

## Criterion E: Product development

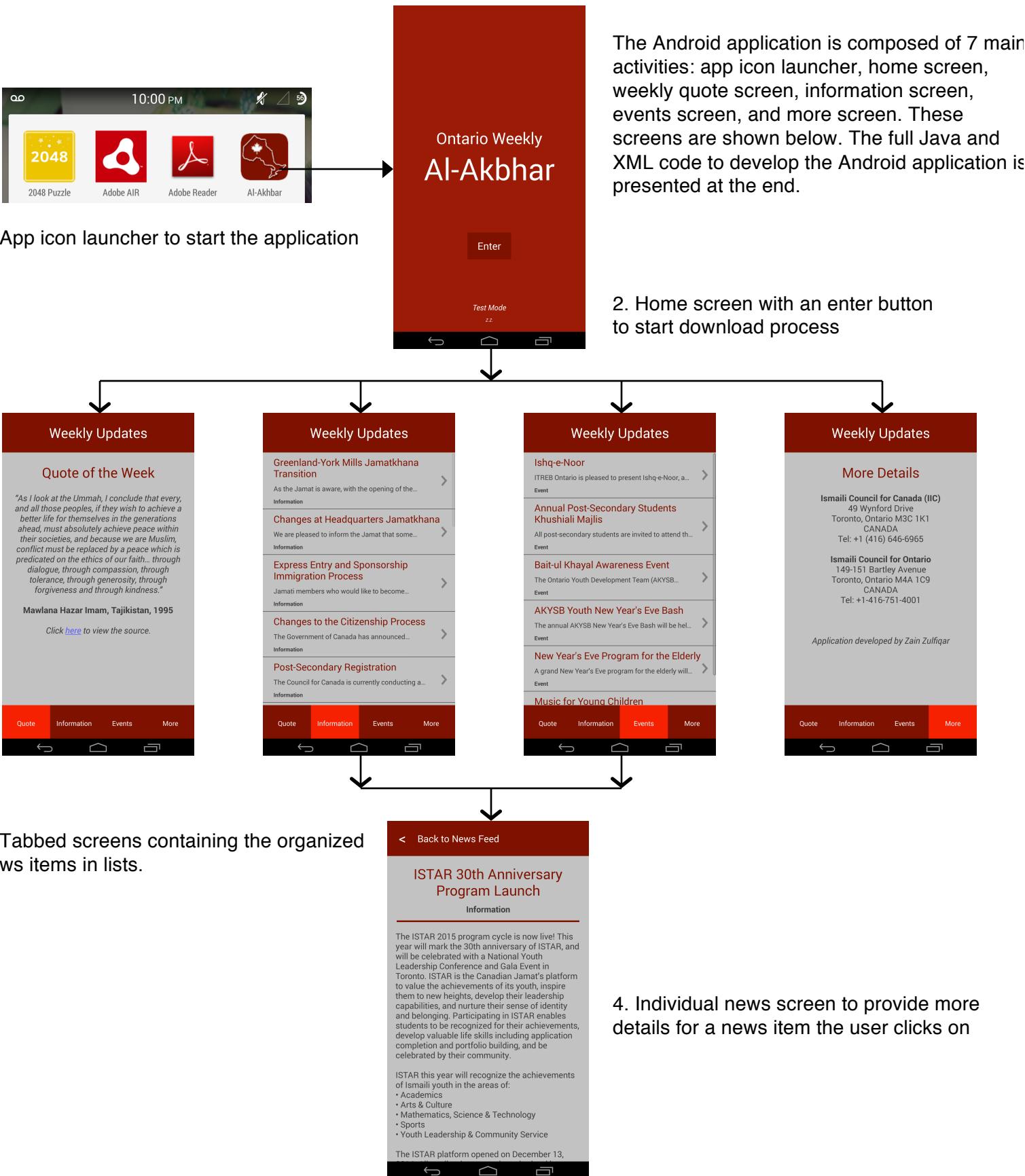
### **Complex techniques used to create the final product**

The complex Java and XML features used to build the structure of the Al-Akhbar Android application are listed below:

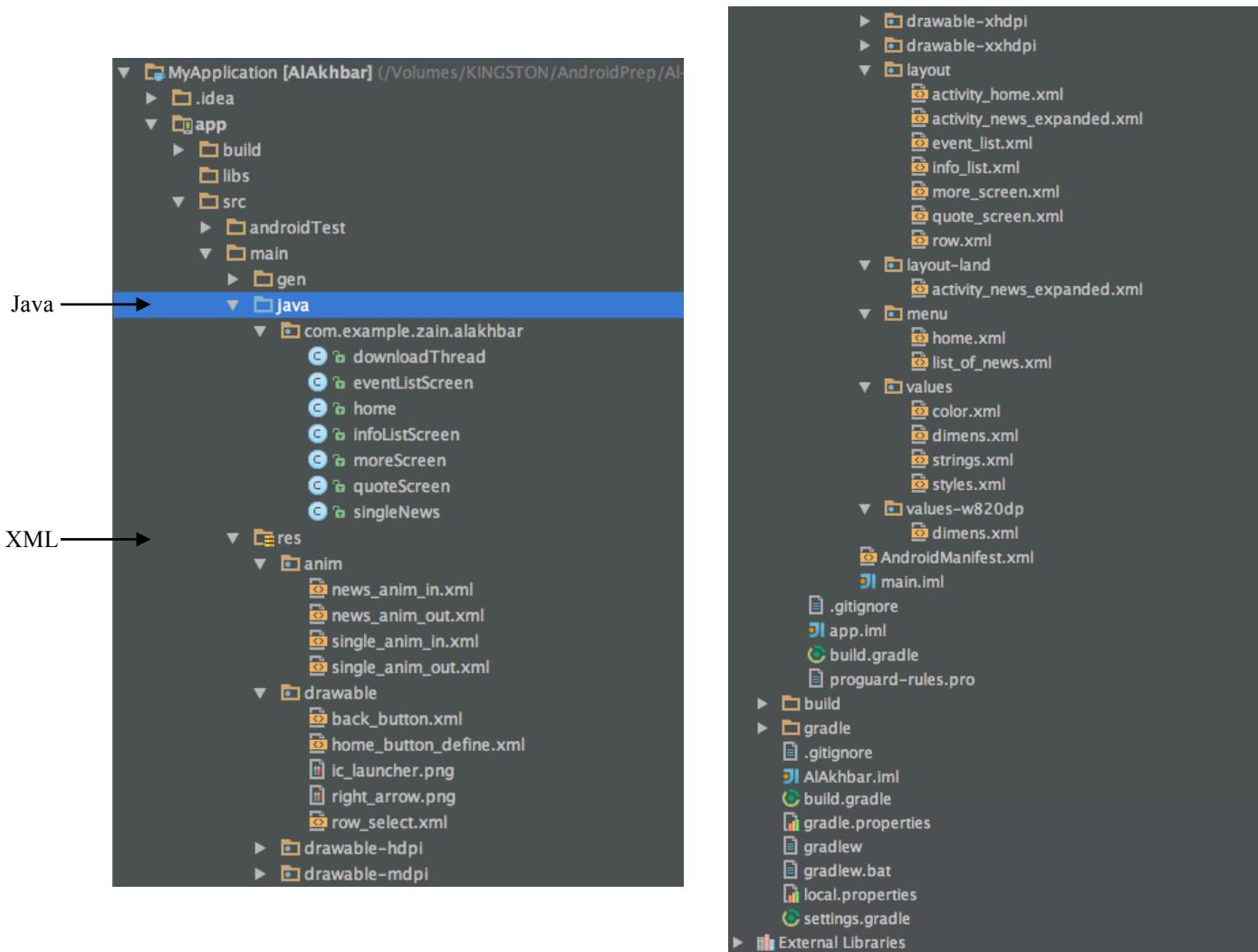
- Client-server based application through HTTP connection.....(4 – 7)
  - Java activities and objects
  - AsyncTask download thread
  - Instantaneous news updates
- Offline application service by reading/writing files in internal storage .....(8 – 12)
  - If statement decision making: Internet connection management: 3G/WiFi
  - Reading/writing device storage permission
  - Buffer streams to write bytes of data
  - Updating app content with JSON parsing - advanced arrays and lists
- Online database to manage news items .....(13 – 15)
  - SQL queries/user interface to update news records
  - Godaddy hosted through personal domain
  - JSON database creation
  - Hashmap key-value pairings to categorize news items
- Use of XML and Java design to create a user-friendly layout.....(16 – 23)
  - Special color scheme and app logo design
  - HTML tags parsing
  - Tabbed activities
  - Custom ListView adapter to customize news items
  - Graphics manipulation, Photoshop
  - Dialog boxes
  - XML based graphical user interface (GUI)

All of the code relating to the extra features of the application is snapshotted at the end of Criterion E. NOTE: The code to develop such extra features was taken from online sources (mentioned in Criterion D), and then manipulated to fit the specifications of my application. Only the structure and concept of the code was copied, the code was completely rewritten by me.

## Overall structure of the Al-Akhbar Android application



Android Studio IDE was used to develop the application. It created its own grade-based compiling files for the Android application. The expanded folders contain all sections where my Java or XML code exists.

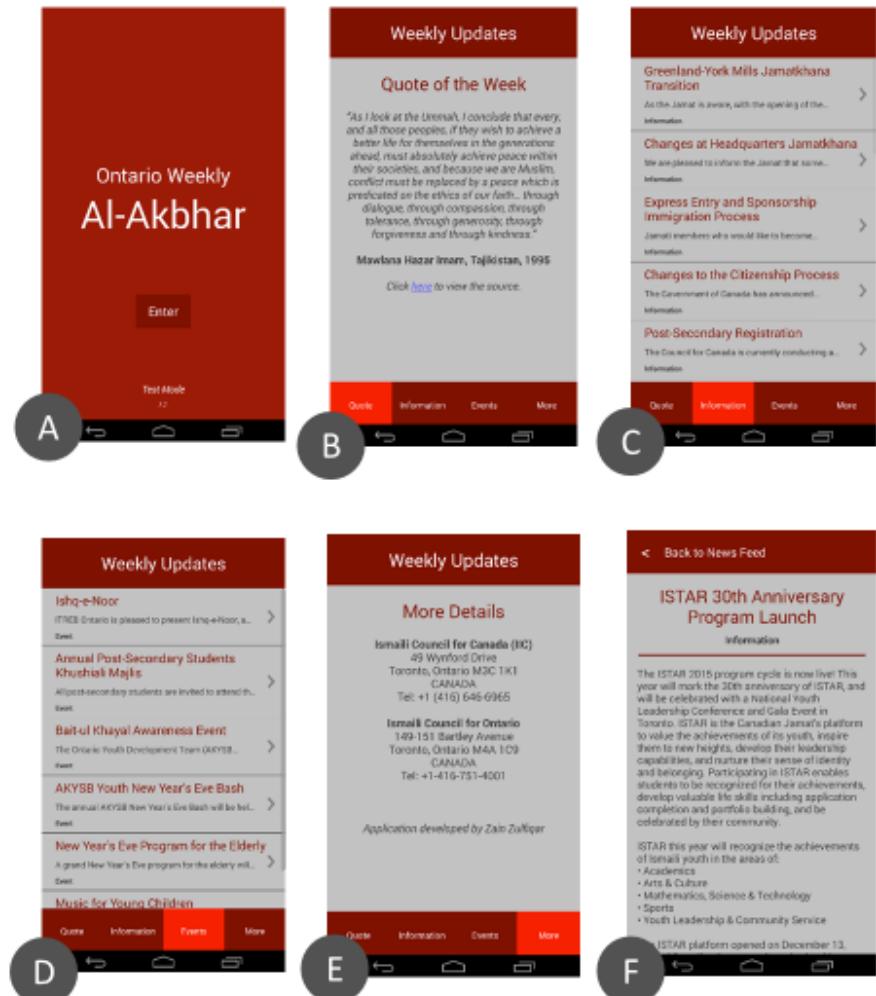


## Client-server based application through HTTP connection

All screens have their own Android OS activity and processes to allow independent memory allocation. This means that if one part of the application fails due to low memory, the background screens that are not being viewed by the user can be closed. This ensures the application has a smooth performance, and does not clog up memory in the user's device<sup>1</sup>.

Every single screen that the user can see has its own activity initialized in the `AndroidManifest.XML` file.

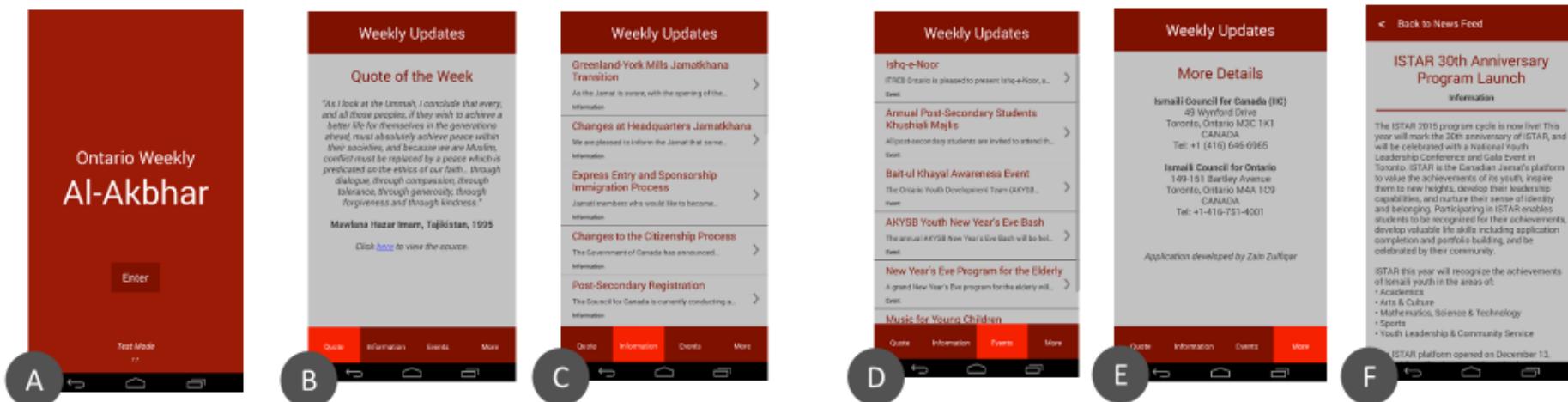
```
<activity
    android:name=".home"
    android:label="@string/app_name"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".quoteScreen"
    android:label="@string/title_activity_list_of_news"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".infoListScreen"
    android:label="@string/title_activity_list_of_news"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".eventListScreen"
    android:label="@string/title_activity_list_of_news"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".moreScreen"
    android:label="@string/title_activity_list_of_news"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".singleNews"
    android:label="@string/title_activity_list_of_news"
    android:screenOrientation="portrait">
</activity>
```



<sup>1</sup> Criterion B, Specification 2: Application must be user friendly and work smoothly with the user.

