CMPE-272 Sec 98 - Enterprise Software Platforms

Assignment #2

Requirement:

Develop a client application that utilizes

1. Apache Karaf Container
2. Implement Twitter APIs

Team Name: Shield

STUDENT NAME	STUDENT ID	GITHUB REPO
Anushri Srinath Aithal	012506897	https://github.com/shriaithal/Shield_dev.git
Anuradha Rajashekar	012409956	https://github.com/AnuradhaIyer/Ansible-Play-book
Nidhi Jamar	010070593	https://github.com/nidhijmr/Shield
Ashwini Shankar Narayan	012506910	https://github.com/Ashwinisnv/Shield_Ashwini.git

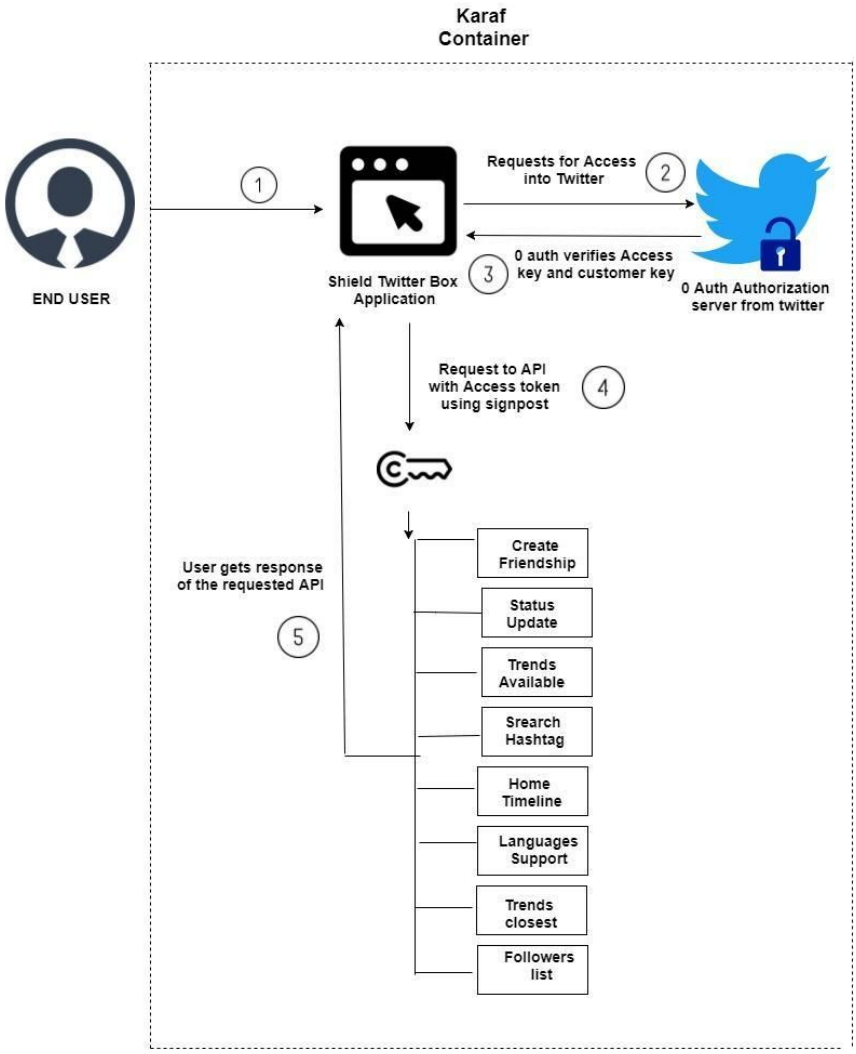
Team GitHub Repository: https://github.com/AnuradhaIyer/Shield.git

Abstract:

The objective of this implementation is to demonstrate Twitter API integration with custom web application and deployment on Apache Karaf Container.

1. Implemented 8 Twitter APIs to illustrate Twitter Web Application integration.
2. OAuth protocol is used to establish authentication between our web application and Twitter.
3. Using Java's Servlet classes and Signpost, Twitter REST APIs are invoked.
4. The responses are received in JSON format which is then used to display informative messages to the end user. We are using JSP and JavaScript to render data to the user.
5. Apache Karaf is used to provide a deployment environment for this module.

