AGUIRRE, PAUL VINCENT S. 05/10/2025
M001— CP102 – Computer Programming 2 PROF: RODRIGO JR. C. BELLEZA

DOCUMENTATION OF CODE

SYSTEM DESCRIPTION

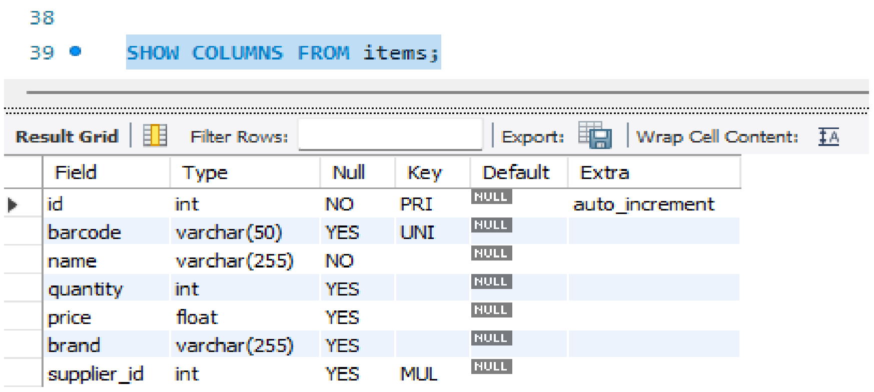
The system is called Inventory Management System. In this system, the developer used Python as the base programming language with a Graphical User Interface made using PyQt5**.** With the help of Qt Designer, the developer was able to create a simple, GUI-based system. For storing data, the developer used a MySQL database managed through MySQL Workbench 8.0 CE.

Key Features:

- GUI built in PyQt5
- Main Menu with navigation to Inventory and Supplier Manger window
- CRUD operations on two related databases tables (items and suppliers)
- Theme Toggle (Light / Dark) via *qdarktheme*

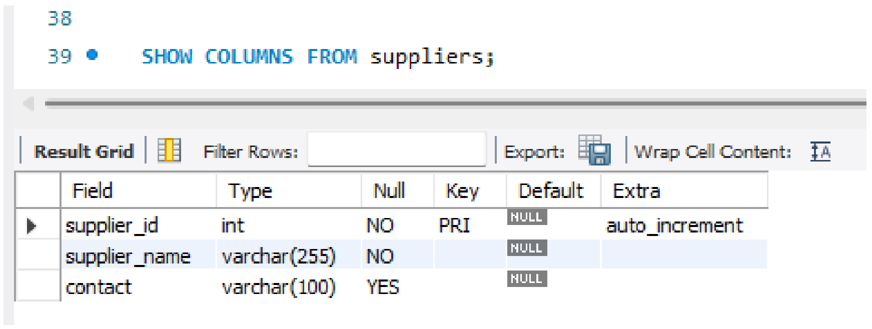
This system allows the user to track items by barcode, name, quantity, price, and brand, as well as manage their suppliers. This is a simple management system that can be used by *sari-sari store* or store staff who need to add, update, delete, and review inventory items. Additionally, the system keeps all contact information of the suppliers so that when an item is nearly out of stock, the user can easily identify and contact the appropriate supplier to restock it.

LIST OF TABLES



```
SHOW COLUMNS FROM items;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
barcode	varchar(50)	YES	UNI	NULL	
name	varchar(255)	NO		NULL	
quantity	int	YES		NULL	
price	float	YES		NULL	
brand	varchar(255)	YES		NULL	
supplier_id	int	YES	MUL	NULL	



```
SHOW COLUMNS FROM suppliers;
```

Field	Type	Null	Key	Default	Extra
supplier_id	int	NO	PRI	NULL	auto_increment
supplier_name	varchar(255)	NO		NULL	
contact	varchar(100)	YES		NULL	

First Screenshot

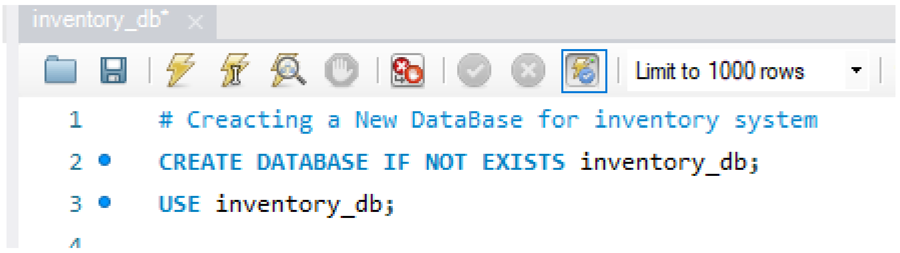
- Each row represents a single product in the inventory and in that table, it has—id, quantity, and price are identified by the system as numeric values. Specifically, price is a floating-point type, which means it can store values up to two decimal places. On the other hand, name and brand are character-based types, similar to strings or alphabetical inputs. The barcode field is also treated as a character type, since barcodes can contain both letters and numbers, depending on how the user creates them. Lastly, the supplier_id field is used to link an item to the supplier that provides it.

Second Screenshot

- In this table it has three fields and those are supplier_id, supplier_name, and their contact and each row hold the information of them.

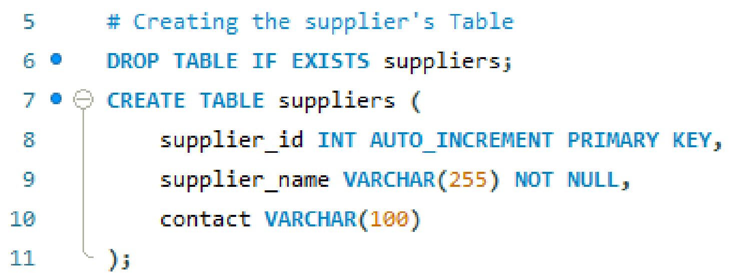
SQL SCRIPT

Creating A New Database



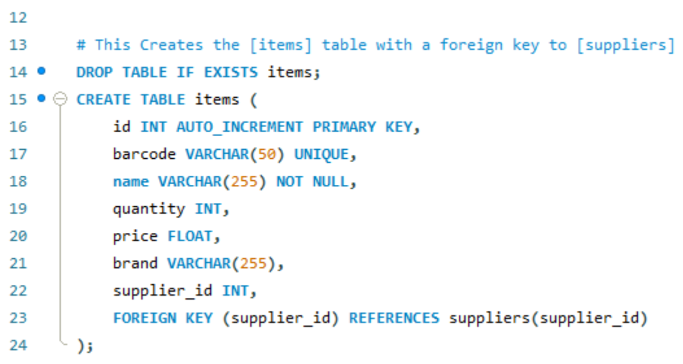
```
# Creacting a New DataBase for inventory system
CREATE DATABASE IF NOT EXISTS inventory_db;
USE inventory_db;
```

Creating the supplier's Table



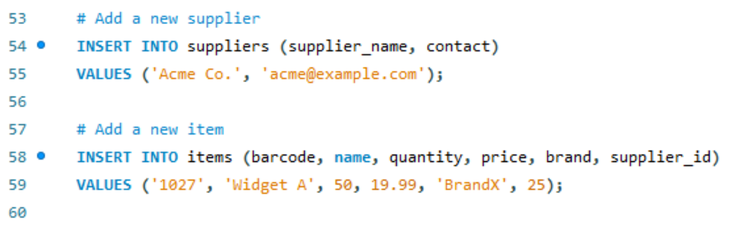
```
# Creating the supplier's Table
DROP TABLE IF EXISTS suppliers;
CREATE TABLE suppliers (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    supplier_name VARCHAR(255) NOT NULL,
    contact VARCHAR(100)
);
```

Creating the item's Table



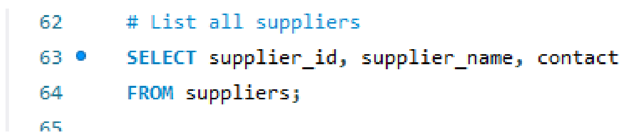
```
# This Creates the [items] table with a foreign key to [suppliers]
DROP TABLE IF EXISTS items;
CREATE TABLE items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    barcode VARCHAR(50) UNIQUE,
    name VARCHAR(255) NOT NULL,
    quantity INT,
    price FLOAT,
    brand VARCHAR(255),
    supplier_id INT,
    FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)
);
```

Creating / Adding an Item to the Item's Table

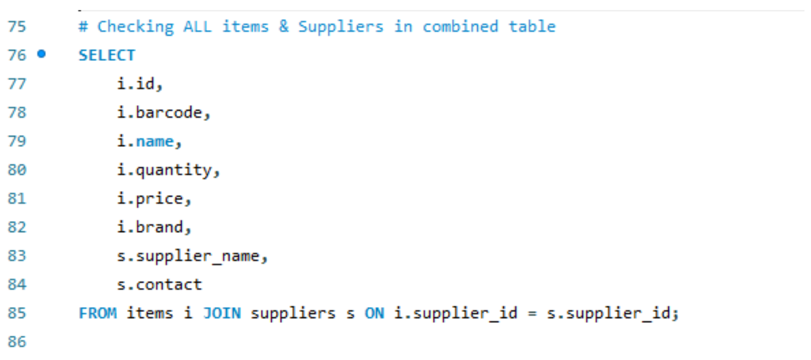

