RCP database and search tool

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Pratik Indravadan Bhatt

Fall 2015

**SAN DIEGO STATE UNIVERSITY**

The Undersigned Faculty Committee Approves the

Thesis of Pratik Indravadan Bhatt:

RCP Database and Search Tool

Carl Eckberg, Chair

Department of Computer Science

Alan Riggins

Department of Computer Science

Timothy Dunster

Department of Mathematics and Statistics

Approval Date

Copyright © 2015

by

Pratik Indravadan Bhatt

All Rights Reserved

dedication

This work is dedicated to my family, friends and San Diego State University faculties for their incredible support, motivation and love. Specially, I am thankful to Professor Carl Eckberg for his continuous feedback and guidance.

ABSTRACT OF THE Thesis

RCP Database and Search Tool

by

Pratik Indravadan Bhatt

Master of Science in Computer Science

San Diego State University, 2015

People in this world who have an interest in reading news and stories, more often face difficulty in reading particular story events and milestones in chronological order. And with that, if someone is more interested into the political news and stories, they know how difficult it is to obtain an overall idea of the actual story events that already happened and follow that story, and hard to find arguments for both sides of an issue.

RCP (Real Clear Politics) is a website that greatly helps with this problem.

This tool is structured and designed to crawl and store RCP data, and at the same time provide functionality to the users, where they can search for political stories in chronological order. The user interface allows you to filter stories based on author name, publishing agency and published date.

All these data are stored in a database and collected from realclearpolitics.com by crawling and reading their RSS feed on a daily basis. The application uses full advantage of responsive layout, where the user can open and use this web tool on their tablet, phone and computer.

This paper provides the ground level concept of a creating and developing user-friendly news search engine, which can be extended to a larger scale, adding more news sources to crawl and expose stored data to users.

TABLE OF CONTENTS

PAGE

ABSTRACT v

LIST OF TABLES ix

LIST OF FIGURES x

ACKNOWLEDGEMENTS xi

CHAPTER

1. INTRODUCTION 1

1.1 Overview 1

1.2 RCP Overview 2

1.3 Motivation 2

1.4 Research Objectives 2

1.5 Tool Structure Overview 2

1.5.1 Crawler 3

1.5.2 Web Service and Database 3

1.5.3 Web Search Tool 3

1.6 Summary of Upcoming Chapters 3

1. TECHNOLOGIES 5

2.1 Python 5

2.1.1 Beautiful Soup 7

2.1.2 Feed Parser 7

2.2 MySQL 8

2.2.1 MySQL For Python 10

2.2.2 Node-MySQL 10

2.3 Node.js 11

2.3.1 NPM 13

2.3.1.1 Express 14

2.3.1.2 Gulp 14

2.3.1.3 Nodemon 14

2.4 Angular.js 14

2.5 jQuery 15

2.6 Bootstrap 16

1. APPLICATION DESIGN AND IMPLEMENTATION 17

3.1 Database 17

3.1.1 Author 17

3.1.2 Agency 18

3.1.3 Item 18

3.2 Crawler and Feed Parser 18

3.2.1 Data Classes 19

3.2.1.1 Author.py 19

3.2.1.2 Agency.py 20

3.2.1.3 Story.py 20

3.2.2 Library Classes 21

3.2.2.1 AuthorLib.py 21

3.2.2.2 AgencyLib.py 22

3.2.2.3 StoryLib.py 23

3.2.2.4 Common.py 26

3.2.3 Crawler 27

3.2.4 Feed Parser 29

3.3 Web Service 31

3.4 Web Search Tool 32

3.4.1 Search by Title 34

3.4.2 Search by Author 34

3.4.3 Search by Agency 35

1. USAGE WORKFLOW 36

4.1 Code Development Setup (IDE) 36

4.1.1 PyCharm 36

4.1.2 Brackets 37

4.1.3 MySQL Workbench 37

4.2 Application Workflow 37

1. CONCLUSION AND FUTURE ENHANCEMENT 39

REFERENCES 40

list of tables

PAGE

Table 3.1. Structure of Item Table 18

list of figures

PAGE

Figure 2.1. Simple code showing string manipulation in python. 6

Figure 2.2. Sample code explaining how to use feed parser library. 8

Figure 2.3 Flowchart explaining structure of data stored and accessed in MySQL [7]. 9

Figure 2.4 Explains how Node,js works [10]. 12

Figure 2.5. Core features of Angular.js [14]. 16

Figure 3.1. Database diagram of RCP database and search tool. 17

Figure 3.2. Structure of web crawler and feed parser 19

Figure 3.3. Date range selection to filter stories. 32

Figure 3.4. Show the details and description of story. 33

Figure 3.5. Explain the importance of number of views. 33

Figure 3.6. Show how user can search the story using different filters. 34

Figure 3.7. Example of search by title. 34

Figure 3.8. Example of search by author. 35

Figure 3.9. Example of search by agency. 35

Figure 4.1. Flowchart of project. 38

acknowledgements

I gratefully acknowledge Professor Carl Eckberg for his constant support, guidance and encouragement during my program of study and my thesis work. Taking this opportunity, I express my gratitude and sincere thanks to Professor Alan Riggins and Professor Timothy Dunster for their feedback, guidance and their presence on my thesis committee. I am very thankful to my parents, family and friends for their love and support.

CHAPTER 1

INTRODUCTION

# 1.1 Overview

The web provided a new way to present information on the Internet and that brought the modern challenges of exposing and sharing that information. Seeing this problem, many computer scientists and big companies such as Google, Yahoo, Microsoft started working on an information retrieval algorithm known as a search engine that will expose links (content or information located) to users [1]. As years passed, they figured out ways to improve these algorithms from preliminary human maintained indexing of famous and popular subjects on the web to automatic indexing and crawlers for storing information that is available on the web [1].

Similarly, the main vision and idea behind the development of this tool is to provide a simplified and user-friendly tool, which has deep search functionality for the users of realclearpolitics.com. All the stories are exposed and accessible to the user through a powerful search tool, that has the capability of searching for stories through keywords in the story title, author name, agency name and story published date. Web tool also provides quick access to the latest stories and past stories using a date range filter such as:

* TODAY: display present day stories.
* YESTERDAY: display yesterday’s stories.
* WEEK: display past 7 days stories starting from the present day.
* MONTH: display past 30 days stories starting from the present day.

One can also choose a range of dates.

One current scenario in which this tool can be helpful, is tracking down the events and news related to 2016 presidential campaigns of Mr. Donald Trump in chronological order.

Another good example to see the power of this search tool is tracking down all events, news, conflicts and controversies related to the Afghanistan War in the past couple of years in chronological order.

# 1.2 RCP Overview

RealClearPolitics is the trusted and non-bias news publishing website, where they attempt to elevate the level of journalism by providing rich content to its users. It was founded in 2000 by two news geeks, Tom Bevan and John McIntyre in their Chicago apartment with a passion for publishing most interesting political stories of the day using the power of the Internet and now they have staff who publishes original content, video and poll.

According to statistics from RealClearPolitics, average monthly unique visitor on the website is 5.9 million and each of them spent around 10 min on the website. It can be considered as the most viewed political website in the USA.

# 1.3 Motivation

In the past 4 years, working as a software developer, I have mostly worked in web development. My role during all these years was to build small as well as large scale web application and web services. I learned a lot, from new technologies in web programming, to how to design large scale applications and working with databases, so assuming all these into consideration I decided to do my thesis work in web and data mining.

# 1.4 Research Objectives

Developing a search engine for as news publishing website brings a lot of challenges. We need to have a fast crawling tool that can collect data from a website, a tool that can parse daily feed provided by website, storing data and presenting that in a nice readable format for great user experience [1]. Maintaining all these are very difficult, as we all know the web is growing fast. Another big challenge we face is to maintain the data and indexing of all that data on a large scale for fast searching speeds and other improvements.

# 1.5 Tool Structure Overview

This tool has been designed and developed using the latest web technologies, a database and object oriented programming. The three main sections of this tool are:

* Crawler
* Web Services and Database
* Web Search Tool

## 1.5.1 Crawler

The crawler is developed in python 2.7, which uses three libraries for parsing HTML raw data and storing that data in the database.

* Beautiful Soup: http://www.crummy.com/software/BeautifulSoup/
* Feed Parser: https://pypi.python.org/pypi/feedparser
* Mysql: https://dev.mysql.com/downloads/connector/python/2.1.html

## 1.5.2 Web Service and Database

Web Service is developed in node.js, which uses three node packages for fetching information from databases and defining urls (routing path), which is publicly exposed and utilize in the web tool. MySQL has been used for storing data in the backend.

* Express: https://github.com/strongloop/express
* Gulp: https://github.com/gulpjs/gulp
* Node-Mysql: https://github.com/felixge/node-mysql

## 1.5.3 Web Search Tool

Web Search Tool is drawn up and developed using HTML, bootstrap, angular.js and jQuery. The core functionality of the web tool is to utilize exposed web services and present data in a very user-friendly way.

* Bootstrap: http://getbootstrap.com/
* Angular.js: https://angularjs.org/
* jQuery: https://jquery.com/

# 1.6 Summary of Upcoming Chapters

* Chapter 1  
  Explains the need and overview of this application, comparing all the details with real world examples.
* Chapter 2  
  Explains all the technologies used to develop this application such as Python, Anglar.js, Node.js, etc.
* Chapter 3  
  Explains the application design, architecture and use case.
* Chapter 4  
  Explains the application workflow and different tool used to write code.
* Chapter 5  
  Explains about the future enhancements and vision for this tool.

CHAPTER 2

TECHNOLOGIES

# 2.1 Python

There are a number of reasons for considering python over other languages and it’s not hard to understand why I choose Python. To begin with, Python is an open source programming language and available under the GNU General Public License (GPL). Performance and stability of python is more intact compared to other languages and that makes this language one of the powerful, flexible and useful programming language among the programmers. It is considered to be an interpreted, interactive and high-level object oriented programming language. In general, python is well documented and very comfortable for any beginner to begin with. There are a lot of useful resources, libraries and communities accessible for the developers, which will be helpful to developers over the path of a project or development.

Core features,

* Interpreted: It’s a scripting language so you don’t need a compiler to compile your code before executing the program. It processes all its code at runtime.
* Interactive: We can directly interact with a python interpreter by writing code in IDLE (default python IDE).
* Object Oriented: Python uses the powerful OOPS concepts of inheritance, abstraction and encapsulation within its objects.