Application Architecture Diagram:

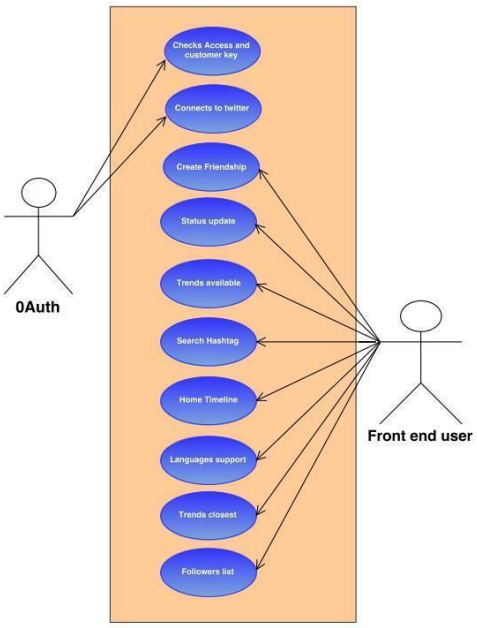


Design Pattern Used:

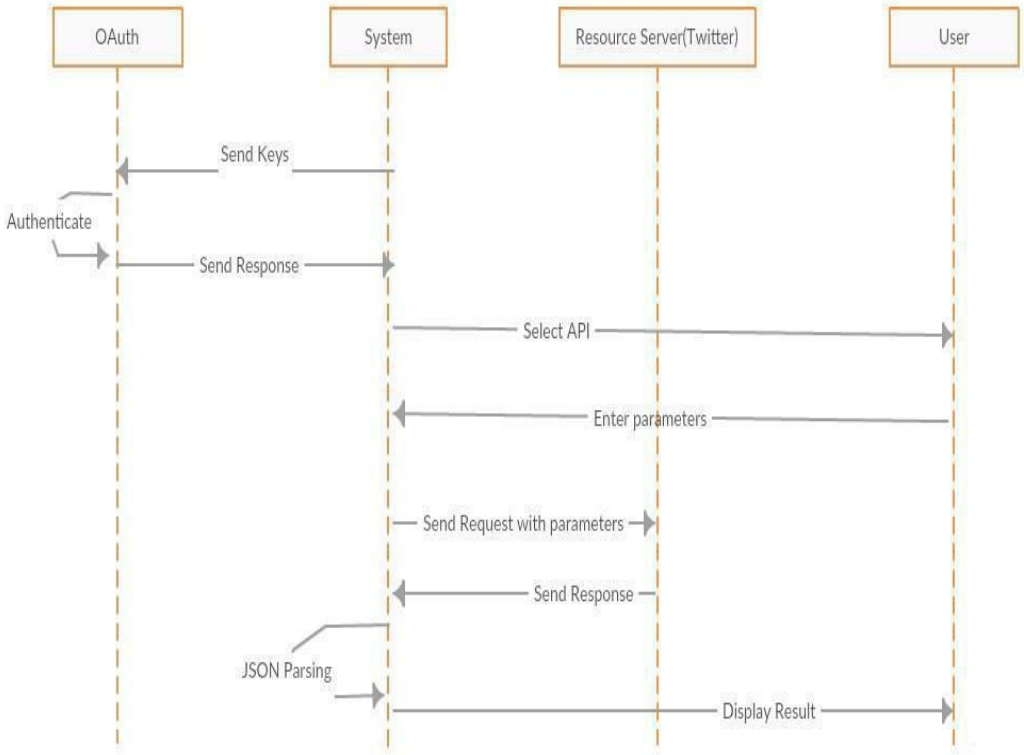
MVC design pattern is used to implement this Web Application. In the application,

1. Model – Java POJO classes which represents the data to be presented to the users.
2. View – JSP pages that represents the visualization of data that the model contains.
3. Controller – Java Servlets that handle both the model and view managing data flow between them.

UML Diagram:



Sequence Diagram:

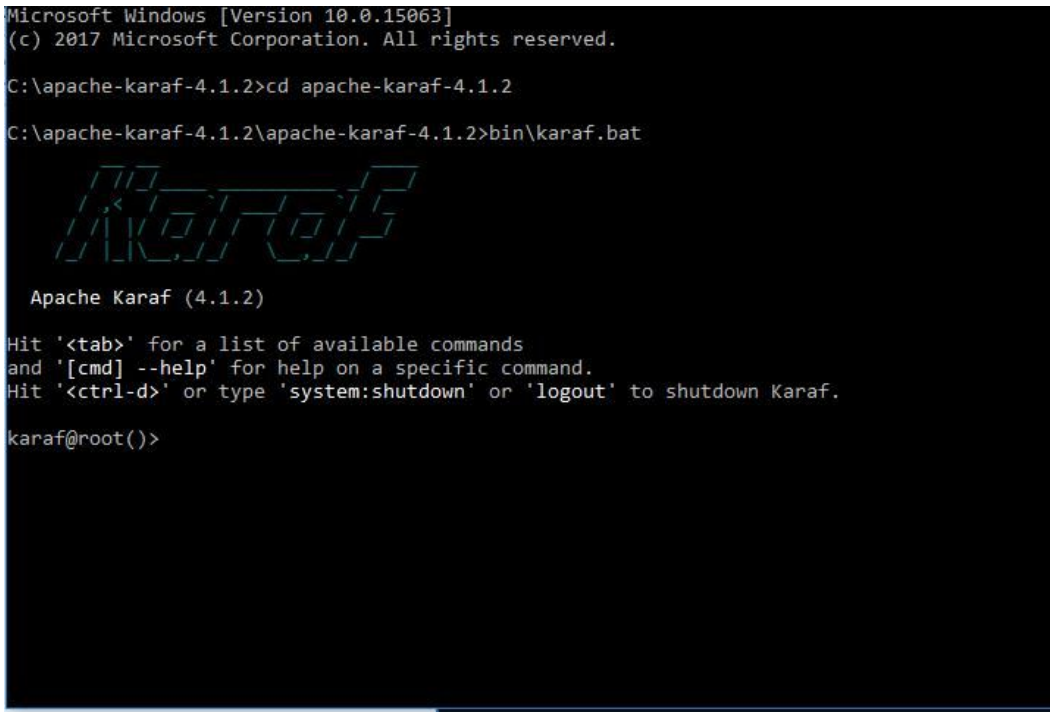


Tools:

1. Apache Karaf Container
2. Java Http Servlet
3. JSP
4. JavaScript
5. Signpost
6. Twitter APIs

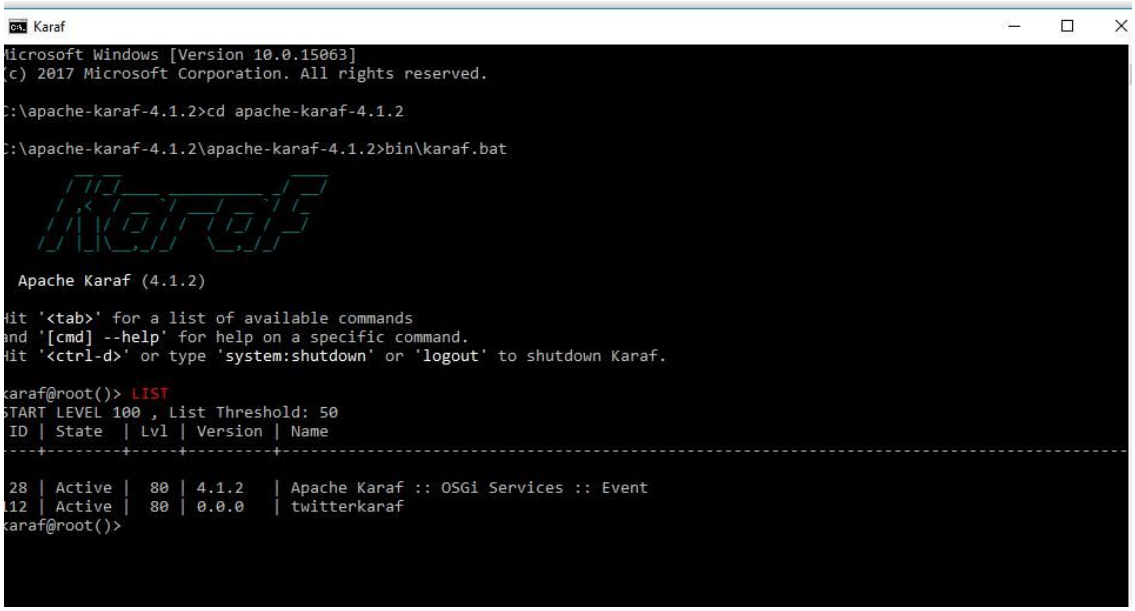
Apache Karaf Installation and WAR deployment:

1. Download apache karaf 4.1.2 and extract the file.
2. In command prompt navigate to the extracted folder bin and launch Karaf container by executing karaf.bat file.

