­­­­

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](#_Toc164172511)

[Part 1 – Planning 2](#_Toc164172512)

[Tasks to be done part 1. 2](#_Toc164172513)

[Tasks to be done part 2 Develop. 2](#_Toc164172514)

[Tasks to be done part 2 Evaluate. 2](#_Toc164172515)

[Time frame 3](#_Toc164172516)

[Problem Outline 3](#_Toc164172517)

[Problem Description 3](#_Toc164172518)

[Pseudocode 4](#_Toc164172519)

[Structure Chart 7](#_Toc164172520)

[Part 2 – Development 8](#_Toc164172521)

[Required files. 8](#_Toc164172522)

[Files 8](#_Toc164172523)

[Folder: 8](#_Toc164172524)

[Python: 8](#_Toc164172525)

[Project code 8](#_Toc164172526)

[Part 3 – Evaluation 16](#_Toc164172527)

[Description 16](#_Toc164172528)

[Development process 16](#_Toc164172529)

[Problems & Improvements 17](#_Toc164172530)

[Developer Summary 17](#_Toc164172531)

[Sources 17](#_Toc164172532)

[Chat GPT Conversation. 18](#_Toc164172533)

# 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 3 main functions. 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

START

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”

INCREMENT Row number

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 receipt information

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 receipt information

PRINT receipt information

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.”

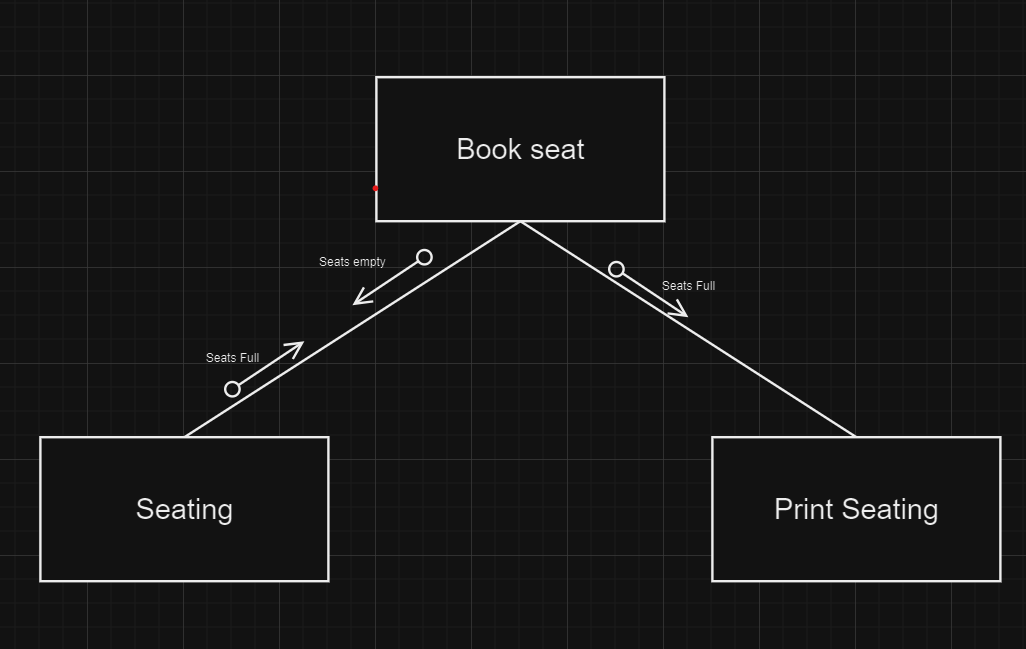
Call FUNCTION Program Function

ENDCASE

ENDFUNCTION

Call FUNCTION Program Function

## Structure Chart



# Part 2 – Development

## 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 os

import glob

import random

#Save directory path to a variable

directory= os.getcwd()

#Defines CLASS Booking

class Booking:

    #Defines METHOD Seating

    def seating(self,seats):

        #Creates the array that contains the seating arrangement

        seats = [

        [1, 0, 0, 0, 1, 1, 0, 0, 1, 0],

        [0, 0, 0, 0, 1, 0, 1, 0, 0, 0],

        [0, 1, 0, 0, 0, 1, 0, 1, 0, 0],

        [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],

        [0, 0, 1, 1, 0, 0, 0, 1, 1, 0],

        [1, 0, 1, 0, 0, 0, 1, 0, 0, 0],

        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0],

        [0, 0, 1, 0, 1, 0, 1, 0, 0, 0],

        ]

