DOCUMENTATION

ResCite Team

Contents

1. Introduction

1.1 Preface

Citations of a particular article are other articles, books or papers that refer to that article. A citation acknowledges the relevance of other articles and books in that topic of discussion. The number of citations of a paper is an indicator of the impact of the paper. The number of citations of the papers written by an author or published in a journal is an indicator of the quality of the researcher/journal in research.

Currently, there are many existing software that provide information about the citations of papers. But the required information is not available in an organised manner. ResCite overcomes this shortcoming by providing results in an organised way – sorted by the number of citations or by date, along with providing other relevant features.

1.2 Users

1. **Research Community:** The research community can use this software for the following purposes:
   a) Researchers can use the software to know about the research work done by their peer. They can also assess the experience of a fellow researcher in research. So when a researcher wants to collaborate with another researcher, this can be used as one of the parameters to decide.
   b) Researchers can assess the quality of a journal using the software. The researcher could use this software to decide which journal to publish his paper in.

2. **Students:** The students can use the software to assess the quality of research done by a professor before applying for research projects under professors.

3. **Research Analysts:** This software can be used by research analysts to assess the quality of research going on in an institute or place, by assessing the quality of researchers working in the place.

2. Features

2.1 Target features

Given the name of the author/journal, the software gives all corresponding papers sorted by number of citations or by date. The search can also be done for any user-given year range. The software displays the h-index, i-index of the author/journal and statistics like the total number of citations and number of citations per paper. For any paper, the software also shows the list of citations to the paper.

2.2 Additional features

1. **Multiple Tabs**:

The software provides different features such as search, view citations and view profile. The user may want to use several of these features at the same time. Also the user may also want to make multiple searches at the same time. The multiple tabs feature comes as a great addition to the software in this regard. In today's Google Chrome era, where tabs are a part of everyday internet usage, the tab feature enhances the feel of using the software.

The implementation of the multiple tabs feature has been done as follows. There is a Main Window which handles the tab feature. Every time the user opens a new tab, a new tab object is created with a pre-defined layout. WPF provides a basic tab controller option, with the main window initially being created with one tab. Various functionalities were then added to the tabs, namely – adding new tabs, closing tabs, transferring control when a tab is created, closed or selected. Each tab runs on a separate thread. So the search in each tab is independent of the searches in other tabs.

2. **Shortcut keys:**

Shortcut keys have also been assigned to commonly used functions relating to tabs. The shortcut keys have been set similar to that in Google Chrome for the ease of the user. The following are the shortcut keys used:

i) Ctrl + T – Open new tab
ii) Ctrl + W – Close current tab
iii) Ctrl + Tab – Move between tabs from left to right
iv) Ctrl + Shift + Tab – Move between tabs from right to left
v) Ctrl + <tab number n> - Go to the nth tab (0< n <9)
vi) Ctrl + 9 – Go to the last tab

3. **Pagination:**

The author or journal name that the user is searching for can have up to thousands of results. The pagination feature divides these results into pages of 20 results each. The software displays the results as pages and the user can move between pages using the page number provision provided at the bottom.

When a particular page number is clicked, the results corresponding to that page alone are queried out. This feature makes the software memory space-efficient since all results corresponding to the author/journal need not be stored. The required results alone can be obtained. This feature has been implemented by building the query using the page number chosen. By default, the results of the first page are displayed.

4. **Preview pane:**

In any general citation analysis software, the user needs to go to the particular paper to get the details of the paper. However in ResCite, we have implemented a preview pane to improve the user-experience. When the user clicks on the paper, a preview of the paper is shown in the right side of the screen, showing details of the name of the paper, co-authors, abstract and the publication URL. Hence the user need not unnecessarily navigate between pages. If the user gets the required information, he can go ahead with his next search and only otherwise he needs to follow the link to the paper. All the features in the preview pane are selectable.

When a particular author/journal name is searched, the details of all the corresponding papers are queried. The titles of the papers are displayed as results and the remaining details are displayed as a preview to the paper when clicked on the paper.