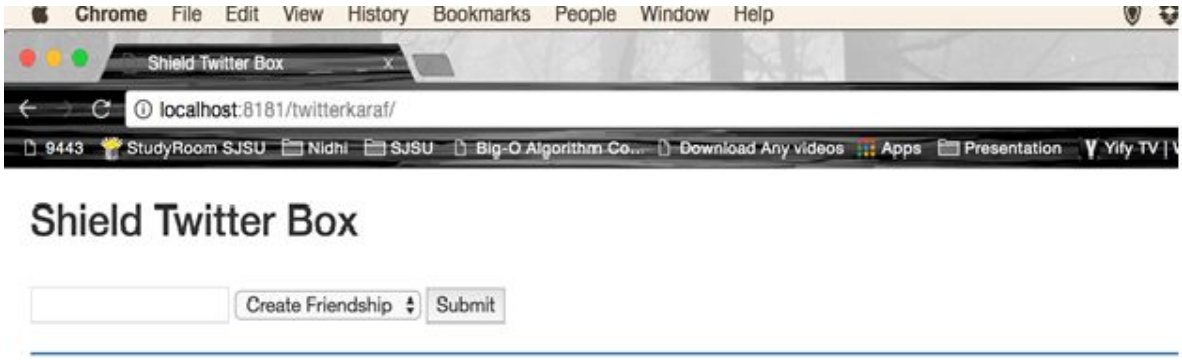


3. On successful launch of Karaf, install the war and http features to enable deployment of WAR on Karaf Container. Run the following commands
    a. feature:install war
    b. feature:install http
    c. feature:install webconsole
4. Java Servlet is used to handle the HTTP GET and POST calls from UI. Twitter APIs are invoked using the request data sent by end users.
5. Maven is used as a build tool to package the entire application into a WAR file.

6. To deploy the WAR on Apache Karaf, follow the below steps:
    a. Go to .m2 folder, copy the project war
    b. Go to Apache Karaf→ deploy folder
    c. Place the WAR and start the Karaf container
    d. Go to command prompt and run command LIST
7. On Karaf command prompt, run the command feature:list to ensure that your module has been deployed.



8. Run the application on your browser at http://localhost:8181/[WAR name]. In our client application, the WAR name is “twitterkaraf” hence launch http://localhost:8181/twitterkaraf



Get Twitter API keys:

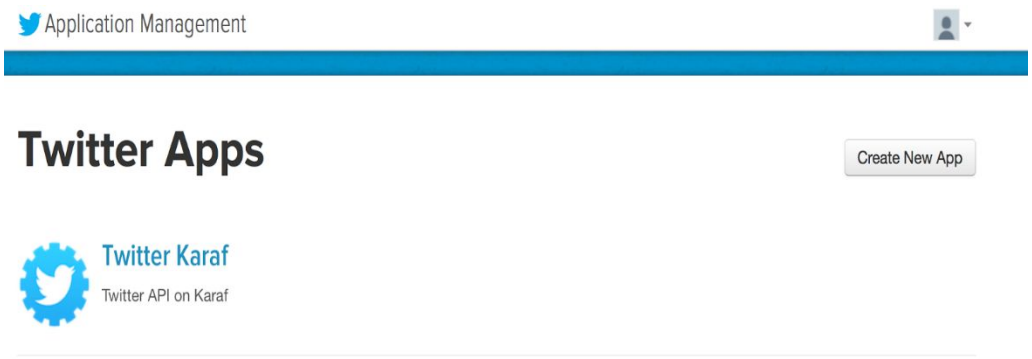
Following are the prerequisites to integrate Twitter APIs in any application:

1. One should have a Twitter User account.

2. App should be created on Twitter Application Management Portal (https://apps.twitter.com/) to obtain tokens such as API Key, API Secret, Access Token and Access Secret Token.

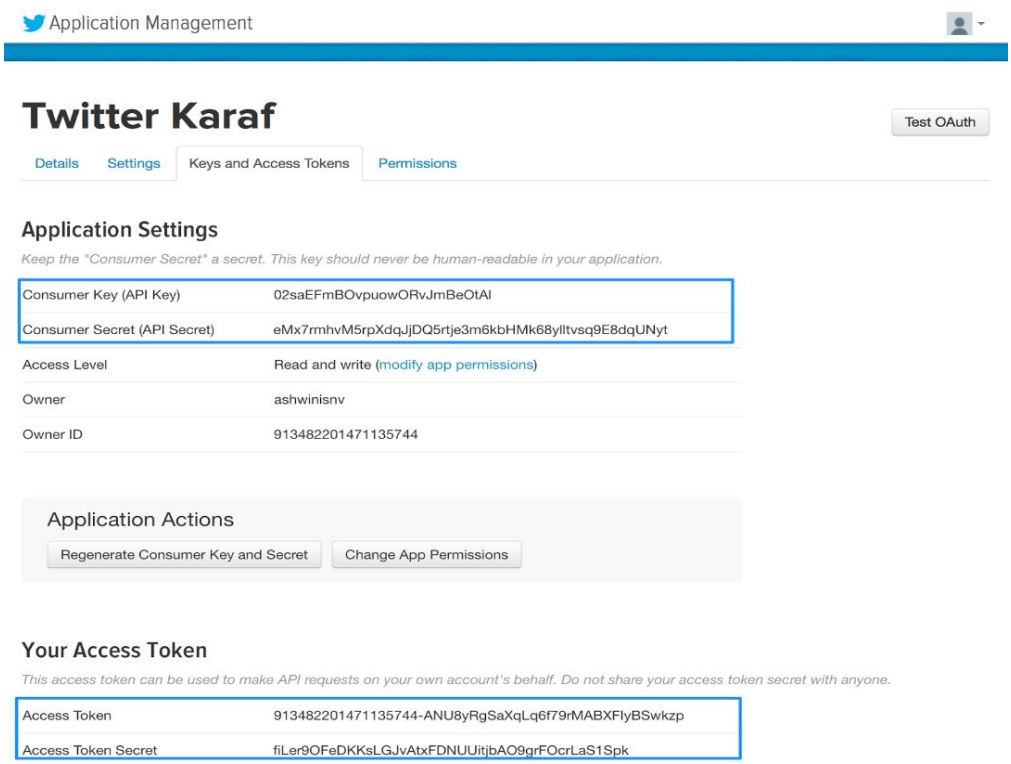
The detailed steps to get Twitter API keys are as follows:

1. Create a Twitter account if you do not have one already.
2. Go to https://apps.twitter.com/. This link will take you to Twitter Application Management Portal. Login here using your Twitter user account credentials.
3. Click on 'Create New App' on the top right corner. This will enable us to create a new application.



Twitter Application Management Portal

4. Fill out the form with application name, description, website and callback URL. Agree to the terms and click on 'Create your Twitter application'.
5. On the next page, go to tab 'Keys and Access Tokens'. You can now see your API Key and API Secret created.
6. Scroll down and click on 'Create my access token' to create Access Token and Access Token Secret.



Get Access tokens and API Keys on Twitter Developer Account

Twitter APIs Implementation Details:

Below are the Twitter APIs implemented in this application

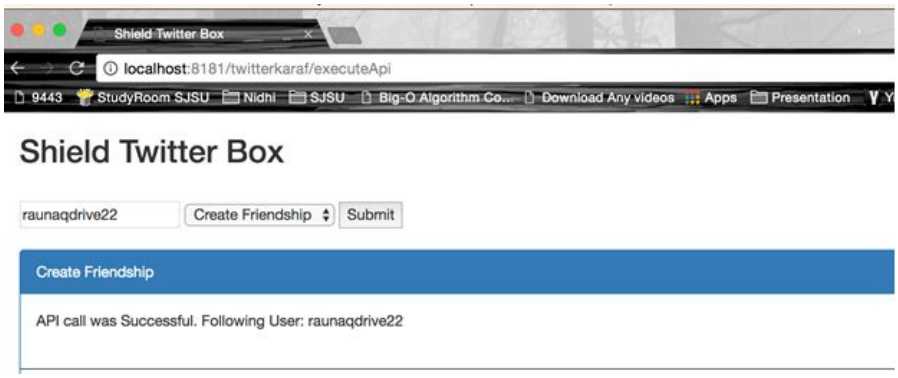
1. **POST friendships/create:** Add a new follower. Takes twitter handler as input.
2. **GET status/update:** Update Twitter status with the input string.
3. **GET trends/available:** Fetch the trending tweet topics
4. **GET search/Hashtag:** Find max of 10 tweets based on the input hash tag.
5. **GET Home/Timeline:** Find all tweets on your timeline
6. **GET help/languages:** Find all languages supported by Twitter
7. **GET followers/list:** Fetch all the followers list
8. **GET trends/closest:** Find the trending tweets closest to your locality.

Signpost is used to make the Rest Calls using the Twitter OAuth tokens. Below is a sample Http Post and Get call executed using Signpost

