**MOVIE DATABASE USING BINARY SEARCH TREE**

**TEAM MEMBERS:**

SAURABH THAKUR - 13BCE0081

VIVEK PAWAR – 13BCE0585

VAIBHAV VISHAL – 13BCE0743

**Abstract**

* The main idea of the project is to create a movie database using the ordering property of a binary search tree (abbreviated as bst). The property of the bst is that for any node n in the bst, the key of n is greater than the key of all nodes in its left subtree. Similarly, the key of n is less or equal to the key of all nodes in its right subtree.
* The movies along with their details such as year of release, rating’s received are stored in the bst.
* Cast and directors is just an additional data which can be accessed by one of the features in the program called advanced search.
* The binary search tree not only contains above information but also a key
* No two movies in the database have the same id which ensures the data validation.
* For example the movies are stored in the following pattern:

**Aim :**

To create a movie database which consists a list of movies along with their details using the non-linear data structure **binary search tree (bst).**

**Objective :**

The objective of the proposed idea is to use the non-linear data structure (binary search tree) for performing operations of search, insertion, deletion and display of movie data on request as a result to understand the implementation of the binary search tree of achieving the same.

**Applicability :**

The movie database acts as a small storage of information related to movies. It allows to access information efficiently and at a faster rate. The database can be accessed in an offline mode (**under no internet access**).

However, **updating of any data requires internet access**.

The proposed idea can be applied in a number of other ways

By just changing the database, wherein the whole functionality remains the same.

Ex. The vit faculty database is a good example of the type which includes information about the faculties with their contact and cabin number.

**Introduction**

* A movie database stores data related to movies like the movie name, year of release, rating’s recieved etc.
* A binary search tree consists of nodes which are capable of holding information. The nodes can be differentiated on the basis of a key value.
* A bst can be used to store movie related information where every node of the tree holds data of a particular movie.
* In the current project implementation key value is the name of the movie.
* If **collisions** occur the unique **imdb id** is used to get the order clear which not only serves as collision handler but also give access to all the movie database online at imdb
* Various operations can be performed on a bst.
* These include
* Insertion of a movie in the database
* Deletion of a movie from the database
* Searching for a movie in the database
* Traversal implementation (inorder, preorder and postorder)
* The operation is obtained as input from the user and the corresponding task is performed.
* At any point of time, the bst order is maintained in the database.
* The project code has been developed using the programming language **‘python’** and the **interface** is also been developed using the same software.

**Existing method and drawbacks**

* The developed movie database is a mini version of imdb (internet movie database).
* The **internet movie database** (abbreviated **imdb**) is an online database of information related to films, television program, and video games, taking in actors, production crew, fictional characters, biographies, plot summaries, and trivia.
* To access this database internet connection is required. This causes a limitation in accessing information from this database. Despite being such a vast database it does not function without internet access.
* **Existing method** may be to store the movies in a list or a dictionary which are inbuilt data structures in python ,but the searching operation would lead to worst case scenario of the order of n.
* This **huge searching time** may limit the access and usage to the database and hence **defeat the purpose of accessing a movie database** .

**Proposed method and advantages**

* The proposed method is of creating a database using the renounced data structure called Binary search tree.
* Clearly Binary search tree compared to list or dictionary database reduces the **average case** scenario of order of ‘n’ to merely to order of **O(log(n)).**
* It also works connected with the network and the largest movie database existing ie. **IMDB**
* But which could work without the internet access (which we call an offline mode).
* For update of information itself the user is required to go online. This method works similar to the vit faculty database wherein student’s can easily locate required faculties cabin no. And contact info from anywhere (inside or outside the campus) without the compulsion of having an active internet connection.
* It also provides complete access to the movie information as it prints all at once (which inludes movie name, cast, year of release and rating). This saves the users time as searching for each attribute is not required.
* For operational purpose the insertion and deletion operations have been included which enables the user to include or remove any movie from the database.
* It also performs traversals just to display the properties of the binary search tree. The tree can be printed in any of the three possible traversal orders namely ‘**pre-order’, ‘inorder’ and ‘post order’**.
* However, **the main operation of the database is the searching operation** which allows the user to retrieve complete information about any movie.

