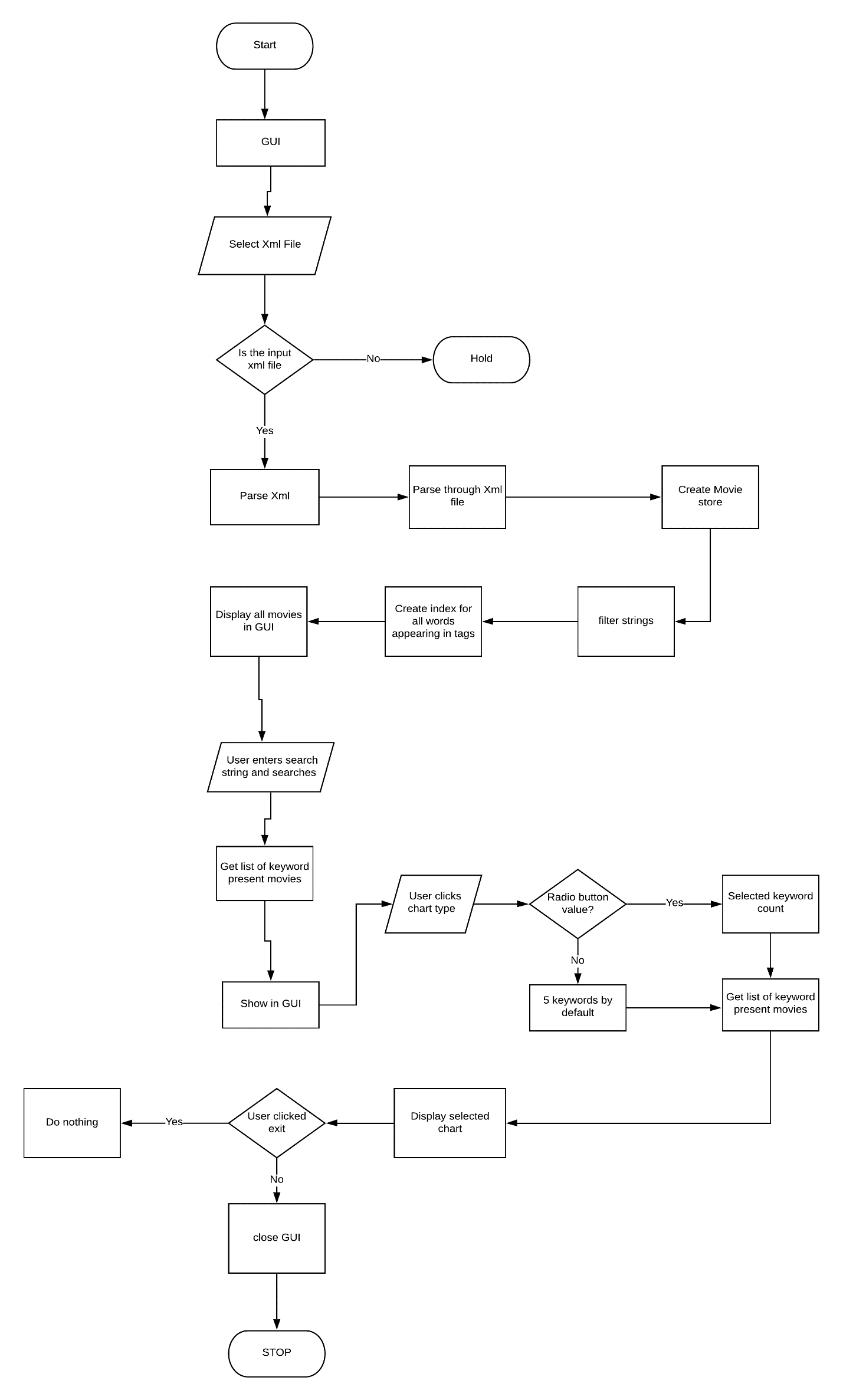
Done By:

