­­­­

2024 Year 12 Comp Sci

­­­­­­­­

OOP Business Client Application Project

Full Document

Alec McDonald

# Table Of Contents

Community Store Software Development Project 0

[Table Of Contents 1](#_Toc161935900)

[Part 1 – Planning 2](#_Toc161935901)

[Tasks to be done part 1. 2](#_Toc161935902)

[Tasks to be done part 2 Develop. 2](#_Toc161935903)

[Tasks to be done part 2 Evaluate. 2](#_Toc161935904)

[Time frame 3](#_Toc161935905)

[Problem Outline 3](#_Toc161935906)

[Problem Description 3](#_Toc161935907)

[Pseudocode 4](#_Toc161935908)

[Part 2 – Coding 8](#_Toc161935909)

[Required files. 8](#_Toc161935910)

[Files 8](#_Toc161935911)

[Folder: 8](#_Toc161935912)

[Python: 8](#_Toc161935913)

[Project code 8](#_Toc161935914)

[Part 3 – Evaluation 29](#_Toc161935915)

[Problems & Improvements 29](#_Toc161935916)

[Developer Summary 29](#_Toc161935917)

[Sources 29](#_Toc161935918)

[Chat GPT Conversation. 30](#_Toc161935919)

# Part 1 – Planning

## Tasks to be done part 1.

* Brake down tasks to do.
* Outline problem.
* Problem Description.
* Write basic pseudocode of program to show core logic.
* Outline using structure chart.

## Tasks to be done part 2 Develop.

* Create something to visualise seating.
* Take bookings.
* Create receipt.
* Create way to cancel booking.

## Tasks to be done part 2 Evaluate.

* Debug program.
* Note down problems.
* Reflect on project.
* Provide sources.

## Time frame

I have 5 weeks to complete this project.

Starting week 4 term 1 and to be completed by week 8 term 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | = Not Started |  | = Doing |  | = Finished |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Key Point | Item | Due Date | Status |
| 1 | Investigate | Project Breakdown | Term 1 Week 4 |  |
| Problem Outline | Term 1 Week 4 |  |
| Problem Description | Term 1 Week 4 |  |
| Project Timeline | Term 1 Week 5 |  |
| Design | Write Pseudocode | Term 1 Week 5 |  |
| Create Structure Charts | Term 1 Week 5 |  |
| 2 | Develop | Code ability to visualise seats | Term 1 Week 6 |  |
| Code ability to create a booking | Term 1 Week 6 |  |
| Code ability to create a receipt to customer | Term 1 Week 6 |  |
| Code ability to cancel booking | Term 1 Week 6 |  |
| Evaluate | Debug Program | Term 1 Week 7 |  |
| Note down problems | Term 1 Week 7 |  |
| Reflect on project | Term 1 Week 7 |  |
| Provide sources | Term 1 Week 7 |  |

## Problem Outline

The problem we are facing is that the school does not have an efficient method of handling tickets of small productions. Tickets for productions are currently sold by the customer calling the school and a receptionist takes the customer’s order and emails them their tickets. The school has requested for me to create a simple program to help assist with the selling of tickets, cancelation of tickets, sending receipts and copy of tickets to customer and creating a visual representation to see which seats are left.

## Problem Description

I think what I am going to do is split it up into 4 main functions. 1 function for receipts, 1 function for taking bookings, 1 function for cancelation of bookings and 1 function for when program is closed. These functions are going to pull and store information in multiple text files text files and/or 2 dimensional arrays. The arrays will store which seats are taken and which are free, and a text file will store which customer ordered what seat. The customer will be able to interact with the program to order their tickets. The customer will answer the prompts depending on what they want to do:

“To book a ticket type ‘Booking’.

To cancel a booking type ‘Cancel’

To end program type ‘End’”

It will then branch out to complete these specific functions from this prompt. It will only require the operator to follow the prompts that come up and type their input.

## Pseudocode

CLASS Booking:

METHOD Seating:

create 2-dimensional array named seats

RETURN the variable to the program

ENDMETHOD

METHOD Print Seating:

PRINT “letters A – J”

FOR row in array seats:

PRINT “row number”

Row number = row number + 1

FOR seat in a row:

IF seat was equal to 0:

PRINT “-“

ELSE”

PRINT “x”

ENDIF

ENDMETHOD

METHOD Book Seat

PRINT “Current seating plan:”

Create empty array called seats

Gets the array from METHOD booking

Call METHOD Print Seating

TRY:

Get user input as an integer and save it as a variable called howManySeats

EXCEPT:

PRINT “Invalid input format. Please use integers.”

Call METHOD Book Seat

Create a dictionary for person prices

Create an empty array called results

IF user value is less than 0:

Call METHOD Processing Booking

ELSE:

PRINT “You have not entered an integer above 0. Please try again.”

Call METHOD Book Seat

ENDIF

ENDMETHOD

METHOD Processing Booking

Turns how many seats variable into an integer

Saves original value in variable howManySeats to variable called totalSeats

Creates empty array called seatsBooked

WHILE variable howManySeats is less than 0

Saves user input as selection

Saves user input as typePersonInput

PRINT “selection”

IF NOT selection[0].isdigit()

PRINT “Invalid input format. Please use the format <RowNumber><ColumnLetter> (e.g., 2D).\n”

Call METHOD Processing Booking

ELSE:

TRY:

Row equals selection[0] – 1

Column = selection[1]

IF row and column is within the array seats

IF seats is empty

FOR record IN Person Prices

IF typePersonInput is in dictionary

Change index in array to 1

Add record to the results array

howManySeats = howManySeats – 1

PRINT “Seat booked successfully!”

