Web Programming

Project

Student Name: Alan Pedersen

Student ID: P225139

Date: 05/04/2020

Table of Contents

[Table of Figures 2](#_Toc36913856)

[Table of Tables 2](#_Toc36913857)

[Project Background 3](#_Toc36913858)

[Project Development Environment 3](#_Toc36913859)

[Database Creation 4](#_Toc36913860)

[Introduction 4](#_Toc36913861)

[Data Review 4](#_Toc36913862)

[Data Normalisation 4](#_Toc36913863)

[Data Importation 5](#_Toc36913864)

[Additional Tables and Views 5](#_Toc36913865)

[Database Creation 6](#_Toc36913866)

[Project Design 7](#_Toc36913867)

[Web Page Design 7](#_Toc36913868)

[The Query Page 7](#_Toc36913869)

[The Top 10 Page 8](#_Toc36913870)

[Functional Design 8](#_Toc36913871)

[The Query Page 8](#_Toc36913872)

[The Top 10 Page 9](#_Toc36913873)

[Risk Analysis 9](#_Toc36913874)

[The Query Page 9](#_Toc36913875)

[The Top 10 Page 9](#_Toc36913876)

[Conclusions 9](#_Toc36913877)

[Testing 10](#_Toc36913878)

[Introduction 10](#_Toc36913879)

[Static Analysis 10](#_Toc36913880)

[Functional Testing 11](#_Toc36913881)

[The Query Page 11](#_Toc36913882)

[The Top 10 Page 15](#_Toc36913883)

[Compatibility Testing 16](#_Toc36913884)

# Table of Figures

[Figure 1: normalised database schema 6](#_Toc36913885)

[Figure 2: query page prototype 7](#_Toc36913886)

[Figure 3: top 10 page prototype 8](#_Toc36913887)

[Figure 4: the query page 11](#_Toc36913888)

[Figure 5: searching by movie Title 11](#_Toc36913889)

[Figure 6: searching by Title and Year From 12](#_Toc36913890)

[Figure 7: searching by Title and Year To 12](#_Toc36913891)

[Figure 8: searching by Title, Year From and Year To 13](#_Toc36913892)

[Figure 9: searching by Title and date ranges 13](#_Toc36913893)

[Figure 10: including Genre values in the search 14](#_Toc36913894)

[Figure 11: searching using all constraints 14](#_Toc36913895)

[Figure 12: The Top 10 page 15](#_Toc36913896)

[Figure 13:search page Internet Explorer 16](#_Toc36913897)

[Figure 14: top 10 page Internet Explorer 16](#_Toc36913898)

[Figure 15: search page Chrome 17](#_Toc36913899)

[Figure 16: top 10 page Chrome 17](#_Toc36913900)

# Table of Tables

[Table 1: Studio field normalisation summary 5](#_Toc36913901)

[Table 2: system files 10](#_Toc36913902)

# Project Background

SMT Movie Rental requires an application to allow their clients to search their movie database.

The company currently maintains an Excel spreadsheet listing the available movies and their current rental status. The current system is difficult to maintain and is not available for searching by customers.

The company requires an online system that will allow its customers to search the database of available movies online.

The project requires the development of two sub-systems. The first is the search system. This will consist of a form allowing the user to search the database by Title, Genre, Rating and year or any combination thereof. The second sub-system will display a chart showing the ten movies most frequently searched for.

# Project Development Environment

The project will be developed using public domain software.

The project requires the development of a database to store details of the available movies. MySQL server was selected as it provides a robust and scalable solution. MySQL is used extensively for web site development and has an active developer and support community.

The users will interact with the system online. This will require development of web pages and the use of a web server. Apache was selected for the web server. Apache is a robust and well supported server; it is used to run approximately 40% of web services used worldwide.

The user interface will be developed using HTML forms with PHP scripting providing the interface between the web form and the database. Extensive validation will be undertaken on the server to protect the system against malicious usage.

Initial data validation of the supplied data will be undertaken using Excel. The initial page design and layout will be undertaken using an online tool. Development of the code will be undertaken using Visual Studio Code.

# Database Creation

## Introduction

The data for the Movies database was supplied as an Excel document. The document contains one header record and 2298 data records. Each record has 11 fields.

## Data Review

Each record has a unique ID value stored in the “ID” field. These values were checked, and all were found to be unique.

The title of each film is stored in the “Title” field. The stored values were inspected and numerous occurrences of “special characters” such as commas, apostrophes, brackets and hashes were noted. The presence of these characters was addressed by wrapping the title in quotes when creating the SQL commands to load the data. Embedded apostrophes were escaped in the SQL to stop them being interpreted as character delimitators.

The lengths of the titles were checked and found to range between two characters and eight-two characters.

## Data Normalisation

It was noted that most of the columns contained only a small number of unique values. The database was normalised by creating a separate library table to hold the values from these fields. Foreign key constraints were used to link the library tables back to the data table.

Two fields were not normalised as there were no efficiencies to be gained. The “RecRetPrice” field holds a decimal value ranging from 7.99 to 276.45 with 77 unique values. The “year” field holds a four-digit year code.

On inspection it was noted that in many records the Version field stores multiple version codes separated by commas. A separate table was constructed to store the version data allowing multiple version records to be stored for each movie. The table links to the movie table using the movie ID value and itself links to a table holding the version codes.

Library tables were created for the following fields:

* Studio
* Status
* Sound
* Version
* Rating
* Genre
* Aspect

For each of the fields to be normalised the following process was undertaken

1. Make a list of unique values
2. Sort the values into ascending order
3. Count the number of occurrences of each value
4. Check the total number of occurrences
5. Assign a numeric ID code to each value

A table summarising the results for the Studio field is presented below:



Table 1: Studio field normalisation summary

There are eight unique studio values, the total number of occurrences is correct.