All code files are organized in their appropriate folders. This allows anyone from the IIC to customize the application without any confusion regarding the location of the source code files. The files are appropriately named by their function so the IIC understands the application structure. Classes help with data abstraction. The IIC does not need to understand the inner-details of the Android activity and can have an abstract understanding of the class functions. Additionally, modularity of classes allows for the subdividing of code between different committees in the IIC to independently work on different sections of the application<sup>2</sup>.

Relevant Java classes work together to structure the whole application.



<sup>2</sup> Criterion B, Specification 1: Create an application/system to display weekly news updates, event information, and Quote of the Week from Ismaili Institution Canada (IIC).

Permissions to access the Internet/storage settings of the device are needed to establish a client-server system. The application informs the user of the changes that the application will make to their device. This gives the user an option to “opt-out” of the system if they do not feel comfortable, making sure the user is aware of the application’s operation and trust this new system the IIC is trying to implement<sup>3</sup>.

### Establishing reading/writing device storage permissions

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

The following permissions are defined for the application in order:

- Check current WiFi state
- Access the Internet
- Check Internet connection state
- Write to the external storage
- Read from the external storage
- Read the phone's memory and processing state

---

<sup>3</sup> Criterion B, Specification 2: Application must be user friendly and work smoothly with the user.

A thread is launched to download the news items from the host when the “Enter” button is pressed. The thread establishes a back-end connection with the IIC database, while the user continues to scroll through the present items in the application, so that the update process is parallel and does not interrupt application usage<sup>4</sup>. This concurrent programming makes the interface more fluid, and user-friendly<sup>5</sup> as there is no delay in the loading of items.

### downloadThread.java file showing the background thread process when downloading news

```
import ...  
/*  
 * Created by Zain on 12/25/2014.  
 */  
  
//download thread  
public class downloadThread extends AsyncTask<Void, Void, String> {  
  
    public downloadThread(Context context) {  
    }  
  
    @Override  
    protected void onPreExecute() {  
    }  
  
    //perform these operations in the background thread  
    @Override  
    protected String doInBackground(Void... params) {  
  
        String fileName = "thisweek.json";           //create this file on device  
        String DownloadUrl = "http://zainzulfiqar.com/thisweek.json";      //download from this URL  
  
        //download the file  
        try {  
            File root = android.os.Environment.getExternalStorageDirectory();  
  
            //make a folder on the root directory, names AlAkbar  
            File dir = new File (root.getAbsolutePath() + "/AlAkbar");  
            if(dir.exists()==false) {  
                dir.mkdirs();  
            }  
  
            URL url = new URL(DownloadUrl); //use the link specified before  
            File file = new File(dir, fileName);  
  
            //log what is happening for easy debugging  
            long startTime = System.currentTimeMillis();  
            Log.d("DownloadManager", "download beginning");  
            Log.d("DownloadManager", "download url:" + url);  
            Log.d("DownloadManager", "downloaded file name:" + fileName);  
        } catch (Exception e) {  
            Log.e("DownloadManager", "Error in download thread: " + e.getMessage());  
        }  
    }  
}
```

Java method run in the background to maintain concurrency

URL to fetch database file from

Check if folder directory already exists

Define and create the directory to save the downloaded file in

<sup>4</sup> "AsyncTask." AsyncTask | Android Developers. Accessed January 17, 2015. <http://developer.android.com/reference/android/os/AsyncTask.html>.

<sup>5</sup> Criterion B, Specification 2: Application must be user friendly and work smoothly with the user.

## Offline application service by reading/writing files in internal storage

Bufferstream is used to save the downloaded database content onto storage for offline usage. Users can check for reminders/important dates and time while away from home, on the go, making the application more convenient. Reading operations with files are slow. To minimize the operations from the online database, a buffer stream is used to store the database file into RAM and save time when processing the file onto the local storage<sup>6</sup>.

downloadThread.java file downloading the information and saving the bytes into a file

```
// Open a connection to that URL.  
URLConnection ucon = url.openConnection();  
  
//Define InputStreams to read from the URLConnection.  
InputStream is = ucon.getInputStream();  
BufferedInputStream bis = new BufferedInputStream(is);  
  
//Read bytes to the Buffer until there is nothing more to read(-1).  
ByteArrayBuffer baf = new ByteArrayBuffer(5000);  
int current = 0;  
while ((current = bis.read()) != -1) {  
    baf.append((byte) current);  
}  
  
// Convert the Bytes read to a String.  
FileOutputStream fos = new FileOutputStream(file);  
fos.write(baf.toByteArray());  
fos.flush();  
fos.close();
```

Establish a buffer stream to write bytes of data into a file

Save file into a readable String so that the contents can be understood by Java

<sup>6</sup> "BufferedInputStream." BufferedInputStream | Android Developers. Accessed January 18, 2015. <http://developer.android.com/reference/java/io/BufferedInputStream.html>

Internet connection is checked to ensure the database can be downloaded. Based on this, the application either launches the Internet settings, or the offline system. Nested if statements help to determine the connection state by returning Boolean values.<sup>7</sup> This system prevents any confusion of why content may not be updating properly or why news items don't appear.

In the home.java file, check to see if internet connection is available, to see if download thread can be launched

```
//check to see if connected to network
public boolean isConnected(Context context) {

    ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo netinfo = cm.getActiveNetworkInfo();

    if (netinfo != null && netinfo.isConnectedOrConnecting()) {
        android.net.NetworkInfo wifi = cm.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
        android.net.NetworkInfo mobile = cm.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);

        if((mobile != null && mobile.isConnectedOrConnecting()) || (wifi != null && wifi.isConnectedOrConnecting())) return true;
        else return false;
    } else
        return false;
}
```

Check to see if local WiFi connection

Or if it is a 3G/mobile connection

Return **true** if any of these conditions are met/in progress (internet connection available), otherwise return **false**

<sup>7</sup> "Detect Whether There Is an Internet Connection Available on Android." Stack Overflow. Accessed January 17, 2015. <http://stackoverflow.com/questions/4238921/detect-whether-there-is-an-internet-connection-available-on-android>.

During first time application use, there is no database file in storage. Each scenario has its own conditional if statement to ensure application runs well all the time no matter what the connection state. This makes the application easier to use, as the user does not have to deal with Internet error handling when fetching the news.

Check to see if Internet connection is needed, if database file already exists, and if offline content is available

```
File root = android.os.Environment.getExternalStorageDirectory();
File file = new File (root.getAbsolutePath() + "/AlAkhbar/thisweek.json");
if(!file.exists() && !isConnected(home.this)) {           //if file is not there
    System.out.println(""+file.exists());
    buildDialog(home.this).show();
}

else if(file.exists() && !isConnected(home.this)) {           //if file is there, and connection is not there
    Intent i = new Intent(view.getContext(), quoteScreen.class);
    startActivity(i);
}

else if(isConnected(home.this)) {           //if file is there, and connection is there
    downloadNews.execute();
    Intent i = new Intent(view.getContext(), quoteScreen.class); //setup the intent
    startActivity(i);           //start the new screen
}
```

If file and internet connection does not exist (first time app use), bring dialog box to redirect user to the internet settings page

If file exists, but no internet connection, use the file and display the content on a new screen (offline connection)

If internet exists (even if file does or doesn't exist), launch the download thread to update database file onto user's storage, and show the content.

A dialog box informs the user if connection needed. The reason for the connection is explained, and the user is taken to settings. Dialog boxes help guide the user in the direction of the application process. Users cannot perform any other action that may crash the application. This ensures the application runs as it was designed, and that the user avoids accidental problems<sup>8</sup>.

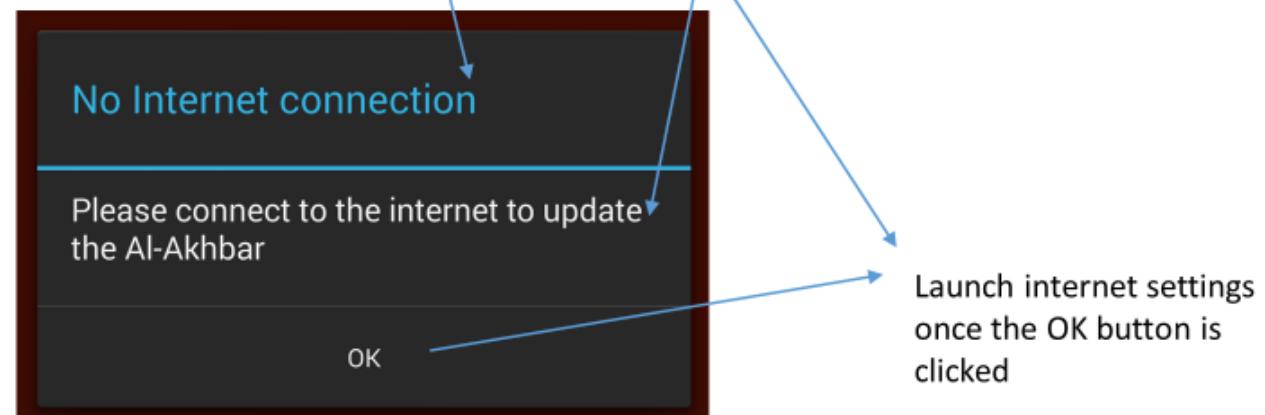
If no internet connection, and no database file in local storage (first time use), spawn dialog box for internet connection

```
//spawn an alert dialog to inform the user that there is no Internet connection
public AlertDialog.Builder buildDialog(Context c) {

    AlertDialog.Builder builder = new AlertDialog.Builder(c, AlertDialog.THEME_HOLO_DARK);
    builder.setTitle("No Internet connection");
    builder.setMessage("Please connect to the internet to update the Al-Akhbar for first-time use.");

    builder.setPositiveButton("OK", (dialog, which) -> {
        dialog.dismiss(); //on clicking the OK button, do this
        startActivity(new Intent(Settings.ACTION_WIFI_SETTINGS)); //launch the Android system
    });

    return builder;
}
```



<sup>8</sup> Criterion B, Specification 2: Application must be user friendly and work smoothly with the user.

The downloaded JSON file is filtered to display the relevant items in each tab<sup>9</sup>. This organization helps the user see relevant details in the one location. The user is able to see information that relates to their interests or needs, instead of getting overwhelmed with unnecessary details. News items are stored in array hash-map key-value pairs to package details together as one object, efficiently passing information throughout the application. Packaging items together ensures all related content is delivered in one compressed process to reduce CPU power.

Update news content from news database file after parsing the JSON formatted file in local storage  
– eventListScreen.java, moreScreen.java, and infoListScreen.java

```
//read the downloaded JSON file from online SQL server, and parse it into a string format
try {
    File yourFile = new File(Environment.getExternalStorageDirectory(), "AlAkbar/thisweek.json");
    FileInputStream stream = new FileInputStream(yourFile);
    try {
        FileChannel fc = stream.getChannel();
        MappedByteBuffer bb = fc.map(FileChannel.MapMode.READ_ONLY, 0, fc.size());
        jsonStr = Charset.defaultCharset().decode(bb).toString();
    }
    finally {
        stream.close();
    }
}

//create a JSON object, to parse file
JSONObject jsonObj = new JSONObject(jsonStr);

// specify the table name (table name is an array, that contains all the element of the table) of the JSON file
news = jsonObj.getJSONArray(NEWSTHISWEEK);

// looping through all the elements in the table
for (int i = 0; i < news.length(); i++) {
    JSONObject c = news.getJSONObject(i);

    String title = c.getString(TITLE);
    String body = c.getString(BODY);
    String subtitle = c.getString(SUBTITLE);
    String newsType = c.getString(NEWSTYPE);

    //use if statement to filter what content we want to display in this activity
    if (newsType.equals("e")) {
        newsType = "Event";

        // tmp hashmap for single contact
        newsItems = new HashMap<String, String>();

        // adding each child node to HashMap key => value
        newsItems.put(TITLE, title);
        newsItems.put(BODY, body);
        newsItems.put(SUBTITLE, subtitle);
        newsItems.put(NEWSTYPE, newsType);

        // adding news to news list
        newsList.add(newsItems);
    }
}
```

Use a for loop to go through **all** of the news elements in the database file, until no more records to read

Read news database file from storage and stream it into a String variable

Create JSON object in Java to interpret JSON formatting of database file in the String variable

Special string variable to store each element of the news from the JSON database file.

Filter through to take only the news relevant to the tab user is on. If user on events tab, only news with “e” tag taken, if user on information tab, only news with “i” tag taken.

Combine the news elements to create one news element to add into the list presented to user.

<sup>9</sup> "How to Build a Mobile App with an App Engine Backend (Tutorial)." Google Developers. Accessed January 18, 2015.

## Online database to manage news items

Database file is hosted on the GoDaddy server 24/7. This means users can access news item updates at any time that is convenient for them. All users are updated instantaneously and simultaneously<sup>10</sup>.

Online database file hosted through GoDaddy domain servers – [zainzulfiqar.com/thisweek.json](http://zainzulfiqar.com/thisweek.json)

The screenshot shows the cPanel File Manager interface. The left sidebar displays a tree view of the directory structure under '/home/puresonic'. The 'public\_html' folder is expanded, showing its contents: etc, logs, mail, public\_ftp, public\_html (which is selected), and tmp. The main panel lists files in the 'public\_html' directory:

	Name	Size
📁	projects	4 KB
📁	webDown	4 KB
📄	CNAME	22 bytes
📄	index.html	3.02 KB
📄	sitemap.xml	521 bytes
📄	SourceSansPro-Light.otf	121.56 KB
📄	styles.css	1.34 KB
📄	thisweek.json	25.95 KB

A blue arrow points from the 'thisweek.json' row in the table to the URL 'http://zainzulfiqar.com/thisweek.json' located below the table.

