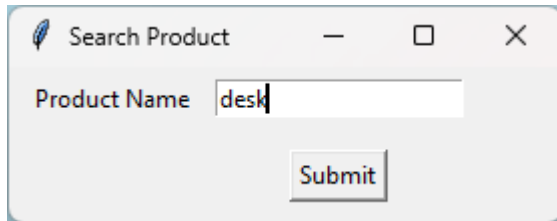
An information dialog box titled "Info" with a feather icon and a close button (X). It displays a blue information icon and the message "Stock for Product ID 4 updated successfully." An "OK" button is at the bottom right.

| Product ID | Name | Description | Category | Price | Quantity |
|------------|------------|--------------------------------------|-------------|----------|----------|
| 1 | Laptop | 15 inch laptop with 8GB RAM | Electronics | 50000.00 | 10 |
| 2 | Smartphone | Latest Android smartphone | Electronics | 25000.00 | 20 |
| 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 | 15 |
| 4 | Watch | Stylish timepiece for wrist elegance | Fashion | 4500.00 | 5 |

| | inventory_id | product_id | quantity |
|---|--------------|------------|----------|
| ▶ | 1 | 1 | 10 |
| | 2 | 2 | 20 |
| | 3 | 3 | 15 |
| | 4 | 4 | 5 |
| * | NULL | NULL | NULL |

Search Product:

Partial matching is provided for efficiency

A screenshot of a 'Search Product' dialog box. It has a title bar with a feather icon, the text 'Search Product', and standard window controls (minimize, maximize, close). Inside, there is a label 'Product Name' followed by a text input field containing the word 'desk'. Below the input field is a 'Submit' button.


Search Product

Product Name

Submit

| Product ID | Name | Description | Category | Price | Quantity |
|------------|------------|----------------------|-----------|---------|----------|
| 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 | 15 |

View Transaction Log:

View Transacti...

Product ID


4

Submit

| Transaction ID | Type | Quantity Changed | Timestamp |
|----------------|--------|------------------|---------------------|
| 1 | REMOVE | 5 | 2024-10-18 12:00:19 |

| | transaction_id | product_id | transaction_type | quantity_changed | timestamp |
|---|----------------|------------|------------------|------------------|---------------------|
| ▶ | 1 | 4 | REMOVE | 5 | 2024-10-18 12:00:19 |
| * | NULL | NULL | NULL | NULL | NULL |


Update Price:


 Update Price

Product ID

New Price

Submit

 Info



Price for Product ID 4 updated successfully.

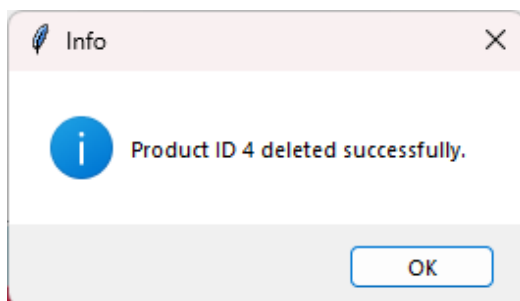
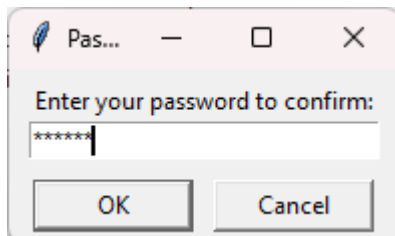
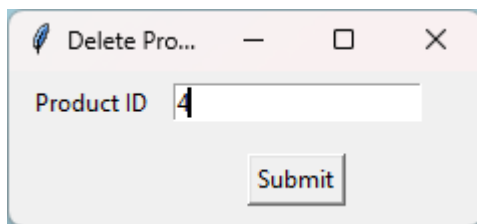
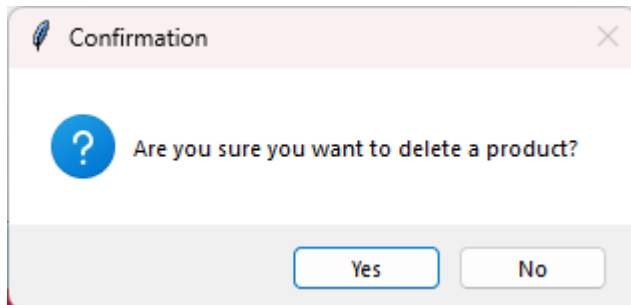
OK

