**PROJECT REPORT**

**QA SYSTEM**

***PARADIGM:***

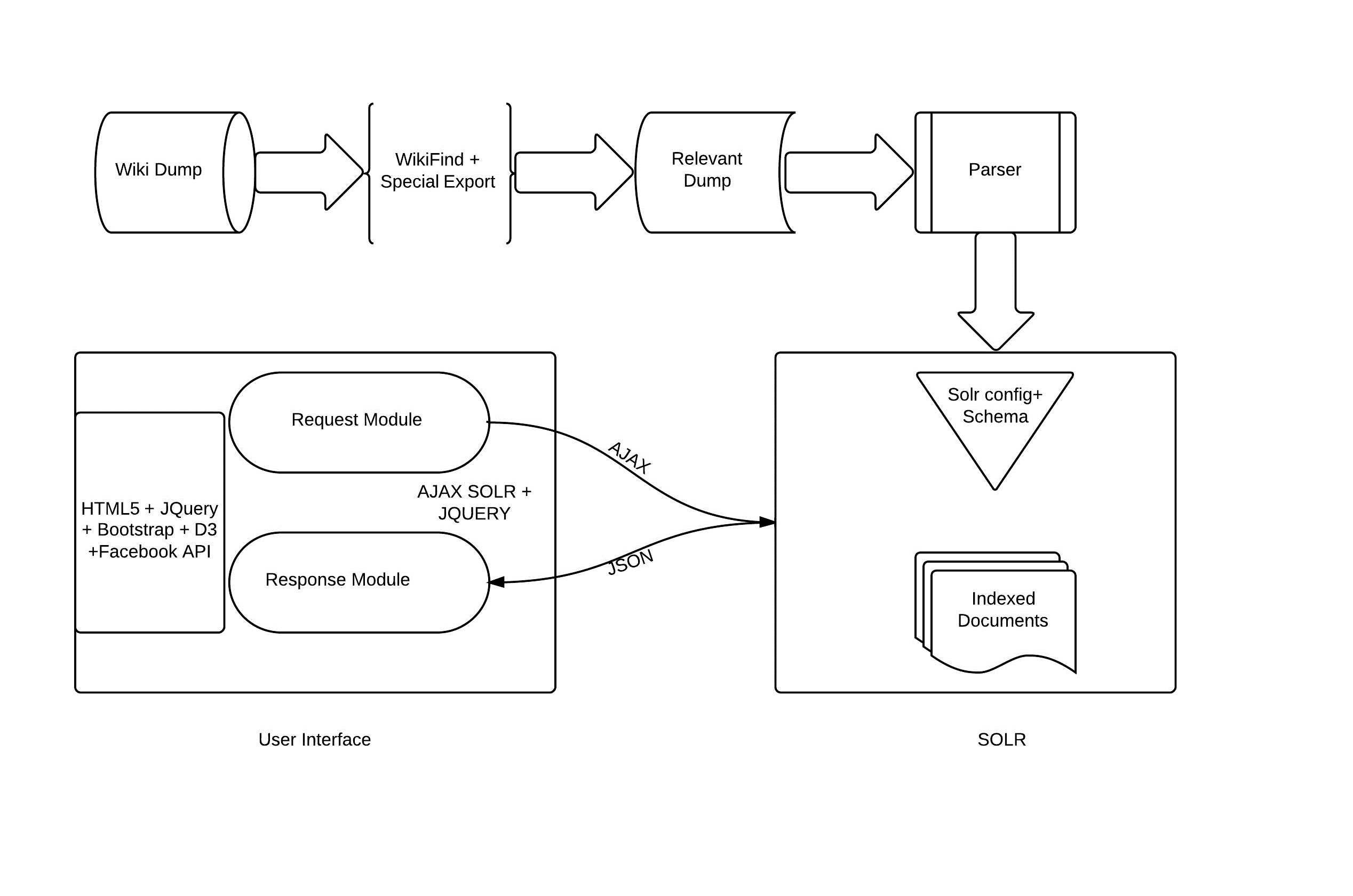
*Aswin Bharadwaj Ramesh*

*Deepak Ravishankar Ramkumar*

*Aravindhan Thanigachalam*

*Vivekanandh Vel Rathinam*

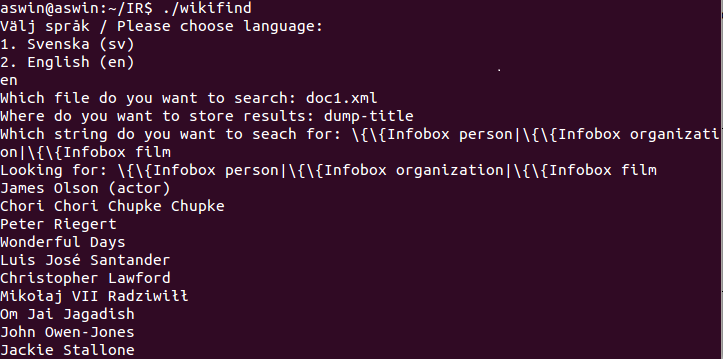
**SYSTEM DIAGRAM**

****

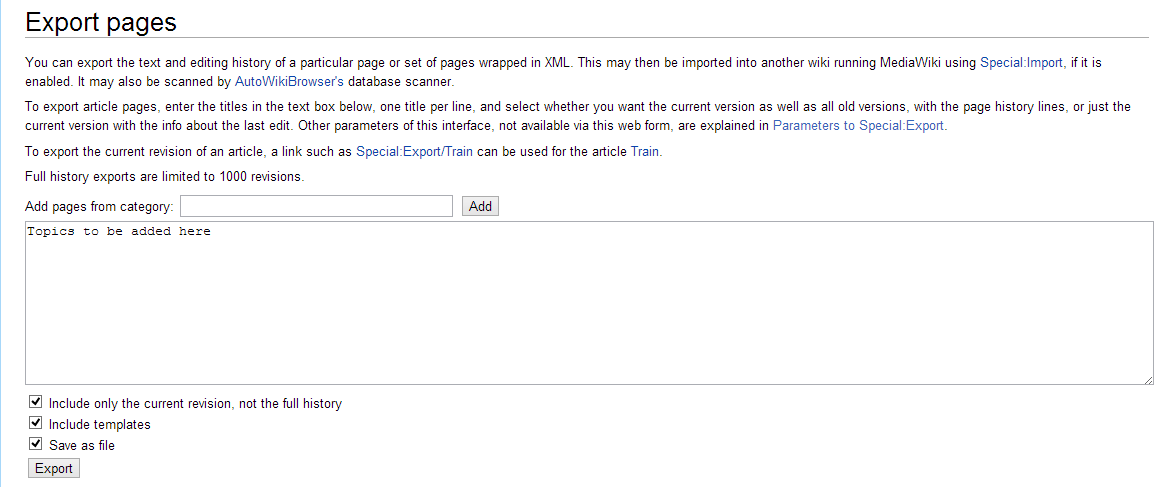
**SYSTEM DESCRIPTION**

1. Processing of the Wikipedia dump

We are using an online tool ‘WIKIFIND’, a software written in C++ to find topics of Wikipedia pages related to People, Organization and Films.



An equal proportion of these topics are given as input to another online tool ‘EXPORT PAGES’ in the Wikipedia site <http://en.wikipedia.org/wiki/Special:Export> to generate the XML dump.



The infoboxes from each document in the XML dump are parsed using a program written in Java and then the parsed contents are converted to the SOLR XML format. Each SOLR XML document will have the following fields.

1. **Person**

Name, image, caption, birth name, birth date, birth place, death date, death place, residence, nationality, other names, citizenship, education, occupation, known for, organization, notable works, title, religion, spouse, children, parents and awards.

1. **Organization**

Name, former name, image, caption, abbreviation, motto, formation, founder, extinction, type, purpose, professional title, headquarters, location, coordinates, services, leader title and leader name.