5. **Favourites:**

The user may have some regular searches that he repeats often. ResCite implements a ‘Favourites’ feature to remember these searches. So no, the user need not set all the parameters again and need not type the search query every time. The software remembers it for him. At any point, the user can add a search to his favourites, select a search from his favourites, delete a particular search from his favourites or clear his list of favourites. The user can also get the results for any search saved in the favourites by clicking on the ‘View Search’ button.

Each search added to the favourites is stored as a Query object which has all the parameters for the search (query string, sort preference, etc.). The list for favourites is stored in a file named ‘.favourites’. Every time the application starts, it loads the list of favourites. When a new search is added/deleted to favourites, it makes the changes to the file during runtime. When the user clicks on ‘View Search’, a new tab with those search parameters is initiated.

6. **Export:**

Sometimes the user may wish to have the entire list of papers for an author or journal. The export option allows the user to download the entire list of results for an author or journal as a TSV file. A TSV file contains all the papers published by (in) an author (journal) separated by tabs. The user can select where the TSV file is to be saved. The TSV file can be opened in all commonly used software like Notepad, Wordpad and Microsoft Excel.

7. **Progress bar:**

The progress bar gives a graphical representation of how much of the search is complete. This feature keeps the user engaged when he is waiting for the results. The progress bar is present behind the search bar and is made visible by reducing the opacity of the search bar when the search is initiated. The search bar remains active when the progress bar is being displayed.

8. **Cancel:**

The user may have searched for a wrong query and want to cancel the search when it is in progress. The cancel symbol at the right end of the search bar allows the user to do this.

9. **Status bar:**

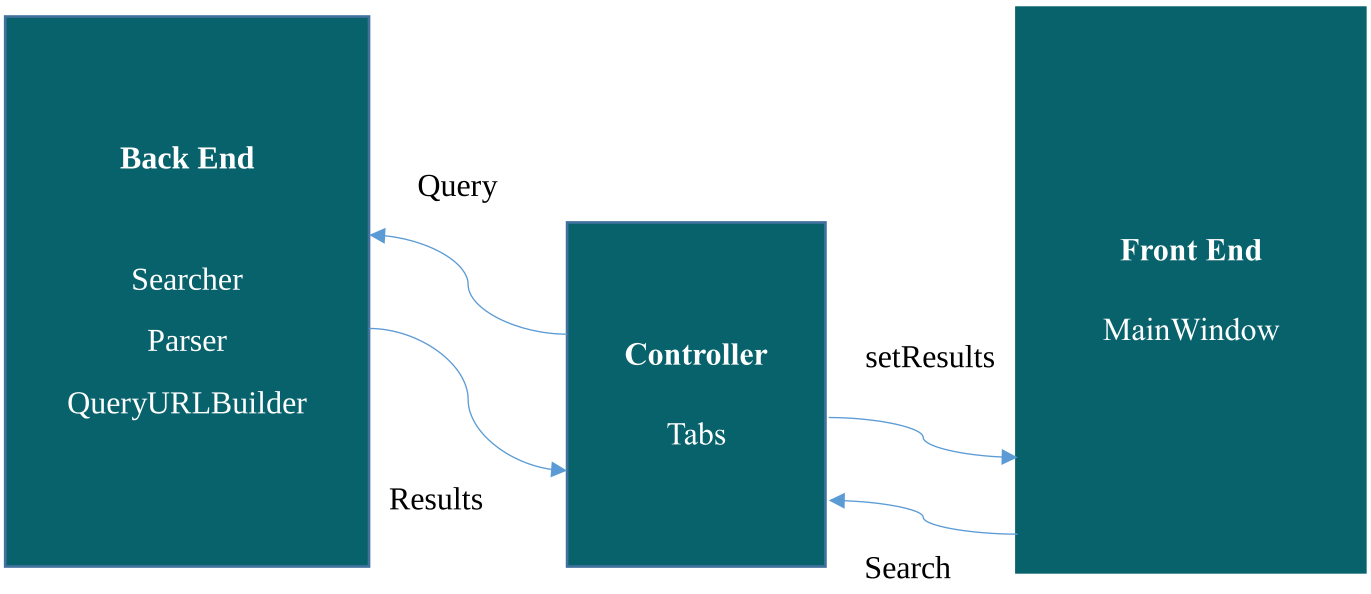
The user may want to know the status of the function being performed. The status bar at the bottom of the page displays the current status of the feature being implemented**.**

10. **Help page:**

At any point, the user is not sure of how a particular feature can be used, the help button is available at the top right of the screen. For first-time users, the welcome screen has a ‘Getting started’ option which also leads to the Help page. The help page opens as a html page in a new tab.