**SYSTEM ARCHITECTURE**

**MAIN**

**The main python program driven by graphical user interface on Windows platform.**

**IMDB API**

**The IMDB API provides an access to all the info about a particular movie available online at the site named as www.imdb.com**

**TKINTER INTERFACE**

**The Tkinter interface allows to write a simple event driven program which responds to keyboard inputs and clicks**

**BTREE MODULE**

**The module builds a binary search tree out of 870 movie data base saved in nodes.txt**

**Class Diagram**

**1)Movie**

* Name
* IMDB ID
* Year
* RATING

**2)Tree Node**

* Key
* Payload
* Leftchild
* Rightchild
* Parent

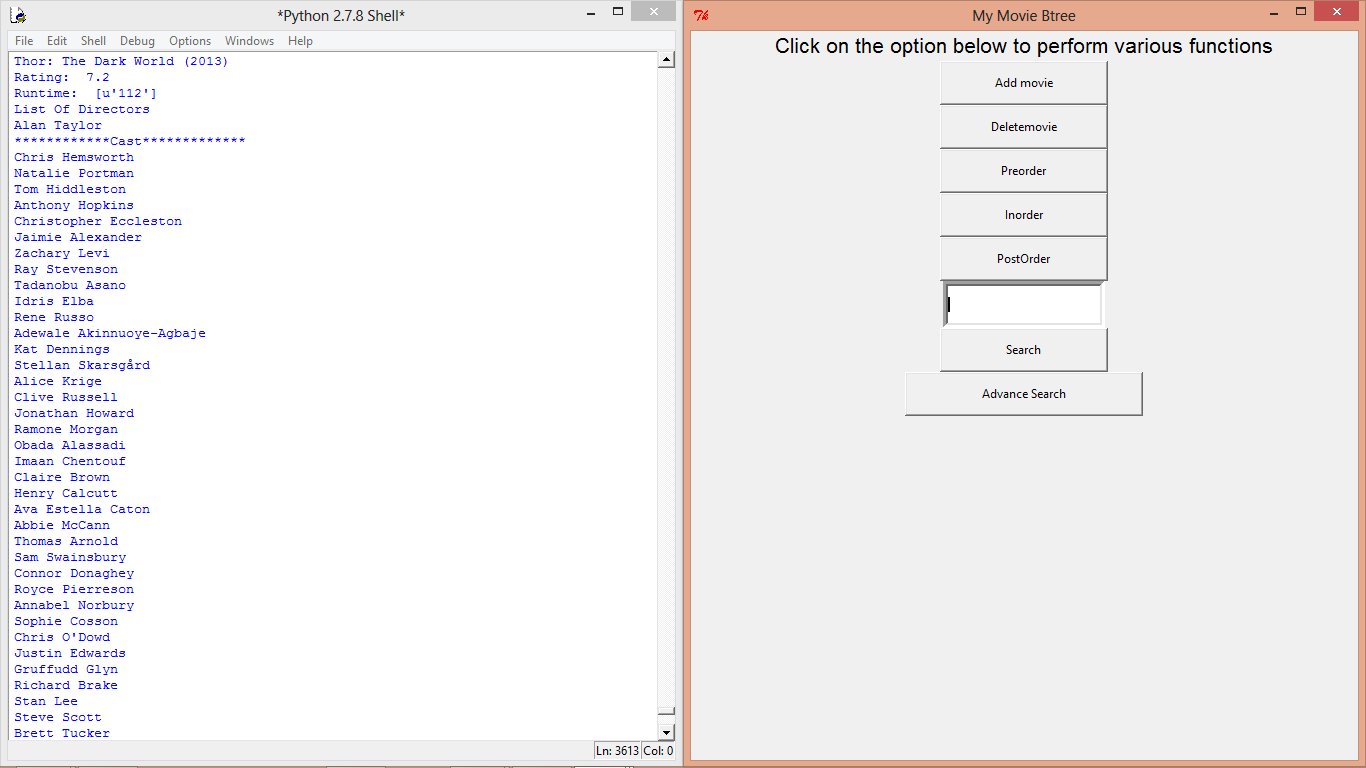
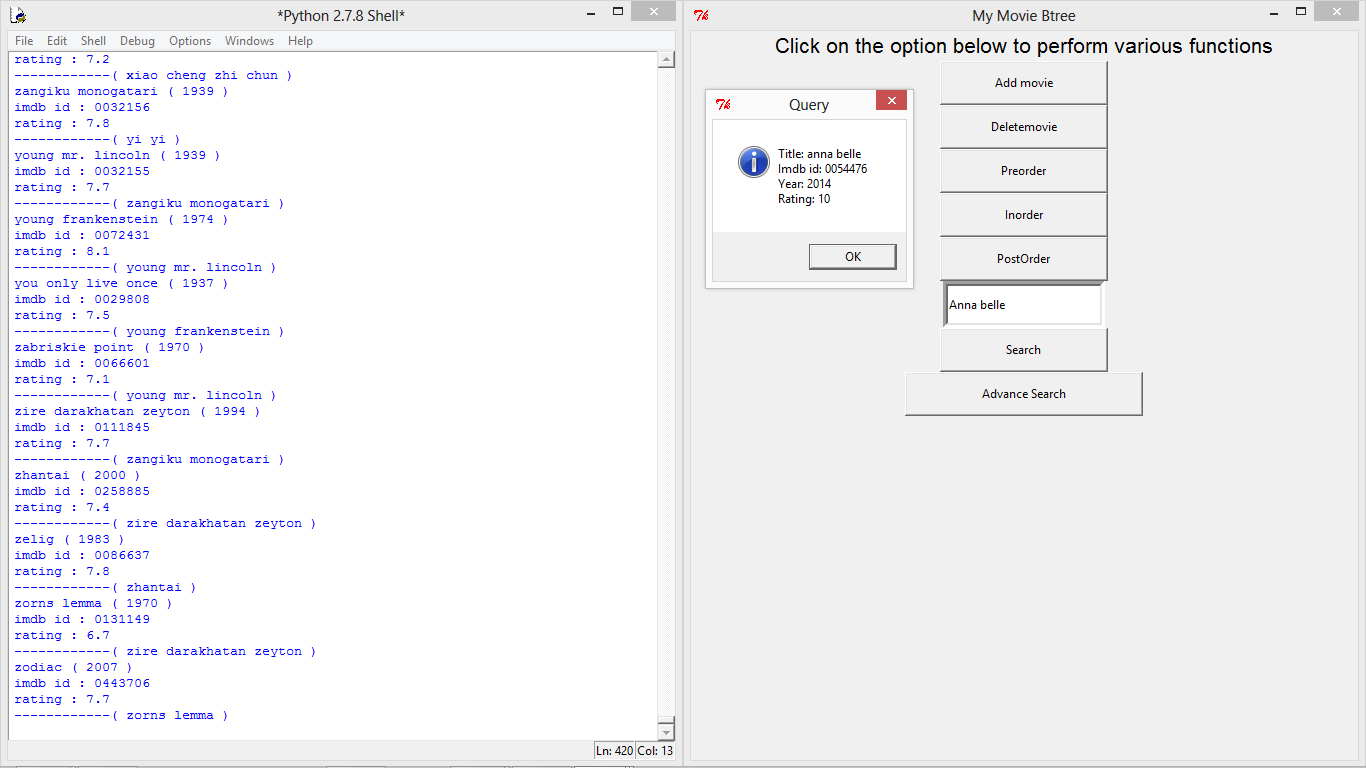
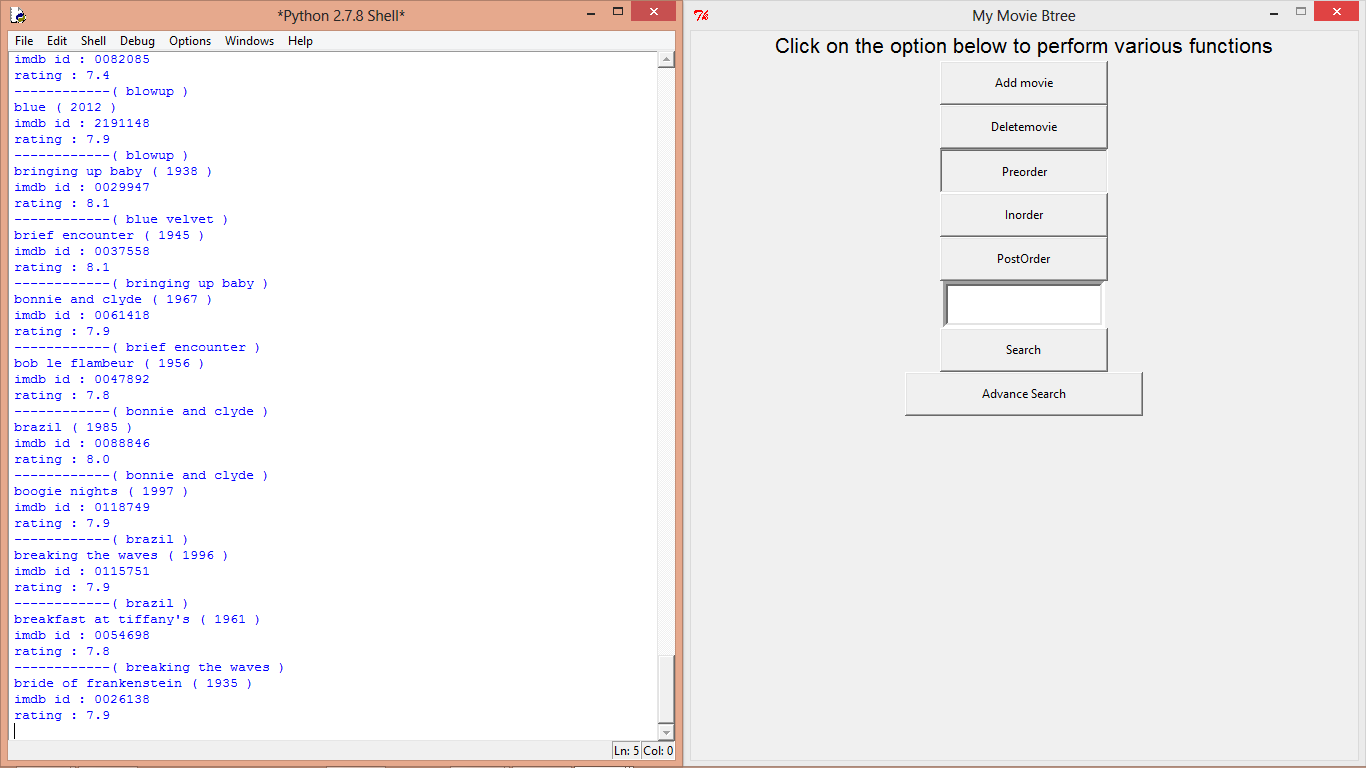
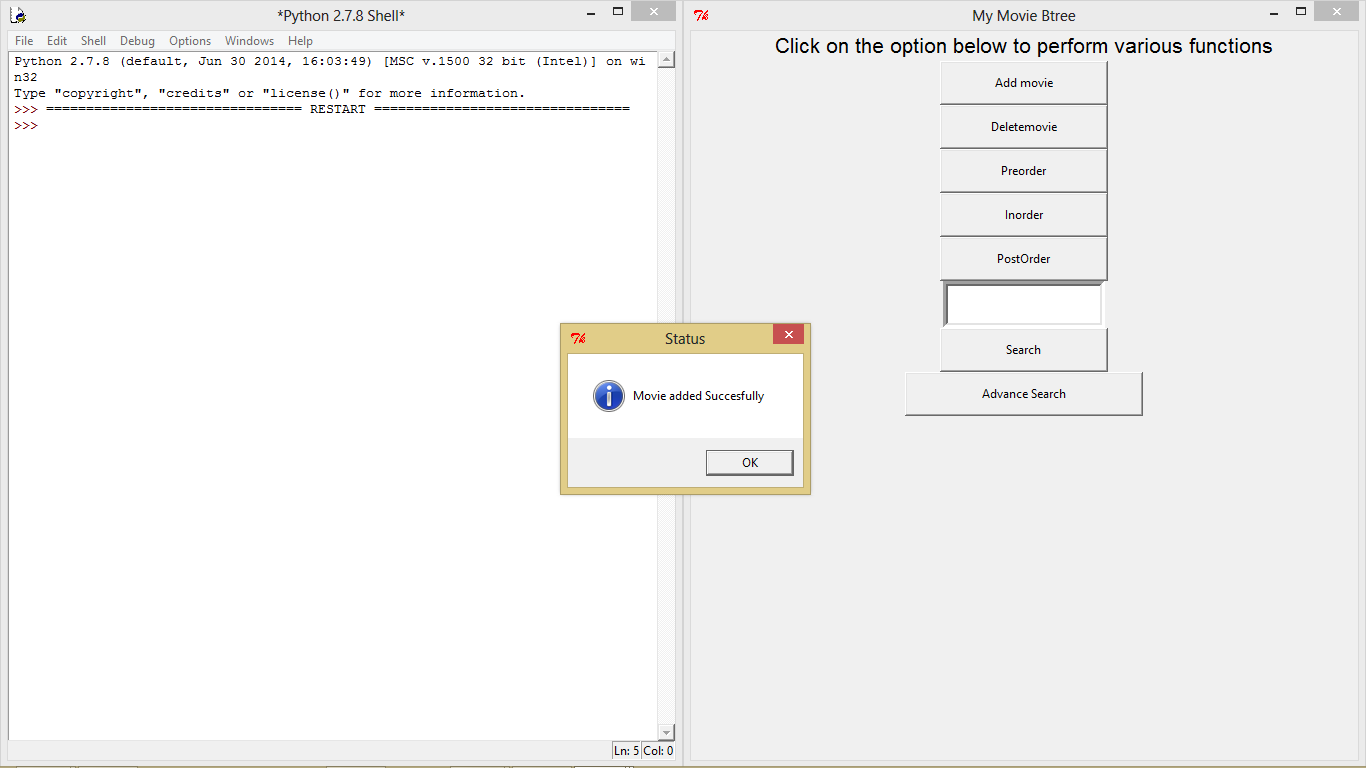
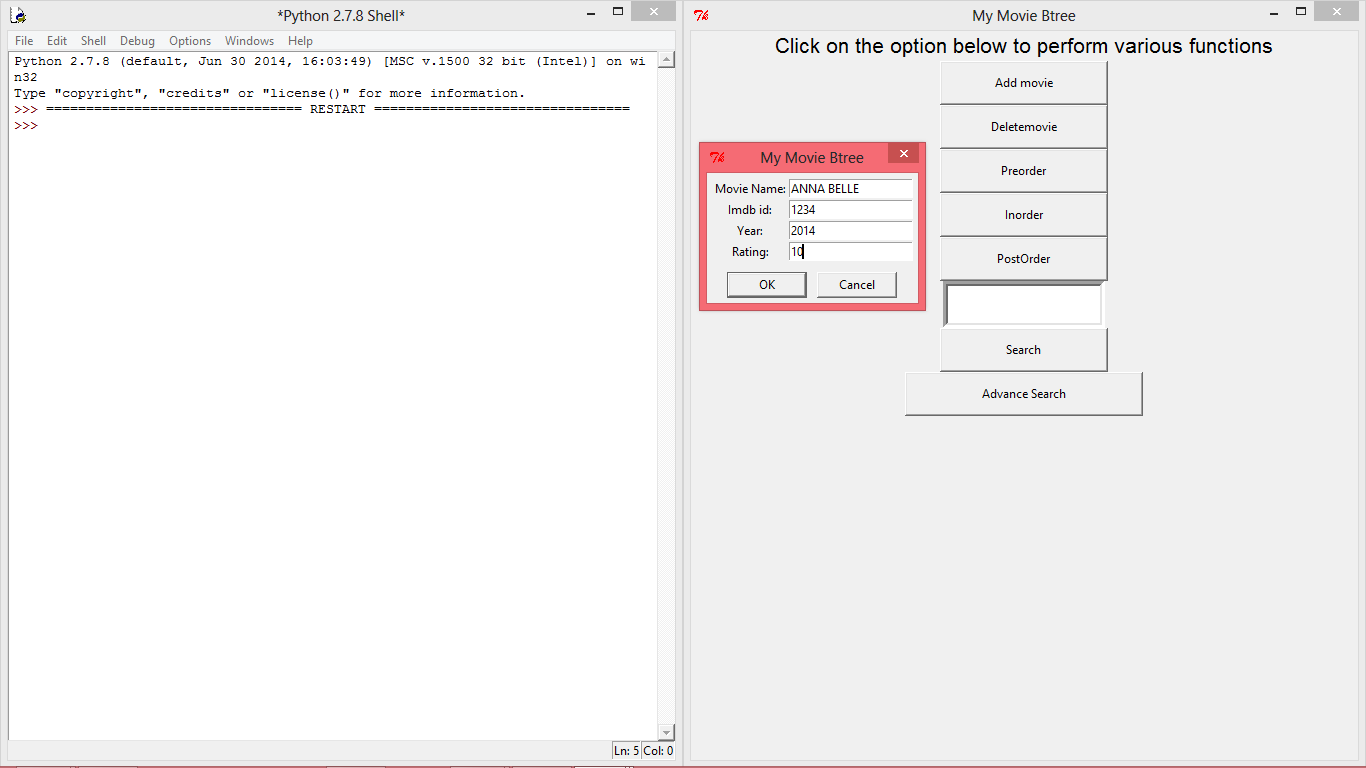
**3)Binary Search tree Functions:**

* PUT
* GET
* Find min
* Delete
* Inorder
* Preorder
* Postorder

**GUI**

* Add movie
* Delete movie
* Search
* Post order
* Inorder
* Preorder
* Advance search

**TESTING**

****

Clearly the above images show the testing and output

The database has been tested and used with around **900 hundred movies** .

**CONCLUSION**

* Clearly the program is a great place to store all the list of movies and the use or modify database convieniently as and when required.
* The use of Binary search tree is justified as to speed up the search for which the database is made of
* The online version also helps to get all the information about the movies available at fingertips with a easy user friendly interface.

**FUTURE ENHANCEMENTS**

* The online version is slow and lacks the server request optimizations, clearly it needs some more work with usage of knowledge in networks we can optimize the advance search.
* The database can also have an additional feature of adding the newly released movies automatically to the database when connected to the network.
* Additional way to build a more robust data base with database management systems like SQL

**REFERENCES**

* <http://interactivepython.org/runestone/static/pythonds/Trees/bst.html>
* Youtube Videos related to Graphical user interface on Tkinter
* Stackoverflow Questions related to Binary search tree DELETION of node