        #Return array to the rest of the program

        return(seats)

    #Defines METHOD printSeating

    def printSeating(self, seats):

        #Print letters

        print("  A B C D E F G H I J")

        #Creates variable ans sets it to 1

        rowNumber = 1

        #For row in seats prints the row number and row. If value in array = 0 it prints - and if value = 1 it prints X

        for row in seats:

            print(f"{rowNumber} ", end='')

            rowNumber += 1

            for seat in row:

                if seat == 0:

                    print('-', end=' ')

                else:

                    print('X', end=' ')

            print()

    #Defines METHOD bookSeat

    def bookSeat(self):

        #Gets the seats and prints them from the METHOD seating and printSeating

        print("\n\n\nCurrent seating plan:")

        seats = []

        seats = b.seating(seats)

        b.printSeating(seats)

        #Exception handling. Try the user input and if incorect value is entered runs the except and prints the error statement and runs the METHOD bookSeat

        try:

            howManySeats= int(input("how many seats do you want to book? "))

        except (IndexError, ValueError):

            print("Invalid input format. Please use integers.")

            b.bookSeat()

        #Creates the personPrices dictionary

        personPrices = {

            "adult": {

                "name": 'a',

                "price": 30.00

            },

            "child": {

                "name": 'c',

                "price": 10.00

            },

            "student": {

                "name": 's',

                "price": 5.00

            },

            "concession\_holder": {

                "name": 'h',

                "price": 15.00

            }

        }

        #Creates an empty array

        results = []

        #Makes sure that the selection is more than 0

        if howManySeats > 0:

            #Calls the METHOD processingBooking

            b.processingBooking(results,personPrices,howManySeats,seats)

        #If selection was less than 0

        else:

            #Print error message and calls the METHOD bookSeat

            print("you have not entered an integer above 0. Please try again")

            b.bookSeat()

    #Defines METHOD processingBooking

    def processingBooking(self, results,personPrices,howManySeats,seats):

        #Converts variable into an integer

        howManySeats = int(howManySeats)

        #Saves how many seats user wanted to book in a different variable for latter use

        totalSeats = howManySeats

        #Creates an empty array

        seatsBooked = []

        #While howManySeats is greater than 0 it will keep iterating through the loop

        while howManySeats > 0:

            #Gets user input for what seat they want to book

            selection = input("Choose a seat (e.g., 2D): ")

            #Gets user input

            typePersonInput = input("If seat is for adult type: A\nIf seat is for a child type: C\nIf seat is for a student type: S\nIf seat is for a concession holder type: H\nWho is seat for? ")

            #Print selection

            print(selection[0])

            #Checks if selection[0] is not a digit

            if not selection[0].isdigit():

                #Print error message and calls METHOD processingBooking

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

                b.processingBooking(results,personPrices,howManySeats,seats)

            #If is a digit

            else:

                #Tries the folowing code

                try:

                    #Seperates the row and column index from the selection

                    row = int(selection[0]) - 1

                    col = ord(selection[1].upper()) - ord('A')

                    #Checks if selection is within the range of the array

                    if 0 <= row < len(seats) and 0 <= col < len(seats[0]):

                        if seats[row][col] == 0:

                            #Check if typePersonInput is in person prices

                            for staff\_id, record in personPrices.items():

                                if typePersonInput.lower() in record["name"].split()[-1].lower():

                                    #Change the item in array from 0 to 1

                                    seats[row][col] = 1

                                    #Update the result array

                                    results.append(record)

                                    #Subtracts 1 from the variable untill the while loop stops

                                    howManySeats -= 1

                                    #Prints a message to say booking was succesfull and updates the array

                                    print("Seat booked successfully!\n")

                                    seatsBooked.append(selection)

                                    #Stops the loop

                                    break

                                #If item is not in record

                                else:

                                    #Print error message and continue on with program

                                    print("Invalid type of person input. Please try again")

                                    continue

                        #If does not equal 0  prints error message

                        else:

                            print("Sorry, that seat is already taken.\n")

                    #If seat is out of array range print error message

                    else:

                        print("Invalid seat selection.\n")

                #If code breaks in try prints error message

                except (IndexError, ValueError):