3. Architecture

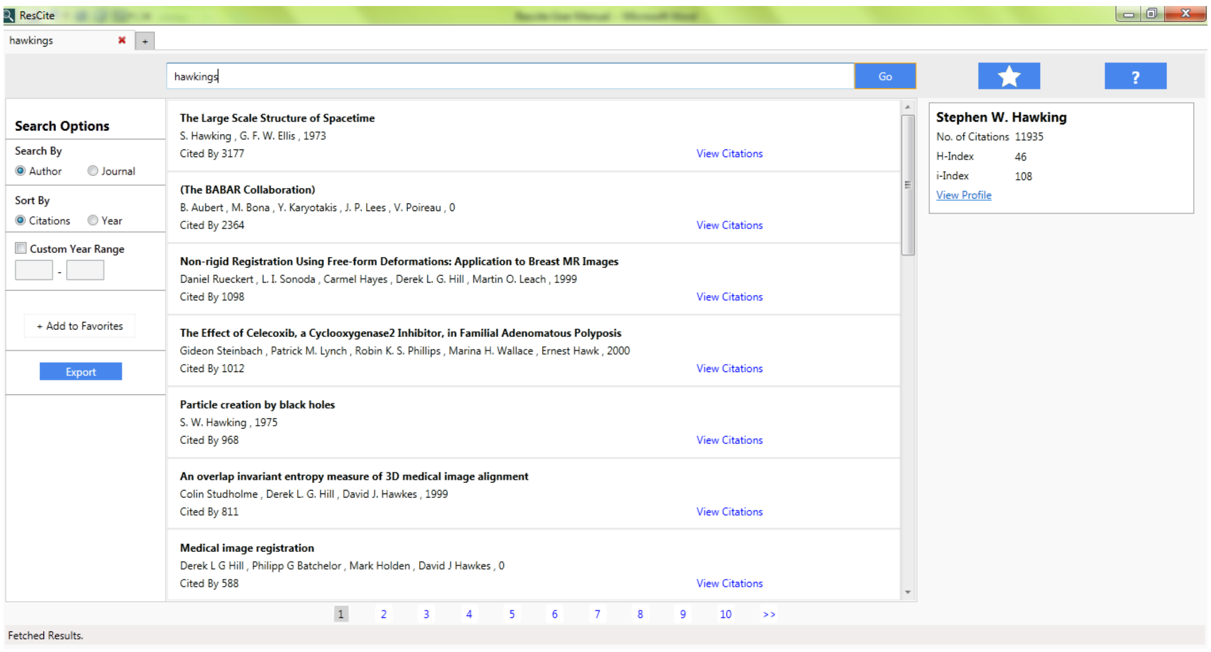
3.1 High level design



The FrontEnd(UI) gets the search parameters (query string, sort preferences, query type, etc.) and passes it to the Controller. The controller forms a query object using the search parameters and passes it to the searcher. The searcher passes the query object to QueryURLBuilder to get the query URL. Using this query URL, the Parser gets the response from the source and extracts the relevant data and stores it in the Results object. There are two sources for extracting the results – namely Google scholar and Microsoft Academic Search. Each source has its own QueryURLBuilder and Parser. The searcher returns the Results object to the Controller which prompts the UI to display the results.

3.2 Interface design

The interface has been designed in an efficient way such that the control flows from the left side of the screen to the centre to the right side. The left pane contains the search options, which the user fills in. A checkbox is available for searching in a custom year range. The textboxes for filling in the years are greyed out initially. When the custom year range option is checked, the textboxes accept user input. The export and ‘Add to favourite’ buttons are present below the search options.


