THE UNIVERSITY OF SYDNEY

School of Information Technologies
Dr. Ying Zhou

COMP5347: Web Application Development **1.Sem./2017**

Assignment Two: Data Analytic Web Application

Group Work: 20% **02.05.2017**

Introduction

In this assignment, you will work as a group to build a small data analytic web application. Each group can have up to **three** members.You will demonstrate that you are able to design and implement a typical three-tiered web application. You will also show that your application can communicate with third party web site through published web services.

You will be given a data set, containing some historical data. You will need to query and compute various summary statistics at server side and present the results on a web page. You may also need to download latest data from the original data source and append that to the data set.

Data Set

The main data set contains a number of files in JSON formats storing revision histories of Wikipedia articles. The data set was created by querying wikipedia API (`https://www.mediawiki.org/wiki/API:Main_page`) in October, 2016. The articles are selected from Featured Articles (`https://en.wikipedia.org/wiki/Wikipedia:Featured_articles`).

Each file stores the entire revision history of an article up to October, 2016. The file is named after the article's title. Each revision is stored as an JSON object, with many properties. The whole file contains an array of many revision objects. Not all properties will be used in the assignment. Below are the relevant ones:

- `title`: stores the title of the article
- `timestamp`: stores the date and time a revision was made
- `user`: stores the user name or ip address of the user that made the revision
- `anon`: the presence of the field indicates that the revision is made by anonymous users.

Explanation of other properties can be found from corresponding MediaWiki API document `https://www.mediawiki.org/wiki/API:Revisions`

Below are examples of two typical revision objects:

```
{
        "sha1" : "b5413ea7b866e5f05455dc008030843998ecac98",
        "title" : "Science and technology of the Song dynasty",
        "timestamp" : "2016-08-17T14:53:13Z",
        "parsedcomment": "...",
        "revid" : 734921057,
        "anon" : "",
        "user" : "58.165.156.2",
        "parentid" : 734921000,
        "size" : 75701
},

{
        "sha1" : "a8196e24dae6ee601cea1eed51b93023a41852b0",
        "title" : "Science and technology of the Song dynasty",
        "timestamp" : "2016-08-09T17:02:40Z",
        "parsedcomment": "..."
        "revid" : 733714887,
        "user" : "Kanguole",
        "parentid" : 732603490,
        "minor" : "",
        "size" : 75643
}
```


Both revisions are from an article with title "Science and technology of the Song dynasty". The first revision is made by an anonymous user as indicated by the presence of property 'anon'. The value of 'user' property is an IP address. The second revision is made by a registered user with name "Kanguole".

The data set also contains two extra text files: `admin.txt` and `bot.txt`. The `admin.txt` file contains a list of all administrators in English Wikipedia. The `bot.txt` contains a list of all bot users in English Wikipedia. Administrators can perform special actions on wiki pages, some of which are recorded as revisions. The bot users have registered names, but are not human editors. They are scripts designed for automatic tasks, such as fixing grammatical errors or detecting vandalism. Many bot user's actions would result in new revisions.

Functional Requirements

Your application should compute various statistics at *overall* data set level and at *individual article* level.

For overall statistics, you need to find out and display the following as text:

- The article with the most number of revisions.
- The article with the least number of revisions.
- The article edited by largest group of registered users. Each wiki article is edited by a number of users, some making multiple revisions. The number of unique users is a good indicator of an article's popularity.
- The article edited by smallest group of registered users.
- The article with the longest history (measured by age). For each article, the revision with the smallest timestamp is the first revision, indicating the article's creation time. An article's age is the duration between *now* and its creation time.
- Article with the shortest history (measured by age).

You also need to make two charts:

- A bar chart of revision number distribution by year and by user type across the whole data set. Figure 1 shows an example.
- A pie chart of revision number distribution by user type across the whole data set. Figure 2 shows an example.