One of the nice features of python is its portability; a tool developed in python can be ported to any platforms without making any alterations to its code and libraries. Python programs can be extended and a tool developed in python can be used and called from any other languages like C++ or JAVA [2]. It’s effortless to practice, learn and read, have wide collection of different libraries, provides effective support for working with files and databases, scalable and also possesses the ability of GUI programming. I have worked with different languages such as C#, Java, Perl, Python, C, C++ over the years and I can say each language has its own power. Choosing python over other languages for my crawler is based on the power of python in string processing. As we all know it is reckoned as one of the most powerful scripting languages and is utilized worldwide for string processing in Bioinformatics for DNA sequencing, used by Google for query parsing and creating automated bots and crawlers to collect information from the web.

Python is the better choice if you are working with textual data and processing. Like most programming languages, python has the same basic type of string variable, but over this it allocates memory to hold strings; hence the developer does not have to think about memory allocation. Moreover python has some extraordinary “string-handling” features that you will not find in other tools. In Python, strings are immutable. Python also provides greater support for working with files.

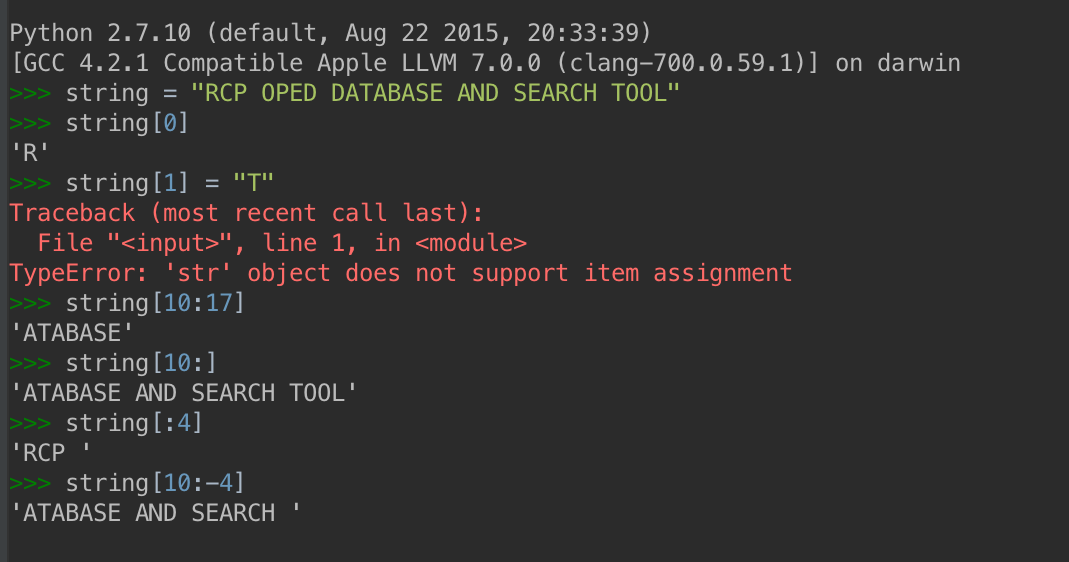


Figure 2.1. Simple code showing string manipulation in python.

Paid and open source Python IDE’s are available for a developer such as PyCharm, Sublime Text 3, Notepad++, Eclipse, VIM, IDLE etc. I have used Python 2.7 and Pycharm for the development as it provides great support to the developer with auto code completion, debugging and GIT integration. The latest version of python available for development is 3.5 and one can download python from its official website by selecting a proper platform of installation. Python can be installed using command line arguments or using the installer package, which is also available for download.

## 2.1.1 Beautiful Soup

Beautiful Soup is an open source python library, which is designed for pulling data out of HTML and XML and provides a quick solution in a project like web crawler. Leonard Richardson wrote this library in 2004 and since then it has helped in saving the time of developers and programmers around the globe. There are numerous open source libraries available, which can be used to parse HTML, but Beautiful Soup provides better support in HTML parsing, fetching data from HTML and even updating HTML tags at runtime.

It offers better functionality and simple methods for traversing, searching and updating an HTML tree structure from which you can extract all the information you need. And all this doesn’t take a hundred of lines of code in your application [3]. If you are working with Unicode characters, you will know how hard it is to deal with UTF encoding and decoding. Beautiful Soup automatically manages this process for you, unless the HTML document has not defined or it can’t detect it. It provides flexibility and substantial speed improvement compared to other python libraries such as lxml and html5lib [3].

It’s easy to download and install using the command line “pip install beautifulsoup4”. Packages are available for different Linux systems such as Debian, Ubuntu and Fedora. It works effectively with both recent versions of python i.e. 2.7 and 3.5 and it’s available under the MIT license.

## 2.1.2 Feed Parser

Most of the news agency and data publishing websites have their own RSS feeds, which expose the data published by those websites to a third party user. And all these RSS feeds, which have XML format, are used by mobile applications that assist the user to read all different stories from different sources in one place. Different web tools and software’s for collecting information on a daily basis use these feeds.

The feed Parser library, which is open source, provides powerful functionality of parsing this XML content and fetching data out of it. It can handle different RSS version RSS 0.90, Netscape 0.91, Userland RSS 0.91, RSS 0.92, RSS 0.93, RSS 9.94, RSS 1.0, RSS 2.0, Atom 0.3 and Atom 1.0 [4]. It can’t be used as standalone application for parsing an RSS feed; one needss to integrate this module within some python program.

It takes a single argument as input parameter and that is, the url of the RSS feed.

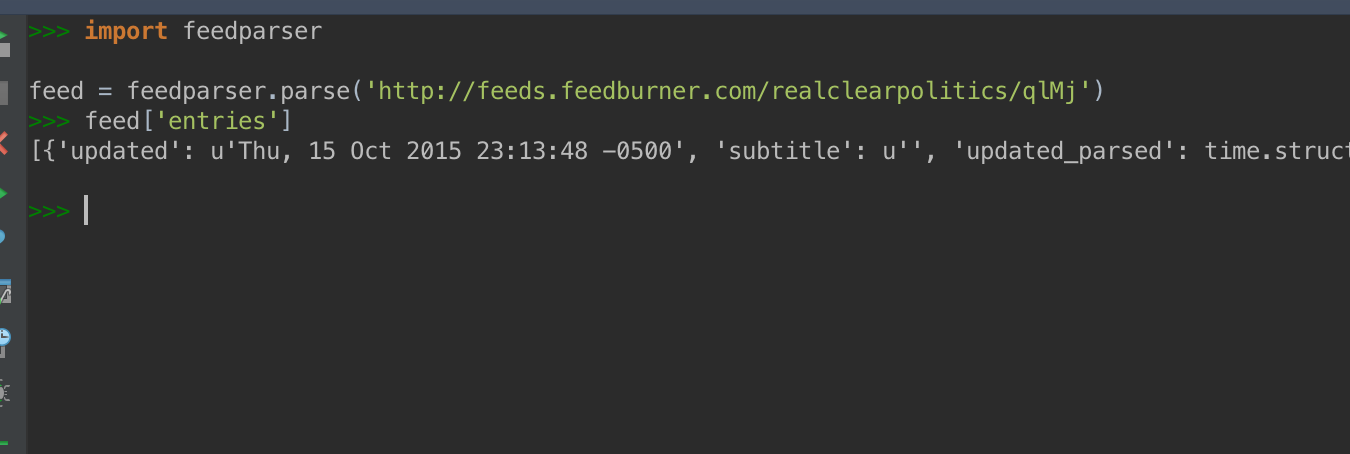
****

Figure 2.2. Sample code explaining how to use feed parser library.

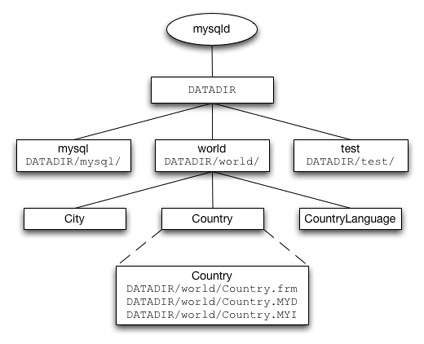
# 2.2 MySQL

“RDBMS Short for Relational Database Management System, software that stores data within tables, columns, and rows and supports relationships between tables. Examples include MySQL, Oracle, Informix, and Microsoft SQL Server” [5, p. 657].

MySQL is a Relational Database Management System (RDBMS), which is available freely. MySQL package is developed and supported by Oracle Corporation [6]. There are dozens of other RDBMS databases, available in the market and all of those use Structured Query Language (SQL). SQL is the standard language used to communicate with databases in all database management systems.

As we all know how important the data are, it's tough and challenging task a for developer and database management systems to provide a perfect solution for storing and accessing data in the most efficient way. Here comes MySQL, which is designed and developed to provide excellent performance, reliability and usability and since computers are better with handling data, database management systems came into the picture [6].

* Relational:   
  Relational database works on the concept of divide and conquer; it stores all closely related information in separate tables rather than keeping all this information at one place [6]. The database is very well structured and organized at ground level for speed improvement. MySQL has its own objects such as tables, views, columns, keys, trigger etc. that provide great hands on power and flexibility to its users [6]. These well-defined objects and rules help to enhance the power of MySQL by providing exceptional functionality against redundancy, inconsistency atomicity and isolation.
* Open Source:  
  MySQL is available under the GNU General Public License (GPL), which means anyone can use and modify this software without paying a single penny. But if you desire to use MySQL commercially than one can buy the license and embed MySQL code for commercial use [6].
* Speed, Scalability and Ease of use:  
  A dedicated server, desktop, or even your laptop can run MySQL comfortably with all your applications running on the side. “It was initially projected to handle large-scale databases and if you are using a dedicated server for MySQL then it offers powerful configuration capabilities such as taking advantage of memory, I/O operation and CPU utilization” [6]. MySQL has been serving the community for several years and it can also be used to create a cluster, which provides robust functionality and usefulness to its users who want to access the database through the Internet.
* Portability:  
  MySQL server can be set up on different platforms such as Linux, MacOS and windows and run on a wide range of architectures including x86, X64, PowerPC etc.



**Figure 2.3 Flowchart explaining structure of data stored and accessed in MySQL [7].**

## 2.2.1 MySQL For Python

Python provides a wide range of libraries and connectors to work together with the database. MySQL Connector for Python is developed in pure Python and has no dependency except for the Python Standard Library [8]. It allows any python program to connect to a MySQL database using the API. It supports all the versions of python from 2.6, 2.7, 3.1 and later.

Here is the list of all features, stated almost verbatim, that the MySQL connector provides to its users [8]:

* Support for almost all features provided by MySQL Servers up to and including MySQL Server version 5.7 [8].
* Converting parameter values back and forth between Python and MySQL data types, for example Python **datetime** and MySQL **DATETIME**. You can turn automatic conversion on for convenience, or off for optimal performance [8].
* All MySQL extensions to standard SQL syntax [8].
* Protocol compression, which enables compressing the data stream between the client and server [8].
* Connections using TCP/IP sockets and on Unix using Unix sockets [8].
* Secure TCP/IP connections using SSL [8].
* Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library [8].