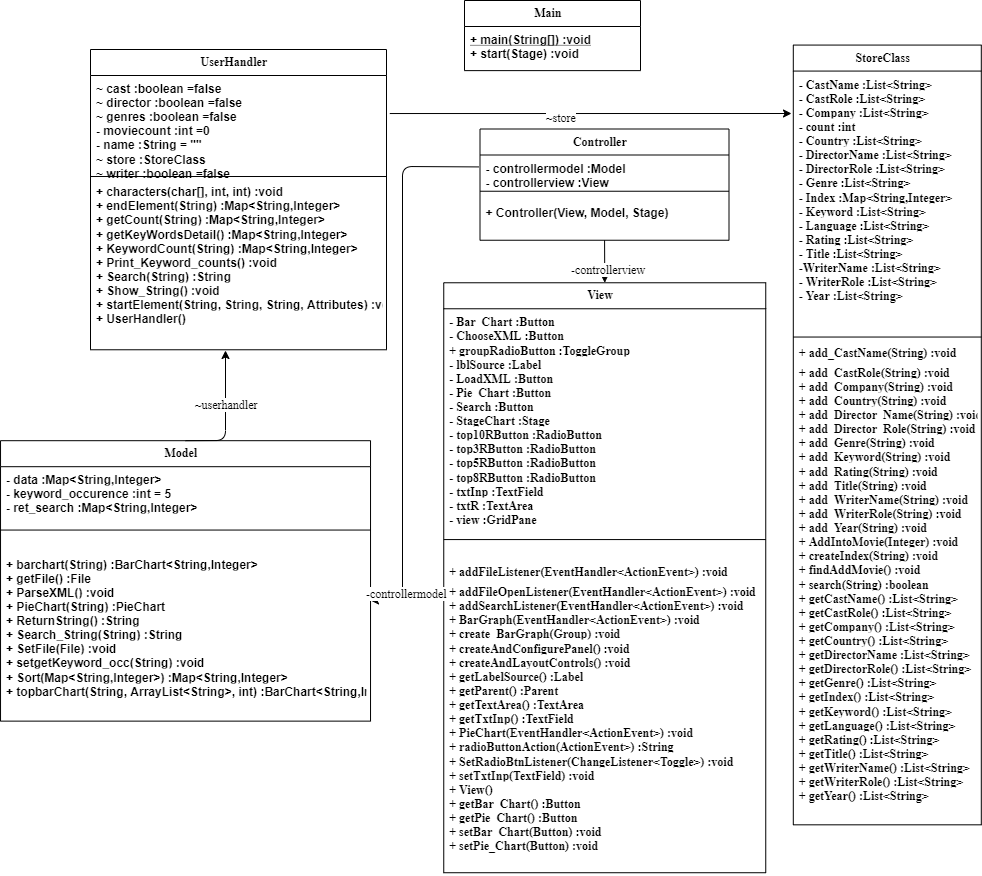
Binay Dhawa 101187399

Riwaj Shrestha 101722868

|  |  |
| --- | --- |
| A picture of a winding road and trees  XML KEYWORD SEARCH SYSTEM  Assignment 2 | Abstract  Search and Parse through IMDB xml file using an index built from java fx  BINAY DHAWA  COS80007 Advanced Java |



**FLOW CHART**

**UML Diagra****m:**

**Report**

The Application creates an index for each word while parsing to note down the xml file. As the index needs to be created only once during each session, the parsing needs to happen just once so a SAX parser is being used.

When The user enters a word. The application searches over the saved indexes’ and returns the movies found containing those keywords. While parsing, the index also counts the occurrence of the word within the file.

When the user clicks on Chart the top 10 Keyword existing within the file is returned in the form of a Chart. The user can select in between top, 10,8,5 and 3 ranking words. When the user searches for a word the search space for the top 10 Keyword is narrowed to the movie list containing the keywords.

The top 10 keyword will then return the 10 recurring keywords within the list; however, the recurring count of the keyword is taken from the global scale.