```
# Add a new supplier
INSERT INTO suppliers (supplier_name, contact)
VALUES ('Acme Co.', 'acme@example.com');

# Add a new item
INSERT INTO items (barcode, name, quantity, price, brand, supplier_id)
VALUES ('1027', 'Widget A', 50, 19.99, 'BrandX', 25);
```

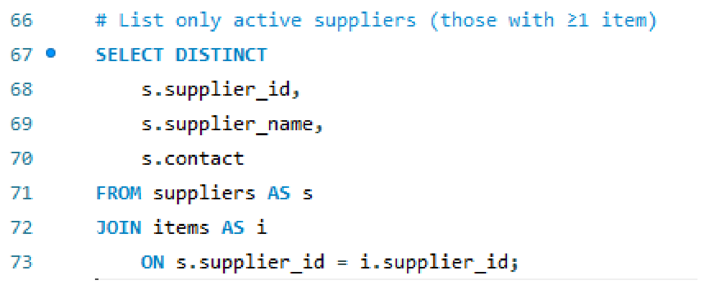
\

Listing All Suppliers

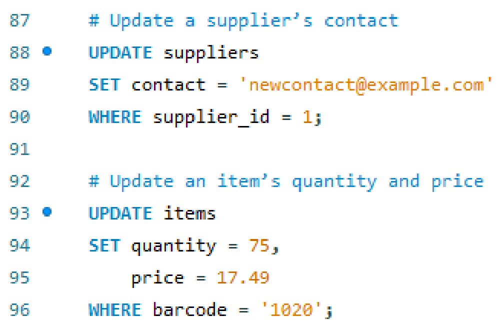

```
# List all suppliers
SELECT supplier_id, supplier_name, contact
FROM suppliers;
```

List all Items with their respective suppliers

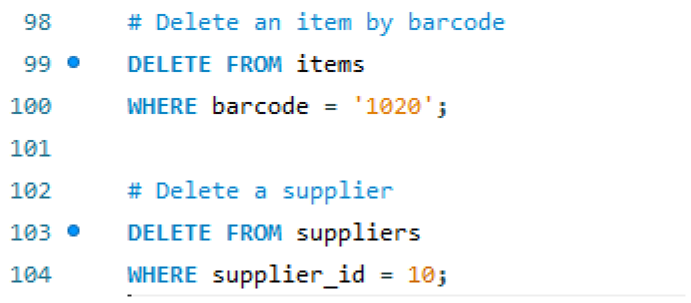

```
# Checking ALL items & Suppliers in combined table
SELECT
    i.id,
    i.barcode,
    i.name,
    i.quantity,
    i.price,
    i.brand,
    s.supplier_name,
    s.contact
FROM items i JOIN suppliers s ON i.supplier_id = s.supplier_id;
```


```
# List only active suppliers (those with ≥1 item)
SELECT DISTINCT
    s.supplier_id,
    s.supplier_name,
    s.contact
FROM suppliers AS s
JOIN items AS i
    ON s.supplier_id = i.supplier_id;
```

Updating a current value


```
# Update a supplier's contact
UPDATE suppliers
SET contact = 'newcontact@example.com'
WHERE supplier_id = 1;

# Update an item's quantity and price
UPDATE items
SET quantity = 75,
    price = 17.49
WHERE barcode = '1020';
```

Deleting a row or a value / Item



```
# Delete an item by barcode
DELETE FROM items
WHERE barcode = '1020';

# Delete a supplier
DELETE FROM suppliers
WHERE supplier_id = 10;
```

SOURCE CODE

File name: *inventory_db.sql*

```
# Creacting a New DataBase for inventory system
CREATE DATABASE IF NOT EXISTS inventory_db;
USE inventory_db;

# Creating the supplier's Table
DROP TABLE IF EXISTS suppliers;
CREATE TABLE suppliers (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    supplier_name VARCHAR(255) NOT NULL,
    contact VARCHAR(100)
);

# This Creates the [items] table with a foreign key to [suppliers]
DROP TABLE IF EXISTS items;
CREATE TABLE items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    barcode VARCHAR(50) UNIQUE,
    name VARCHAR(255) NOT NULL,
    quantity INT,
    price FLOAT,
    brand VARCHAR(255),
    supplier_id INT,
    FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)
);

# Inserting 10 Records in the Table
INSERT INTO suppliers (supplier_name, contact) VALUES
('Stationery World', 'contact@stationeryworld.com'),
('Office Supplies Co.', 'info@officesupplies.com'),
('School Essentials', 'support@schoolessentials.com'),
('Paper & Ink', 'sales@paperandink.com'),
('Fast Delivery', 'service@fastdelivery.com'),
('Bright Ideas', 'hello@brightideas.com'),
('Creative Supplies', 'contact@creativesupplies.com'),
('Mega Office', 'info@megaoffice.com'),
('Supplies Plus', 'support@suppliesplus.com'),
('Quality Goods', 'sales@qualitygoods.com');

# Insert 10 records into [items] while assigning supplier_id for each
INSERT INTO items (barcode, name, quantity, price, brand, supplier_id) VALUES
('1001', 'Pencil', 50, 0.50, '2B', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Stationery World')),
('1002', 'Eraser', 40, 0.20, 'CleanCo', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Office Supplies Co.')),
('1003', 'Notebook', 30, 2.50, 'NotePro', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'School Essentials')),
('1004', 'Marker', 20, 1.00, 'ColorMax', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Paper & Ink')),
('1005', 'Pen', 100, 0.75, 'WriteWell', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Fast Delivery')),
('1006', 'Ruler', 25, 1.50, 'MeasureUp', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Bright Ideas')),
('1007', 'Scissors', 15, 3.00, 'CutRight', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Creative
Supplies')),
('1008', 'Glue', 35, 1.25, 'StickIt', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Mega Office')),
('1009', 'Stapler', 10, 4.00, 'FastFix', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Supplies Plus')),
('1010', 'Highlighter', 60, 0.90, 'BrightMark', (SELECT supplier_id FROM suppliers WHERE supplier_name = 'Quality
Goods'));


# Add a new supplier
INSERT INTO suppliers (supplier_name, contact)
VALUES ('Acme Co.', 'acme@example.com');

# Add a new item
INSERT INTO items (barcode, name, quantity, price, brand, supplier_id)
VALUES ('1027', 'Widget A', 50, 19.99, 'BrandX', 25);



# List all suppliers
SELECT supplier_id, supplier_name, contact
FROM suppliers;

# List only active suppliers (those with ≥1 item)
SELECT DISTINCT
    s.supplier_id,
    s.supplier_name,
    s.contact
FROM suppliers AS s
```

```
JOIN items AS i
    ON s.supplier_id = i.supplier_id;

# Checking ALL items & Suppliers in combined table
SELECT
    i.id,
    i.barcode,
    i.name,
    i.quantity,
    i.price,
    i.brand,
    s.supplier_name,
    s.contact
FROM items i JOIN suppliers s ON i.supplier_id = s.supplier_id;

# Update a supplier’s contact
UPDATE suppliers
SET contact = 'newcontact@example.com'
WHERE supplier_id = 1;

# Update an item’s quantity and price
UPDATE items
SET quantity = 75,
    price = 17.49
WHERE barcode = '1020';

# Delete an item by barcode
DELETE FROM items
WHERE barcode = '1020';

# Delete a supplier
DELETE FROM suppliers
WHERE supplier_id = 10;
```

File name: *theme.py*

```
# theme.py
import qdarktheme

_current = "light"

def setup(app):
    """
    Apply the current theme (light or dark) to the QApplication.
    Uses setup_theme() which applies palette, stylesheet, and icons.
    """
    qdarktheme.setup_theme(_current)

def toggle(app):
    """
    Toggle between light and dark themes at runtime.
    """
    global _current
    _current = "dark" if _current == "light" else "light"
    qdarktheme.setup_theme(_current)
```

File name: *main.py*

```
import sys
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QWidget,
    QVBoxLayout, QAction, QMenuBar
)
import theme
from inventory import InventoryWindow
from supplier_manager import SupplierManagerWindow

class MainMenu(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Main Menu")

        # Making it big and docked at right in the middle
        self.setGeometry(500, 300, 720, 440)

        # Central widget + layout
        cw = QWidget()
        self.setCentralWidget(cw)
        layout = QVBoxLayout(cw)
        layout.setContentsMargins(50, 50, 50, 50)
        layout.setSpacing(20)

        # Buttons
        self.btnInventory = QPushButton("Manage Inventory")
        self.btnSuppliers = QPushButton("Manage Suppliers")
        self.btnExit = QPushButton("Exit")

        for btn in (self.btnInventory, self.btnSuppliers, self.btnExit):
            btn.setMinimumHeight(60)
            layout.addWidget(btn)