## Data Importation

The original data was modified within Excel to create the data to be loaded into MYSQL.

Within each movie record the values of the fields to be normalised were replaced with the appropriate index value from the library table. The Title field was wrapped in single quotes and any special characters escaped. The individual fields were combined into a comma delimited list and enclosed in brackets.

The version data table was created by making a record for each entry in the version field. The number of records created for each movie ranged between one to three. Each record consisted of the movie index number and the index value from the version library table.

A formatted set of records was created for each of the library tables.

An SQL statement was written to create each table, followed by another SQL statement to import the data. The reformatted records were pasted from Excel into the file of SQL commands.

SQL statements were added to create the required index fields and foreign key constraints were written.

## Additional Tables and Views

An additional table was created to store the number of times a move was searched for. This table only stores records for movies that have a search record and consists of two fields one with the movie ID code and one with the search count.

To simplify the interaction with the database three views were created.

1. A view was created for display in the search page of the application. The view links the movie table with the library tables and includes the search count. The table facilitates the interaction of the search page with the database.
2. A view was created to list the 10 most frequently searched for movies
3. A view was created to add records to the search count table. The table only holds records for movies that have been searched for, if a movie has not previously been searched for it must be added to the table before the count can be incremented.

## Database Creation

SQL statements were run to create the database tables and views, insert the data and create the indexes.

A schema diagram was created to show the structure of the database:

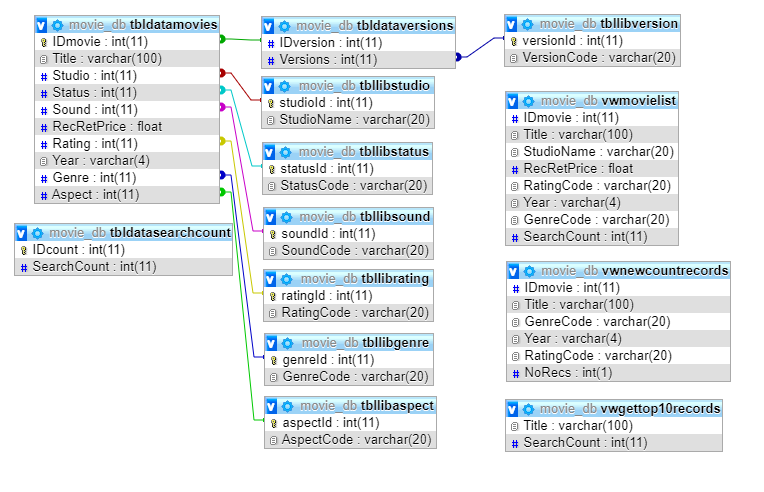


Figure 1: normalised database schema

# Project Design

## Web Page Design

The system will use two web pages, one to query the database and one to present the top 10 chart. Each page will include a menu to allow navigation between the pages.

### The Query Page

A prototype design for the query page is presented below:

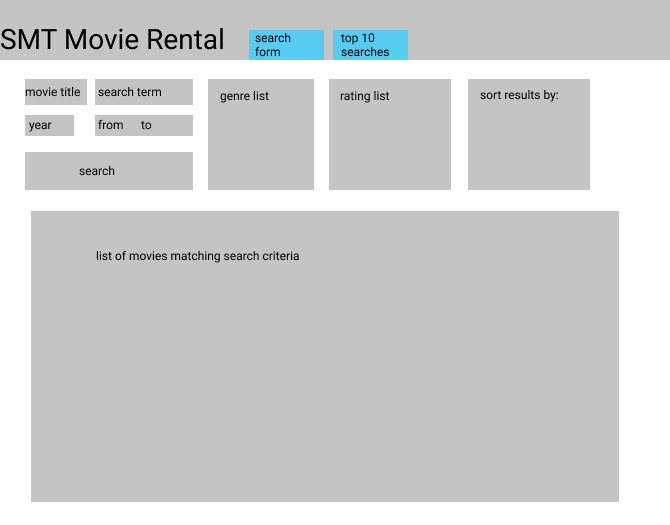


Figure 2: query page prototype

The form allows the user to search by Title, Year, Genre and Rating.

To search by title the user enters a search term in the “search term” text field. The search term is treated as a wildcard and any titles containing the search term will be returned. The text field is a free form entry element with no restrictions on content.

The user can search for movies released during and after a selected year using the “From” element, before and during a selected year using the “To” element and between two years using both elements. The searches will include the selected years. The maximum and minimum values of the elements will be restricted to the maximum and minimum values stored in the database. The two form elements will use the numeric type.

The user can search by genre by selecting one or more genres from a list. The list will be populated using the library table in the database. By default, no genre element will be selected.

The user can search by rating by selecting one or more ratings from a list. The list will be populated using the library table in the database. By default, no rating element will be selected.

The user can sort results of the query by selecting the field name to sort by.

### The Top 10 Page

A prototype design for the top 10 page is presented below:

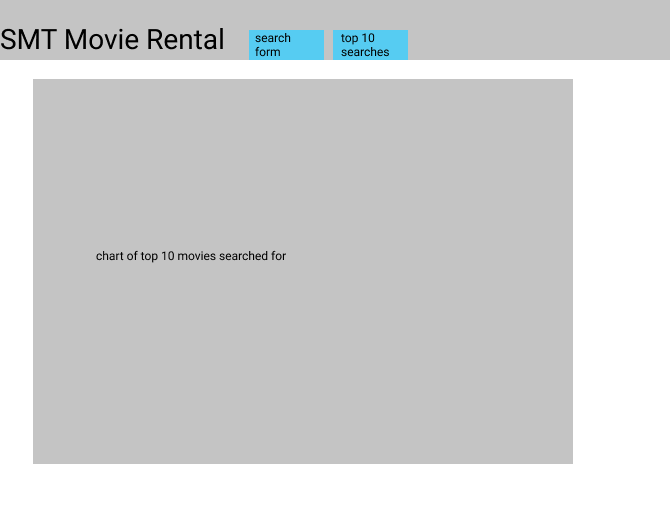


Figure 3: top 10 page prototype

The web page will display a chart showing the top 10 most searched for movies.

