SOCIAL NETWORKING FUSION

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Yogesh Rajaram Isawe

FALL
2016
SOCIAL NETWORKING FUSION

A Project

by

Yogesh Rajaram Isawe

Approved by:

________________________________, Committee Chair
Dr. Jun Dai

________________________________, Second Reader
Dr. Jinsong Ouyang

_____________________________
Date

iii
Student: Yogesh Rajaram Isawe

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator
Dr. Ying Jin

Date

Department of Computer Science
Abstract
of
SOCIAL NETWORKING FUSION
by
Yogesh Rajaram Isawe

Popular social medias like Facebook and Twitter provide native mobile applications. These applications consume significant amount of memory on mobile devices and provide news-feeds only from their own data center. Users need to install dedicated applications to view news from each of such social medias. The ‘Social Networking Fusion’ combines news-feeds from multiple social medias into one application. It has separate view-sections that displays the news-feeds from each different social media service. It also has a fusion timeline that combines news-feeds from separate social accounts. The app also has a functionality to compose new posts or updates and upload them to Facebook and Twitter accounts.

The Social networking fusion is a native android mobile application. It is implemented using Java, XML, REST APIs and other third party Android-libraries. Performance of the social networking fusion app is evaluated by comparing memory, data and battery usage with that of Facebook and Twitter apps. All apps are tested
by creating a neutral lab environment on an Android device. This comparison is performed by simulating Monte Carlo’s random sampling method.

_______________________, Committee Chair
Dr. Jun Dai

_______________________
Date
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. DESIGN</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Application Focus</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Data Usage</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Fusion Timeline</td>
<td>4</td>
</tr>
<tr>
<td>2.4 Compose</td>
<td>5</td>
</tr>
<tr>
<td>2.5 Comparative Study</td>
<td>5</td>
</tr>
<tr>
<td>2.6 Use Cases</td>
<td>6</td>
</tr>
<tr>
<td>3. IMPLEMENTATION</td>
<td>12</td>
</tr>
<tr>
<td>3.1 User Interface</td>
<td>13</td>
</tr>
<tr>
<td>3.2 RecyclerView</td>
<td>15</td>
</tr>
<tr>
<td>3.3 Login Screens</td>
<td>17</td>
</tr>
<tr>
<td>3.4 News-feed Screens</td>
<td>18</td>
</tr>
<tr>
<td>3.5 GET Facebook News-feed</td>
<td>19</td>
</tr>
<tr>
<td>3.6 GET Twitter News-feed</td>
<td>20</td>
</tr>
<tr>
<td>3.7 Fusion Timeline</td>
<td>21</td>
</tr>
</tbody>
</table>
3.8 JSON Responses and Custom Objects ........................................... 23
3.9 Log-out .................................................................................. 24
3.10 Compose ............................................................................... 26

4. EVALUATION ........................................................................... 29
   4.1 Monte Carlo Sampling Rules ............................................... 29
   4.2 Monte Carlo Sampling Stages ............................................. 29
   4.3 Diagnosis ........................................................................... 30
   4.4 Readings ............................................................................ 30
   4.5 Observations ...................................................................... 33
   4.6 Disclosure ........................................................................... 33

5. CONCLUSION AND FUTURE WORK ............................................. 34

Appendix A. Android Manifest ...................................................... 35
Appendix B. App Build Gradle ...................................................... 36
Appendix C. App Class ................................................................. 37
Appendix D. MainActivity Class .................................................. 38
Appendix E. MainActivity Layout ............................................... 43
Appendix F. Twitter Fragment ....................................................... 44
Appendix G. Twitter Layout ........................................................ 50
Appendix H. Twitter News-feed Object ....................................... 51
Appendix I. Twitter RecyclerView Adapter ............................... 53
Appendix J. Facebook Fragment ................................................... 56
Appendix K. Facebook Layout ................................................................. 62
Appendix L. Facebook News-feed Object .................................................. 63
Appendix M. Facebook RecyclerView Adapter ........................................ 65
Appendix N. Fusion Timeline ..................................................................... 68
Appendix O. Fusion Timeline Layout .......................................................... 77
Appendix P. Fusion Timeline Object ............................................................ 78
Appendix Q. Fusion Timeline RecyclerView Adapter .................................. 82
Appendix R. Compose Fragment .................................................................. 86
References ................................................................................................. 89
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Battery Usage</td>
<td>30</td>
</tr>
<tr>
<td>2. Memory Usage</td>
<td>31</td>
</tr>
<tr>
<td>3. Data Usage</td>
<td>32</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Facebook Account Login Use Case</td>
<td>6</td>
</tr>
<tr>
<td>2. Twitter Account Login Use Case</td>
<td>7</td>
</tr>
<tr>
<td>3. Facebook News-feed Use Case</td>
<td>9</td>
</tr>
<tr>
<td>4. Twitter News-feed Use Case</td>
<td>10</td>
</tr>
<tr>
<td>5. Fusion Timeline Use Case</td>
<td>11</td>
</tr>
<tr>
<td>6. Facebook App Id Registration</td>
<td>12</td>
</tr>
<tr>
<td>7. Twitter App Id Registration</td>
<td>13</td>
</tr>
<tr>
<td>8. Login Screens</td>
<td>17</td>
</tr>
<tr>
<td>9. News-feed Screens</td>
<td>18</td>
</tr>
<tr>
<td>10. Fusion Timeline Screen</td>
<td>21</td>
</tr>
<tr>
<td>11. Logout Options</td>
<td>24</td>
</tr>
<tr>
<td>12. Compose Screen</td>
<td>26</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

Social networking is very popular nowadays. Screening news-feeds and sharing posts or updates are the principal functionalities of social networking. Facebook and Twitter are the most popular social networking services. Almost everyone has accounts on these social medias. Facebook and Twitter have websites as the main gateway for social networking. They also have mobile applications that users commonly use to browse through the latest updates and posts along with support for other functionalities. Users of Facebook and Twitter can use either websites or mobile apps for social networking. Smartphone applications are very popular for providing such information at fingertips and on a glance.

Facebook’s mobile app provides a news-feed containing posts and updates from Facebook only. Similarly, Twitter’s mobile app provides a news-feed of posts and updates from Twitter only. Users with accounts on both social medias have to download their respective apps to view all news-feeds. It is inconvenient to change between apps to view news-feeds from multiple accounts. Another drawback is that Facebook and Twitter apps require large on-board storage space and smartphones have a limited amount of that space.
Hence, there is need for a common platform that provides most commonly used functionalities of social medias, like viewing news-feeds and posting updates. The social networking fusion app displays news-feeds from both Facebook and Twitter social medias. It has a fusion timeline that displays posts and updates from both social medias. The fusion timeline displays unambiguous posts and updates in chronological order. The application also has a functionality to compose new posts or updates and send them to either one or both Facebook and Twitter accounts.

Performance of the social networking fusion application is evaluated against Facebook and Twitter apps by comparing memory, data and battery usage to display same amount of posts and updates. All applications are tested by creating a neutral lab environment on an Android device. This comparison is performed by simulating Monte Carlo’s random sampling method.

Android is the most popular operating system for smartphones. It has vast repositories of API’s and supporting libraries. Facebook and Twitter also provide SDK’s that integrates with Android apps. Thus, the social networking fusion is implemented as an Android app. The app displays Facebook and Twitter’s news-feeds using Android’s tab layout. Users can control flow of data to the social networking fusion app by specifying number of posts and updates to display. Fusion timeline of the app combines and displays posts and updates from news-feeds of both
Facebook and Twitter. It is a chronologically sorted unambiguous list of posts and updates. Users can compose new posts or updates from the app and upload them to either one or both Facebook and Twitter accounts.
Chapter 2
DESIGN

2.1 Application Focus
The social networking fusion app is designed to provide information at a glance. News-feeds are displayed to users on successful logins to Facebook and Twitter accounts. Fusion timeline is available only if the user is logged in with both the accounts. Users may choose to logoff from their accounts at any time from the options menu.

2.2 Data Usage
User controls data usage by specifying number of posts and updates to be displayed in news-feeds. When number of posts and updates are not specified, by default the app displays 20 posts and updates in news-feeds and 40 posts and updates in fusion timeline.

2.3 Fusion Timeline
Fusion timeline displays posts and updates from both Facebook and Twitter news-feeds. Fusion timeline is chronologically sorted using a custom comparator that compares creation time of posts and updates.
Users with accounts on both Facebook and Twitter social medias may post same post or update on both social medias. Fusion timeline checks for these ambiguous posts and updates using a specific implementation of selection algorithm. The custom algorithm checks for similarity in the contents of the posts or updates posted by the same user on different accounts. If the ambiguity is found, then the older post or update is removed and only latest occurrence of the post or update is added to the fusion timeline view.

2.4 Compose

The app allows users to compose new posts or updates. The composed post or update can be uploaded to either one or both Facebook and Twitter accounts.

2.5 Comparative Study

Performance of the social networking fusion app is evaluated by comparing data, memory and battery usage with Facebook and Twitter apps for displaying newsfeeds with same number of posts. Monte Carlo’s random sampling technique is used for comparative study [1]. The simulated Monte Carlo’s technique captures data, memory and battery usage by apps for the same number of posts. Information collected by this simulation provides insight on how data, memory and battery usage can be minimized by using the social networking fusion app for displaying newsfeeds from Facebook and Twitter social medias.
2.6 Use Cases

Use Case 1: User login for Facebook account

Primary Actor: Mobile app user

Brief: Social networking fusion app requires valid access token to make RESTful service requests to Facebook server. Access token is generated for users on successful log-in with permissions to access data. The app checks for valid access token when launched. If no access token is found, app displays login fragment to user.

Precondition: Access token and permissions are null.

Post condition: On success, valid access token is generated for user. Permissions to request data are acquired by app.
Use Case 2: User login for Twitter account

Primary Actor: Mobile app user

Brief: Social networking fusion app requires valid session token to make RESTful service requests to Twitter server. Session token is generated for users on successful log-in with permissions to access data. The app checks for valid session token when launched. If no session token is found, app displays login fragment to user.

![Diagram of Twitter Account Login Use Case](image)

Figure 2. Twitter Account Login Use Case

Precondition: Session token and permissions are null.

Post condition: On successful login session token is generated for user. Permissions to request data from Twitter server are acquired by the app.
Use Case 3: Log out from Facebook account
Primary Actor: Authenticated app user
Brief: Users can log out from Facebook account from options menu in app.
Precondition: App contains valid access token and permissions.
Post condition: Access token, permissions and data are removed from app.

Use Case 4: Log out from Twitter account
Primary Actor: Authenticated app user
Brief: Users can log out from Twitter account from options menu in app.
Precondition: App contains valid session token and permissions.
Post condition: Session token, permissions and data are removed from app.