        # Button signals
        self.btnInventory.clicked.connect(self.open_inventory)
        self.btnSuppliers.clicked.connect(self.open_suppliers)
        self.btnExit.clicked.connect(self.close)

        # Menu bar with Theme toggle and Exit
        mb = QMenuBar(self)
        self.setMenuBar(mb)
        file_menu = mb.addMenu("File")
        exit_action = QAction("Exit", self)
        exit_action.triggered.connect(self.close)
        file_menu.addAction(exit_action)

        view = mb.addMenu("View")
        toggle = QAction("Toggle Theme", self)
        toggle.triggered.connect(lambda: theme.toggle(QApplication.instance()))
        view.addAction(toggle)

    def open_inventory(self):
        self.inv_win = InventoryWindow()
        self.inv_win.show()

    def open_suppliers(self):
        self.sup_win = SupplierManagerWindow()
        self.sup_win.show()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    theme.setup(app)
    w = MainMenu()
    w.show()
    sys.exit(app.exec_())
```

File name: *supplier_manager.py*

```
import sys, os
from PyQt5.QtWidgets import (
    QMainWindow, QApplication, QWidget, QVBoxLayout, QHBoxLayout,
    QTableWidget, QTableWidgetItem, QLineEdit, QPushButton,
    QLabel, QMessageBox, QAction, QMenuBar
)
import theme
from inventory import InventoryDB, DB_CONFIG
```

```python
class SupplierManagerWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Supplier Manager")
        self.resize(600, 400)

        # Theme toggle in menu
        mb = QMenuBar(self)
        self.setMenuBar(mb)
        view = mb.addMenu("View")
        t = QAction("Toggle Theme", self)
        t.triggered.connect(lambda: theme.toggle(QApplication.instance()))
        view.addAction(t)

        # Central layout
        cw = QWidget()
        self.setCentralWidget(cw)
        v = QVBoxLayout(cw)

        # Table
        self.tableSuppliers = QTableWidget(0, 3)
        self.tableSuppliers.setHorizontalHeaderLabels(["ID", "Name", "Contact"])
        v.addWidget(self.tableSuppliers)

        # Form inputs
        form = QHBoxLayout()
        form.addWidget(QLabel("Name:"))
        self.name_input = QLineEdit()
        form.addWidget(self.name_input)
        form.addWidget(QLabel("Contact:"))
        self.contact_input = QLineEdit()
        form.addWidget(self.contact_input)
        v.addLayout(form)

        # Buttons
        btns = QHBoxLayout()
        self.btnAdd = QPushButton("Add")
        self.btnUpdate = QPushButton("Update")
        self.btnDelete = QPushButton("Delete")
        self.btnRefresh = QPushButton("Refresh")
        for btn in (self.btnAdd, self.btnUpdate, self.btnDelete, self.btnRefresh):
            btns.addWidget(btn)
        v.addLayout(btns)
        # Use the same DB class and configuration as inventory.py
        self.db = InventoryDB(**DB_CONFIG)

        # Wire up signals
        self.btnAdd.clicked.connect(self.add_supplier)
        self.btnUpdate.clicked.connect(self.update_supplier)
        self.btnDelete.clicked.connect(self.delete_supplier)
        self.btnRefresh.clicked.connect(self.load_suppliers)
        self.tableSuppliers.currentCellChanged.connect(self.on_select)

        # Initial load
        self.load_suppliers()

    def add_supplier(self):
        name = self.name_input.text().strip()
        contact = self.contact_input.text().strip()
        if not name:
            return QMessageBox.warning(self, "Input Error", "Name required.")
        sid = self.db.create_supplier(name, contact)
        QMessageBox.information(self, "Added", f"Supplier ID {sid}")
        self.load_suppliers()

    def update_supplier(self):
        row = self.tableSuppliers.currentRow()
        if row < 0:
            return QMessageBox.warning(self, "Select", "Select a supplier first.")
        sid = int(self.tableSuppliers.item(row, 0).text())
        name = self.name_input.text().strip()
        contact = self.contact_input.text().strip()
        self.db.update_supplier(sid, name=name, contact=contact)
        QMessageBox.information(self, "Updated", "Supplier updated.")
        self.load_suppliers()

    def delete_supplier(self):
        row = self.tableSuppliers.currentRow()
        if row < 0:
            return QMessageBox.warning(self, "Select", "Select a supplier first.")
        sid = int(self.tableSuppliers.item(row, 0).text())
        self.db.delete_supplier(sid)
        QMessageBox.information(self, "Deleted", "Supplier deleted.")
        self.load_suppliers()

    def load_suppliers(self):
        recs = self.db.read_suppliers()
        self.tableSuppliers.setRowCount(0)
        for sid, name, contact in recs:
            r = self.tableSuppliers.rowCount()
            self.tableSuppliers.insertRow(r)
            self.tableSuppliers.setItem(r, 0, QTableWidgetItem(str(sid)))
            self.tableSuppliers.setItem(r, 1, QTableWidgetItem(name))
            self.tableSuppliers.setItem(r, 2, QTableWidgetItem(contact))
        self.tableSuppliers.resizeColumnsToContents()
        self.tableSuppliers.resizeRowsToContents()

    def on_select(self, row, col):
        if row < 0:
            return
        self.name_input.setText(self.tableSuppliers.item(row, 1).text())
        self.contact_input.setText(self.tableSuppliers.item(row, 2).text())


if __name__ == '__main__':
    app = QApplication(sys.argv)
    theme.setup(app)
    w = SupplierManagerWindow()
    w.show()
    sys.exit(app.exec_())
```

File name: *inventory.py*

```
import sys, os
from PyQt5 import uic
from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox, QTableWidgetItem, QAction, QMenuBar
from PyQt5.QtCore import Qt, QEvent
import pymysql
import theme

DB_CONFIG = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Password123456',
    'database': 'inventory_db'
}

class InventoryDB:

    def __init__(self, host, user, password, database):
        self.conn = pymysql.connect(
            host=host, user=user, password=password,
            database=database, cursorclass=pymysql.cursors.Cursor
        )

    def close(self):
        self.conn.close()

    def create_supplier(self, name, contact):

        with self.conn.cursor() as cur:

            cur.execute("""
                SELECT supplier_id FROM suppliers
                WHERE supplier_name = %s AND contact = %s
            """, (name, contact))
            row = cur.fetchone()

            if row:
                return row[0]

            cur.execute("""
                INSERT INTO suppliers (supplier_name, contact)
                VALUES (%s, %s)
            """, (name, contact))

        self.conn.commit()
        return cur.lastrowid

    def read_suppliers(self):

        with self.conn.cursor() as cur:
            cur.execute("SELECT supplier_id, supplier_name, contact FROM suppliers")

            return cur.fetchall()

    def read_active_suppliers(self):
        """Return suppliers that still have at least one item."""

        with self.conn.cursor() as cur:
            cur.execute("""
                SELECT DISTINCT s.supplier_id, s.supplier_name, s.contact
                FROM suppliers s
                JOIN items i ON s.supplier_id = i.supplier_id
            """)

            return cur.fetchall()
```

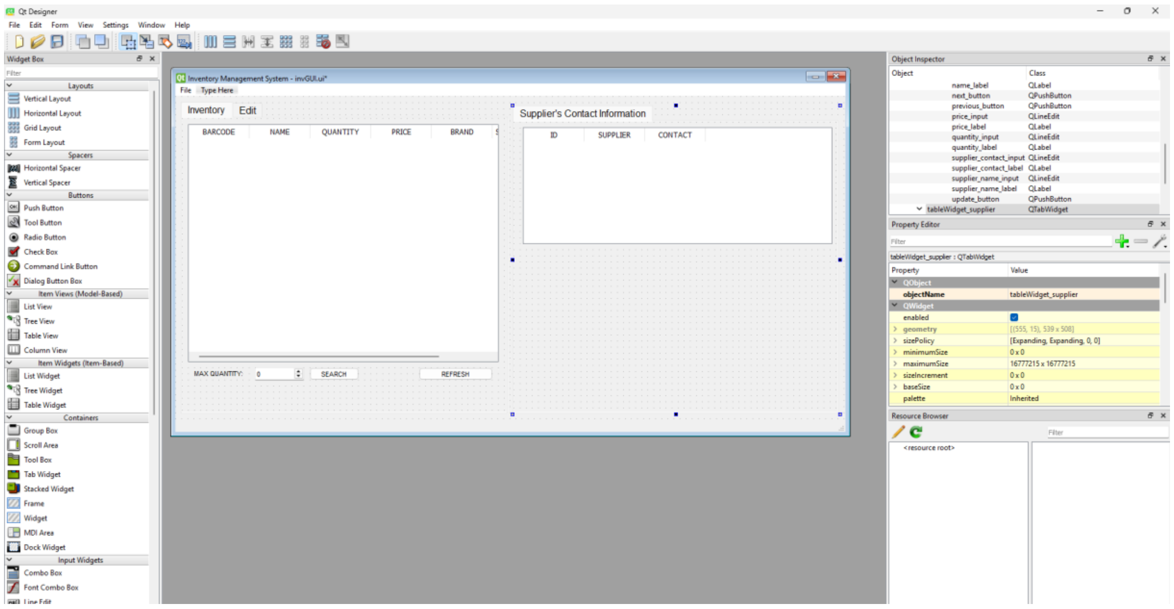
```python
def update_supplier(self, supplier_id, name=None, contact=None):
    fields = {}