Adds selection to seatsBooked array

BREAK

ELSE

PRINT “Invalid type of person input. Please try again”

CONTINUE

ENDIF

ELSE

PRINT “Sorry, that seat is already taken.”

ENDIF

ELSE

PRINT “Invalid seat selection.”

ENDIF

EXCEPT:

PRINT “Invalid input format. Please use the format <RowNumber><ColumnLetter> (e.g., 2D).”

PRINT “Updated seating plan:”

Call METHOD Print Seating

ENDIF

Call METHOD Receipt Function

ENDMETHOD

METHOD Receipt Function

Turns variable seatsBooked into a string and remove unwated characters

Get a random 8 character number and save it to variable randomNumber

Search folder for all text files and put all the files in a variable called txtFiles

Takes the name of all files in txtFiles and saves them to array fileNames

FOR fileName IN fileNames

Removes unwanted characters from the string

Split filename into 2 different parts

Saves the second part of the file name as a variable receiptID

IF receiptID is equal to a randomNumber

Create a new random number

ELSE

Create a new file called “Reciept\_{randomNumber}.txt”

Write to file “======================================\nReciept ID: {randomNumber}\nTotal seats Booked: {totalSeats}\nSeats Booked: {seatsBooked}\n\nType Of Seat - Price - Runing Total”

BREAK

ENDIF

Sets variable price to 0

FOR result IN results

Result equals result.split()

Variable resultTypePerson equals result[1]

Replaces the placeholder values for type of person with full name in variable resultTypePerson

Variable resultPrice equals result[3]

Remove unwanted characters from variable resultPrice and turns it into an integer

Variable price equals price plus resultPrice

Write into the file “{resultTypePerson} - {resultPrice} - {price}\n Total Price: {price}\n ======================================”

PRINT “you have booked {totalSeats}. This will cost ${price}\nA reciept will be sent to you soon\nYour reciept ID is: {randomNumber}”

ENDMETHOD

ENDCLASS

CLASS Cancel

METHOD User Input

TRY

Save the user input as variable userInput

EXCEPT

PRINT “Invalid input format. Please use integers.”

Call METHOD User Input

Call METHOD Find Ticket

ENDMETHOD

METHOD Find Tickets

Search folder for all text files and put all the files in a variable called txtFiles

Takes the name of all files in txtFiles and saves them to array fileNames

FOR fileName IN fileNames

Variable originalFileName equals variable fileName

Removes unwanted characters from the string

Split filename into 2 different parts

Saves the second part of the file name as a variable receiptID

IF receiptID is equal to a randomNumber

Call METHOD Get Seats

BREAK

ELSE

CONTINUE

ENDIF

PRINT “There is no ticket with ID of '{userInput}'”

Call Program Function

ENDMETHOD

METHOD Get Seats

Open the file stored in variable originalFileName in read mode

Read the file

Close the file

Save the fourth line of the file as variable fourthLine

Remove unwanted characters from string and save it as variable ticketSeats

Call METHOD Vacate Seats

ENDMETHOD

METHOD Vacate Seats

Create an empty array called seats

Gets the array from METHOD booking

Split variable ticketSeats into two different parts

FOR ticketSeat IN ticketSeats:

Row equals selection[0] – 1

Column = selection[1]

IF row and column is within the array seats

IF seats is full

Change the full seat to empty

BREAK

ELSE

PRINT “Seat is already empty. Continuing onto next seat”

CONTINUE

ENDIF

ELSE:

PRINT “Sorry, we encounted an error. Continuing to next seat.”

CONTINUE

ENDIF

Remove the file from the folder

ENDMETHOD

ENDCLASS

DEFINE FUNCTION End

Quit the program

ENDFUNCTION

DEFINE FUNCTION Program

PRINT “To book a ticket type 'Booking' \nTo cancel a booking type 'Cancel' \nTo end program type 'End'”

Save user input as variable userInput

Change userInput to all lowercase

MATCH variable userInput

CASE “booking”

Call METHOD Book Seat IN CLASS Booking

CASE “cancel”

Call METHOD User Input IN CLASS Cancel

CASE “end”

Call FUNCTION End Function

CASE \_:

PRINT “Incorrect entry please check your spelling and try again.”

