

LAUNDRY COUNTER

A MINI PROJECT REPORT

18CSC207J - ADVANCED PROGRAMMING
PRACTICE

Submitted by

AANYA GUPTA : RA2111003011729
SANSKRITI VAISHNAV:RA2111003011724

Under the guidance of

Dr.G.Balamurugan

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY



SRM

INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled "Laundry Counter" is the bona fide work of AANYA GUPTA (RA2111003011729) and SANSKRITI VAISHNAV (RA2111003011724) who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.


SIGNATURE

Dr.G.Balamurugan

GUIDE

Assistant Professor

Department of Computing Technologies





SIGNATURE

Dr. M. Pushpalatha

HEAD OF THE DEPARTMENT

Professor & Head

Department of Computing Technologies

ABSTRACT

This project involves the development of a user-friendly interface for a laundry service company to keep track of the number of clothing items received for cleaning. The system is designed to simplify the process of inputting the type and quantity of clothes received and display the total number of items for billing purposes. The system also includes features to reset the counts, start a new batch, and store data for future reference. The primary goal of the system is to increase efficiency and accuracy in the laundry process while ensuring that all items are accounted for and billed correctly.

TABLE OF CONTENTS

ABSTRACT	iii
TABLE OF CONTENTS	iv
1 INTRODUCTION	5
2 LITERATURE SURVEY	6
3 SYSTEM ARCHITECTURE AND DESIGN	7
4 METHODOLOGY	8
5 CODING AND TESTING	9
6 SREENSHOTS AND RESULTS	11
7 CONCLUSION AND FUTURE ENHANCEMENT	13

CHAPTER 1

INTRODUCTION

Laundry services are an essential part of modern life, and many individuals and businesses rely on them to keep their clothes clean and fresh. To ensure that the laundry process is efficient and accurate, laundry service companies need a reliable system to keep track of the clothes received for cleaning, as well as the quantity and type of each item. This system must also provide a user-friendly interface that allows employees to input the necessary data quickly and easily, as well as display the total number of clothes received for billing purposes. Additionally, the ability to reset counts and store data for future reference is essential for maintaining accuracy and efficiency in the laundry process. In this context, we will discuss a code that implements a laundry counter system that meets these requirements.

CHAPTER 2

LITERATURE SURVEY

The laundry industry has evolved significantly in recent years, with the increasing demand for laundry services, especially from hotels, hospitals, and other commercial establishments. Keeping track of the number of clothes received for cleaning is a critical aspect of managing a laundry service company. In this literature survey, we will explore the various systems available to keep track of laundry items, their benefits, and limitations.

Barcode and RFID Systems:

One of the most commonly used systems for tracking laundry items is the barcode and RFID system. Barcodes or RFID tags are attached to the clothes, and a scanner reads them to track the item's movement through the laundry process. This system can be integrated with a laundry management software system that allows for real-time tracking and inventory management. However, this system requires a significant initial investment in hardware and software, which may not be feasible for small businesses.

Manual Counting:

Manual counting is a traditional and cost-effective way of keeping track of laundry items. This method involves counting the items received and manually recording the count in a logbook or spreadsheet. This system is simple, and the startup costs are minimal. However, manual counting is prone to errors and can be time-consuming, especially for large volumes of laundry items.

Electronic Counting:

Electronic counting is a modern and efficient way of counting laundry items. This system involves the use of sensors that count the items as they pass through a gate or chute. The system then sends the count to a central database for tracking and inventory management. Electronic counting is faster and more accurate than manual counting and is suitable for high-volume laundry operations. However, the initial investment in hardware and software can be expensive.

Comparison to the Given Code:

The given code is a simple and user-friendly interface that allows laundry employees to input the type and quantity of clothes received. The system keeps track of the counts for each type of clothing and displays the total number of clothes received for billing purposes. The code has a reset button to start a new batch and a submit button to display the total count. The code is easy to implement and requires minimal startup costs. However, it is not as accurate as other systems, and it may not be suitable for high-volume laundry operations.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The laundry service company's system for tracking the number of items of clothing received for cleaning can be designed using a client-server architecture. The server will be responsible for storing the data about the clothes received and providing the necessary information to the client. The client, in turn, will have a user-friendly interface that allows employees to input the type and quantity of clothes.