## 2.2.2 Node-MySQL

We can install the MySQL package for a Node.js through npm, which is an official package manager for all Node.js packages. This can be considered as a MySQL driver that will be used by Node.js, which facilitates the programmer to connect with databases on the server. It is available under open source MIT license and full source code can be accessed on Github. This package is totally written in JavaScript, supported by Oracle developers and sponsored by a lot of companies. Its properties include

* Fastest among all packages available for Node.js and MySQL
* Provides support for prepared statements, SSL and compression.
* Well documented and tested with large number of tests.

# 2.3 Node.js

Modern day web development is extremely different from what earlier it used to be. The rise of JavaScript within web technologies surprised everyone with its power by running on the client side and server side together, which was hard to imagine a few years ago.

The core architecture of Node.js uses the power of Chrome’s V8 JavaScript engine that is supported by Google. Ryan Dahl, the creator of Node.js, primarily wrote the whole platform in C++ so it is fast. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient [9]. “Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world” [9]. The aim was to create an architecture that can provide real time websites a push and notify its client, something like Gmail. Over the period from when the web was introduced to the world, finally we can develop a web application having two-way connections, which mean both client, and server can communicate with each other, allowing them to share data freely. In one sentence: “Node.js shines in real-time web applications employing push technology over websockets” [10].

Before Node.js was introduced we all had the option of using Java Applets and Flash, which used to provide the same functionality with dependency and sandbox environment that uses the web as a transport protocol to connect with a client [10]. So now with the introduction of Node.js, it plays an important role for many big companies who take full advantage of its power and benefits.

Advantages:

* Node.js is much faster than other scripting languages like Ruby, Python or Perl. The reason behind this is Node.js uses JavaScript, which is incredibly fast on Chrome’s V8 JavaScript engine.
* Single thread architecture provide best support in handling thousands of concurrent connections with minimal overhead.
* The biggest advantage is people know how to use JavaScript as they have been using since long time on the browser that handles all client side event. It is arguably the most popular programming language.
* JavaScript being common language for both client side and server side reduces the impedance mismatch between the two programming environments. Both sides can communicate via JSON that work the same on both sides of the equation. Duplicate form validation code can be shared between server and client, etc.

Examples where Node.js should be used are Chat Application, API to access database, queued input and data processing applications, etc.

How does it work?

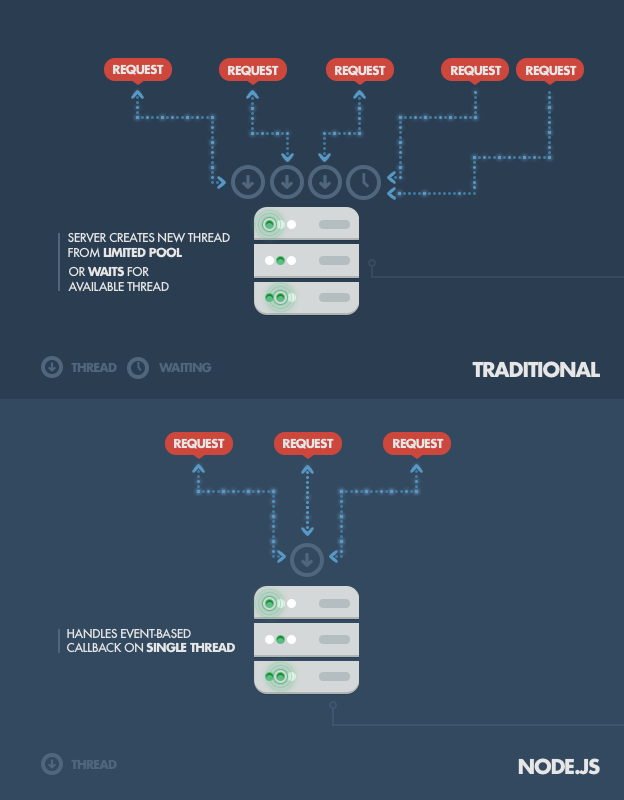


Figure 2.4 Explains how Node,js works [10].

As we all know, Node.js uses a non-blocking and event-driven approach to keeping it lightweight; that doesn’t mean it will dominate the entire web development but for sure it is filling out a particular gap in web development. Node.js is not designed to do large computations or processing, but it is more meant for building fast, scalable network applications. The reason behind this is that it holds the capability of handling multiple connections with high throughput and that is very important for scalability.

Unlike traditional web servers, which create a new thread for processing each web request, Node.js takes advantage of only a single thread by using non-blocking calls and supporting tons of concurrent connection at the same time. And this is the reason why Node.js server cannot be used for high computational programming, where the first request can block all other requests waiting to be executed in a single thread. Secondly, developers need to be really careful not to allow an exception bubbling up to the core (topmost) Node.js event loop, which will cause the Node.js instance to terminate (effectively crashing the program) [10].

## 2.3.1 NPM

While discussing about Node.js and importance of Node.js in current scenarios, one should also focus on package management provided by the community. Node Package Manager is the tool known as npm, which provide functionality for all developers to publish their packages, which can be installed into Node.js environment. NPM is the same as Microsoft nugget and Ruby Gems where all packages are publicly available via the online repository.

* “It is an online repository for the publishing of open-source Node.js projects” [11].
* “It is a command-line utility for interacting with said repository that aids in package installation, version management, and dependency management” [11].

Anyone can publish a new package in NPM and it will be listed under the official repository. Full list of all packages published can be found on https://www.npmjs.com/ and CLI tool can be used to install that package in your project.

Here are some of the famous Node.js packages, some of them used in this thesis:

* Express
* Gulp
* Socket.io
* Jade
* Node-MySQL

### 2.3.1.1 Express

Express is an open source NPM package available under MIT license, which was originally written by T J Holowaychuk and currently maintained by Douglas Christopher Wilson. It provides important, robust tool for HTTP servers for all single page applications, web sties and web APIs.

Features [12]:

* Robust routing
* Focus on high performance
* Super-high test coverage
* HTTP helpers (redirection, caching, etc)
* View system supporting 14+ template engines
* Content negotiation
* Executable for generating applications quickly

### 2.3.1.2 Gulp

Gulp is a build system where you can define automated task such as minifying JavaScript, preprocessing CSS file or reloading the browser. Gulp is built on Node.js and written in JavaScript, so one has to use JavaScript for defining a gulp file in the project.

### 2.3.1.3 Nodemon

Nodemon is a plugin that can be used with a gulp; it will watch the entire directory in which it was installed and started. As soon as it will detect any change in the file, the task of nodemon is to restart the server automatically.

# 2.4 Angular.js

There are a number of JavaScript structural frameworks available for creating dynamic web pages, single page applications and focusing on data binding in web. The main reason to select Angular.js over others is it’s an open source framework available under the MIT license developed in 2009 by Misko Hevery and Adam Abrons and now maintained by Google. It offers a complete package, compare to other frameworks where we need to integrate the two frameworks in order to get functionality similar to Angular.js e.g. knockout.js and Sammy.js.

“AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology” [13].

Angular uses the MVC (Model View Controller) paradigm for creating Rich Internet Applications (RIA), that helps developers to keep their code clean and optimized on the client side. It provides cross browser compatibility by handling JavaScript code that is suitable for each individual browser.

Core Features:

* Data-binding: automatic synchronization of data between implementation and view.
* Scope: object that helps controller and view to interact with each other.
* Controller: functions or implementation in a scope.
* Services: comes with different built in services that help in Ajax request ($http), encoding and decoding of Unicode characters ($sce) etc.
* Directives: template that can be used with HTML for DOM manipulation, creating custom tag and widgets. It has built in directives.
* Routing: helps in switching view.
* Dependency Injection

Angular.js depends on three main components:

* ng-app: defines that this is the Angular.js application.
* ng-model: binds the value of Angular.js with view i.e. HTML.
* ng-bind: bind data within HTML tags.

# 2.5 jQuery

jQuery is a fast, small and powerful library that is built on top of JavaScript. It makes things very simple like HTML document traversal and manipulation, event handling, animation, and Ajax by providing easy-to-use API that works across a multiple browsers.

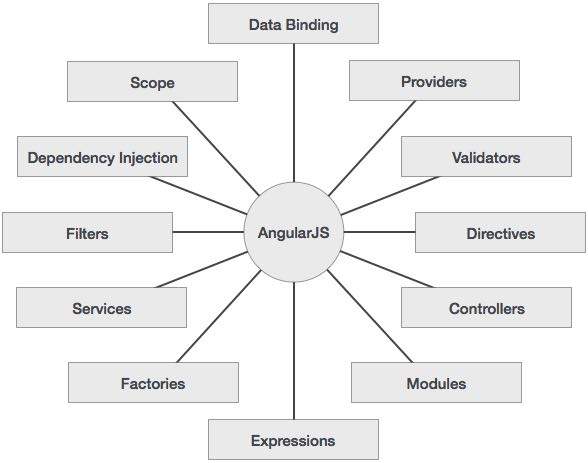


Figure 2.5. Core features of Angular.js [14].

Advantages:

* Easy to use
* Powerful JavaScript Library
* Open source
* Well documented and good community
* Ajax Support
* Handle DOM manipulation and events

# 2.6 Bootstrap

Originally created by a designer and a developer at Twitter, Bootstrap has become one of the most popular front-end frameworks and open source projects in the world. It is very useful with its built-in CSS styles, icons and grid structure pattern while designing any web application. It provides support for responsive layout so any application designed with bootstrap will be compatible with all devices (mobiles, laptop, tablets, etc.).

