Social Media Data Collection Design Document

# Instructions

## Server Installation Instructions

1. Clone the Django Server project from gitHub.  
 git clone -b master [git@github.com:ericwangtengyu/DjangoServer\_CyberBullying.git](mailto:git@github.com:ericwangtengyu/DjangoServer_CyberBullying.git)

2. Install Python and Django on your machine. Instructions:   
 <https://docs.djangoproject.com/en/1.6/intro/install/>

3. Run the django server. Cd to the directory of manage.py.(It is in the django Server project you   
 get from github).  
 Run Command: $python manage.py runserver (Your IP address):(Your port number).  
 Eg. $python manage.py runserver 127.0.0.1:8080

4. Clone the DataCollection project from gitHub.

git clone -b master [git@github.com:ericwangtengyu/DataCollection.git](mailto:git@github.com:ericwangtengyu/DataCollection.git)

5. Open the project using your favorite IDE like Eclipse.

6. Locate datacollection.properties file under the project and update the URL and port to the URL and port you use to start the django server.

## Android Installation Instructions

1. Clone the DataCollection project from gitHub

git clone -b master [git@github.com:ericwangtengyu/DataCollection.git](mailto:git@github.com:ericwangtengyu/DataCollection.git)

2. Obtain a copy of the FacebookSDK and import it into Eclipse

<https://developers.facebook.com/docs/android/>

3. Import DataCollection Android Side into your Android enabled Eclipse.

4. For the project, go to Properties, Android, and at the bottom remove the existing Facebook SDK, hit Add, and select the downloaded copy

5. As of right now, pending receiving code from Max, run the project on an Android device, and then copy the printed out hash and email it to either [tom-werner@uiowa.edu](mailto:tom-werner@uiowa.edu) or [tengyu-wang@uiowa.edu](mailto:tengyu-wang@uiowa.edu) and request that they add it to the list of hashes for the Facebook app on the developer site. The location to add them is under Settings.

6. Further instructions will follow once the full app code is added to gitHub and it is tested on a phone.

## Executing Instruction

1. Run the Message Siphon application on your phone.
2. Follow the instructions on your phone. Log in Facebook, twitter and enter your phone number to create a new user in our server.
3. Go to the DataCollection project. There are 2 entries in this project. CollectFacebookData.java and CollectTwitterData.java. Running on first entry will collect all the Facebook data of all the users stored in our server and then upload these data to our server. Running on the second entry will collect all twitter data.

## Testing Instructions

1. Log into Facebook as a user authenticated to work on the project, currently Tom Werner and Tengyu Wang. People requesting access need to friend one of the two and ask to be added as developers or administrators to the app and then accept the notification.
2. Once you are logged into Facebook, run Main.java from the tests package in the Server Side code. Follow the onscreen instructions, and be very careful while following the interactive Facebook tests.
3. After each test concludes, a summary of the results is printed out.

# Project Description

The goal of this project is to retrieve as much data about a user’s social media interactions as possible. Data will be collected from a user’s Facebook and Twitter accounts as well as their SMS messages. This is achieved using two mobile apps as well as a server application. Surveys will also be distributed to all users. After all data is collected it will be available on the webserver for easy queries and analysis.