ResCite

hawkings

hawkings

Search Options

Search By
[✓ Author] Journal

Sort By
[✓ Citations] Year

Custom Year Range

+ Add to Favorites

Export

The Large Scale Structure of Spacetime
S. Hawking , G. F. W. Ellis , 1973
Cited By 3177 View Citations

(The BABAR Collaboration)
B. Aubert , M. Bona , Y. Karyotakis , J. P. Lees , V. Poireau , 0
Cited By 2364 View Citations

Non-rigid Registration Using Free-form Deformations: Application to Breast MR Images
Daniel Rueckert , L. I. Sonoda , Carmel Hayes , Derek L. G. Hill , Martin O. Leach , 1999
Cited By 1098 View Citations

The Effect of Celecoxib, a Cyclooxygenase2 Inhibitor, in Familial Adenomatous Polyposis
Gideon Steinbach , Patrick M. Lynch , Robin K. S. Phillips , Marina H. Wallace , Ernest Hawk , 2000
Cited By 1012 View Citations

Particle creation by black holes
S. W. Hawking , 1975
Cited By 968 View Citations

An overlap invariant entropy measure of 3D medical image alignment
Colin Studholme , Derek L. G. Hill , David J. Hawkes , 1999
Cited By 811 View Citations

Medical image registration
Derek L G Hill , Philipp G Batchelor , Mark Holden , David J Hawkes , 0
Cited By 588 View Citations

1 2 3 4 5 6 7 8 9 10 >>

Stephen W. Hawking

No. of Citations	11935
H-Index	46
i-Index	108

View Profile

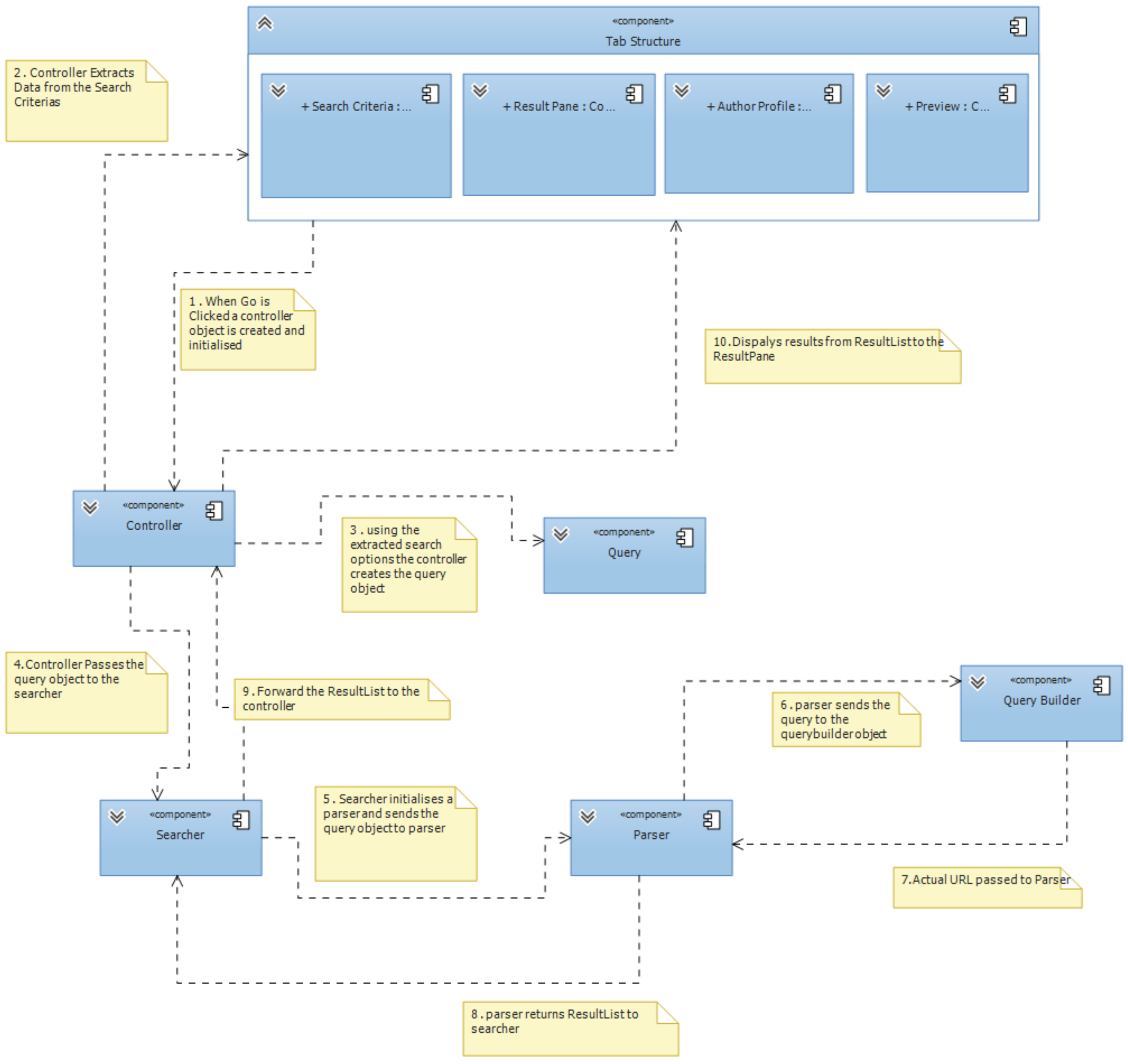
Fetched Results.

