THE UNIVERSITY OF SYDNEY

School of Information Technologies

Web Application Development (COMP5347)

Assignment 2: Data Analytic Web Application (WikiLatic)

Group Work: 20%

Submit your project before 11:45 PM 28th of May,2018 on COMP5347 Canvas site

Introduction

In this assignment, you will work as a group to build a small data analytic web application. Each group can have **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 analytics 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.

Dataset

The main dataset contains a number of files in JSON formats storing revision histories of Wikipedia articles. The dataset was created by querying wikipedia API[1]. The articles are selected from Featured Articles[2].

Each file stores the entire revision history of an article up to March, 2018. The file is named after the article's title. Each revision is stored as a 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

[1] https://www.mediawiki.org/wiki/API:Main_page
[2] https://en.wikipedia.org/wiki/Wikipedia:Featured_articles

- 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 at this page[3]

Below are examples of two typical revision objects:


```
[
 {
    "revid": 827326552,
    "parentid": 827320601,
    "minor": "",
    "user": "Ohnoitsjamie",
    "timestamp": "2018-02-24T02:23:05Z",
    "size": 209610,
    "sha1": "9cdb3cffb3f6c751639d23d9f6b4859ba1d9f3d0",
    "parsedcomment": "…",
    "title": "San Francisco"
  },
  {
    "revid": 827320601,
    "parentid": 827029197,
    "user": "65.31.169.21",
    "anon": "",
    "timestamp": "2018-02-24T01:37:37Z",
    "size": 209630,
    "sha1": "fd694a124237ad8d2822eb774303e9dc93ca9c54",
    "parsedcomment": "…",
    "title": "San Francisco"
   },
   …
]
```


[3] https://www.mediawiki.org/wiki/API:Revision

Both revisions are from an article with title "San Francisco". The first revision is made by a registered user with name "Ohnoitsjamie". The second 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 dataset 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

Main/Landing Page

The Landing page should display key information about the application, WikiLatic, in the form of text and images; e.g., description of the available functionality with some sample analytics graphs that can be generated through the application. Also, it should provide two options; Sign-up and Login. Users cannot access or use available analytics functions until they create a valid account and login.

All users must sign-up before they can see and interact with the application functionality. So, your application should allow users to create an account first. The user must provide first name and last name, email address, username, and password. You need to do appropriate data validation to ensure valid data is entered before creating an account. You do not need to implement any verification for the sign-up process. Once all data is entered correctly, an account should be created and maintained in the database.

Once an account is created successfully, a user should be able to login using their user name and password. Users should be able to see and interact with the analytics functionality only upon successful login.

Wiki Analytics Functions

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

Overall Analytics

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

- Titles of the three articles with highest number of revisions. This is the default behavior.
- Titles of the three articles with lowest number of revisions. This is the default behavior.
- The user should be provided with a way to change the number of articles for highest and lowest number of revisions, the same number should be applied to both categories.
- 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 top 3 articles 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).

Beside the above functionality, you also need to show:

- A bar chart of revision number distribution by year and by user type across the whole dataset. 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.