    if name is not None:
        fields['supplier_name'] = name

    if contact is not None:
        fields['contact'] = contact

    if not fields:
        return

    cols = ", ".join(f"{k}=%s" for k in fields)
    vals = list(fields.values()) + [supplier_id]

    with self.conn.cursor() as cur:
        cur.execute(f"UPDATE suppliers SET {cols} WHERE supplier_id=%s", vals)

    self.conn.commit()

def delete_supplier(self, supplier_id):
    with self.conn.cursor() as cur:
        cur.execute("DELETE FROM suppliers WHERE supplier_id=%s", (supplier_id,))
    self.conn.commit()

def create_item(self, barcode, name, qty, price, brand, sup_id):
    with self.conn.cursor() as cur:
        cur.execute("SELECT barcode FROM items WHERE barcode=%s", (barcode,))
        if cur.fetchone():
            return False
        cur.execute(
            "INSERT INTO items (barcode,name,quantity,price,brand,supplier_id) "
            "VALUES (%s,%s,%s,%s,%s,%s)",
            (barcode, name, qty, price, brand, sup_id)
        )
    self.conn.commit()
    return True

def read_items(self):
    with self.conn.cursor() as cur:
        cur.execute(
            "SELECT i.barcode, i.name, i.quantity, i.price, i.brand, "
            "s.supplier_name FROM items i "
            "LEFT JOIN suppliers s ON i.supplier_id=s.supplier_id"
            )
            return cur.fetchall()

    def update_item(self, barcode, **fields):
        if not fields:
            return
        cols = ", ".join(f"{k}=%s" for k in fields)
        vals = list(fields.values()) + [barcode]
        with self.conn.cursor() as cur:
            cur.execute(f"UPDATE items SET {cols} WHERE barcode=%s", vals)
        self.conn.commit()

    def delete_item(self, barcode):
        with self.conn.cursor() as cur:
            cur.execute("DELETE FROM items WHERE barcode=%s", (barcode,))
        self.conn.commit()

# ─ GUI layer ─────────────────

class InventoryWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi(os.path.join(os.path.dirname(__file__), 'invGUI.ui'), self)

        # Theme toggle
        mb = QMenuBar(self)
        self.setMenuBar(mb)
        view = mb.addMenu("View")
        toggle = QAction("Toggle Theme", self)
        toggle.triggered.connect(lambda: theme.toggle(QApplication.instance()))
        view.addAction(toggle)



        # Spinbox range
        if hasattr(self,'quantity_filter_input'):
            self.quantity_filter_input.setMaximum(1_000_000)

        # DB
        self.db = InventoryDB(**DB_CONFIG)
        self.current_row = 0

        # Field navigation
        self.barcode_input.returnPressed.connect(self.name_input.setFocus)
        self.name_input.returnPressed.connect(self.quantity_input.setFocus)
        self.quantity_input.returnPressed.connect(self.price_input.setFocus)
        self.price_input.returnPressed.connect(self.brand_input.setFocus)
        self.brand_input.returnPressed.connect(self.supplier_name_input.setFocus)
        self.supplier_name_input.returnPressed.connect(self.supplier_contact_input.setFocus)
        self.supplier_contact_input.returnPressed.connect(self.create_item)

        # Buttons
        self.add_or_create_button.clicked.connect(self.create_item)
        self.update_button.clicked.connect(self.update_item)
        self.delete_button.clicked.connect(self.delete_item)
        self.refresh_button.clicked.connect(lambda: self.load_items(None))
        self.previous_button.clicked.connect(self.on_previous)
        self.next_button.clicked.connect(self.on_next)
        self.first_button.clicked.connect(self.on_first)
        self.last_button.clicked.connect(self.on_last)

        # Table
        self.table.currentCellChanged.connect(self.on_item_select)
        self.table.installEventFilter(self)
        if hasattr(self,'quantity_filter_input'):
            self.quantity_filter_input.editingFinished.connect(self.apply_filter)

        # Initialize
        self.load_items(None)
        self.load_suppliers()
        self.table.setFocus()
        if self.table.rowCount(): self.table.selectRow(0)

    def create_item(self):

        if not (b:=self.barcode_input.text().strip()) or not (n:=self.name_input.text().strip()) or not
(qt:=self.quantity_input.text().strip()):
            return QMessageBox.warning(self,"Input Error","Barcode, Name & Quantity required.")

        try:
            q=int(qt); p=float(self.price_input.text() or 0.0)
            br=self.brand_input.text().strip()
            sup=self.supplier_name_input.text().strip()
            c=self.supplier_contact_input.text().strip()

        except Exception as e:
            return QMessageBox.warning(self,"Input Error",str(e))

        sid=self.db.create_supplier(sup,c)

        if not self.db.create_item(b,n,q,p,br,sid):
            return QMessageBox.warning(self,"Duplicate","Barcode exists.")
        QMessageBox.information(self,"Success","Item added.")
        self._clear_inputs()
        self.load_items(None); self.load_suppliers()
        last=self.table.rowCount()-1

        if last>=0: self.table.setCurrentCell(last,0)
        self.table.setFocus()

    def apply_filter(self):
        self.load_items(self.quantity_filter_input.value())

    def load_items(self, max_q):
        self.table.setRowCount(0)
        for rec in self.db.read_items():
            if max_q is None or rec[2]<=max_q:
                r=self.table.rowCount(); self.table.insertRow(r)
                for c,v in enumerate(rec):
                    self.table.setItem(r,c,QTableWidgetItem(str(v)))
        row=min(self.current_row,self.table.rowCount()-1)
        if row>=0:
            self.table.setCurrentCell(row,0)
            self.table.scrollToItem(self.table.item(row,0))

    def load_suppliers(self):
        self.tableWidget.setRowCount(0)
        for rec in self.db.read_active_suppliers():
            r=self.tableWidget.rowCount(); self.tableWidget.insertRow(r)
            for c,v in enumerate(rec):
                self.tableWidget.setItem(r,c,QTableWidgetItem(str(v)))
        self.tableWidget.resizeColumnsToContents()
        self.tableWidget.resizeRowsToContents()

    def on_item_select(self,row,col,*_):
        if row<0: return
        self.current_row=row
        self.barcode_input.setText(self.table.item(row,0).text())
        self.name_input.setText(self.table.item(row,1).text())
        self.quantity_input.setText(self.table.item(row,2).text())
        self.price_input.setText(self.table.item(row,3).text())
        self.brand_input.setText(self.table.item(row,4).text())
        self.supplier_name_input.setText(self.table.item(row,5).text())

    def on_first(self):
        if self.table.rowCount(): self.table.setCurrentCell(0,0)
```

```python
    def on_last(self):
        last=self.table.rowCount()-1
        if last>=0: self.table.setCurrentCell(last,0)
    def on_next(self):
        r,c=self.table.currentRow(),self.table.currentColumn()
        self.table.setCurrentCell(min(r+1,self.table.rowCount()-1),c)
    def on_previous(self):
        r,c=self.table.currentRow(),self.table.currentColumn()
        self.table.setCurrentCell(max(r-1,0),c)

    def update_item(self):
        b=self.barcode_input.text().strip()
        fields={}
        if self.name_input.text(): fields['name']=self.name_input.text().strip()
        if self.quantity_input.text(): fields['quantity']=int(self.quantity_input.text())
        if self.price_input.text(): fields['price']=float(self.price_input.text())
        if self.brand_input.text(): fields['brand']=self.brand_input.text().strip()
        if self.supplier_name_input.text():
            sid=self.db.create_supplier(
                self.supplier_name_input.text().strip(),
                self.supplier_contact_input.text().strip()
            )
            fields['supplier_id']=sid
        if fields:
            self.db.update_item(b,**fields)
            QMessageBox.information(self,"Updated","Item updated.")
            self._clear_inputs(); self.load_items(None); self.load_suppliers()
            self.table.setFocus()
        else:
            QMessageBox.warning(self,"No Changes","Modify at least one field.")