**STORE:**

The selected file is parsed using a SAX XML parser, while parsing through the keyword, the a list is being created for each xml tag, The program assumes that each element tag can occur more than once, it creates a list of Title, Year and Rating, Country, etc. The idea behind these lists is that as the xml file could contain more than one element of a kind, these elements cannot be stored as a string as the string would get overwritten by a new string after each element reappears. The List Class is easy to access and store thus a List of string containing the respective values are created. The Lists are then put into a new HashMap designed to store a list of list of Strings. The HashMap is created each time a movie tag is opened. When a movie tag is closed the HashMap is put into another HashMap that is specifically designed to store the previous HashMap. The existing lists and HashMap is then initialized as a new HashMap for the next round of Xml tags.

A HashMap of indexes is also created that keeps the value of each keyword. The keyword is put inside the HashMap only if the keyword is unique and is not existing in the array of keyword that is marked as to be filtered off.

HashMap and Lists are used instead of an array mainly because of their time and speed complexity while retrieving data and scalability for the application. The HashMap furthermore quickens the time for search.

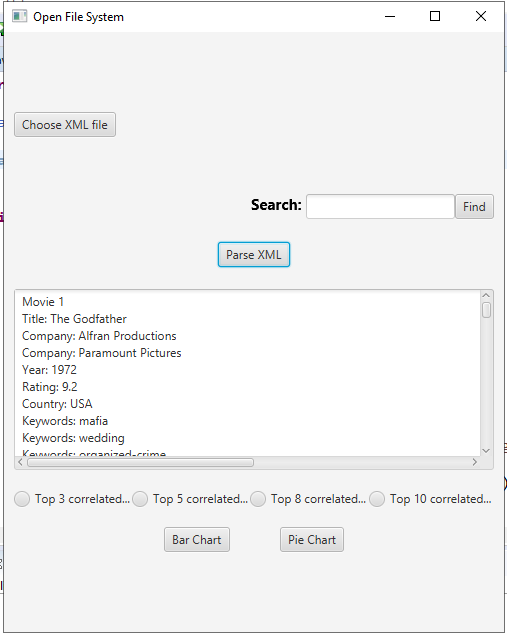
**MVC:**

The Application is created in the format of Model View and Controller. Each element can exist on its own and feature addition will not disintegrate the working features of the current application. Model view and controller are all bound together at the main method. There are event handlers implemented and bound on controller for model and view. The Model stores all the method or the brain of the application whereas the view is the GUI itself. Controller class is where interaction between GUI and model does take place. The parsing, creating index happens while the user parses an index file. This interaction is happening between the model and the SAX parser. There is no direct communication happening in between the view and the model.

