Mid term project

✅ Project: Grocery List Manager
🛒 What it does:
Stores grocery items with prices.
Calculates total cost
Uses dictionary, list, loop, functions, thread, lambda, append, list comprehension, and operators — all very simply.

Made by:
Alishba riyaz (336310)
Hani javed (346291)

code:

```
import threading
# to use thread
# Step 1: Create a dictionary of items and their prices
grocery = {
"apple": 2,
"banana": 1,
"milk": 3
}
# Step 2: Function to calculate total cost
def calculate_total():
print("Calculating total...")
total = 0
for item, price in grocery.items():  # loop through dictionary
total += price  # using operator +
print("Total Cost:", total)

# Step 3: Run total calculation in a thread
t = threading.Thread(target=calculate_total)
t.start()
t.join()

# Step 4: Add a new item using append and list
new_items = []
new_items.append(("bread", 2))  # add tuple to list
new_items.append(("eggs", 4))

# Step 5: Add new items to dictionary using loop
for name, price in new_items:
grocery[name] = price  # add to dictionary

# Step 6: Use list comprehension to get all item names
item_names = [item for item in grocery]
print("Items:", item_names)

# Step 7: Use lambda to sort items by price
sorted_items = sorted(grocery.items(), key=lambda x: x[1])
print("Sorted by Price:", sorted_items)
```

✅ Output

```
Calculating total...
Total Cost: 6
Items: ['apple', 'banana', 'milk', 'bread', 'eggs']
Sorted by Price: [('banana', 1), ('apple', 2), ('bread', 2), ('milk', 3), ('eggs', 4)]
```

Concepts Used In Code

Operators

total += price

Loops

for item in grocery

List/Tuple/Dict

grocery, new_items

Functions

calculate_total()

Threads

threading.Thread

List Comprehension

[item for item in grocery]

Lambda

key=lambda x: x[1]

Append

new_items.append()