1. **Films**

Name, image, caption, film name, director, producer, writer, screenplay, based on, narrator, starring, music, cinematography, studio, distributor, released, runtime, country, language, budget and gross.

1. Indexing the SOLR formatted XML documents

After setting up the SOLR instance in any of the servlet containers, the field entries that are present in the XML document are to be defined in schema.xml. The SOLR schema will be explained in the later sections.

After the field definitions are made in schema.xml, the SOLR formatted XML documents are indexed into SOLR using the command given below

java -jar post.jar \*.xml

1. User Interface

The User Interface gives out the possible types of queries that the user can ask by loading it in drop boxes and also giving out a text box for the user to fill out the name of a person, organization or a film to be queried upon.

User has to select the respective document type to be searched on. The input the user enters in the text box is the name of the document to be searched.

The answer to be returned is based upon the questions selected from the drop boxes. A query is formed and sent to the SOLR server and the response from the server is parsed in the client side to show the results. Various features of this system will be explained in the next section.

**ADDITIONAL FEATURES OF THE QA SYSTEM**

1. ***Auto Suggest:***

We are providing the user with a set of autosuggested results which pops down in a combo as he types ahead in the search box. A facet query is sent to the SOLR server as and when the search box is changed.

1. ***Automatic Spell Correction:***

A request handler has been created in SOLR to handle spelling corrections. Whenever a query is sent to SOLR, the system checks for a JSON response with the results. If there are no results, a spell checking query is fired to SOLR which gives a response with a list of spelling suggestions. The system takes the first suggestion from the response and answers the question.

1. ***Related Trivia:***

Information related to the particular query is also displayed to the user.

1. ***Tag Cloud:***

This feature has been provided in our user interface. It lists the top 100 personalities, organizations and films. The heuristic for selecting the top 100 documents is to sort the documents in the decreasing order of information present in them.

1. ***Clustering:***

The name of the persons that are indexed are grouped based on the person’s occupation and displayed on the UI using a widget. The Facet pivoting in SOLR is used to perform the grouping.

1. ***Film Statistics:***

The top most films are displayed for each year based on their gross and budget. A combo has been presented in the User interface to select the year. A pop up window displays the name of the films and their gross. A pie chart is plotted with the gross or budget of the top six films for the selected year.

1. **Share on Facebook :**

The users will be allowed to share the application on facebook. The application has been hosted at <http://qasysapp.herokuapp.com/>.

**CONFIGURATION DETAILS IN SOLR**

1. Schema

All the fields in the SOLR XML document are indexed and stored as a string value using the SOLR StrField class. A unique ID is also created for each document dynamically using the SOLR UUIDField class.

Field types has been created for the autocomplete and the spelling suggestion feature. For the autocomplete feature, the corresponding field type indexes the names as n-grams which makes retrieval possible in the autosuggestion implementation.

<fieldType name="text\_autocomplete" class="solr.TextField"> <analyzer type="index"> <tokenizer class="solr.WhitespaceTokenizerFactory"/> <filter class="solr.LowerCaseFilterFactory"/> <filter class="solr.EdgeNGramFilterFactory" minGramSize="1" maxGramSize="25" /> </analyzer> <analyzer type="query"> <tokenizer class="solr.WhitespaceTokenizerFactory"/> <filter class="solr.LowerCaseFilterFactory"/> </analyzer> </fieldType>

Another field type for spelling correction has been created to split the names of the documents before indexing it. The white space tokenizer has been used to perform this task. The spell correction component performs the corrections comparing with the individual tokens in the index.

<fieldType name="text\_general" class="solr.TextField" >

<analyzer type="index">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />

<filter class="solr.SynonymFilterFactory" synonyms="index\_synonyms.txt" />

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />

<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" />

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

Field names which are comma delimited like the occupation of a person are delimited and indexed using the pattern tokenizer factory.

<fieldType name="str\_split\_on\_comma" class="solr.TextField">

<analyzer>

<tokenizer class="solr.PatternTokenizerFactory" pattern="[,/]s\*" />

<filter class="solr.LowerCaseFilterFactory" />

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />

<filter class="solr.TrimFilterFactory" />

</analyzer>

</fieldType>

The location field type is defined to hold the coordinates of a location.

<fieldType name="location" class="solr.LatLonType" subFieldSuffix="\_coordinate"/>

<dynamicField name="\*\_coordinate” type="tdouble" indexed="true” stored="false" />

1. Request Handlers

A request handler named select has been created to perform the default querying in SOLR. Number of rows to be returned and the default field to be searched has been specified as parameters.

Another request handler has been created for spelling corrections. This request handler uses a custom defined search component to perform the correction. This search component uses the DirectSolrSpellChecker and WordBreakSolrSpellChecker classes from SOLR.

<requestHandler name="/spell" class="solr.SearchHandler" startup="lazy">

<lst name="defaults">

<str name="df">name\_spell</str>

<str name="spellcheck.dictionary">default</str>

<str name="spellcheck.dictionary">wordbreak</str>

<str name="spellcheck">on</str>

<str name="spellcheck.extendedResults">true</str>

<str name="spellcheck.count">100</str>

<str name="spellcheck.alternativeTermCount">50</str>

<str name="spellcheck.maxResultsForSuggest">50</str>

<str name="spellcheck.collate">true</str>

<str name="spellcheck.collateExtendedResults">true</str>

<str name="spellcheck.maxCollationTries">100</str>

<str name="spellcheck.maxCollations">50</str>

</lst>

<arr name="last-components">

<str>spellcheck</str>

</arr>

</requestHandler>

<searchComponent name="spellcheck" class="solr.SpellCheckComponent">

<str name="queryAnalyzerFieldType">text\_general</str>

<lst name="spellchecker">

<str name="name">default</str>

<str name="field">name\_spell</str>

<str name="classname">solr.DirectSolrSpellChecker</str>

<str name="distanceMeasure">internal</str>

<float name="accuracy">0.3</float>

<int name="maxEdits">2</int>

<int name="minPrefix">1</int>

<int name="maxInspections">5</int>

<int name="minQueryLength">4</int>

<float name="maxQueryFrequency">0.01</float>

<float name="thresholdTokenFrequency">.01</float>

</lst>

<lst name="spellchecker">

<str name="name">wordbreak</str>

<str name="classname">solr.WordBreakSolrSpellChecker</str>

<str name="field">name\_spell</str>

<str name="combineWords">true</str>

<str name="breakWords">true</str>

<int name="maxChanges">10</int>

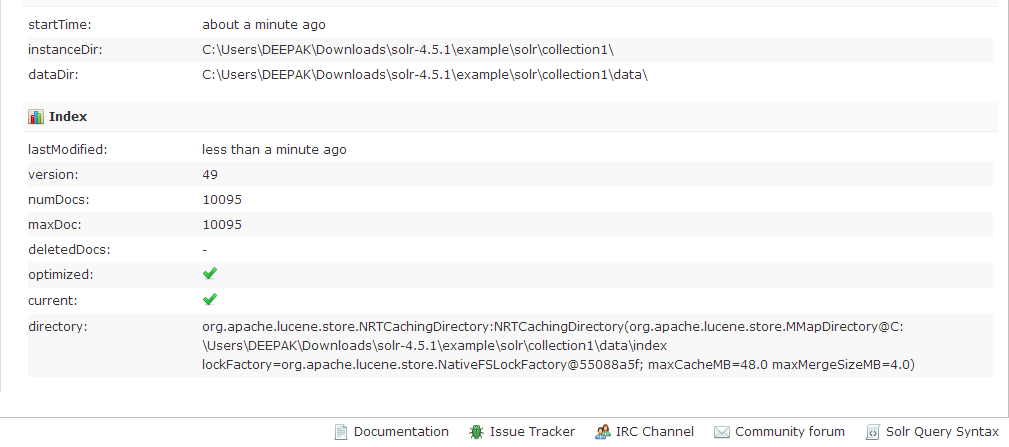
</lst>

</searchComponent>

**SOLR STATS**

1. **Index Size**

We have indexed 10095 documents in the SOLR server.