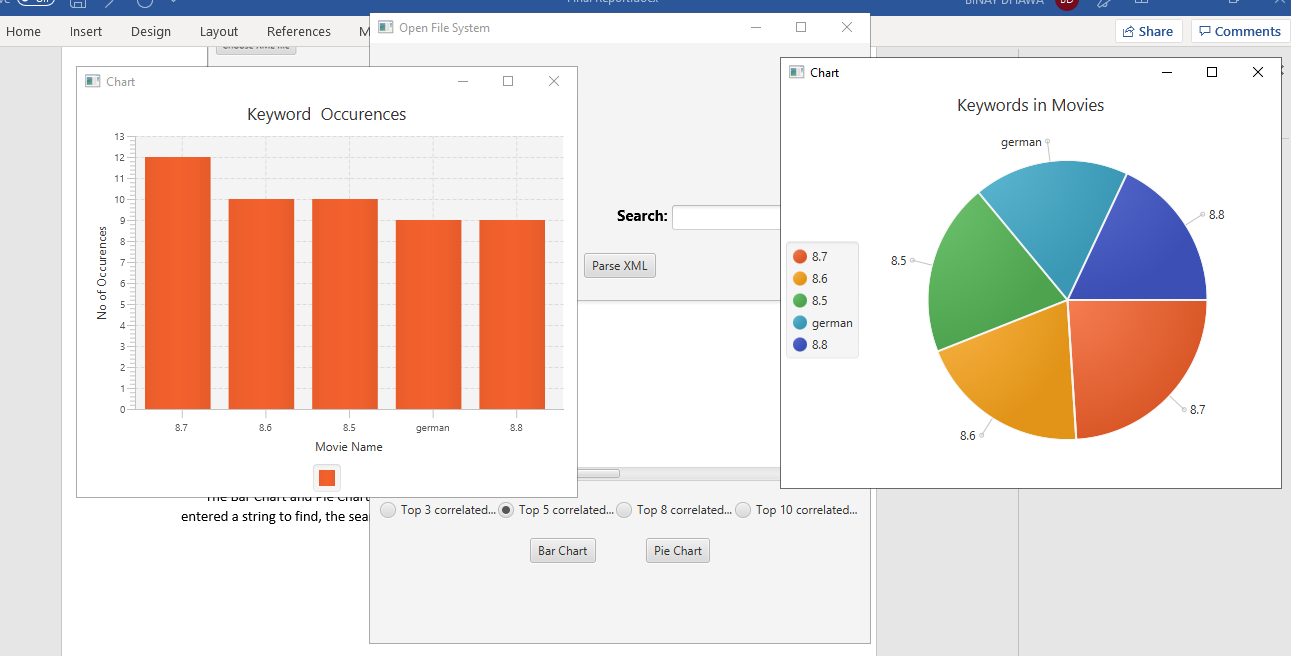
GUI for the Application

The GUI for the application is shown as above

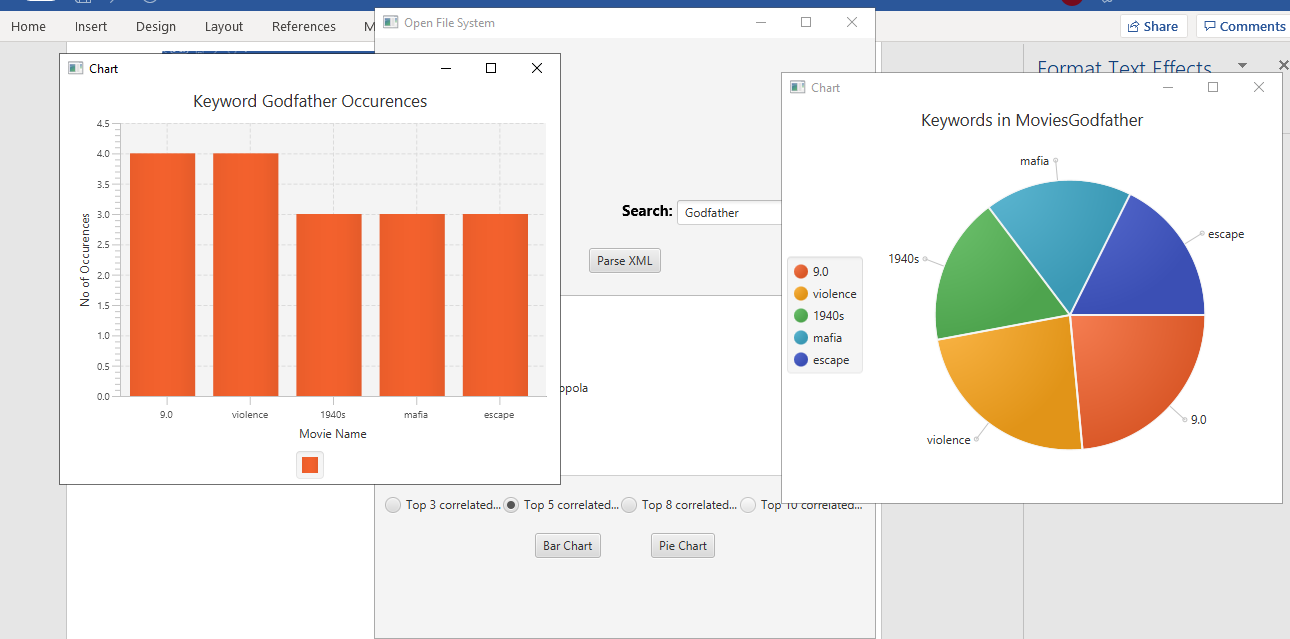
The Choose **XML button** is used to load the file onto view. The **Parse XML** button is used to parse the xml file to parse xml file to index of our program. Subsequently, it also displays the parsed file in form of text.



