Jacob McKenzie N10260480

CAB301 Assignment

DVD Management Software

Contents

[Algorithm for Displaying the Top Ten 2](#_Toc41313235)

[Pseudo Code 2](#_Toc41313236)

[Time complexity analysis 3](#_Toc41313237)

[Functional tests 4](#_Toc41313238)

[Main menu 4](#_Toc41313239)

[Staff Login 4](#_Toc41313240)

[Staff Menu 5](#_Toc41313241)

[Add DVD Menu 5](#_Toc41313242)

[Remove Movie Menu 6](#_Toc41313243)

[Register new member Menu 6](#_Toc41313244)

[Find Member Phone Number Menu 7](#_Toc41313245)

[Member login Menu 7](#_Toc41313246)

[Client Menu 8](#_Toc41313247)

[Display all DVDs Menu 8](#_Toc41313248)

[Borrow DVD Menu 10](#_Toc41313249)

[Return DVD Menu 10](#_Toc41313250)

[Show Loaned Menu 11](#_Toc41313251)

[Top Ten Menu 12](#_Toc41313252)

[Appendix 14](#_Toc41313253)

# Algorithm for Displaying the Top Ten

## Pseudo Code

function TopTen(root)

    //Declare a new empty Binary tree.

    d <- ∅

    //itterate over the Lexographicly sorted array from the current binary tree

    foreach (movie in root.ToArray())

        //if null then we initilise the new binary tree

        if(d = null)

            d.Value <- new BinaryTree(movie)

        //otherwise we add the movie to the new tree sorting by popularity

        else

            //initilise at the point in the tree we are working with

            point <- d.Value

            //initilise a temporay value to let us update the point refrence if

            //the next child is null

            temp <- null

            //Create a new tree with just the thing we want, this is the leaf

            //we will eventually add

            toAdd <- new BinaryTree(movie)

            //while we aren’t at the point to add

            while(true)

                //assign the current point to temp, to save the refrence

                temp <- point

                //compare the current node with the value to add.

                //if the node is less then move the current point down the left branch

                if(point.Value.Popularity < toAdd.Value.Popularity)

                    point <- point.LeftChild

                    //if that branch is a dead end, don't worry,

                    //just update the refrence held by temp, and then we're done.

                    if(point = null)

                        temp.LeftChild <- toAdd

                        break

                //otherwise we travel down the right branch.

                else

                    point <- point.RightChild

                    //again, if that branch does not exist,

                    //just assign it to our added value and exit.

                    if(point = null)

                        temp.RightChild <- toAdd

                        break

    //finally, now that we have got a new binary tree sorted by popularity,

    //Convert it to an array for easy iteration in display,

    //and return the up to ten items.

    return d.ToArray().Range(0,10)

 function BinaryTree.ToArray()

    //Create a new Array of movies.

    movies <- ∅

    //Recursive function defined so as to access the movies array

    function RecursiveInOrder(Node)

        //if null then we don't need to keep going deeper.

        if(Node = null)

            return

        //first add all the left children

        RecursiveInOrder(Node.LeftChild)

        //then add the root

        movies.Add(Node.Value)

        //then all the right children

        RecursiveInOrder(Node.RightChild)

    //Start the recursive function at the root.

    RecursiveInOrder(BinaryTree)

    //return the arrayed movies

    return movies

## Time complexity analysis

The Algorithm will traverse the entire tree when iterating over it in the foreach loop, and again when converting the tree to an array in returning it. The binary insert preformed has a Average time complexity of log n, and a worst case time complexity of n. as such the average complexity will be n log n, and worst be n2.

# Functional tests

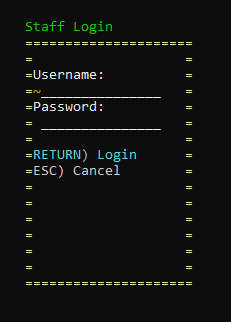
For the sake of shortening this, all menus that use the esc to exit use the same code, with just a different menu index, and they all work. As such I won’t mention them. The same is true for the main, client and staff menus, as they are static accepting simple numeric input, there is quite literally a single switch stamen of logic, and they all work. As such they have been exempt from this reporting bar, the main menu, and screenshots of all pages.