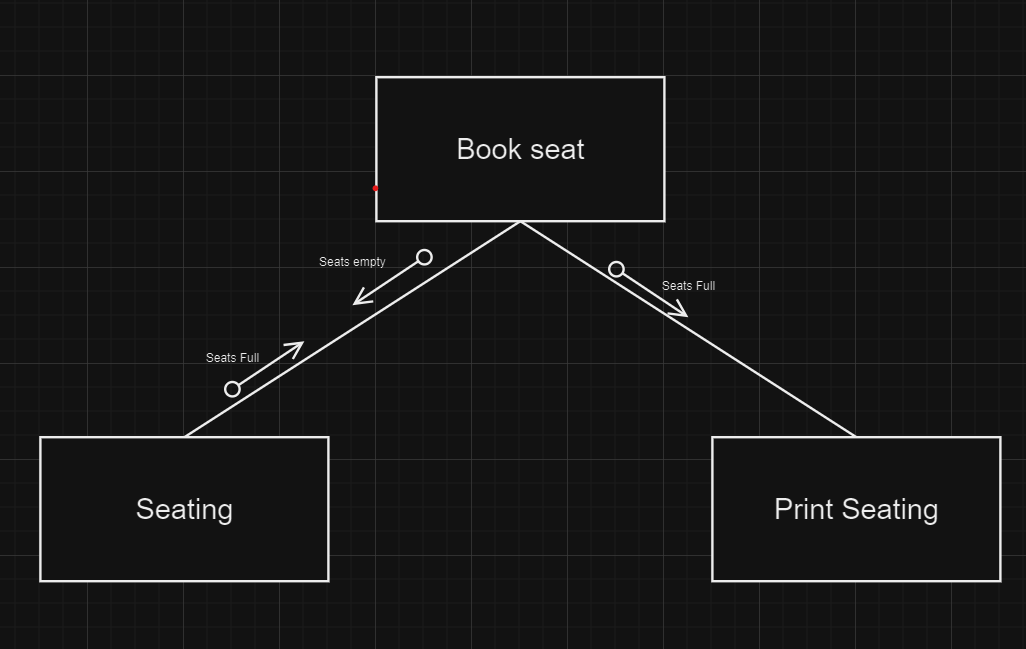
Call FUNCTION Program Function

ENDCASE

ENDFUNCTION

Call FUNCTION Program Function

# Structure Chart



# Part 2 – Coding

## Required files.

A screenshot of a computer

Description automatically generated

## Files

### Folder:

<https://greatsoutherngrammar-my.sharepoint.com/personal/alec_mcdonald_student_gsg_wa_edu_au/Documents/.Year%2012/Computer%20Science/Task%201>

### Python:

<https://greatsoutherngrammar-my.sharepoint.com/personal/alec_mcdonald_student_gsg_wa_edu_au/Documents/.Year%2012/Computer%20Science/Task%201/Task_1_Code.py>

## Project code

# Importing extensions for the program to work

import tkinter as tk

from tkinter import \*

import csv

import os

import datetime

# Defining Button and Window sizes as variables

buttonHeight = 4

buttonWidth = 20

buttonSpacingX = 10

buttonSpacingY = 10

windowSize = ("400x700")

# Defining function to do all options that revolve around the category system. Contains: addCategory Function, deleteCategory Function, listCategory Function as well as the means to call each of the functions using tkinter buttons

def updateCategory():

    #Defining function to add a category to the system (adding a CSV file to the directory where this file is located)

    def addCategory():

        # Creating tkinter window

        addCategoryGui = tk.Toplevel(updateCategoryGui)

        addCategoryGui.geometry(windowSize)

        addCategoryGui.title("Category List")

        # Creating a listbox of all the categories (all the CSV files in the directory)

        csvFileList = os.listdir()

        categoryList = [f for f in csvFileList if f.endswith(".csv")]

        listbox = tk.Listbox(addCategoryGui)

        # Searching domain for files and puting them into listbox

        for file in categoryList:

            listbox.insert(tk.END, file)

        listbox.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Adding labbel under list box in same window

        addCategoryGui.title("Add Category")

        label = tk.Label(addCategoryGui, text="Enter category to be Added:")

        label.pack()

        # Adding entry window so user can type a value

        entry = tk.Entry(addCategoryGui)

        entry.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Function to write a CSV into the directory

        def buttonClick():

            # Pulling the tk.entry value into function and putting it into a variable

            addCategory = entry.get()

            # Using above variable to create a CSV with users intended name

            open(addCategory + ".csv", "w")

            # Creating a tkinter window

            buttonClickGui = tk.Toplevel(addCategoryGui)

            buttonClickGui.geometry(windowSize)

            buttonClickGui.title("Add Category")

            # Adding a label to the window

            label = tk.Label(buttonClickGui, text="The category '" + addCategory + "' has been added")

            label.pack(padx=buttonSpacingX, pady=buttonSpacingY)

            # Adding close window button to close the window

            closeWindow = tk.Button(buttonClickGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=buttonClickGui.destroy)

            closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

            # Closing unessasary windows

            addCategoryGui.destroy()

        # Adding a button to window under the entry prompt to call the buttonClick function

        button = tk.Button(addCategoryGui, text="Add", width=buttonWidth, height=buttonHeight, command=buttonClick)

        button.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Adding putton to close window

        closeWindow = tk.Button(addCategoryGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=addCategoryGui.destroy)

        closeWindow.pack()

    # Defining function to delete a category

    def deleteCategory():

        deleteCategoryGui = tk.Toplevel(updateCategoryGui)

        deleteCategoryGui.title("Delete Text File")

        deleteCategoryGui.geometry(windowSize)

        # Creating a listbox of all the categories (all the CSV files in the directory)

        csvFileList = os.listdir()

        categoryList = [f for f in csvFileList if f.endswith(".csv")]

        listbox = tk.Listbox(deleteCategoryGui)

        # Searching domain for files and puting them into listbox

        for f in categoryList:

            listbox.insert(tk.END, f)

        listbox.pack(padx=10, pady=10)

        # Defining function to remove selected category from the system (selected csv from the directory)

        def deleteSelectedCategory():

            # Setting variable as the users selection in the list box

            selection = listbox.curselection()

            # If Statement to delete the category from system (CSV from directory)

            if selection:

                categoryToDelete = listbox.get(selection[0])

                os.remove(categoryToDelete)

                listbox.delete(selection[0])

        # Button to delete the category/CSV

        deleteButton = tk.Button(deleteCategoryGui, text="Delete", width=buttonWidth, height=buttonHeight, command=deleteSelectedCategory)

        deleteButton.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Button to close the tkinter window

        closeWindow = tk.Button(deleteCategoryGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=deleteCategoryGui.destroy)

        closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Defining function to listall categories in system (CSV in directory)

    def listCategory():

        # Creating tkinter window

        listCategoryGui = tk.Toplevel(updateCategoryGui)

        listCategoryGui.geometry(windowSize)

        listCategoryGui.title("Category List")

        # Creating a listbox of all the categories (all the CSV files in the directory)

        csvFileList = os.listdir()

        categoryList = [f for f in csvFileList if f.endswith(".csv")]

        listbox = tk.Listbox(listCategoryGui)