Database file exported to local website storage, hosted 24/7 on GoDaddy

<sup>10</sup> Criterion B, Specification 5: Create a server-end application to allow IIC committee members to input information into the database and keep the users updated.

The online database uses SQL or a visual GUI to input news records. The system is user-friendly, allowing the IIC to update their users in a fast and easy process. IIC users of any skill level can update the database, allowing flexibility when dividing the database entries within the committee.<sup>11</sup>

### Online database hosted through GoDaddy with phpMyAdmin software

Database provides SQL option for advanced users

GUI interface to “import” data, for users without SQL experience

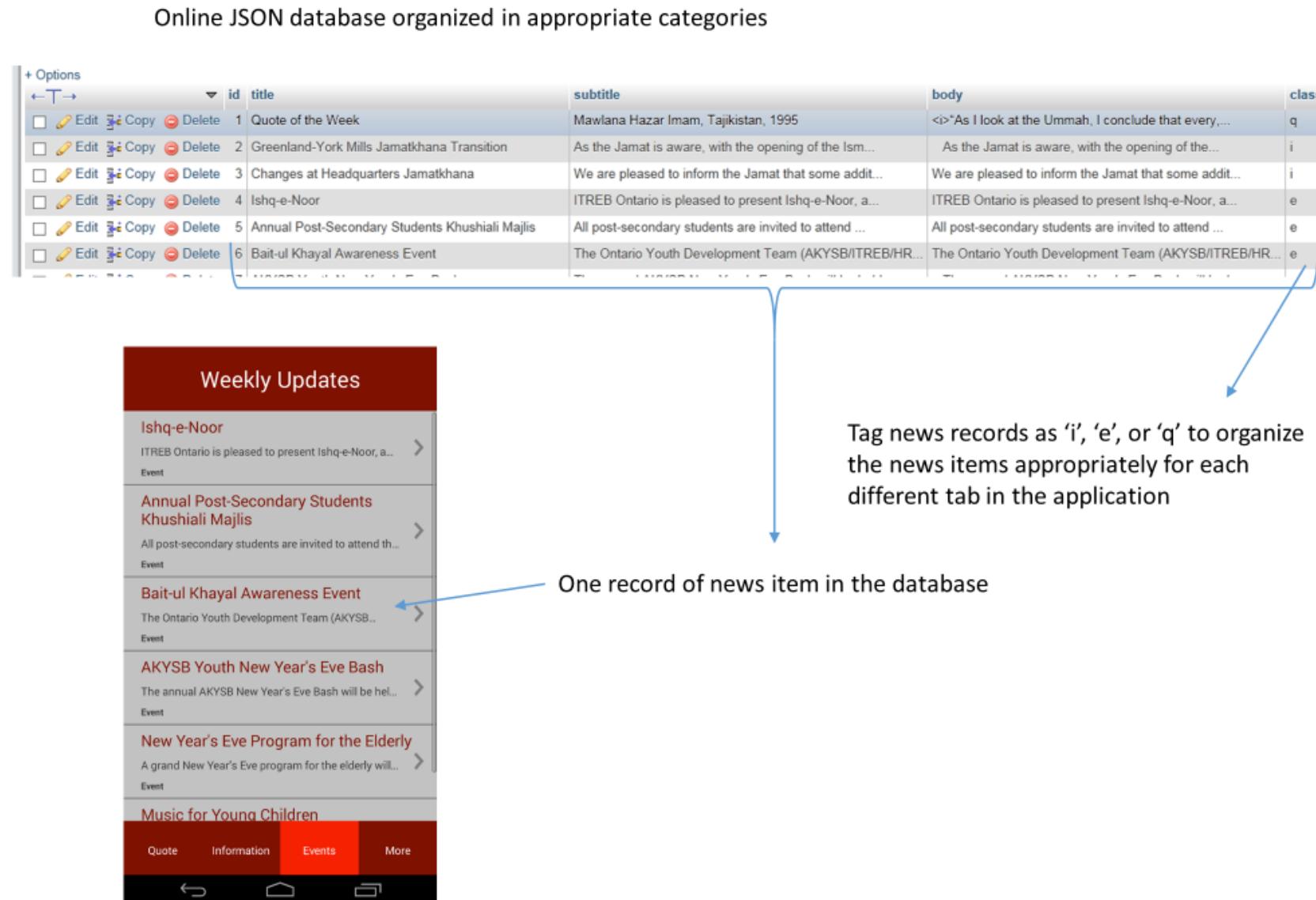
The screenshot shows the phpMyAdmin interface with the following details:

- Left Sidebar:** Shows the "information\_schema" database.
- Top Bar:** Shows "Server: localhost" and various navigation tabs: Structure, SQL, Search, Insert, Export, Import, Privileges, and Opera.
- Server Tab Content:**
  - Traffic:** Received: 2.4 GiB, Sent: 34 GiB, Total: 36.3 GiB.
  - Connections:** max. concurrent connections: 45, Failed attempts: 238 (39.74%, 0.11%), Aborted: 911 (152.1%, 0.42%), Total: 219 k (36.6 k, 100.00%).
  - Processes:** Kill 219216 cpses\_puWlwCG1Up, Kill 219217 cpses\_puWlwCG1Up.
- Bottom Right:** A pink button labeled "SHOW PROCESSLIST".

Database server has been running in the background, ever since application was designed

<sup>11</sup> Criterion B, Specification 5: Create a server-end application to allow IIC committee members to input information into the database and keep the users updated.

“Class” to organize news<sup>12</sup>.

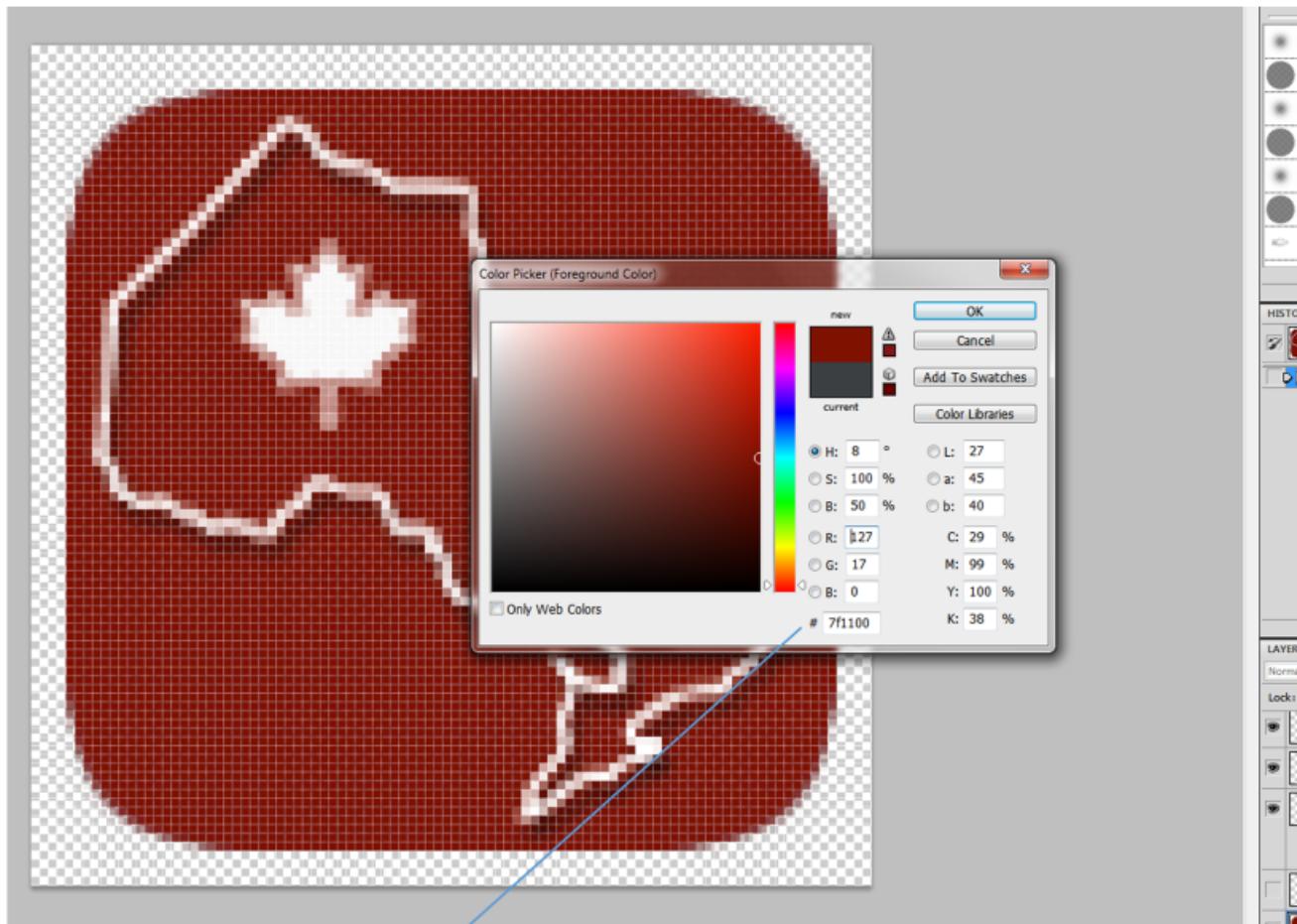


<sup>12</sup> Criterion B, Specification 6: Database entries are user friendly for the IIC, and organize content appropriately and automatically.

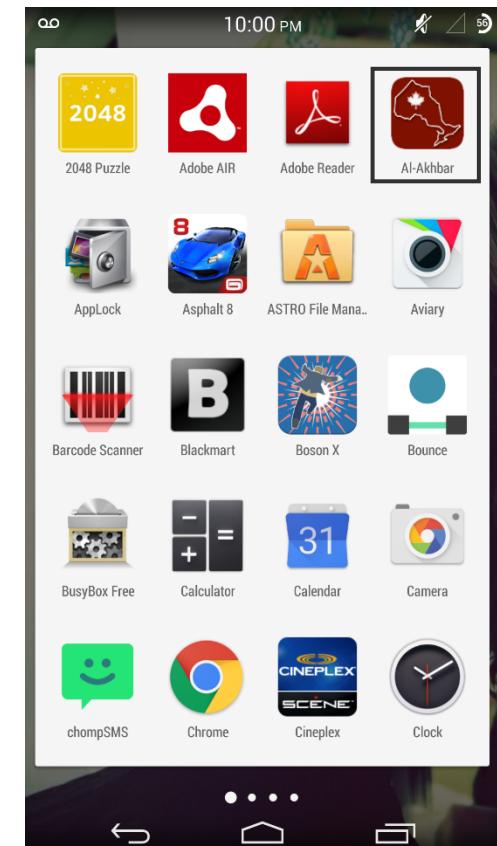
## Use of XML and Java design to create a user-friendly interface

An application logo, designed in Photoshop, needed to reflect the application's purpose: a news service for Ontario Ismailis. The outline of the Ontario map was placed. The background was chosen to be red, so that the color scheme matches the IIC's. Also, the red background helps to quickly find the app icon in the white app drawer<sup>13</sup>.

Custom app logo/icon designed to represent the Android application



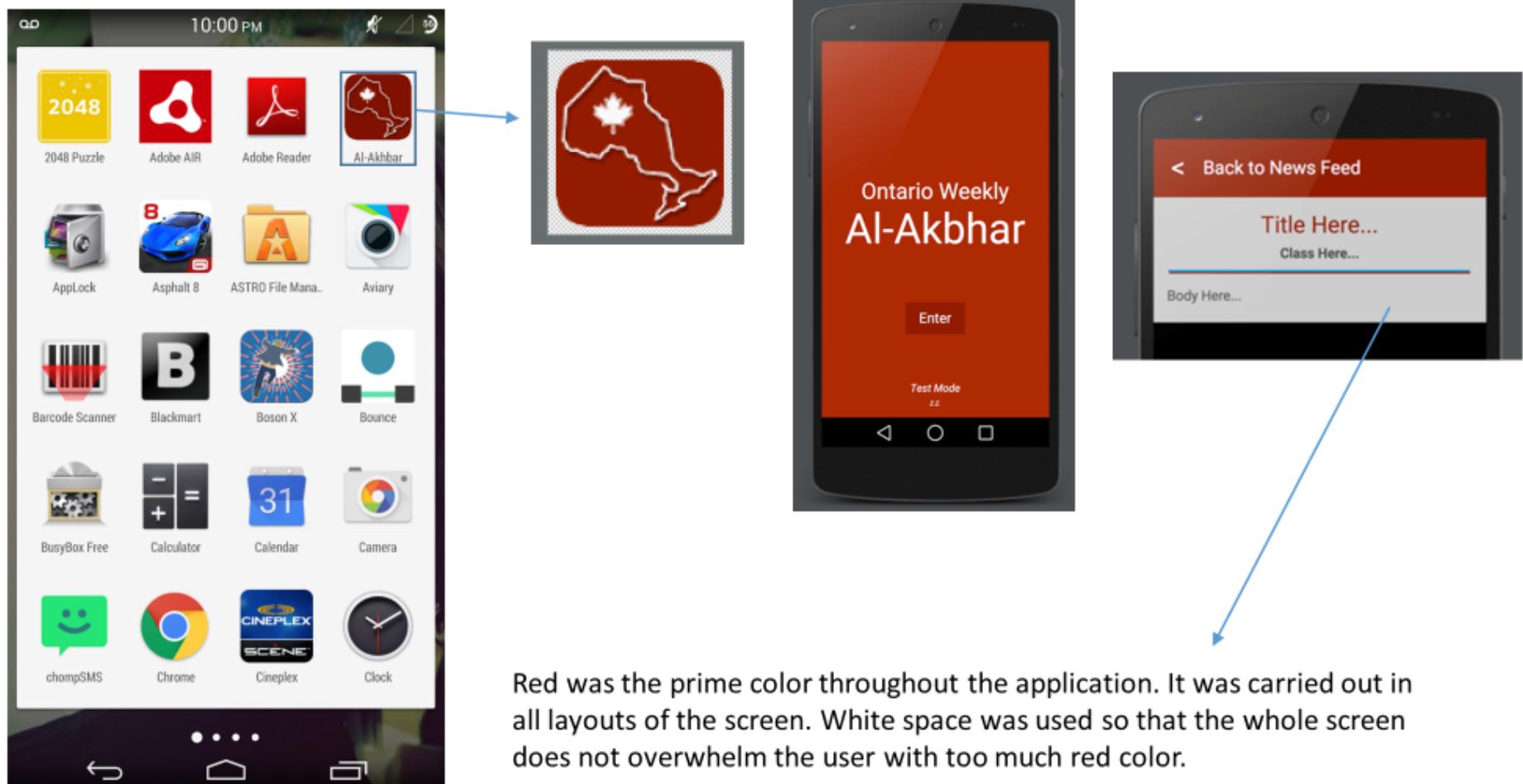
Specific color scheme of red was matched to the IIC's theme, design was implemented in Photoshop



<sup>13</sup> Criterion B, Specification 3: Application should be visually appealing, with a logo.