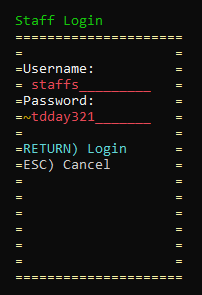
## Main menu

|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| ‘1’ key presses | The staff login menu will be displayed | As expected |
| ‘2’ key pressed | The Member login menu will be displayed | As expected. |
| ‘3’ key or ‘escape’ key pressed | The app will exit | As expected. |

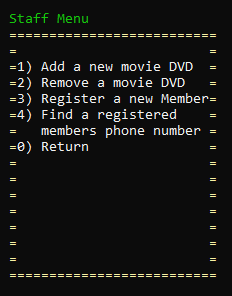
## Staff Login

|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Use ‘TAB’ or the ‘up arrow’ or ‘down arrow’ pressed | The yellow ~ changes position to reflect the selected input | As expected |
| Enter a character into the selected input using the appropriate key | The input is reflected in the selected input box. | As expected |
| Username or password incorrect, pressing ‘return’ | Fields flash red to indicate failure | As expected |
| Username ‘staff’ and password ‘today123’ entered correctly, ‘return’ pressed | The staff menu is displayed. | As expected |



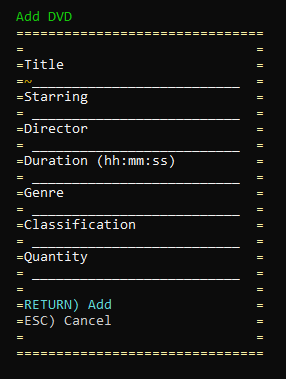


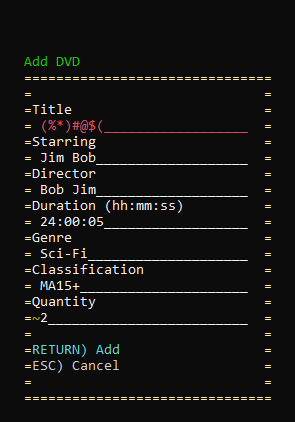
## Staff Menu



## Add DVD Menu

|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Invalid text entered into a respective field (or field empty) and ‘return’ is pressed | First sequentially invalid field flashes red | As expected |
| All fields filled with valid values and ‘return’ pressed (title already in database) | Staff menu is displayed, and the quantity of the DVD is incremented by one | As expected |
| All fields filled with valid values and ‘return’ pressed (title not in database) | Staff menu is displayed and the DVD is added to the database. | As expected |
| ‘TAB’ or ‘up arrow’ or ‘down arrow’ are pressed | The yellow ~ changes position to reflect the selected input | As expected |



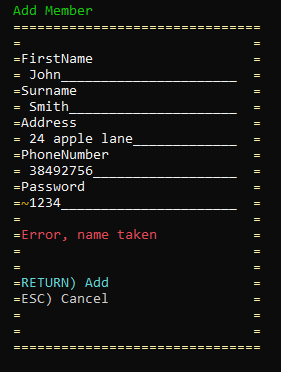


|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Invalid input in text entry or not registered film ‘return’ pressed | Text entry field flashes red and appropriate message displayed | As expected |
| Valid registered movie title in Text entry, ‘return’ pressed | Staff menu displayed | As expected |