CHAPTER 3

APPLICATION DESIGN AND IMPLEMENTATION

# 3.1 Database

The database is the essential component of any application. So in the beginning phase of software design, we always built database diagram. Database diagram is used to describe database architecture and how the information will be stored in a database. Below is the database diagram of the application:

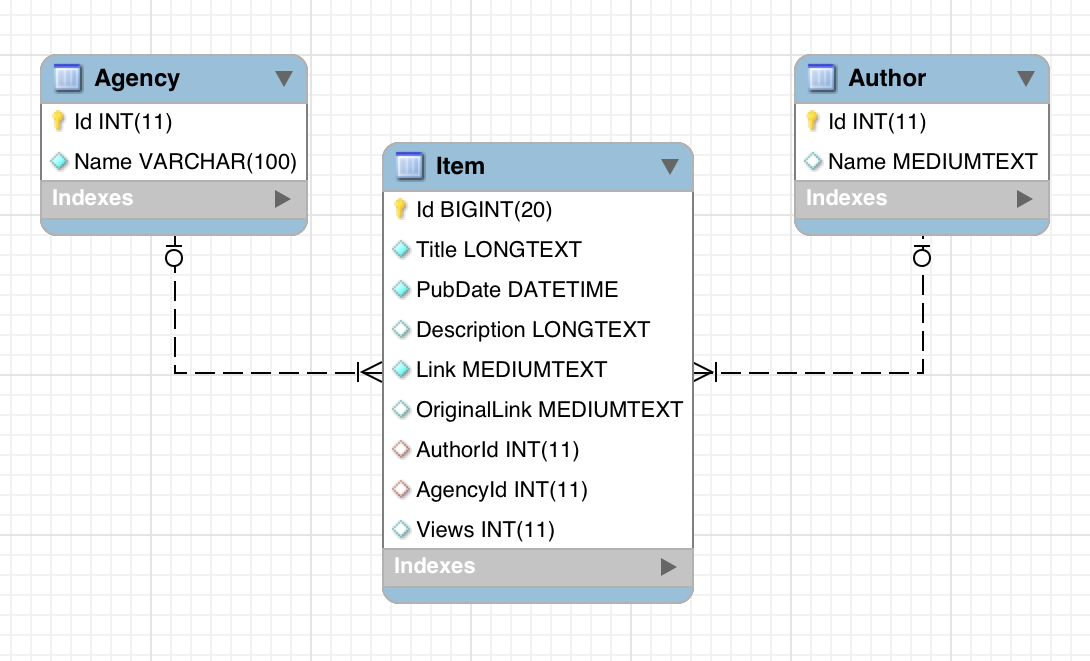


Figure 3.1. Database diagram of RCP database and search tool.

## 3.1.1 Author

Author table stores the name of a person who actually wrote the article. The NAME field in the table is of VARCHAR type. Considering the name of the author will be text and not more than 100 characters set its size to 100. ‘Id’ field is PRIMARY KEY, which will be a unique value in the table.

## 3.1.2 Agency

Agency table stores the name of the agency that published the article. Considering the name of that agency may be long enough, NAME field in the table is of type MEDIUMTEXT. That can store more characters compared to VARCHAR. ‘Id’ field is PRIMARY KEY, which will be a unique value in the table.

## 3.1.3 Item

Table 3.1. Structure of Item Table

|  |  |  |
| --- | --- | --- |
| **FIELD** | **TYPE** | **DESCRIPTION** |
| Title | LONGTEXT | Stores the title of a story. |
| PubDate | DATETIME | Stores the article published date. |
| Description | LONGTEXT | Stores the description of article, which can be of few lines. |
| Link | MEDIUMTEXT | Stores the link of RCP website, so that user can open article published in RealClearPolitics from search tool. |
| OriginalLink | MEDIUMTEXT | Stores the original link of the source website, so that user can open original article published by any respective from search tool. |
| AuthorId | INT(11) | This is FOREIGN KEY attached with Id field of author table. |
| AgencyId | INT(11) | This is FOREIGN KEY attached with Id field of agency table. |
| Views | INT(11) | Maintain the number of views; this field will be incremented by one, when any user opens the story to read. |

# 3.2 Crawler and Feed Parser

Crawler and Feed Parser use the same structure (code) of the project to insert story, author and agency into the database. The section in which they differ is the way they both parse data for stories. Crawler start crawling website since 2008 and continue till Oct 2015, by opening the link to RealClearPolitics that is in a specific format:

http://www.realclearpolitics.com/**YEAR**/**MONTH**/**DAY**

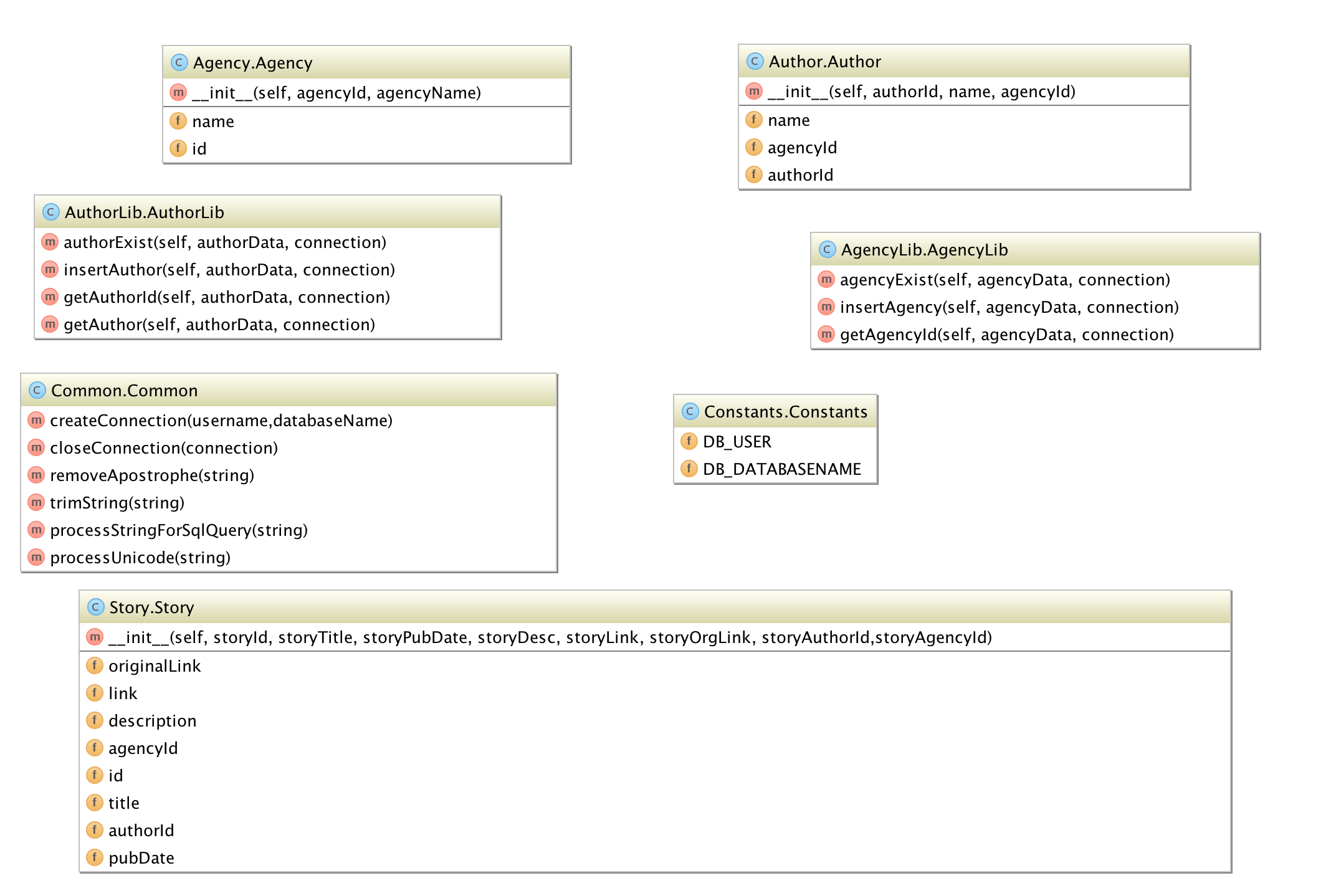


Figure 3.2. Structure of web crawler and feed parser

Feed Parser uses the fixed RSS feed link that is officially provided by RealClearPolitics. RSS feed links share data in XML format. Below is the RSS link from where data is retrieved on a daily basis:

http://feeds.feedburner.com/realclearpolitics/qlMj

## 3.2.1 Data Classes

### 3.2.1.1 Author.py

Defines the data fields that are related to author table. Always used to create an object of the Author.

\_\_author\_\_ **=** 'Pratik'  
**from** Common **import** Common  
  
**class Author:  
 def \_\_init\_\_**(self, authorId, name)**:** self.authorId **=** authorId  
 self.name **=** Common.processStringForSqlQuery(name)

### 3.2.1.2 Agency.py

Defines the data fields that are related to agency table. Always used to create an object of the Agency.

\_\_author\_\_ **=** 'Pratik'  
**from** Common **import** Common  
  
**class Agency:  
 def \_\_init\_\_**(self, agencyId, agencyName)**:** self.id **=** agencyId  
 self.name **=** Common.processStringForSqlQuery(agencyName)

### 3.2.1.3 Story.py

Defines the data fields that are related to Story table. Used to create an object of the Story.

\_\_author\_\_ **=** 'Pratik'  
**from** Common **import** Common  
  
**class Story:  
 def \_\_init\_\_**(self, storyId, storyTitle, storyPubDate, storyDesc, storyLink, storyOrgLink, storyAuthorId,storyAgencyId)**:** self.id **=** storyId  
 self.title **=** Common.processStringForSqlQuery(storyTitle)  
 self.pubDate **=** storyPubDate  
 self.description **=** Common.processStringForSqlQuery(storyDesc)  
 self.link **=** Common.processStringForSqlQuery(storyLink)  
 self.originalLink **=** Common.processStringForSqlQuery(storyOrgLink)  
 self.authorId **=** storyAuthorId  
 self.agencyId **=** storyAgencyId

## 3.2.2 Library Classes

### 3.2.2.1 AuthorLib.py

Defines all the function related to author such as inserting author into the database, checking whether author exists in the database, getting author data from database.

\_\_author\_\_ **=** 'Pratik'  
**import** sys  
  
*# contains all the function related to author***class AuthorLib:** *# check whether author exists in database or not* **def authorExist**(self, authorData, connection)**:  
 try:** cursor **=** connection.cursor()  
 select **=** "SELECT count(\*) from Author WHERE Name='{0}'".format(authorData.name)  
 cursor.execute(select)  
 counter **=** cursor.fetchone()  
 **if** counter[0] **==** 0**:  
 return** False  
 **else:  
 return** True  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : authorExist : {0}".format(e)  
  
 *# method to insert data into database* **def insertAuthor**(self, authorData, connection)**:  
 try:  
 if not** self.authorExist(authorData, connection)**:** cursor **=** connection.cursor()  
 insert **=** "INSERT INTO Author (Name) VALUES ('{0}')".format(authorData.name)  
 cursor.execute(insert)  
 connection.commit()  
 **return** self.getAuthorId(authorData,connection)  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : insertAuthor : {0}".format(e)  
  
 *# get author Id from database* **def getAuthorId**(self, authorData, connection)**:  
 try:** cursor **=** connection.cursor()  
 select **=** "SELECT Id from Author WHERE Name='{0}'".format(authorData.name)  
 cursor.execute(select)  
 **return** cursor.fetchone()[0]  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : getAuthorId : {0}".format(e)  
  
 *# get author object from database based on author name* **def getAuthor**(self, authorData, connection)**:  
 try:** cursor **=** connection.cursor()  
 select **=** "SELECT \* from Author WHERE Name='{0}'".format(authorData.name)  
 cursor.execute(select)  
 **return** cursor.fetchone()  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : getAuthor : {0}".format(e)

### 3.2.2.2 AgencyLib.py

Defines all the function related to an agency such as inserting agency into a database, fetching agency data from database and checking whether agency already exists in the table or not.