    def delete_item(self):
        b=self.barcode_input.text().strip()
        self.db.delete_item(b)
        QMessageBox.information(self,"Deleted","Item deleted.")
        self._clear_inputs(); self.load_items(None); self.load_suppliers()
        self.table.setFocus()

    def _clear_inputs(self):
        for w in (self.barcode_input,self.name_input,self.quantity_input,
                  self.price_input,self.brand_input,
                  self.supplier_name_input,self.supplier_contact_input):
            w.clear()

    def update_supplier(self, supplier_id, name=None, contact=None):
        """Update name and/or contact of a supplier by ID."""
        fields = {}
        if name is not None:
            fields['supplier_name'] = name
        if contact is not None:
            fields['contact'] = contact
        if not fields:
            return
        cols = ", ".join(f"{k}=%s" for k in fields)
        vals = list(fields.values()) + [supplier_id]
        with self.conn.cursor() as cur:
            cur.execute(f"UPDATE suppliers SET {cols} WHERE supplier_id=%s", vals)
        self.conn.commit()

    def delete_supplier(self, supplier_id):
        """Delete a supplier by ID."""
        with self.conn.cursor() as cur:
            cur.execute("DELETE FROM suppliers WHERE supplier_id=%s", (supplier_id,))
        self.conn.commit()

    def eventFilter(self,src,event):
        if src is self.table and event.type()==QEvent.KeyPress:
            r,c=self.table.currentRow(),self.table.currentColumn()
            if event.key()==Qt.Key_Right:
                self.table.setCurrentCell(min(r+1,self.table.rowCount()-1),c);return True
            if event.key()==Qt.Key_Left:
                self.table.setCurrentCell(max(r-1,0),c);return True
        return super().eventFilter(src,event)

    def keyPressEvent(self,event):
        r,c=self.table.currentRow(),self.table.currentColumn()
        if event.key()==Qt.Key_Down:
            self.table.setCurrentCell(min(r+1,self.table.rowCount()-1),c)
        elif event.key()==Qt.Key_Up:
            self.table.setCurrentCell(max(r-1,0),c)
        else:
            super().keyPressEvent(event)

    def closeEvent(self,ev):
        self.db.close(); super().closeEvent(ev)
```

SCREENSHOTS OF OUTPUT

AGUIRRE, PAUL VINCENT S. 05/10/2025
M001— CP102 – Computer Programming 2 PROF: RODRIGO JR. C. BELLEZA

Making the GUI in Qt Designer


Qt Designer
File Edit Form View Settings Window Help
Widget Box
Inventory Management System - invGUI.ui*
Inventory Edit
BARCODE NAME QUANTITY PRICE BRAND
MAX QUANTITY: 0 SEARCH REFRESH
Supplier's Contact Information
ID SUPPLIER CONTACT
Resource Browser
<resource root>

Object Inspector

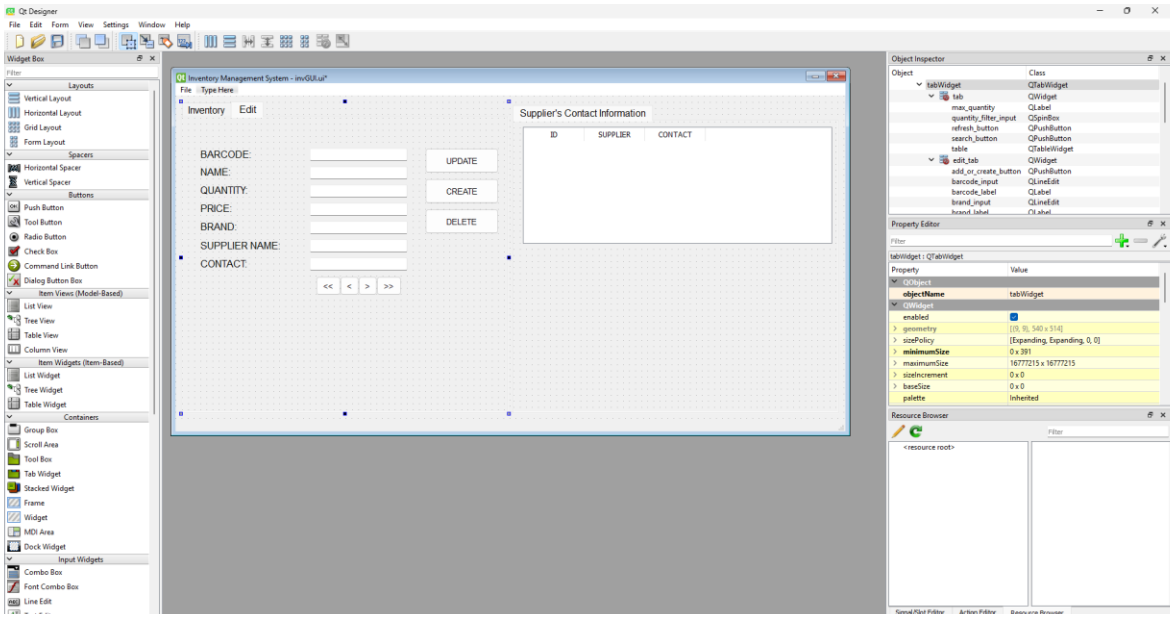
Object	Class
name_label	QLabel
next_button	QPushButton
previous_button	QPushButton
price_input	QLineEdit
price_label	QLabel
quantity_input	QLineEdit
quantity_label	QLabel
supplier_contact_input	QLineEdit
supplier_contact_label	QLabel
supplier_name_input	QLineEdit
supplier_name_label	QLabel
update_button	QPushButton
tableWidget_supplier	QTabWidget

Property Editor

tableWidget_supplier : QTabWidget

Property	Value
QObject	
objectName	tableWidget_supplier
QWidget	
enabled	✓
geometry	[(555, 15), 539 x 508]
sizePolicy	[Expanding, Expanding, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited

Inventory Window



Object Inspector

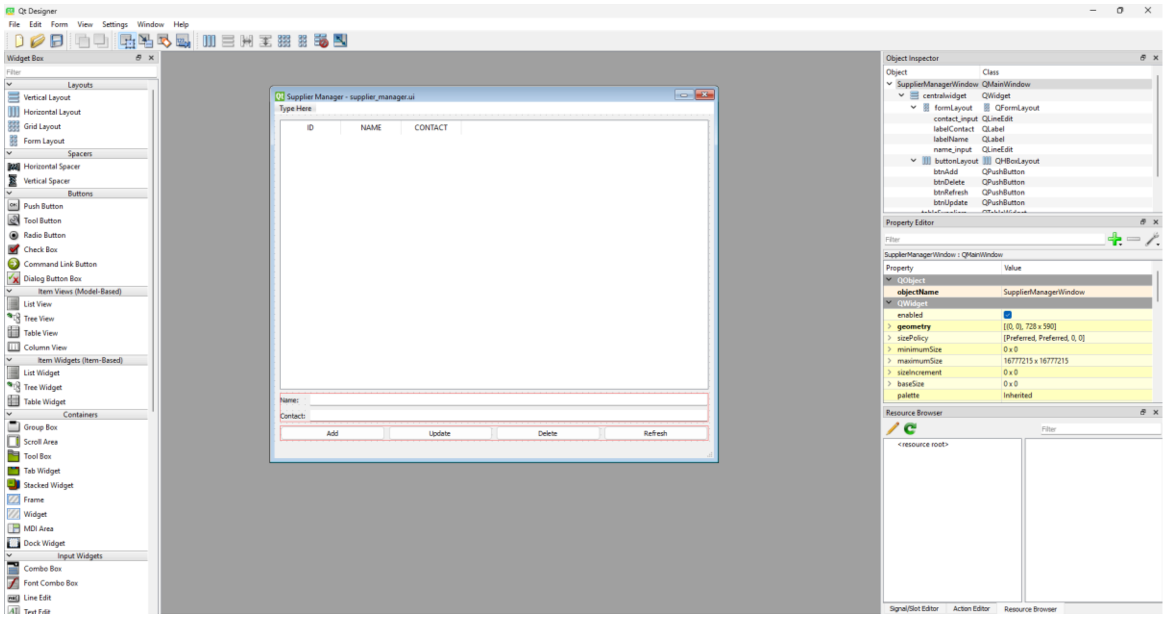
Object	Class
tabWidget	QTabWidget
tab	QWidget
max_quantity	QLabel
quantity_filter_input	QSpinBox
refresh_button	QPushButton
search_button	QPushButton
table	QTableWidget
edit_tab	QWidget
add_or_create_button	QPushButton
barcode_input	QLineEdit
barcode_label	QLabel
brand_input	QLineEdit
brand_label	QLabel

Property Editor

tabWidget : QTabWidget

Property	Value
QObject	
objectName	tabWidget
QWidget	
enabled	✓
geometry	[(9, 9), 540 x 514]
sizePolicy	[Expanding, Expanding, 0, 0]
minimumSize	0 x 391
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited

Supplier Window



Object Inspector

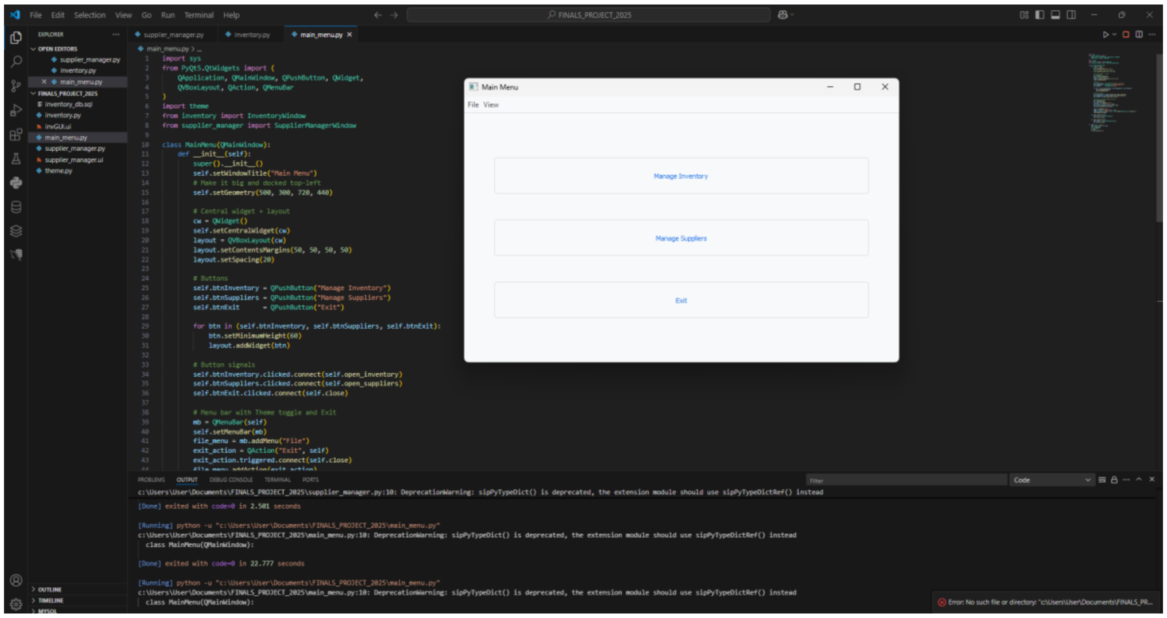
Object	Class
SupplierManagerWindow	QMainWindow
centralwidget	QWidget
formLayout	QFormLayout
contact_input	QLineEdit
labelContact	QLabel
labelName	QLabel
name_input	QLineEdit
buttonLayout	QHBoxLayout
btnAdd	QPushButton
btnDelete	QPushButton
btnRefresh	QPushButton
btnUpdate	QPushButton

Property Editor

SupplierManagerWindow : QMainWindow

Property	Value
QObject	
objectName	SupplierManagerWindow
QWidget	
enabled	✓
geometry	[(0, 0), 728 x 590]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited

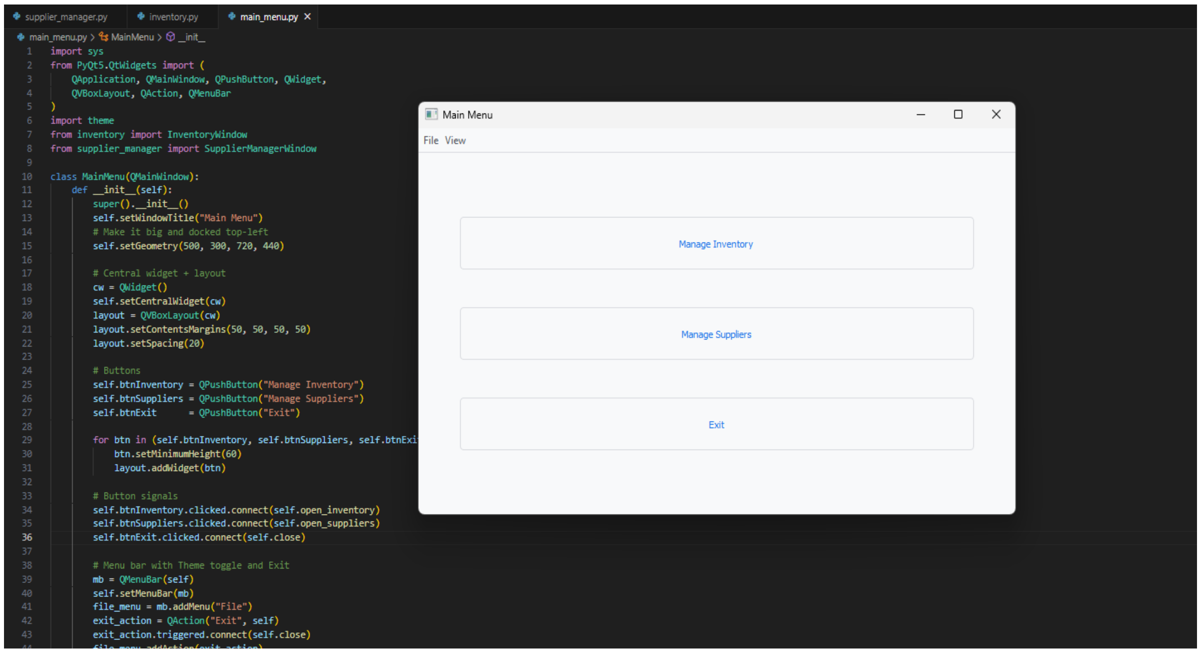
Main Window