Use Case 5: Request for Facebook news-feed
Primary Actor: Authenticated app user
Brief: Social networking fusion app gives user control over data. User can specify number of updates and posts to get from Facebook server.
Precondition: Valid access token with permissions to request data from Facebook server.

Post condition: JSON response is obtained after successful completion of GET request. JSON response contains posts and updates from Facebook server.

Use Case 6: Request for Twitter news-feed

Primary Actor: Authenticated app user

Brief: Social networking fusion app gives user control over data. User can specify number of updates and posts to get from Twitter server.
Precondition: Valid session token with permissions to request data from Twitter server.

Post condition: JSON response is obtained after successful completion of GET request to Twitter server. JSON response contains posts and updates from Twitter server.

Use Case 7: Request for fusion timeline view
Primary Actor: Authenticated app user
Brief: Social networking fusion app gives user control over data. User can specify number of updates and posts to display in fusion timeline view.
Precondition: App contains both Facebook access token and Twitter session token to request data from both Facebook and Twitter servers.

Post condition: JSON responses with posts and updates from both Facebook and Twitter servers.
Chapter 3

IMPLEMENTATION

The social networking fusion application combines news-feeds from Facebook and Twitter’s social medias. To receive news-feeds from Facebook and Twitter server, app makes asynchronous HTTP GET requests via REST API’s. Facebook and Twitter provides Android SDKs that support these requests. To integrate these SDK’s with Android app, an app id must be created on Facebook and Twitter’s developer sites. In addition, these SDKs should be imported into Android app’s build system. Figure 6 and Figure 7 shows app ids created on Facebook and Twitter’s developer sites respectively.

![App ID Registration](image)

Figure 6. Facebook App Id Registration
3.1 User Interface

Android’s tab layout is used to display fixed number of tabs in horizontal orientation [2]. It is included with ‘android.support.design.widget’ library of android. Tabs organize content at high level by creating different view sections [3]. Tabs are created using ViewPager class of Android’s ‘android.support.v4.view’ library [4]. The ViewPager provides animations to swipe left and right between tabs and they can efficiently organize tabs for the app. ViewPager requires custom implementation of PagerAdapter that generates pages to display. These pages are created using Fragments. Fragments are reusable components with their own lifecycle and views. They are contained by activity and represent operations and interfaces running within the activity [5]. Following code snippet shows implementation of ViewPager in XML file [4]. The code snippet is reusing code from Android documentation based on Android open source project license [6].
FragmentPagerAdapter class is implemented to create a PagerAdapter for ViewPager that creates tab layout for application. This implementation of FragmentPagerAdapter keeps each page fragment in memory to load them immediately when user visits that page. Methods getItem(int tab_id) and getCount() of adapter returns fragment instance to be loaded at tab_id position in the tab layout and number of pages to create in tab layout respectively. Following code snippet shows implementation of FragmentPagerAdapter [6][7].
3.2 RecyclerView

RecyclerView is used to display news-feeds from Facebook, Twitter and to display fusion timeline. Following code snippet shows declaration of RecyclerView for Twitter news-feed [6][8]. Similar declarations are made for Facebook news-feed and fusion timeline.

```java
@override
public CharSequence getPageTitle(int position) {
    switch (position) {
    case 0:
        return "Twitter";
    case 1:
        return "Facebook";
    case 2:
        return "Fusion";
    case 3:
        return "Compose";
    }
    return null;
}
```

```xml
<android.support.v7.widget.RecyclerView
    android:id="@+id/TwRecyclerViewIn"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@+id/limitTextIn"
    android:background="@color/tw_blue_default"
    android:paddingTop="5dp"
    android:scrollbars="vertical" />
```
RecyclerView displays list of complex items. These items can display custom objects from data sets. List items can be updated at runtime for network events like updating images in image-view widgets of list item. Item-views in RecyclerView are recycled to display similar information. RecyclerView uses layout managers that positions item-views inside RecyclerView. Data is provided by adapters of datasets to the layout manager. The layout manager decides whether to reuse view object or to create new one [8]. Different list-items are created to display posts and updates from Facebook and Twitter. These list-items can display posts or updates with information like name of creator, time of creation, message, hashtags and image. Image is downloaded and displayed only when it is visible to user. This is achieved by using android library called ‘Picasso library’. Following code snippet shows implementation of onBindViewHolder method [6][9]. This method updates views that are currently visible to user.

```java
@override
public void onBindViewHolder(MyViewHolder holder, int position) {
    TweetObject curTweet = tweets.get(position);
    holder.twCreator.setText(curTweet.getUser_name() + " on " + curTweet.getTime());
    holder.twTweet.setText(curTweet.getPost() + "n" + curTweet.getMedia_url);

    if (curTweet.getMedia_url() != null && !curTweet.getMedia_url().isEmpty()) {
        holder.tweetImage.setVisibility(View.VISIBLE);
        Picasso.with(context).load(curTweet.getMedia_url()).
            error(ic_launcher)
            .placeholder(ic_launcher)
            .into(holder.tweetImage);
    } else {
        holder.tweetImage.setVisibility(View.GONE);
    }
}
```
3.3 Login Screens

Users need to login into their Facebook and Twitter accounts to display news-feeds. When application is launched it checks for the login information using AccessToken class of Facebook SDK and Session class of Twitter SDK. If user is not authenticated, then login fragment is displayed to user. By logging in their Facebook and Twitter accounts users grant permissions to retrieve information and perform actions on their accounts. These permissions are displayed to users on login page and users can then choose to grant them or cancel the login process. Figure 8 shows login screens for Facebook and Twitter.
3.4 News-feed Screens

After successful login into accounts users are redirected to news-feed fragments. News-feed fragments have EditText layout widget where users can input number of posts or updates to be displayed in the news-feed. If number of posts and updates to display is not provided, then Facebook and Twitter news-feed fragments retrieve 20 posts or updates from user’s accounts while fusion timeline retrieves total of 40 posts or updates. Figure 9 shows news-feed screens for Twitter and Facebook accounts.

![News-feed Screens](image_url)

Figure 9. News-feed Screens
3.5 GET Facebook News-feed

Facebook news-feed fragment makes GraphRequest to Facebook using ‘HttpMethod.GET’ REST call to ‘me/feed’ endpoint. Following code snippet shows GraphRequest API call [10]. The code snippet is reusing code from Facebook documentation based on developer license [11].

```java
private void getFbData(Bundle params) {
    new GraphRequest(
        AccessToken.getCurrentAccessToken(),
        "me/feed",
        params,
        HttpMethod.GET,
        new GraphRequest.Callback() {
            public void onCompleted(GraphResponse response) {
                JSONObject fbResponse = response.getJSONObject();
                parseResponse(fbResponse);
            }
        }
    ).executeAsync();
}
```

This is an asynchronous call that when successful returns a JSON response file. Parameters returned in JSON response are set using Bundle class to add them in the REST call. Code to parse JSON can be found in Appendix F, J and N.
3.6 GET Twitter News-feed

Twitter SDK provides TwitterCore kit to make REST service calls to Twitter via TwitterApiClient class. Valid session token is required for this REST service call, it is obtained when user logs in with Twitter account into the app. Twitter news-feed fragment makes asynchronous request to Twitter using a getHomeTimeline() method of Hometimeline interface. Class MyTwitterApiClient in the app extends TwitterApiClient class of Twitter to implement Hometimeline interface. On successful completion of REST call to Twitter, it sends a JSON response file. This JSON response contains list of posts and updates. Following code snippet shows GET REST API call to Twitter [12]. It is available to developers under Twitter developer terms [13].

```java
private void getTwData(String s) {
    TwitterSession session = Twitter.getSessionManager()
        .getActiveSession();
    new MyTwitterApiClient(session).getHomeTimeline().show(s)
        .enqueue(new Callback<List>() {
            @Override
            public void success(Result<List> result) {
                List array = new ArrayList(result.data);
                parseJson(array);
            }
            @Override
            public void failure(TwitterException exception) {
            }
        });
}
```
3.7 Fusion Timeline

Fusion timeline consists of posts and updates from both Facebook and Twitter newsfeeds. Figure 10 shows fusion timeline screen. It is available when user is logged in with both Facebook and Twitter accounts.

![Fusion Timeline Screen](Image)
Fusion timeline displays unambiguous list of posts and updates sorted in a chronological order. The sorting is achieved by overriding sort method of Collections interface to compare Data fields of objects. Following code snippet shows implementation of Comparator to compare Date fields.

```java
public static Comparator<FusionObject> PostDate = new Comparator<FusionObject>() {
    @Override
    public int compare(FusionObject lhs, FusionObject rhs) {
        Date lhsTime = new Date(lhs.getComTime());
        Date rhsTime = new Date(rhs.getComTime());
        return lhsTime.compareTo(rhsTime);
    }
};
```

Fusion timeline displays unambiguous list of posts and updates. It is made unambiguous by removing old version of similar posts or updates posted by same user on different accounts. For this a customized selection algorithm is created. This algorithm compares contents of posts or updates posted on different accounts by the same user. If ambiguity is found, then older post or update is removed and only recent post or update is displayed. Following code snippet shows processFeed() method that implements this customized selection algorithm.
3.8 JSON responses and Custom Objects

JSON responses from Facebook and Twitter servers contain arrays of JSON objects where each object represents a post or update. Each JSON object is an attribute-value pair. To retrieve information from JSON responses, they are parsed using the parseJSON() method. Refer Appendix F and J for Facebook and Twitter JSON parsing codes respectively. The parseJSON() method stores information from JSON
file into list of custom-objects. Separate custom-objects are created to represent posts or updates from Facebook and Twitter news-feeds. As the data returned by Facebook server is different than that returned by Twitter server.

Object representing posts or updates from fusion timeline is constructed by combining custom-objects of Facebook and Twitter. Appendix H, L and P contains source codes of custom-objects for Facebook, Twitter news-feeds and fusion timeline.

3.9 Log-out

The social networking fusion app users can logout from any of their accounts from sign-out options menu. It removes the user’s information and news-feed screens from the app. Figure 11 shows logout options menu.

![Logout Options](image)

Figure 11. Logout Options
LoginManager class of Facebook SDK contains active access token and a logout() method. This logout() method removes Facebook account user’s information and the access token.