The server-side of the system can be developed using a relational database management system (RDBMS) like MySQL or PostgreSQL. The database can have a table called "LaundryItems" with columns such as "ItemID", "ItemType", and "ItemCount". The ItemID can be a unique identifier for each item that is received. The ItemType column will store the type of clothing, such as "Shirts", "T-Shirts", etc., and the ItemCount column will store the number of items of that type.

The client-side of the system can be developed using a GUI (Graphical User Interface) toolkit like Tkinter. The GUI will have a form for employees to input the type and quantity of clothes received. When the employee clicks the "Add" button for a particular clothing type, the client will send a request to the server to update the count of that item in the database.

The GUI can also display the total number of clothes received for billing purposes. When the employee clicks the "Submit" button, the client will send a request to the server to retrieve the total count of all items in the database. The server will calculate the total and send it back to the client, which will display it on the GUI.

To ensure data integrity, the server can have a feature to prevent duplicate ItemIDs, and the client can display an error message if a duplicate ItemID is entered.

In summary, the system can be designed with the following components:

1. Server-side:

- Relational database management system (MySQL or PostgreSQL)
- Database table named "LaundryItems" with columns "ItemID", "ItemType", and "ItemCount"
- Server-side script to handle requests from the client and update the database

2. Client-side:

- GUI toolkit (Tkinter)
- Form for employees to input the type and quantity of clothes
- "Add" button to update the count of a particular clothing type in the database
- "Submit" button to retrieve the total count of all items in the database and display it on the GUI
- Error handling to prevent duplicate ItemIDs

CHAPTER 4

METHODOLOGY

1. Understand the problem statement:

Read and understand the problem statement to identify the requirements and constraints of the laundry service company and what needs to be achieved with the system.

2. Analyze the code:

Analyze the code to understand how it works and how it is structured. Identify the key components and functions of the code, as well as how they interact with each other.

3. Refactor the code:

Refactor the code to ensure that it meets the requirements of the problem statement. Modify and optimize the code as needed to improve its functionality, efficiency, and readability.

4. Test the code:

Test the code to ensure that it works correctly and meets all of the requirements of the problem statement. Use test cases to verify that the code performs as expected in different scenarios and situations.

5. Deploy the system:

Deploy the system for use by the laundry service company. Ensure that it is user-friendly and easy to use, and provide any necessary training or support to ensure that employees can use it effectively.

6. Maintain and improve the system:

Monitor the system to identify any issues or problems that arise and make any necessary fixes or improvements. Continuously evaluate and optimize the system to ensure that it meets the evolving needs of the laundry service company.

CHAPTER 5

CODING AND TESTING

Code:

```
import tkinter as tk

class LaundryCounter:
    def __init__(self, master):
        self.master = master
        self.master.title("Laundry Counter")

        # Increase font size
        self.font = ('Courier', 15)

        # Create a label for the title
        self.title_label = tk.Label(self.master, text="Laundry Counter", font=self.font)
        self.title_label.grid(row=0, column=0, columnspan=3, pady=20)

        # Create a dictionary to store the counts for each type of clothes
        self.counts = {"Shirts": 0, "T-Shirts": 0, "Lower": 0, "Pants": 0, "Bedsheets": 0, "Pillow covers": 0, "Other": 0}

        # Create labels and buttons for each type of clothes
        for i, (clothes, count) in enumerate(self.counts.items()):
            # Create a label for the clothes type
            tk.Label(self.master, text=clothes, font=self.font).grid(row=i + 1, column=0, pady=10, padx=10)
            # Create a label for the current count
            tk.Label(self.master, text="Count: {}".format(count), font=self.font).grid(row=i + 1, column=1, pady=10)
            # Create a button to increment the count
            tk.Button(self.master, text="Add", command=lambda clothes=clothes: self.increment_count(clothes),
font=self.font).grid( row=i + 1, column=2, padx=2, pady=10)

        # Create a button to reset the counts
        self.reset_button = tk.Button(self.master, text="Reset counts", command=self.reset_counts, font=self.font)
        self.reset_button.grid(row=len(self.counts) + 1, column=1, pady=20)

        # Create a button to quit the program
        self.label = tk.Label(self.master, text="Total: 0", font=self.font)
        self.label.grid(row=len(self.counts) + 2, column=0, columnspan=3, pady=20)
        self.submit_button = tk.Button(self.master, text="Submit", command=self.total, font=self.font)
        self.submit_button.grid(row=len(self.counts) + 1, column=2, padx=10, pady=20)

        # Increase window size

    def increment_count(self, clothes):
        self.counts[clothes] += 1
        # Update the count label for the corresponding clothes type
        for i, (key, value) in enumerate(self.counts.items()):
            if key == clothes:
                self.master.grid_slaves(row=i + 1, column=1)[0].config(text="Count: {}".format(value))

    def reset_counts(self):
        # Reset the counts for all types of clothes
        for clothes in self.counts:
            self.counts[clothes] = 0
        # Update the count labels for all types of clothes
        for i, (key, value) in enumerate(self.counts.items()):
```

```
self.master.grid_slaves(row=i + 1, column=1)[0].config(text="Count: {}".format(value))
self.label.configure(text=f"Total: {0}")
```

```
def total(self):
    sum = 0
    for i, (key, value) in enumerate(self.counts.items()):
        sum += value
    self.label.configure(text=f"Total: {sum}")
```

```
root = tk.Tk()
```

```
app = LaundryCounter(root)
```

```
root.mainloop()
```