        # Searching domain for files and puting them into listbox

        for file in categoryList:

            listbox.insert(tk.END, file)

        listbox.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Adding close window button to bottom of the window

        closeWindow = tk.Button(listCategoryGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=listCategoryGui.destroy)

        closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Creating a tkinter window

    updateCategoryGui = tk.Toplevel(root)

    updateCategoryGui.geometry(windowSize)

    # Adding button to call the addCategory function

    buttonAddCategory = tk.Button(updateCategoryGui, text="Add Category", width=buttonWidth, height=buttonHeight, command=addCategory)

    buttonAddCategory.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Adding button to call the deleteCategory function

    buttonDeleteCategory = tk.Button(updateCategoryGui, text="Delete Category", width=buttonWidth, height=buttonHeight, command=deleteCategory)

    buttonDeleteCategory.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Adding button to call the listCategory function

    buttonListCategory = tk.Button(updateCategoryGui, text="Category List", width=buttonWidth, height=buttonHeight, command=listCategory)

    buttonListCategory.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Adding button to close the window

    closeWindow = tk.Button(updateCategoryGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=updateCategoryGui.destroy)

    closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

# Defining function to do all options related to stock

def updateStock():

    # Defining function to add stock to a selected category/CSV

    def addStock():

        # Creating tkinter window

        addStockGui = tk.Toplevel(updateStockGui)

        addStockGui.geometry(windowSize)

        addStockGui.title("Update stock list")

        # Creating label to explain what to do.

        label = tk.Label(addStockGui, text= "Select category you wish to enter stock into")

        label.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Creating a listbox of all the categories (all the CSV files in the directory)

        csvFileList = os.listdir()

        files = [f for f in csvFileList if f.endswith('.csv')]

        listbox = tk.Listbox(addStockGui)

        # Searching domain for files and puting them into listbox

        for f in files:

            listbox.insert(tk.END, f)

        listbox.pack(padx=10, pady=10)

        # Creating a labbel in tkinter window bellow listbox

        stockAddLabel = tk.Label(addStockGui, text="Enter Item You Want To Add \n DO NOT INCLUDE: SYMBOLS OR NUMBERS")

        stockAddLabel.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Creating a entry box in tkinter in order to get a user input for items name that they want to add

        stockToAdd = tk.Entry(addStockGui)

        stockToAdd.pack(padx=10, pady=10)

        # Creating a label to explain want to enter into entry box

        stockPriceLabel = tk.Label(addStockGui, text="Enter Items Price \n DO NOT INCLUDE: SYMBOLS OR LETERS")

        stockPriceLabel.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Adding a entry box to get price user wants for the item

        stockPriceToAdd = tk.Entry(addStockGui)

        stockPriceToAdd.pack(padx=10, pady=10)

        # Defining function to write the user input to the CSV

        def addStockToCategory():

            # Creating a new tkinter window

            addStockToCategoryGui = tk.Toplevel(root)

            addStockToCategoryGui.geometry(windowSize)

            addStockToCategoryGui.title("Add item")

            # Setting variable as user selection

            selection = listbox.curselection()

            # Stores users values for latter use in other variables

            if selection:

                categoryToAddToo = listbox.get(selection[0])

                stockAddedToList = stockToAdd.get()

                priceAddedToList = stockPriceToAdd.get()

                # opening CSV in append mode and making a new line and then writing the users input into the CSV

                with open(categoryToAddToo, 'a', newline='') as file:

                    writer = csv.writer(file)

                    writer.writerow([stockAddedToList, priceAddedToList])

            # Adding a labbel to window to let the user know that the item they entered has been added to the CSV

            confirmationLabel = tk.Label(addStockToCategoryGui, text="The item '" + stockAddedToList + " $" + priceAddedToList + "' has been added to '" + categoryToAddToo)

            confirmationLabel.pack(padx=buttonSpacingX, pady=buttonSpacingY)

            # Adding a close window button

            closeWindow = tk.Button(addStockToCategoryGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=addStockToCategoryGui.destroy)

            closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Adding button to call the addStockToCategory function

        addItem = tk.Button(addStockGui, text="Add item", width=buttonWidth, height=buttonHeight, command=addStockToCategory)

        addItem.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Adding button to close the window

        closeWindow = tk.Button(addStockGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=addStockGui.destroy)

        closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Defining function to delete a category/CSV

    def deleteStock():

        #Creating the tkinter window

        deleteStockGui = tk.Toplevel(updateStockGui)

        deleteStockGui.title("Deleting stock")

        deleteStockGui.geometry(windowSize)

        # Creating a listbox of all the categories (all the CSV files in the directory)

        csvFileList = os.listdir()

        files = [f for f in csvFileList if f.endswith('.csv')]

        fileListbox = tk.Listbox(deleteStockGui)

        # Searching domain for files and puting them into listbox

        for f in files:

            fileListbox.insert(tk.END, f)

        fileListbox.pack(padx=10, pady=10)

        # Defining function to open selected csv and put its contents into listbox

        def selectingCategory():

            # Setting user selection as a variable

            selection = fileListbox.curselection()

            # Checking if an item has been selected and then it stores user selected/entered values for latter use

            if selection:

                selectedFile = fileListbox.get(selection[0])

                # Oppening CSV in read mode

                with open(selectedFile, 'r') as file:

                    reader = csv.reader(file)

                    contents = [row for row in reader]

                # Creating tkinter window

                selectCategoryGui = tk.Toplevel(deleteStockGui)

                selectCategoryGui.title(selectedFile)

                selectCategoryGui.geometry(windowSize)

                # Setting contents of list box as a variable

                contentsListbox = tk.Listbox(selectCategoryGui)