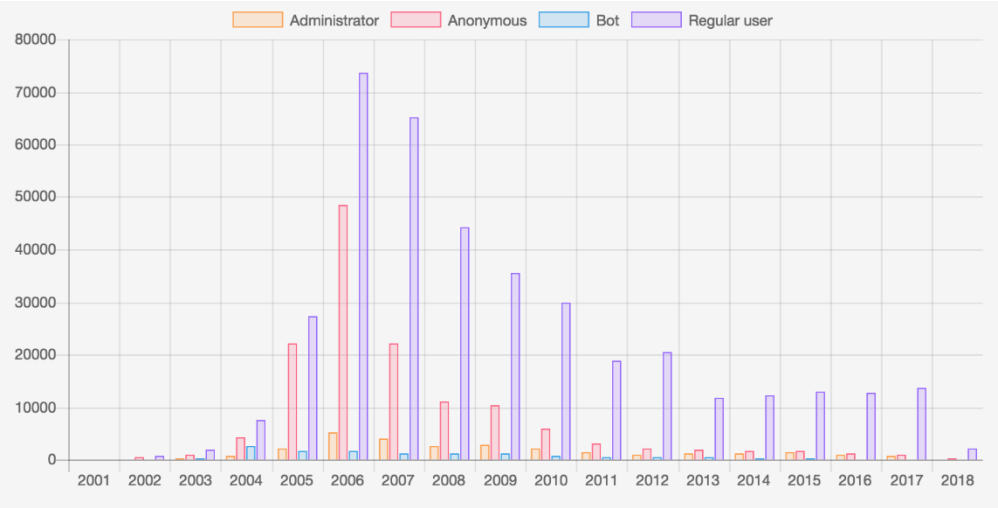

Figure 1: Example bar chart showing overall yearly revision number distribution

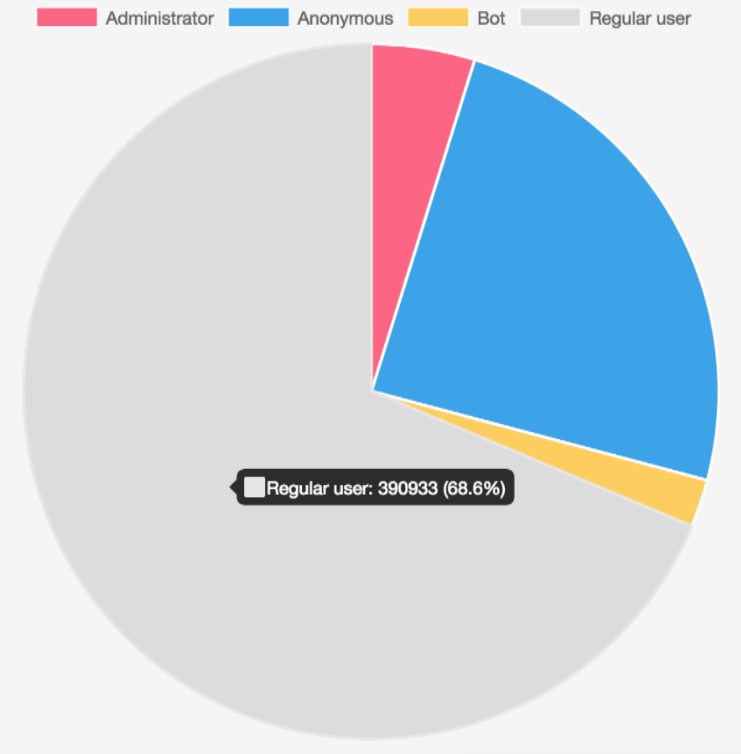

Figure 2: Example pie chart of revision number distribution by user type

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.

Individual Article Analytics

The individual page analytics 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 should also 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, 2018 and the latest revision of "Australia" in the database was made on 10:00pm, 20th of April, 2018; 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, 2018, 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 "Horst-schlaemma" for article "Germany".