## Remove Movie Menu

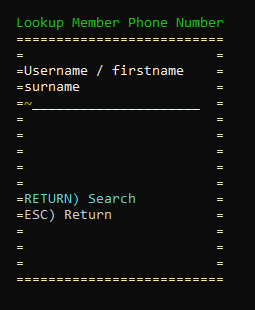
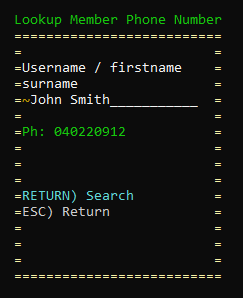
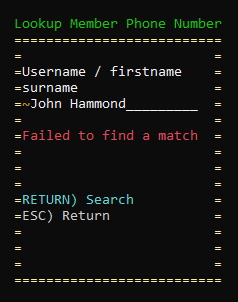
## Register new member Menu

|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Invalid text entered into a respective field (or field empty) and ‘return’ is pressed | First sequentially invalid field flashes red and error message displayed | As expected |
| All fields filled with valid values and ‘return’ pressed (user could not be added) | error message displayed | As expected |
| All fields filled with valid values and ‘return’ pressed (User does not exist) | Staff menu is displayed, and the User is added to the list of registered members | As expected |
| ‘TAB’ or ‘up arrow’ or ‘down arrow’ are pressed | The yellow ~ changes position to reflect the selected input | As expected |



## Find Member Phone Number Menu

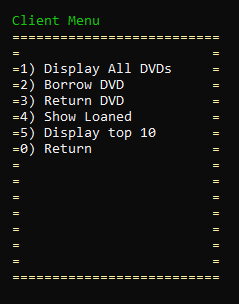
|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Invalid input in text entry or unregistered username ‘return’ pressed | Appropriate error message displayed | As expected |
| Valid registered Username / full name in Text entry, ‘return’ pressed | User phone number displayed. | As expected |