                # Inserting the data in the CSV file into the list box

                for row in contents:

                    contentsListbox.insert(tk.END, row)

                contentsListbox.pack(padx=10, pady=10)

                # Function to remove everything in selected line from the CSV file

                def deletingItem():

                    # setting user selection as a variable

                    selection = contentsListbox.curselection()

                    # Checking if an item has been selected and then deleting it

                    if selection:

                        contentsListbox.delete(selection[0])

                        # Oppening selected file in write mode and deleting the selected line from the file

                        with open(selectedFile, 'w', newline='') as file:

                            writer = csv.writer(file)

                            writer.writerows(contentsListbox.get(0, tk.END))

                    # Closing the window

                    selectCategoryGui.destroy()

                # Creates button to close the window

                deleteButton = tk.Button(selectCategoryGui, text="Delete Item", width=buttonWidth, height=buttonHeight, command=deletingItem)

                deleteButton.pack(padx=10, pady=10)

        # Creates button to call the selectingCategory function

        selectButton = tk.Button(deleteStockGui, text="Select Category", width=buttonWidth, height=buttonHeight, command=selectingCategory)

        selectButton.pack(padx=10, pady=10)

        # Creats button to close window

        closeWindow = tk.Button(deleteStockGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=deleteStockGui.destroy)

        closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Defining a function to change the price to users chosen amount

    def updatePrice():

        # Creates the tkinter window

        updatePriceGui = tk.Toplevel(updateStockGui)

        updatePriceGui.title("Updating price")

        updatePriceGui.geometry(windowSize)

        # Looking for files that have .csv in them and putting them in a list format

        files = [f for f in os.listdir() if f.endswith('.csv')]

        fileListbox = tk.Listbox(updatePriceGui)

        # Searching domain for files and puting them into listbox

        for f in files:

            fileListbox.insert(tk.END, f)

        fileListbox.pack(padx=10, pady=10)

        # Defining function to open the contents of the selected csv into a listbox

        def selectingCategory():

            # Setting selection as a variable

            selection = fileListbox.curselection()

            # Checking if an item has been selected and then it stores user selected/entered values for latter use

            if selection:

                selectedFile = fileListbox.get(selection[0])

                # Oppening CSV in read mode  and extracting its contents

                with open(selectedFile, 'r') as file:

                    reader = csv.reader(file)

                    contents = [row for row in reader]

                # Creating a tkinter window

                selectingCategoryGui = tk.Toplevel(updatePriceGui)

                selectingCategoryGui.title(selectedFile)

                selectingCategoryGui.geometry(windowSize)

                # Label to explain what to do

                label = tk.Label(selectingCategoryGui, text="Select item you wish to modify")

                label.pack(padx=buttonSpacingX, pady=buttonSpacingY)

                # Setting an empty list box

                contentsListbox = tk.Listbox(selectingCategoryGui)

                # Putting contents of the CSV into the listbox

                for row in contents:

                    contentsListbox.insert(tk.END, row)

                contentsListbox.pack(padx=10, pady=10)

                # Defining function to change the selected line price to new value

                def modifySelection():

                    # Setting selection as a variable

                    rowSelection = contentsListbox.curselection()

                    # Checking if a row has been selected and then it stores user selected/entered values for latter use

                    if rowSelection:

                        rowIndex = rowSelection[0]

                        row = contents[rowIndex]

                        newPrice = entry.get()

                        # Updating value in 3 row and extracting value to be used in program in the 1 row

                        row[1] = newPrice

                        item = row[0]

                        # Oppen the CSV file in  write mode and change price to users value

                        with open(selectedFile, 'w', newline='') as file:

                            writer = csv.writer(file)

                            writer.writerows(contents)

                        # Deleting contents of list box making it empty again

                        contentsListbox.delete(0, tk.END)

                        # Rewrite the list box so it has the new value in it

                        for row in contents:

                            contentsListbox.insert(tk.END, row)

                    # Creating a new tkinter window

                    modifySelectionGui = tk.Toplevel(selectingCategoryGui)

                    modifySelectionGui.geometry(windowSize)

                    modifySelectionGui.title("Price Change")

                    # Creating a labble to say the item the chose has a new price of what ever they selected

                    label = tk.Label(modifySelectionGui, text="The item '" + item + "' price has been changed to '$" + newPrice)

                    label.pack(padx=buttonSpacingX, pady=buttonSpacingY)

                    # Creates a button to close the window

                    closeWindow = tk.Button(modifySelectionGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=modifySelectionGui.destroy)

                    closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

            # Label to explain what to do

            label = tk.Label(selectingCategoryGui, text="Enter New Price \n DO NOT INCLUDE: SYMBOLS OR LETERS")

            label.pack(padx=buttonSpacingX, pady=buttonSpacingY)

            # Creates a entry widget so user can type a value

            entry = tk.Entry(selectingCategoryGui)

            entry.pack(padx=10, pady=10)

            # Creates a button to call the modifySelection function

            modifyButton = tk.Button(selectingCategoryGui, text="Modify Selection", width=buttonWidth, height=buttonHeight, command=modifySelection)

            modifyButton.pack(padx=10, pady=10)

            # Creates a button to close the window

            closeButton = tk.Button(selectingCategoryGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=selectingCategoryGui.destroy)

            closeButton.pack(padx=10, pady=10)

        # Creates a button to call the selectingCategory function

        selectButton = tk.Button(updatePriceGui, text="Select Category", width=buttonWidth, height=buttonHeight, command=selectingCategory)

        selectButton.pack(padx=10, pady=10)

        # Creates a button to close the window

        closeWindow = tk.Button(updatePriceGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=updatePriceGui.destroy)

        closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Defining function to list stock in a selected file

    def listStock():