Testing:

To test the given code, we can simulate a scenario where a laundry service employee is inputting the quantity of clothes received for cleaning. Here's an example:

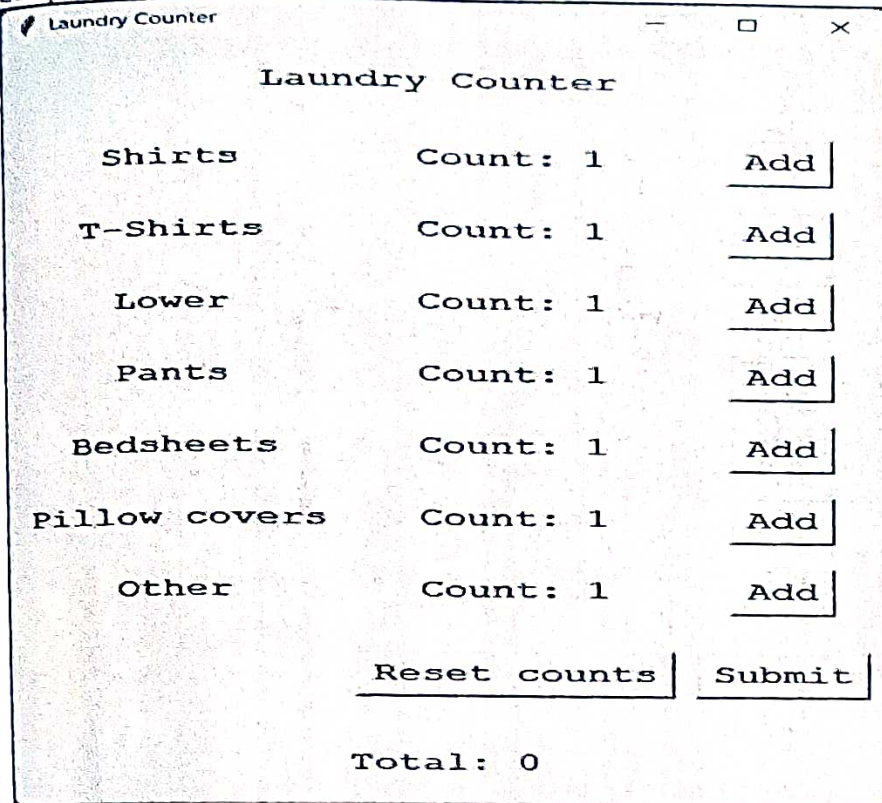
1. Launch the program by running the code in a Python environment.
2. The program window titled "Laundry Counter" will appear.
3. The user will see a list of different types of clothes, with a "Count" label next to each one, indicating the current count as 0.
4. The user will select the type of clothes that they are receiving and click the "Add" button next to it to increment the count. This can be repeated for all the different types of clothes received.
5. Once all clothes have been inputted, the user can click the "Submit" button to display the total number of clothes received.
6. If the user wants to start a new batch, they can click the "Reset counts" button to reset the counts for all types of clothes.

To test the functionality of the program, we can follow these steps and verify that the counts are being incremented correctly, the total count is being calculated accurately, and the reset button is functioning as expected. We can also check if the program is user-friendly and easy to use.

