Technical Design Document

# Author

Joseph D Sullivan <[JSulli40@Student.SCF.edu](mailto:JSulli40@Student.SCF.edu)>

# Date

January 25, 2025

# Chapter

01

# Assignment

01

# Repository

<https://github.com/JosephDSullivan/COP2373/blob/main/src/chapter01/jsulli40_chapter01_assignment01.py>

# Program Description

This program is designed to pre-sell a limited number of cinema tickets.

The program prompts users for their desired number of tickets and displays the remaining count after each transaction until all tickets are sold. At completion, the user is notified, and the total number of transactions is displayed.

# Logic

1. The program starts by calling the main function.
2. main calls get\_sale with parameters for total tickets and maximum tickets per transaction.
3. get\_sale manages ticket transactions, calling get\_trans for each buyer.
4. get\_trans collects and validates user input using validate\_trans.
5. validate\_trans ensures input integrity before proceeding with the sale.
6. The program loops until all tickets are sold, then terminates with a final message.

# Functions

## main

Entry function for when code is invoked directly.

### Parameter(s)

* None

### Constant(s)

* None

### Variable(s)

* None

### Logic

1. Calls the get\_sale function with parameters for the total tickets available (20) and the maximum tickets per transaction (4).

### Return(s)

* None

## get\_sale

Manages ticket sales until all tickets are sold.

### Parameter(s)

* total\_ticket\_avail (int): Total number of tickets available for sale.
* max\_per\_trans (int): Maximum tickets allowed per transaction.

### Constant(s)

* None

### Variable(s)

* ticket\_remain (int): Tracks the remaining tickets.
* msg\_init (str): Welcome message for the user.
* msg\_sale (str): Displays transaction details.
* msg\_final (str): End-of-sale message.

### Logic

1. Display an initial welcome message with the total tickets available.
2. Enter a loop to process transactions until tickets run out.
3. Call get\_trans to get a valid transaction from the user.
4. Subtract the tickets sold from the total remaining.
5. Display details of the transaction.
6. Notify the user when all tickets are sold.

### Return(s)

* None

## get\_trans

Prompts the user for the number of tickets they want to purchase and validates their input.

### Parameter(s)

* max\_per\_trans (int): Maximum tickets allowed per transaction.

### Constant(s)

* None

### Variable(s)

* msg (str): Prompt message for the user.
* is\_trans\_valid (bool): Tracks whether the input is valid.
* input\_trans (str): User input for the number of tickets.

### Logic

1. Prompt the user for input.
2. Validate the input using validate\_trans.
3. Repeat until the input is valid.
4. Return the validated number of tickets as an integer.

### Return(s)

* int: Validated number of tickets.

## validate\_trans

Validates the user's transaction input.

### Parameter(s)

* trans (str): User input to validate.
* max\_per\_trans (int): Maximum tickets allowed per transaction.
* show\_warning (bool): Flag to show warnings for invalid input

### Constant(s)

* None

### Variable(s)

* trans\_int (int): Converted integer value of the user input

### Logic

1. Attempt to convert the input to an integer.
2. Check if the integer is non-negative and within the allowed range.
3. Display appropriate warnings for invalid input.
4. Return True if valid; otherwise, False.

### Return(s)

* bool: True if the transaction is valid, False otherwise.