The centre pane contains the search bar and the results. The search bar has been designed similar to the Google search bar, the most widely used search engine, to improve the user experience. A progress bar behind the search bar shows the progress of the search and keeps the user engaged while the results are being extracted. The results are displayed below the search bar, twenty results at a time. A scroll bar is available for seeing the twenty results. The page numbers are displayed at the bottom similar to Google search. The View Citations option in the right corner of each result, when clicked, opens a new tab showing the list of citations to the paper. When the result is clicked, the control flows to the right panel, which shows the preview of the paper.

The right panel constantly displays the author/journal profile at the top. When any one of the results is selected, the preview of the paper is displayed below the profile.

The add tab button is present to the right of the last tab. Every new tab gets attached to the right of the available tabs. The show favourites and help buttons are present at the right top corner. The status bar at the bottom dynamically displays the status of the search.

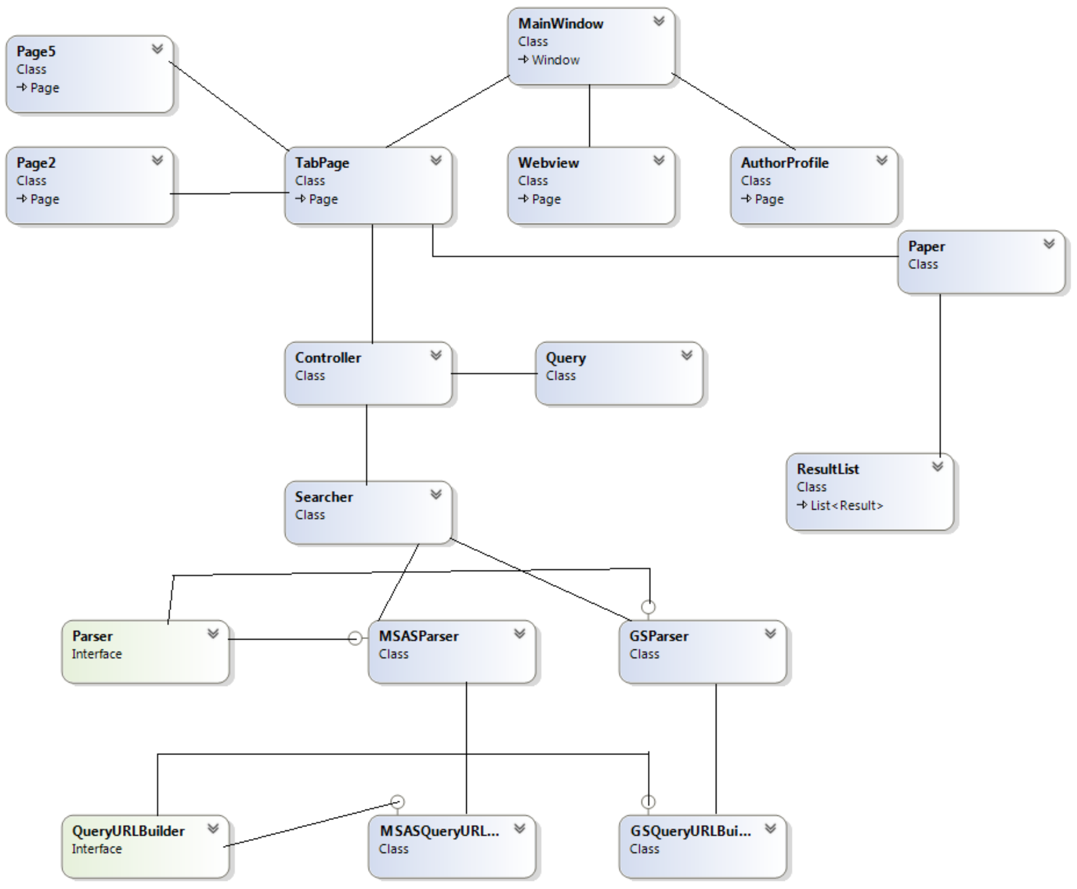
3.3 Class level design

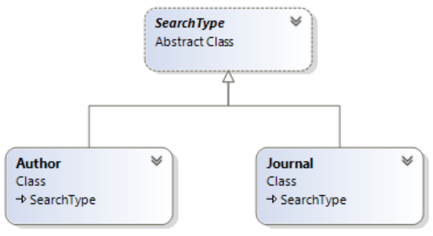

«component»
Tab Structure
+ Search Criteria : ...
+ Result Pane : Co...
+ Author Profile : ...
+ Preview : C...
2. Controller Extracts Data from the Search Criterias
1. When Go is Clicked a controller object is created and initialised
10. Dispalys results from ResultList to the ResultPane
«component»
Controller
3. using the extracted search options the controller creates the query object
«component»
Query
4. Controller Passes the query object to the searcher
9. Forward the ResultList to the controller
6. parser sends the query to the querybuilder object
«component»
Query Builder
«component»
Searcher
5. Searcher initialises a parser and sends the query object to parser
«component»
Parser
7. Actual URL passed to Parser
8. parser returns ResultList to searcher