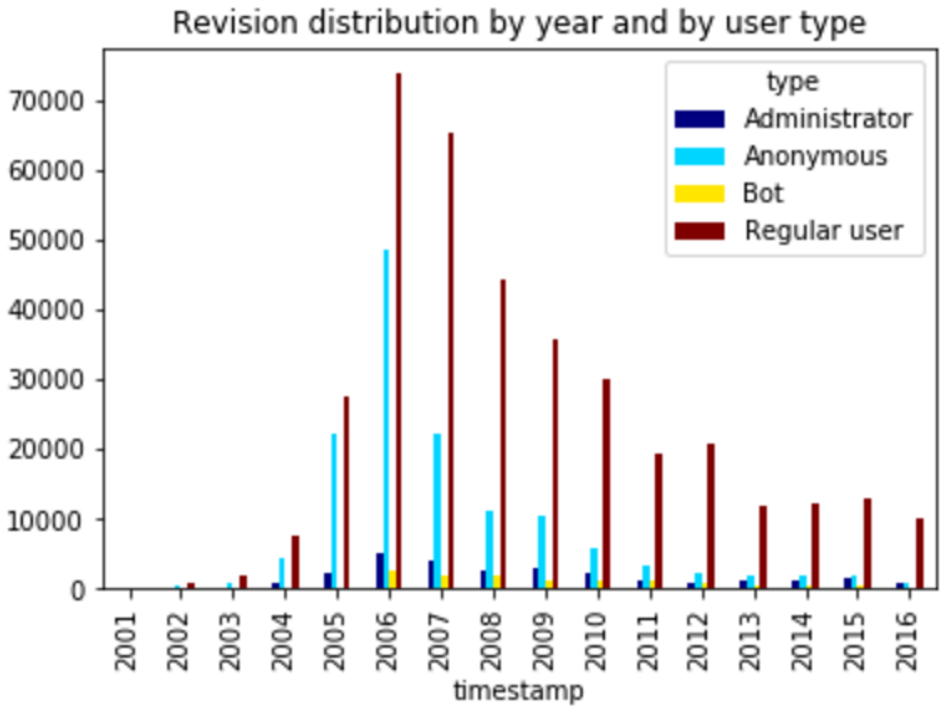

Figure 1: Example bar chart showing overall yearly revision number distribution

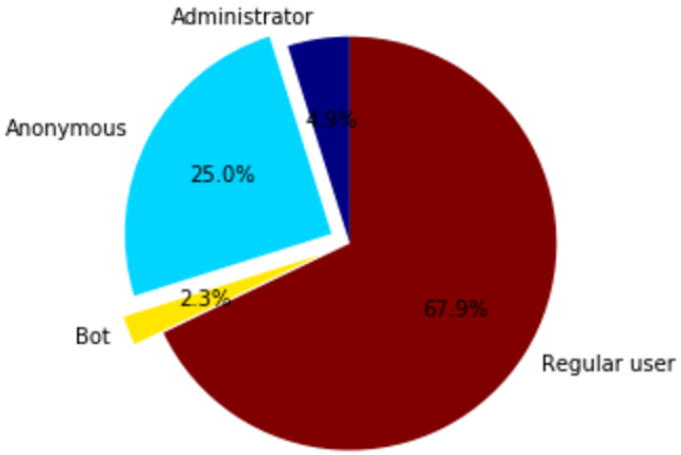

Figure 2: Example pie chart of revision number distribution by user type

We are interested in four types of users: anonymous, administrator, bot and regular user. Revisions made by anonymous users are indicated by the "anon" field in the revision JSON object. Revisions without "anon" field can be made by the other three types of users. You will need to compare the user name with the names in the two text files to determine if a user is administrator, bot or just regular ones.

The text summary should always be displayed on the page in "overall" state. You should provide a way for an end user to switch between the two charts. This could be a button or a link.

For individual page statistics, you should provide a mechanism, for instance, a simple drop down list, for your application's end user to select an article from a list of all available article titles in the data set. You can use other more user friendly mechanisms or add features, such as total number of revisions, next to an article title, to assist with the selection. You may also allow end users to enter the first few letters and use autocomplete feature to quickly locate the title of interest.

Once an end user selects an article, your application need to check if the history of that article in the database is up to date. We consider histories less than one day old as up to date. For instance, if a user has selected article "Australia" at 8:00pm on 21st of April, 2017 and the latest revision of "Australia" in the database was made on 10:00pm, 20th of April, 2017; the history is considered as up to date and you do not need to do anything. However, if the latest revision of "Australia" was made on 10:00am, 20th of April, 2017, the history is considered as not up to date. You need to query MediaWiki API to pull all possible new revisions made after last update.