# Visual Design

Mobile Authentication App

SMS Collection and Survey App

Webserver

- Authentication Data  
-Phone Number

- SMS messages  
-Surveys

Facebook Collection

- Facebook Authentication

- IM data  
- User activity

Twitter Collection

- Twitter data

- Twitter Authentication

Data Analysis

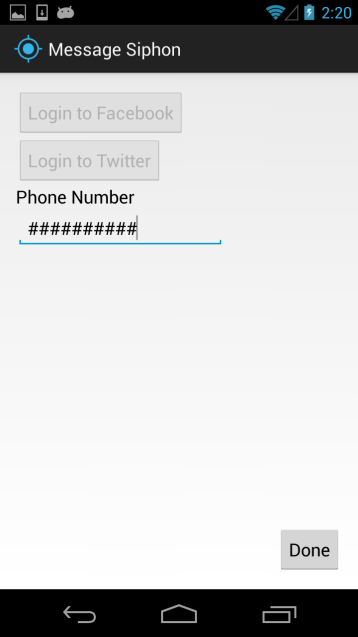
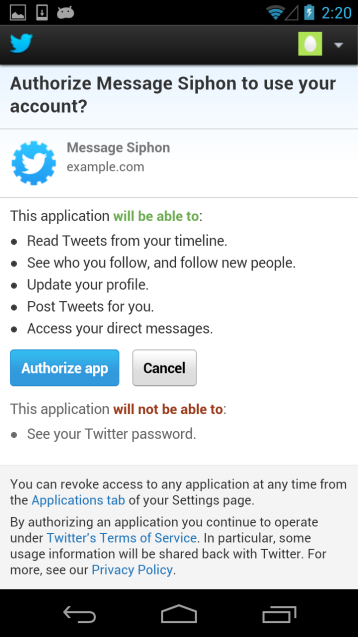
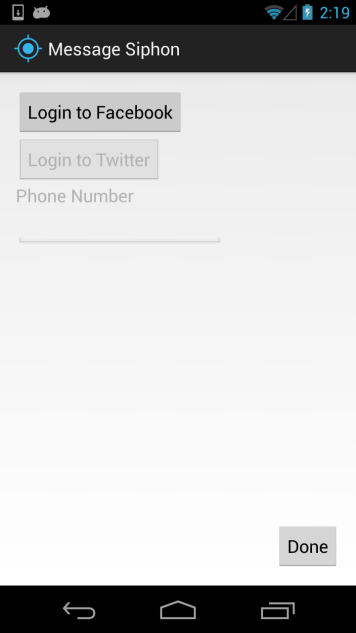
-Requested Data

# Mobile Applications

The mobile app side of the project is divided into two parts, the authentication and SMS/survey apps. This division is due to the nature of each app; the authentication app is one that will only need to be run once while the survey app will be run periodically. This makes it ideal for also collecting SMS messages and refreshing Facebook authentication.

## Authentication App

The authentication app collects the user’s Facebook and Twitter authentication tokens, which are then used by the collection applications to collect a user’s data. These tokens allow us to collect the user data without ever being exposed to the user’s passwords, as the respective API’s of Facebook and Twitter handle the logins. After collecting the authentication tokens the app prompts the user for their phone number. When all of the needed data is collected the user then uploads the data to the webserver, creating a new user on the server.



Here we can see the initial application prompting a Facebook login, the Twitter authentication page, and the app with all data obtained, ready to upload.

Unfortunately the Facebook token is only valid for 2 months; however the survey app will refresh the token when needed.

## SMS and Survey App

The SMS and Survey app is still in development. Currently the app can collect all SMS messages and successfully upload to the server, and the survey capability will be added as further details about the requirements of the surveys become available.

This app will also ensure that the Facebook token is continually valid.

# Data Collection Applications

The actual data collection programs operate on the server, accessing the webserver interface. They query the server for user’s access tokens and using those to open the API’s for Facebook and Twitter the applications collect all the information they can.

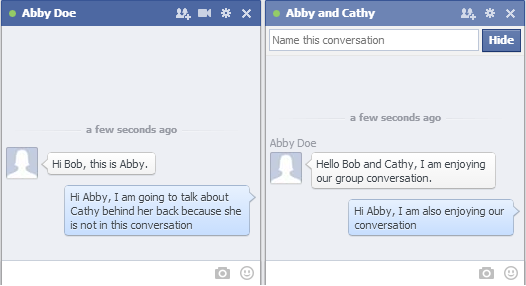
## Facebook Collection

The Facebook collection application collects two types of data on a user, their Facebook IM data and their timeline activity.

### Facebook IM Collection

For IM data the application will download all instant messages that have occurred since the last download. On the first run the application can download all messages going back to the first conversations the user has had since creating their Facebook. The data is grouped into Conversations and Messages. A Conversation is a group of all messages between a strict set of users. A message is a literal message between users in a conversation.