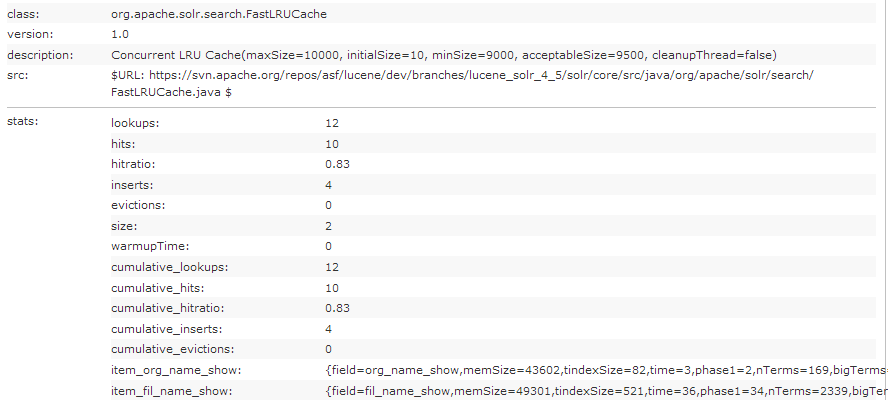
****

1. **Stats on Caching**

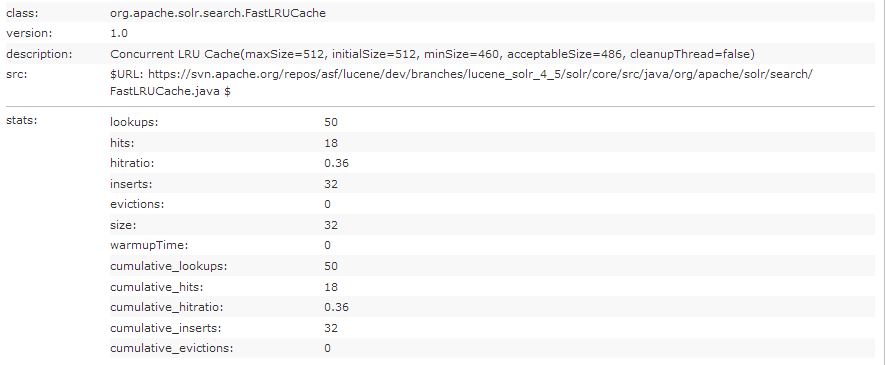
**Document Cache:**

****

**Field Value Cache:**

****

**Filter cache**

****

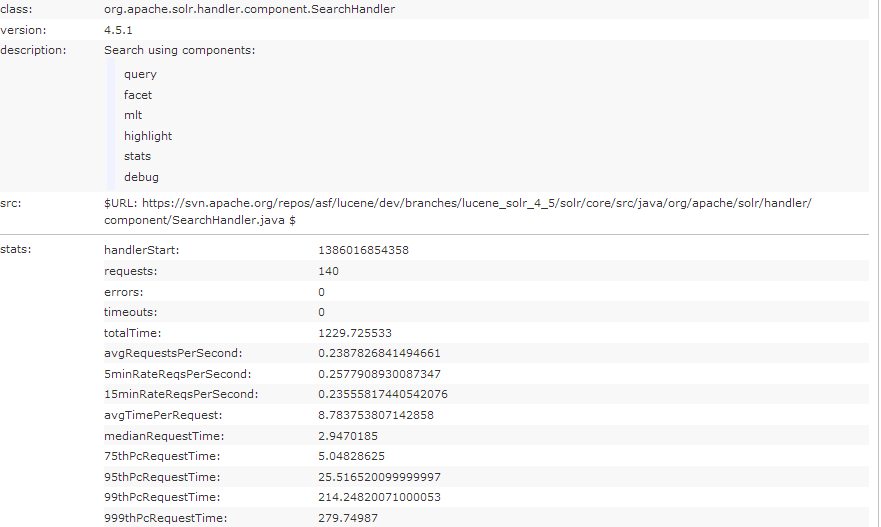
**Query Result Cache**

****

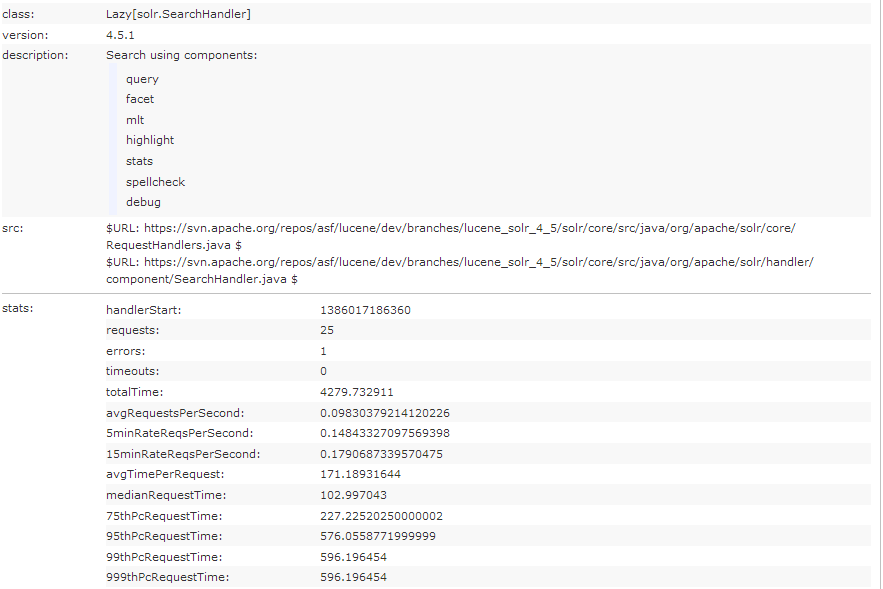
1. **Stats on Performance**

**Response Time:**

1. **Default query handler**

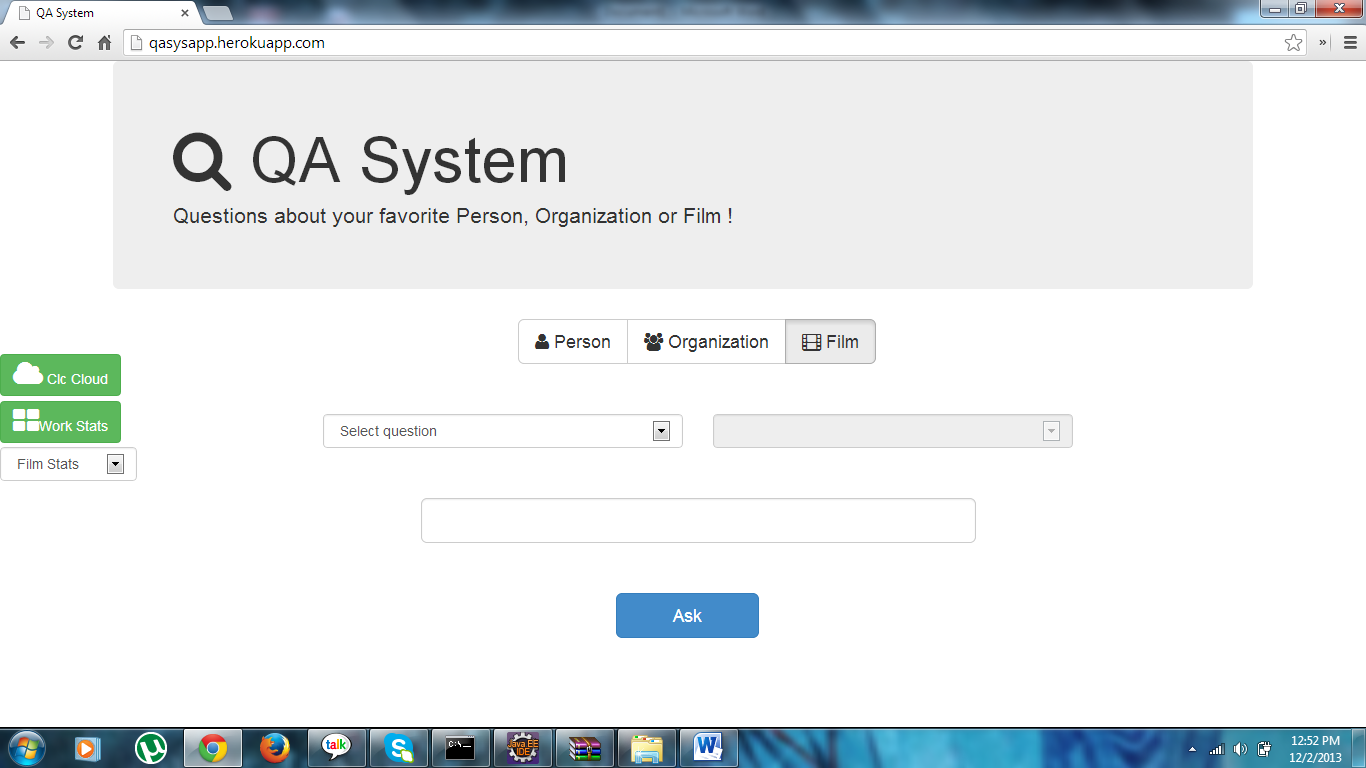