Twitter SDK contains SessionManager class with clearActiveSession() method. This clearActiveSession() method removes Twitter account user’s information and session token. Following code snippet shows implementation for logout options menu. The code snippet is reusing code from Android documentation based on Android open source project license [6].

```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.facebook_logout) {
        LoginManager.getInstance().logOut();
        finish();
        startActivity(getIntent());
        return true;
    }
    if (id == R.id.twitter_signout) {
        Twitter.getSessionManager().clearActiveSession();
        Toast.makeText(getApplicationContext(), "All accounts are cleared", Toast.LENGTH_SHORT).show();
        finish();
        startActivity(getIntent());
    }
    return super.onOptionsItemSelected(item);
}
```
3.10 Compose

Users of the social networking fusion app can compose new posts or updates in the compose section. The composed post or update can be uploaded to either one or both Facebook and Twitter accounts. Figure 12 shows compose screen.

Figure 12. Compose Screen
Twitter SDK contains TwitterApiClient class and SessionServices interface. The SessionServices interface contains methods to upload posts and updates to Twitter server. Following code snippet shows how to upload tweets to the Twitter server. Following code snippet shows POST REST API call to Twitter [12]. It is available to developers under Twitter developer terms [13].

```java
private void postToTwitter(Session activeSession, String s) {
    if (activeSession != null) {
        TwitterApiClient twitterApiClient = TwitterCore.getInstance().getApiClient();
        StatusesService statusesService = twitterApiClient.getStatusesService();
        Call<Tweet> call = statusesService.update(s, null, null, null, null, null, null, null, null);
        call.enqueue(new Callback<Tweet>() {
            @Override
            public void success(Result<Tweet> result) {
                Toast.makeText(getContext(), "Tweet sent!", Toast.LENGTH_LONG).show();
            }

            @Override
            public void failure(TwitterException exception) {
                Toast.makeText(getContext(), "Something went wrong!", Toast.LENGTH_LONG).show();
            }
        });
    } else {
        Toast.makeText(getContext(), "Login Required!", Toast.LENGTH_LONG).show();
    }
}
```
The social networking fusion app makes POST request to Facebook’s RESTful API using GraphRequest class to upload posts or updates. It is a HttpMethod that requires active access token. Following code snippet shows POST request to Facebook’s GraphRequest API [14]. The code snippet is reusing code from Facebook documentation based on developer license [11].

```java
private void postToFacebook(AccessToken loggedIn, String s) {
    if (loggedIn != null) {
        Bundle params = new Bundle();
        params.putString("message", s);
        new GraphRequest(accessToken.getCurrentUserAccessToken(),
                        "/me/feed",
                        params,
                        HttpMethod.POST,
                        new GraphRequest.Callback() {
                            public void onCompleted(GraphResponse response) {
                                Toast.makeText(getActivity(), "Post uploaded to!", Toast.LENGTH_LONG).show();
                            }
                        }).executeAsync();
    } else {
        Toast.makeText(getActivity(), "Login Required!", Toast.LENGTH_LONG).show();
    }
}
```
Chapter 4
EVALUATION
Monte Carlo’s sampling technique is used to record readings and then to evaluate data obtained. Facebook, Twitter and the social networking fusion apps are compared to display same number of posts and updates with similar medias. These apps are tested in a neutral testing environment to evaluate memory usage, data usage and battery usage.

4.1 Monte Carlo Sampling Rules
1. Readings are noted for same amount of posts and updates.
2. Readings are taken using uniform testing environment i.e. with same device.
3. Readings noted are to display same posts and updates with similar media.

4.2 Monte Carlo Sampling Stages
1. App loads from start i.e. launcher activities onCreate() method.
2. App downloads initial data.
3. Posts and updates are displayed on screen.
4. App is closed and removed from mobile devices cached memory.
4.3 Diagnosis

Information is obtained using diagnosis tools provided by the Android device:

1. Memory usage: Settings->Storage->Apps->App
2. Data usage: Settings->Data Usage->App
3. Battery usage: Settings->Battery->App

4.4 Readings

Information collected for downloading and displaying same amount content at different instances.

Settings: Auto play option turned off on both Facebook and Twitter apps

SNF: Social Networking Fusion app

<table>
<thead>
<tr>
<th>Sample</th>
<th>Facebook (mAh)</th>
<th>Twitter (mAh)</th>
<th>SNF (mAh)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1</td>
<td>8</td>
<td>9.5</td>
<td>8.5</td>
</tr>
<tr>
<td>Sample 2</td>
<td>9</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Sample 3</td>
<td>8.2</td>
<td>8.7</td>
<td>9</td>
</tr>
<tr>
<td>Sample 4</td>
<td>8.7</td>
<td>8.25</td>
<td>8</td>
</tr>
<tr>
<td>Sample 5</td>
<td>9</td>
<td>8.5</td>
<td>8.2</td>
</tr>
<tr>
<td>Average</td>
<td>8.58</td>
<td>8.79</td>
<td>8.34</td>
</tr>
</tbody>
</table>
Table 2. Memory Usage

<table>
<thead>
<tr>
<th></th>
<th>Facebook</th>
<th>Twitter memory</th>
<th>SNF memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>1220 kb</td>
<td>0880 kb</td>
<td>0344 kb</td>
</tr>
<tr>
<td>Set 2</td>
<td>2030 kb</td>
<td>1190 kb</td>
<td>0280 kb</td>
</tr>
<tr>
<td>Set 3</td>
<td>1920 kb</td>
<td>0790 kb</td>
<td>0510 kb</td>
</tr>
<tr>
<td>Set 4</td>
<td>0940 kb</td>
<td>0690 kb</td>
<td>0290 kb</td>
</tr>
<tr>
<td>Set 5</td>
<td>2090 kb</td>
<td>1020 kb</td>
<td>0600 kb</td>
</tr>
<tr>
<td>Set 6</td>
<td>1750 kb</td>
<td>0700 kb</td>
<td>0450 kb</td>
</tr>
<tr>
<td>Set 7</td>
<td>1600 kb</td>
<td>0790 kb</td>
<td>0320 kb</td>
</tr>
<tr>
<td>Set 8</td>
<td>1470 kb</td>
<td>0800 kb</td>
<td>0710 kb</td>
</tr>
<tr>
<td>Set 9</td>
<td>1500 kb</td>
<td>0950 kb</td>
<td>0670 kb</td>
</tr>
<tr>
<td>Set 10</td>
<td>2010 kb</td>
<td>0760 kb</td>
<td>0520 kb</td>
</tr>
<tr>
<td>Average</td>
<td>1653 kb</td>
<td>857 kb</td>
<td>469.4 kb</td>
</tr>
</tbody>
</table>
Table 3. Data Usage

<table>
<thead>
<tr>
<th>Set</th>
<th>Facebook data</th>
<th>SNF data</th>
<th>Twitter data</th>
<th>SNF data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>162.73 kb</td>
<td>84 kb</td>
<td>83.64 kb</td>
<td>34 kb</td>
</tr>
<tr>
<td>Set 2</td>
<td>317 kb</td>
<td>143 kb</td>
<td>93 kb</td>
<td>43 kb</td>
</tr>
<tr>
<td>Set 3</td>
<td>260 kb</td>
<td>127 kb</td>
<td>78 kb</td>
<td>40 kb</td>
</tr>
<tr>
<td>Set 4</td>
<td>331.82 kb</td>
<td>142.14 kb</td>
<td>116.36 kb</td>
<td>43.14 kb</td>
</tr>
<tr>
<td>Set 5</td>
<td>195 kb</td>
<td>98 kb</td>
<td>105 kb</td>
<td>32 kb</td>
</tr>
<tr>
<td>Set 6</td>
<td>389.20 kb</td>
<td>147.20 kb</td>
<td>201.82 kb</td>
<td>47.20 kb</td>
</tr>
<tr>
<td>Set 7</td>
<td>189 kb</td>
<td>99 kb</td>
<td>169 kb</td>
<td>39 kb</td>
</tr>
<tr>
<td>Set 8</td>
<td>203.64 kb</td>
<td>158.10 kb</td>
<td>79 kb</td>
<td>38.10 kb</td>
</tr>
<tr>
<td>Set 9</td>
<td>172.73 kb</td>
<td>86 kb</td>
<td>95.35 kb</td>
<td>36 kb</td>
</tr>
<tr>
<td>Set 10</td>
<td>306 kb</td>
<td>148 kb</td>
<td>197 kb</td>
<td>38.10 kb</td>
</tr>
<tr>
<td>Average</td>
<td>252.03 kb</td>
<td>123.50 kb</td>
<td>115.23 kb</td>
<td>39.69 kb</td>
</tr>
</tbody>
</table>
4.5 Observations

The information obtained by sampling storage usage, data usage and battery usage of all apps for displaying same content provides sufficient data that is evaluated as following.

1. Facebook and Twitter apps use more memory than that of the social networking fusion app.

2. Facebook and Twitter apps store data on devices on-board storage space. The social networking fusion app does not store data on devices on-board storage space.

3. Facebook and Twitter uses more data than the social networking fusion app to display similar content.

4. Battery usage by the social networking app is less compared to Facebook and Twitter apps.

4.6 Disclosure

The social networking fusion app proposes a scalable industrial solution to bring these popular social networking services together in a common platform. The app allows users to access and view their social medias through one app. It also depends on the data made available by Facebook and Twitter services. Facebook and Twitter apps are developed to provide complete social networking experience for their social medias.
Chapter 5

CONCLUSION AND FUTURE WORK

The social networking fusion app displays news-feeds from Facebook and Twitter social medias in one app. Fusion timeline combines posts and updates from news-feeds of both Facebook and Twitter. It displays unambiguous posts and updates in chronological order.

The app successfully displays information at a glance and allows users to control data by limiting information displayed in news-feeds. Users can compose new posts or updates and upload to either one or both Facebook and Twitter accounts. The social networking fusion app also has minimal on-board storage requirement and it does not store any information on devices on-board storage space.