\_\_author\_\_ **=** 'Pratik'  
**import** sys  
  
**class AgencyLib:** *# check whether agency exists in database or not* **def agencyExist**(self, agencyData, connection)**:  
 try:** cursor **=** connection.cursor()  
 select **=** "SELECT count(\*) from Agency WHERE Name='{0}'".format(agencyData.name)  
 cursor.execute(select)  
 counter **=** cursor.fetchone()  
 **if** counter[0] **==** 0**:  
 return** False  
 **else:  
 return** True  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : agencyExist : {0}".format(e)  
  
 *# insert agency into database* **def insertAgency**(self, agencyData, connection)**:  
 try:  
 if not** self.agencyExist(agencyData, connection)**:** cursor **=** connection.cursor()  
 insert **=** "INSERT INTO Agency (Name) VALUES ('{0}')".format(agencyData.name)  
 cursor.execute(insert)  
 connection.commit()  
 **return** self.getAgencyId(agencyData, connection)  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : insertAgency : {0}".format(e)  
  
 *# get agency Id from database* **def getAgencyId**(self, agencyData, connection)**:  
 try:** cursor **=** connection.cursor()  
 select **=** "SELECT Id from Agency WHERE Name='{0}'".format(agencyData.name)  
 cursor.execute(select)  
 **return** cursor.fetchone()[0]  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : getAgencyId : {0}".format(e)

### 3.2.2.3 StoryLib.py

Defines all the function related to Story such as inserting story into a database, parsing HTML and XML to create story object, checking whether story exists into database or not.

\_\_author\_\_ **=** 'Pratik'  
**import** sys  
**from** bs4 **import** BeautifulSoup  
**import** urllib **as** url  
**from** Story **import** Story  
**from** HTMLParser **import** HTMLParser  
  
**class StoryLib:** *# parse the HTML or XML to get description and return story object* **def getStroy**(self, title, pubDate, link, authorId, agencyId)**:  
 try:** description **=** ""  
 originalLink **=** ""  
 rcpHandle **=** url.urlopen(link)  
 rcpPageData **=** rcpHandle.read()  
 rcpBeautyPage **=** BeautifulSoup(rcpPageData, "html.parser")  
  
 articleData **=** rcpBeautyPage.find(id**=**"article\_body")  
 **if not** articleData **==** None**:  
 for** article **in** articleData.find\_all('p')**:** description **=** description **+** (article.get\_text().encode("utf8")) **+** "<br />"  
  
 articleOuterLinkDiv **=** rcpBeautyPage.find(id**=**"article\_source\_link")  
 **if not** articleOuterLinkDiv **==** None**:  
 for** child **in** articleOuterLinkDiv.children**:  
 if** len(child) **>** 0**:** originalLink **=** (child.get('href'))  
 **else:** articleData **=** rcpBeautyPage.find(id**=**"alpha")  
 *# new design support* **if** articleData **==** None**:** articleData **=** rcpBeautyPage.find\_all("div", class\_**=**"entry-body-text")  
  
 **if** len(articleData) **>** 0**:  
 for** article **in** articleData[0].find\_all('p')**:** description **=** description **+** (article.get\_text().encode("utf8")) **+** "<br />"  
  
 articleOuterLinkDiv **=** rcpBeautyPage.find(id**=**"article\_source\_link")  
 **if not** articleOuterLinkDiv **==** None**:  
 for** child **in** articleOuterLinkDiv.children**:  
 if** len(child) **>** 0**:** originalLink **=** (child.get('href'))  
 **else:** articleOuterLinkDiv **=** rcpBeautyPage.find\_all("span", class\_**=**"more-link")  
 **if** len(articleOuterLinkDiv) **>** 0**:  
 for** linka **in** articleOuterLinkDiv[0].find\_all('a')**:** originalLink **=** linka.get('href')  
 **else:** articleData **=** rcpBeautyPage.find\_all("div", class\_**=**"article-body-text")  
  
 **if** len(articleData) **>** 0**:  
 for** article **in** articleData[0].find\_all('p')**:** description **=** description **+** (article.get\_text().encode("utf8")) **+** "<br />"  
  
 articleOuterLinkDiv **=** rcpBeautyPage.find(id**=**"article\_source\_link")  
 **if not** articleOuterLinkDiv **==** None**:  
 for** child **in** articleOuterLinkDiv.children**:  
 if** len(child) **>** 0**:** originalLink **=** (child.get('href'))  
 **else:** articleOuterLinkDiv **=** rcpBeautyPage.find\_all("span", class\_**=**"more-link")  
 **if** len(articleOuterLinkDiv) **>** 0**:  
 for** linka **in** articleOuterLinkDiv[0].find\_all('a')**:** originalLink **=** linka.get('href')  
  
 **else:** *# old design support* **if not** articleData **==** None**:  
 for** article **in** articleData.find\_all('p')**:** description **=** description **+** (article.get\_text().encode("utf8")) **+** "<br />"  
  
 articleOuterLinkDiv **=** rcpBeautyPage.find(id**=**"article\_source\_link")  
 **if not** articleOuterLinkDiv **==** None**:  
 for** child **in** articleOuterLinkDiv.children**:  
 if** len(child) **>** 0**:** originalLink **=** (child.get('href'))  
 **else:** articleOuterLinkDiv **=** rcpBeautyPage.find\_all("span", class\_**=**"more-link")  
 **if** len(articleOuterLinkDiv) **>** 0**:  
 for** linka **in** articleOuterLinkDiv[0].find\_all('a')**:** originalLink **=** linka[0].get('href')  
  
 parser **=** HTMLParser()  
 story **=** Story(0, title, pubDate, parser.unescape(description), link, originalLink, authorId, agencyId)  
 **return** story  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : getStroy : {0}".format(e)  
  
 *# check whether story exist or not* **def storyExist**(self, story, connection)**:  
 try:** cursor **=** connection.cursor()  
 select **=** "SELECT count(\*) from Item WHERE Title='{0}' AND PubDate='{1}' AND Link='{2}' AND OriginalLink='{3}'".format(  
 story.title, story.pubDate, story.link, story.originalLink)  
 cursor.execute(select)  
 counter **=** cursor.fetchone()  
  
 **if** counter[0] **==** 0**:  
 return** False  
 **else:  
 return** True  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : storyExist : {0}".format(e)  
  
 *# insert story into database* **def insertStory**(self, story, connection)**:  
 try:** cursor **=** connection.cursor()  
  
 **if not** self.storyExist(story, connection)**:** insert **=** "INSERT INTO Item (PubDate,Description,Link,OriginalLink,AuthorId,Title,AgencyId) VALUES ('{0}','{1}','{2}', '{3}',{4},'{5}','{6}')".format(  
 story.pubDate, story.description, story.link, story.originalLink, story.authorId, story.title,  
 story.agencyId)  
 cursor.execute(insert)  
 connection.commit()  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : insertStory : {0}".format(e)

### 3.2.2.4 Common.py

Defines all the common method that will be used in through the project. All the methods are defined in this class are static. The method includes creating and closing the database connection; remove special character and process string for SQL query.

\_\_author\_\_ **=** 'Pratik'  
**import** mysql.connector  
  
**class Common:  
  
 @**staticmethod  
 **def createConnection**(username,databaseName)**:  
 return** mysql.connector.connect(user**=**username, database**=**databaseName,buffered**=**True)  
  
 **@**staticmethod  
 **def closeConnection**(connection)**:** connection.close()  
  
 **@**staticmethod  
 **def removeApostrophe**(string)**:  
 return** string.replace("'", "''")  
  
 **@**staticmethod  
 **def trimString**(string)**:  
 return** string.strip()  
  
 **@**staticmethod  
 **def processStringForSqlQuery**(string)**:  
 return** Common.trimString(Common.processUnicode(Common.removeApostrophe(string)))  
  
 **@**staticmethod  
 **def processUnicode**(string)**:  
 return** string.replace("\r", " ").replace("\n", " ").replace("\t", '').replace("\"", "")

## 3.2.3 Crawler

Crawler is the function that uses the Beautiful Soup Python library to parse HTML retrieved from the links. These parse data are stored in the database using Library classes of Story (StoryLib.py), Author (AuthorLib.py) and Agency (AgencyLib.py). Years, number of months in each year and number of days in each month that need to be crawled are already defined in an array and dictionaries.

\_\_author\_\_ **=** 'Pratik'  
**import** sys  
**from** bs4 **import** BeautifulSoup  
**import** urllib **as** url  
**from** datetime **import** datetime  
**from** Common **import** Common  
**from** Constants **import** Constants  
**from** Agency **import** Agency  
**from** AgencyLib **import** AgencyLib  
**from** Author **import** Author  
**from** AuthorLib **import** AuthorLib  
**from** Story **import** Story  
**from** StoryLib **import** StoryLib  
  
years **=** [2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015]  
  
months **=** {2006**:** 12, 2007**:** 12, 2008**:** 12, 2009**:** 12, 2010**:** 12, 2011**:** 12, 2012**:** 12, 2013**:** 12, 2014**:** 12, 2015**:** 10}  
  
days **=** {1**:** 31, 2**:** 28, 3**:** 31, 4**:** 30, 5**:** 31, 6**:** 30, 7**:** 31, 8**:** 31, 9**:** 30, 10**:** 9, 11**:** 30, 12**:** 31}  
  