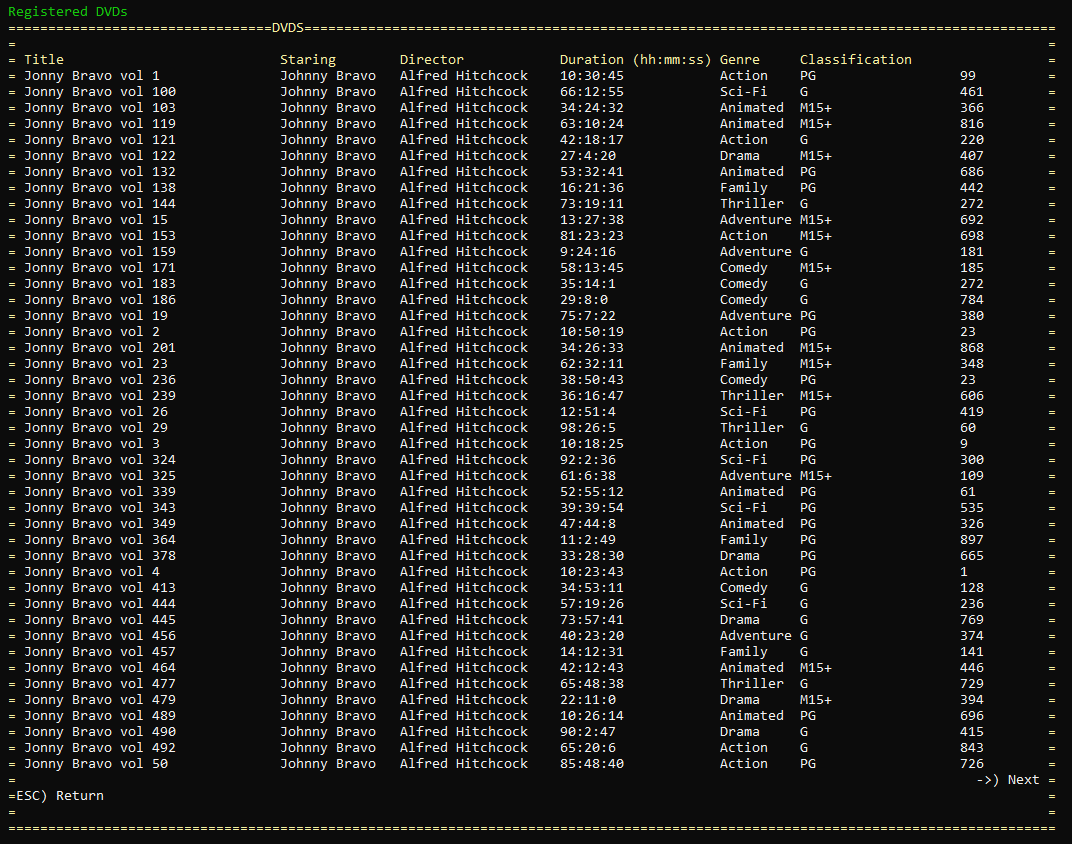
## Member login Menu

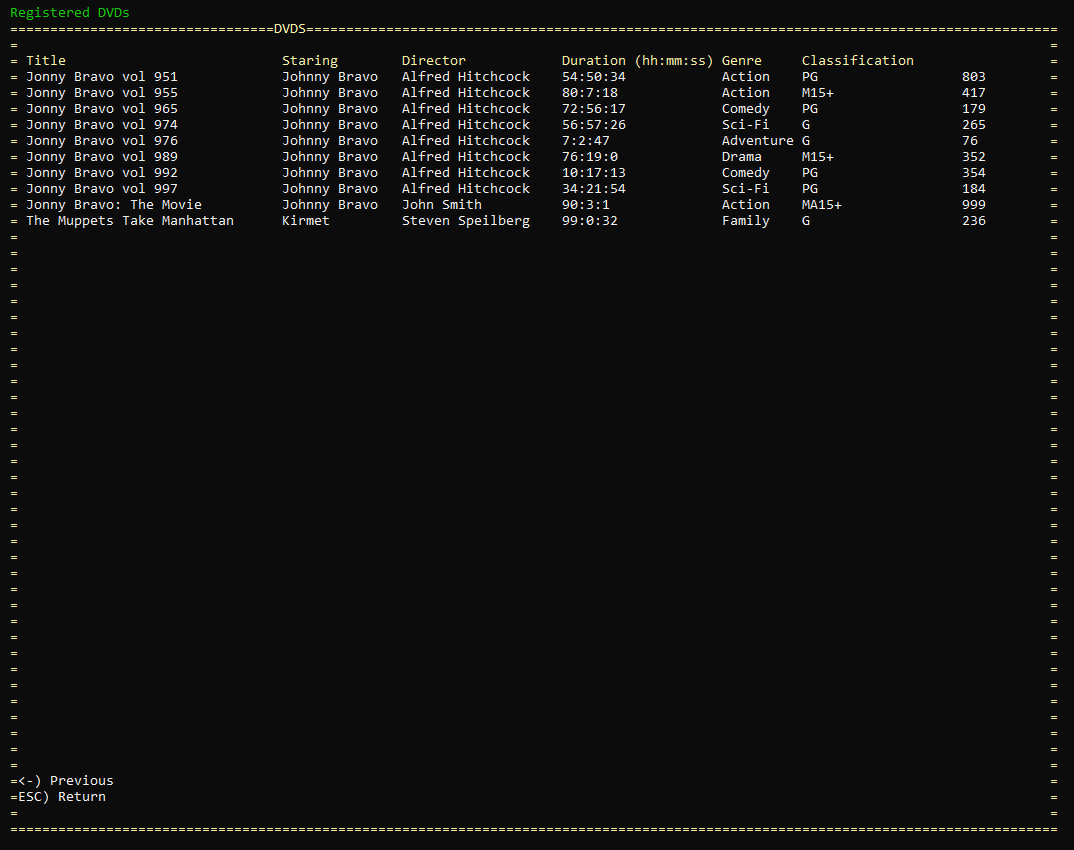
|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Use ‘TAB’ or the ‘up arrow’ or ‘down arrow’ pressed | The yellow ~ changes position to reflect the selected input | As expected |
| Enter a character into the selected input using the appropriate key | The input is reflected in the selected input box. | As expected |
| Username or password incorrect, pressing ‘return’ | Fields flash red to indicate failure | As expected |
| Valid username and password entered correctly, ‘return’ pressed | The Client menu is displayed | As expected |