There is always scope for improvements in any software program. This application is designed for combining social medias into one application. Currently it combines Facebook and Twitter social networking services. In the future, this app can include more social medias, given that the social medias make their news-feeds available to app developers. User interface can be improved by including additional information fields, and compose screen can be customized to upload multimedia to the accounts.
Appendix A

ANDROID MANIFEST

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.yogidevelopers.yogiaandroid.socialfusion">

    <uses-permission android:name="android.permissionINTERNET" />

    <application
        android:name=".main.App"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Social Fusion"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:node="replace">

        <meta-data
            android:name="com.facebook.sdk.ApplicationId"
            android:value="511298599079648" />

        <activity
            android:name=".main.MainActivity"
            android:windowSoftInputMode="stateHidden"
            android:label="Social Fusion"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity>
            android:name="com.twitter.sdk.android.core.identity.OAuthActivity" />

        <activity
            android:name="com.facebook.FacebookActivity"
            android:configChanges="keyboard|keyboardHidden|screenLayout|screenSize|orientation"
            android:label="Social Fusion"
            android:theme="@android:style/Theme.Translucent.NoTitleBar" />

        <activity
            android:name="com.facebook.CustomTabActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
            </intent-filter>
        </activity>

        <data android:scheme="511298599079648" />
    </application>

    <provider
        android:name="com.facebook.facebookcontentprovider"
        android:authorities="com.facebook.app.facebookcontentprovider511298599079648"
        android:exported="true" />

    <meta-data
        android:name="io.fabric.ApiKey"
        android:value="8b3e5efadd3f63ed8b6802d791d86a170e63d22b" />

</manifest>
```
```kotlin
buildscript {
    repositories {
        maven { url 'https://maven.fabric.io/public' }
    }
    dependencies {
        classpath 'io.fabric.tools:gradle:1.+'
    }
}
apply plugin: 'com.android.application'
apply plugin: 'io.fabric'

android {
    compileSdkVersion 23
    buildToolsVersion "24.0.3"

    defaultConfig {
        applicationId "com.yogidevelopers.yogia_android_socialfusion"
        minSdkVersion 15
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
repositories {
    mavenCentral()
    maven { url 'https://maven.fabric.io/public' }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile 'com.twitter.sdk.android.twitter:2.1.1@aar'
    compile 'com.android.support:appcompat-v7:23.4.0'
    compile 'com.android.support:design:23.4.0'
    compile 'com.android.support:support-v4:23.4.0'
    compile 'com.facebook.android:facebook-android-sdk:4.5'
    compile 'com.squareup.picasso:picasso:2.5.2'
    testCompile 'junit:junit:4.12'
}
```
Appendix C

APP CLASS

```java
package com.yogidevelopers.yogiamdroid.socialfusion.main;
import android.app.Application;
import com.twitter.sdk.android.Twitter;
import com.twitter.sdk.android.core.TwitterAuthConfig;
import io.fabric.sdk.android.Fabric;

/**
 * Created by yogi on 10/2/16.
 */
public class App extends Application {

    private static final String TWITTER_KEY = "9R0DwiMBjAJaHfYXCB5SHsJMB";
    private static final String TWITTER_SECRET = "MnBBz3Jv68kkTyLCZU69mSiaH8RZwoXGbrRVSrkvXjgVdXew";

    private static App singleton;

    public static App getInstance() {
        return singleton;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        singleton = this;
        TwitterAuthConfig authConfig = new TwitterAuthConfig(TWITTER_KEY, TWITTER_SECRET);
        Fabric.with(this, new Twitter(authConfig));
    }
}
```
Appendix D

MAINACTIVITY CLASS

```java
package com.yogidevelopers.yogiaandroid.socialfusion.main;
import android.content.Intent;
import android.support.design.widget.TabLayout;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;
import com.facebook.FacebookSdk;
import com.facebook.login.LoginManager;
import com.twitter.sdk.android.Twitter;
import com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.fcbb.facebook.FBLoginFragment;
import com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.fcbb.recyclerview/Object;
import com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.fusn.recyclerView.FusionObject;
import com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.fusn.FusionOptFrag;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {
    ArrayList<Object> frqbf;
    ArrayList<TweetObject> frqtw = new ArrayList<>();
    String twlist, fblist;
    ArrayList<FusionObject> fusionFeed;

    /**
     * The (PagerAdapters) that provide
     * fragments for each of the sections. We use a
     * (PagerAdapter) derivative, which will keep every
     */
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    // Create the adapter that will return a fragment for each of the three
    // primary sections of the activity.
    mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());
    mViewPager.setAdapter(mSectionsPagerAdapter);
    // Set up the ViewPager with the sections adapter.
    mViewPager = (ViewPager) findViewById(R.id.container);
    mViewPager.setAdapter(mSectionsPagerAdapter);
    LayoutInflater layout = (LayoutInflater) findViewById(R.id.tabs);
    viewPager.setAdapter(mViewpager);
    FacebookSdk.sdkInitialize(getApplication());
    // TODO
    try {
        PackageInfo info = getPackageManager().getPackageInfo("com.yogidevelopers.yogandroid.socialfusion", PackageManager.GET_SIGNATURES);
        for (Signature signature : info.signatures) {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(signature.toByteArray());
            Log.d("KeyHash", Base64.encodeToString(md.digest(), Base64.DEFAULT));
        } catch (PackageManager.NameNotFoundException e) {
            // catch (NoSuchAlgorithmException e) {
        } catch (SignatureException e) {
        }
    }
}
```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here.
    // The action bar will automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    //oinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        if (id == R.id.facebook_logout) {
            LoginManager.getInstance().logOut();
            finish();
            startActivity(getIntent());
            return true;
        }
        if (id == R.id.twitter_signout) {
            Twitter.getSessionManager().clearActiveSession();
            Toast.makeText(getApplicationContext(), "All accounts are cleared", Toast.LENGTH_SHORT).show();
            finish();
            startActivity(getIntent());
        }
    }
    return super.onOptionsItemSelected(item);
}

/**
 * A placeholder fragment containing a single view.
 */
public static class PlaceholderFragment extends Fragment {
    /**
     * The fragment argument representing the section number for this
     * fragment.
     */
```
private static final String ARG_SECTION_NUMBER = "section_number";

public PlaceholderFragment() {
}

/**
 * Returns a new instance of this fragment for the given section
 * number.
 *
 * @param sectionNumber
 * @return
 */
public static PlaceholderFragment newInstance(int sectionNumber) {
    PlaceholderFragment fragment = new PlaceholderFragment();
    Bundle args = new Bundle();
    args.putInt(ARG_SECTION_NUMBER, sectionNumber);
    fragment.setArguments(args);
    return fragment;
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.fragment_main, container, false);
    TextView textView = (TextView) rootView.findViewById(R.id.section_label);
    textView.setText(String.format("Hello World from section: %d", getArguments().getInt(ARG_SECTION_NUMBER)));
    return rootView;
}

/**
 * A FragmentPagerAdapter that returns a fragment corresponding to
 * one of the sections/tabs/pages.
 */
public class SectionsPagerAdapter extends FragmentPagerAdapter {
    public SectionsPagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int position) {
        // getItem is called to instantiate the fragment for the given page.
        // Return a PlaceholderFragment (defined as a static inner class below).
        switch (position) {
        case 0:
            return new TwitterLoginFragment();
    }
```java
case 2:
    return new FusionOptFragment();
case 3:
    return new FLoginFragment();
case 4:
    return new PostFragment();
default:
    return PlaceholderFragment.newInstance(position + 1);
}

@Override
public int getCount() {
    // Show 3 total pages.
    return 4;
}

@Override
public CharSequence getPageTitle(int position) {
    switch (position) {
        case 0:
            return "Twitter";
        case 1:
            return "Facebook";
        case 2:
            return "Fusion";
        case 3:
            return "Compose";
    }
    return null;
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    Fragment page = getSupportFragmentManager().findFragmentByTag("android:switcher:" + R.id.container + ":" + mViewPager.getCurrentItem());
    if (page != null) {
        page.onActivityResult(requestCode, resultCode, data);
    } else Log.e("MainAct", "Null Fragment");
}
```
Appendix E

MAINACTIVITY LAYOUT

```xml
<!-- Other XML code here -->
```
Appendix F

TWITTER FRAGMENT

```java
package com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.twtr;

import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.Toast;
import com.twitter.sdk.android.Twitter;
import com.twitter.sdk.android.core.Result;
import com.twitter.sdk.android.core.Session;
import com.twitter.sdk.android.core.TwitterException;
import com.twitter.sdk.android.core.identity.TwitterLoginButton;
import com.yogidevelopers.yogiaandroid.socialfusion.R;
import org.json.JSONArray;
import org.json.JSONException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.TimeZone;

/**
 * A simple (Fragment) subclass.
 */
public class TwitterLoginFragment extends Fragment {
    private TwitterLoginButton loginButton;
    private RelativeLayout twitterFeedView;
    private EditText tweetLimitEditText;
    private Button setTweetLimit;
    private ArrayList<TweetObject> tweets = new ArrayList<>();
```
private RecyclerView recyclerView;
private RecyclerViewAdapterTW recyclerViewAdapterTW;
private int counter = 0;

public TwitterLoginFragment() {
    // Required empty public constructor
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View v = inflater.inflate(R.layout.fragment_twlogin, container, false);
    loginButton = (TwitterLoginButton) v.findViewById(R.id.twitter_login_button);
tweeterFeedView = (RelativeLayout) v.findViewById(R.id.tweetIn);
tweetLimitEditText = (EditText) v.findViewById(R.id.limitText);
setTweetLimit = (Button) v.findViewById(R.id.limitSetButton);
recyclerView = (RecyclerView) v.findViewById(R.id.twRecyclerView);
recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
recyclerViewAdapterTW = new RecyclerViewAdapterTW(new ArrayList<TweetObject>(), getActivity());
recyclerView.setAdapter(recyclerViewAdapterTW);
    // Tweets = getArguments().getParcelableArrayList("tWFeed");
    recyclerViewAdapterTW.loadNewTweets(tweets);
    Session activeSession = Twitter.getSessionManager().getActiveSession();
    if (activeSession != null) {
        getTweets();
    } else {
        twitterLogin();
    }

    return v;
}

private void twitterLogin() {
private void twitterLogin() {
    loginButton.setCallback(new Callback<TwitterSession>() {
        @Override
        public void success(Result<TwitterSession> result) {
            TwitterSession session = result.data;
            // Twitter.getInstance().core.getSessionManager().getActiveSession()
            Toast.makeText(getActivity(), msg, Toast.LENGTH_LONG).show();
        }
        @Override
        public void failure(TwitterException exception) {
            String msg = exception.toString();
            Toast.makeText(getActivity(), msg, Toast.LENGTH_LONG).show();
        }
    });
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == 7) {
        setTweetLimit.setVisibility(View.VISIBLE);
        if (counter < 7) {
            setTweetLimit.setOnClickListener(new View.OnClickListener() {
                if (counter < 7) {
                    counter += 1;
                    if (tweetLimitEText.getText().length() == 0) {
if(tweets.size() != 0) {
    recyclerAdapterTN.loadNewTweets(tweets);
} else {
    tweets.clear();
    tweetsLimitText.setText(tweetsLimitText.getText().toString());
}

recyclerView.getAdapter().notifyDataSetChanged();
}
else {
    tweetsLimit.setVisibility(View.GONE);
    new android.os.Handler().postDelayed(
        new Runnable() {
            public void run() {
                Toast.makeText(getBaseContext(), "Requests to server available", Toast.LENGTH_LONG).show();
                counter = 0;
                setTweetLimit.setVisibility(View.VISIBLE);
            }
        }, 900000);
    recyclerAdapterTN.loadNewTweets(tweets);
    Toast.makeText(getBaseContext(), "Please limit no of requests to server!", Toast.LENGTH_LONG).show();
}