**def crawler**()**:  
 for** year **in** years**:** numOfMonths **=** months[year]  
 **for** month **in** range(1,numOfMonths **+**1)**:  
 if** month **==** 2**:  
 if** year **%** 4 **==** 0**:** numOfDays **=** 29  
 **else:** numOfDays **=** days[month]  
 **else:** numOfDays **=** days[month]  
 **for** day **in** range(1,numOfDays**+**1)**:** monthStr **=** "%02d" **%** month  
 dayStr **=** "%02d" **%** day  
  
 thisUrl **=** 'http://www.realclearpolitics.com/{0}/{1}/{2}'.format(year, monthStr, dayStr)  
 **print** thisUrl  
 handle **=** url.urlopen(thisUrl)  
  
 pageData **=** handle.read()  
 beautyPage **=** BeautifulSoup(pageData, "html.parser")  
 **try:** alphaData **=** beautyPage.find(id**=**"alpha")  
  
 pubDate **=** datetime(year,month,day)  
  
 content\_data **=** []  
 **if not** alphaData **==** None**:  
 for** content **in** alphaData.contents **:  
 if** str(content.encode("utf8")).startswith(' -') **or** str(content.encode("utf8")).startswith(' - ') **:** content\_data.append(content)  
  
 agencyLib **=** AgencyLib()  
 authorLib **=** AuthorLib()  
 storyLib **=** StoryLib()  
 connection **=** Common.createConnection(Constants.DB\_USER,Constants.DB\_DATABASENAME)  
  
 counter **=** 0  
 **for** a **in** alphaData.find\_all('a')**:  
 if** "http://" **in** a.get('href')**:  
 if not** counter **==** len(content\_data) **:** content **=** content\_data[counter]  
 authorDetails **=** content[content.index('-')**+**1**:**].split(',')  
 **if** len(authorDetails) **>** 1**:** agency **=** Agency(0,authorDetails[1].encode("utf8"))  
 agencyId **=** agencyLib.insertAgency(agency,connection)  
 author **=** Author(0, authorDetails[0].encode("utf8"))  
 authorId **=** authorLib.insertAuthor(author, connection)  
 **else:** agency **=** Agency(0,authorDetails[0].encode("utf8"))  
 agencyId **=** agencyLib.insertAgency(agency,connection)  
 author **=** Author(0, "")  
 authorId **=** authorLib.insertAuthor(author, connection)  
 **if** "http://www.realclearpolitics.com/" **in** a.get('href')**:** link **=** a.get('href')  
 title **=** a.get\_text().decode('utf8')  
 story **=** storyLib.getStroy(title,pubDate,link,authorId,agencyId)  
 storyLib.insertStory(story, connection)  
 counter **=** counter **+** 1  
 **else:** link **=**a.get('href')  
 originalLink **=** a.get('href')  
 title **=** a.get\_text().decode('utf8')  
  
 story **=** Story(0,title, pubDate, "", link, originalLink,authorId, agencyId)  
 storyLib.insertStory(story, connection)  
 counter **=** counter **+** 1  
  
 Common.closeConnection(connection)  
 **except:** e **=** sys.exc\_info()[0]  
 **print** "ERROR : crawler : {0}".format(e)  
  
**def main**()**:** crawler()  
  
**if** \_\_name\_\_ **==**'\_\_main\_\_'**:** main()

## 3.2.4 Feed Parser

Feed Parser class uses open source feedparser Python library to extract data from daily feed provided by RealClearPolitics. Feed from the website is in XML format, so basically a feed parser has the different code compare to crawler, but at the end they both share the same structures of the project to insert data into database.

\_\_author\_\_ **=** 'Pratik'  
**from** datetime **import** datetime  
**from** Common **import** Common  
**from** Constants **import** Constants  
**from** Agency **import** Agency  
**from** AgencyLib **import** AgencyLib  
**from** Author **import** Author  
**from** AuthorLib **import** AuthorLib  
**from** StoryLib **import** StoryLib  
**import** feedparser  
  
**def feedParser**()**:** feed **=** feedparser.parse('http://feeds.feedburner.com/realclearpolitics/qlMj')  
  
 authorLib **=** AuthorLib()  
 storyLib **=** StoryLib()  
 agencyLib **=** AgencyLib()  
  
  
 connection **=** Common.createConnection(Constants.DB\_USER,Constants.DB\_DATABASENAME)  
  
 **for** entry **in** feed['entries']**:** authorDetails **=** entry['author'].split(',')  
 **if** len(authorDetails) **>** 1**:** agency **=** Agency(0,authorDetails[1].encode("utf8"))  
 agencyId **=** agencyLib.insertAgency(agency,connection)  
 author **=** Author(0, authorDetails[0].encode("utf8"))  
 authorId **=** authorLib.insertAuthor(author, connection)  
  
 pubDate **=** datetime.strptime(entry['updated'].split(',')[1].split('-')[0].strip(), '%d %b %Y %H:%M:%S').date()  
 story **=** storyLib.getStroy(entry['title'],pubDate,entry['link'],authorId,agencyId)  
 storyLib.insertStory(story, connection)  
 **else:** agency **=** Agency(0,authorDetails[0].encode("utf8"))  
 agencyId **=** agencyLib.insertAgency(agency,connection)  
 author **=** Author(0, "")  
 authorId **=** authorLib.insertAuthor(author, connection)  
  
 **if** authorId **==** None**:** authorId **=**0  
 pubDate **=** datetime.strptime(entry['updated'].split(',')[1].split('-')[0].strip(), '%d %b %Y %H:%M:%S').date()  
 story **=** storyLib.getStroy(entry['title'],pubDate,entry['link'],authorId,agencyId)  
 storyLib.insertStory(story, connection)  
  
 Common.closeConnection(connection)  
  
  
**def main**()**:** feedParser()  
  
**if** \_\_name\_\_ **==**'\_\_main\_\_'**:** main()

# 3.3 Web Service

Web Service is extremely important part of client and server architecture over the Internet. In general, it is a standardized way where application interacts with data stored on the server using the XML format, JSON, SOAP etc. Using a web service application can take full advantage of front end and back end individually. The web service provides scalability to application; where both web and mobile application of the product can consume the same web service to interact with data from the server.

Node.js is used to develop web services in this project, where RCP search tool consumes this web service for getting the story from the server based on search keywords and filters.

package.json is the configuration file for this web service, that includes the name of the package, version, description and dependencies (express package, gulp package, etc).

{

"name": "wecrawlerapi",

"version": "1.0.0",

"description": "Api to fetch detail from mysql database, that are stored by crawling realclearpolitics.com and reading daily feed.",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "Pratik Bhatt",

"license": "GPL",

"dependencies": {

"express": "^4.13.3",

"gulp": "^3.9.0",

"gulp-nodemon": "^2.0.4",

"mysql": "^2.9.0"

}

}

# 3.4 Web Search Tool

This is the last, but not the least section of the project. Search tool exposes the real purpose of all this work and provide the power to search through the database in the most efficient way.

Initially, the tool will show all the latest stories from the database i.e. Today’s stories. You can also filter these stories based on a date range by selecting different options available in this tool. Today, Yesterday, Week and Month are the options that will allow you to filter stories based on date range.

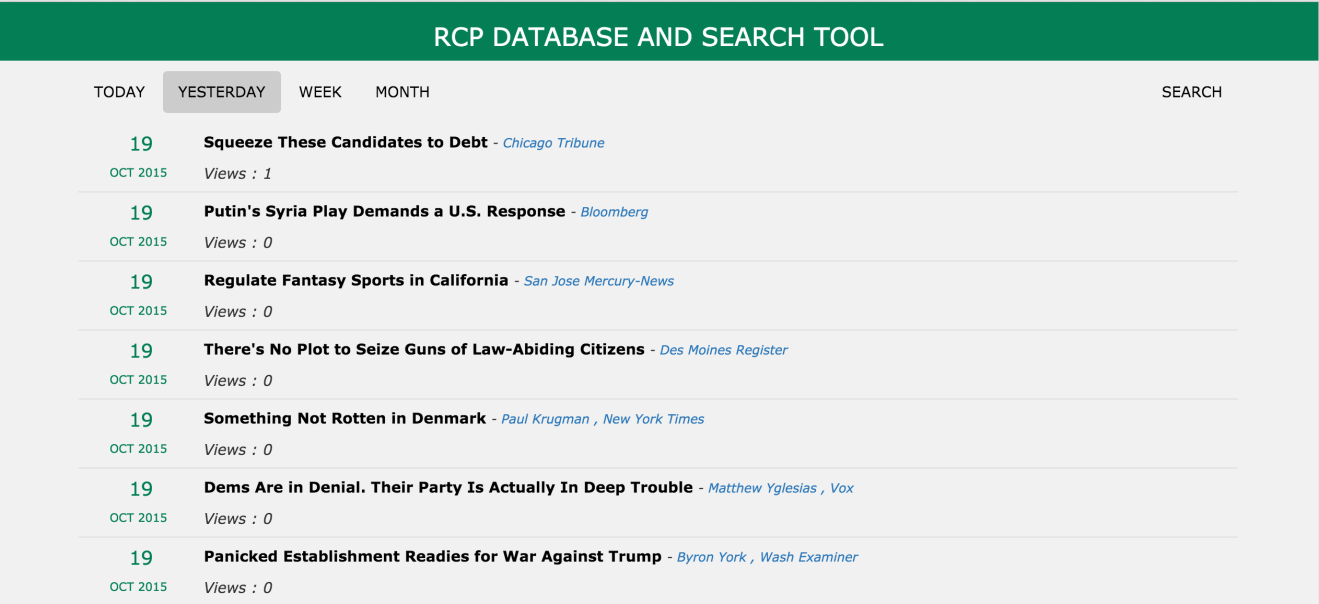


Figure 3.3. Date range selection to filter stories.

You can view any story description and details by clicking the title of the story, which will open a page that has a description, link to RealClearPolitics article and even link to the original article. The description from RCP feed and crawled data is mainly to obtain an overview of any story published in different sources around the world. So sometime user may want to read the story in detail and this was the main purpose of providing the original article link on the page.

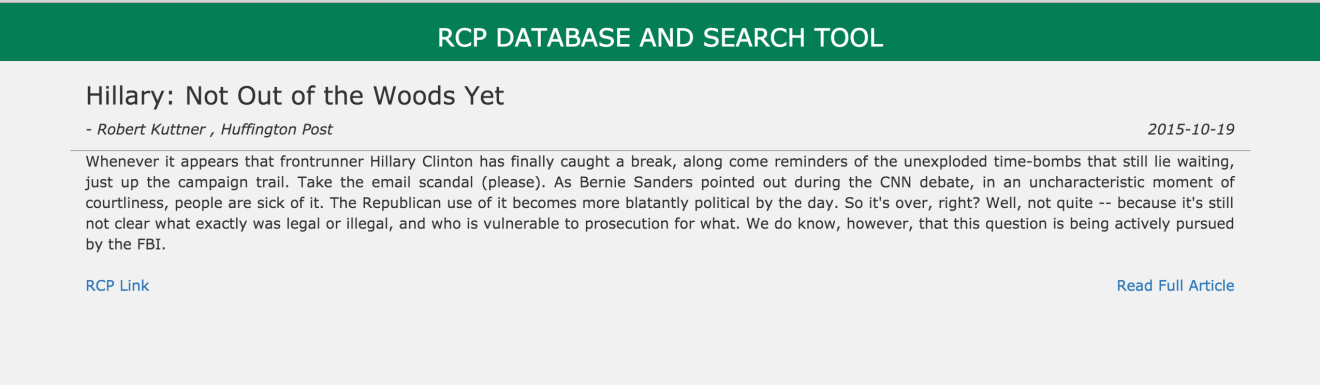


Figure 3.4. Show the details and description of story.