There should be a message indicating if a data pulling request is made and if so, how many new revisions have been downloaded. It is possible that a data pulling request is made, but the article has no new revision to be downloaded.

For the selected article, display the following summary information:

- The title

- The total number of revisions
- The top 5 regular users ranked by total revision numbers on this article, and the respective revision numbers.

You also need to produce three charts:

- A bar chart of revision number distributed by year and by user type for this article.
- A pie chart of revision number distribution based on user type for this article.
- A bar chart of revision number distributed by year made by one or a few of the top 5 users for this article.

For the last chart, make sure you provide a way to select the users.

Figure 3 shows an example of the yearly revision distribution using data in file "Germany.json". Figure 4 shows a pie chart of user type distribution for article "Germany". Figure 5 shows a bar chart of the year revision distribution of user "GermanJoe" for article "Germany".

The charts should not be displayed in one page, you should provide a mechanism for end users to switch among charts.

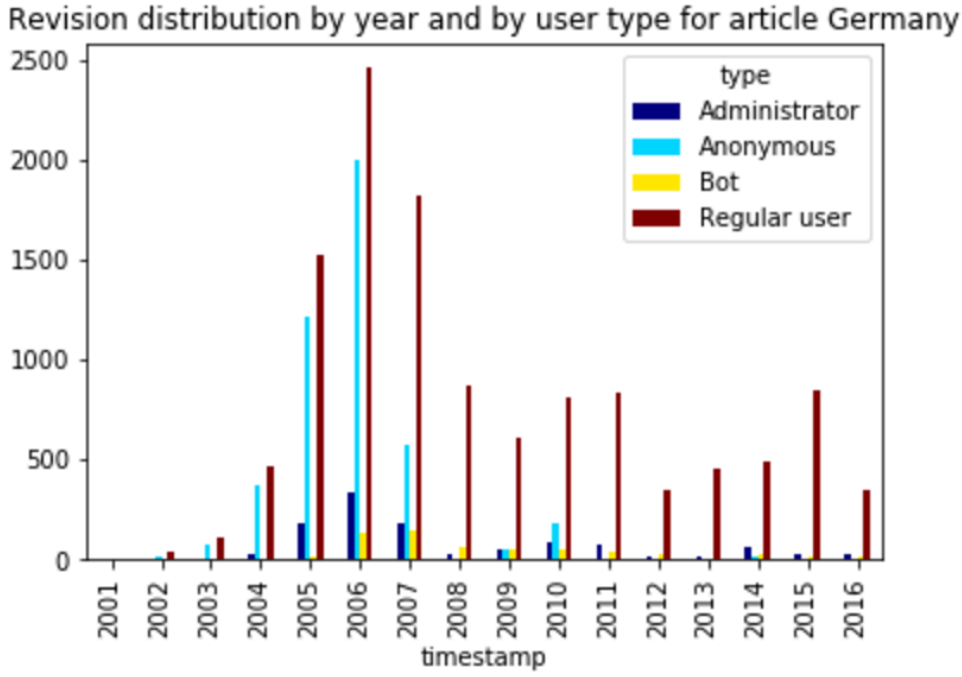

Figure 3: Example bar chart showing yearly revision number distribution

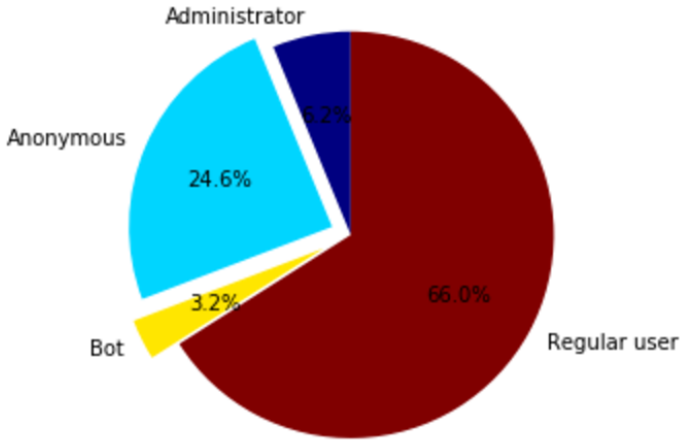

Figure 4: Example Pie chart showing user type distribution

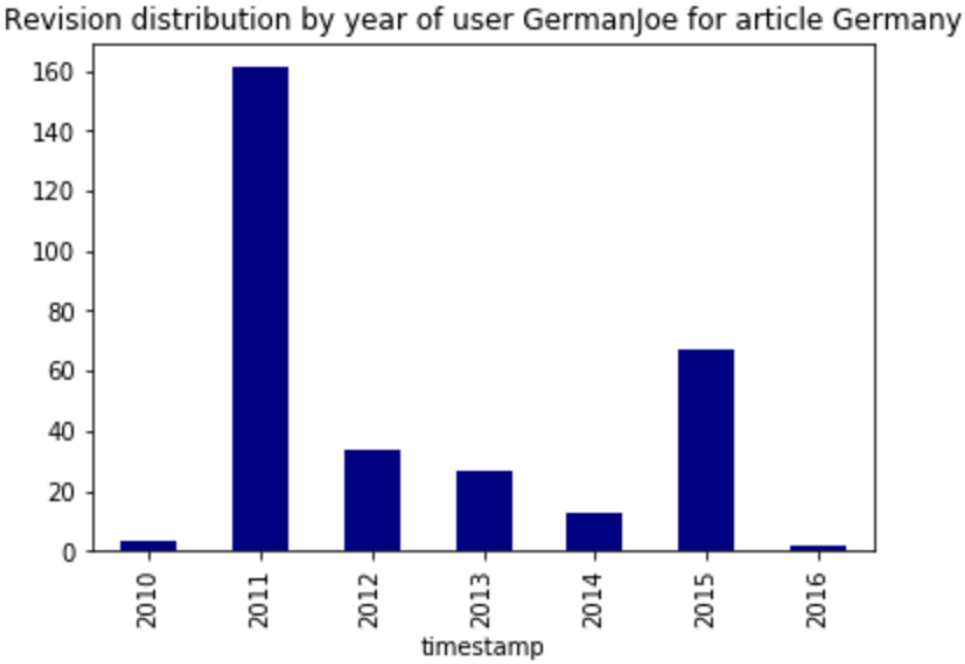

Figure 5: Example single user yearly revision distribution

Design and Implementation Requirements

You application should operate on a single page, with all communications between client and server happen in asynchronous style. You should design your own layout. Below is a recommended navigation structure:

- overall (showing summary text)
  - yearly distribution bar chart (as default)
  - user type distribution pie chart

- individual (showing top 5 users)