The MainWindow class is the main application class. It contains the TabPage, WebView and AuthorProfile classes in it. They are basically part of the Tab Structure. The TabPage consists of search criteria such as data members and has a result pane, author profile viewer and a preview pane. AuthorProfileGUI is used to view the profile of the selected author. Whenever a page is selected, the abstract of the page and related links are displayed in the preview pane.

When the 'View Publication' button in the Preview pane is clicked the original webpage of the publication is displayed in a new tab in the webView Object. Whenever the 'View profile' is clicked from the authorProfileGUI object, the author profile is displayed in a new tab in an authorProfile Object.

After entering the query and selecting the search options, a new thread is created when the Go button is clicked. Through the use of multithreading, the tab continues to respond to the user while the search is going on.





When the Go button is clicked, a new controller object is created and the reference to the current tabPage object is passed to the controller. The controller then extracts the search options from the tabPage object and stores the extracted information in a query object. Then it creates a searcher object and passes the query to the searcher. On receiving the query, the searcher creates a GSParser (used to get results from Google Scholar) and forwards the query object to GSQueryBuilder. The GSQueryBuilder object builds the required URL from the query and sends it back to the parser.

The parser object makes a query to the Google Scholar Database using the URL and extracts the information required for parsing. The parsers used are HTMLAgility Pack (Microsoft Public License) and NewtonSoft JSON (MIT license). The parser object then extracts the required data and creates a resultList object. It forwards it to the searcher and the searcher passes it to the controller.

The Controller checks if the author profile is available. If it is not, it gets the author profile from Microsoft Academic Search. For this, the controller creates a MSParser object. The controller also creates the Query object and passes it to the QueryURLBuilder . The URL from the QueryURLBuilder is passed to the MSParser object which gets the required author profile data.