Here if you will click on “RCP Link” it will take you to the article published by RealClearPolitics or if you will click on “Read Full Article” it will take you to the original article published by the Huffington Post.

When any user opens the article to read, the tool will increase the number of views for that article into the database. This information can be helpful to the other user who is using this tool to go through the most viewed stories of the day, a week, a month or maybe a year.

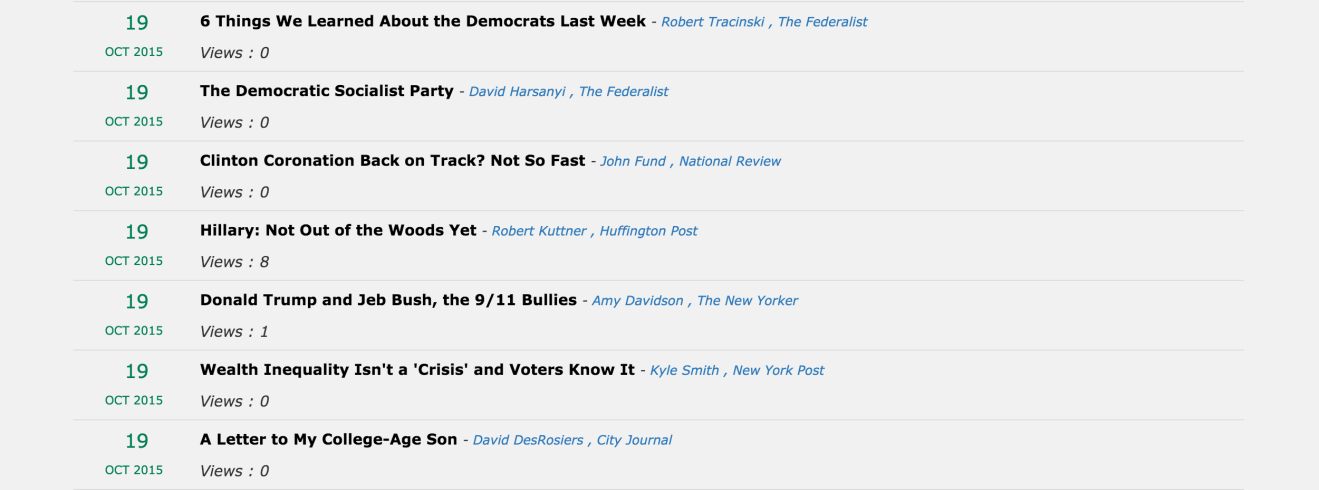


Figure 3.5. Explain the importance of number of views.

Now for searching story, tool query the database using three different filters: Title, Author and Agency.

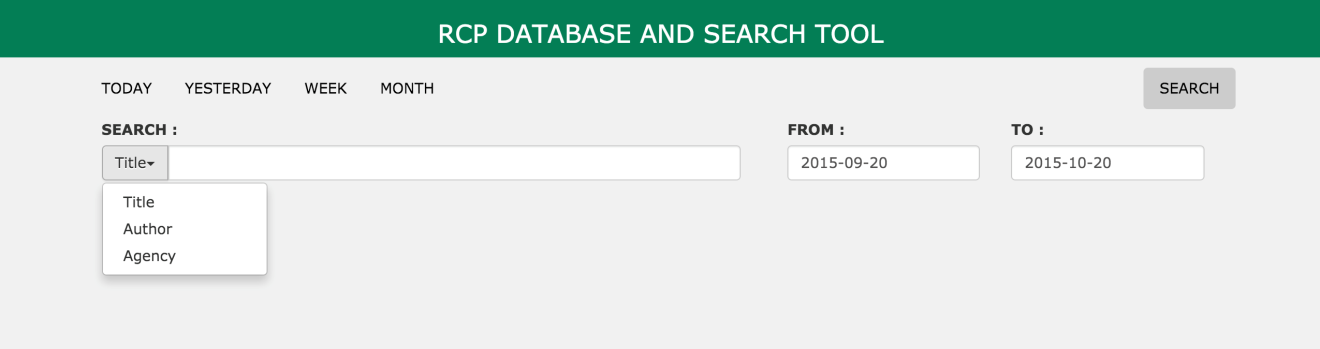


Figure 3.6. Show how user can search the story using different filters.

## 3.4.1 Search by Title

Below is the example of searching story by title; Figure 3.7 shows how search tool will fetch all the stories related to keyword “obamacare” that were published between specified date range.

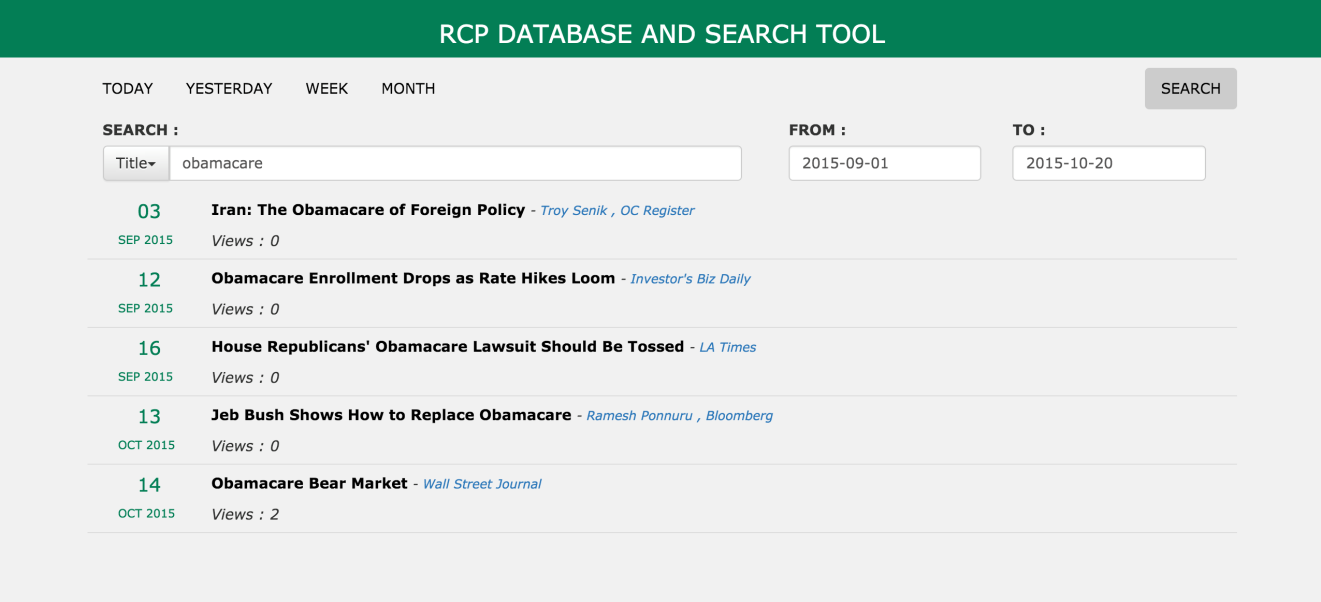


Figure 3.7. Example of search by title.

You can click on either author or agency name directly to filter stories based on author or agency, instead of typing the name or keyword in the search text box. Finally, it is the choice of the user, how they want to filter these stories. Some of the stories were directly published by the agency, so in that case you will not find an author name for that story.

## 3.4.2 Search by Author

Here search tool will fetch all the stories written by “E.J. Dionne” within specified date range.

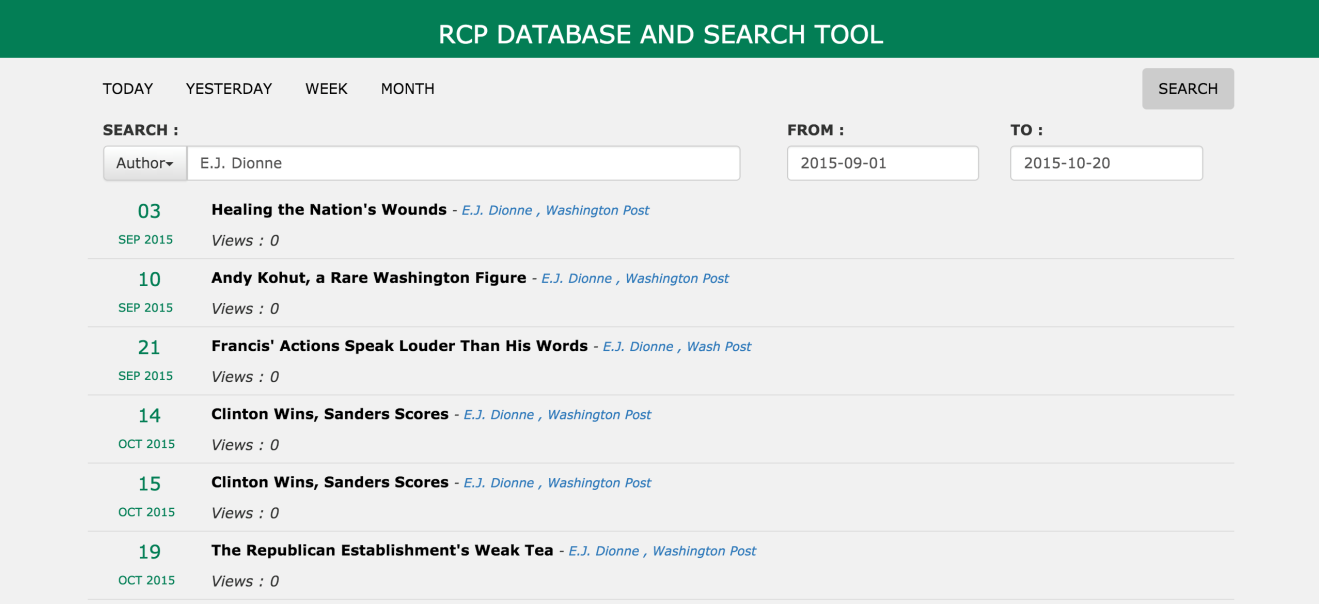


Figure 3.8. Example of search by author.

## 3.4.3 Search by Agency

Here search tool will fetch all the stories published by “The Week” within specified date range.