The IIC uses red in their website, emails, and mail packages. Ismailis in Ontario associate red with the IIC. To represent the same committee, a red color scheme was used. Implementing the graphics used by the IIC into the application became easy, without worrying about color matching or redesigning.<sup>14</sup>

### Red color scheme carried out through the application



Red was the prime color throughout the application. It was carried out in all layouts of the screen. White space was used so that the whole screen does not overwhelm the user with too much red color.

<sup>14</sup> Criterion B, Specification 3: Application should be visually appealing, with a logo.

The IIC can manually place HTML tags into news records to customize specific sections of the items. This helps to guide the user's eye to emphasize important deadlines, highlight hyperlinks, and uniquely format each item for effective information density<sup>15</sup>.

HTML tags are inserted into the database file with the news content so that each news can be individually customized

```
//get data  
Intent i = getIntent();  
String[] newsExpandedValues = i.getStringArrayExtra("newsData");  
  
//update  
TextView title = (TextView) findViewById(R.id.singleTitle);  
TextView newsType = (TextView) findViewById(R.id.singleClass);  
TextView body = (TextView) findViewById(R.id.singleBody);  
title.setText(newsExpandedValues[0]);  
newsType.setText(newsExpandedValues[1]);  
  
// body setup  
body.setText(Html.fromHtml(newsExpandedValues[2]));  
body.setMovementMethod(LinkMovementMethod.getInstance());
```

Java HTML parser, converting the tags into visual styles when user views the news.

```
HTML tags such as <br> to create line  
breaks, <b> to bold specific sections of  
text  
  
body.setText(Html.fromHtml("<b>Ismaili Council for Canada (IIC)</b><br>" +  
    "49 Wynford Drive<br>" +  
    "Toronto, Ontario M3C 1K1<br>" +  
    "CANADA<br>" +  
    "Tel: +1 (416) 646-6965<br><br>" +  
    "<b>Ismaili Council for Ontario</b><br>" +  
    "149-151 Bartley Avenue<br>" +  
    "Toronto, Ontario M4A 1C9<br>" +  
    "CANADA<br>" +  
    "Tel: +1-416-751-4001<br><br><br>" +  
    "<i>Application developed by Zain Zulfiqar</i>"));
```



<sup>15</sup> Criterion B, Specification 3 and 4: Application should be visually appealing, with a logo + application should provide important contact information of the IIC.

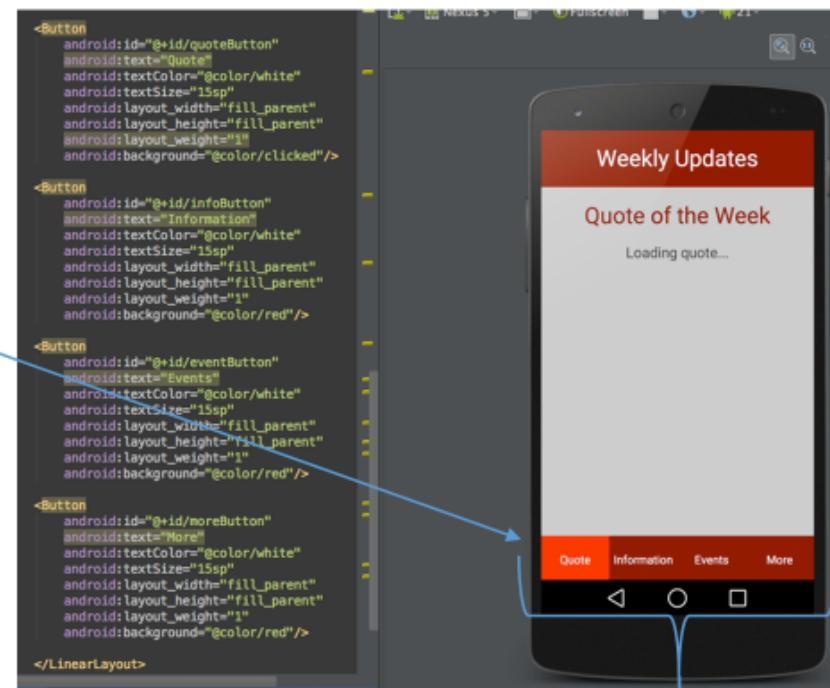
Every tab is designed separately for each screen. The tab on which the user is currently on is highlighted in a brighter red, so that the user is aware of where they are<sup>16</sup>.

Every news item is organized and placed into its appropriate category. These sections are separated through tabs.

```
//initialize the quote screen - setup the layout
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.quote_screen);
    newsList = new ArrayList<HashMap<String, String>>();

    //buttons leading to the other activities in the application
    infoButton = (Button) findViewById(R.id.infoButton);
    infoButton.setOnClickListener(view -> {
        Intent i = new Intent(view.getContext(), infoListScreen.class);
        i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
        overridePendingTransition(0,0); //0 for no animation
        startActivity(i);
    });
    eventButton = (Button) findViewById(R.id.eventButton);
    eventButton.setOnClickListener((view) -> {
        Intent i = new Intent(view.getContext(), eventListScreen.class);
        i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
        overridePendingTransition(0,0); //0 for no animation
        startActivity(i);
    });
    moreButton = (Button) findViewById(R.id.moreButton);
    moreButton.setOnClickListener((view) -> {
        Intent i = new Intent(view.getContext(), moreScreen.class);
        i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
        overridePendingTransition(0,0); //0 for no animation
        startActivity(i);
    });
}
```

A set of buttons to create the tabs, present under each screen. The button onClickListeners link to the different Java activates/screen once tapped. NOTE: buttons will change depending on the screen the user is already on



Tabs designed in XML. Same XML layout used for every screen. The screen the user is currently on will be a brighter red.

<sup>16</sup> Criterion B, Specification 2 and 3: Application must be user friendly and work smoothly with the user + application should be visually appealing, with a logo.

A custom ListView adapter was designed for news items to make the list elements more detailed and visually pleasing<sup>17</sup>. The new ListView displays the title, a description of the news, and states the type of item. This gives more information to the users and they do not have to keep clicking back and forth to see if the item they tapped has the information they need. The new custom design is efficient and helpful<sup>18</sup>.

Default Android ListView elements were changed, and a custom XML based theme was applied

```
//Updating parsed JSON data into ListView using a custom adapter
//All listviews are made of data and adapters. The adapter handles the design, theme, and source of the information.
/*To create a custom adapter, a customized row.xml file was coded to make a theme of the individual listview elements,
and the data is provided using the JSON list.*/
ListAdapter adapter = new SimpleAdapter(
    eventListScreen.this, newsList,
    R.layout.row, new String[] { TITLE, SUBTITLE,
    NEWSTYPE,BODY }, new int[] { R.id.rowTitle,
    R.id.rowSubtitle, R.id.rowClass, R.id.rowBody });

setListAdapter(adapter);
```

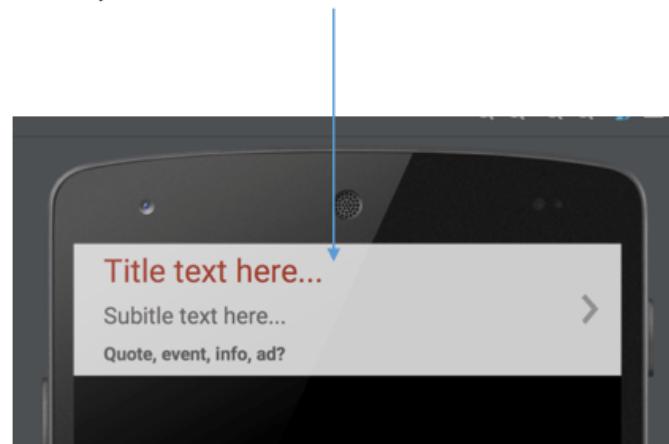
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/row_select">

    <TextView
        android:id="@+id/rowTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Title text here..."
        android:textColor="@color/rowTitle"
        android:textSize="21sp"
        android:paddingLeft="20dp"
        android:paddingStart="20dp"
        android:paddingRight="20dp"
        android:paddingEnd="20dp"
        android:paddingTop="6dp"
        android:paddingBottom="8dp"/>

    <TextView...>
    <TextView...>
    <TextView...>
    <Imageview...>
</RelativeLayout>
```

All news items parsed through the JSON database file are displayed through this adapter.

XML code used to design each individual element of one news item displayed in the list. The title, subtitle, and body of the news all have different themes.



<sup>17</sup> "Android Custom ListView with Image and Text." *Android Custom ListView with Image and Text*. Accessed January 16, 2015.

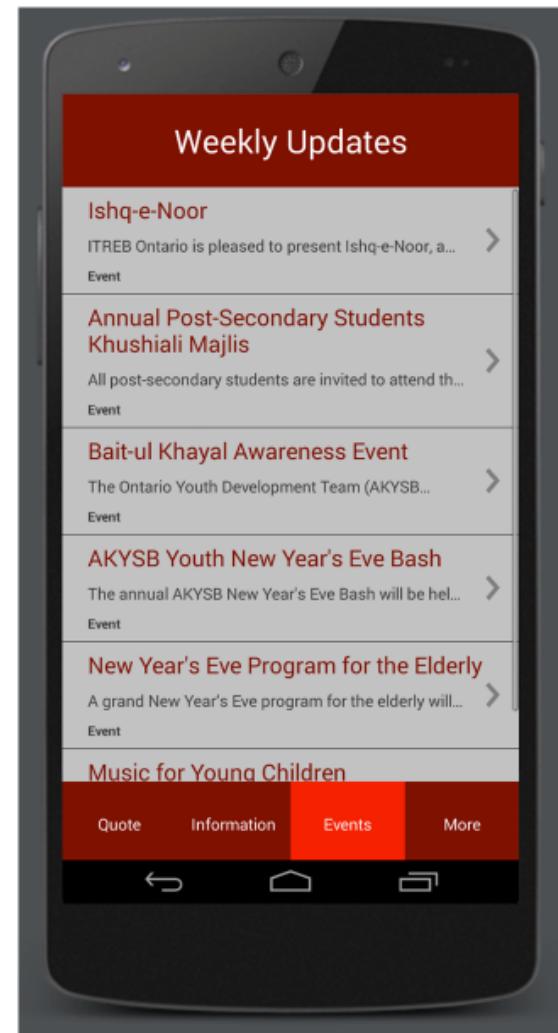
<sup>18</sup> Criterion B, Specification 1 and 3: Create an application/system to display weekly news updates, event information, and Quote of the Week from Ismaili Institution Canada (IIC) + application should be visually appealing, with a logo.

Old Android ListView theme VS. the new custom XML based ListView adapter designed for the news list

Old



New

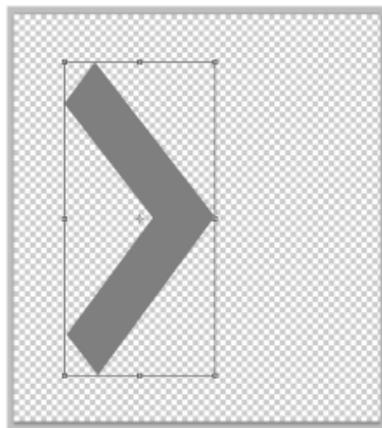
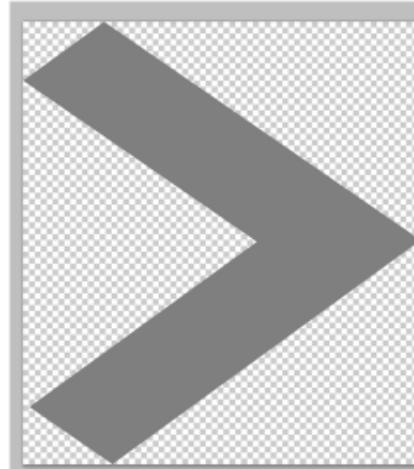


This graphic was manipulated in Photoshop<sup>19</sup>. The magic wand tool was used to remove the white background for transparency. It was scaled and skewed down so that the arrow has a slim design. This arrow was placed at the end of the custom ListView adapter to give it a “click to expand” type operation so that the user knows there are additional details.

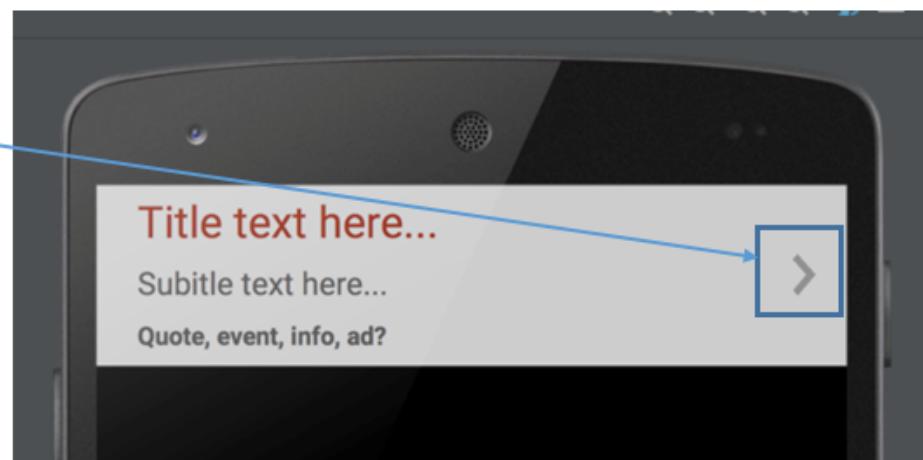
Manipulation of graphics to aid the user. An arrow was added on the side of every news element.



Magic wand in Photoshop was used to delete the white background and make the arrow transparent



Arrow was scaled and resized to make it smaller. The slim design bring a clean design to every news item.

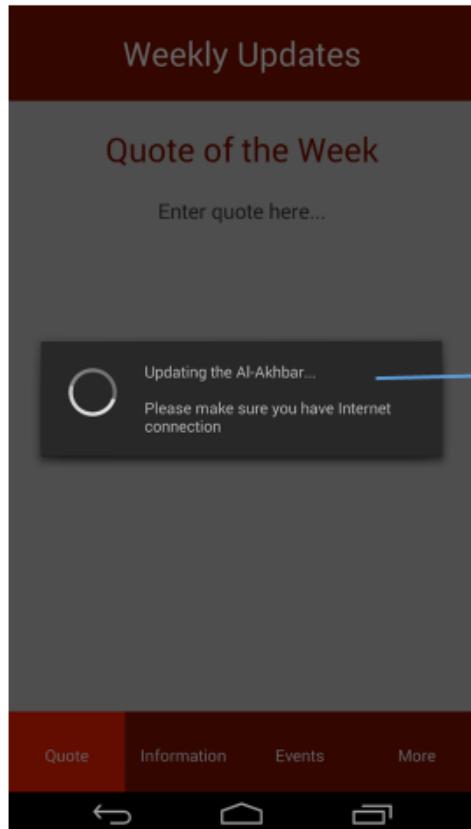


<sup>19</sup> "Right\_grey\_arrow Clip Art." Right Grey Arrow Clip Art. Accessed December 27, 2014. <http://www.clker.com/clipart-right-grey-arrow.html>

An important principle of design is to ensure the user is never left alone/confused. When the news items are being populated into the ListView, each device will take a different amount of processing time. A dialog box with a progress bar displayed while the ListView gets updated educates the user of all the processes – important for creating a fluid user experience<sup>20</sup>.

Dialog boxes were placed while background thread downloadThread.java downloads the news