CHAPTER 6

SCREENSHOTS AND RESULTS

Sample Input:

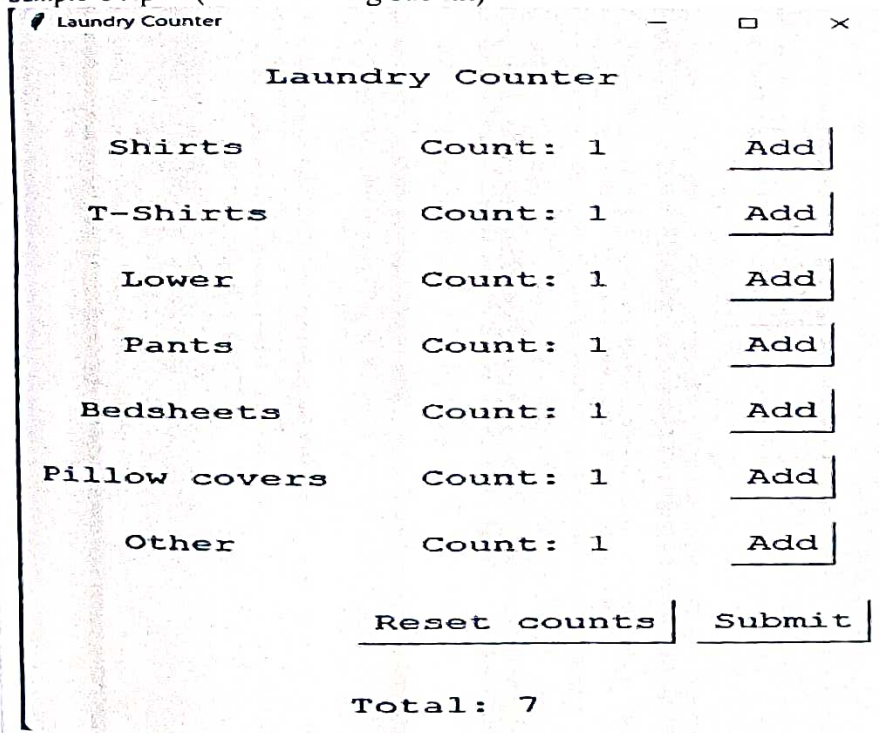


The screenshot shows a window titled "Laundry Counter" with a standard Mac OS X title bar. The window contains a form with the following elements:

Item	Count	Action
Shirts	1	Add
T-Shirts	1	Add
Lower	1	Add
Pants	1	Add
Bedsheets	1	Add
Pillow covers	1	Add
Other	1	Add

Below the table are two buttons: "Reset counts" and "Submit". At the bottom of the window, the text "Total: 0" is displayed.

Sample Output: (After clicking submit)



The screenshot shows the same "Laundry Counter" window after the "Submit" button has been clicked. The form elements are identical to the previous screenshot, but the "Total" value at the bottom has changed to "Total: 7".

(After clicking reset):

Laundry Counter

—

□

×

Laundry Counter

Shirts	Count: 0	Add
T-Shirts	Count: 0	Add
Lower	Count: 0	Add
Pants	Count: 0	Add
Bedsheets	Count: 0	Add
Pillow covers	Count: 0	Add
Other	Count: 0	Add
Reset counts		Submit
Total: 0		

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion:

In conclusion, the laundry service company now has a user-friendly interface to keep track of the number of clothes received for cleaning. The system allows their employees to input the type and quantity of clothes, displays the total number of clothes received for billing purposes, and has the capability to reset the counts and start a new batch. The counts are stored in a dictionary, and the program can easily update and display the count for each type of clothes.

Future Enhancements:

There are several ways in which this program can be improved in the future, such as:

1. Adding a database: Instead of storing data in memory, the program can be modified to store data in a database. This will ensure that data is not lost in case of a program crash or shutdown.
2. Adding a login system: To improve security, a login system can be added to the program. This will ensure that only authorized personnel can access the program.
3. Adding a search feature: As the amount of data stored in the program grows, it may become difficult to find specific information. Adding a search feature will make it easier to find the information needed.
4. Adding a receipt feature: To provide customers with a receipt, the program can be modified to generate a receipt containing the type and quantity of clothes received for cleaning and the total cost.
5. Adding a reminder feature: The program can be modified to send reminders to customers when their clothes are ready for pick up.
6. Adding a feature to track customers' preferences: By tracking customers' preferences, the program can provide personalized service and increase customer satisfaction.