Assignment: Python Programming for GUI Development

Name: G. Shiva Dhanashekar

Register Number: 192311318

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 inventoryrelated 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 inventoryrelated tasks and provide insights based on the data.

Tasks:

1. Create a diagram to illustrate how data flows between the user, the Pythonapplication, and the database.
2. Build an application that integrates with a database (e.g., SQLite or MySQL) tomanage inventory data.
3. Allowuserstoinput,update,andviewinventorydata.

Provide functionality to check stock levels, reorder points, and generate reports.

1. 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

U

s

e

r

I

n

p

u

t

(

I

n

v

e

n

t

o

r

y

,

P

a

r

a

m

e

t

e

r

s

)

I

n

v

e

n

t

o

r

y

S

y

s

t

e

m

(

D

a

t

a

H

a

n

d

l

i

n

g

)

Data Processing (Prepare

Data)

Optimization

Module

(

Optimize Data

)

User Interface Reports

Reports/Logs

(

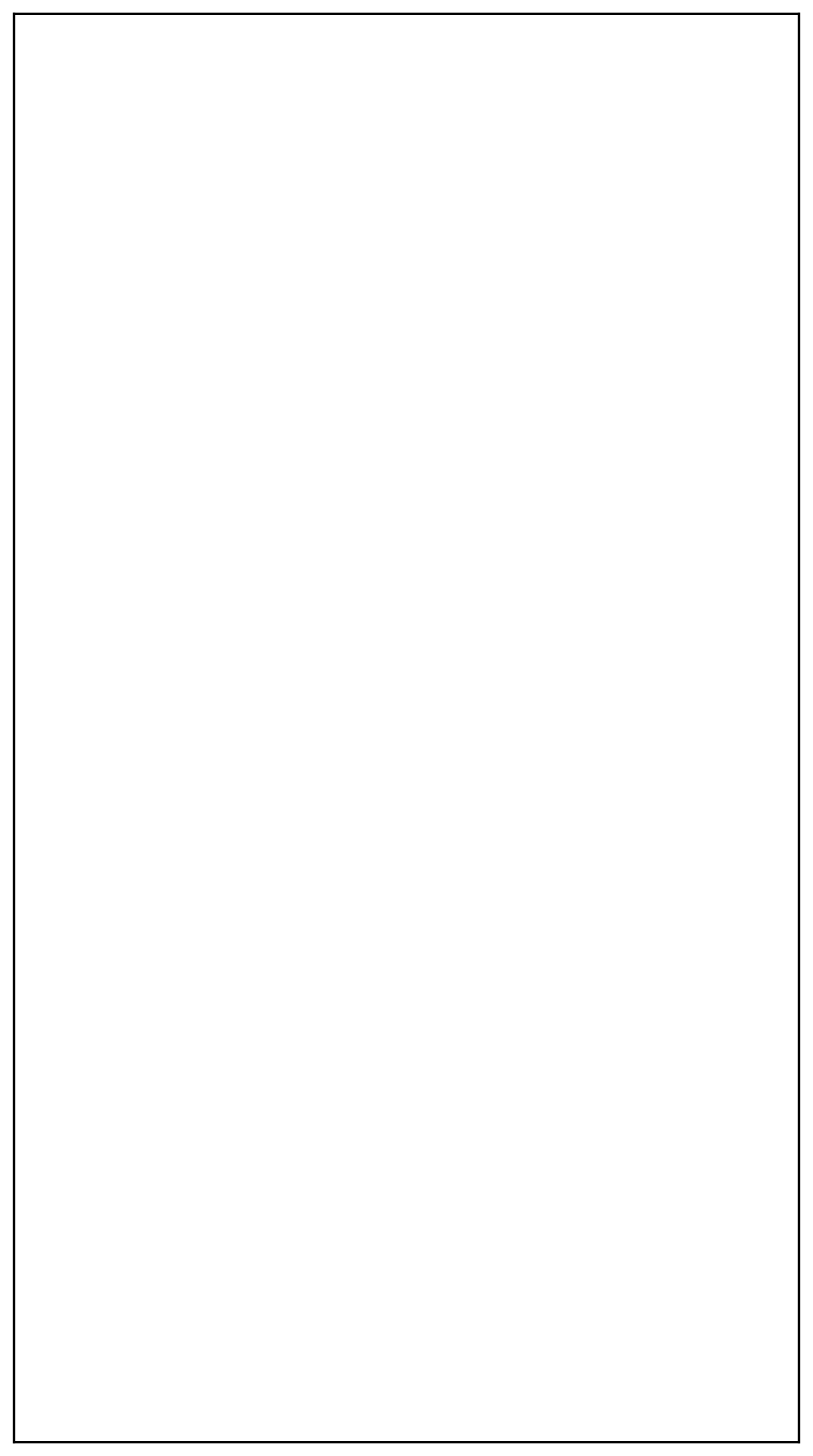
Display

Data) |(Store Results)

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()

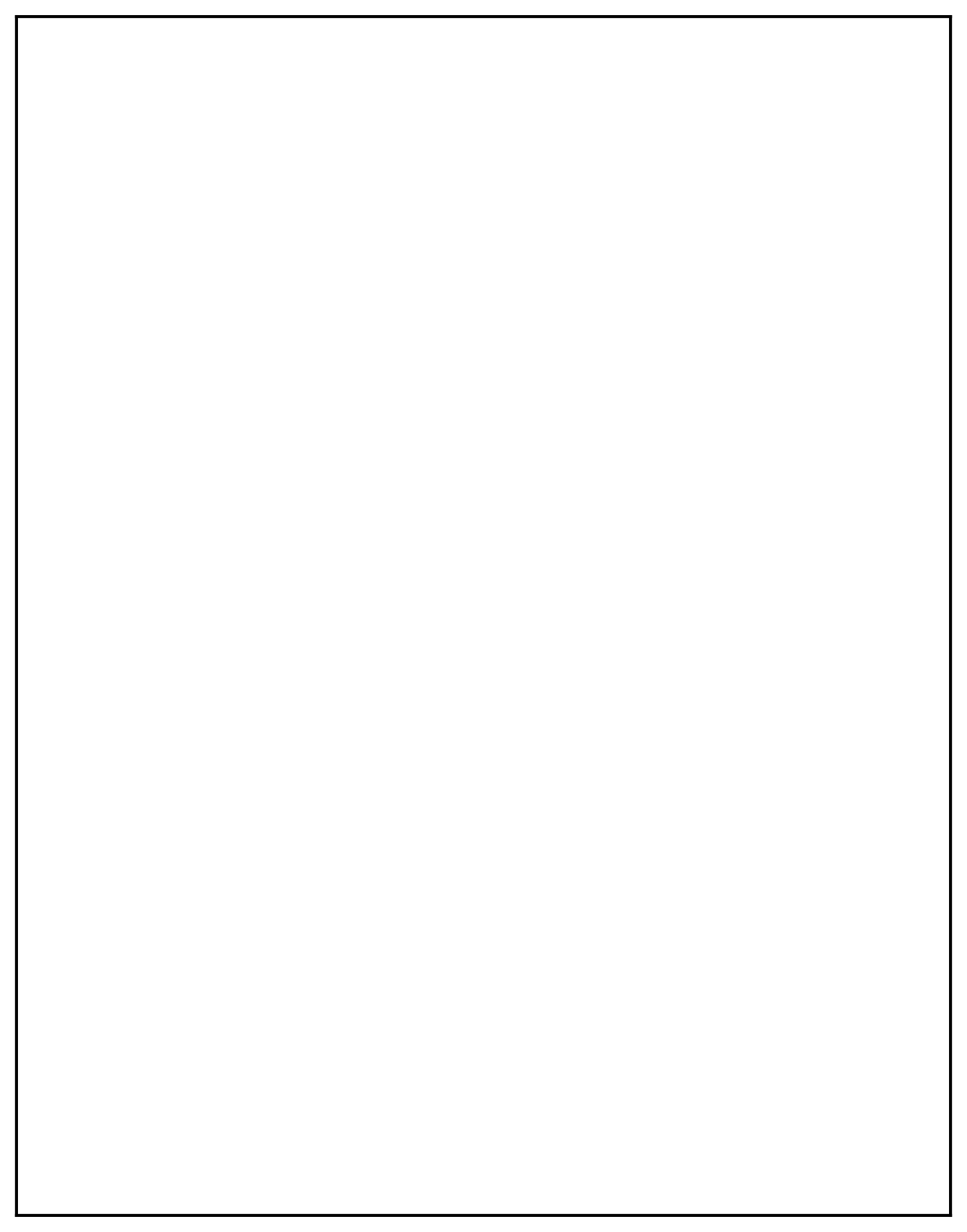
def check\_reorder(conn): cursor = conn.cursor() cursor.execute('''SELECT \* FROM inventory WHERE