## Functional Design

### The Query Page

The query page will use a web form to obtain the search criteria from the user.

The form will use a set of queries to populate the form elements. The date range elements will use a numeric field type with the minimum and maximum values set to the minimum and maximum values from the database. A list of valid genre and ratings will be obtained from the database and used to populate the appropriate list elements.

The form will post the values to a PHP script which will validate the values before creating an SQL command to run the query.

The query will be constructed using the following rules:

1. The Title value will be treated as a wildcard and be submitted using the formulation “where Title LIKE ‘%search\_term%’
2. There are four possible scenarios for date searching
   1. From date only: search using Year >= from\_value
   2. To date only: search using Year <= to\_value
   3. Both the From and To dates entered:
      1. search using Year BETWEEN from\_value and to\_value
   4. Both the From and To dates entered, but From > To:
      1. search using Year <= from\_value OR Year >= to\_value
3. The selected genre values will be combined into a comma separated list and queried using the IN operator: i.e. WHERE Genre IN (‘genre1’,’genre2’,…)
4. The selected rating values will be combined into a comma separated list and queried using the IN operator: i.e. WHERE Rating IN (‘rating1’,’rating2’,…)
5. The individual elements of the query will be ANDed together.
6. The results from the query will be sorted using the field name selected in the Sort By list. The Title field will be used as the default value.

The SQL command will be submitted to MySQL via a PDO connection.

Any results from the database will be used to create an HTML table that will be displayed below the search criteria. The search criteria will be used to update the “searched” counter through the Movie display view.

If no results are returned by the query a message to that effect will be displayed.

The page has been designed to show the query results and the active search criteria on the same page. This approach allows the user to easily compare the results to the selected criteria and allows the search to be progressively refined. To facilitate this the values entered into the form are maintained across repeated searches. If there are values selected from the Genre or Rating lists these values will be displayed at the top of the list after the search has been run.

### The Top 10 Page

When opened the top 10 page will submit a query to the database to extract the ten most frequently searched for titles and the number of times each has been searched for.

The PHP GD graphics package will be used to create a bar chart showing the number of times each movie has been searched for.

The page will dynamically update each time it is opened.

## Risk Analysis

### The Query Page

The query page uses the POST method to submit form values to the server. This approach includes the form values within the HTTP request making it harder to modify the values. On the server side all values submitted are checked for malicious content before processing.

* The “htmlspecialchars()” PHP function is used to neutralise any HTML code injected into the form.
* Any backslashes are removed using the “stripslashes()” PHP function.

There are six data entry elements within the form. Of these elements only one, the Title search, allows free form input. Text entered via this control is checked for malicious content on the server side.

The two date search elements use the number type and will only accept numeric values.

The final three elements are list controls presenting a list of values from which one or more can be selected.

### The Top 10 Page

The top 10 chart page does not submit anything to the server and does not present a security risk.

### Conclusions

The combination of the controls within the form and server-side validation limits potential avenues for malicious behaviour to occur.

# Testing

## Introduction

There are application uses a total of twelve files. Two files create the web pages displayed by the server, six files of PHP scripting and four files used for formatting controls.

A list of the files used, and their function is presented below:

|  |  |
| --- | --- |
| File Name | Description |
| bootstrap.min.css  bootstrap.min.js  demo.css  jquery.min.js | Bootstrap and CSS formatting files |
| SearchMovies.php | Movie search form and record display |
| Top10.php | Page to display the top 10 chart |
| connect.pdo.php | Creates the PDO connection into the database |
| movie\_list\_genre\_scr.php | Populate the genre select list |
| movie\_list\_rating\_scr.php | Populate the rating select list |
| movie\_year\_limits\_scr.php | Set the maximum and minimum year values in the Year filter controls |
| movie\_list\_scr.php | Build the SQL command to search for movies and display the resulting records in a table |
| movie\_top\_10\_chart\_scr.php | Get the top 10 data and create a chart to display |

Table 2: system files

## Static Analysis

The PHP\_CodeSniffer tool was used to validate the PHP code against the PEAR standard.

The six PHP script files were tested, and any issues identified addressed either by manually editing the file or by using the PHP\_CodeBeautifier utility.

## Functional Testing

### The Query Page

The following screen dump shows the query page when first loaded, the Title field is preselected in the Sort By list:

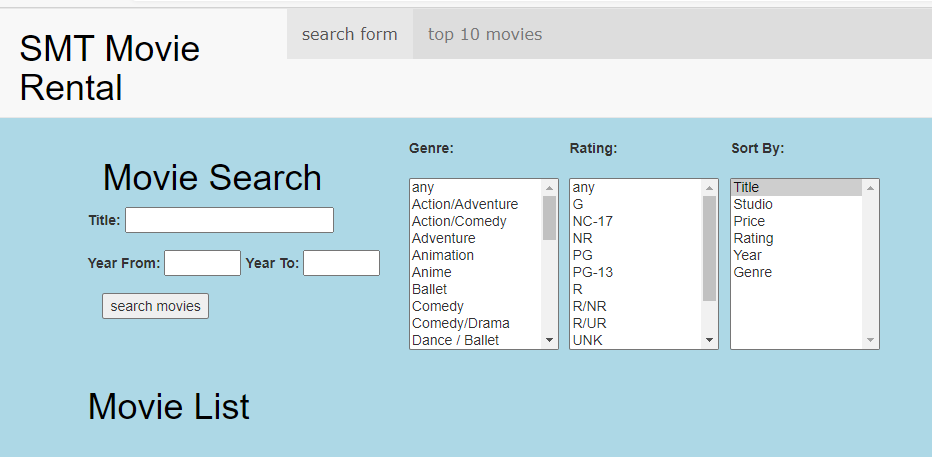


Figure 4: the query page

A search was conducted using the term “sing” in the title box, 18 records were returned all the titles containing “sing” within them:

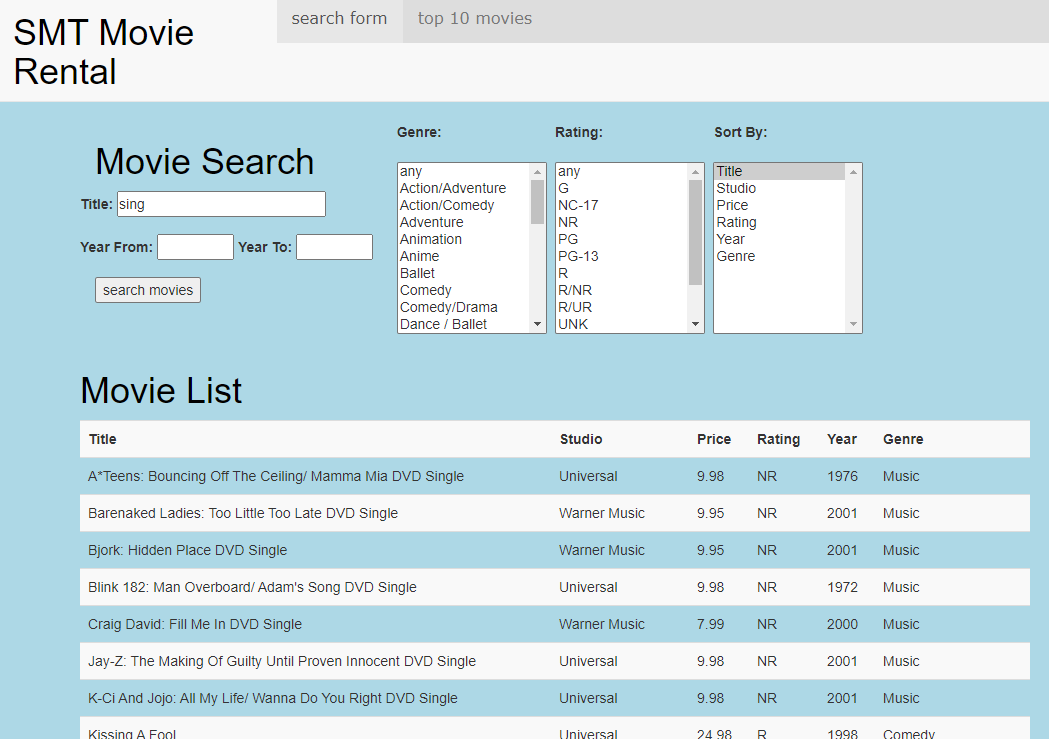


Figure 5: searching by movie Title

A Year From value of 1970 was set and the results ordered by Year, only movies with a Year greater than 1970 are included in the report:

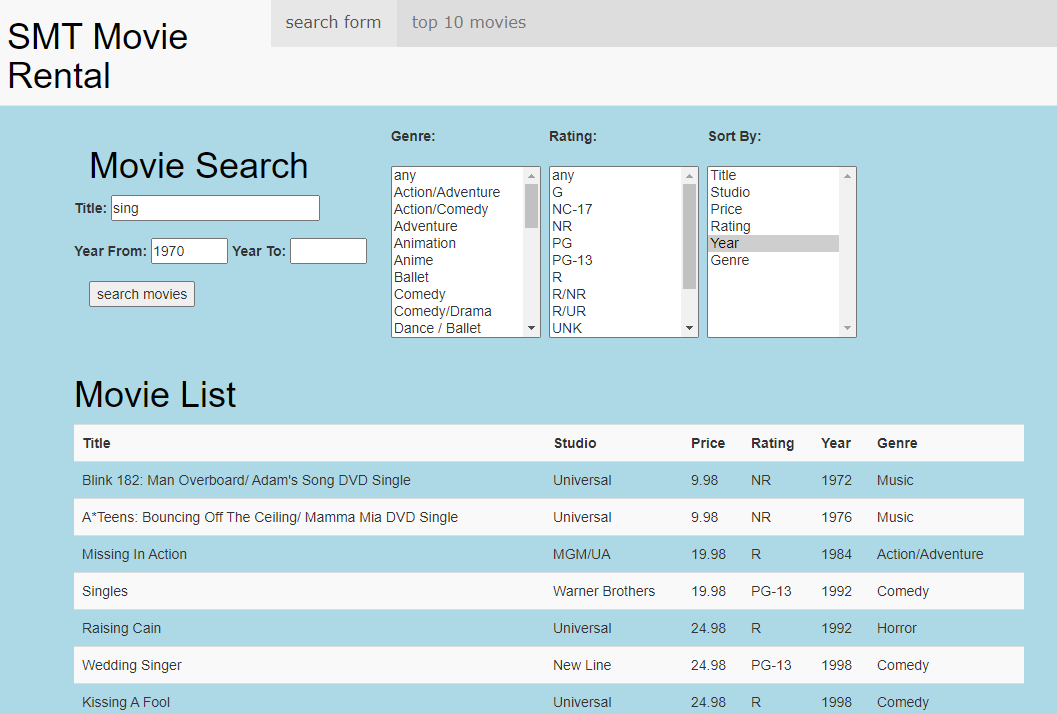


Figure 6: searching by Title and Year From

A Year To value of 1990 was set and the results ordered by Year, only movies with a Year less than 1990 are included in the report:

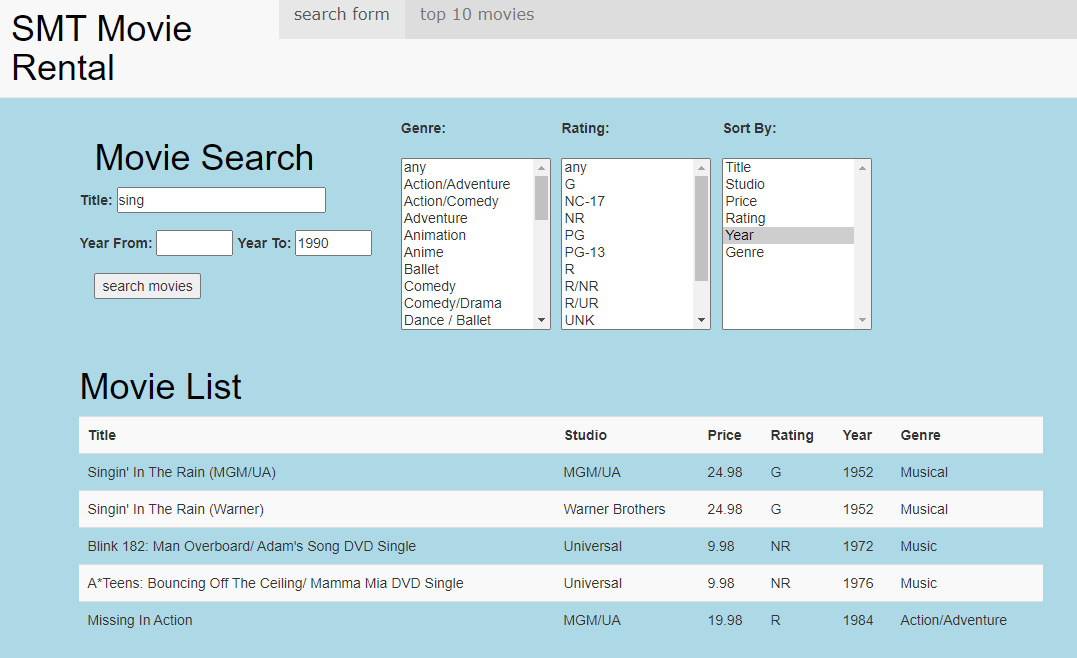


Figure 7: searching by Title and Year To

A Year From value of 1970 and a Year To value of 1990 was set and the results ordered by Year, only movies with a Year between 1970 and 1990 are included in the report:

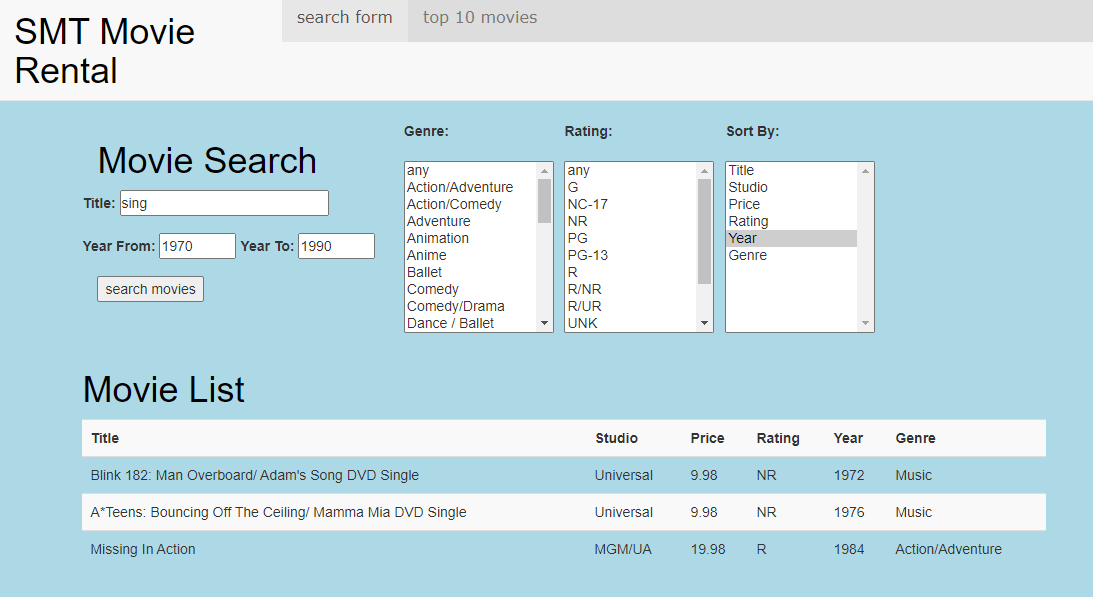


Figure 8: searching by Title, Year From and Year To

The year values were swapped to find movies outside of the selected range:

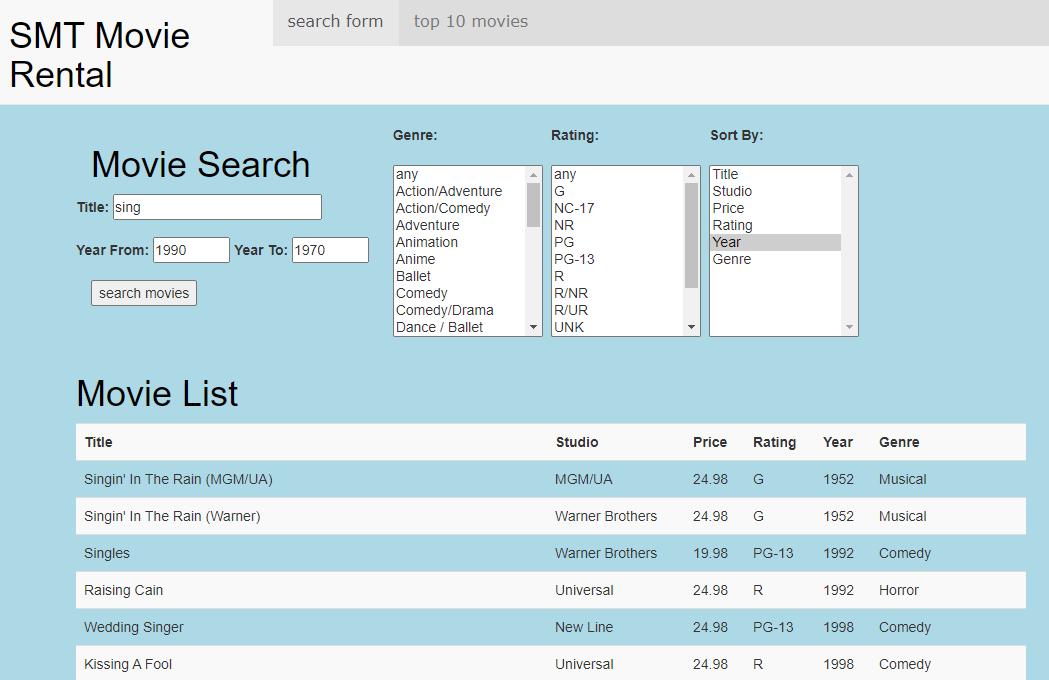


Figure 9: searching by Title and date ranges

The Music, Musical and Comedy Genres were added to the search criteria:

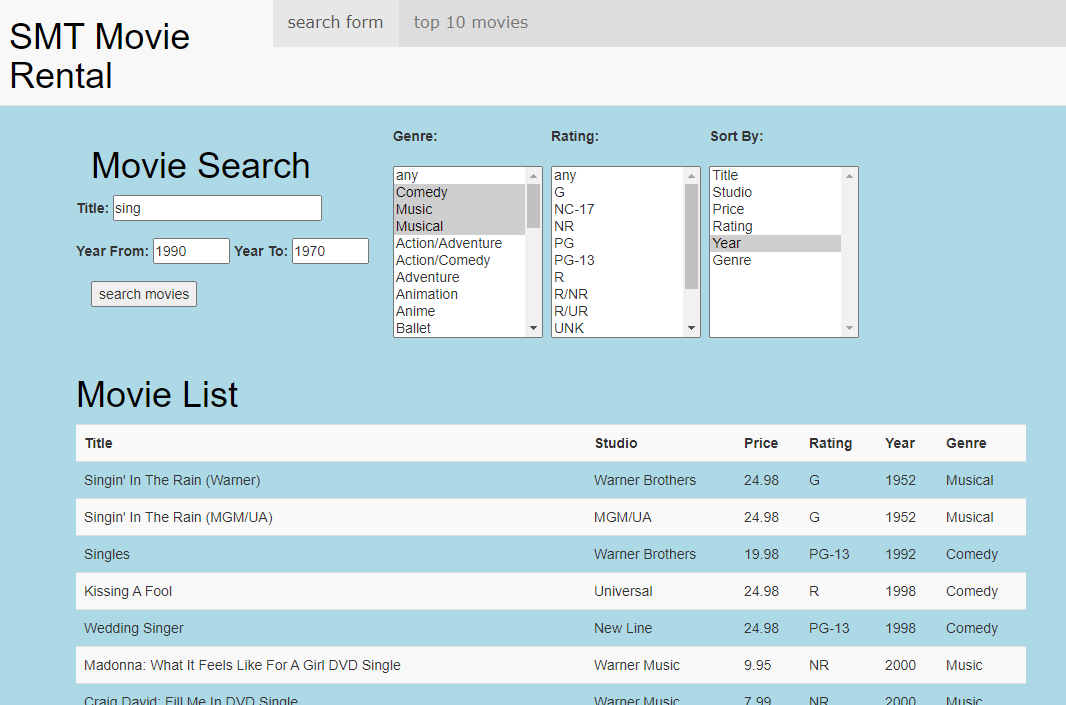


Figure 10: including Genre values in the search

The rating was constrained to G, PG and PG-13 and sorted by Rating:

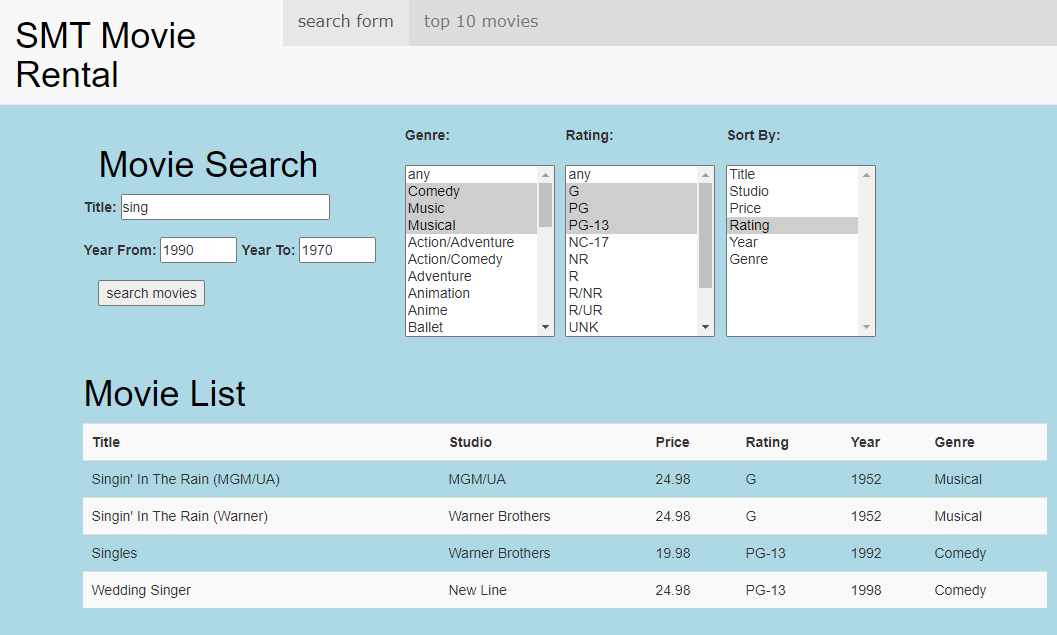


Figure 11: searching using all constraints

All the search elements were tested individually and in combination, the results have not been included for brevity.

### The Top 10 Page

A screen capture of the Top 10 page is presented below:

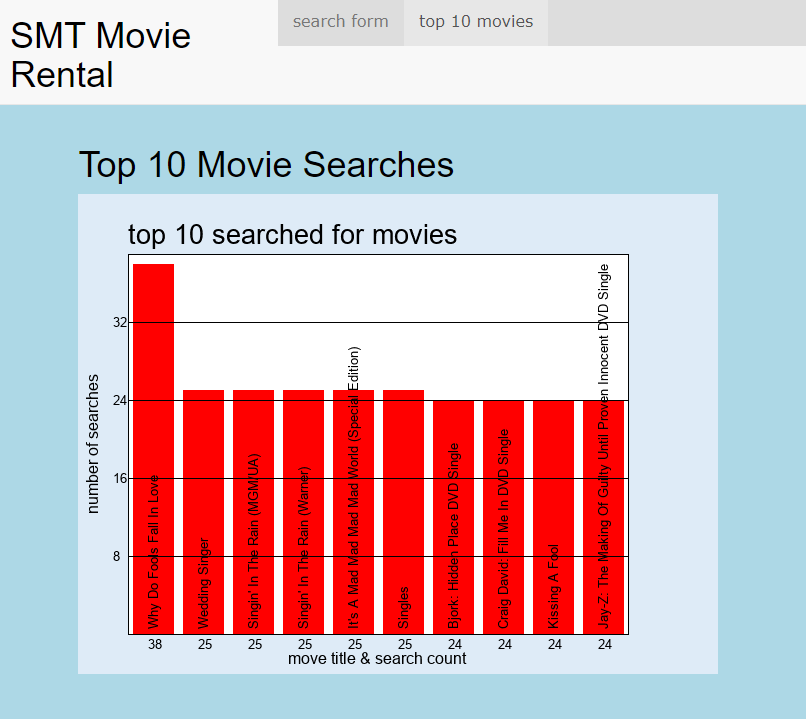


Figure 12: The Top 10 page

## Compatibility Testing

All the functional testing was undertaken using Microsoft Edge. Compatibility across browsers was tested by opening the pages in Internet Explorer and Chrome:

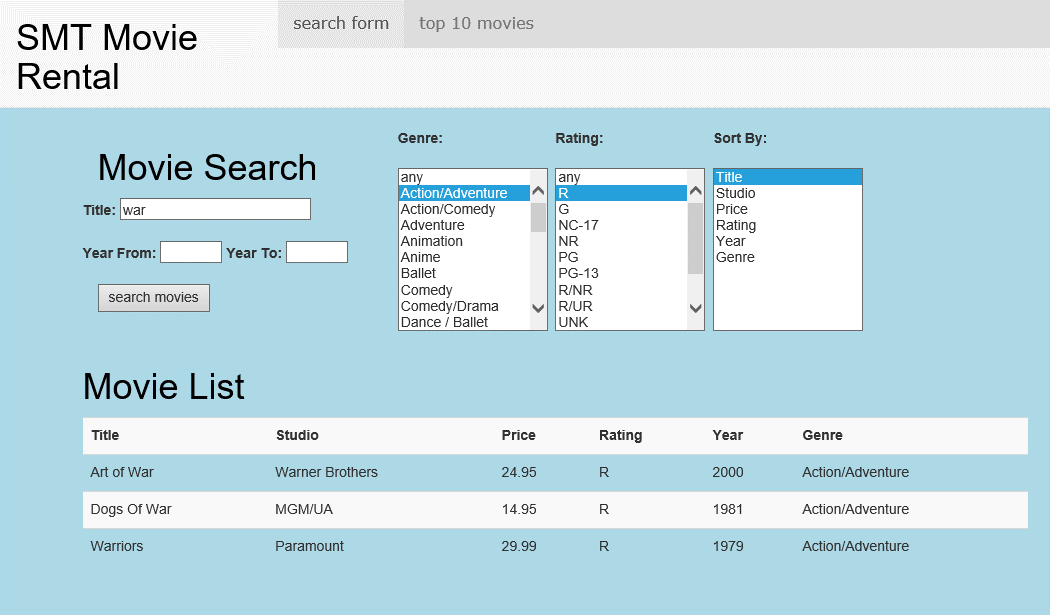


Figure 13:search page Internet Explorer

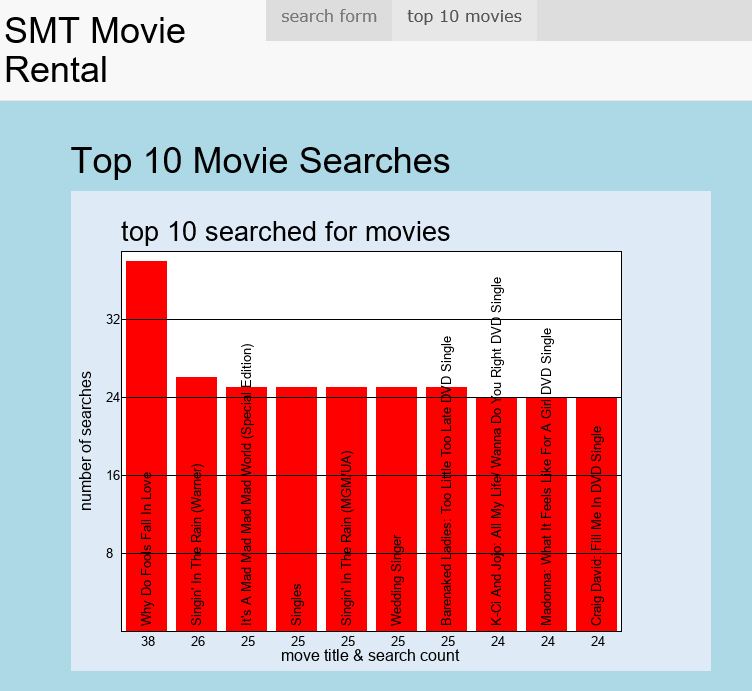


Figure 14: top 10 page Internet Explorer

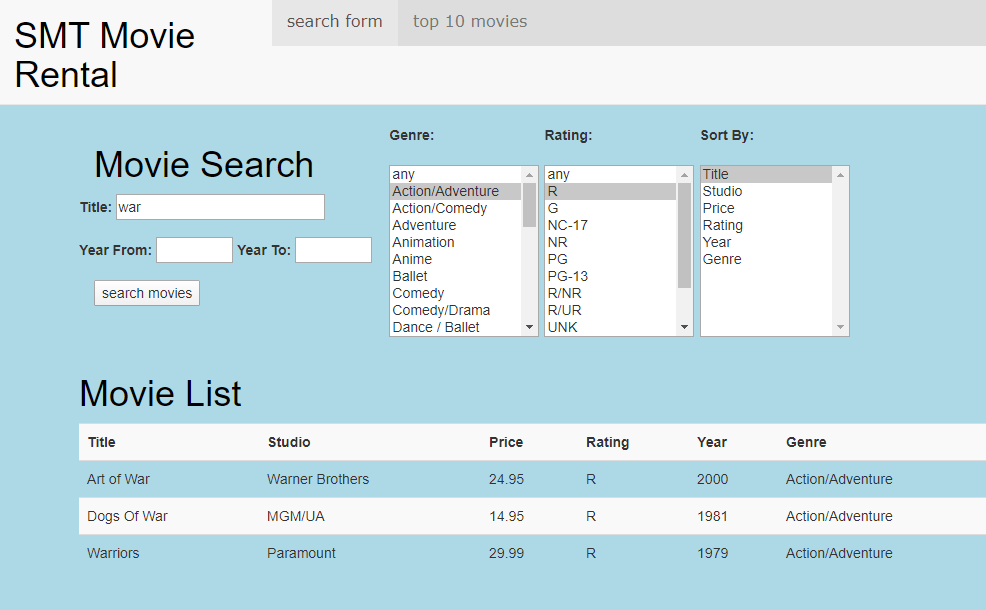


Figure 15: search page Chrome

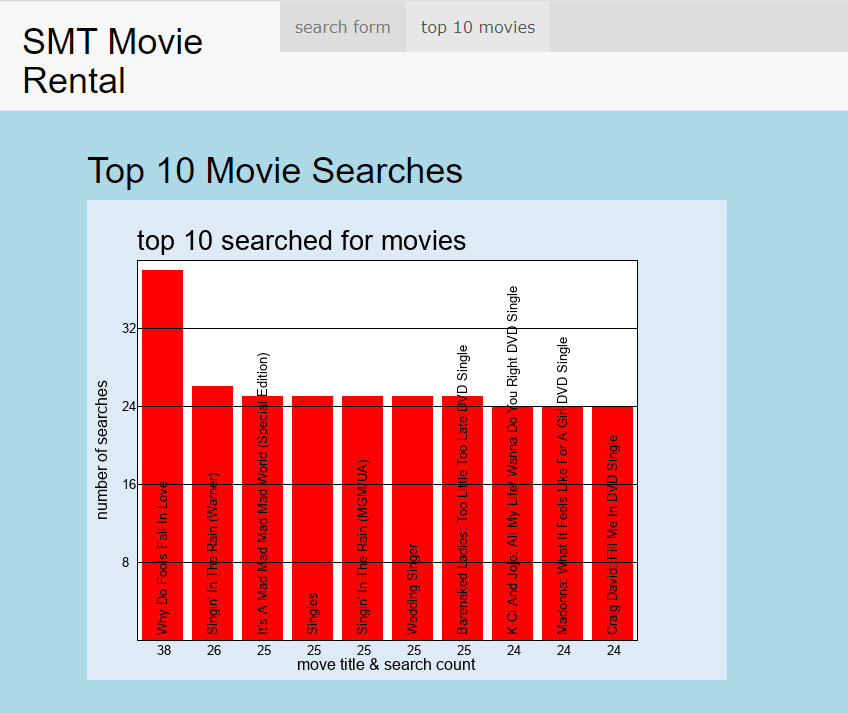


Figure 16: top 10 page Chrome