```
@Override  
protected void onPreExecute() {  
    super.onPreExecute();  
    //launch a dialog box to keep main system process busy with GUI, while the background thread handles the file reading  
    pDialog = new ProgressDialog(eventListScreen.this, ProgressDialog.THEME_HOLO_DARK);  
    pDialog.setMessage("Updating the Al-Akhbar...\\n\\nPlease make sure you have Internet connection");  
    pDialog.setCancelable(false);  
    pDialog.show();  
}
```



Progress bar helps keep the user updated on the download process of the news.

997 words

<sup>20</sup> Criterion B, Specification 2: Application must be user friendly and work smoothly with the user.

## downloadThread.java

```
1 package com.example.zain.alakhbar;
2
3 import android.content.Context;
4 import android.os.AsyncTask;
5 import android.util.Log;
6
7 import org.apache.http.util.ByteArrayBuffer;
8
9 import java.io.BufferedInputStream;
10 import java.io.File;
11 import java.io.FileOutputStream;
12 import java.io.IOException;
13 import java.io.InputStream;
14 import java.net.URL;
15 import java.net.URLConnection;
16
17 /**
18 * Created by Zain on 12/25/2014.
19 */
20
21 //download thread
22 public class downloadThread extends AsyncTask<Void, Void, String> {
23
24     public downloadThread(Context context) {
25
26     }
27
28     @Override
29     protected void onPreExecute() {
30
31     }
32
33     //perform these operations in the background thread
34     @Override
35     protected String doInBackground(Void... params) {
36
37         String fileName = "thisweek.json";           //create this file on device
38         String DownloadUrl = "http://zainzulfiqar.com/thisweek.json";      //download from this URL
39
40         //download the file
41         try {
42             File root = android.os.Environment.getExternalStorageDirectory();
43
```

Screen snapshots of the Java classes/code

```
44     //make a folder on the root directory, names AlAkhbar
45     File dir = new File (root.getAbsolutePath() + "/AlAkhbar");
46     if(dir.exists()==false) {
47         dir.mkdirs();
48     }
49
50     URL url = new URL(DownloadUrl); //use the link specified before
51     File file = new File(dir, fileName);
52
53     //log what is happeneing for easy debugging
54     long startTime = System.currentTimeMillis();
55     Log.d("DownloadManager", "download begining");
56     Log.d("DownloadManager", "download url:" + url);
57     Log.d("DownloadManager", "downloaded file name:" + fileName);
58
59     // Open a connection to that URL.
60     URLConnection ucon = url.openConnection();
61
62     //Define InputStreams to read from the URLConnection.
63     InputStream is = ucon.getInputStream();
64     BufferedInputStream bis = new BufferedInputStream(is);
65
66     //Read bytes to the Buffer until there is nothing more to read.
67     ByteBuffer baf = new ByteBuffer(5000);
68     int current = 0;
69     while ((current = bis.read()) != -1) {
70         baf.append((byte) current);
71     }
72
73     // Convert the Bytes read to a String.
74     FileOutputStream fos = new FileOutputStream(file);
75     fos.write(baf.toByteArray());
76     fos.flush();
77     fos.close();
78     Log.d("DownloadManager", "download ready in" + ((System.currentTimeMillis() - startTime) / 1000) + " sec");
79
80 } catch (IOException e) {
81     Log.d("DownloadManager", "Error: " + e);
82 }
83
84     return null;
85 }
86
87 @Override
88 protected void onPostExecute(String result) {
89
90 }
91 }
```

## eventListScreen.java

```
1 package com.example.zain.alakhbar;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.nio.MappedByteBuffer;
6 import java.nio.channels.FileChannel;
7 import java.nio.charset.Charset;
8 import java.util.ArrayList;
9 import java.util.HashMap;
10
11 import org.json.JSONArray;
12 import org.json.JSONObject;
13
14 import android.app.ListActivity;
15 import android.app.AlertDialog;
16 import android.content.Intent;
17 import android.os.AsyncTask;
18 import android.os.Bundle;
19 import android.os.Environment;
20 import android.util.Log;
21 import android.view.View;
22 import android.widget.AdapterView;
23 import android.widget.AdapterView.OnItemClickListener;
24 import android.widget.Button;
25 import android.widgetListAdapter;
26 import android.widget.ListView;
27 import android.widget.SimpleAdapter;
28 import android.widget.TextView;
29
30 public class eventListScreen extends ListActivity {
31
32     //private ProgressDialog pDialog;
33
34     // URL to get JSON
35     private static String url = "http://zainzulfiqar.com/thisweek.json";
36
37     // JSON Node names
38     private static final String TITLE = "title";
39     private static final String BODY = "body";
40     private static final String SUBTITLE = "subtitle";
41     private static final String NEWSTYPE = "class";
42     private static final String NEWSTHISWEEK = "newsThisWeek";
43
44     private Button quoteButton = null;
45     private Button infoButton = null;
46     private Button moreButton = null;
47
48     // news JSONArray
49     JSONArray news = null;
50
51     // Hashmap for ListView
52     ArrayList<HashMap<String, String>> newsList;
53     HashMap<String, String> newsItems = new HashMap<String, String>();
54
55     //initialize the event screen - setup the layout
56     @Override
57     public void onCreate(Bundle savedInstanceState) {
58         super.onCreate(savedInstanceState);
59         setContentView(R.layout.event_list);
60
61         //buttons leading to the other activities in the application
62         quoteButton = (Button) findViewById(R.id.quoteButton);
63         quoteButton.setOnClickListener(new View.OnClickListener() {
64             @Override
65             public void onClick(View view) {
66                 Intent i = new Intent(view.getContext(), quoteScreen.class);
67                 i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
68                 overridePendingTransition(0,0); //0 for no animation
69                 startActivity(i);
70             }
71         });
72         infoButton = (Button) findViewById(R.id.infoButton);
73         infoButton.setOnClickListener(new View.OnClickListener() {
74             @Override
75             public void onClick(View view) {
76                 Intent i = new Intent(view.getContext(), infoListScreen.class);
77                 i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
78                 overridePendingTransition(0,0); //0 for no animation
79                 startActivity(i);
80             }
81         });
82         moreButton = (Button) findViewById(R.id.moreButton);
83         moreButton.setOnClickListener(new View.OnClickListener() {
84             @Override
85             public void onClick(View view) {
86                 Intent i = new Intent(view.getContext(), moreScreen.class);
87                 i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
88                 overridePendingTransition(0,0); //0 for no animation
89                 startActivity(i);
90             }
91         });
92
93         //initialize the listview
```

```

94     newsList = new ArrayList<HashMap<String, String>>();
95     ListView lv = getListView();
96
97     // Listview on item click listener, launch a seperate activity to show the expanded version of the news
98     lv.setOnItemClickListener(new OnItemClickListener() {
99
100         @Override
101         public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
102             // transfer the listview details to a new class to show individual expanded details
103
104             String title = ((TextView) view.findViewById(R.id.rowTitle))
105                 .getText().toString();
106             String newsType = ((TextView) view.findViewById(R.id.rowClass))
107                 .getText().toString();
108             String body = ((TextView) view.findViewById(R.id.rowBody))
109                 .getText().toString();
110
111             Intent i = new Intent(view.getContext(), singleNews.class);
112             String[] newsValues = new String[] {title, newsType, body};
113             i.putExtra("newsData", newsValues); //bundle it into a newsData named package - easily transfer information together
114             startActivity(i);
115         }
116     });
117
118     //launch the JSON processing script to populate the listview
119     new GetNews().execute();
120 }
121
122 //Async task class to read JSON file from device
123 private class GetNews extends AsyncTask<Void, Void, Void> {
124
125     @Override
126     protected void onPreExecute() {
127         super.onPreExecute();
128         /*//launch a dialog box to keep main system process busy with GUI, while the background thread handles the file reading - also so
129         pDialog = new ProgressDialog(eventListScreen.this, ProgressDialog.THEME_HOLO_DARK);
130         pDialog.setMessage("Updating the Al-Akhbar...\\n\\nPlease make sure you have Internet connection");
131         pDialog.setCancelable(false);
132         pDialog.show();*/
133     }
134
135     //perform these operations in the background thread
136     @Override
137     protected Void doInBackground(Void... arg0) {
138         // Creating the JSON string for data
139         String jsonStr = null;
140
141         Log.d("Response: ", "> " + jsonStr);
142
143         //read the downloaded JSON file from online SQL server, and parse it into a string format
144         try {
145             File yourFile = new File(Environment.getExternalStorageDirectory(), "AlAkhbar/thisweek.json");
146             FileInputStream stream = new FileInputStream(yourFile);
147             try {
148                 FileChannel fc = stream.getChannel();
149                 MappedByteBuffer bb = fc.map(FileChannel.MapMode.READ_ONLY, 0, fc.size());
150
151                 jsonStr = Charset.defaultCharset().decode(bb).toString();
152             }
153             finally {
154                 stream.close();
155             }
156
157             //create a JSON object, to parse file
158             JSONObject jsonObj = new JSONObject(jsonStr);
159
160             // specify the table name (table name is an array, that contains all the element of the table) of the JSON file
161             news = jsonObj.getJSONArray(NEWSTHISWEEK);
162
163             // looping through all the elements in the table
164             for (int i = 0; i < news.length(); i++) {
165                 JSONObject c = news.getJSONObject(i);
166
167                 String title = c.getString(TITLE);
168                 String body = c.getString(BODY);
169                 String subtitle = c.getString(SUBTITLE);
170                 String newsType = c.getString(NEWSTYPE);
171
172                 //use if statement to filter what content we want to display in this activity
173                 if (newsType.equals("e")) {
174                     newsType = "Event";
175
176                     // tmp hashmap for single contact
177                     newsItems = new HashMap<String, String>();
178
179                     // adding each child node to HashMap key => value
180                     newsItems.put(TITLE, title);
181                     newsItems.put(BODY, body);
182                     newsItems.put(SUBTITLE, subtitle);
183                     newsItems.put(NEWSTYPE, newsType);
184
185                     // adding news to news list
186                     newsList.add(newsItems);
187                 }
188             }
189         } catch (Exception e) {
190             e.printStackTrace();

```

```
191         }
192     }
193     return null;
194 }
195
196 //after all of the above processes are done, launch the following
197 @Override
198 protected void onPostExecute(Void result) {
199     super.onPostExecute(result);
200     // Dismiss the progress dialog
201     /*if (pDialog.isShowing())
202      pDialog.dismiss();*/
203
204     //Updating parsed JSON data into ListView using a custom adapter
205     //All listviews are made of data and adapters. The adapter handles the design, theme, and source of the information.
206     /*To create a custom adapter, a customized row.xml file was coded to make a theme of the individual listview elements,
207     and the data is provided using the JSON list.*/
208    ListAdapter adapter = new SimpleAdapter(
209             eventListScreen.this, newsList,
210             R.layout.row, new String[] { TITLE, SUBTITLE,
211             NEWSTYPE, BODY }, new int[] { R.id.rowTitle,
212             R.id.rowSubtitle, R.id.rowClass, R.id.rowBody });
213
214     setListAdapter(adapter);
215 }
216
217 }
218 }
```

## home.java

```
1 package com.example.zain.alakhbar;
2
3 import android.app.Activity;
4 import android.app.AlertDialog;
5 import android.content.Context;
6 import android.content.DialogInterface;
7 import android.content.Intent;
8 import android.net.ConnectivityManager;
9 import android.net.NetworkInfo;
10 import android.os.Bundle;
11 import android.os.Environment;
12 import android.provider.Settings;
13 import android.view.Menu;
14 import android.view.MenuItem;
15 import android.view.View;
16 import android.os.Build;
17 import android.widget.Button;
18 import android.widget.ListView;
19 import android.widget.TextView;
20 import android.app.ActionBar;
21 import android.app.Fragment;
22 import android.view.LayoutInflater;
23 import android.view.ViewGroup;
24 import java.io.BufferedInputStream;
25 import java.io.File;
26 import java.io.FileOutputStream;
27 import java.io.IOException;
28 import java.io.InputStream;
29 import java.net.URL;
30 import java.netURLConnection;
31 import org.apache.http.util.ByteArrayBuffer;
32 import android.app.Activity;
33 import android.os.Bundle;
34 import android.widget.TextView;
35 import android.widget.Toast;
36
37 import java.util.ArrayList;
38
39
40 public class home extends Activity {
41
42     //create a button object
43     private Button menuButton = null;
44
45     //initialize what happens at the start of this activity
46     @Override
47     protected void onCreate(Bundle savedInstanceState) {
48         super.onCreate(savedInstanceState);
49         setContentView(R.layout.activity_home);          //use the XML activity_home layout file to setup the screen
50
51         //setup the download thread - to update the news, "this" refers to launching it on the current activity/screen
52         final downloadThread downloadNews = new downloadThread(this);
53
54         //setup a listener for the button object and link it to the Enter button so that user can go to next screen
55         menuButton = (Button) findViewById(R.id.menuButton);           //use the XML menuButton layout file to setup the button
56         menuButton.setOnClickListener(new View.OnClickListener() {
57             @Override
58             public void onClick(View view) {
59                 File root = android.os.Environment.getExternalStorageDirectory();
60                 File file = new File (root.getAbsolutePath() + "/AlAkhbar/thisweek.json");           //check to see if file exists
61                 if(!file.exists() && !isConnected(home.this)) {           //if file is not there, and no Internet, this is a first time connection
62                     System.out.println(""+file.exists());
63                     buildDialog(home.this).show();
64                 }
65             }
66         }
67     }
68 }
```