private void getNextData(String s) {
    TwitterSession session = Twitter.getSessionManager().getActiveSession();
    new MyTwitterApi(Client(session).getHomeTimeline()).enqueue(new Callback<List>() {
        @Override
        public void success(ResultList result) {
            List array = new ArrayList(result.data);
            parseJSON(array);
        }
        @Override
        public void failure(TwitterException exception) {
            }
    });
}
private void parseJson(List array) {
    if (!array.isEmpty()) {
        String id, timeString, post, user_name = "", user_image = "", user_id = "";
        Date time;
        try {
            JSONArray tweets = new JSONArray(array);
            for (int i = 0; i < tweets.length(); i++) {
                JSONObject curTweet = tweets.getJSONObject(i);
                String media_url = "";
                if (curTweet.has("created_at")) {
                    timeString = curTweet.getString("created_at");
                    time = getDateTime(timeString);
                } else
                    time = null;
                if (curTweet.has("id_str"))
                    id = curTweet.getString("id_str");
                else
                    id = "";
                if (curTweet.has("text"))
                    post = curTweet.getString("text");
                else
                    post = "";
                if (curTweet.has("entities")) {
                    JSONObject entities = curTweet.getJSONObject("entities");
                    if (entities.has("media")) {
                        JSONArray media = entities.getJSONArray("media");
                        JSONObject mediaObj = media.getJSONObject(0);
                        if (mediaObj.has("media_url"))
                            media_url = mediaObj.getString("media_url");
                    }
                }
                if (curTweet.has("user")) {
                    JSONObject user = curTweet.getJSONObject("user");
                    if (user.has("id_str"))
                }
            }
        }
    }
}
private Date getDate(String timeString) {
    SimpleDateFormat twitter_date = new SimpleDateFormat("EEE MMM dd HH:mm:ss ZZZZ yyyy");
    long when = 0;
    try {
        when = twitter_date.parse(timeString).getTime();
    } catch (ParseException e) {
        e.printStackTrace();
    }
    Date localDate = new Date(when + TimeZone.getDefault().getOffset() + (TimeZone.getDefault().inDaylightTime(new Date()) ? TimeZone.getDefault().getDSTSavings() : 0));
    return localDate;
}
Appendix G

TWITTER LAYOUT

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.yogidevelopers.yogandroid.socialfusion.ver_10_22.twtr.TwitterLoginFragment">
    <com.twitter.sdk.android.core.identity.TwitterLoginButton
        android:id="@+id/twitter_login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true" />

    <RelativeLayout
        android:id="@+id/tweetIn"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="gone">
        <EditText
            android:focusableInTouchMode="true"
            android:id="@+id/tweetTextIn"
            android:layout_width="wrap_content"
            android:layout_height="@android:dimen/actionBarSize"
            android:layout_alignParentStart="true"
            android:layout_alignParentStart="true"
            android:hint="Enter number of posts to get"
            android:inputType="number"
            android:maxLines="1" />
        <Button
            android:id="@+id/limitSetButton"
            android:layout_width="wrap_content"
            android:layout_height="actionBarSize"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:text="Fetch" />
    </RelativeLayout>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/TwRecyclerViewIn"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/limitTextIn"
        android:paddingTop="5dp"
        android:scrollbars="vertical" />
</RelativeLayout>
```
Appendix H

TWITTER NEWS-FEED OBJECT
```java
public String getPost() {
    return post;
}

public String getMedia_url() {
    return media_url;
}

public String getUser_name() {
    return user_name;
}

public String getUser_id() {
    return user_id;
}

public String getUser_img() {
    return user_img;
}

@Override
public String toString() {
    return "TweetObject" +
            "\"id\"" + id + \"\" +
            "\"time\"" + time + \"\" +
            "\"post\"" + post + \"\" +
            "\"media_url\"" + media_url + \"\" +
            "\"user_name\"" + user_name + \"\" +
            "\"user_id\"" + user_id + \"\" +
            "\"user_img\"" + user_img + \"\" +
    \};
}

@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeString(id);
    dest.writeLong(time.getTime());
    dest.writeString(post);
    dest.writeString(media_url);
    dest.writeString(user_name);
    dest.writeString(user_id);
    dest.writeString(user_img);
}

public static final Parcelable.Creator<TweetObject> CREATOR = new Parcelable.Creator<TweetObject>() {
    @Override
    public TweetObject createFromParcel(Parcel in) {
        return new TweetObject(in);
    }

    @Override
    public TweetObject[] newArray(int size) {
        return new TweetObject[size];
    }
};
```
Appendix I

TWITTER RECYCLER VIEW ADAPTER

```java
package com.yogidevelopers.yogiandroid.socialfusion.ver_10_22.twtr.recyclerView;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.squareup.picasso.Picasso;
import com.yogidevelopers.yogiandroid.socialfusion.R;
import java.util.List;

import static com.yogidevelopers.yogiandroid.socialfusion.R.mipmap.ic_launcher;

/**
 * Created by yogi on 10/9/16.
 */
public class RecyclerAdapterTW extends RecyclerView.Adapter<MyViewHolder> {
    private List<TweetObject> tweets;
    private Context context;
    private LayoutInflater layoutInflater;

    public RecyclerAdapterTW(List<TweetObject> tweets, Context context) {
        this.tweets = tweets;
        this.context = context;
        this.layoutInflater = LayoutInflater.from(context);
    }

    // Create new views (invoked by the layout manager)
    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = layoutInflater.inflate(R.layout.tweet_single_tweet, parent, false);
        return new MyViewHolder(itemView);
    }

    // Replace the contents of the view with that element
    @Override
    public void onBindViewHolder(MyViewHolder holder, int position) {
        TweetObject curTweet = tweets.get(position);
        holder.twCreator.setText(curTweet.getUser_name() + " on " + curTweet.getTime());
        holder.twText.setText(curTweet.getPost() + "\n" + curTweet.getMedia_url());
        if (curTweet.getMedia_url() != null && curTweet.getMedia_url().isEmpty()) {
            holder.twImage.setVisibility(View.VISIBLE);
            Picasso.with(context).load(curTweet.getMedia_url());
        }
    }

    // Use this method to get the number of movies
    @Override
    public int getItemCount() {
        return tweets.size();
    }
}
```
```java
package com.yogidevelopers.yogandroid.socialfusion.ver_10_22.twtr.recyclerview;

import android.support.v7.widget.RecyclerView;
import android.widget.TextView;
import android.widget.ImageView;
import android.widget.LoadingLayout;

import com.yogidevelopers.yogandroid.socialfusion.R;

/**
 * Created by yogi on 10/9/16.
 */

public class MyViewHolder extends RecyclerView.ViewHolder {
    public TextView tvCreator;
    public TextView tweet;
    public ImageView tweetImage;

    public MyViewHolder(View itemView) {
        super(itemView);
        this.tvCreator = (TextView) itemView.findViewById(R.id.tvCreator);
        this.tweet = (TextView) itemView.findViewById(R.id.tweet);
        this.tweetImage = (ImageView) itemView.findViewById(R.id.tweetImage);
    }
}
```
Appendix J

FACEBOOK FRAGMENT

```java
package com.yogidevelopers.yogandroid.socialfusion.ver_10_22.fcba;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.Toast;
import com.facebook.AccessToken;
import com.facebook.CallbackManager;
import com.facebook.FacebookCallback;
import com.facebook.FacebookException;
import com.facebook.GraphRequest;
import com.facebook.GraphResponse;
import com.facebook.HttpMethod;
import com.facebook.Profile;
import com.facebook.ProfileTracker;
import com.facebook.login.LoginResult;
import com.yogidevelopers.yogandroid.socialfusion.ver_10_22.fcba.recyclerview.FBObject;
import com.yogidevelopers.yogandroid.socialfusion.ver_10_22.fcba.recyclerview.RecyclerAdapterFB;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.TimeZone;

