**Step-by-Step Algorithm**

**1. Initialize the Program**

* Step: Start the program, initialize any necessary data structures (e.g., load the inventory from a file).
* Code: This happens in `main.py` when `read\_inventory()` is called.

**2. Display Menu Options**

* Step: Display the main menu to the user with options like viewing inventory, selling furniture, ordering furniture, or exiting the program.
* Code: The `while True` loop in `main.py` handles displaying the menu and receiving user input.

**3. Handle User Input**

* Step: Based on the user's choice, perform the corresponding action (e.g., view inventory, sell furniture, order furniture).
* Code: In `main.py`, after the user makes a selection, the appropriate function (`display\_inventory`, `sell\_furniture`, `order\_furniture`) is called.

**4. View Inventory**

* Step: If the user chooses to view the inventory, display the current inventory.
* Code: The `display\_inventory()` function in `read\_furniture\_file.py` is called.

**5. Sell Furniture**

* Step: If the user chooses to sell furniture, prompt for customer details and update the inventory.
* Code: The `sell\_furniture()` function in `operation.py` handles this process, including calling `generate\_invoice()` to create an invoice and `write\_inventory()` to update the inventory file.

**6. Order Furniture**

* Step: If the user chooses to order furniture, prompt for employee details and update the inventory.
* Code: The `order\_furniture()` function in `operation.py` handles this process, also calling `generate\_invoice()` to create an order receipt and `write\_inventory()` to update the inventory file.

**7. Generate Invoice/Order Receipt**

* Step: Generate an invoice or order receipt, including details of the transaction.
* Code: The `generate\_invoice()` function in `operation.py` handles this, formatting and writing the receipt to a file.

**8. Exit the Program**

* Step: If the user chooses to exit, terminate the program.
* Code: The `break` statement in the `while True` loop of `main.py` exits the loop, thus ending the program.

**9. Update the Inventory File**

* Step: After selling or ordering furniture, update the inventory file with the new stock levels.
* Code: The `write\_inventory()` function in `write\_furniture\_file.py` writes the updated inventory back to the file.

**Pseudo Code**

1. **Main.py**

START Program

WHILE True:

DISPLAY menu options:

1. View Inventory

2. Sell Furniture

3. Order Furniture

4. Exit

READ user choice

IF choice == 1:

CALL display\_inventory()

ELSE IF choice == 2:

CALL sell\_furniture()

ELSE IF choice == 3:

CALL order\_furniture()

ELSE IF choice == 4:

PRINT "Exiting program."

BREAK loop

ELSE:

PRINT "Invalid choice. Please try again."