## Client Menu



## Display all DVDs Menu

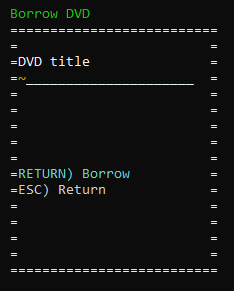
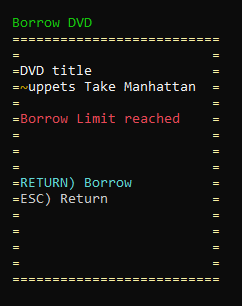


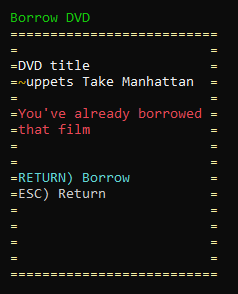


|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Page loaded | Registered movies all displayed in alphabetical order. | As expected, though only the first 44 films are displayed on a given page, arrow based navigation is required to see all of them. |
| Left arrow key pressed | If there is a page prior to the current page, that page will be displayed. | As expected |
| Right Arrow Key Pressed | If there is a page proceeding the current page, it will be displayed. | As expected. |

## Borrow DVD Menu

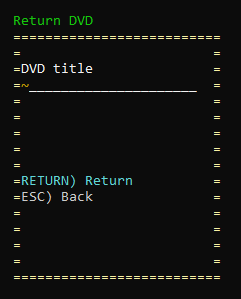
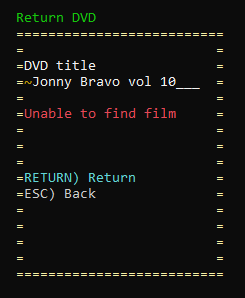
|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Invalid input in text entry or unable to borrow ‘return’ pressed | Appropriate error message displayed | As expected |
| Valid registered movie title in Text entry, ‘return’ pressed | Client Menu displayed, fill added to member borrow tree, and stocked quantity decreased | As expected |