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

                #Print the seating plan

                print("Updated seating plan:")

                b.printSeating(seats)

        #Calls METHOD receiptFunction

        b.receiptFunction(results, totalSeats, seatsBooked)

    #Defines METHOD recieptFunction

    def receiptFunction(self, results, totalSeats, seatsBooked):

        #Turns variable into string and removes unwated characters

        seatsBooked=str(seatsBooked).replace("'","").replace("[","").replace("]","")

        #Gets a random number

        randomNumber = ''.join(random.choices('0123456789', k=8))

        #Searches for all txt files and saves there names to a variable

        txtFiles = glob.glob(os.path.join(directory, "\*.txt"))

        fileNames = [os.path.basename(txtFile) for txtFile in txtFiles]

        #For fileName in fileNames it changes unwated characters and splits the name into parts

        for fileName in fileNames:

            fileName = fileName.replace(".","\_")

            fileName = fileName.split("\_")

            #Gets the part of file name that has the reciept ID

            receiptID = int(fileName[1])

            #Checks if any of the existing receipts have the same number as the randomly generated one

            if receiptID == int(randomNumber):

                b.receiptFunction()

            else:

                continue

        #Sets variable to equal 0

        price = 0

        file = open(f"Reciept\_{randomNumber}.txt", "w")

        file.write(f"======================================\nReciept ID: {randomNumber}\nTotal seats Booked: {totalSeats}\nSeats Booked: {seatsBooked}\n\nType Of Seat - Price - Runing Total\n")

        #For result in results it turns to a string and splits it

        for result in results:

            result = str(results).split()

            #Gets the type of person from the string and replaces the placeholder letter with the full name

            resultTypePerson = result[1]

            resultTypePerson = resultTypePerson.replace("h","CONCESSION HOLDER").replace("a","ADULT").replace("c","CHILD").replace("s","STUDENT").replace(",","").replace("'","")

            #Gets the price related to the type of person and removes unwated characters before tunring into a float

            resultPrice = result[3]

            resultPrice = str(resultPrice).replace("}","").replace(",","").replace("'","").replace("]","")

            resultPrice = float(resultPrice)

            #Adds the price for the person to the total price

            price = price + resultPrice

            #Adds the type of person, price for there ticket and running price to the ticket/reciept

            file.write(f"{resultTypePerson} - {resultPrice} - {price}\n")

        #Writes total price to the ticket and closes the file

        file.write(f"\nTotal Price: {price}\n======================================\n\n\n\n")

        file.close()

        #Prints how many seats the customer booked, the total cost and the receipt ID

        print(f"\nyou have booked {totalSeats}. This will cost ${price}\nA reciept will be sent to you soon\nYour reciept ID is: {randomNumber}")

        programFunction()

#Defines CLASS Cancel

class Cancel:

    #Defines METHOD userInput

    def userInput(self):

        #Tries the user input

        try:

            userInput = int(input("Please enter your ticket ID: "))

        #If encounters an error print error message and call METHOD userInput

        except (IndexError, ValueError):

            print("Invalid input format. Please use integers.")

            c.userInput()

        #Calls METHOD findTicket

        c.findTicket(userInput)

    #Defines METHOD findTicket

    def findTicket(self, userInput):

        #Search for all text files in folder and saves the name into a variable

        txtFiles = glob.glob(os.path.join(directory, "\*.txt"))

        fileNames = [os.path.basename(txtFile) for txtFile in txtFiles]

        #For fileName in fileNames it changes unwated characters and splits the name into parts and saves original name to a variable

        for fileName in fileNames:

            originalFileName = fileName

            fileName = fileName.replace(".","\_")

            fileName = fileName.split("\_")

            #Gets the part of file name that has the reciept ID

            recieptID = int(fileName[1])

            #Checks if receipt ID equals users input

            if recieptID == userInput:

                #Calls METHOD getsSeats and stops the loop

                c.getSeats(originalFileName)

                break

            #If doesnt equal userinput continue the program

            else:

                continue

        #Print error message and calls FUNCTION programFunction

        print(f"There is no ticket with ID of '{userInput}'")

        programFunction()

    #Defines METHOD getSeats

    def getSeats(self,originalFileName):

        #Opens file in read mode and reades the lines

        file = open(originalFileName,"r")

        lines = file.readlines()

        file.close()

        #Saves the fourth line of the txt file

        fourthLine = lines[3]

        fourthLine = fourthLine.split(":")