```python
import sys
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QWidget,
    QVBoxLayout, QAction, QMenuBar
)
import theme
from inventory import InventoryWindow
from supplier_manager import SupplierManagerWindow

class MainMenu(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Main Menu")
        # Make it big and docked top-left
        self.setGeometry(500, 300, 720, 440)

        # Central widget + layout
        cw = QWidget()
        self.setCentralWidget(cw)
        layout = QVBoxLayout(cw)
        layout.setContentsMargins(50, 50, 50, 50)
        layout.setSpacing(20)

        # Buttons
        self.btnInventory = QPushButton("Manage Inventory")
        self.btnSuppliers = QPushButton("Manage Suppliers")
        self.btnExit      = QPushButton("Exit")

        for btn in (self.btnInventory, self.btnSuppliers, self.btnExit):
            btn.setMinimumHeight(60)
            layout.addWidget(btn)

        # Button signals
        self.btnInventory.clicked.connect(self.open_inventory)
        self.btnSuppliers.clicked.connect(self.open_suppliers)
        self.btnExit.clicked.connect(self.close)

        # Menu bar with Theme toggle and Exit
        mb = QMenuBar(self)
        self.setMenuBar(mb)
        file_menu = mb.addMenu("File")
        exit_action = QAction("Exit", self)
        exit_action.triggered.connect(self.close)
```

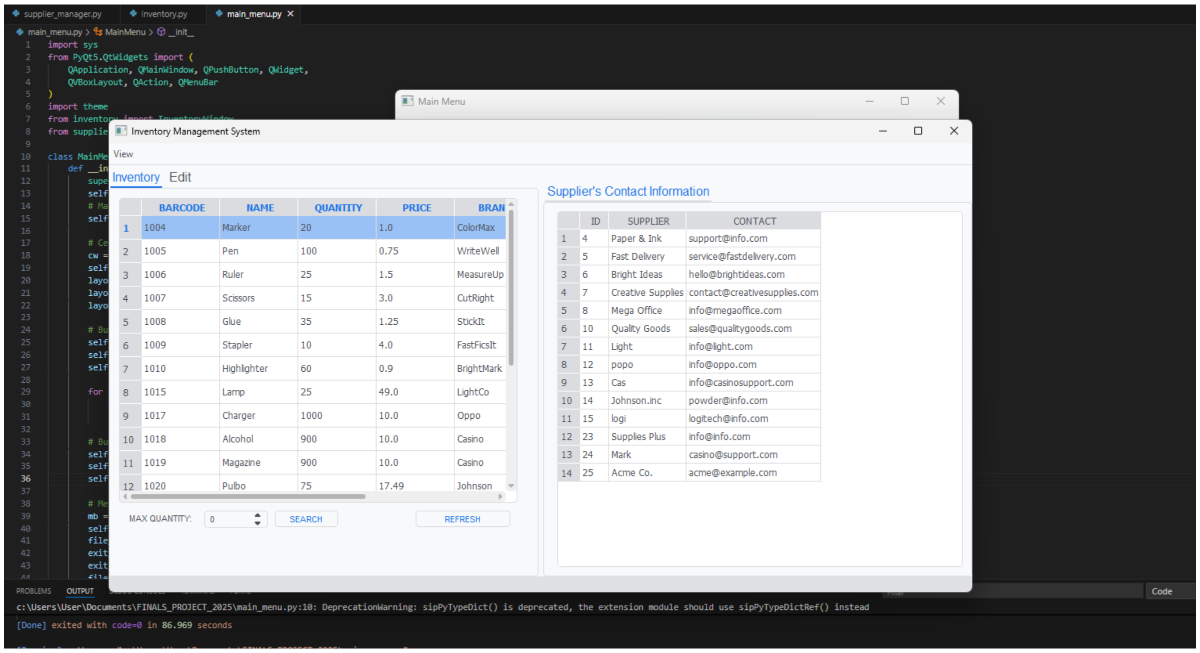
```
c:\Users\User\Documents\FINALS_PROJECT_2025\supplier_manager.py:10: DeprecationWarning: sipPyTypeDict() is deprecated, the extension module should use sipPyTypeDictRef() instead