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

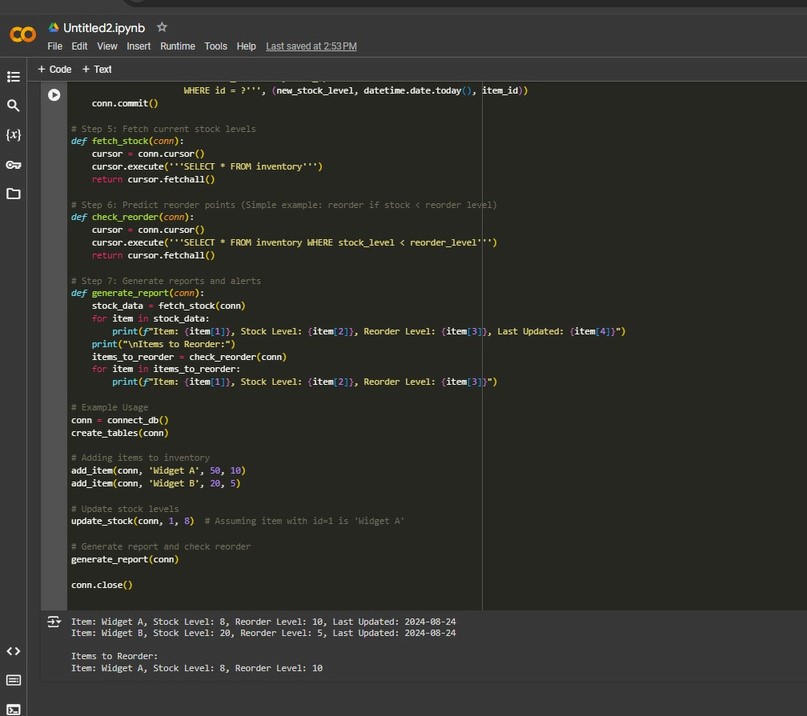
'Widget A' generate\_report(conn) conn.close()

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



The system should allow users to:

* Addnewinventoryitems.
* Updatestocklevels.
* Viewcurrentinventory.
* Receivealertsforitemsthatneedtobereordered.

5.Documentation

1. ThisprojectdoesnotuseanexternalAPI.Instead,itusesalocalSQLitedatabasetomanage inventory data.
2. DatabaseOperations:Createtables,additems,updatestock,fetchstock,checkreorderpoints, andgeneratereports.
3. Managedatabaseinteractionsandgenerateuser-friendlyoutputs.
4. Thedatabaseislocalanddoesnotrequirenetworkconnections.
5. Inventorydataismanuallymanagedbytheuser.
6. Implementmachinelearningalgorithmstopredictfutureinventoryneedsbasedonhistorical data.
7. Enhancetheuserinterfacewithagraphicaluserinterface(GUI)foreasierdatamanagement.
8. Integratewithexternalsystems(e.g.,salesplatforms)forautomatedstockupdates.