```
66         else if(file.exists() && !isConnected(home.this)) { //if file is there, and no Internet, you can still view the news stored on internal storage
67             Intent i = new Intent(view.getContext(), quoteScreen.class);
68             startActivity(i);
69         }
70
71         else if(isConnected(home.this)) { //if file is there, and connected to Internet, update to find latest file on database, and then view the news
72             downloadNews.execute();
73             Intent i = new Intent(view.getContext(), quoteScreen.class); //setup the new activity, referring to the class file of the new screen
74             startActivity(i); //start the new screen
75         }
76     });
77 }
78
79 //check to see if connected to network
80 public boolean isConnected(Context context) {
81
82     ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
83     NetworkInfo netinfo = cm.getActiveNetworkInfo();
84
85     if (netinfo != null && netinfo.isConnectedOrConnecting()) {
86         android.net.NetworkInfo wifi = cm.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
87         android.net.NetworkInfo mobile = cm.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
88
89         if((mobile != null && mobile.isConnectedOrConnecting()) || (wifi != null && wifi.isConnectedOrConnecting())) return true;
90         else return false;
91     } else
92         return false;
93 }
94
95 //spawn an alert dialog to inform the user that there is no Internet connection
96 public AlertDialog.Builder buildDialog(Context c) {
97
98     AlertDialog.Builder builder = new AlertDialog.Builder(c, AlertDialog.THEME_HOLO_DARK);
99     builder.setTitle("No Internet connection");
100    builder.setMessage("Please connect to the internet to update the Al-Akhbar for first-time use.");
101
102    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
103
104        @Override
105        public void onClick(DialogInterface dialog, int which) { //on clicking the OK button, do this...
106            dialog.dismiss();
107            startActivity(new Intent(Settings.ACTION_WIFI_SETTINGS)); //launch the Android system WiFi settings menu
108        }
109    });
110
111    return builder;
112 }
113 }
114 }
```

## infoListScreen.java

```
1 package com.example.zain.alakhbar;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.MappedByteBuffer;
6 import java.io.channels.FileChannel;
7 import java.nio.charset.Charset;
8 import java.util.ArrayList;
9 import java.util.HashMap;
10
11 import org.json.JSONArray;
12 import org.json.JSONObject;
13
14 import android.app.ListActivity;
15 import android.app.ProgressDialog;
16 import android.content.Intent;
17 import android.os.AsyncTask;
18 import android.os.Bundle;
19 import android.os.Environment;
20 import android.os.Parcelable;
21 import android.util.Log;
22 import android.view.View;
23 import android.widget.AdapterView;
24 import android.widget.AdapterView.OnItemClickListener;
25 import android.widget.Button;
26 import android.widgetListAdapter;
27 import android.widget.ListView;
28 import android.widget.SimpleAdapter;
29 import android.widget.TextView;
30
31 public class infoListScreen extends ListActivity {
32
33     //private ProgressDialog pDialog;
34     private static Parcelable mListviewScrollPos = null;
35
36     // URL to get JSON
37     private static String url = "http://zainzulfiqar.com/thisweek.json";
38
39     // JSON Node names
40     private static final String TITLE = "title";
41     private static final String BODY = "body";
42     private static final String SUBTITLE = "subtitle";
43     private static final String NEWSTYPE = "class";
44     private static final String NEWSTHISWEEK = "newsThisWeek";
45
46     private Button quoteButton = null;
47     private Button eventButton = null;
48     private Button moreButton = null;
49
50     // news JSONArray
51     JSONArray news = null;
52
53     // Hashmap for ListView
54     ArrayList<HashMap<String, String>> newList;
55     HashMap<String, String> newsItems = new HashMap<String, String>();
56
57     //initialize the info screen - setup the layout
58     @Override
59     public void onCreate(Bundle savedInstanceState) {
60         super.onCreate(savedInstanceState);
61         setContentView(R.layout.info_list);
62
63         //buttons leading to the other activities in the application
64         quoteButton = (Button) findViewById(R.id.quoteButton);
65         quoteButton.setOnClickListener(new View.OnClickListener() {
66             @Override
67             public void onClick(View view) {
68                 Intent i = new Intent(view.getContext(), quoteScreen.class);
69                 i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
70                 overridePendingTransition(0,0); //0 for no animation
71                 startActivity(i);
72             }
73         });
74         eventButton = (Button) findViewById(R.id.eventButton);
75         eventButton.setOnClickListener(new View.OnClickListener() {
76             @Override
77             public void onClick(View view) {
78                 Intent i = new Intent(view.getContext(), eventListScreen.class);
79                 i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
80                 overridePendingTransition(0,0); //0 for no animation
81                 startActivity(i);
82             }
83         });
84         moreButton = (Button) findViewById(R.id.moreButton);
85         moreButton.setOnClickListener(new View.OnClickListener() {
86             @Override
87             public void onClick(View view) {
88                 Intent i = new Intent(view.getContext(), moreScreen.class);
89                 i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
90                 overridePendingTransition(0,0); //0 for no animation
91                 startActivity(i);
92             }
93         });
94     }
95 }
```

```

94
95     //initialize the listview
96     newsList = new ArrayList<HashMap<String, String>>();
97     ListView lv = getListView();
98
99     // Listview on item click listener, launch a seperate activity to show the expanded version of the news
100    lv.setOnItemClickListener(new OnItemClickListener() {
101
102        @Override
103        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
104            // transfer the listview details to a new class to show individual expanded details
105
106            String title = ((TextView) view.findViewById(R.id.rowTitle))
107                .getText().toString();
108            String newsType = ((TextView) view.findViewById(R.id.rowClass))
109                .getText().toString();
110            String body = ((TextView) view.findViewById(R.id.rowBody))
111                .getText().toString();
112
113            Intent i = new Intent(view.getContext(), singleNews.class);
114            String[] newsValues = new String[] {title, newsType, body};
115            i.putExtra("newsData", newsValues);           //bundle it into a newsData named package - easily transfer information together
116            startActivity(i);
117        }
118    });
119
120    //launch the JSON processing script to populate the listview
121    new GetNews().execute();
122}
123
124 //Async task class to read JSON file from device
125 private class GetNews extends AsyncTask<Void, Void, Void> {
126
127    @Override
128    protected void onPreExecute() {
129        super.onPreExecute();
130        /*launch a dialog box to keep main system process busy with GUI, while the background thread handles the file reading - also so
131        pDialog = new ProgressDialog(infoListScreen.this, ProgressDialog.THEME_HOLO_DARK);
132        pDialog.setMessage("Updating the Al-Akhbar...\\n\\nPlease make sure you have Internet connection");
133        pDialog.setCancelable(false);
134        pDialog.show();*/
135    }
136
137    //perform these operations in the background thread
138    @Override
139    protected Void doInBackground(Void... arg0) {
140        // Creating the JSON string for data
141        String jsonStr = null;
142
143        Log.d("Response: ", "> " + jsonStr);
144
145        //read the downloaded JSON file from online SQL server, and parse it into a string format
146        try {
147            File yourFile = new File(Environment.getExternalStorageDirectory(), "AlAkhbar/thisweek.json");
148            FileInputStream stream = new FileInputStream(yourFile);
149            try {
150                FileChannel fc = stream.getChannel();
151                MappedByteBuffer bb = fc.map(FileChannel.MapMode.READ_ONLY, 0, fc.size());
152
153                jsonStr = Charset.defaultCharset().decode(bb).toString();
154            }
155            finally {
156                stream.close();
157            }
158
159            //create a JSON object, to parse file
160            JSONObject jsonObj = new JSONObject(jsonStr);
161
162            // specify the table name (table name is an array, that contains all the element of the table) of the JSON file
163            news = jsonObj.getJSONArray(NEWSTHISWEEK);
164
165            // looping through all the elements in the table
166            for (int i = 0; i < news.length(); i++) {
167                JSONObject c = news.getJSONObject(i);
168
169                //parse the appropriate information from the JSON file
170                String title = c.getString(TITLE);
171                String body = c.getString(BODY);
172                String subtitle = c.getString(SUBTITLE);
173                String newsType = c.getString(NEWSTYPE);
174
175                //use if statement to filter what content we want to display in this activity
176                if (newsType.equals("i")) {
177                    newsType = "Information";
178
179                    // tmp hashmap for single contact
180                    newsItems = new HashMap<String, String>();
181
182                    // adding each child node to HashMap key => value
183                    newsItems.put(TITLE, title);
184                    newsItems.put(BODY, body);
185                    newsItems.put(SUBTITLE, subtitle);
186                    newsItems.put(NEWSTYPE, newsType);
187
188                    // adding news to news list
189                    newsList.add(newsItems);
190                }
191            }
192        }
193    }
194
195    @Override
196    protected void onPostExecute(Void result) {
197        // dismiss the dialog after getting all news
198        pDialog.dismiss();
199        // update UI from background thread
200        runOnUiThread(new Runnable() {
201            public void run() {
202                // update newsList
203                adapter.notifyDataSetChanged();
204            }
205        });
206    }
207
208    @Override
209    protected void onCancelled(Void result) {
210        // dismiss the dialog after canceling the download
211        pDialog.dismiss();
212    }
213
214}

```

```
191         }
192     } catch (Exception e) {
193         e.printStackTrace();
194     }
195
196     return null;
197 }
198
199 //after all of the above processes are done, launch the following
200 @Override
201 protected void onPostExecute(Void result) {
202     super.onPostExecute(result);
203     // Dismiss the progress dialog
204     /*if (pDialog.isShowing())
205      pDialog.dismiss();*/
206
207     //Updating parsed JSON data into ListView using a custom adapter
208     //All listviews are made of data and adapters. The adapter handles the design, theme, and source of the information.
209     /*To create a custom adapter, a customized row.xml file was coded to make a theme of the individual listview elements,
210     and the data is provided using the JSON list.*/
211    ListAdapter adapter = new SimpleAdapter(
212         infoListScreen.this, newsList,
213         R.layout.row, new String[] { TITLE, SUBTITLE,
214         NEWSTYPE,BODY }, new int[] { R.id.rowTitle,
215         R.id.rowSubtitle, R.id.rowClass, R.id.rowBody });
216
217     setListAdapter(adapter);
218 }
219 }
220 }
```

## moreScreen.java

```
1 package com.example.zain.alakhbar;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.os.AsyncTask;
6 import android.os.Bundle;
7 import android.os.Environment;
8 import android.text.Html;
9 import android.text.method.LinkMovementMethod;
10 import android.util.Log;
11 import android.view.View;
12 import android.widget.Button;
13 import android.widget.TextView;
14
15 import org.json.JSONArray;
16 import org.json.JSONObject;
17
18 import java.io.File;
19 import java.io.FileInputStream;
20 import java.nio.MappedByteBuffer;
21 import java.nio.channels.FileChannel;
22 import java.nio.charset.Charset;
23 import java.util.ArrayList;
24 import java.util.HashMap;
25
26 public class moreScreen extends Activity {
27
28     private Button infoButton = null;
29     private Button eventButton = null;
30     private Button quoteButton = null;
31
32     //initialize the quote screen - setup the layout
33     @Override
34     public void onCreate(Bundle savedInstanceState) {
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.more_screen);
37
38         //buttons leading to the other activities in the application
39         infoButton = (Button) findViewById(R.id.infoButton);
40         infoButton.setOnClickListener(new View.OnClickListener() {
41             @Override
```

```
42     public void onClick(View view) {
43         Intent i = new Intent(view.getContext(), infoListScreen.class);
44         i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
45         overridePendingTransition(0,0); //0 for no animation
46         startActivity(i);
47     }
48 });
49 eventButton = (Button) findViewById(R.id.eventButton);
50 eventButton.setOnClickListener(new View.OnClickListener() {
51     @Override
52     public void onClick(View view) {
53         Intent i = new Intent(view.getContext(), eventListScreen.class);
54         i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
55         overridePendingTransition(0,0); //0 for no animation
56         startActivity(i);
57     }
58 });
59 quoteButton = (Button) findViewById(R.id.quoteButton);
60 quoteButton.setOnClickListener(new View.OnClickListener() {
61     @Override
62     public void onClick(View view) {
63         Intent i = new Intent(view.getContext(), quoteScreen.class);
64         i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
65         overridePendingTransition(0,0); //0 for no animation
66         startActivity(i);
67     }
68 });
69
70 TextView body = (TextView) findViewById(R.id.moreBody);
71 body.setText(Html.fromHtml("<b>Ismaili Council for Canada (IIC)</b><br>" +
72                         "49 Wynford Drive<br>" +
73                         "Toronto, Ontario M3C 1K1<br>" +
74                         "CANADA<br>" +
75                         "Tel: +1 (416) 646-6965<br><br>" +
76                         "<b>Ismaili Council for Ontario</b><br>" +
77                         "149-151 Bartley Avenue<br>" +
78                         "Toronto, Ontario M4A 1C9<br>" +
79                         "CANADA<br>" +
80                         "Tel: +1-416-751-4001<br><br><br><br>" +
81                         "<i>Application developed by Zain Zulfiqar</i>"));
82 }
83 }
```

## quoteScreen.java