| Product ID | Name | Description | Category | Price | Quantity |
|------------|------------|--------------------------------------|-------------|----------|----------|
| 1 | Laptop | 15 inch laptop with 8GB RAM | Electronics | 50000.00 | 10 |
| 2 | Smartphone | Latest Android smartphone | Electronics | 25000.00 | 20 |
| 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 | 15 |
| 4 | Watch | Stylish timepiece for wrist elegance | Fashion | 4000.00 | 5 |

| | product_id | name | description | category | price |
|---|------------|------------|--------------------------------------|-------------|----------|
| ▶ | 1 | Laptop | 15 inch laptop with 8GB RAM | Electronics | 50000.00 |
| | 2 | Smartphone | Latest Android smartphone | Electronics | 25000.00 |
| | 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 |
| | 4 | Watch | Stylish timepiece for wrist elegance | Fashion | 4000.00 |
| ★ | NULL | NULL | NULL | NULL | NULL |

Delete Product:

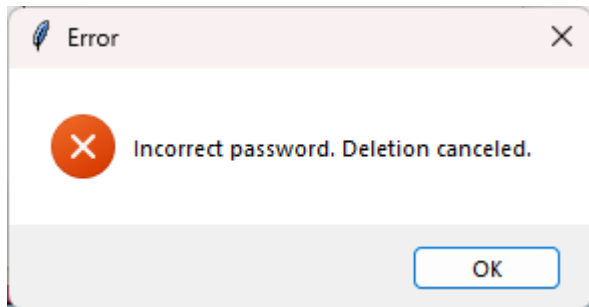
Delete operation pops a conformation window and then asks for PASSWORD.



| Product ID | Name | Description | Category | Price | Quantity |
|------------|------------|-----------------------------|-------------|----------|----------|
| 1 | Laptop | 15 inch laptop with 8GB RAM | Electronics | 50000.00 | 10 |
| 2 | Smartphone | Latest Android smartphone | Electronics | 25000.00 | 20 |
| 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 | 15 |

| | product_id | name | description | category | price |
|---|------------|------------|-----------------------------|-------------|----------|
| ▶ | 1 | Laptop | 15 inch laptop with 8GB RAM | Electronics | 50000.00 |
| | 2 | Smartphone | Latest Android smartphone | Electronics | 25000.00 |
| | 3 | Desk Chair | Ergonomic desk chair | Furniture | 7000.00 |
| * | NULL | NULL | NULL | NULL | NULL |

If PASSWORD for delete operation is wrong it shows the following pop-up



Conclusion:-

In conclusion, the Inventory Management System is a comprehensive and robust software solution that streamlines inventory tracking and management processes, ensuring accuracy, visibility, and operational efficiency. With its cutting-edge features, scalable architecture, and user-friendly interface, this system is an ideal solution for organizations seeking to optimize their inventory management processes and improve overall business performance.

- **Key Takeaways:**

1. Automated inventory tracking and management
2. Enhanced inventory accuracy and visibility
3. Reduced errors and improved operational efficiency
4. Optimized supply chain operations
5. Scalable and secure architecture
6. Secure from unwanted access and deletion

- **Future Implications:**

The Inventory Management System has the potential to revolutionize inventory management processes across various industries, enabling organizations to:

1. Improve customer satisfaction
2. Reduce costs and increase revenue
3. Enhance decision-making capabilities
4. Stay competitive in the market

The Inventory Management System is a powerful tool that can transform inventory management processes, enabling organizations to achieve operational excellence and drive business growth. Its implementation can have a significant impact on an organization's bottom line, making it an essential investment for businesses seeking to stay ahead in today's competitive market.