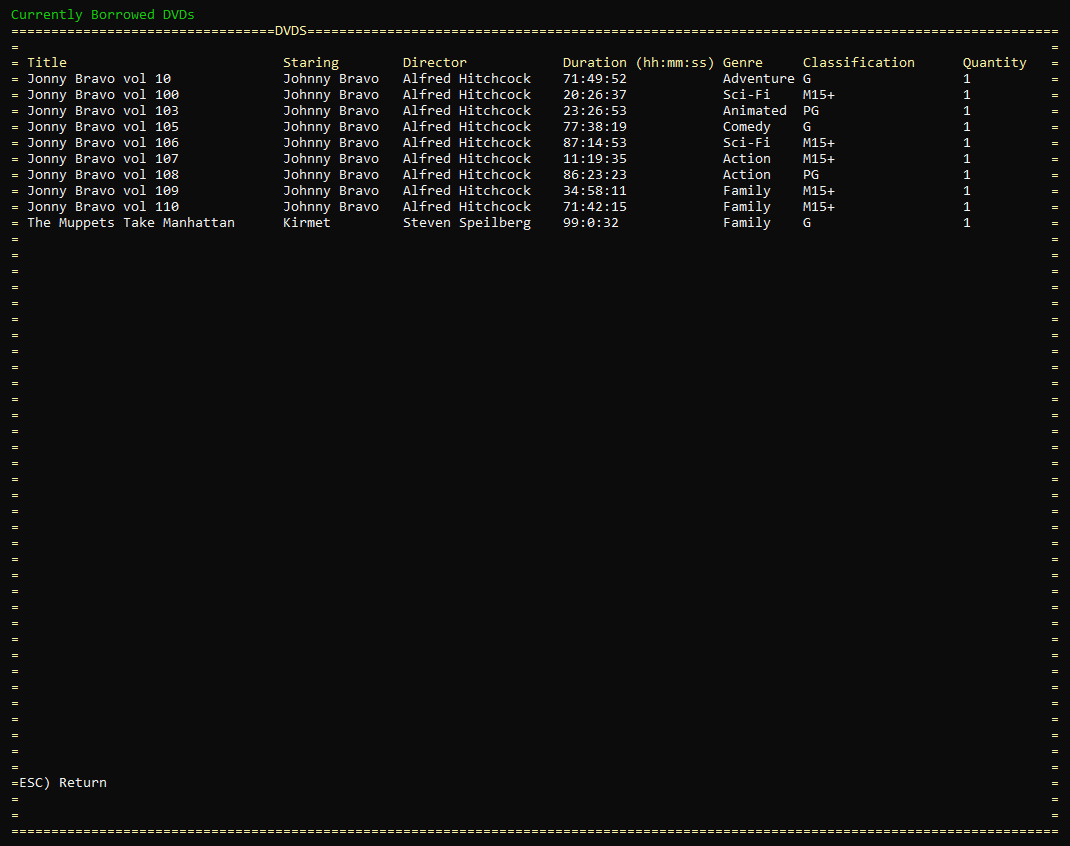
## Return DVD Menu

|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Invalid input in text entry or Film Not borrowed ‘return’ pressed | Appropriate error message displayed | As expected |
| Valid Borrowed movie title in Text entry, ‘return’ pressed | Client Menu displayed and movie returned | As expected |

