Requirements Document

Dale Alleshouse, Joseph Borgert, Ryan Cridelich, Jose Lozada, and Joshua Patton

PRG 410

June 20, 2016

Gholam Ali Shaykhian

Requirements Document

This weeks’ assignment involves our team creating a short requirements document that will serve as an outline for what is required for the rest of the program to be functional and complete. This will include designating the proper input, output and validation rules to follow through the application.

The program must be a Win32 console application written in C++. The purpose of the program is to track the sale of tickets for a movie theater that has 90 chairs. The seats are arranged with ten rows of nine seats.

The program outputs will print directly to a console screen and are the direct result of user input. Outputs for the program include: number of seats available, the total number of tickets being sold, the total cost of the tickets, sale confirmation, and a visual display of the seats available. In addition, there are output errors associated with invalid user input.

The validation rules for the application are as follows:

* seat number = integer between 1 and 9
* row number = R1, R2, R3, R4, R5, R6, R7, R8, R9, R10 (case insensitive)
* commands – commands specified in the *User Commands* section below (case insensitive)

Any input falling outside of the parameters above will result in the following screen output.

> invalid input

I’m sorry, I don’t understand that statement. Please try again.

User Commands

The list outlines the commands defined by the program and their output:

* total – displays the total number of tickets sold and the total amount of money collected.

> total

Total Tickets Sold: ##

Total Sales: $##.##

* seats [row\_number] – displays the total number seats available in the specified row or total seats available in the theater if row\_number is omitted.

> seats

Seats Available: ##

> seats R1

Seats Available in R1: ##

* seatchart – displays available seats in a matrix with a “\*” character representing an occupied seat and a “#” character representing an open seat.

> seatchart

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |  |  |  |  |  |  |  |  |  |
| R 1 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R 2 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R 3 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R4 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R5 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R 6 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R 7 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R 8 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R 9 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |
| R 10 | \* | \* | # | # | # | \* | \* | # | # |  |  |  |  |  |  |  |  |  |  |

* sale [row\_number] [seat\_number] – Allocates a seat as occupied and returns the total price. The user must confirm the price with a y in order to complete the sale. Upon confirmation, the sale will be recorded in the program. If the specified is already occupied, a message of “Seat not available” is shown.

> sale R1 5

Total: $##.##

confirm purchase (y/n)

> sale R2 6

Seat not available