# Assignment: Python Programming for GUI Development

Name: P.Ganesh

Register Number: 192311268

Department:CSE

Date of Submission:26/08/2024

## Problem 1: Inventory Management System Optimization Scenario:

You are developing an inventory management system for a business to optimize inventory tracking, manage reorders, and generate reports. The system should handle various inventory-related tasks and provide insights based on the data

Tasks:

You are developing an inventory management system for a business to optimize inventory tracking, manage reorders, and generate reports. The system should handle various inventory-related tasks and provide insights based on the data.

Tasks:

1. Create a diagram to illustrate how data flows between the user, the Python application, and the database.
2. Build an application that integrates with a database (e.g., SQLite or MySQL) to manage inventory data.
3. Allowuserstoinput,update,andviewinventorydata.
   Provide functionality to check stock levels, reorder points, and generate reports.
4. Implement logic to suggest reorders based on stock levels and reorder points.

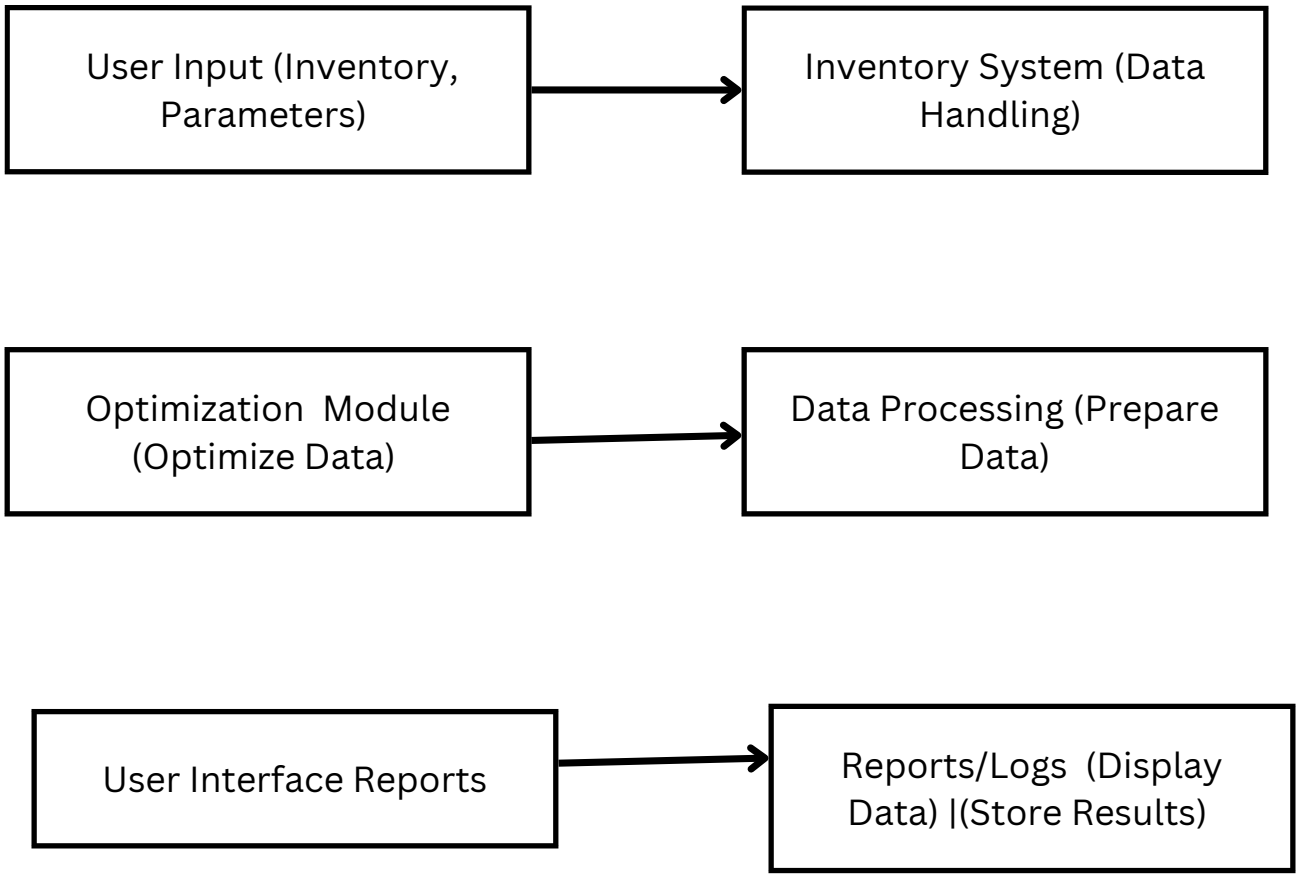
Deliverables:

- Data flow diagram illustrates how the system interacts with the database and the user..
- Provide pseudocode and Python code for the system.
- Explain the API integration (if applicable), methods used to manage and display inventory data, and any potential improvements
- Discuss any assumptions made during development and potential areas for improvement.