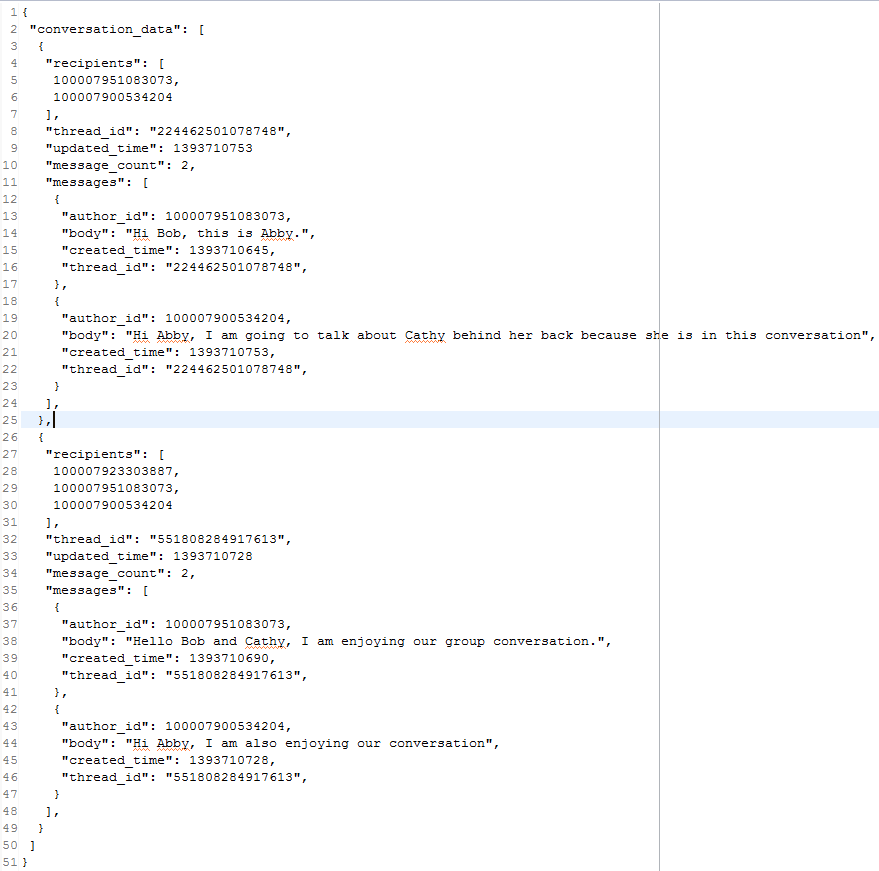
For example:



Here we have two different conversations, the first between Abby and Bob, and the second between Abby, Bob, and Cathy. Within each conversation we see messages, the text send to the other participants.

Our application can collect all of this data and stores it into the webserver’s database and as JSON files.

#### Example JSON Data



### Facebook Activity Collection

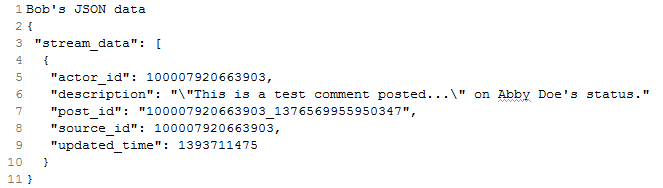
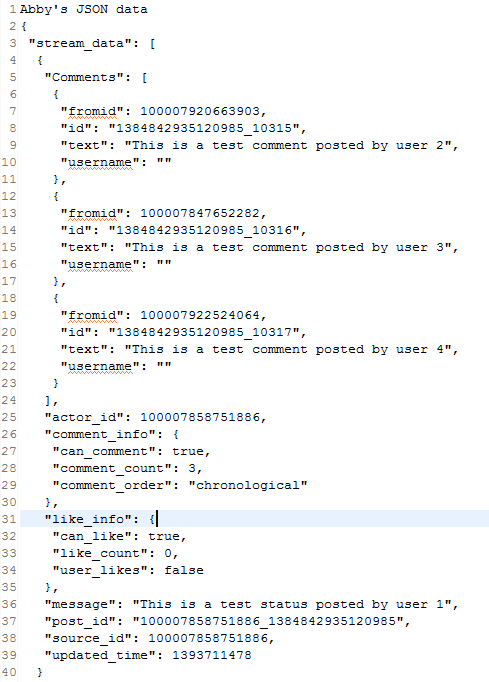
For Facebook Activity we download both a brief summary of a user’s activity, such as comments they have made, things they have liked, and timelines they have posted on. This information is not entirely complete and can be dependent on user privacy settings. As a result of this we also collect information on posts directly involving the user. For any post a user makes, such as statuses, links, photos, and posts on the user’s wall the application collects general post information on these as well as information on all of the comments on the posts.

For example:



Here we see that Abby has posted a status, although a photo or link would work as well. Bob, Cathy, and Doug post comments on this status, giving us an interaction between all four users.