[Done] exited with code=0 in 2.501 seconds

[Running] python -u "c:\Users\User\Documents\FINALS_PROJECT_2025\main_menu.py"
c:\Users\User\Documents\FINALS_PROJECT_2025\main_menu.py:10: DeprecationWarning: sipPyTypeDict() is deprecated, the extension module should use sipPyTypeDictRef() instead
  class MainMenu(QMainWindow):

[Done] exited with code=0 in 22.777 seconds

[Running] python -u "c:\Users\User\Documents\FINALS_PROJECT_2025\main_menu.py"
c:\Users\User\Documents\FINALS_PROJECT_2025\main_menu.py:10: DeprecationWarning: sipPyTypeDict() is deprecated, the extension module should use sipPyTypeDictRef() instead
  class MainMenu(QMainWindow):
```

Sample Output



```python
import sys
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QWidget,
    QVBoxLayout, QAction, QMenuBar
)
import theme
from inventory import InventoryWindow
from supplier_manager import SupplierManagerWindow

class MainMenu(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Main Menu")
        # Make it big and docked top-left
        self.setGeometry(500, 300, 720, 440)

        # Central widget + layout
        cw = QWidget()
        self.setCentralWidget(cw)
        layout = QVBoxLayout(cw)
        layout.setContentsMargins(50, 50, 50, 50)
        layout.setSpacing(20)

        # Buttons
        self.btnInventory = QPushButton("Manage Inventory")
        self.btnSuppliers = QPushButton("Manage Suppliers")
        self.btnExit      = QPushButton("Exit")

        for btn in (self.btnInventory, self.btnSuppliers, self.btnExi
            btn.setMinimumHeight(60)
            layout.addWidget(btn)

        # Button signals
        self.btnInventory.clicked.connect(self.open_inventory)
        self.btnSuppliers.clicked.connect(self.open_suppliers)
        self.btnExit.clicked.connect(self.close)

        # Menu bar with Theme toggle and Exit
        mb = QMenuBar(self)
        self.setMenuBar(mb)
        file_menu = mb.addMenu("File")
        exit_action = QAction("Exit", self)
        exit_action.triggered.connect(self.close)
```

Main Window



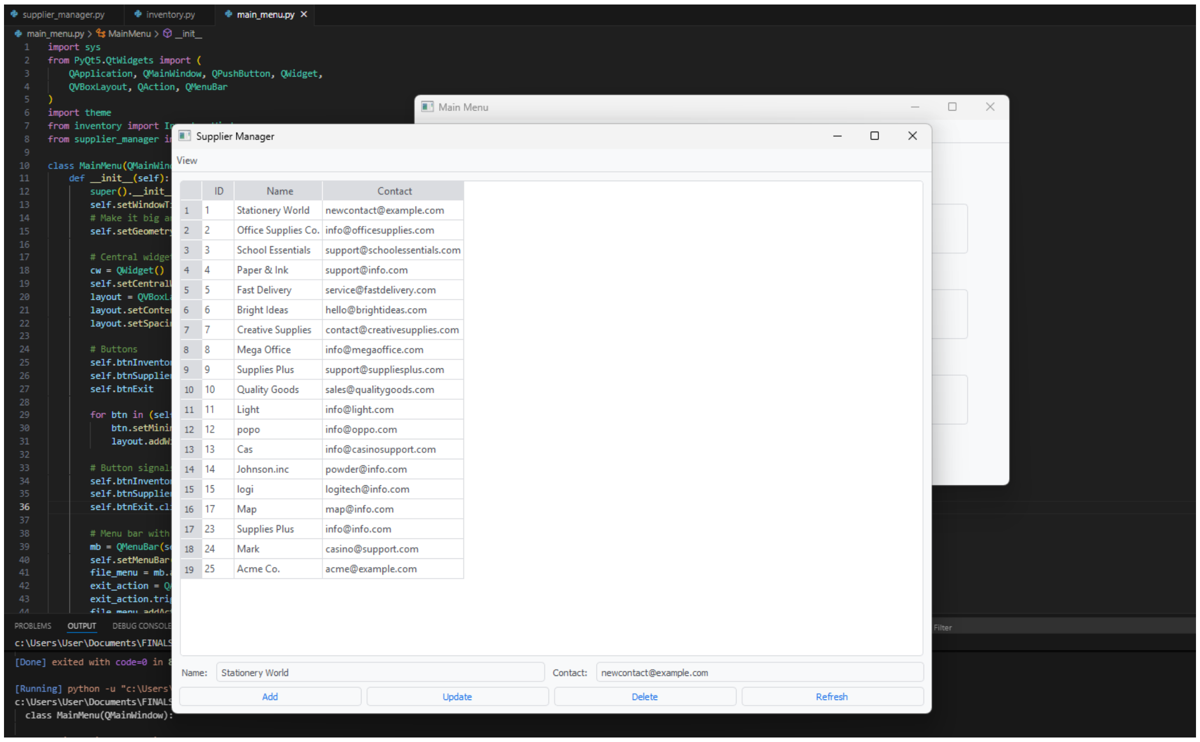
	BARCODE	NAME	QUANTITY	PRICE	BRAN
1	1004	Marker	20	1.0	ColorMax
2	1005	Pen	100	0.75	WriteWell
3	1006	Ruler	25	1.5	MeasureUp
4	1007	Scissors	15	3.0	CutRight
5	1008	Glue	35	1.25	StickIt
6	1009	Stapler	10	4.0	FastFicsIt
7	1010	Highlighter	60	0.9	BrightMark
8	1015	Lamp	25	49.0	LightCo
9	1017	Charger	1000	10.0	Oppo
10	1018	Alcohol	900	10.0	Casino
11	1019	Magazine	900	10.0	Casino
12	1020	Pulbo	75	17.49	Johnson

MAX QUANTITY: 0

	ID	SUPPLIER	CONTACT
1	4	Paper & Ink	support@info.com
2	5	Fast Delivery	service@fastdelivery.com
3	6	Bright Ideas	hello@brightideas.com
4	7	Creative Supplies	contact@creativesupplies.com
5	8	Mega Office	info@megaoffice.com
6	10	Quality Goods	sales@qualitygoods.com
7	11	Light	info@light.com
8	12	popo	info@oppo.com
9	13	Cas	info@casinosupport.com
10	14	Johnson.inc	powder@info.com
11	15	logi	logitech@info.com
12	23	Supplies Plus	info@info.com
13	24	Mark	casino@support.com
14	25	Acme Co.	acme@example.com

```
c:\Users\User\Documents\FINALS_PROJECT_2025\main_menu.py:10: DeprecationWarning: sipPyTypeDict() is deprecated, the extension module should use sipPyTypeDictRef() instead

[Done] exited with code=0 in 86.969 seconds
```

Inventory Window



```
import sys
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QWidget,
    QVBoxLayout, QAction, QMenuBar
)
import theme
from inventory import I
from supplier_manager i

class MainMenu(QMainWin
    def __init__(self):
        super().__init_
        self.setWindowT
        # Make it big a
        self.setGeometr

        # Central widge
        cw = QWidget()
        self.setCentral
        layout = QVBoxL
        layout.setConte
        layout.setSpaci

        # Buttons
        self.btnInvento
        self.btnSupplie
        self.btnExit

        for btn in (sel
            btn.setMini
            layout.addW

        # Button signal
        self.btnInvento
        self.btnSupplie
        self.btnExit.cl

        # Menu bar with
        mb = QMenuBar(s
        self.setMenuBar
        file_menu = mb.
        exit_action = Q
        exit_action.tri
```

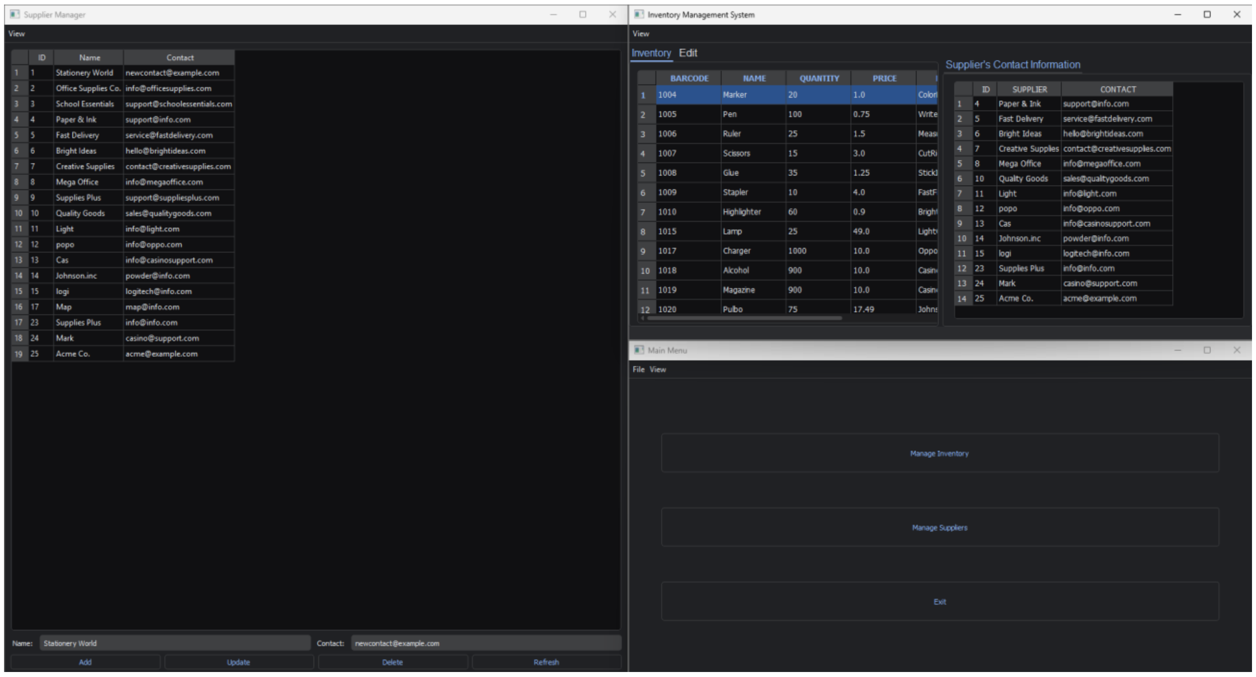
```
c:\Users\User\Documents\FINALS