****

1. **Spell Check handler**

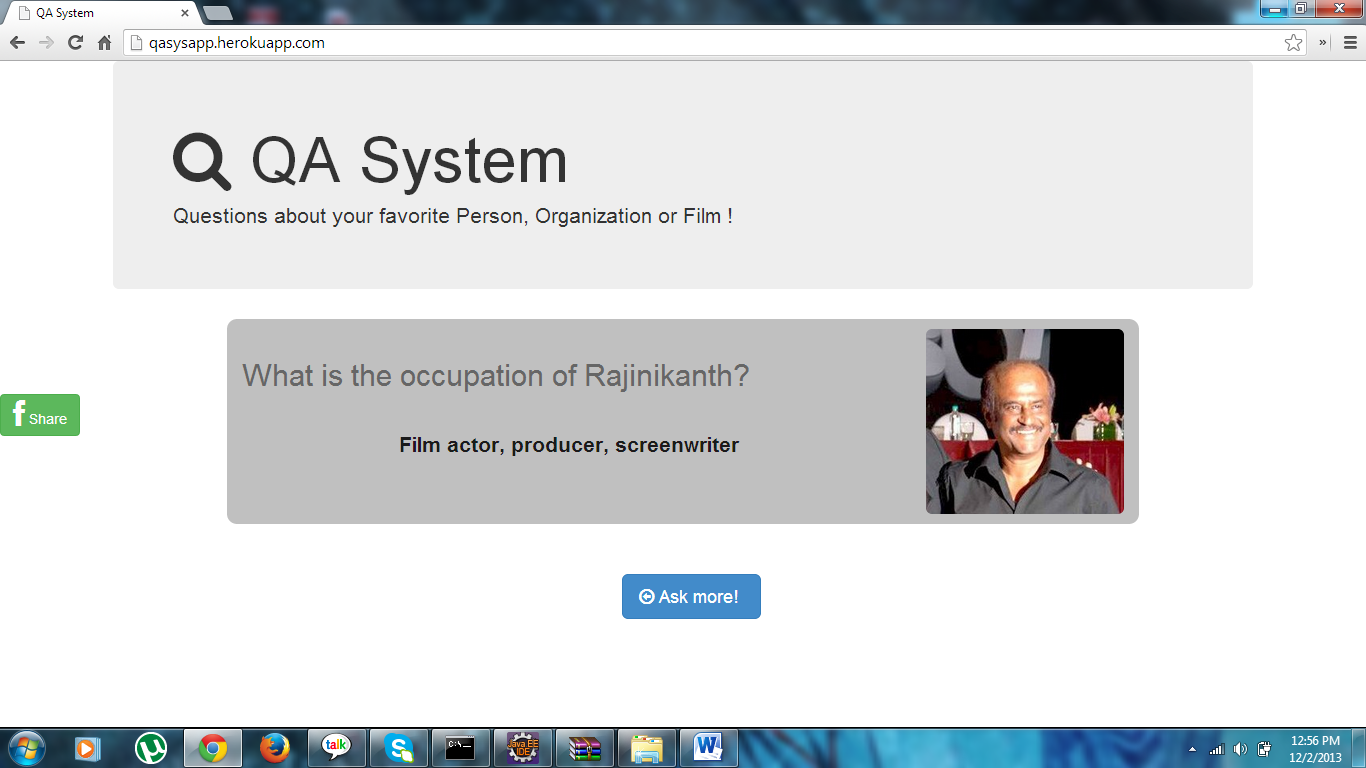
****

**UI SCREENSHOTS:**

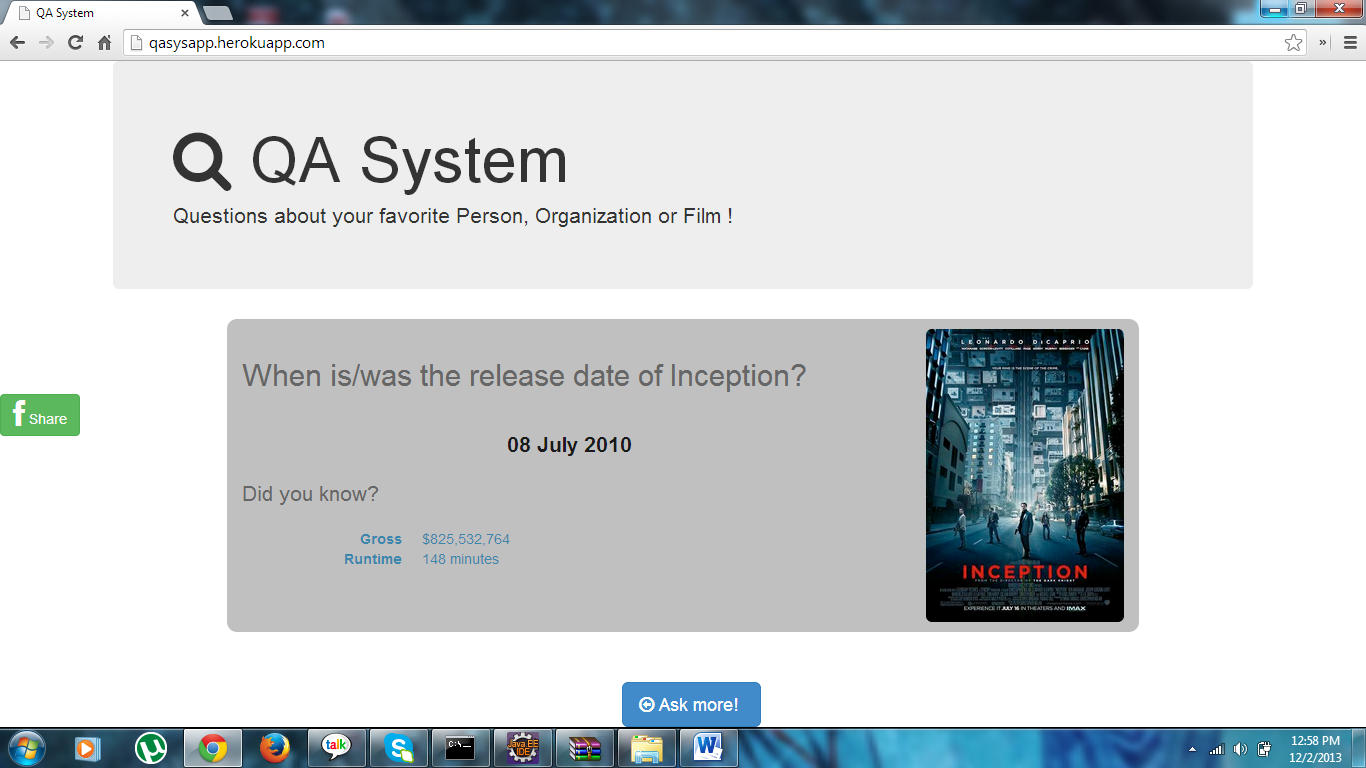
**Basic System:**



**Query regarding a Personality:**



**Query about a Film:**

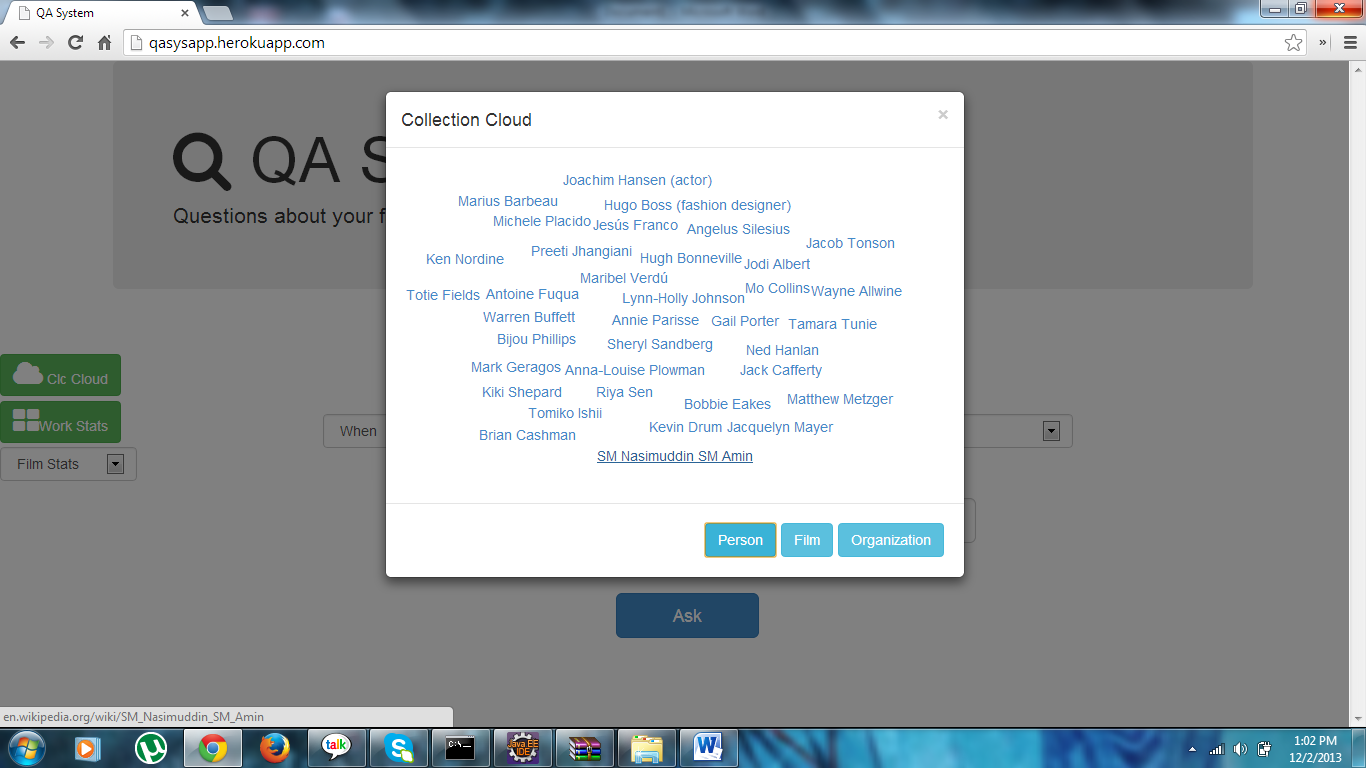


**Query about an Organization:**

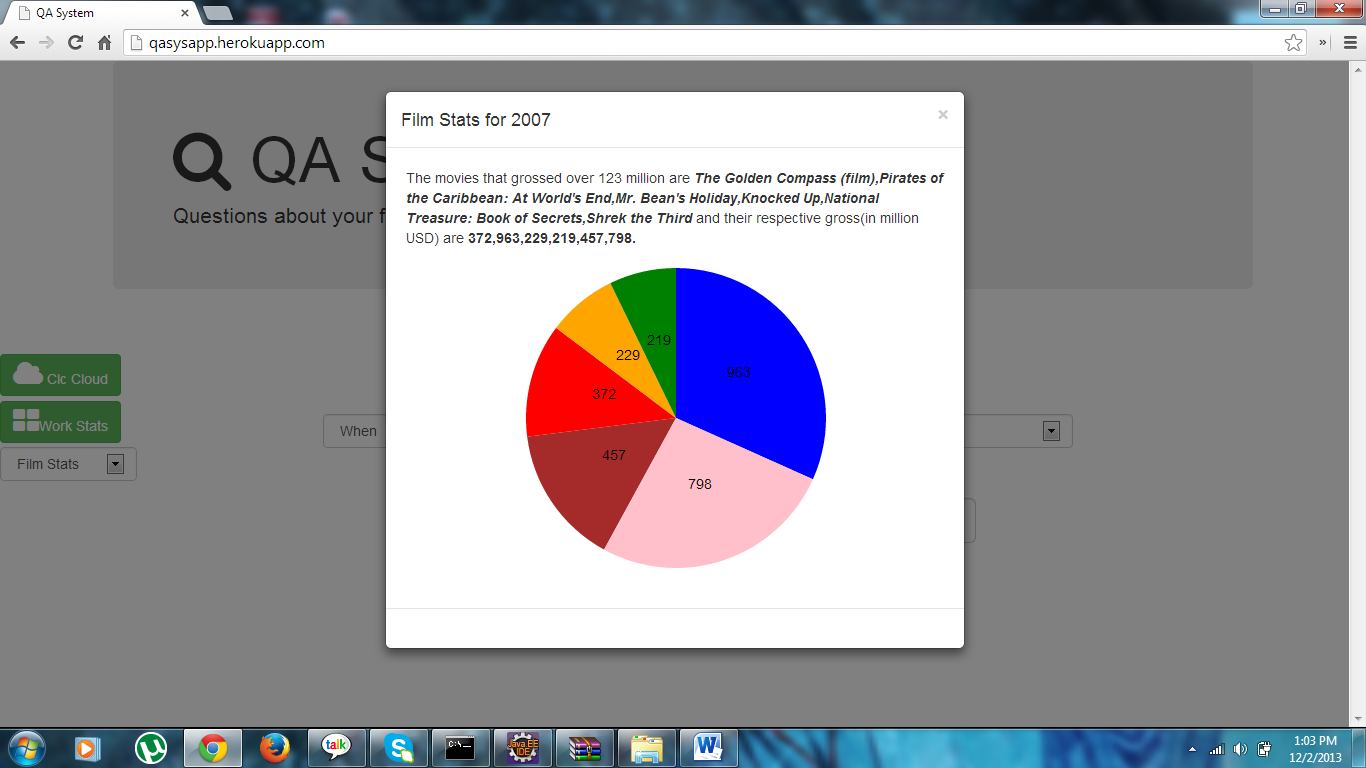


**ADDITIONAL FEATURES (USP’s):**

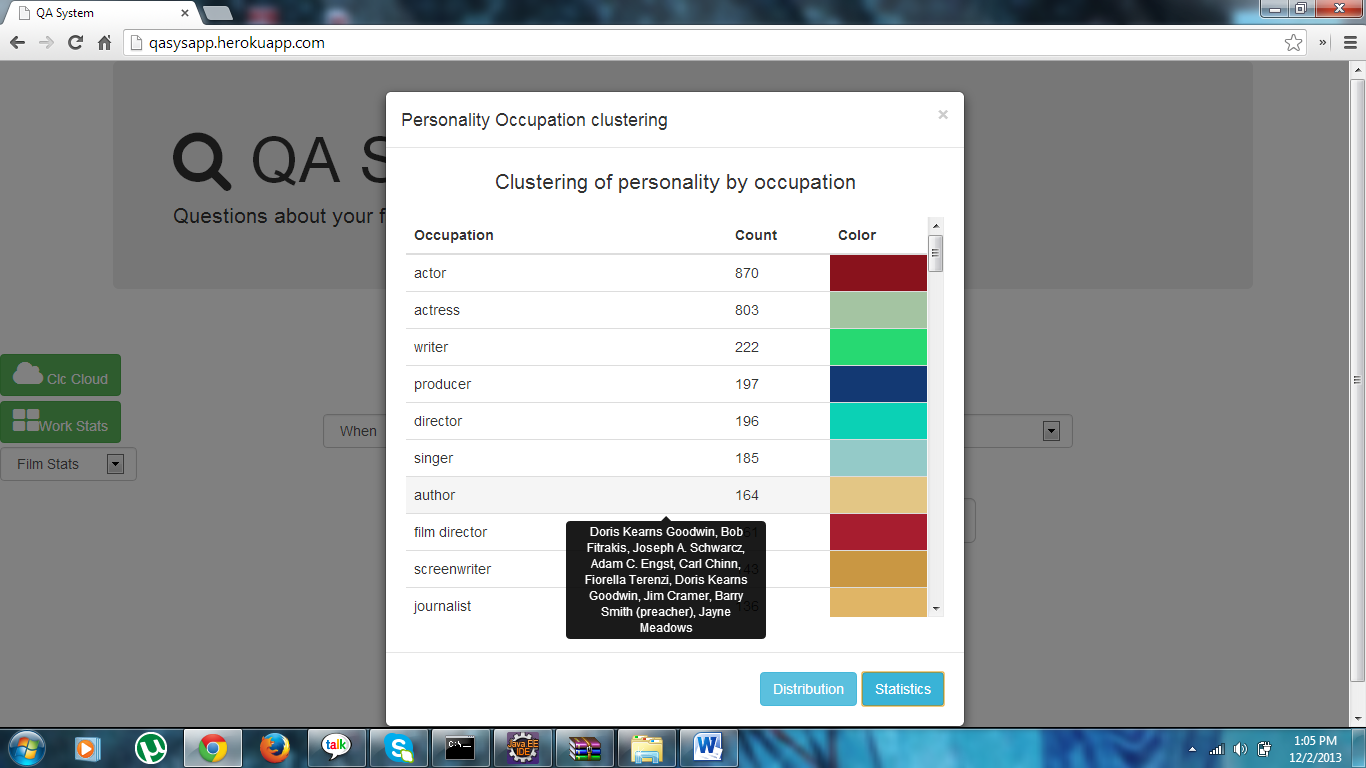