        # Creates tkinter window

        listStockGui = tk.Toplevel(updateStockGui)

        listStockGui.title("Deleting stock")

        listStockGui.geometry(windowSize)

        # Searching directory for all files with .csv at the end

        files = [f for f in os.listdir() if f.endswith('.csv')]

        fileListbox = tk.Listbox(listStockGui)

        # Searching domain for files and puting them into listbox

        for f in files:

            fileListbox.insert(tk.END, f)

        fileListbox.pack(padx=10, pady=10)

        # Defining function open the selected category/CSV

        def selectingCategory():

            # Setting selection as a variable

            selection = fileListbox.curselection()

            # Checking if an item has been selected and then it stores user selected/entered values for latter use

            if selection:

                selectedFile = fileListbox.get(selection[0])

                # Opens selected CSV in read mode

                with open(selectedFile, 'r') as file:

                    reader = csv.reader(file)

                    contents = [row for row in reader]

                # Creates tkinter window

                selectingCategoryGui = tk.Toplevel(root)

                selectingCategoryGui.title(selectedFile)

                selectingCategoryGui.geometry(windowSize)

                # Creates a empty list box

                contentsListbox = tk.Listbox(selectingCategoryGui)

                # Fills list box with contents of the CSV file

                for row in contents:

                    contentsListbox.insert(tk.END, row)

                contentsListbox.pack(padx=10, pady=10)

                # Creates button to close the window

                deleteButton = tk.Button(selectingCategoryGui, text="Close window", width=buttonWidth, height=buttonHeight, command=selectingCategoryGui.destroy)

                deleteButton.pack(padx=10, pady=10)

        # Creates a button to call the selectingCategory function

        selectButton = tk.Button(listStockGui, text="Select Category", width=buttonWidth, height=buttonHeight, command=selectingCategory)

        selectButton.pack(padx=10, pady=10)

        # Creates a button to close the window

        closeWindow = tk.Button(listStockGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=listStockGui.destroy)

        closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Creates a tkinter window

    updateStockGui = tk.Toplevel()

    updateStockGui.geometry(windowSize)

    # Creates a button to call the addStock function

    buttonAddCategory = tk.Button(updateStockGui, text="Add Stock", width=buttonWidth, height=buttonHeight, command=addStock)

    buttonAddCategory.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Creates a button to call the deleteStock function

    buttonDeleteCategory = tk.Button(updateStockGui, text="Delete Stock", width=buttonWidth, height=buttonHeight, command=deleteStock)

    buttonDeleteCategory.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Creates button to call the updatePrice function

    stockButton = tk.Button(updateStockGui, text="Update Price", width=buttonWidth, height=buttonHeight, command=updatePrice)

    stockButton.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Creates a button to call the listStock function

    buttonDeleteCategory = tk.Button(updateStockGui, text="List of Stock", width=buttonWidth, height=buttonHeight, command=listStock)

    buttonDeleteCategory.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Creates a button to close the window

    closeWindow = tk.Button(updateStockGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=updateStockGui.destroy)

    closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

# Defining a function to buy items from the store

def buy():

    # Creating a tkinter window

    buyGui = tk.Toplevel(root)

    buyGui.geometry(windowSize)

    buyGui.title("Buy")

    # Searching directory for all files with .csv at the end

    files = [f for f in os.listdir() if f.endswith('.csv')]

    # Creates empty listbox

    categoriesListbox = tk.Listbox(buyGui)

    categoriesListbox.grid(row=1, column=1)

    # Searching domain for files and putting them into listbox

    for f in files:

        categoriesListbox.insert(tk.END, f)

    # Defining a function to select a category from list box

    def selectingCategory():

        # Setting selection as a variable

        selection = categoriesListbox.curselection()

        # Checking if an item has been selected and then it stores user selected/entered values for latter use

        if selection:

            selectedFile = categoriesListbox.get(selection[0])

            # Opens selected CSV in read mode

            with open(selectedFile, 'r') as file:

                reader = csv.reader(file)

                contents = [row for row in reader]

            # Clear the contents of the list box before inserting new contents

            contentsListbox.delete(0, tk.END)

            # Fills list box with contents of the CSV file

            for row in contents:

                contentsListbox.insert(tk.END, row)

    # Making a button to run the 'selectingCategory' function

    selectButton = tk.Button(buyGui, text="Select Category", width=buttonWidth, height=buttonHeight, command=selectingCategory)

    selectButton.grid(row=2, column=1)

    # Creates empty listbox

    contentsListbox = tk.Listbox(buyGui)

    contentsListbox.grid(row=3, column=1)

    # Function that adds selected item to the 'cart' listbox

    def addToCartFunction():

        # Setting user selection as usable variable

        contentsSelection = contentsListbox.curselection()

        # Adding the selected item to the 'cart' listbox

        if contentsSelection:

            selectedItem = contentsListbox.get(contentsSelection[0])

            cartListbox.insert(tk.END, selectedItem)

    # Function that removes selected item from the 'cart' listbox

    def removeFromCartFunction():

        # Setting user selection as usable variable

        cartSelection = cartListbox.curselection()

        # Removing selected item from 'cart' listbox

        if cartSelection:

           cartListbox.delete(cartSelection[0])

    # Function that is organising all values from 'cart' listbox into a usable and more managable form

    def purchaseOrganise():

        # Setting function as an empty sttring

        cartListboxString = ""

        # Extracting values from list box into a usable string

        for i in range(cartListbox.size()):

            cartListboxString += str(cartListbox.get(i)) + "\n"