[Done] exited with code=0 in 8

[Running] python -u "c:\Users\
c:\Users\User\Documents\FINALS
  class MainMenu(QMainWindow):
```

	ID	Name	Contact
1	1	Stationery World	newcontact@example.com
2	2	Office Supplies Co.	info@officesupplies.com
3	3	School Essentials	support@schoolessentials.com
4	4	Paper & Ink	support@info.com
5	5	Fast Delivery	service@fastdelivery.com
6	6	Bright Ideas	hello@brightideas.com
7	7	Creative Supplies	contact@creativesupplies.com
8	8	Mega Office	info@megaoffice.com
9	9	Supplies Plus	support@suppliesplus.com
10	10	Quality Goods	sales@qualitygoods.com
11	11	Light	info@light.com
12	12	popo	info@oppo.com
13	13	Cas	info@casinosupport.com
14	14	Johnson.inc	powder@info.com
15	15	logi	logitech@info.com
16	17	Map	map@info.com
17	23	Supplies Plus	info@info.com
18	24	Mark	casino@support.com
19	25	Acme Co.	acme@example.com

Name: Stationery World Contact: newcontact@example.com

Add Update Delete Refresh

Supplier Window



	ID	Name	Contact
1	1	Stationery World	newcontact@example.com
2	2	Office Supplies Co.	info@officesupplies.com
3	3	School Essentials	support@schoolessentials.com
4	4	Paper & Ink	support@info.com
5	5	Fast Delivery	service@fastdelivery.com
6	6	Bright Ideas	hello@brightideas.com
7	7	Creative Supplies	contact@creativesupplies.com
8	8	Mega Office	info@megaoffice.com
9	9	Supplies Plus	support@suppliesplus.com
10	10	Quality Goods	sales@qualitygoods.com
11	11	Light	info@light.com
12	12	popo	info@oppo.com
13	13	Cas	info@casinosupport.com
14	14	Johnson.inc	powder@info.com
15	15	logi	logitech@info.com
16	17	Map	map@info.com
17	23	Supplies Plus	info@info.com
18	24	Mark	casino@support.com
19	25	Acme Co.	acme@example.com

Name: Stationery World Contact: newcontact@example.com

Add Update Delete Refresh

	BARCODE	NAME	QUANTITY	PRICE	
1	1004	Marker	20	1.0	Colori
2	1005	Pen	100	0.75	Write
3	1006	Ruler	25	1.5	Meas
4	1007	Scissors	15	3.0	CutRi
5	1008	Glue	35	1.25	StickI
6	1009	Stapler	10	4.0	FastF
7	1010	Highlighter	60	0.9	Bright
8	1015	Lamp	25	49.0	Light
9	1017	Charger	1000	10.0	Oppo
10	1018	Alcohol	900	10.0	Casin
11	1019	Magazine	900	10.0	Casin
12	1020	Pulbo	75	17.49	Johns

Supplier's Contact Information

	ID	SUPPLIER	CONTACT
1	4	Paper & Ink	support@info.com
2	5	Fast Delivery	service@fastdelivery.com
3	6	Bright Ideas	hello@brightideas.com
4	7	Creative Supplies	contact@creativesupplies.com
5	8	Mega Office	info@megaoffice.com
6	10	Quality Goods	sales@qualitygoods.com
7	11	Light	info@light.com
8	12	popo	info@oppo.com
9	13	Cas	info@casinosupport.com
10	14	Johnson.inc	powder@info.com
11	15	logi	logitech@info.com
12	23	Supplies Plus	info@info.com
13	24	Mark	casino@support.com
14	25	Acme Co.	acme@example.com

Dark Mode Window

*****Please Refer to README.txt for detailed explanation of how every button in the application works. Thank you!***

AGUIRRE, PAUL VINCENT S. 05/10/2025
M001— CP102 – Computer Programming 2 PROF: RODRIGO JR. C. BELLEZA

REFLECTION

I wouldn't call myself a great programmer—every time I tried to make this program work, I ran into tracebacks or errors that left me so frustrated I had to turn my attention to other tasks, bringing my progress to a halt. This happened more times than I can count: missing imports, mismatched function parameters, and elusive bugs that seemed to pop up just when I thought I'd nailed the logic. The constant cycle of write-test-debug felt overwhelming at times, and I found myself questioning whether I would ever move beyond these roadblocks. Eventually, after some time away from coding, I decided to use one of my old projects—my inventory system—which I had built a month ago for an exam. Though it felt like taking the easy route, it meant I could stand on familiar ground and dedicate my energy to mastering two new skills: building a graphical user interface and connecting it to my program and database.

Diving into GUI development turned out to be surprisingly rewarding. I found the process exciting—especially the moment when a button finally triggered the exact function I intended. Seeing a window render correctly on the first try, or having a dropdown populate dynamically with live data, offered a sense of accomplishment I hadn't felt with my previous command-line interfaces. While design isn't my strongest suit, I aimed for simplicity and user-friendliness by choosing clean layouts, clear labels, and consistent spacing. I even added a theme switch so users can toggle between light and dark modes, which introduced me to the challenge of maintaining readability and color contrast across different styles. To guide my learning, I owe thanks to Professor Dean and Educational Youtubers. Professor Dean taught me and my classmates at school how a GUI works and many more, On YouTube, Coding With Tim introduced me to PyQt fundamentals and demonstrated how to structure and link a GUI, while The Power of Machine Learning & Data Science they taught me how to code the GUI too but their design inspired me to do it as well. From Code First with Hana, I learned the basics of loading data from a database. Beyond it all, I browsed Stack Overflow and GitHub repositories—not to copy code, but to see how experienced developers tackled similar challenges, from thread management to error handling—and to learn efficient ways to query a SQLite database.

Whenever I hit a roadblock, I took a step back to rest and then returned with fresh eyes. Turning mistakes into a personal knowledge base. Sometimes I even consulted AI tools to break down complex concepts into digestible bullet points, which helped me study more efficiently and revealed new perspectives on optimization. This iterative cycle of building, reflecting, and refining gradually boosted my confidence and deepened my understanding of GUI frameworks.

In Conclusion, this project taught me that encountering errors is simply a natural part of the learning process. By combining past work with targeted, new skills, I was able to push through frustration and deliver a functional, intuitive application. Along the way, I improved not only my technical abilities—such as event handling, database integration, and responsive design—but also my resilience and problem-solving mindset. I will always move forward with conviction and determination to achieve my goal, No matter what it takes.

APPENDICES

Instructions

Performance Indicator:
Develop an object-oriented Python application that uses either **PyQt6** or **Tkinter** for its graphical user interface (GUI). The program must include a **minimum of three windows/forms**, including a main menu, and implement **full CRUD (Create, Read, Update, Delete)** operations on a **MySQL database**. The database should consist of **at least two related tables** using **Structured Query Language (SQL)**.

Final Course Output Documentation Requirements:
Submit a PDF document containing the following sections:

1. **System Description**
   - Provide an overview of your system, including its purpose and intended users.
2. **List of Tables**
   - Describe each database table, including columns, data types, constraints (e.g., NOT NULL, PRIMARY KEY, FOREIGN KEY).
3. **SQL Scripts**
   - Include all SQL scripts used to:
     - Create the database and tables
     - Add, edit, delete, and search records
4. **Complete Python Source Code**
   - Submit the full source code of your application.
5. **Screenshots of Output**
   - Provide screenshots of all forms/windows in your program,
   - clearly showing the GUI layout and functionality.
6. **Learning Reflection**
   - Reflect on your learning experience in the CP102 course.
   - Discuss key takeaways, challenges you faced and how you overcame them, and areas where you think you can improve in computer programming.

Submission Requirements:

- A **PDF file** containing the final course output documentation.
- A **ZIP file** containing your project files, including:
  - .ui file (if applicable for GUI design)
  - .py file(s) (Python source code)
  - .sql file (SQL script for database setup)