        #Seperates the seats from the rest of information on that line

        ticketSeats = fourthLine[1].replace(" ","")

        ticketSeats = ticketSeats.replace("'","").replace("\n'","")

        #Calls METHOD vacateSeats

        c.vacateSeats(ticketSeats, originalFileName)

    #Defines METHOD vacateSeats

    def vacateSeats(self, ticketSeats, originalFileName):

        #Gets seats array from METHOD seating in class booking

        seats = []

        seats = b.seating(seats)

        #Tunrs into a string and splits them into individual seats

        ticketSeats = str(ticketSeats).split(",")

        #For ticketseat in ticketSeats

        for ticketseat in ticketSeats:

            #Seperates the row and column index from the selection

            row = int(ticketseat[0]) - 1

            col = ord(ticketseat[1].upper()) - ord('A')

            #Checks if selection is within the range of the array

            if 0 <= row < len(seats) and 0 <= col < len(seats[0]):

                #Checks if item in array is equal to 1

                if seats[row][col] == 1:

                    #Change the item in array from 1 to 0

                    seats[row][col] = 0

                #If seat is not equal to 1 print error message and continue the program

                else:

                    print("Seat is already empty. Continuing onto next seat\n")

                    continue

            #If not in array print error message and continue with program

            else:

                print("Sorry, we encounted an error. Continuing to next seat.\n")

                continue

        #Remove the text file from folder

        os.remove(originalFileName)

        programFunction()

#Defines function to stop the program

def endFunction():

    exit()

#Defines function that starts the program

def programFunction():

    print("\n\n\n\n\nTo book a ticket type 'Booking' \nTo cancel a booking type 'Cancel' \nTo end program type 'End'\n")

    userInput = input("What would you like to do? ")

    userInput = userInput.lower()

    #Checking if user input matches and running related code

    match userInput:

        case "booking":

            b.bookSeat()

        case "cancel":

            c.userInput()

        case "end":

            endFunction()

        case \_:

            print("\nIncorrect entry please check your spelling and try again\n")

            programFunction()

b = Booking()

c = Cancel()

programFunction()

# Part 3 – Evaluation

## Description

I believe my program meets all requirements I outlined in part 1. It allows the user to book out seats to a show and get a ticket sent to them and allows the user to cancel there booking and open the seats back up to be booked out by someone else. It provides a relatively simple design and is easy to use. The user just needs to follow the prompts that appear in the terminal. When they launch the program, they get asked what they want to do, book a seat, cancel a booking, or close the program. User simply needs to type what they want to do.

If they type ‘booking’ it, then shows them a seating plan showing what seats are taken and what are free it then asks how many of these seats do you want to book. After you type in an integer it then asks you what seat you would like to book and makes sure you use the right format. It then asks the user if they are a student, adult, child, etc and corrects the user if they enter the wrong value. After this it tells the use how many seats they have booked and the total cost. It also provides the user with there receipt ID which can be used to cancel a booking.

If they had typed ‘cancel’ at the start it would have asked them for their ticket ID which they then have to enter it will then run through all the seats in the ticket and make them available again in the array for someone to book out again.

If they had typed ‘end’ it will simply stop the program from running.

## Development process

I had started the development process much latter than I had wished and did not finish it on time due to starting late. I encountered difficulties with object-oriented programming and gaps in my knowledge led to the use of just the basics of it. Given more time and a better understanding of object-oriented programming the development of this software would have been much quicker and to a higher quality. I made use of stubs and control structures to help my coding which can be found in previous commits on my GitHub. [SpicyLeviathan/Task\_1 (github.com)](https://github.com/SpicyLeviathan/Task_1)

### Problems & Improvements

* If the program is closed it does not update the available seats array which means that multiple people can book the same seats
* It does not limit the number of seats you can book which means you can book out more seats than are available.
* You can not get tickets for more than 1 show.
* Does not email out the receipt or record the email of the person booking so you can’t email out the ticket to whoever booked the seats.
* People can lie about being a child or adult and get tickets for cheaper.

## Developer Summary

I feel after looking back through part 1 and 2 requirements that I have satisfied most points that the program and planning should meet. The program runs with minimal flaws completing all parameters set out by the school. The planning for the program meets all required parameters as well and I believe with a little more time this program will be of great use to the school in the 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 ­­­­­