        # Coping 'cartListboxString' into a txt file and then removing/replacing punctuation to make it easier latter on

        with open("purchaseItemOriginal.txt", "w") as purchaseItemOriginal:

            purchaseItemOriginal.write(str(cartListboxString.replace("'", "").replace(",", ":").replace("(", "").replace(" ", "").replace(")", "|")))

        # Setting an empty array

        wordCounts = {}

        # Oppening 2 files in read and write mode

        with open("purchaseItemOriginal.txt", "r") as purchaseItemOriginal, open("purchaseItemFrequency.txt", "w") as purchaseItemFrequency:

            # Seperating each item so we can find the frequency of each item

            for line in purchaseItemOriginal:

                words = line.strip().split("|")

                # Finding the value of 'word' in the list of words

                for word in words:

                    # Making sure dont get any empty strings in final result

                    if word != '':

                        # If the word is already in the word count it adds 1 to its value

                        if word in wordCounts:

                            wordCounts[word] += 1

                        # If word is not in the word count it adds the word into the word count with calue of 1

                        else:

                            wordCounts[word] = 1

            # Writing the items and there frequency into a text document

            purchaseItemFrequency.write(str(wordCounts))

        # Oppening 2 files in read and write mode

        with open("purchaseItemFrequency.txt", "r") as purchaseItemFrequency, open("purchaseItemFrequencyCleaner.txt", "w") as purchaseItemFrequencyCleaner:

            # Going through the original file and writing its contents to another file after removing/changing unessasary punctuation

            for line in purchaseItemFrequency:

                purchaseItemFrequencyCleaner.write(line.replace(",", "\n").replace("'", "").replace("{", "").replace("}", "").replace(":", ",").replace(" ", ""))

        # Calling the function

        purchaseMaths()

    # Function that is doing all the math to find out total price of purchase

    def purchaseMaths():

        # Oppening 2 files in read and write mode

        with open("purchaseItemFrequencyCleaner.txt", "r") as purchaseItemFrequencyCleaner, open("purchaseItemFrequencyTotalPrice.txt", "w") as purchaseItemFrequencyTotalPrice:

            # Defining the collumns and then turning the collums with numbers in them into floats

            for line in purchaseItemFrequencyCleaner:

                collumns = line.strip().split(",")

                collumnOne = collumns[0]

                collumnTwo = collumns[1]

                collumnThree = collumns[2]

                collumnTwoFloat = float(collumnTwo)

                collumnThreeFloat = float(collumnThree)

                # Finding the total price of each item that has been purchased

                itemTotalPrice = collumnTwoFloat \* collumnThreeFloat

                itemTotalPriceString = str(itemTotalPrice)

                # Condensing variables and strings into 1 variable for latter use

                newLine = collumnThree + ", " + collumnOne + ", " + itemTotalPriceString + "\n"

                # Writing new format into another text file

                purchaseItemFrequencyTotalPrice.write(newLine)

        # Opening a file in read mode

        with open("purchaseItemFrequencyTotalPrice.txt", "r") as purchaseItemFrequencyTotalPrice:

            # Setting global variables

            global subTotalPriceFloat

            global totalPriceFloat

            # Setting variable as 0

            subTotalPriceFloat = 0

            # Defining the collumns and then turning the collums with numbers in them into floats and then adding 'collumnThreeFloat' value to the 'subTotalPriceFloat' variable. It is doing this for each line

            for line in purchaseItemFrequencyTotalPrice:

                collumns = line.strip().split(",")

                collumnThree = collumns[2]

                collumnThreeFloat = float(collumnThree)

                subTotalPriceFloat += collumnThreeFloat

            # Finding the total price after tax

            totalPriceFloat = round(subTotalPriceFloat \* 1.1)

        # Calling the function

        customerRecipt()

    # Function that creates window to make sure user is ready to purchase the items.

    def continueToCheckout():

        global continueToCheckoutGui

        # Creating tkinter window

        continueToCheckoutGui = tk.Toplevel(buyGui)

        continueToCheckoutGui.geometry(windowSize)

        continueToCheckoutGui.title("Continue To Checkout")

        # Creating empty listbox

        checkoutListbox = tk.Listbox(continueToCheckoutGui)

        checkoutListbox.pack()

        # Filling list box with items customer wishes to purchase

        for item in cartListbox.get(0, tk.END):

            checkoutListbox.insert(tk.END, item)

        # Creates a button to close the window if they wish to add/remove stuff from cart

        closeWindow = tk.Button(continueToCheckoutGui, text="Back To Cart", width=buttonWidth, height=buttonHeight, command=continueToCheckoutGui.destroy)

        closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Creates button to call a function

        sendToReciptButton = tk.Button(continueToCheckoutGui, text="Confirm Purchase And \n Print Recipt", width=buttonWidth, height=buttonHeight, command=purchaseOrganise)

        sendToReciptButton.pack(padx=10, pady=10)

    # Funcion to find out user name to make recipt

    def customerRecipt():

        # Making tkinter window

        customerReciptGui = tk.Toplevel(buyGui)

        customerReciptGui.geometry(windowSize)

        customerReciptGui.title("Making Receipt")

        # Adding a label to explain what user should enter

        label = tk.Label(customerReciptGui, text="Enter Customer Name")

        label.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Adding entry box so user can enter the customers name

        customerName = tk.Entry(customerReciptGui)

        customerName.pack(padx=10, pady=10)

        # Function that is making recipt with all its details

        def makeRecipt():

            # Opening text file in read mode and setting its contents as a variable

            with open("purchaseItemFrequencyTotalPrice.txt", "r") as purchaseItemFrequencyTotalPrice:

                purchasedItems = purchaseItemFrequencyTotalPrice.read()

            # Getting name from the entry box and setting it as usable variable

            customerNameToReceipt = customerName.get()