**Tag Cloud for easy navigation:**

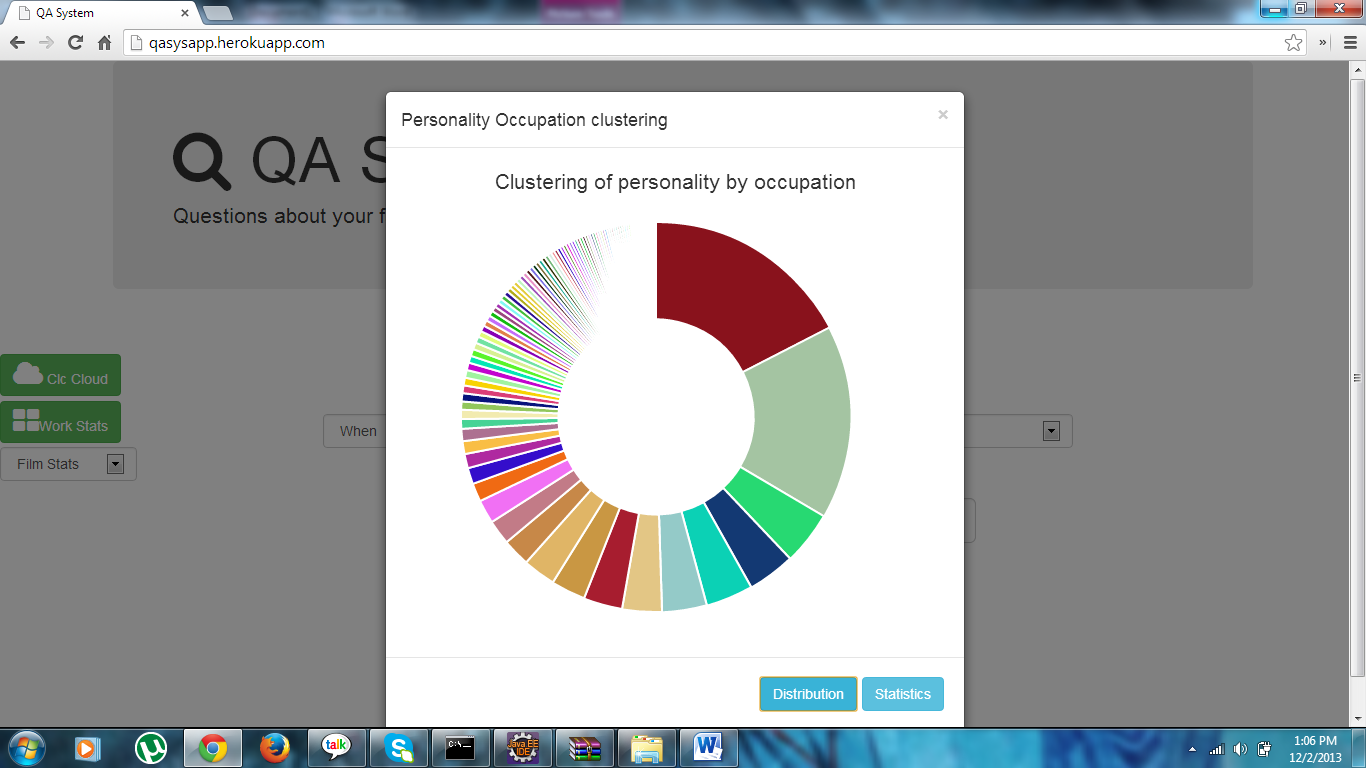


**Dynamic Film Statistics represented graphically based on the year selected by the user:**

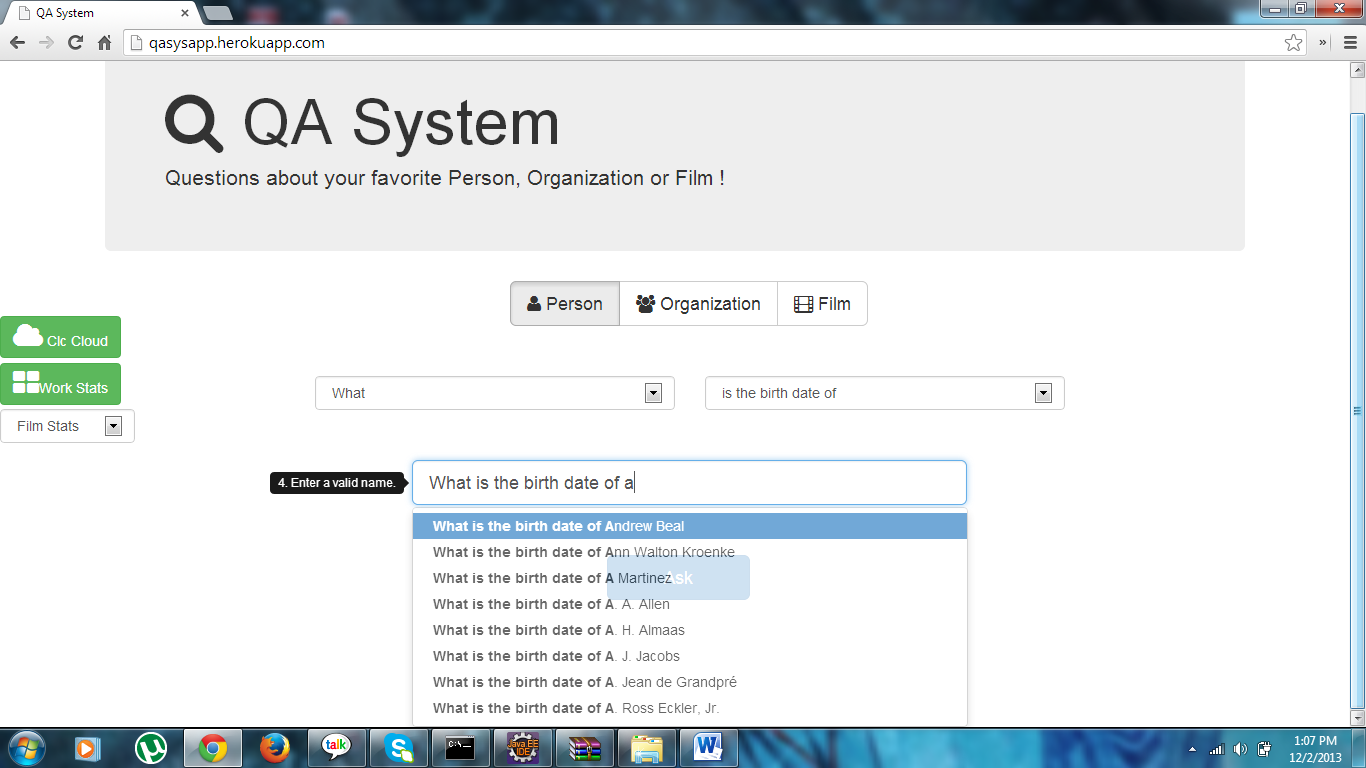


**Clustering of Data:**

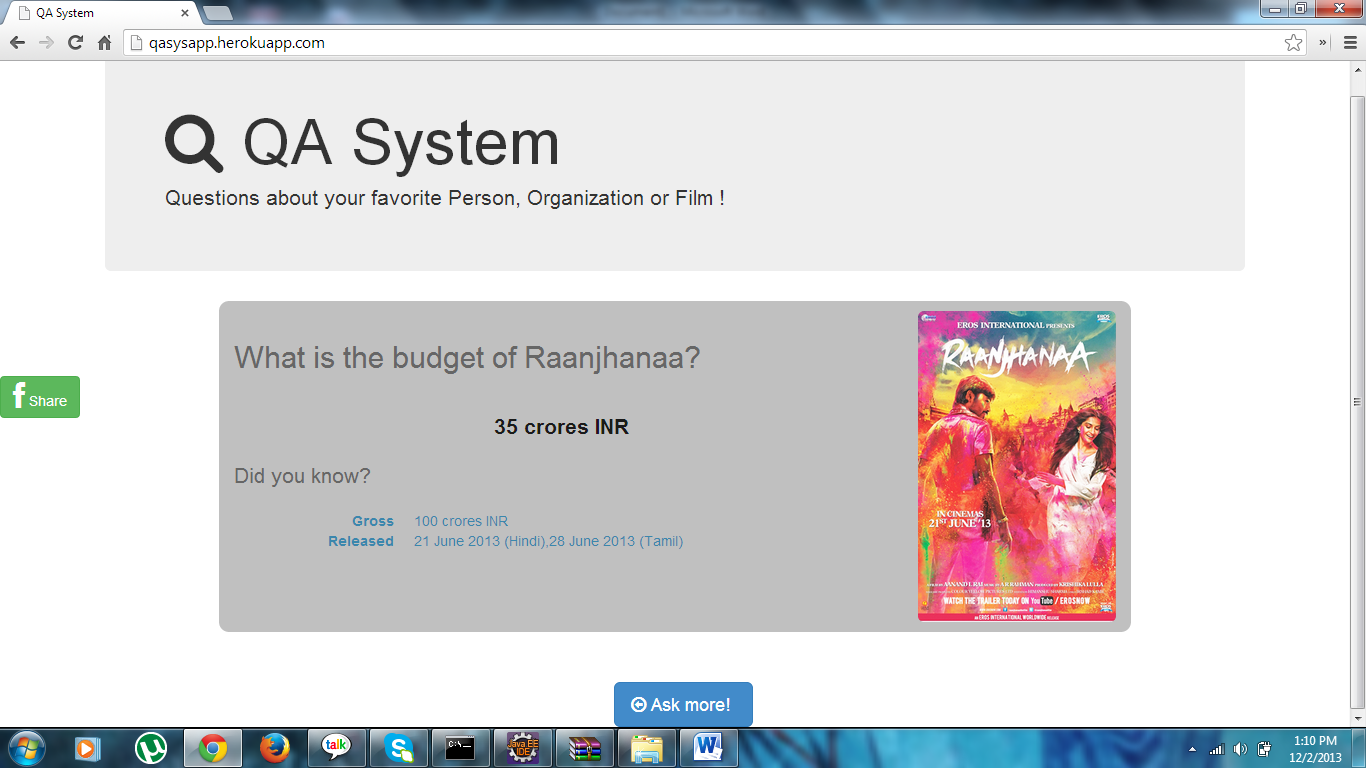




**Auto Suggest:**

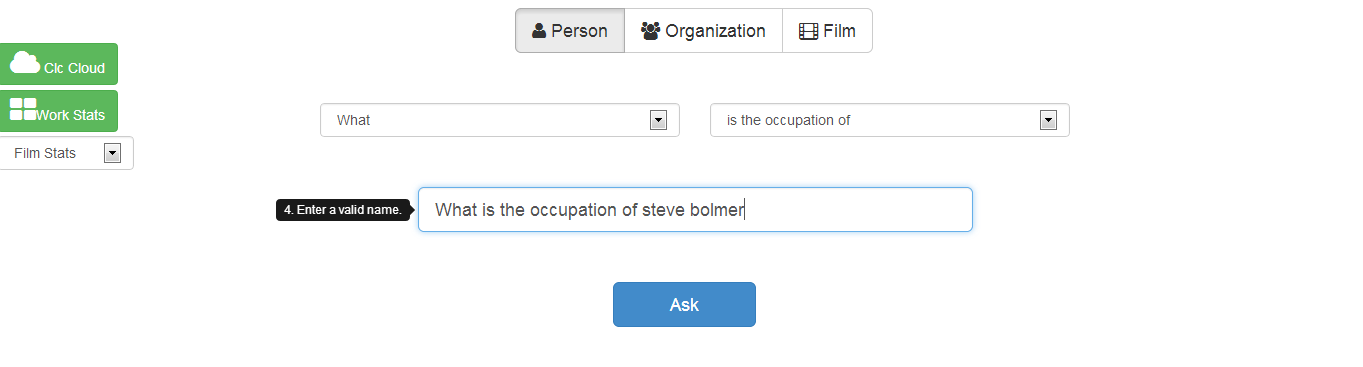


**Image Retrieval:**

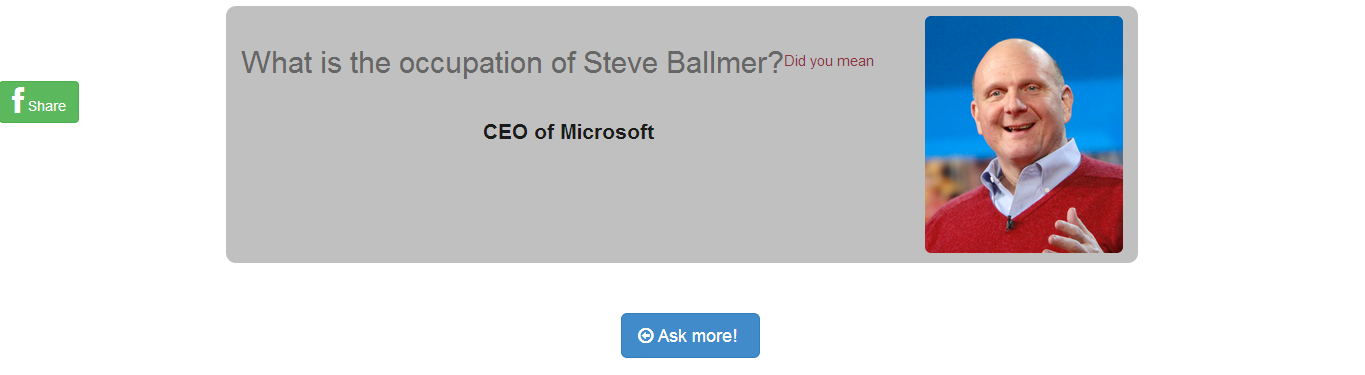