```
1 package com.example.zain.alakhbar;
2
3 import android.app.Activity;
4 import android.app.ListActivity;
5 import android.app.ProgressDialog;
6 import android.content.Context;
7 import android.content.Intent;
8 import android.net.ConnectivityManager;
9 import android.net.NetworkInfo;
10 import android.os.AsyncTask;
11 import android.os.Bundle;
12 import android.os.Environment;
13 import android.os.Handler;
14 import android.text.Html;
15 import android.text.method.LinkMovementMethod;
16 import android.util.Log;
17 import android.view.View;
18 import android.widget.AdapterView;
19 import android.widget.AdapterView.OnItemClickListener;
20 import android.widget.Button;
21 import android.widget.ListAdapter;
22 import android.widget.ListView;
23 import android.widget.SimpleAdapter;
24 import android.widget.TextView;
25 import android.widget.Toast;
26
27 import org.json.JSONArray;
28 import org.json.JSONObject;
29
30 import java.io.File;
31 import java.io.FileInputStream;
32 import java.nio.MappedByteBuffer;
33 import java.nio.channels.FileChannel;
34 import java.nio.charset.Charset;
35 import java.util.ArrayList;
36 import java.util.HashMap;
37 import java.util.Set;
38
39 public class quoteScreen extends Activity {
40
41     //private ProgressDialog pDialog;
42
43     // URL to get JSON
44     private static String url = "http://zainzulfiqar.com/thisweek.json";
45
46     // JSON Node names
47     private static final String BODY = "body";
48     private static final String NEWSTYPE = "class";
49     private static final String NEWSTHISWEEK = "newsThisWeek";
50
51     private Button infoButton = null;
52     private Button eventButton = null;
53     private Button moreButton = null;
54
55     // news JSONArray
56     JSONArray news = null;
57
58     // Hashmap for ListView
59     ArrayList<HashMap<String, String>> newsList;
60     HashMap<String, String> newsItems = new HashMap<String, String>();
61
62     //initialize the quote screen - setup the layout
63     @Override
64     public void onCreate(Bundle savedInstanceState) {
```

```

65     super.onCreate(savedInstanceState);
66     setContentView(R.layout.quote_screen);
67     newsList = new ArrayList<HashMap<String, String>>();
68
69     //buttons leading to the other activities in the application
70     infoButton = (Button) findViewById(R.id.infoButton);
71     infoButton.setOnClickListener(new View.OnClickListener() {
72         @Override
73         public void onClick(View view) {
74             Intent i = new Intent(view.getContext(), infoListScreen.class);
75             i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
76             overridePendingTransition(0,0); //0 for no animation
77             startActivity(i);
78         }
79     });
80     eventButton = (Button) findViewById(R.id.eventButton);
81     eventButton.setOnClickListener(new View.OnClickListener() {
82         @Override
83         public void onClick(View view) {
84             Intent i = new Intent(view.getContext(), eventListScreen.class);
85             i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
86             overridePendingTransition(0,0); //0 for no animation
87             startActivity(i);
88         }
89     });
90     moreButton = (Button) findViewById(R.id.moreButton);
91     moreButton.setOnClickListener(new View.OnClickListener() {
92         @Override
93         public void onClick(View view) {
94             Intent i = new Intent(view.getContext(), moreScreen.class);
95             i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
96             overridePendingTransition(0,0); //0 for no animation
97             startActivity(i);
98         }
99     });
100
101    //launch the JSON processing script to populate the listview
102    new GetNews().execute();
103}
104
105 //when back button pressed on this activity
106 @Override
107 public void onBackPressed() {
108     //leave blank so that you can't go back to the home screen (going back causes a task execution error)
109 }
110
111 //Async task class to read JSON file from device
112 private class GetNews extends AsyncTask<Void, Void, Void> {
113
114     @Override
115     protected void onPreExecute() {
116         super.onPreExecute();
117         /*//launch a dialog box to keep main system process busy with GUI, while the background thread handles the file reading - also so that user knows what is going on
118         pDialog = new ProgressDialog(quoteScreen.this, ProgressDialog.THEME_HOLO_DARK);
119         pDialog.setMessage("Updating the Al-Akhbar... \n\nPlease make sure you have Internet connection");
120         pDialog.setCancelable(false);
121         pDialog.show();*/
122     }
123
124     //perform these operations in the background thread
125     @Override
126     protected Void doInBackground(Void... arg0) {
127         // Creating the JSON string for data
128         String jsonStr = null;
129
130         //display log responses so it is easy to debug application
131         Log.d("Response: ", "> " + jsonStr);
132

```

```

133     //read the downloaded JSON file from online SQL server, and parse it into a string format
134     try {
135         File yourFile = new File(Environment.getExternalStorageDirectory(), "/AlAkbar/thisweek.json");
136         FileInputStream stream = new FileInputStream(yourFile);
137         try {
138             FileChannel fc = stream.getChannel();
139             MappedByteBuffer bb = fc.map(FileChannel.MapMode.READ_ONLY, 0, fc.size());
140
141             jsonStr = Charset.defaultCharset().decode(bb).toString();
142         }
143         finally {
144             stream.close();
145         }
146
147         //create a JSON object, to parse file
148         JSONObject jsonObj = new JSONObject(jsonStr);
149
150         // specify the table name (table name is an array, that contains all the element of the table) of the JSON file
151         news = jsonObj.getJSONArray(NEWSTHISWEEK);
152
153         // looping through all the elements in the table
154         for (int i = 0; i < news.length(); i++) {
155             JSONObject c = news.getJSONObject(i);
156
157             //parse the appropriate information from the JSON file
158             String body = c.getString(BODY);
159             String newsType = c.getString(NEWSTYPE);
160
161             //use if statement to filter what content we want to display in this activity
162             if (newsType.equals("q")) {
163                 // tmp hashmap for single contact
164                 newsItems = new HashMap<String, String>();
165
166                 // adding each child node to HashMap key => value
167                 newsItems.put(BODY, body);
168
169                 // adding news to news list
170                 newsList.add(newsItems);
171             }
172         }
173     } catch (Exception e) {
174         e.printStackTrace();
175     }
176
177     return null;
178 }
179
180 //after all of the above processes are done, launch the following
181 @Override
182 protected void onPostExecute(Void result) {
183     super.onPostExecute(result);
184     // Dismiss the progress dialog
185     /*if (pDialog.isShowing())
186     pDialog.dismiss();*/
187
188     //Updating parsed JSON data into ListView
189     String quoteBodyValue = newsItems.get(BODY);
190     TextView body = (TextView) findViewById(R.id.quoteBody);
191
192     /*set an HTML element on Java string so that HTML tags can be recognized
193     this allows for the data to be uniquely customized/styled by typing in HTML tags in the database itself*/
194     body.setText(Html.fromHtml(quoteBodyValue));
195     body.setMovementMethod(LinkMovementMethod.getInstance());
196 }
197 }
198 }
```

## singleNews.java

```
1 package com.example.zain.alakhbar;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.text.Html;
7 import android.text.method.LinkMovementMethod;
8 import android.view.Menu;
9 import android.view.MenuItem;
10 import android.view.View;
11 import android.os.Build;
12 import android.widget.Button;
13 import android.widget.ListView;
14 import android.widget.TextView;
15 import android.app.ActionBar;
16 import android.app.Fragment;
17 import android.view.LayoutInflater;
18 import android.view.ViewGroup;
19 import android.widget.Toast;
20
21 import java.util.ArrayList;
22
23
24 public class singleNews extends Activity {
25
26     private Button backButton = null;
27
28     @Override
29     protected void onCreate(Bundle savedInstanceState) {
30         super.onCreate(savedInstanceState);
31         setContentView(R.layout.activity_news_expanded);
32
33         backButton = (Button) findViewById(R.id.backButton);
34
35         backButton.setOnClickListener(new View.OnClickListener() {
36             @Override
37             public void onClick(View view) {
38                 finish(); //use finish to end this activity, and go back to the previous one. This continues off the state of the previous activity
39             }
40         });
41
42     //get data
43     Intent i = getIntent();
44     String[] newsExpandedValues = i.getStringArrayExtra("newsData");
45
46     //update
47     TextView title = (TextView) findViewById(R.id.singleTitle);
48     TextView newsType = (TextView) findViewById(R.id.singleClass);
49     TextView body = (TextView) findViewById(R.id.singleBody);
50     title.setText(newsExpandedValues[0]);
51     newsType.setText(newsExpandedValues[1]);
52
53     // body setup
54     body.setText(Html.fromHtml(newsExpandedValues[2]));
55     body.setMovementMethod(LinkMovementMethod.getInstance());
56
57 }
```

## Screen snapshots of the XML code

### AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.zain.alakhbar" >
4          <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
5          <uses-permission android:name="android.permission.INTERNET" />
6          <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
7          <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
8          <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
9          <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
10         <application
11             android:allowBackup="true"
12             android:icon="@drawable/ic_launcher"
13             android:label="@string/app_name"
14             android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
15                 <activity
16                     android:name=".home"
17                     android:label="@string/app_name"
18                     android:screenOrientation="portrait">
19                         <intent-filter>
20                             <action android:name="android.intent.action.MAIN" />
21
22                             <category android:name="android.intent.category.LAUNCHER" />
23                         </intent-filter>
24                     </activity>
25                     <activity
26                         android:name=".quoteScreen"
27                         android:label="@string/title_activity_list_of_news"
28                         android:screenOrientation="portrait">
29                     </activity>
30                     <activity
31                         android:name=".infoListScreen"
32                         android:label="@string/title_activity_list_of_news"
33                         android:screenOrientation="portrait">
34                     </activity>
35                     <activity
36                         android:name=".eventListScreen"
37                         android:label="@string/title_activity_list_of_news"
38                         android:screenOrientation="portrait">
39                     </activity>
40                     <activity
41                         android:name=".moreScreen"
42                         android:label="@string/title_activity_list_of_news"
43                         android:screenOrientation="portrait">
44                     </activity>
45                     <activity
46                         android:name=".singleNews"
47                         android:label="@string/title_activity_list_of_news"
48                         android:screenOrientation="portrait">
49                     </activity>
50                 </application>
51             </manifest>
```

## activity\_home.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingLeft="@dimen/activity_horizontal_margin"
6     android:paddingRight="@dimen/activity_horizontal_margin"
7     android:paddingTop="@dimen/activity_vertical_margin"
8     android:paddingBottom="@dimen/activity_vertical_margin"
9     android:background="#9C1B07"
10    tools:context=".home">
11
12    <TextView
13        android:text="@string/menu_title1"
14        android:textColor="#FFFFFF"
15        android:textSize="35sp"
16        android:layout_width="wrap_content"
17        android:layout_height="wrap_content"
18        android:layout_centerHorizontal="true"
19        android:layout_centerVertical="true"
20        android:layout_above="@+id/title2"/>
21
22    <TextView
23        android:id="@+id/title2"
24        android:text="@string/menu_title2"
25        android:textColor="#FFFFFF"
26        android:textSize="66sp"
27        android:layout_width="wrap_content"
28        android:layout_height="wrap_content"
29        android:layout_centerHorizontal="true"
30        android:layout_centerVertical="true"/>
31
32    <LinearLayout
33        android:layout_width="match_parent"
34        android:layout_height="match_parent"
35        android:weightSum="2"
36        android:orientation="vertical">
37
38        <TextView
39            android:text=" "
40            android:layout_width="match_parent"
41            android:layout_height="match_parent"
42            android:layout_weight="1"/>
43
44        <LinearLayout
45            android:layout_width="fill_parent"
46            android:layout_height="fill_parent"
47            android:layout_weight="1"
48            android:gravity="center_horizontal|center_vertical">
49
50            <Button
51                android:id="@+id/menuButton"
52                android:layout_gravity="center"
53                android:layout_width="wrap_content"
54                android:layout_height="50dp"
55                android:text="Enter"
56                android:textSize="22sp"
```

```
57         android:gravity="center_horizontal|center_vertical"
58         android:textColor="#FFFFFF"
59         android:background="@drawable/home_button_define"/>
60
61     </LinearLayout>
62
63 </LinearLayout>
64
65 <TextView
66     android:id="@+id/testMode"
67     android:text="Version 1.00"
68     android:textColor="#FFFFFF"
69     android:textSize="17sp"
70     android:layout_width="wrap_content"
71     android:layout_height="wrap_content"
72     android:layout_centerHorizontal="true"
73     android:layout_centerVertical="true"
74     android:layout_alignParentBottom="true"
75     android:textStyle="italic"
76     android:layout_marginBottom="20dp"/>
77
78 <TextView
79     android:text="Z.Z."
80     android:textColor="#FFFFFF"
81     android:textSize="10sp"
82     android:layout_width="wrap_content"
83     android:layout_height="wrap_content"
84     android:layout_centerHorizontal="true"
85     android:layout_centerVertical="true"
86     android:layout_alignParentBottom="true"
87     android:textStyle="italic"/>
88
89 </RelativeLayout>
```

## activity\_news\_expanded.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:weightSum="20">

        <Button
            android:id="@+id/backButton"
            android:layout_weight="17"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text=" &#60; "
            android:textColor="@color/white"
            android:textSize="30sp"
            android:background="@drawable/home_button_define"/>

        <TextView android:id="@+id/headerNews"
            android:layout_weight="3"
            android:text="Back to News Feed"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"
            android:textColor="@color/white"
            android:background="@color/rowTitle"
            android:gravity="left"
            android:textSize="20sp"
            android:paddingTop="20dp"
            android:paddingBottom="20dp"/>
    </LinearLayout>

    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@color/rowback">
        <LinearLayout
            android:orientation="vertical" android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/singleTitle"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Title Here..."
                android:textSize="25dp"
                android:textColor="@color/rowTitle"
                android:gravity="center"
                android:layout_marginEnd="10dp"
                android:layout_marginStart="10dp"
                android:paddingTop="15dp"
                android:layout_marginBottom="7dp"/>
        </LinearLayout>
    </ScrollView>

```

```
57
58     <TextView
59         android:id="@+id/singleClass"
60         android:layout_width="match_parent"
61         android:layout_height="wrap_content"
62         android:text="Class Here..."
63         android:textSize="15dp"
64         android:textStyle="bold"
65         android:textColor="@color/rowSubtitle"
66         android:gravity="center"
67         android:layout_marginBottom="10dp"/>
68
69     <View
70         android:id="@+id/colored_bar"
71         android:layout_width="fill_parent"
72         android:layout_height="3dp"
73         android:background="@color/red"
74         android:layout_marginLeft="17dp"
75         android:layout_marginRight="17dp"/>
76
77     <TextView
78         android:id="@+id/singleBody"
79         android:layout_width="match_parent"
80         android:layout_height="wrap_content"
81         android:text="Body Here..."
82         android:textSize="15dp"
83         android:textColor="@color/rowSubtitle"
84         android:layout_marginTop="15dp"
85         android:layout_marginBottom="20dp"
86         android:layout_marginLeft="15dp"
87         android:layout_marginRight="15dp"
88         android:textColorHighlight="@color/rowTitle"
89         android:textColorHint="@color/white"/>
90
91     </LinearLayout>
92 </ScrollView>
93 </LinearLayout>
```