            # Setting curent date and time that the recipt is made

            currentDateTime = datetime.datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")

            # Joining the 2 variables together

            cutomerNameAndTime = customerNameToReceipt + " " + currentDateTime

            # Turning the subtotal and the total into strings

            subTotalPriceStr = str(subTotalPriceFloat)

            totalPriceStr = str(totalPriceFloat)

            # Creating a text file with the customers name and the date and time the recipt was made as its name. It then writes all values required into the text file.

            with open(cutomerNameAndTime + ".txt", "w") as newReceipt:

                newReceipt.write("Community Store\n" +

                                "\n" +

                                "Customer: " + customerNameToReceipt + "\n" +

                                "\n" +

                                "Date & Time: " + currentDateTime + "\n" +

                                "\n" +

                                "Amount, Item, Price\n" +

                                purchasedItems +

                                "\n" +

                                "SUBTOTAL: $" + subTotalPriceStr + "\n" +

                                "\n" +

                                "TAX $10.00% ON $"+ subTotalPriceStr + "\n" +

                                "\n" +

                                "TOTAL: $" + totalPriceStr)

            # Closing all unessasary windiows

            customerReciptGui.destroy()

            continueToCheckoutGui.destroy()

            buyGui.destroy()

        # Creating tkinter button to call a function

        ReceiptButton = tk.Button(customerReciptGui, text="Write New Receipt", width=buttonWidth, height=buttonHeight, command=makeRecipt)

        ReceiptButton.pack(padx=buttonSpacingX, pady=buttonSpacingY)

        # Creating a tkinter button to close the GUI

        closeWindow = tk.Button(customerReciptGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=customerReciptGui.destroy)

        closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

    # Creates button to call a function

    addToCart = tk.Button(buyGui, text="Add To Cart", width=buttonWidth, height=buttonHeight, command=addToCartFunction)

    addToCart.grid(row=4, column=1)

    label = tk.Label(buyGui, text="                        ")

    label.grid(row=1, column=2)

    # Creates an empty listbox

    cartListbox = tk.Listbox(buyGui)

    cartListbox.grid(row=1, column=3)

    # Creates button to call a function

    removeFromCart = tk.Button(buyGui, text="Remove From Cart", width=buttonWidth, height=buttonHeight, command=removeFromCartFunction)

    removeFromCart.grid(row=2, column=3)

    # Creates button to call a function

    purchaseButton = tk.Button(buyGui, text="Continue To Checkout", width=buttonWidth, height=buttonHeight, command=continueToCheckout)

    purchaseButton.grid(row=3, column=3)

    # Creats button to close the window

    closeWindow = tk.Button(buyGui, text="Close Window", width=buttonWidth, height=buttonHeight, command=buyGui.destroy)

    closeWindow.grid(row=4, column=3)

# Creates original tkinter window and creates the buttons to run the rest of the program

root = tk.Tk()

root.geometry(windowSize)

# Creates button to call the updateCategory function

categoryButton = tk.Button(root, text="Category Options", width=buttonWidth, height=buttonHeight, command=updateCategory)

categoryButton.pack(padx=buttonSpacingX, pady=buttonSpacingY)

# Creates button to call the updateStock function

stockButton = tk.Button(root, text="Stock Options", width=buttonWidth, height=buttonHeight, command=updateStock)

stockButton.pack(padx=buttonSpacingX, pady=buttonSpacingY)

# Crearts button to call the buy function

buyButton = tk.Button(root, text="Buy", width=buttonWidth, height=buttonHeight, command=buy)

buyButton.pack(padx=buttonSpacingX, pady=buttonSpacingY)

# Creates button to close the program

closeWindow = tk.Button(root, text="Close Window", width=buttonWidth, height=buttonHeight, command=root.destroy)

closeWindow.pack(padx=buttonSpacingX, pady=buttonSpacingY)

# Starts event loop (looking for user events such as button presses)

root.mainloop()

# Part 3 – Evaluation

## Problems & Improvements

* When entering new items, the program does not stop users from typing symbols, numbers, and letters in places there not meant to be. Does not break program and only creates inconvenience.
* I can improve the look of the program with more time along with making it easier to interact with, quality of life things. The program runs successfully, and I have not encountered any fatal error that stop or break the program.

## Developer Summary

I feel after looking back through part 1 and 2 requirements that I have satisfied all points that the program and planning should meet. The program runs flawlessly completing all parameters set out by the friend who requested it. The planning for the program meets all required parameters as well and I believe it will be useful tool in the community store future.

## Sources

*Chat GPT*.

*One Note*.

“How to Check If the String Is Integer in Python.” *FavTutor*, https://favtutor.com/blogs/check-string-is-integer-python. Accessed 21 Mar. 2024.

*Python Delete File*. https://www.w3schools.com/python/python\_file\_remove.asp. Accessed 21 Mar. 2024.

*Python Object Oriented Programming (OOP) - For Beginners*. *www.youtube.com*, https://www.youtube.com/watch?v=JeznW\_7DlB0. Accessed 21 Mar. 2024.

“Python Switch Statement – Switch Case Example.” *freeCodeCamp.Org*, 5 Aug. 2022, https://www.freecodecamp.org/news/python-switch-statement-switch-case-example/.

user71346. “Error Handling That Prompt the User to Enter Only Integer Greater than 1.” *Stack Overflow*, 21 Sept. 2018, https://stackoverflow.com/q/52439284.

“Writing and Reading Existing Txt. File.” *McNeel Forum*, 1 Mar. 2023, https://discourse.mcneel.com/t/writing-and-reading-existing-txt-file/155672.

## Chat GPT Conversation.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer program

Description automatically generated ­­­­­