Kimberly Bridglal – 811000371

Warren O’Connell – 811000293

Aaron Yuk Low – 811003468

COMP 3550 PROJECT: TWI-LITE

**About**

Being bombarded with ‘Too Much Information’ is a common problem faced by the vast majority of internet user. Nothing can be done about the amount of information being flooded into the world but the information being brought to users can be filtered to show more relevant information to any given user. Twi-Lite proposes to do so by finding out what the user is interested in, getting terms related to it from an external service and finding those terms in given text (in his case tweets) and showing it to the user in an attempt to control what is seen. When 30 relevant tweets are received, the feed stops until the user clicks to button to get a new set. The current set is then replaced by a new set of 30. These Tweets can be seen in the Home page after login. The trending page shows the user to top 20 most followed topics so that they may chose to follow too. The Lite tweets page allows users to see tweets based on the popular topics in sets of 30 just like the home page. For each tweet, the user has the option of clicking on the publisher’s picture or screen name to be taken to their twitter page. Also, hashtags and external links are also clickable to make it easier to reach more information related to a tweet that is deemed interesting to the user. The profile page is where users control their interest. To add an interest, the user just needs to add a string in the box provided. To remove an interest, the user needs only click on the button of that interest and it is gone.

Processing the information based on tag words means that this service, fed any text, can perform a similar job of sifting information. This can range from services similar to Twitter like Facebook or even News feeds and many others. The benefit of using NodeJS for implementation is that the client and server are separated. This means that the front end it totally separate from the server. The server has a relatively small workload because it is only responsible for streaming, authentication and returning information stored in the database. All other processing is performed on the client machine.

**Instructions**

In the ‘db’ folder contained in the root folder, the file ‘localhost.sql’ should be imported into the mysql server. The user credentials are host: "localhost", user: "tweetydb", password: "tweetydb" and database: "tweetydb" when making the database user.

Once the database is setup and running, simply call ‘node app.js’ from the root folder and visit ‘locallhost:3000’ in a browser to begin use. The download link is provided at the bottom of this document. A table containing the locations of the project objectives precedes it.

| Category | Specifics | URL / File & Line # | Mark |
| --- | --- | --- | --- |
| REST Services |  |  |  |
|  | Utilize information from Twitter API | <http://localhost:3000/lite.html>  http://localhost:3000/home.html | 2 |
|  | Perform GET request generated from developed API | app.js  line 394  line 402 | 2 |
|  | Perform POST request from developed API | App.js  Line 295  Line 391 | 2 |
|  | Perform DELETE request from developed API | NA | 2 |
|  | Create two GET request | app/scripts/trending.js line 32 | 2 |
|  | Create two POST request | app/index.html line 27 | 2 |
|  | Create one PUT request | NA | 2 |
|  | Create one DELETE request | App/scripts/profile.js line 116 | 2 |
|  | Accept at least two parameters from client | http://localhost:3000/register.html | 2 |
|  | At least two methods should be authenticated | App.js  Line 191  Line 389 | 2 |
|  | Utilization of session within the developed API | App.js  Line 137  Creates session used in checkAuth | 2 |
|  |  |  |  |
| Cloud platform | **(Heroku / DO)** | NA |  |
|  | Successfully configure cloud service and needed software | NA | 10 |
|  | Successfully deploy application code to the cloud | NA | 10 |
|  |  |  |  |
| Database Connectivity |  |  |  |
|  | Successfully connect to database | App.js  Line 23 | 2 |
|  | Create user with associated database | App.js line 23 | 2 |
|  | Insert data into the database | App.js line 203 | 2 |
|  | Read data from database | App.js line 82 | 2 |
|  | Utilization of Limit & Offset | App.js line 419 | 2 |
|  |  |  |  |
| Data Visualization |  |  |  |
|  | Successfully Implementation of at least one visualization | http://localhost:3000/trending.html | 10 |
|  | Applicability to the problem |  | 2 |
|  | AJAX based information | App/scripts/trending.js Line 30 | 5 |
|  |  |  |  |
|  |  |  |  |
| CS Topic | **(Modularity / Scalability / Efficiency / Extensibility)** |  |  |
|  | Identification of at least one CS topic | Extensibility | 2 |
|  | Justification of the design factors selected | Any text based service, including twitter and facebook, to news and blogs can integrate this service easily | 10 |
|  | How well was the idea Implementation |  | 5 |
|  |  |  |  |
| IO Problem |  |  |  |
|  | Solution |  | 4 |
|  | Innovative |  | 5 |
|  | Works as Described |  | 5 |

Make the charts interactive [5]

http://localhost:3000/trending.html

Use Git for deployment [5]

https://github.com/renocon/twi-lite.git

The code and instruction on running the application

<https://github.com/renocon/twi-lite.git>

code can be cloned from ^