## event\_list.xml

```
1 <LinearLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="wrap_content"
4     android:layout_height="fill_parent"
5     android:orientation="vertical"
6     android:id="@+id/r1Layout"
7     android:background="@color/rowback"
8     android:weightSum="20">
9
10    <TextView android:id="@+id/headerNews"
11        android:text="Weekly Updates"
12        android:layout_height="wrap_content"
13        android:layout_width="fill_parent"
14        android:textColor="@color/white"
15        android:background="@color/rowTitle"
16        android:gravity="center"
17        android:textSize="30sp"
18        android:padding="20dp"
19        android:layout_weight="0"/>
20
21    <ListView
22        android:id="@+id/list"
23        android:layout_width="fill_parent"
24        android:layout_height="fill_parent"
25        android:layout_below="@+id/headerNews"
26        android:smoothScrollbar="true"
27        android:choiceMode="singleChoice"
28        android:layout_weight="4">
29    </ListView>
30
31    <LinearLayout
32        android:layout_width="match_parent"
33        android:layout_height="match_parent"
34        android:layout_below="@+id/list"
35        android:layout_marginBottom="-6dp"
36        android:orientation="horizontal"
37        android:layout_weight="16"
38        android:weightSum="4">
39
40        <Button
41            android:id="@+id/quoteButton"
42            android:text="Quote"
43            android:textColor="@color/white"
44            android:textSize="15sp"
45            android:layout_width="fill_parent"
46            android:layout_height="fill_parent"
47            android:layout_weight="1"
48            android:background="@color/red"/>
49
50        <Button
51            android:id="@+id/infoButton"
52            android:text="Information"
53            android:textColor="@color/white"
54            android:textSize="15sp"
55            android:layout_width="fill_parent"
56            android:layout_height="fill_parent"
```

```
57         android:layout_weight="1"
58         android:background="@color/red"/>
59
60     <Button
61         android:id="@+id/eventButton"
62         android:text="Events"
63         android:textColor="@color/white"
64         android:textSize="15sp"
65         android:layout_width="fill_parent"
66         android:layout_height="fill_parent"
67         android:layout_weight="1"
68         android:background="@color.clicked"/>
69
70     <Button
71         android:id="@+id/moreButton"
72         android:text="More"
73         android:textColor="@color/white"
74         android:textSize="15sp"
75         android:layout_width="fill_parent"
76         android:layout_height="fill_parent"
77         android:layout_weight="1"
78         android:background="@color/red"/>
79
80     </LinearLayout>
81
82 </LinearLayout>
```

## info\_list.xml

```
1 <LinearLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="wrap_content"
4     android:layout_height="fill_parent"
5     android:orientation="vertical"
6     android:id="@+id/r1Layout"
7     android:background="@color/rowback"
8     android:weightSum="20">
9
10    <TextView android:id="@+id/headerNews"
11        android:text="Weekly Updates"
12        android:layout_height="wrap_content"
13        android:layout_width="fill_parent"
14        android:textColor="@color/white"
15        android:background="@color/rowTitle"
16        android:gravity="center"
17        android:textSize="30sp"
18        android:padding="20dp"
19        android:layout_weight="0"/>
20
21    <ListView
22        android:id="@+id/list"
23        android:layout_width="fill_parent"
24        android:layout_height="fill_parent"
25        android:layout_below="@+id/headerNews"
26        android:smoothScrollbar="true"
27        android:choiceMode="singleChoice"
28        android:layout_weight="4">
29    </ListView>
30
31    <LinearLayout
32        android:layout_width="match_parent"
33        android:layout_height="match_parent"
34        android:layout_below="@+id/list"
35        android:layout_marginBottom="-6dp"
36        android:orientation="horizontal"
37        android:layout_weight="16"
38        android:weightSum="4">
39
40        <Button
41            android:id="@+id/quoteButton"
42            android:text="Quote"
43            android:textColor="@color/white"
44            android:textSize="15sp"
45            android:layout_width="fill_parent"
46            android:layout_height="fill_parent"
47            android:layout_weight="1"
48            android:background="@color/red"/>
49
50        <Button
51            android:id="@+id/infoButton"
52            android:text="Information"
53            android:textColor="@color/white"
54            android:textSize="15sp"
55            android:layout_width="fill_parent"
56            android:layout_height="fill_parent"
```

```
57         android:layout_weight="1"
58         android:background="@color.clicked"/>
59
60     <Button
61         android:id="@+id/eventButton"
62         android:text="Events"
63         android:textColor="@color/white"
64         android:textSize="15sp"
65         android:layout_width="fill_parent"
66         android:layout_height="fill_parent"
67         android:layout_weight="1"
68         android:background="@color/red"/>
69
70     <Button
71         android:id="@+id/moreButton"
72         android:text="More"
73         android:textColor="@color/white"
74         android:textSize="15sp"
75         android:layout_width="fill_parent"
76         android:layout_height="fill_parent"
77         android:layout_weight="1"
78         android:background="@color/red"/>
79
80     </LinearLayout>
81
82 </LinearLayout>
```

## more\_screen.xml

```
1 <LinearLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:orientation="vertical"
6     android:background="@color/rowback"
7     android:weightSum="20">
8
9     <TextView android:id="@+id/headerNews"
10        android:text="Weekly Updates"
11        android:layout_height="wrap_content"
12        android:layout_width="fill_parent"
13        android:textColor="@color/white"
14        android:background="@color/rowTitle"
15        android:gravity="center"
16        android:textSize="30sp"
17        android:padding="20dp"
18        android:layout_weight="0"/>
19
20     <LinearLayout
21         android:layout_width="fill_parent"
22         android:layout_height="fill_parent"
23         android:orientation="vertical"
24         android:layout_weight="4">
25
26         <TextView
27             android:id="@+id/moreTitle"
28             android:layout_width="fill_parent"
29             android:layout_height="wrap_content"
30             android:text="More Details"
31             android:gravity="center"
32             android:textColor="@color/rowTitle"
33             android:textSize="30sp"
34             android:paddingTop="20dp"
35             android:paddingBottom="20dp"/>
36
37         <TextView
38             android:id="@+id/moreBody"
39             android:layout_width="fill_parent"
40             android:layout_height="wrap_content"
41             android:text="details here..."
42             android:textColor="@color/rowSubtitle"
43             android:textSize="19sp"
44             android:gravity="center"
45             android:layout_marginBottom="20dp"
46             android:layout_marginLeft="20dp"
47             android:layout_marginRight="20dp"/>
48
49     </LinearLayout>
50
51     <LinearLayout
52         android:layout_width="match_parent"
53         android:layout_height="match_parent"
54         android:layout_below="@android:id/list"
55         android:layout_marginBottom="-6dp"
56         android:orientation="horizontal">
```

```
57     android:layout_weight="16"
58     android:weightSum="4">
59
60     <Button
61         android:id="@+id/quoteButton"
62         android:text="Quote"
63         android:textColor="@color/white"
64         android:textSize="15sp"
65         android:layout_width="fill_parent"
66         android:layout_height="fill_parent"
67         android:layout_weight="1"
68         android:background="@color/red"/>
69
70     <Button
71         android:id="@+id/infoButton"
72         android:text="Information"
73         android:textColor="@color/white"
74         android:textSize="15sp"
75         android:layout_width="fill_parent"
76         android:layout_height="fill_parent"
77         android:layout_weight="1"
78         android:background="@color/red"/>
79
80     <Button
81         android:id="@+id/eventButton"
82         android:text="Events"
83         android:textColor="@color/white"
84         android:textSize="15sp"
85         android:layout_width="fill_parent"
86         android:layout_height="fill_parent"
87         android:layout_weight="1"
88         android:background="@color/red"/>
89
90     <Button
91         android:id="@+id/moreButton"
92         android:text="More"
93         android:textColor="@color/white"
94         android:textSize="15sp"
95         android:layout_width="fill_parent"
96         android:layout_height="fill_parent"
97         android:layout_weight="1"
98         android:background="@color.clicked"/>
99
100    </LinearLayout>
101
102 </LinearLayout>
```

## quote\_screen.xml

```
1 <LinearLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:orientation="vertical"
6     android:background="@color/rowback"
7     android:weightSum="20">
8
9     <TextView android:id="@+id/headerNews"
10        android:text="Weekly Updates"
11        android:layout_height="wrap_content"
12        android:layout_width="fill_parent"
13        android:textColor="@color/white"
14        android:background="@color/rowTitle"
15        android:gravity="center"
16        android:textSize="30sp"
17        android:padding="20dp"
18        android:layout_weight="0"/>
19
20     <LinearLayout
21         android:layout_width="fill_parent"
22         android:layout_height="fill_parent"
23         android:orientation="vertical"
24         android:layout_weight="4">
25
26         <TextView
27             android:id="@+id/quoteTitle"
28             android:layout_width="fill_parent"
29             android:layout_height="wrap_content"
30             android:text="Quote of the Week"
31             android:gravity="center"
32             android:textColor="@color/rowTitle"
33             android:textSize="30sp"
34             android:paddingTop="20dp"
35             android:paddingBottom="20dp"/>
36
37         <TextView
38             android:id="@+id/quoteBody"
39             android:layout_width="fill_parent"
40             android:layout_height="wrap_content"
41             android:text="Loading quote..."
42             android:textColor="@color/rowSubtitle"
43             android:textSize="19sp"
44             android:gravity="center"
45             android:layout_marginBottom="20dp"
46             android:layout_marginLeft="20dp"
47             android:layout_marginRight="20dp"/>
48
49     </LinearLayout>
50
51     <LinearLayout
52         android:layout_width="match_parent"
53         android:layout_height="match_parent"
54         android:layout_below="@android:id/list"
55         android:layout_marginBottom="-6dp"
56         android:orientation="horizontal">
```

```
57     android:layout_weight="16"
58     android:weightSum="4">>
59
60     <Button
61         android:id="@+id/quoteButton"
62         android:text="Quote"
63         android:textColor="@color/white"
64         android:textSize="15sp"
65         android:layout_width="fill_parent"
66         android:layout_height="fill_parent"
67         android:layout_weight="1"
68         android:background="@color.clicked"/>
69
70     <Button
71         android:id="@+id/infoButton"
72         android:text="Information"
73         android:textColor="@color/white"
74         android:textSize="15sp"
75         android:layout_width="fill_parent"
76         android:layout_height="fill_parent"
77         android:layout_weight="1"
78         android:background="@color/red"/>
79
80     <Button
81         android:id="@+id/eventButton"
82         android:text="Events"
83         android:textColor="@color/white"
84         android:textSize="15sp"
85         android:layout_width="fill_parent"
86         android:layout_height="fill_parent"
87         android:layout_weight="1"
88         android:background="@color/red"/>
89
90     <Button
91         android:id="@+id/moreButton"
92         android:text="More"
93         android:textColor="@color/white"
94         android:textSize="15sp"
95         android:layout_width="fill_parent"
96         android:layout_height="fill_parent"
97         android:layout_weight="1"
98         android:background="@color/red"/>
99
100    </LinearLayout>
101
102 </LinearLayout>
```

## row.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="wrap_content"
5     android:background="@drawable/row_select">
6
7     <TextView
8         android:id="@+id/rowTitle"
9         android:layout_width="match_parent"
10        android:layout_height="wrap_content"
11        android:text="Title text here..."
12        android:textColor="@color/rowTitle"
13        android:textSize="21sp"
14        android:paddingLeft="20dp"
15        android:paddingStart="20dp"
16        android:paddingRight="20dp"
17        android:paddingEnd="20dp"
18        android:paddingTop="6dp"
19        android:paddingBottom="8dp"/>
20
21     <TextView
22         android:id="@+id/rowSubtitle"
23         android:layout_below="@+id/rowTitle"
24         android:layout_width="wrap_content"
25         android:layout_height="wrap_content"
26         android:text="Subtitle text here..."
27         android:textColor="@color/rowSubtitle"
28         android:textSize="15sp"
29         android:paddingLeft="20dp"
30         android:paddingStart="20dp"
31         android:paddingBottom="9dp"
32         android:paddingRight="40dp"
33         android:paddingEnd="40dp"
34         android:maxLines="1"
35         android:ellipsize="end"/>
36
37     <TextView
38         android:id="@+id/rowClass"
39         android:layout_below="@+id/rowSubtitle"
40         android:layout_width="wrap_content"
41         android:layout_height="wrap_content"
42         android:text="Quote, event, info, ad?"
43         android:textColor="@color/rowSubtitle"
44         android:textSize="12sp"
45         android:textStyle="bold"
46         android:paddingLeft="20dp"
47         android:paddingStart="20dp"
48         android:paddingBottom="7dp"
49         android:paddingRight="40dp"
50         android:paddingEnd="40dp"/>
51
52     <TextView
53         android:id="@+id/rowBody"
54         android:layout_below="@+id/rowClass"
55         android:layout_width="wrap_content"
56         android:layout_height="wrap_content"
```

```
57     android:text="The body here is hidden..."  
58     android:visibility="gone"  
59     android:textColor="@color/rowSubtitle"  
60     android:textSize="12sp"  
61     android:textStyle="bold"  
62     android:paddingLeft="20dp"  
63     android:paddingStart="20dp"  
64     android:paddingBottom="15dp"  
65     android:paddingRight="40dp"  
66     android:paddingEnd="40dp"/>/  
67  
68 <ImageView  
69     android:padding="15dp"  
70     android:layout_width="41dp"  
71     android:layout_height="50dp"  
72     android:adjustViewBounds="true"  
73     android:scaleType="fitXY"  
74     android:src="@drawable/right_arrow"  
75     android:layout_alignParentEnd="true"  
76     android:layout_alignParentRight="true"  
77     android:layout_centerInParent="true"  
78     />  
79 </RelativeLayout>
```