  - yearly distribution bar chart (as default)
  - user type distribution pie chart
  - user(s) yearly distribution bar chart

You should use JavaScript to implement both front and back end of the application. The back end application should use `Node.js` framework. The back end storage system should be MongoDB. You are allowed to use other popular JavaScript libraries not covered in this course.

Mark Distribution

- Report: 5 points
  - 3 points for describing the structure of the server application
  - 2 points for describing the structure of the front end
- Functional Requirements: 10 points
  - 5 points for overall statistics and charts
  - 2 points for the article summary and the two article level charts
  - 2 points for top 5 users and the user-yearly distribution bar chart
  - 1 point for performance, your application should not have a latency greater than a couple of minutes. This excludes the Mediawiki API delay.
- Presentation and style: 5 points
  - 1.5 points for single page style
  - 1.5 points for asynchronous style communication
  - 2 points for proper layout, navigation structure and page style

Deliverable and Submission Guideline

- Report Submit a report with cover sheet in week 12 demo. The report should be up to 4 pages not including the cover page. The report should focus on the description of MVC structure and the interaction among them. You may include code snippet but keep it to minimal.
- **Source code submission**
  Submit a zip file with your project files in proper project structure on eLearning site, before 5:00pm on Tuesday 30^{th} of May, 2017 (week 12).

- **System Demo**
  Each student will demo to their tutor in week 12 lab with the submitted version and a demo data set. Note the demo data set will be different to the released data set. Sample charts and statistics based on the released data set will be published for you to check the correctness of your application logic.