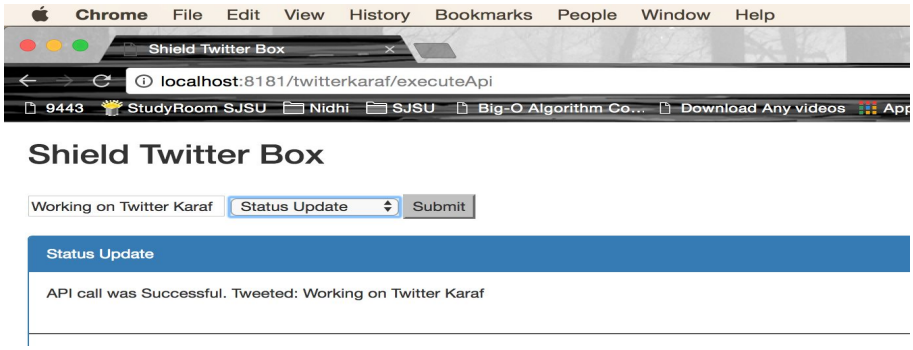
```java
/**
 * Execute Http Post calls to invoke the post APIs
 *
 * @param apiUrl
 * @return
 * @throws OAuthMessageSignerException
 * @throws OAuthExpectationFailedException
 * @throws OAuthCommunicationException
 * @throws IOException
 * @author Nidhi Jamar
 * @author Anuradha Rajashekar
 */
private HttpResponse executeHttpPost(String apiUrl) throws OAuthMessageSignerException,
        OAuthExpectationFailedException, OAuthCommunicationException, IOException {
    HttpPost httprequest = new HttpPost(apiUrl);
    getOAuthConsumer().sign(httprequest);
    HttpClient client = new DefaultHttpClient();
    HttpResponse httpresponse = client.execute(httprequest);
    int statusCode = httpresponse.getStatusLine().getStatusCode();
    System.out.println(statusCode + ":" + httpresponse.getStatusLine().getReasonPhrase());
    return httpresponse;
}

/**
 * Execute Http Get calls to invoke the Get APIs
 *
 * @param apiUrl
 * @return
 * @throws OAuthMessageSignerException
 * @throws OAuthExpectationFailedException
 * @throws OAuthCommunicationException
 * @throws IOException
 * @author Anushri Srinath Aithal
 * @author Ashwini Shankar Narayan
 */
private HttpResponse executeHttpGet(String apiUrl) throws OAuthMessageSignerException,
        OAuthExpectationFailedException, OAuthCommunicationException, IOException {
    HttpGet httprequest = new HttpGet(apiUrl);
    getOAuthConsumer().sign(httprequest);
    HttpClient client = new DefaultHttpClient();
    HttpResponse httpresponse = client.execute(httprequest);
    int statusCode = httpresponse.getStatusLine().getStatusCode();
    System.out.println(statusCode + ":" + httpresponse.getStatusLine().getReasonPhrase());
    return httpresponse;
}
```

Client Application Screenshots for various APIs implemented:

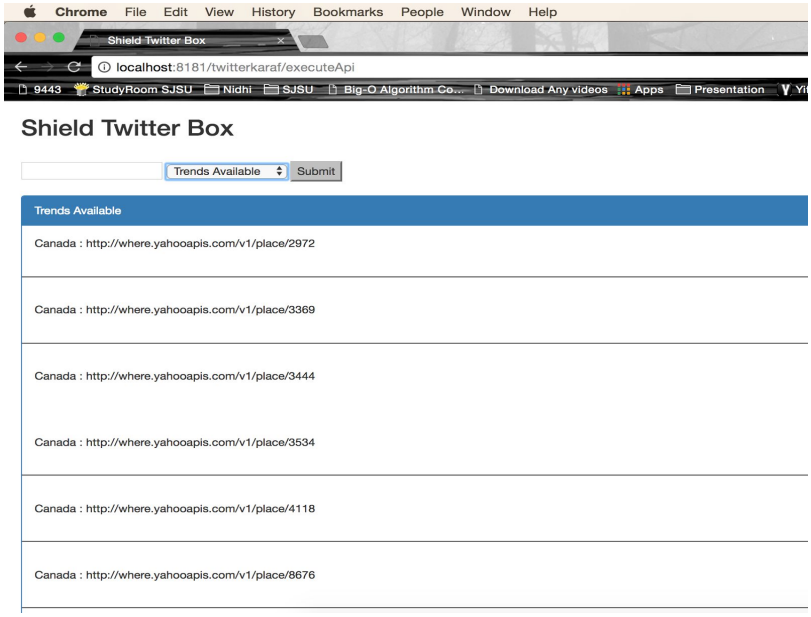
1. **Create Friendship**



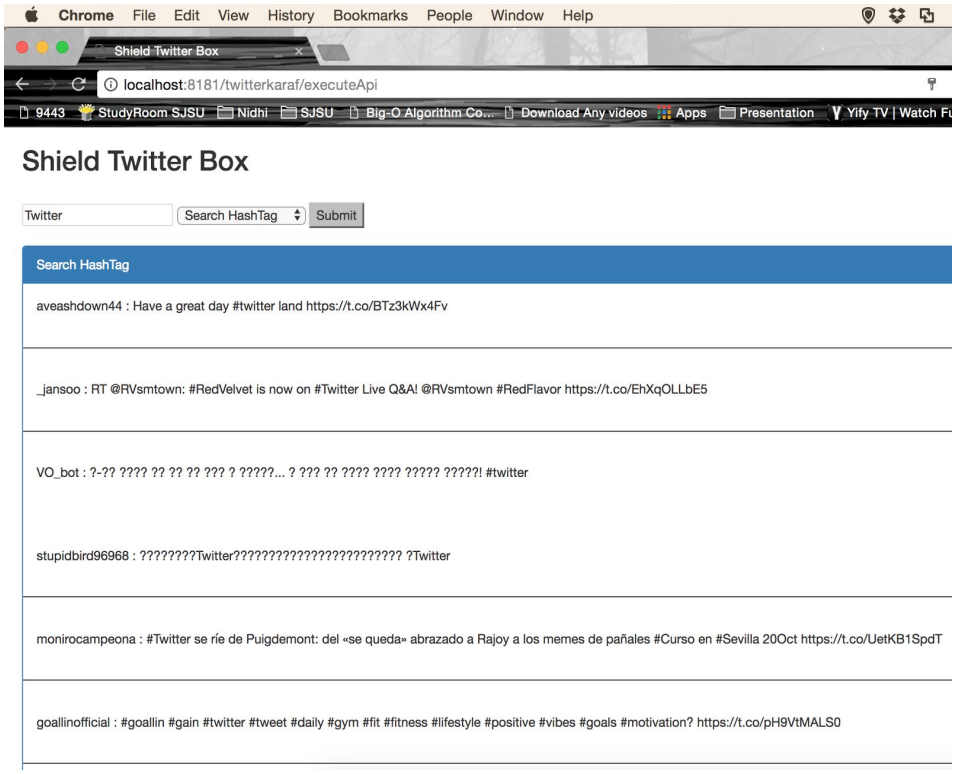
2. Status Update



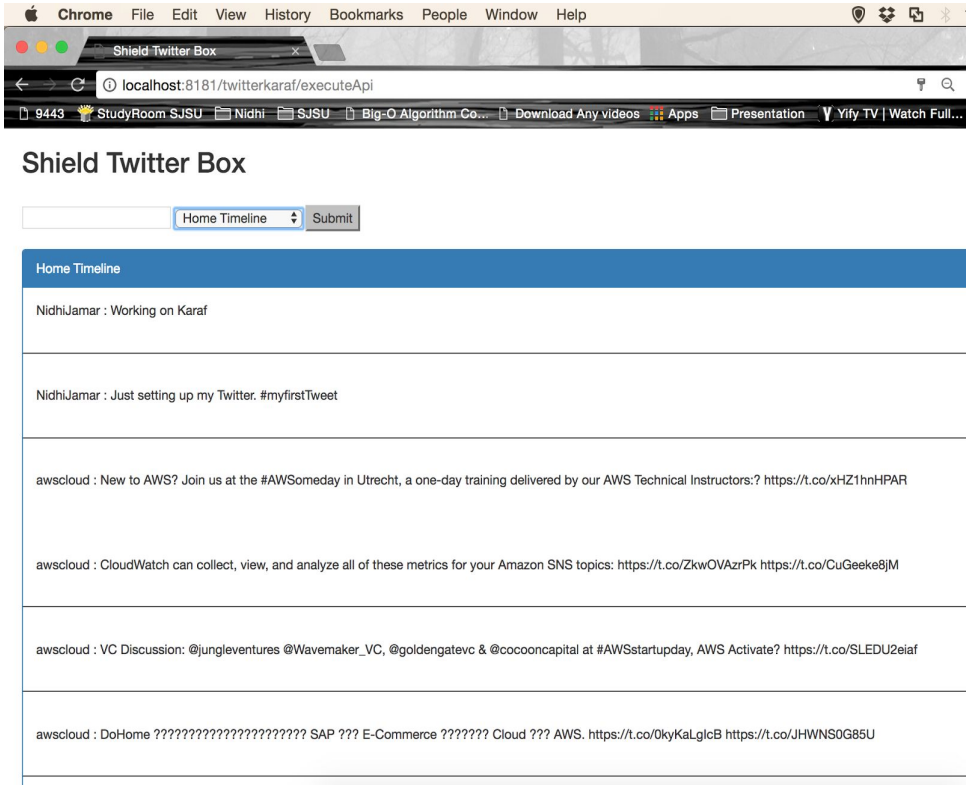
3. Trends Available



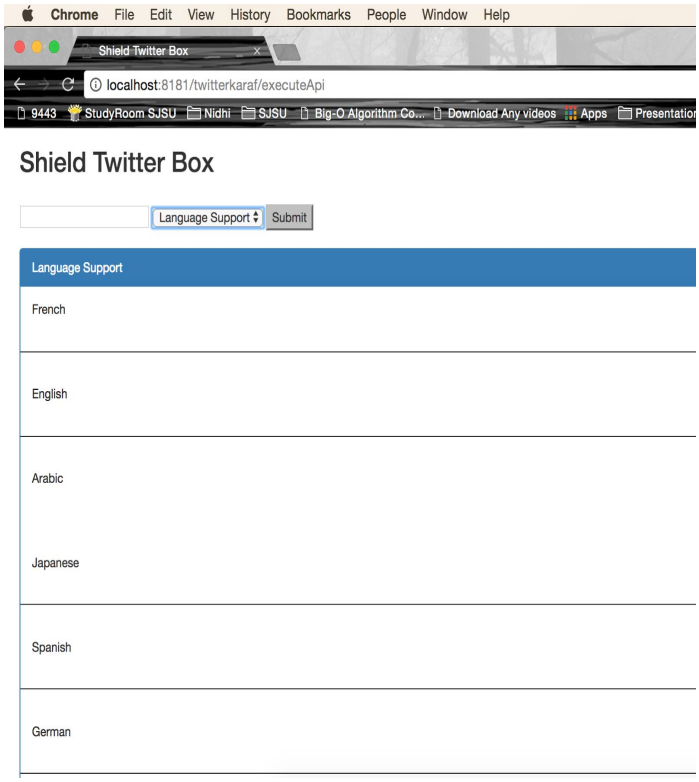
4. Search Hashtag



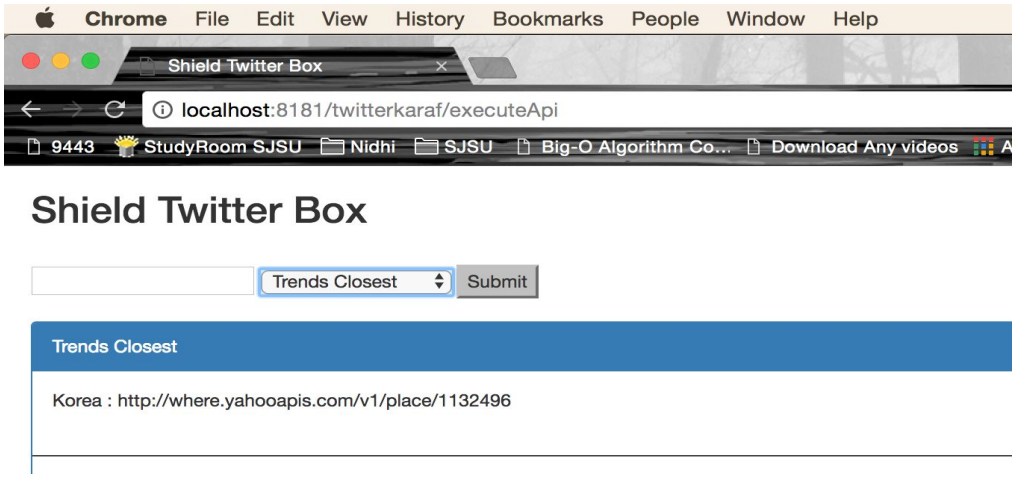
5. Home Timeline