//@
// A simple (Blank) Fragment subclass.
//
public class FBLoginFragment extends Fragment {
    //initialize this single-tone object ...factory pattern... inside onCreateView()
    private CallbackManager callbackManager;
    private ProfileTracker profileTracker;
    private LoginButton loginButton;
    private RelativeLayout feedView;
    private Button getFeedButton;
```
private EditText getEmptyText;

private RecyclerView recyclerView;
private RecyclerView.Adapter FB_recyclerAdapterFB;
private ArrayList<FBOBJECT> feed = new ArrayList<>();

public FBLoginFragment() {
    // Required empty public constructor
}

@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putParcelableArrayList("FbFeedIn", feed);
}

@Override
public void onResume() {
    super.onResume();
    recyclerViewAdapterFB.loadNewFBPosts(feed);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View v = inflater.inflate(R.layout.fragment_fblogin, container, false);
    loginButton = (LoginButton) v.findViewById(R.id.login_button);
    feedView = (RelativeLayout) v.findViewById(R.id.feedInLogIn);
    callbackManager = CallbackManager.Factory.create();
    getFeedButton = (Button) v.findViewById(R.id.inputlimitTIN);
    getLimitText = (EditText) v.findViewById(R.id.InputText);
    recyclerView = (RecyclerView) v.findViewById(R.id.fbRecyclerFeedTIN);
    recyclerView.setAdapter(fbRecyclerAdapterFB);
    recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
    recyclerViewAdapterFB = new recyclerViewAdapterFB(new ArrayList<>(), getActivity());
    recyclerView.setAdapter(recyclerViewAdapterFB);

    if (savedInstanceState == null || !savedInstanceState.containsKey("FbFeedIn")) {
        Log.e("SavedInstanceState", " Is Empty!");
    } else {
        Log.e("SavedInstanceState", " Success!");
        feed = savedInstanceState.getParcelableArrayList("FbFeedIn");
        recyclerViewAdapterFB.loadNewFBPosts(feed);
    }
}
recyclerViewFB.loadNewFBPosts(feed);
Access_token loggedIn = AccessToken.getCurrentAccessToken();
if (loggedIn == null) {
    loadFeedView();
} else {
    login();
}
return v;

private void login() {
    loginButton.setReadPermissions("public_profile", "user_posts");
    loginButton.setFragment(this);
    loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
        @Override
        public void onSuccess(LoginResult loginResult) {
            getProfile();
            // downloadFeed.feedToMainFeedData("From Login Freg to Main!");
            loadFeedView();
        }

        @Override
        public void onCancel() {
            Toast.makeText(getActivity(), "Login Cancelled!", Toast.LENGTH_LONG).show();
        }

        @Override
        public void onError(FacebookException error) {
            Toast.makeText(getActivity(), "Login Error!", Toast.LENGTH_LONG).show();
        }
    });
    loginButton.clearPermissions();
    loginButton.setPublishPermissions("publish_actions");
}

private void loadFeedView() {
    loginButton.setVisibility(View.GONE);
    feedView.setVisibility(View.VISIBLE);
final Bundle params = new Bundle();
params.putString("fields", "id,from.created_time,message,story,caption,link,name,picture,permalink_url,message_tags");

@Override
public void onClick(View v) {
    if (getLimitText.getText().length() == 0) {
        if (feed.size() == 0)
            recyclerViewFB.loadNewFBPosts(feed);
        else
            getFBData(params);
    } else {
        feed.clear();
        String limit = getLimitText.getText().toString();
        params.putString("limit", limit);
        getFBData(params);
    }
    recyclerViewFB.getAdapter().notifyDataSetChanged();
}

private void getFBData(Bundle params) {
    new GraphRequest(
        AccessToken.getCurrentAccessToken(),
        "/new/feed",
        params, HttpMethod.GET,
        new GraphRequest.Callback() {
            public void onCancelled(GraphResponse response) {
                // handle error
            }
            public void onCompleted(GraphResponse response) {
                JSONObject feedResponse = response.getJSONObject();
                parseResponse(feedResponse);
            }
        }).executeAsync();
}

private void parseResponse(JSONObject jsonObject) {
    if (jsonObject != null) {
        String id, creator = "", timeString, story, message, picture, url;
        Date time;
Date time;
JSONObject from;

try {
    JSONArray data = jsonObject.getJSONArray("data");
    for (int i = 0; i < data.length(); i++) {
        JSONObject current = data.getJSONObject(i);
        if (current.has("id"))
            id = current.getString("id");
        else
            id = "";
        if (current.has("from")) {
            from = current.getJSONObject("from");
            creator = from.getString("name");
        } else
            creator = null;
        if (current.has("created_time")) {
            timeString = current.getString("created_time");
            time = getDateTime(timeString);
        } else {
            time = null;
        }
        if (current.has("story"))
            story = current.getString("story");
        else
            story = "";
        if (current.has("message"))
            message = current.getString("message");
        else
            message = "";
        if (current.has("picture"))
            picture = current.getString("picture");
        else
            picture = "";
        if (current.has("permalink_url"))
            url = current.getString("permalink_url");
        else
            url = "";
        FBOBJECT post = new FBOBJECT(id, creator, time, story, message, picture, url);
        this.feed.add(post);
    }
} catch (JSONException e) {
private Date getDate(String timeString) {
    SimpleDateFormat fb_dateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssZ");
    long when = 0;
    try {
        when = fb_dateFormat.parse(timeString).getTime();
    } catch (ParseException e) {
        e.printStackTrace();
    }
    Date localDate = new Date(when + TimeZone.getDefault().getRawOffset() +
        (TimeZone.getDefault().inDaylightTime(new Date()) ? TimeZone.getDefault().getDSTSavings() : 0));
    return localDate;
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    callbackManager.onActivityResult(requestCode, resultCode, data);
}

private void getProfile() {
    if (profile.getCurrentProfile() == null) {
        profileTracker = new ProfileTracker(oldProfile, currentProfile) {
            Toast.makeText(getContext(), "Welcome, " + currentProfile.getName(), Toast.LENGTH_LONG).show();
            profileTracker.stopTracking();
        };
    } else {
        Profile profile = profile.getCurrentProfile();
        Toast.makeText(getContext(), "Welcome, " + profile.getName(), Toast.LENGTH_LONG).show();
    }
}
Appendix K

FACEBOOK LAYOUT

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.yogidevelopers.yogifindroid.socialfusion.ver_18_22.fbck.FBLoginFragment">

    <!--Login Button
    android:id="@+id/Login_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:layout_marginBottom="30dp"
    android:layout_marginTop="30dp" />

    <RelativeLayout
        android:id="@+id/feedinLogin"
        android:visibility="gone"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <EditText
            android:id="@+id/inputTextIN"
            android:layout_width="wrap_content"
            android:layout_height="@actionBarSize"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_topleftOf="@+id/inputLimitIN"
            android:hint="Enter number of posts to get"
            android:maxLength="1" />
        <Button
            android:id="@+id/inputLimitIN"
            android:layout_width="wrap_content"
            android:layout_height="@actionBarSize"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:text="Fetch" />
        <android.widget.recyclerview.RecyclerView
            android:id="@+id/FBRecycleFeedIN"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_below="@+id/inputTextIN"
            android:paddingTop="5dp"
            android:background="@color/cos_facebook_blue"
            android:scrollbars="vertical" />
    </RelativeLayout>
</RelativeLayout>
```
Appendix L

FACEBOOK NEWS-FEED OBJECT

```java
package com.yogidevelopers.yogandroid.socialFusion.ver_18_22.fbK.recycleView;
import android.os.Parcel;
import android.os.Parcelable;
import java.util.Date;

/**
 * Created by yogi on 9/12/16.
 */
public class FObject implements Parcelable {

    private String id;
    private String creator;
    private Date time;
    private String story;
    private String message;
    private String picture;
    private String url;

    public FObject(String id, String creator, Date time,
            String story, String message,
            String picture, String url) {
        this.id = id;
        this.creator = creator;
        this.time = time;
        this.story = story;
        this.message = message;
        this.picture = picture;
        this.url = url;
    }

    private FObject(Parcel in) {
        id = in.readString();
        creator = in.readString();
        time = new Date(in.readLong());
        story = in.readString();
        message = in.readString();
        picture = in.readString();
        url = in.readString();
    }

    public String getId() {
        return id;
    }

    public String getCreator() {
        return creator;
    }

    public Date getTime() {
        return time;
    }
```
public String getStory() {
    return story;
}

public String getMessage() {
    return message;
}

public String getPicture() {
    return picture;
}

public String getURL() {
    return url;
}

@Override
public String toString() {
    return "FObject" + 
           "id: " + id + "\n" + 
           "creator: " + creator + "\n" + 
           "time: " + time + "\n" + 
           "story: " + story + "\n" + 
           "message: " + message + "\n" + 
           "picture: " + picture + "\n" + 
           "url: " + url + "\n" + 
    ';
}

@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeString(id);
    dest.writeString(creator);
    dest.writeLong(time.getTime());
    dest.writeString(story);
    dest.writeString(message);
    dest.writeString(picture);
    dest.writeString(url);
}

public static final Parcelable.Creator<FObject> CREATOR = new Parcelable.Creator<FObject>() {
    public FObject createFromParcel(Parcel in) {
        return new FObject(in);
    }
};

public FObject[] newArray(int size) {
    return new FObject[size];
}
Appendix M

FACEBOOK RECYCLERVIEW ADAPTER

```java
package com.yogidevelopers.yogiaodroid.socialfusion.ver_10_22.fcbk.recyclerView;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.squareup.picasso.Picasso;
import com.yogidevelopers.yogiaodroid.socialfusion.R;
import java.util.List;
import static com.yogidevelopers.yogiaodroid.socialfusion.R.mipmap.ic_launcher;

/**
 * Created by yogi on 9/12/16.
 */
public class RecyclerViewAdapterFB extends RecyclerView.Adapter<MyViewHolder> {

    private List<FBOBJECT> dataset;
    private Context context;
    private LayoutInflater layoutInflater;

    public RecyclerViewAdapterFB(List<FBOBJECT> dataset, Context context) {
        this.dataset = dataset;
        this.context = context;
        this.layoutInflater = LayoutInflater.from(context);
    }

    // Create new views (invoked by the layout manager)
    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = layoutInflater.inflate(R.layout.fb_single_post, parent, false);
        return new MyViewHolder(v);
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(MyViewHolder holder, int position) {
        // - get element from your dataset at this position
        // - replace the contents of the view with that element
        FBOBJECT currPost = dataset.get(position);

        holder.postText.setText(currPost.getText());
        holder.postTime.setText(currPost.getTime());

        Picasso.with(context).load(currPost.getPicture()).is_empty()
            .error(R.mipmap.ic_launcher);
    }
}
```
```java
package com.yogidevelopers.yogia.android.socialfusion.ver_10_22.fcbk.recyclerView;

import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.yogidevelopers.yogiaandroid.socialfusion.R;

/**
 * Created by yogi on 9/12/16.
 */
public class MyViewHolder extends RecyclerView.ViewHolder {

    // Provide a reference to the views for each data item
    // Complex data items may need more than one view per item, and
    // you provide access to all the views for a data item in a view holder

    public TextView poster;
    public TextView post;
    public ImageView image;

    public MyViewHolder(View itemView) {
        super(itemView);

        this.post = (TextView) itemView.findViewById(R.id.fbpost);
        this.poster = (TextView) itemView.findViewById(R.id.fbPoster);
        this.image = (ImageView) itemView.findViewById(R.id.postImage);
    }
}

public int getItemCount() {
    return dataset != null ? dataset.size() : 0;
}

public void loadNewFbPosts(List<FbObject> feed) {
    dataset = feed;
    notifyDataSetChanged();
}
```
<xml version="1.0" encoding="utf-8">
  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="5dp"
      android:background="@color/cardview_light_background">
    <TextView
      android:id="@id/fbposter"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_toLeftOf="@id/postImage"
      android:textSize="18sp" />
    <TextView
      android:id="@id/fbpost"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_below="@id/fbposter"
      android:layout_marginTop="2dp"
      android:layout_toLeftOf="@id/postImage" />
    <ImageView
      android:id="@id/postImage"
      android:layout_width="100dp"
      android:layout_height="100dp"
      android:layout_alignParentEnd="true"
      android:layout_alignParentRight="true"
      android:scaleType="fitCenter" />
  </RelativeLayout>
Appendix N

FUSION TIMELINE

package com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.fusn;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.facebook.AccessToken;
import com.facebook.GraphRequest;
import com.facebook.GraphResponse;
import com.facebook.HttpMethod;
import com.twitter.sdk.android.Twitter;
import com.twitter.sdk.android.core.Callback;
import com.twitter.sdk.android.core.Result;
import com.twitter.sdk.android.core.Session;
import com.twitter.sdk.android.core.TwitterException;
import com.twitter.sdk.android.core.TwitterSession;
import com.yogidevelopers.yogiaandroid.socialfusion.R;
import com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.twtr.MyTwitterApiClient;
import com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.fusn.recyclerview.FusnListAdapterFusn;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.List;
import java.util.TimeZone;