The **Bar Chart** **and Pie Chart** displays both Bar and Pie Chart respectively. When the user hasn’t entered a string to find, the search space is widened to the whole index.

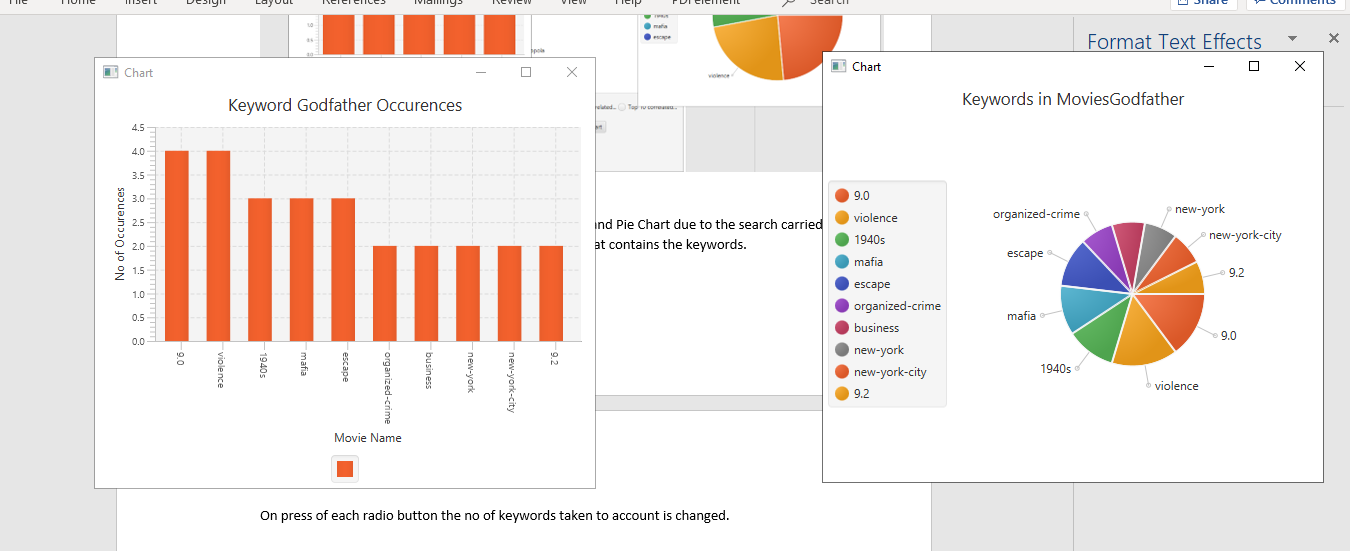


When the user **searches** for a keyword the space is narrowed to the list containing the user entered string.

****

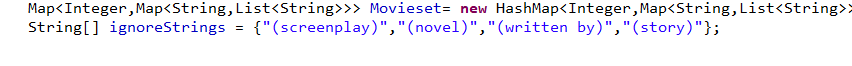
We can see that the keywords have changed in the Bar and Pie Chart due to the search carried out by the user. The Text Area also displays the movie list that contains the keywords.

On press of each radio button the no of keywords taken to account is changed.



The above diagram shows results for top 10 recurring keywords for the movies returned from Godfather keyword.

It needs to be noted that there are a few keywords that are filtered out programmatically due to high appearance and their irrelevancy in the search.



The keywords appearing on the above picture are not considered by the system.