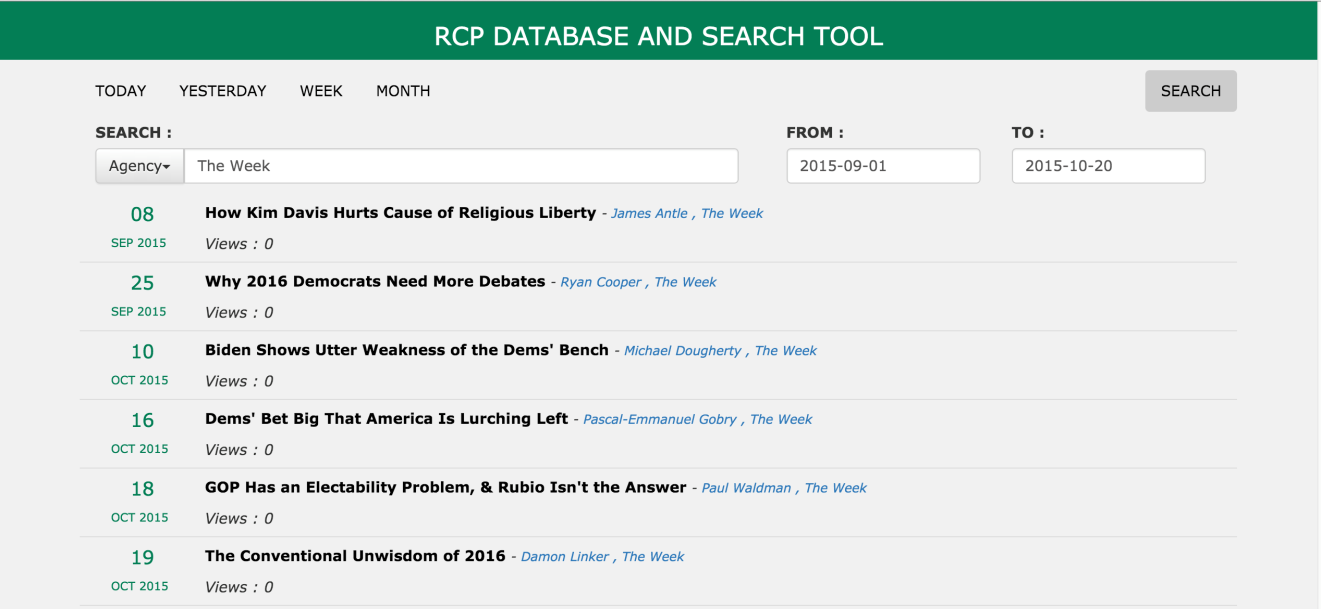


Figure 3.9. Example of search by agency.

CHAPTER 4

USAGE WORKFLOW

# 4.1 Code Development Setup (IDE)

An IDE is a software program, which is used by programmers and developers to write code for software development and known as an integrated development environment. The purpose of using an IDE is to get better productivity and accuracy in code as most of the IDE provide a powerful source code editor, build tools, debugger, version control system and even simplified way of working with GUI. Nowadays some of the IDEs even provides built-in compilers and interpreters unit test support, and refactoring tools.. Popular IDE available in the market for different programming languages are Eclipse, Netbeans, Visual Studio, Pycharm etc.

## 4.1.1 PyCharm

PyCharm is an IDE used to for programming in Python. It was designed and developed by JetBrains and available in two different licenses: Community Edition with Apache License and Professional Edition with a proprietary license. It provides cross platform support and you can run it on Windows, Mac and Linux. Similar to other IDE it provides code analysis, debugger, version control integration and support for the Django framework.

Core Features:

* Intelligent Code Editor
  + Syntax highlighting
  + Code formatting and indentation
  + Line and block commenting
* Easy Code Navigation and Search
  + Go to declaration, method, class.
  + Find usage
* Code analysis, Refactoring and Debugging
* Well Documented
* Django IDE + JavaScript, HTML

## 4.1.2 Brackets

This is an open source IDE written in HTML, CSS and JavaScript with main focus on web development. Adobe Systems designed and developed Brackets and it is available under open source MIT license. It can be installed on all platform Windows, Mac and Linux.

Features:

* Quick Edit
* Well Documented
* Live Preview
* Open Source

## 4.1.3 MySQL Workbench

Workbench is a visual tool used by database architects, developers and DBAs. It is used to design the database, SQL development (writing stored procedures and queries) and database administration such as server configuration, backup, etc.

# 4.2 Application Workflow

In order to obtain a complete working solution we need to host a web search tool, web services and database in such a way that all these three can interact with each other. At the same time, the feed parser needs to be run continuously on the server and check for the latest story. Web search tool can be hosted in apache and web service is developed in Node.js, so we need to install Node.js on the server and that will run web services for us. Most of modern hosting services offer support for MySQL database.

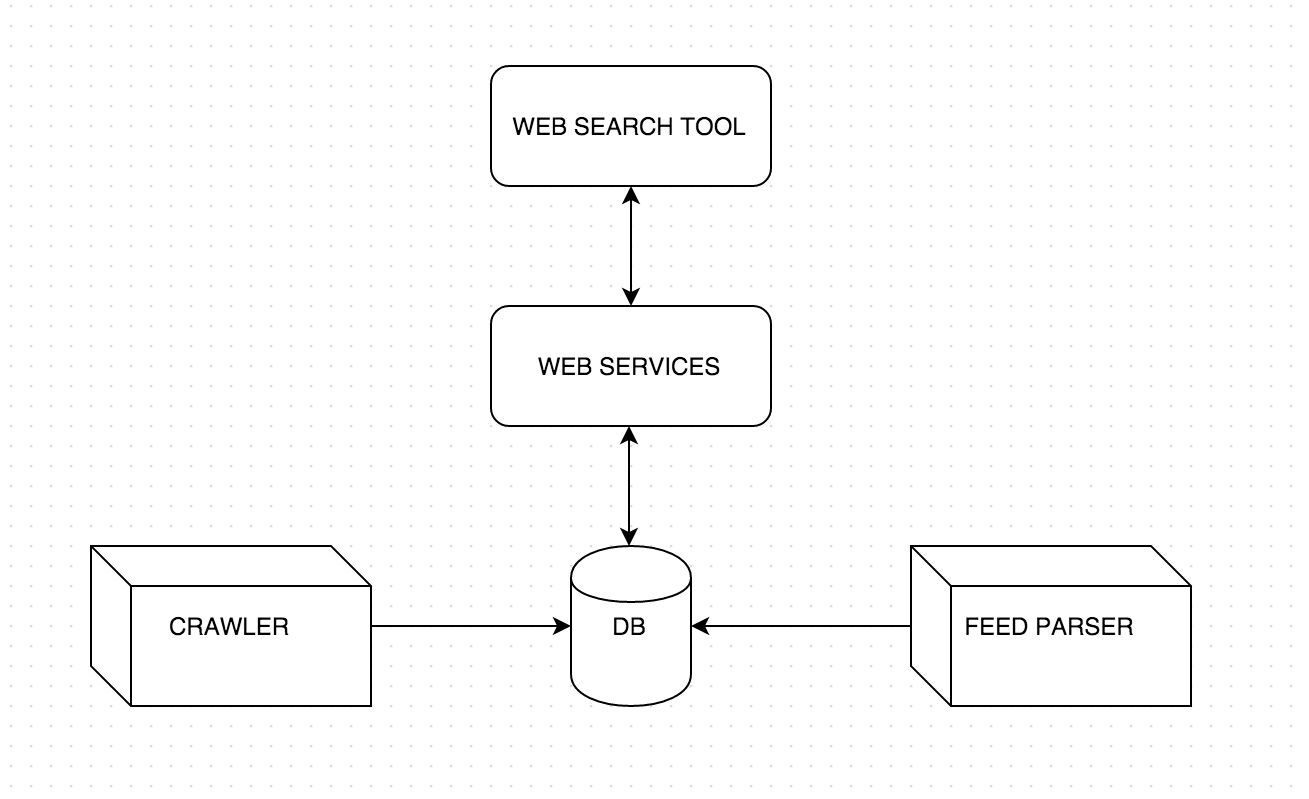


Figure 4.1. Flowchart of project.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

There are lots of mobile applications and web tools available in the market for reading stories from different sources, but none of these provide the power of searching a story and following that particular story from start to end. This search tool provides the functionality of searching all stories in chronological order. The tough task here was to collect old data, as I needed to crawl a website for collecting old data and store all that in the database. The major obstacle was the crawler, as mostly all websites make some improvement to some extent in their design every year. Another tough task was to provide a fast search tool; user must see the search result on the screen soon after typing a search keyword. Taking all this into consideration, different technologies were used to develop the different module and parts of the application. This tool right now provides support only for realclearpolticis.com but we can add more sources as part of future enhancement, so that user can get more stories from different sources. Adding search prediction can also be considered as one of the major enhancements. There are lots of other possibilities for enhancement like user should be able to comment on a particular story, mark favorite story and can select story that he wants to follow every day.

REFERENCES

[1] S. Brin and L. Page, *Reprint of: The anatomy of a large-scale hypertextual web search engine*, Comput. Netw*.,* 56 (2012), pp. 3825-3833.

[2] C. H. Swaroop, *Features of Python*. A Byte of Python, http://www.ibiblio.org/g2swap/byteofpython/read/features-of-python.html, accessed December 2015, n.d.

[3] L. Richardson, *Beautiful soup*. Crummy, http://www.crummy.com/software/  
BeautifulSoup/, accessed December 2015, Sept. 29, 2015.

[4] K. McKee and M. Pilgrim, *Feedparser 5.2.1.* Python, https://pypi.python.org/pypi/  
feedparser, accessed December 2015, n.d.

[5] S. Suehring, *MySQL Bible*, Wiley Publishing, Inc., New York, 2001.

[6] MySQL, *1.3.1 What is MySQL*. MySQL, https://dev.mysql.com/doc/refman/5.1/  
en/what-is-mysql.html, accessed December 2015, n.d.

[7] Xtec, *Guía rápida de administración de MySQL*. Xtec, http://www.xtec.cat/~acastan/textos/  
Administracion%20de%20MySQL.html, accessed December 2015, n.d.

[8] MySQL*, Chapter 1 introduction to MySQL connector/python*. MySQL, https://dev.mysql.com/doc/connector-python/en/connector-python-introduction.html, accessed December 2015, n.d.

[9] Linux Foundation, *Node.js*. Linux Foundation, https://nodejs.org/en/, accessed December 2015, n.d.

[10] T. Capan, *Why the hell would i use Node.js? A case-by-case tutorial*. Toptal,. http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js, accessed October 2015, n.d.

[11] N. Reed, *What is Npm?* docs.nodejitsu.com, https://docs.nodejitsu.com/articles/getting-started/npm/what-is-npm, accessed December 2015, Aug. 26, 2011.

[12] D. C. Wilson and T. J. Holowaychuk*, Strongloop/express*. GitHub, https://github.com/strongloop/express, accessed December 2015, Oct. 27, 2015.

[13] AngularJS, What is Angular? AngularJS, https://docs.angularjs.org/guide/  
introduction, accessed December 2015, n.d.

[14] Tutorials Point, *What is AngularJS*. Tutorials Point, http://www.tutorialspoint.com/  
angularjs/angularjs\_overview.htm, accessed December 2015, n.d.