END Program

1. **Read\_furniture\_file.py**

**FUNCTION read\_inventory(filename):**

**CREATE empty list inventory**

**OPEN file with name filename FOR reading**

**FOR each line in file:**

**PARSE line to extract id, manufacturer, product, quantity, price**

**ADD dictionary with parsed values to inventory list**

**CLOSE file**

**RETURN inventory**

**END FUNCTION**

**FUNCTION display\_inventory(inventory):**

**PRINT "Available Furniture"**

**PRINT column headers**

**FOR each item in inventory:**

**PRINT item details (id, manufacturer, product, quantity, price)**

**END FUNCTION**

1. **write\_furniture\_file.py**

**FUNCTION write\_inventory(inventory, filename):**

**OPEN file with name filename FOR writing**

**FOR each item in inventory:**

**FORMAT item details as a string**

**WRITE string to file**

**CLOSE file**

**END FUNCTION**

1. **operations.py**

**FUNCTION sell\_furniture():**

**READ customer name**

**CREATE empty list transaction\_details**

**WHILE True:**

**CALL display\_inventory()**

**READ furniture\_id from user**

**READ quantity from user**

**FOR each item in inventory:**

**IF item.id == furniture\_id:**

**IF item.quantity >= quantity:**

**DECREASE item.quantity by quantity**

**ADD item details (id, manufacturer, product, quantity, price) to transaction\_details**

**ELSE:**

**PRINT "Insufficient stock"**

**ASK user if they want to sell another item**

**IF user does not want to continue:**

**BREAK loop**

**CALL generate\_invoice(transaction\_details, customer\_name, is\_sale=True)**

**CALL write\_inventory(inventory)**

**END FUNCTION**

**FUNCTION order\_furniture():**

**READ employee name**

**CREATE empty list transaction\_details**

**WHILE True:**

**CALL display\_inventory()**

**READ furniture\_id from user**

**READ quantity from user**

**FOR each item in inventory:**

**IF item.id == furniture\_id:**

**INCREASE item.quantity by quantity**

**ADD item details (id, manufacturer, product, quantity, price) to transaction\_details**

**ASK user if they want to order another item**

**IF user does not want to continue:**

**BREAK loop**

**CALL generate\_invoice(transaction\_details, employee\_name, is\_sale=False)**

**CALL write\_inventory(inventory)**

**END FUNCTION**

**FUNCTION generate\_invoice(transaction\_details, customer\_name=None, employee\_name=None, is\_sale=True):**

**GET current date and time as now**

**FORMAT date as date\_str**

**IF is\_sale:**

**SET filename to "invoice\_customerName\_date.txt"**

**ELSE:**

**SET filename to "order\_employeeName\_date.txt"**

**CLEAN filename to remove spaces**

**SET total\_amount to 0**

**OPEN file with name filename FOR writing**

**IF is\_sale:**

**WRITE customer name to file**

**ELSE:**

**WRITE employee name to file**

**WRITE date and time to file**

**WRITE column headers (ID, Manufacturer, Product, Quantity, Unit Price, Total) to file**

**FOR each item in transaction\_details:**

**CALCULATE line\_total as item.quantity \* item.price**

**ADD line\_total to total\_amount**

**WRITE item details and line\_total to file**

**IF is\_sale:**

**CALCULATE vat as total\_amount \* 0.13**

**CALCULATE total\_with\_vat as total\_amount + vat**

**SET shipping\_cost to 50**

**CALCULATE grand\_total as total\_with\_vat + shipping\_cost**

**WRITE total\_amount, VAT, total\_with\_vat, shipping\_cost, and grand\_total to file**

**ELSE:**

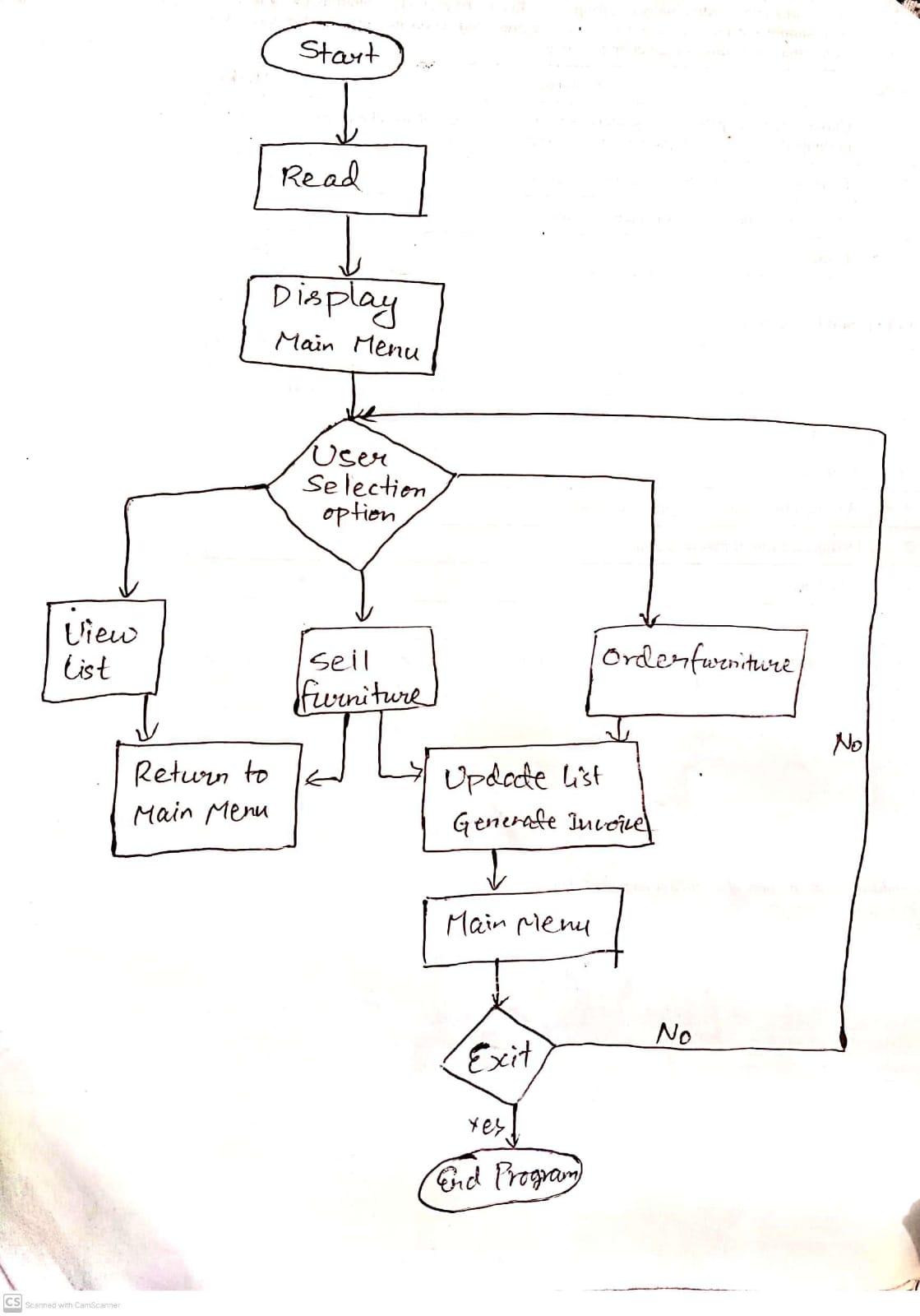
**WRITE total\_amount to file**

**CLOSE file**

**PRINT "Invoice generated"**

**END FUNCTION**

**Flow Chart**

****