/**
 * A simple (Fragment) subclass.
 */
public class FusionOptFrag extends Fragment {
    private TextView emptyView;
    private RelativeLayout feedView;
    private Button getFeed;
    private EditText setLimitText;
private int counter = 0;

private RecyclerView recyclerView;
private RecyclerViewAdapter<FS fusionAdapter;
private ArrayList<FragmentObject> fusionFeed = new ArrayList<>();

public FusionOptFrag() {
  // Required empty public constructor
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View v = inflater.inflate(R.layout.fragment_optfusion, container, false);
    emptyView = (TextView) v.findViewById(R.id.emptyFusion);
    feedView = (RelativeLayout) v.findViewById(R.id.fusionRelativeLayout);
    setTitleText = (EditText) v.findViewById(R.id.titleLimitFusion);
    getFeed = (Button) v.findViewById(R.id.limitFSetButton);
    recyclerView = (RecyclerView) v.findViewById(R.id.recyclerView);
    recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
    fusionAdapter = new RecyclerViewAdapter<FS(getActivity()), new ArrayList<FragmentObject>();
    recyclerView.setAdapter(fusionAdapter);

    fusionAdapter.loadNews(fusionFeed);
    AccessToken fbLoggedIn = AccessToken.getCurrentAccessToken(); // Facebook
    Session twittergetSession = twitter.getSessionManager().getSession(); // Twitter
    if (fbLoggedIn != null && twittergetSession != null) {
        emptyView.setVisibility(View.GONE);
        feedView.setVisibility(View.VISIBLE);
    } else {
        feedView.setVisibility(View.GONE);
        emptyView.setVisibility(View.VISIBLE);
    }

    return v;
}

private void loadData() {

```java
if (counter < 7) {
    getFeed.setVisibility(View.VISIBLE);
}

@Override
public void onClick(View v) {
    if (counter < 7) {
        counter += 1;
        if (setLimitText.getText().length() == 0) {
            if (fusionFeed.size() <= 0)
                fusionAdapter.loadNew([fusionFeed]);
            else
                loadDefaultData();
        } else {
            fusionFeed.clear();
            loadSpecData(setLimitText.getText().toString());
        }
        recyclerView.getAdapter().notifyDataSetChanged();
    } else {
        getFeed.setVisibility(View.GONE);
        new android.os.Handler().postDelayed() {
            Toast.makeText(getContext(), "Requests to server available", Toast.LENGTH_LONG).show();
            counter = 0;
            getFeed.setVisibility(View.VISIBLE);
            fusionAdapter.loadNew([fusionFeed]);
            Toast.makeText(getContext(), "Please limit no of requests to server!", Toast.LENGTH_LONG).show();
        }, 9000000;
    }
}

private void loadDefaultData() {
    getFbData("20");
    getTdata("20");
    processFeed();
}
```
```java
private void loadSpecData(String s) {
    getData(s);
    getTwxData(s);
    processFeed();
}

private void getTwxData(String s) {

    TwitterSession session = Twitter.getSessionManager().getActiveSession();
    new MyTwitterApiClient(session).getStatus().enqueue(new Callback<List>() {
        @Override
        public void success(ResultList result) {
            Log.e("Twitter Success!", "My RESPONSE!");
            List array = new ArrayList(result.data);
            parseTwxJson(array);
        }

        @Override
        public void failure(TwitterException exception) {
        }
    });

    private void parseTwxJson(List array) {
        if (!array.isEmpty()) {
            String id, timeString, nTags = "", post, user_name = "", user_image = "", user_id = "";
            long time;
            try {
                JSONArray tweets = new JSONArray(array);
                for (int i = 0; i < tweets.length(); i++) {
                    JSONObject curTweet = tweets.getJSONObject(i);
                    StringBuilder tags = new StringBuilder(""");
                    String media_url = "";
                    if (curTweet.has("created_at")) {
                        timeString = curTweet.getString("created_at");
                        time = getTweetsDate(timeString);
                    } else
                        time = 0;
                    if (curTweet.has("id_str"))
                        id = curTweet.getString("id_str");
                    else
                        id = "";
```
if (curTweet.has("text")) {
    post = curTweet.getString("text");
    tags.append(addTags(post));
} else {
    post = "";
}

if (curTweet.has("entities")) {
    JSONObject entities = curTweet.getJSONObject("entities");
    if (entities.has("media")) {
        JSONArray media = entities.getJSONArray("media");
        JSONObject mediaObj = media.getJSONObject(0);
        if (mediaObj.has("media_url"))
            media_url = mediaObj.getString("media_url");
    }
}

if (curTweet.has("user")) {
    JSONObject user = curTweet.getJSONObject("user");
    if (user.has("id_str"))
        user_id = user.getString("id_str");
    if (user.has("name"))
        user_name = user.getString("name");
    if (user.has("profile_image_url"))
        user_image = user.getString("profile_image_url");
}

hTags = tags.toString();
FusionObject fusionObj = new FusionObject("Tw", time, hTags, user_id, null, null, null, null, null,
    post, media_url, user_name, user_id, user_image);
    Log.e("Tw fusion time", String.valueOf(new Date(time)));
//
    this.fusionFeed.add(fusionObj);
    Log.d("Tw to ", fusionObj.getHashtags());
} catch (JSONException e) {
    e.printStackTrace();
}
}
private long getUTCTime(String timeString) {
    SimpleDateFormat twitter_date = new SimpleDateFormat("EEE MMM dd HH:mm:ss ZZZZZ yyyy");
    long when = 0;
    try {
        when = twitter_date.parseString().getTime();
    } catch (ParseException e) {
        e.printStackTrace();
    }
    Date localDate = new Date(when + TimeZone.getDefault().getRawOffset() +
        (TimeZone.getDefault().inDaylightTime(now Date()) ? TimeZone.getDefault().getDSTSavings() : 0));
    return localDate.getTime();
}

private void getFBData(String s) {
    Bundle params = new Bundle();
    params.putString("fields", "id,from,created_time,message,caption,link,name,picture,permalink_url,media_tags");
    params.putString("limit", "1");
    new GraphRequest{
        "Access_token", getCurrentAccessToken(),
        "/me/feed",
        parameters, // or null
        HttpMethod.GET,
        (response) -> {
            JSONObject fResponse = response.getJSONObject();
            parseFBResponse(fResponse);
        }
    }.executeAsync();
}

private void parseFBResponse(JSONObject fResponse) {
    if (fResponse == null) {
        String id, creator, timeString, story, message, hasTags, picture, url;
        JSONObject from;
        long time;
    }
    try {
        JSONArray array = fResponse.getJSONArray("data");
        for (int i = 0; i < array.length(); i++) {
            JSONObject current = array.getJSONObject(i);
            if (current.containsKey("id")) {
                id = current.getString("id");
            } else {
                id = ""
            }
        }
    }
if (current.has("from")) {
  from = current.getJSONObject("from");
  creator = from.getString("name");
} else
  creator = null;

if (current.has("created_time")) {
  timeString = current.getString("created_time");
  time = getF0Date(timeString);
} else {
  time = 0;
}

if (current.has("story"))
  story = current.getString("story");
else
  story = "";

if (current.has("message")) {
  message = current.getString("message");
  hashtags = addTags(message);
} else {
  message = "";
  hashtags = "";
}

if (current.has("picture"))
  picture = current.getString("picture");
else
  picture = "";

if (current.has("permalink_url"))
  url = current.getString("permalink_url");
else
  url = "";

FusionObject fFusionObj = new FusionObject("FB", time, hashtags, id, creator, story, message, picture, url,
null, null, null, null, null);
this.fusioFeed.add(fFusionObj);
Log.i("FB tt: ", fFusionObj.getHashTags());
}

private long getF0Date(String timeString) {
```
    SimpleDateFormat fb_dateFormat = new SimpleDateFormat("yyyy-MM-dd'T'hh:mm:ssZ");
    long when = 0;
    try {
        when = fb_dateFormat.parse(timeString).getTime();
    } catch (ParseException e) {
        e.printStackTrace();
    }
    Date localDate = new Date(timeZone.getDefault().getTimeZone().getRawOffset() + (TimeZone.getDefault().getOffset().inDaylightTime(new Date()) ? TimeZone.getDefault().getDSTSavings() : 0));
    return localDate.getTime();
}

private String addTags(String message) {
    StringBuilder tags = new StringBuilder(""");
    if (message.length() > 0) {
        String words[] = message.split(" ");
        for (String w : words) {
            if (w.startsWith("#"))
                tags.append(w);
        }
    }
    return tags.toString();
}

private void processFeed() {
    Collections.sort(fusioFeed, FusionObject.PostDate);
    Collections.reverse(fusioFeed);
    ArrayList<Integer> removeIndexes = new ArrayList<Integer>();
    for (int i = 0; i < fusioFeed.size()-1; i++) {
        FusionObject ithObj = fusioFeed.get(i);
        for (int j = i + 1; j < fusioFeed.size(); j++) {
            FusionObject jthObj = fusioFeed.get(j);
            if (ithObj.getHashTags().compareTo(jthObj.getHashTags()) == 0
                && ithObj.getObjectType().equals(jthObj.getObjectType())
                && ithObj.getHashTags().equals(""")
                && jthObj.getHashTags().equals(""")
                && ithObj.getId().equals(jthObj.getId())) {
                removeIndexes.add(j);
            }
        }
    }
    fusioFeed.remove(removeIndexes);
Date jthDate = new Date(jthObj.getTime());
int r = iDate.compareTo(jthDate);
Log.e("Times", "i th"+iDate+- +j is"+jthDate+" result is: "+r);
if (r < 0) {
    Log.i("Rm Ith", iDate.toString());
    removeIndexes.add(i);
} else {
    Log.i("Rm Jth", jthObj.toString());
    removeIndexes.add(j);
}
}
for (int k = 0; k < removeIndexes.size(); k++) {
    int index = removeIndexes.get(k);
    Log.e("Removing", fusioFeed.get(index).toString());
    fusioFeed.remove(index);
}
fusionAdapter.loadNews(fusioFeed);
Appendix P

FUSION TIMELINE OBJECT

```java
package com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.fusn.recyclerviewFusn;

import android.os.Parcel;
import android.os.Parcelable;
import java.util.Comparator;
import java.util.Date;

/**<*
 * Created by yogi on 10/13/16.
 */

public class FusionObject implements Parcelable {

    private String objType;
    private long conTime;
    private String hashTms;
    private String id;