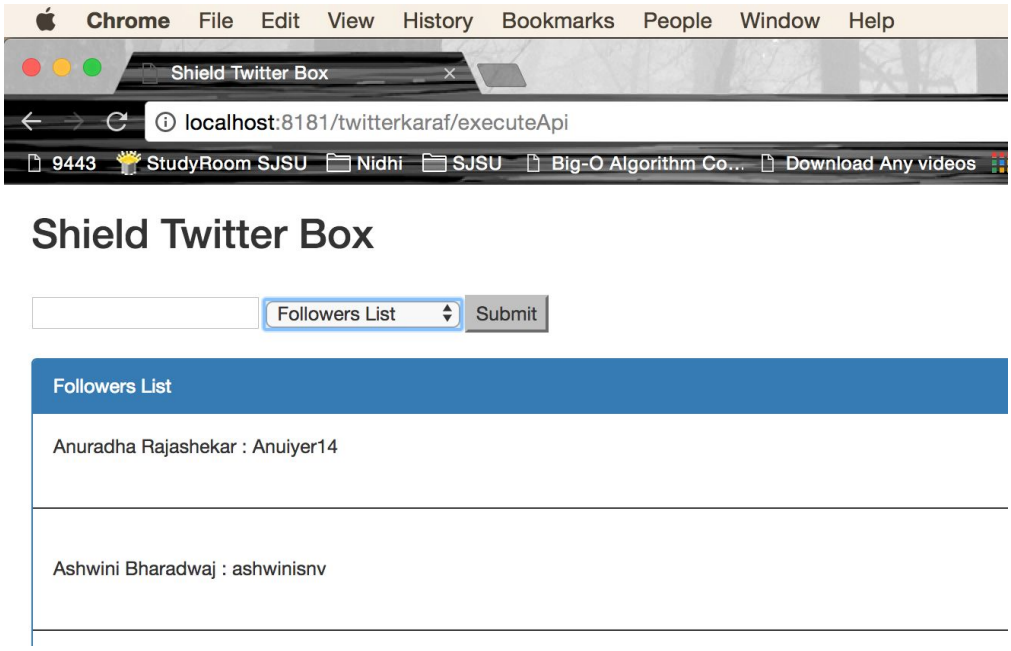
6. Language Support



7. Trends Closest



8. **Followers List**



JUnit Test Cases:

Twitter API results in different data sets each time we invoke the API. Test cases are written to verify if a successful data fetch is done. Below is the screenshot of unit test results.