Create an Inventory Management System

Creating an Inventory Management System involves several components, including user authentication, data management, and a graphical user interface (GUI). Below is a simplified example using Python with a basic GUI implemented using tkinter. This example includes functionalities to add, edit, and delete products, track inventory levels, and generate low-stock alerts.

**Prerequisites**

You will need the following libraries:

* tkinter for the GUI
* sqlite3 for database management
* pandas for reporting (optional)

**Structure**

1. **Database Setup**: Create a SQLite database to store product information.
2. **User Authentication**: Simple authentication mechanism.
3. **GUI Implementation**: Create a graphical interface for the inventory management.

**Code Implementation**

import tkinter as tk

from tkinter import messagebox, simpledialog

import sqlite3

import pandas as pd

# Database Setup

def init\_db():

conn = sqlite3.connect('inventory.db')

c = conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS products

(id INTEGER PRIMARY KEY, name TEXT, quantity INTEGER, price REAL)''')

conn.commit()

conn.close()

# User Authentication

def authenticate(username, password):

# For simplicity, let's use a hardcoded username and password

return username == "admin" and password == "password"

# Inventory Management Class

class InventoryManager:

def \_\_init\_\_(self):

self.init\_db()

def init\_db(self):

init\_db()

def add\_product(self, name, quantity, price):

conn = sqlite3.connect('inventory.db')

c = conn.cursor()

c.execute("INSERT INTO products (name, quantity, price) VALUES (?, ?, ?)",

(name, quantity, price))

conn.commit()

conn.close()

def edit\_product(self, product\_id, name, quantity, price):

conn = sqlite3.connect('inventory.db')

c = conn.cursor()

c.execute("UPDATE products SET name=?, quantity=?, price=? WHERE id=?",

(name, quantity, price, product\_id))

conn.commit()

conn.close()

def delete\_product(self, product\_id):

conn = sqlite3.connect('inventory.db')

c = conn.cursor()

c.execute("DELETE FROM products WHERE id=?", (product\_id,))

conn.commit()

conn.close()

def get\_low\_stock(self, threshold):

conn = sqlite3.connect('inventory.db')

c = conn.cursor()

c.execute("SELECT \* FROM products WHERE quantity < ?", (threshold,))

low\_stock = c.fetchall()

conn.close()

return low\_stock

def generate\_report(self):

conn = sqlite3.connect('inventory.db')

df = pd.read\_sql\_query("SELECT \* FROM products", conn)

conn.close()

return df

# GUI Implementation

class InventoryApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Inventory Management System")

self.manager = InventoryManager()

self.create\_widgets()

def create\_widgets(self):

tk.Label(self.root, text="Product Name").grid(row=0, column=0)

tk.Label(self.root, text="Quantity").grid(row=1, column=0)

tk.Label(self.root, text="Price").grid(row=2, column=0)

self.name\_entry = tk.Entry(self.root)

self.name\_entry.grid(row=0, column=1)

self.quantity\_entry = tk.Entry(self.root)

self.quantity\_entry.grid(row=1, column=1)

self.price\_entry = tk.Entry(self.root)

self.price\_entry.grid(row=2, column=1)

tk.Button(self.root, text="Add Product", command=self.add\_product).grid(row=3, column=0, columnspan=2)

tk.Button(self.root, text="View Low Stock", command=self.view\_low\_stock).grid(row=4, column=0, columnspan=2)

tk.Button(self.root, text="Generate Report", command=self.generate\_report).grid(row=5, column=0, columnspan=2)

def add\_product(self):

name = self.name\_entry.get()

quantity = int(self.quantity\_entry.get())

price = float(self.price\_entry.get())

self.manager.add\_product(name, quantity, price)

messagebox.showinfo("Success", "Product added successfully!")

def view\_low\_stock(self):

low\_stock\_items = self.manager.get\_low\_stock(5) # Example threshold

low\_stock\_msg = "\n".join([f"{item[1]}: {item[2]} units" for item in low\_stock\_items])

messagebox.showinfo("Low Stock Products", low\_stock\_msg if low\_stock\_msg else "No low stock products.")

def generate\_report(self):

df = self.manager.generate\_report()

messagebox.showinfo("Inventory Report", df.to\_string())

if \_\_name\_\_ == "\_\_main\_\_":

init\_db()

username = simpledialog.askstring("Username", "Enter username:")

password = simpledialog.askstring("Password", "Enter password:", show='\*')

if authenticate(username, password):

root = tk.Tk()

app = InventoryApp(root)

root.mainloop()

else:

messagebox.showerror("Authentication Failed", "Invalid username or password.")