**Automatic Spell Correction:**

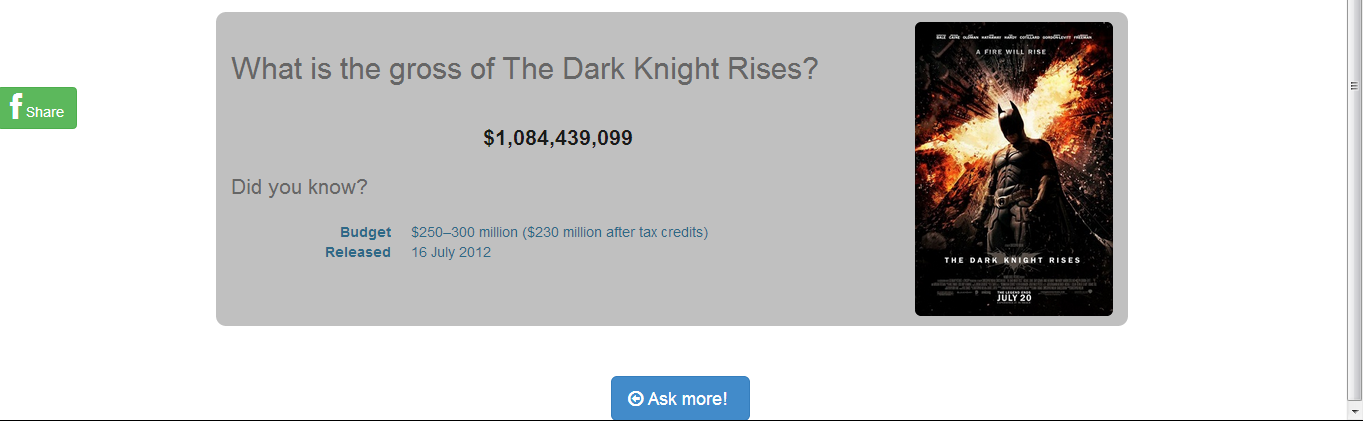
**Spelling Mistake:**



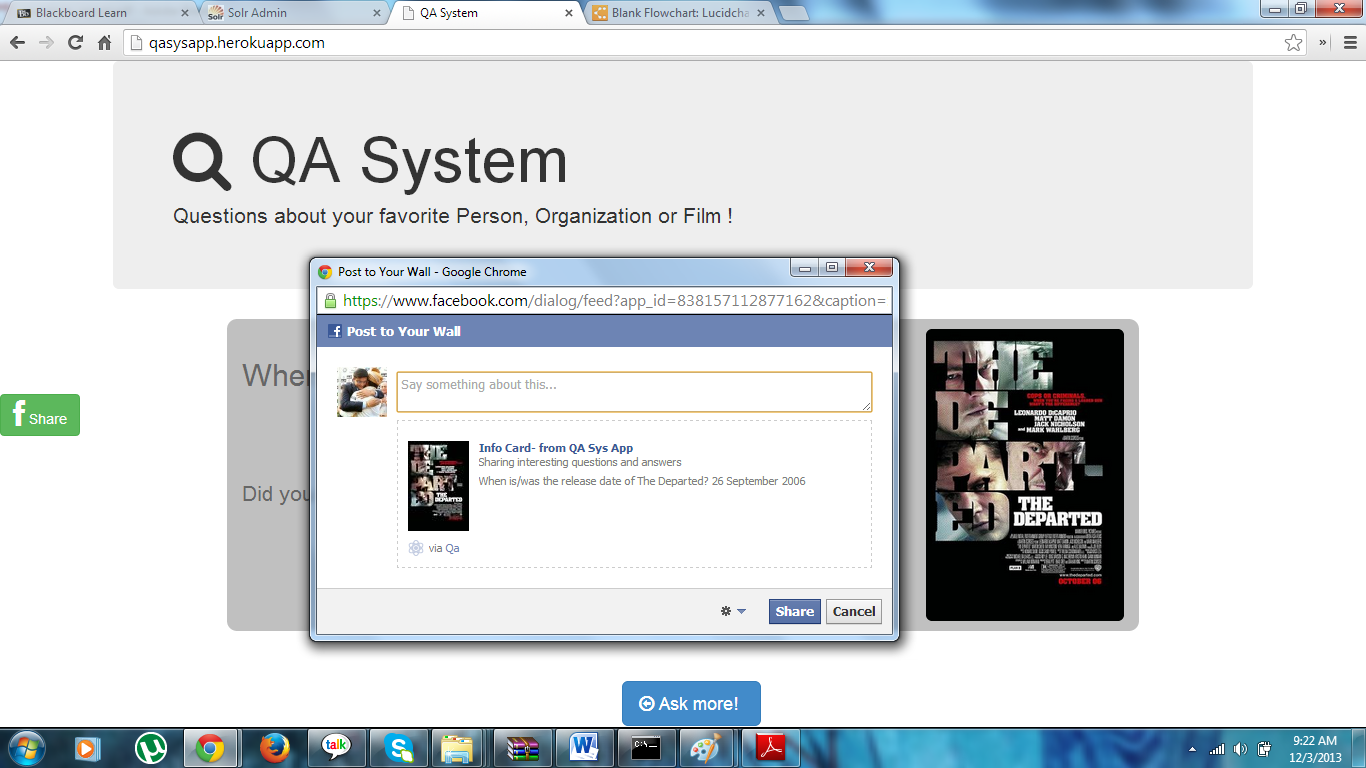
**Corrected Version:**



**Related Trivia that varies depending upon the query:**



**Option to share on Facebook:**



**FUTURE SCOPE:**

* The system could be augmented to show the birthplace and residence of a personality on a map by using the latitude and longitude coordinates.
* The auto suggestion feature for current personalities could be ranked based on the number of followers they have on twitter. This would serve as a measure of their popularity.
* Natural language Processing (NLP) can be implemented in order to support free text queries.
* The system could be enhanced to support speech recognition based responses.
* The large amount of data available could be rendered more effectively on the front end.
* As of now we have indexed around 10000 documents that can be categorized broadly into three topics. The system is scalable and more topics and its related documents can be included to better the user experience.

**INDIVIDUAL MEMBER CONTRIBUTION:**

**User Interface:**

* *Aswin Bharadwaj Ramesh*
* *Deepak Ravishankar Ramkumar*

**Solr Configuration:**

* *Vivekanandh Vel Rathinam*

**Parsing of data:**

* *Aravindhan Thanigachalam*