    //FB Fields
    private String fb_creator;
    private String fb_story;
    private String fb_message;
    private String fb_picture;
    private String fb_url;
    private String tw_post;
    private String tw_media_url;
    private String tw_user_name;
    private String tw_user_id;
    private String tw_user_img;

    public String getObjType() {
        return objType;
    }

    public long getConTime() {
        return conTime;
    }

    public String getId() {
        return id;
    }

    public String getFb_story() {
        return fb_story;
    }

    public String getFb_message() {
        return fb_message;
    }

    public String getFb_picture() {
        return fb_picture;
    }
```
public String getFb_url() {
    return fb_url;
}

public String getTw_url() {
    return tw_url;
}

public String getFb_media_url() {
    return fb_media_url;
}

public String getFb_user_name() {
    return fb_user_name;
}

public String getTw_user_name() {
    return tw_user_name;
}

public String getFb_user_id() {
    return fb_user_id;
}

public String getTw_user_id() {
    return tw_user_id;
}

public String getFb_user_img() {
    return fb_user_img;
}

public String getTw_user_img() {
    return tw_user_img;
}

public String getHashtags() {
    return hashtags;
}

public String getFb_creator() {
    return fb_creator;
}

public FusionObject(String objType, long conTime, String hashtags, String id, String fb_creator,
                     String fb_story, String fb_message, String fb_picture, String fb_url,
                     String tw_post, String tw_media_url, String tw_user_name, String tw_user_id,
                     String tw_user_img) {
    this.objType = objType;
    this.conTime = conTime;
    this.hashtags = hashtags;
    this.id = id;
    this.fb_creator = fb_creator;
    this.fb_story = fb_story;
    this.fb_message = fb_message;
    this.fb_picture = fb_picture;
    this.fb_url = fb_url;
    this.tw_post = tw_post;
    this.tw_media_url = tw_media_url;
    this.tw_user_name = tw_user_name;
    this.tw_user_id = tw_user_id;
    this.tw_user_img = tw_user_img;
}
private FusionObject(Parcel in) {
    id = in.readString();
    comTime = in.readLong();
    hashtags = in.readString();
    fb_creator = in.readString();
    fb_story = in.readString();
    fb_message = in.readString();
    fb_picture = in.readString();
    fb_url = in.readString();
    tw_post = in.readString();
    tw_media_url = in.readString();
    tw_user_name = in.readString();
    tw_user_id = in.readString();
    tw_user_img = in.readString();
}

@Override
public String toString() {
    return "FusionObject" +
           "@ObjectType=" + objType + ","
            + "_fb_time=" + String.valueOf(new Date(comTime)) + ","
            + "_hashtags=" + hashtags + ","
            + "_id=" + id + ","
            + "_fb_creator=" + fb_creator + ","
            + "_fb_story=" + fb_story + ","
            + "_fb_message=" + fb_message + ","
            + "_fb_picture=" + fb_picture + ","
            + "_fb_url=" + fb_url + ","
            + "_tw_post=" + tw_post + ","
            + "_tw_media_url=" + tw_media_url + ","
            + "_tw_user_name=" + tw_user_name + ","
            + "_tw_user_id=" + tw_user_id + ","
            + "_tw_user_img=" + tw_user_img + ";"
    
    @Override
public int describeContents() {
        return 0;
    }

    @Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeString(fb_creator);
    dest.writeLong(comTime);
    dest.writeString(hashtags);
    dest.writeString(fb_story);
}
dest.writeString(fb_message);
dest.writeString(fb_picture);
dest.writeString(fb_url);
dest.writeString(tw_post);
dest.writeString(tw_media_url);
dest.writeString(tw_user_name);
dest.writeString(tw_user_id);
dest.writeString(tw_user_img);
}

public static final Parcelable.Creator<FusionObject> CREATOR = new Parcelable.Creator<FusionObject>() {
    public FusionObject createFromParcel(Parcel in) {
        return new FusionObject(in);
    }

    public FusionObject[] newArray(int size) {
        return new FusionObject[size];
    }
};

public static Comparator<FusionObject> PostDate = new Comparator<FusionObject>() {
    @Override
    public int compare(FusionObject lhs, FusionObject rhs) {
        Date lhsTime = new Date(lhs.getComTime());
        Date rhsTime = new Date(rhs.getComTime());
        return lhsTime.compareTo(rhsTime);
    }
};
Appendix Q

FUSION TIMELINE RECYCLER VIEW ADAPTER

```java
package com.yogadevelopers.yogiandroid.socialfusion.ver_10_22.fusn.recyclerViewFusn;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.squareup.picasso.Picasso;
import com.yogadevelopers.yogiandroid.socialfusion.R;
import java.util.List;

import static com.yogadevelopers.yogiandroid.socialfusion.R.mipmap.ic_launcher;

// Created by yogi on 10/10/16.

public classRecyclerAdapterF5 extends RecyclerView.Adapter<MyFusionViewHolder> {

    private List<FusionObject> news;
    private Context context;
    private LayoutInflater layoutInflater;

    public RecyclerAdapterF5(Context context, List<FusionObject> news) {
        this.context = context;
        this.news = news;
        this.layoutInflater = LayoutInflater.from(context);
    }

    @Override
    public MyFusionViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = layoutInflater.inflate(R.layout.fusion_entry, parent, false);
        return new MyFusionViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(MyFusionViewHolder holder, int position) {
        // Get element from your dataset at this position
        // Replace the contents of the view with that element
        FusionObject newsObj = news.get(position);
        String newsText = new StringBuilder();
        String newsPoster;

        if (newsObj.getObjType().equals("TN")) {
            newsPoster = newsObj.getTw_user_name();
        }

        // Other code...
    }

    public void updateData(List<FusionObject> news) {
        this.news = news;
        notifyDataSetChanged();
    }

    // Other methods...
}
```

```java
83
public void loadNews(List<FusionObject> newsList) {
    news = newsList;
    notifyDataSetChanged();
}
```
package com.yogidevelopers.yogiaandroid.socialfusion.ver_10_22.fusn.recyclerViewFusn;

import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.yogidevelopers.yogiaandroid.socialfusion.R;

/**
 * Created by yogi on 10/10/16.
 */
public class MyFusionViewHolder extends RecyclerView.ViewHolder{
    // Provide a reference to the views for each data item
    // Complex data items may need more than one view per item, and
    // you provide access to all the views for a data item in a view holder

    public TextView newsPoster;
    public TextView newsText;
    public ImageView newsImage;

    public MyFusionViewHolder(View itemView) {
        super(itemView);

        this.newsPoster = (TextView) itemView.findViewById(R.id.newsPoster);
        this.newsText = (TextView) itemView.findViewById(R.id.newsText);
        this.newsImage = (ImageView) itemView.findViewById(R.id.newsImage);
    }
}
<xml version="1.0" encoding="utf-8">
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp">
    
        <TextView
            android:id="@+id/newsPoster"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_toLeftOf="@id/newsImage"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/newsText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_below="@+id/newsPoster"
            android:layout_marginTop="2dp"
            android:layout_toLeftOf="@id/newsImage" />

        <ImageView
            android:id="@+id/newsImage"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:scaleType="fitCenter" />

    </RelativeLayout>
Appendix R

COMPOSE FRAGMENT

```java
package com.yogidevelopers.yogiaandroid.socialfusion.main;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.facebook.AccessToken;
import com.facebook.GraphRequest;
import com.facebook.GraphResponse;
import com.facebook.HttpMethod;
import com.facebook.login.LoginManager;
import com.twitter.sdk.android.Twitter;
import com.twitter.sdk.android.core.Callback;
import com.twitter.sdk.android.core.Result;
import com.twitter.sdk.android.core.Session;
import com.twitter.sdk.android.core.TwitterApiClient;
import com.twitter.sdk.android.core.TwitterCore;
import com.twitter.sdk.android.core.TwitterException;
import com.twitter.sdk.android.core.models.Tweet;
import com.yogidevelopers.yogiaandroid.socialfusion.R;

import retrofit2.Call;

/**
 * A simple @link Fragment subclass.
 * @author		HMR
 */
public class PostFragment extends Fragment {

    private Button postFacebook;
    private Button postTweet;
    private EditText composeText;

    public PostFragment() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View v = inflater.inflate(R.layout.fragment_post, container, false);

        postFacebook = (Button) v.findViewById(R.id.postToFacebook);
    }
```
postTweet = (Button) v.findViewById(R.id.postTweet);
composeText = (EditText) v.findViewById(R.id.composePost);

//Post to Facebook
final AccessToken loggedin = AccessToken.getCurrentAccessToken();
	postFacebook.setOnClickListener(v -> {
		if (composeText.getText().length() == 0) {
			Toast.makeText(getContext(), "Ops! Seems like you forgot to compose...", Toast.LENGTH_LONG).show();
		} else {
			postToFacebook(loggedIn, composeText.getText().toString());
		}
	});

//Post to Twitter
final Session activeSession = Twitter.getSessionManager().getSession();

postTweet.setOnClickListener(v -> {
	if (composeText.getText().length() == 0) {
		Toast.makeText(getContext(), "Ops! Seems like you forgot to compose...", Toast.LENGTH_LONG).show();
	} else {
		postToTwitter(activeSession, composeText.getText().toString());
	}
	});

// @FormUrlEncoded
// @POST("/1.1/statuses/update.json")
// void update(@Field("status") String var1,
// @Field("in_reply_to_status_id") Long var2,
// @Field("possibly_sensitive") Boolean var3,
// @Field("lat") Double var4,
// @Field("long") Double var5,
// @Field("place_id") String var6,
// @Field("display_coordinates") Boolean var7,
// @Field("trim_user") Boolean var8,
// @Field("media_ids") String var9, Callback<Tweet> var10);

return v;

private void postToTwitter(Session activeSession, String s) {
	if (activeSession != null) {
	
TwitterApiClient twitterApiClient = TwitterCore.getInstance().getApiClient();
StatusesService statusesService = twitterApiClient.getStatusesService();

Call<Tweet> call = statusesService.update(new null, null, null, null, null, null, null);
call.enqueue(new Callback<Tweet>() {
    @Override
    public void success(Result<Tweet> result) {
        Toast.makeText(getContext(), "Tweet sent!", Toast.LENGTH_LONG).show();
    }
    
    @Override
    public void failure(TwitterException exception) {
        Toast.makeText(getContext(), "Something went wrong!", Toast.LENGTH_LONG).show();
        Log.e("Post Tweet", exception.toString());
    }

    private void postToFacebook(AccessToken loggedIn, String s) {
        if (loggedIn != null) {
            Bundle params = new Bundle();
            params.putString("message", s);

            new GraphRequest(
                AccessToken.getCurrentAccessToken(),
                "/me/feed",
                params,
                HttpMethod.POST,
                (response) -> {
                    Toast.makeText(getContext(), "Post uploaded to Facebook", Toast.LENGTH_LONG).show();
                    Log.e("Posted Successfully", response.toString());
                    handle the result
                }
            ).executeAsync();
        } else {
            Toast.makeText(getContext(), "Login Required!", Toast.LENGTH_LONG).show();
        }
    }
});
References