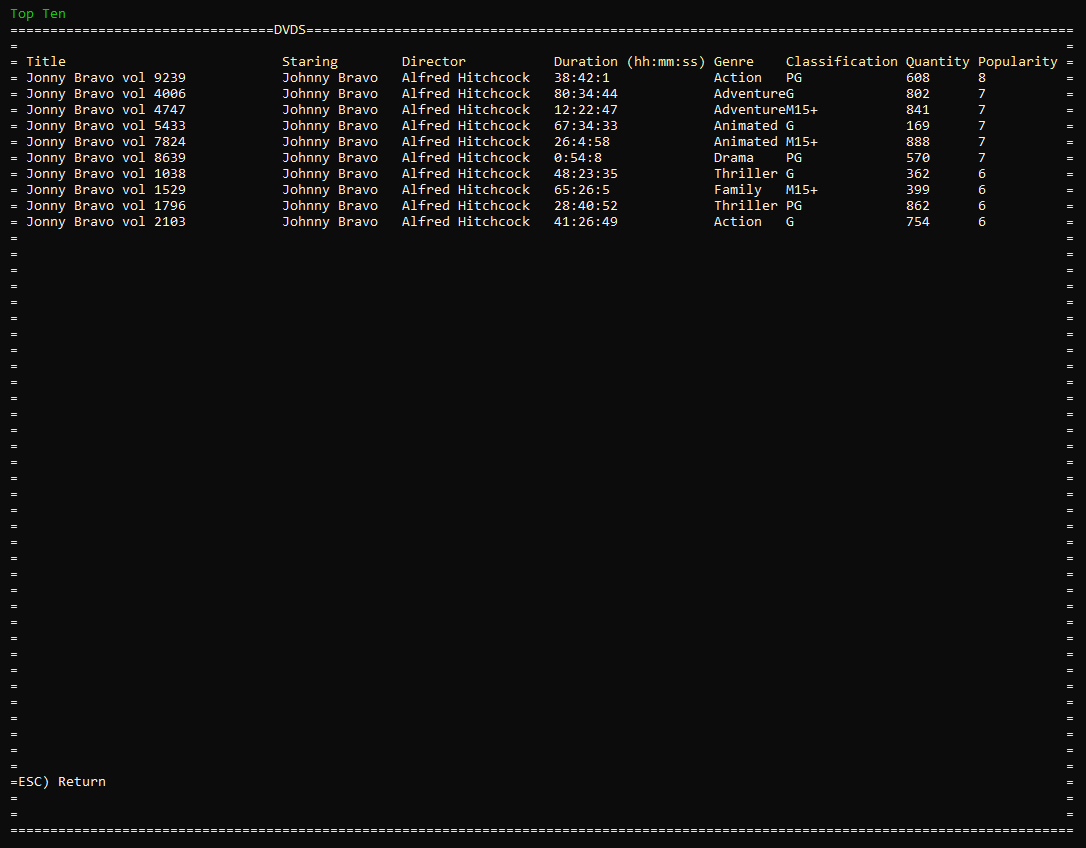


## Show Loaned Menu

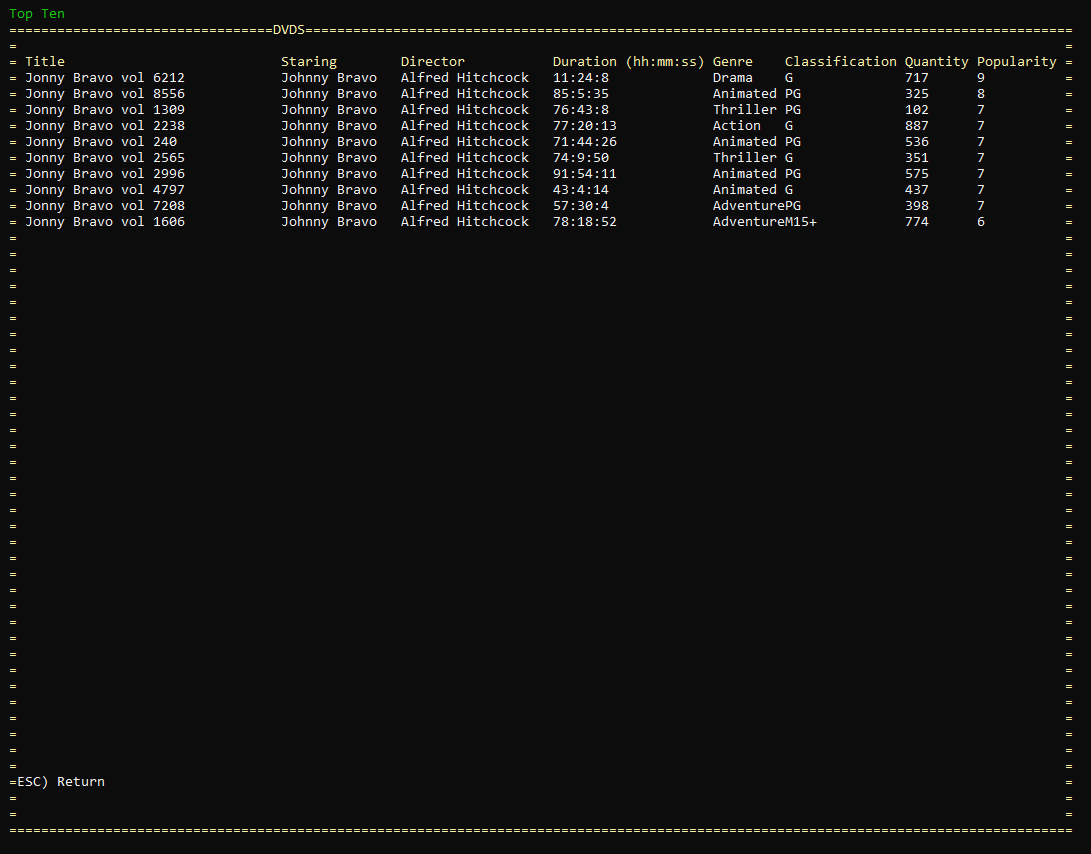
|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Page loaded | Borrowed movies all displayed in alphabetical order. | As expected. |
|  |  |  |



## Top Ten Menu



|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual |
| Page loaded | The ten most borrowed films are displayed in alphabetical order. | As expected. |
|  |  |  |



# Appendix

How to navigate tutorial