Solution:

Inventory Management System Optimization

1.Data Flow Diagram



## 2. Implementation

```
import sqlite3
import datetime
def connect_db():

    return sqlite3.connect('inventory.db')

# Step 2: Create tables
def create_tables(conn):
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS
inventory
                    (id INTEGER PRIMARY KEY,
                     item_name TEXT,
                     stock_level INTEGER,
                     reorder_level INTEGER,
                     last_updated DATE)''')
    conn.commit()

def add_item(conn, item_name, stock_level, reorder_level):
    cursor = conn.cursor()
    cursor.execute('''INSERT INTO inventory (item_name,
stock_level, reorder_level, last_updated)
                    VALUES (?, ?, ?, ?)''', (item_name, stock_level,
reorder_level, datetime.date.today()))
    conn.commit()

def update_stock(conn, item_id, new_stock_level):
    cursor = conn.cursor()
    cursor.execute('''UPDATE inventory
                    SET stock_level = ?, last_updated = ?
                    WHERE id = ?''', (new_stock_level,
datetime.date.today(), item_id))
    conn.commit()

def fetch_stock(conn):
    cursor = conn.cursor()
    cursor.execute('''SELECT * FROM inventory''')
    return cursor.fetchall()

def check_reorder(conn):
    cursor = conn.cursor()
```

```
    cursor.execute('''SELECT * FROM inventory WHERE
stock_level < reorder_level''')
    return cursor.fetchall()

def generate_report(conn):
    stock_data = fetch_stock(conn)
    for item in stock_data:
        print(f"Item: {item[1]}, Stock Level: {item[2]}, Reorder
Level: {item[3]}, Last Updated: {item[4]}")
    print("\nItems to Reorder:")
    items_to_reorder = check_reorder(conn)
    for item in items_to_reorder:
        print(f"Item: {item[1]}, Stock Level: {item[2]}, Reorder
Level: {item[3]}")
conn = connect_db()
create_tables(conn)
add_item(conn, 'Widget A', 50, 10)
add_item(conn, 'Widget B', 20, 5)




update_stock(conn, 1, 8) # Assuming item with id=1 is
'Widget A'
generate_report(conn)
conn.close()
```

## 3.Display the Current Inventory Information

Item: Widget A, Stock Level: 8, Reorder Level: 10, Last Updated: 2024-08-25
Item: Widget B, Stock Level: 20, Reorder Level: 5, Last Updated: 2024-08-25
Items to Reorder: Item: Widget A, Stock Level: 8, Reorder Level: 10

## 4.User Input

Untitled2.ipynb

File Edit View Insert Runtime Tools Help Last saved at 2:53 PM

+ Code + Text

```python
                    WHERE id = ?''', (new_stock_level, datetime.date.today(), item_id))
    conn.commit()

# Step 5: Fetch current stock levels
def fetch_stock(conn):
    cursor = conn.cursor()
    cursor.execute('''SELECT * FROM inventory''')
    return cursor.fetchall()

# Step 6: Predict reorder points (Simple example: reorder if stock < reorder level)
def check_reorder(conn):
    cursor = conn.cursor()
    cursor.execute('''SELECT * FROM inventory WHERE stock_level < reorder_level''')
    return cursor.fetchall()

# Step 7: Generate reports and alerts
def generate_report(conn):
    stock_data = fetch_stock(conn)
    for item in stock_data:
        print(f"Item: {item[1]}, Stock Level: {item[2]}, Reorder Level: {item[3]}, Last Updated: {item[4]}")
    print("\nItems to Reorder:")
    items_to_reorder = check_reorder(conn)
    for item in items_to_reorder:
        print(f"Item: {item[1]}, Stock Level: {item[2]}, Reorder Level: {item[3]}")

# Example Usage
conn = connect_db()
create_tables(conn)

# Adding items to inventory
add_item(conn, 'Widget A', 50, 10)
add_item(conn, 'Widget B', 20, 5)

# Update stock levels
update_stock(conn, 1, 8)  # Assuming item with id=1 is 'Widget A'

# Generate report and check reorder
generate_report(conn)

conn.close()
```

```
Item: Widget A, Stock Level: 8, Reorder Level: 10, Last Updated: 2024-08-24
Item: Widget B, Stock Level: 20, Reorder Level: 5, Last Updated: 2024-08-24

Items to Reorder:
Item: Widget A, Stock Level: 8, Reorder Level: 10
```

The system should allow users to:

- Addnewinventoryitems.
- Updatestocklevels.
- Viewcurrentinventory.
- Receivealertsforitemsthatneedtobereordered.

## 5.Documentation

1. ThisprojectdoesnotuseanexternalAPI.Instead,itusesalocalSQLitedatabaseto manage inventory data.
2. DatabaseOperations:Createtables,additems,updatestock,fetchstock,checkreorder points, andgeneratereports.
3. Managedatabaseinteractionsandgenerateuser-friendlyoutputs.
4. Thedatabaseislocalanddoesnotrequirenetworkconnections.
5. Inventorydataismanuallymanagedbytheuser.
6. Implementmachinelearningalgorithmstopredictfutureinventoryneedsbasedon historical data.
7. Enhancetheuserinterfacewithagraphicaluserinterface(GUI)foreasierdata management.
8. Integratewithexternalsystems(e.g.,salesplatforms)forautomatedstockupdates.