Now the controller has the all the required data and passes the resultList to the tabPage object of the corresponding tab. The paper, author list, year, citation count and the 'View Citation' URL are stored and displayed. When the 'View Citations' button is clicked, the list of citations is opened in a new tab. A new Controller object is created and a similar process is repeated.

3.4 Sources

The required data is collected from the following three websites:

1. **Google Scholar** (http://scholar.google.com): Given an author/journal name, the corresponding results and author profile are extracted from Google scholar, whenever it exists.

2. **Microsoft Academic Search API** (http://academic.research.microsoft.com): Given an author/journal name, the corresponding results and profile are extracted from Microsoft Academic Search.

3. **SCImago journal and Country Rank** (www.scimagojr.com): Given any journal, the corresponding statistics are extracted from Scimago Journal and Country Rank.

4. Testing

4.1 Test plan

We have implemented the following Test Plans.

1. Unit Test for each Method in the code.
2. Integration Tests between different modules of the code
3. Use Case Testing to check for the Corner cases.
4. Stress Testing

4.2. Unit Testing

Unit Tests are written for each method where we test their functionalities of the method independently by mocking the necessary dependencies of the method.

Unit test cases are created for the following classes.

- Controller
- Exporter
- Favourites
- GSParser
- MSASParser
- Query
- Searcher

These test cases can be found in the ResciteTest Folder.

4.3. Integration Test

Interaction between different modules of the Rescite are tested to test the robustness and the functionality of the Application.

4.4 Use Case Testing

Following Use Case have been tested which cover all the Corner Cases possible

1. No Internet Connection
   A Pop Up Box alerting the User about the non-availability of Network

2. Search String is NULL
   A Message Box is popped out asking the User to enter a Valid String

3. String with No Expected Results
   No Results Page is displayed in the Results Pane.

4. Connection Time-out
   If the results take too long to be fetched, User is informed by a Pop Up Box stating that the results couldn’t be fetched.

5. Couldn’t Connect to the Data Sources

If a Connection to the Data Sources couldn’t be due to Unavoidable circumstances, a pop up informs the same to the User.

6. Cancel Search

When the User cancels a search, the current search under progress stops and the Tab is ready to accept a new Search from the User.

7. Clicking the GO button while search is going on

The current search stops and the tab is ready to accept a new Search from the User.

8. Close Tab while search is going on

The current search stops and the tab is closed. Any other tab opened previously is ready to accept the search query from the User.

9. Moving to different tabs while search is going on

The User can navigate through the tabs using the Shortcuts provided or by mouse click.

10. Parallel search on multiple tabs

The User can perform multiple searches on different tabs concurrently

11. Number of pages in the Result Set

The number of pages provided to view all the results of a Search Query fetches the results on each page click. The Number of pages is restricted by the response we get from the Data Sources

12. Application Close on closing the Last remaining tab

The application closes on the close of the last remaining tab

13. No repetition in the Favourites List

The same search query will not be added to the Favourites List.

14. Delete/Clear Favourites in case of No Favourites in the Favourites List

Delete/Clear Favourites are available only when there are Favourites in the Favourites list.

15. Favourites List restored on Restart of Application

Each the Application starts, it reads the Favourites List from .favourite file

16. No export in case of Empty Result Set
User cannot export an Empty result set . Export functionality is only available on a non empty Result Set

17. Cannot Add To Favourites on Empty Result Set
User cannot add a search to Favourites if the Result Set is empty. Add to Favourites is only available on a non-empty Result Set.

18. Result Pane cleared on Fresh Search on the Same Tab
The results pane clears the results before a new Search is made.

4.5 Stress Testing

Multiple Searches are done concurrently on different tabs to check the robustness of the Application. Different searches are done for both Author/Journal making sure that we can navigate and use the remaining tabs without affecting their individual performance. A good number of searches are searched for and the robustness of the Application is ensured

User should be able to open multiple number of tabs and the application supports any number of tabs to open.

4.6 Alpha Testing

The Application was distributed among a group of 10 people using it for their Research and Post graduate Applications.