#### Example JSON Data



## Twitter Collection

### Twitter Timeline Collection

Twitter Timeline collection consists of two parts, the user Timeline and the mentions Timeline.  
The user Timeline contains all the tweets on their homepage. In Twitter, all the tweets authored by a certain user will appear on his homepage.



The mention Timeline contains all the tweets in which the user is mentioned. It appears on the interactions tab of the user.



### TimeLine Conversation Building

A conversation in Twitter consists of a list of tweets. Each tweet in a conversation replies to its upper tweet. A conversation in Twitter is like below:



However, there is no Twitter API to get such kind of conversations directly. Fortunately, there is a JSON field called “inReplyToStatusID” which specifies which tweet the current tweet is a reply to. Therefore we are able to construct the conversation by combining tweets together.

We can build the Twitter Conversation in the following 3 steps.

1) Collect all the tweets from user timeline.

2) Collect all the tweets from mentions timeline.

3) We build conversations by combining 1) and 2).

4) After step 3, for all the conversations which are not closed, we use the JSON field “in\_reply\_to\_status\_id\_str” to get a tweet by tweet ID until there are no “open” conversations.

By saying a conversation is not closed, we mean the first tweet of this conversation is a reply to another tweet. However, we are not able to get that tweet by just combining user timeline and mentions timeline because this tweet is not made by the user and did not mention the user. Therefore, we need to get additional tweets to complete such conversations.

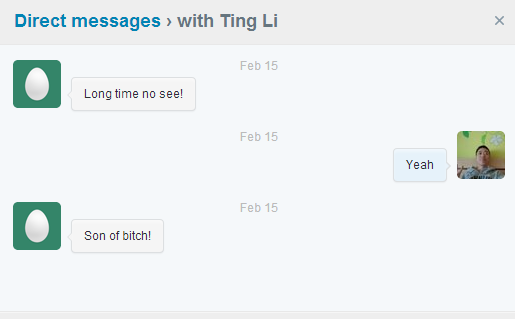
Something unique about Twitter conversations is that they are organized in a tree structure.

For example: A, B, C, and D are all tweets.  
 A->B->C and A->B->D are 2 conversations. Each conversation in Twitter is defined   
 to be a “Root” to “Leaf” path.

Because a message can appear in multiple Twitter conversations, now we are using  
 the combination of root message ID and leaf message ID to identify a conversation.

### Twitter DirectMessage Collection

Direct messages are the messages in Twitter’s inbox and outbox. The following is an example of direct messages in Twitter.



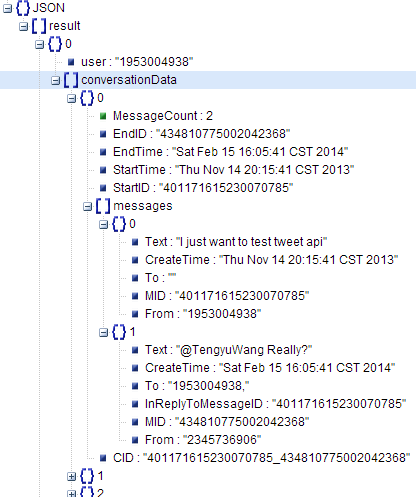
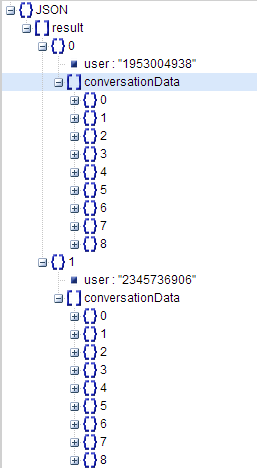
We collect all direct messages since the last data collection.

### Twitter Direct Message Conversation

A direct message conversation in Twitter is defined to be all the direct messages between 2 users.  
In the above example, we see a direct message conversation between the user and Ting Li. Direct message conversations in Twitter are not in a tree structure. Instead, each conversation is a list of messages.  
In a direct conversation, a conversation is defined to be all the messages between 2 users. Therefore, we can use sender and recipient to uniquely identify a conversation.

### Twitter JSON object

We organize the data we get from twitter into JSONObject and post the JSONObject to database.



# Available Data

Our webserver currently stores all collected data in a SQLite database and can be accessed through either HTTP GET requests or directly through the database. The SQLite database is highly flexible and supports filtering as well as python scripting.

For Facebook collection, the data is organized around the user. Each user has conversations and each conversation has messages. From here the entire text of a group of user’s interactions can obtained.