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

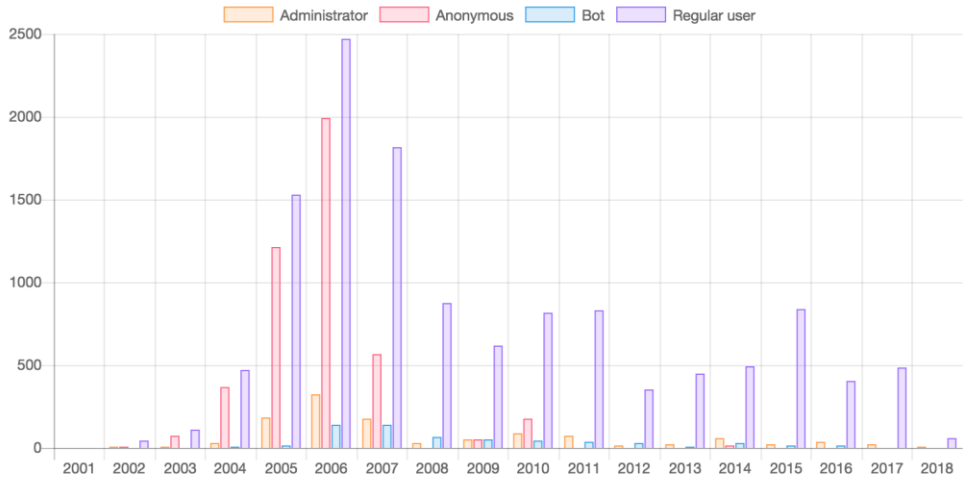

Figure 3: Example bar chart showing yearly revision number distribution

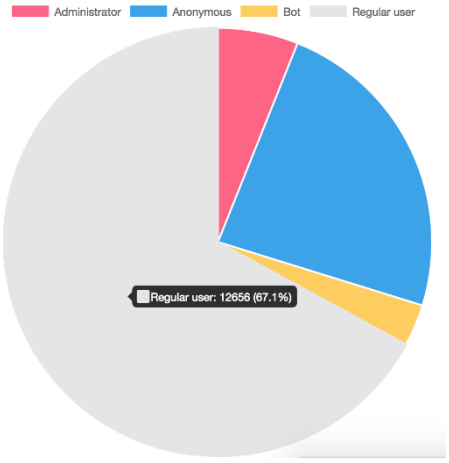

Figure 4: Example Pie chart showing user type distribution

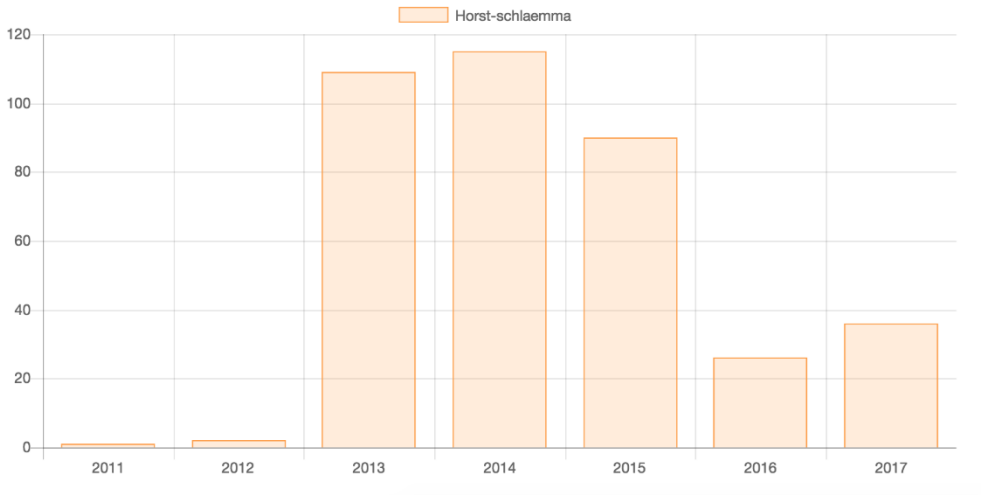

Figure 5: Example single user yearly revision distribution

Author Analytics

In this page, you should enable the end user to display all articles that are changed (or have revisions made) by a specific author. To do so, you should allow the end user to an author name in a free text format.

You should display the articles' names along with number of revisions made by that author next to it. The end user should also be able to select to see time stamps of all revisions made to an article, if that author made more than revision to an article he is attributed with.

Design and Implementation Requirements

You application should operate on a single page, with all communications between client and server happen in asynchronous style. For simplicity, it is allowed to implement the main/landing page functions in a separate page. The other functions (overall, individual and author analytics) must be implemented within a single page (following SPA principles).

You should design your own layout. You should use the MVC pattern to structure your application and interaction among components. The design and UI interfaces should be user-friendly and intuitive.

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.

Deliverable and Submission Guideline

- **Source code submission**
  Submit a zip file with your project files in proper project structure on eLearning site, before 11:45 PM on Monday 28th of May,2018.

- **System Demo**
  Each student will demo to their tutor during 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. All team members must attend the demo and participate in the demo. All team members must be prepared to answer questions about the code and design of their application.